Raccolta Esercizi Esame

4 settembre 2017

Indice

Capitolo 1

ANNO 2016-2017

1.1 § Compitino di ASD del 20-02-17

Esercizio 1. Compito A
Sia A un vettore di interi *ordinato in modo crescente*, e sia k un intero. Si consideri il seguente codice ricorsivo.

```
PROC(A, k)
 1: if length[A] = 0 then
 2:    return NO
 3: else
 4:    m ← ⌈(length[A])/2⌉;
 5:    if A[m] = k then
 6:       return YES
 7:    else if A[m] > k then
 8:       B ← newArray(m − 1){B è un nuovo array di lunghezza m − 1}
 9:       for i ← 1 to m − 1 do
10:          B[i] ← A[i]
11:       end for
12:       return PROC(B, k)
13:    else
14:       B ← newArray(length[A] − m){B è un nuovo array di lunghezza length[A] − m}
15:       for i ← 1 to length[A] − m do
16:          B[i] ← A[i + m]
17:       end for
18:       return PROC(B, k)
19:    end if
20: end if
```

a. Determinare e risolvere l'equazione ricorsiva di complessità di PROC, evidenziando i passaggi eseguiti.

b. In quali casi PROC(A, k) restituisce "YES"? Motivare opportunamente la risposta.

Esercizio 2. Compito A
Siano dati in input k vettori $A_1, \ldots, A_k$ di numeri naturali, ognuno *ordinato in modo crescente*. Sia n la quantità di elementi presenti *complessivamente* nei vettori, ovvero $n = \sum_{i=1}^{k} length[A_i]$. Si consideri il problema di produrre in output il vettore *ordinato in modo crescente* B, unione di $A_1, \ldots, A_k$.

- Si proponga tramite pseudo-codice una procedura efficiente per risolvere il problema proposto nel caso in cui k sia costante rispetto ad n. Si determini la complessità e dimostri la correttezza della procedura proposta.

- Si proponga lo pseudo-codice di una procedura per risolvere il problema proposto nel caso generico avente complessità $O(n \log k)$. Si determini la complessità e dimostri la correttezza della procedura proposta.

Esercizio 3. Compito A
Si risolva la seguente equazione ricorsiva di complessità, mostrando i passaggi eseguiti.

$$T(n) = \begin{cases} \Theta(1) & \text{se } n = 1 \\ 3T(\frac{n}{4}) + \Theta(n\sqrt{n}) & \text{se } n \geq 2 \end{cases}$$

Esercizio 4. Compito A
Si rappresenti un albero binario di ricerca la cui visita in *post-order* ha come risultato 4, 9, 6, 14, 17, 25, 15, 10.
Si effettuino poi le seguenti operazioni *nell'ordine dato* e *disegnando l'albero risultante dopo ogni operazione*.

- cancellazione: 15;
- inserimenti: 26, 5;
- cancellazioni: 9, 6, 17.

Esercizio 5. Compito A
Si argomentino le seguenti affermazioni relative all'algoritmo di ordinamento *heapsort*.

1. Heapsort su un vettore A di n elementi ha complessità $O(n^2)$. Vero o falso?
2. Heapsort su un vettore A di n elementi ha complessità $\Omega(n^2)$. Vero o falso?
3. Qual è la complessità di Heapsort nel caso peggiore su un vettore A di n elementi?

Esercizio 6. Compito A
I due informatici russi Nikolaj Gigantov e Andrej Vielskij devono memorizzare alcune date $g/m/a$ nell'intervallo 01/01/1900–31/12/1999 (cioè $g \in [1, 31]$, $m \in [1, 12]$, e $a \in [1900, 1999]$) in una tabella di hash $H[0, \cdots, \ell - 1]$ di dimensione ℓ, nella quale le collisioni vengono gestite mediante *chaining*. Si assuma per semplicità che tutti i mesi abbiano 31 giorni.
Quali delle seguenti funzioni di hash sono adatte allo scopo? Perché?

1. $h(g, m, a) = \Big((g-1) + (m-1) + (a-1900)\Big) \mod \ell$
2. $h(g, m, a) = \Big((g-1) \cdot 100 \cdot 12 \cdot 31 + (m-1) \cdot 100 \cdot 12 \cdot 31 + (a-1900) \cdot 100 \cdot 12 \cdot 31\Big) \mod \ell$
3. $h(g, m, a) = \Big((g-1) \cdot 100 \cdot 12 + (m-1) \cdot 100 + (a-1900)\Big) \mod \ell$

Esercizio 7. Compito A
Sia dato un BST *completo* T con n nodi contenenti chiavi naturali senza ripetizioni. Sia inoltre dato un intervallo di naturali $[a, b]$ con $a \leq b$, e si assuma $b - a$ *costante rispetto ad* n.
Si proponga tramite pseudo-codice una procedura *efficiente* che stampi tutte e sole le chiavi di T appartenenti all'intervallo $[a, b]$.
Qual è la complessità della procedura proposta?

Esercizio 1. Compito B
Si risolva la seguente equazione ricorsiva di complessità, mostrando i passaggi eseguiti.

$$T(n) = \begin{cases} \Theta(1) & \text{se } n = 1 \\ 3T(\frac{n}{4}) + \Theta(n^{3/2}) & \text{se } n \geq 2 \end{cases}$$

Esercizio 2. Compito B
Siano dati in input k vettori $A_1, \ldots, A_k$ di numeri naturali, ognuno *ordinato in modo decrescente*. Sia n la quantità di elementi presenti *complessivamente* nei vettori, ovvero $n = \sum_{i=1}^{k} length[A_i]$. Si consideri il problema di produrre in output il vettore *ordinato in modo decrescente* B, unione di $A_1, \ldots, A_k$.

- Si proponga tramite pseudo-codice una procedura efficiente per risolvere il problema proposto nel caso in cui k sia costante rispetto ad n. Si determini la complessità e dimostri la correttezza della procedura proposta.
- Si proponga lo pseudo-codice di una procedura per risolvere il problema proposto nel caso generico avente complessità $O(n \log k)$. Si determini la complessità e dimostri la correttezza della procedura proposta.

Esercizio 3. Compito B
Sia A un vettore di interi *ordinato in modo crescente*, e sia k un intero. Si consideri il seguente codice ricorsivo.

```
FUNZ(A, k)
 1: if length[A] = 0 then
 2:    return NO
 3: else
 4:    m ← ⌈length[A]/2⌉;
 5:    if A[m] = k then
 6:       return YES
 7:    else if A[m] < k then
 8:       B ← newArray(length[A] − m){B è un nuovo array di lunghezza length[A] − m}
 9:       for i ← 1 to length[A] − m do
10:          B[i] ← A[i + m]
11:       end for
12:       return FUNZ(B, k)
13:    else
14:       B ← newArray(m − 1){B è un nuovo array di lunghezza m − 1}
15:       for i ← 1 to m − 1 do
16:          B[i] ← A[i]
17:       end for
18:       return FUNZ(B, k)
19:    end if
20: end if
```

a. Determinare e risolvere l'equazione ricorsiva di complessità di FUNZ, evidenziando i passaggi eseguiti.

b. In quali casi FUNZ(A, k) restituisce "YES"? Motivare opportunamente la risposta.

Esercizio 4. Compito B
Si argomentino le seguenti affermazioni relative all'algoritmo di ordinamento *heapsort*.

1. Heapsort su un vettore A di n elementi ha complessità $O(n^2)$. Vero o falso?
2. Heapsort su un vettore A di n elementi ha complessità $\Omega(n^2)$. Vero o falso?
3. Qual è la complessità di Heapsort nel caso peggiore su un vettore A di n elementi?

Esercizio 5. Compito B
Si rappresenti un albero binario di ricerca la cui visita in *post-order* ha come risultato 4, 9, 6, 14, 18, 20, 15, 10.
Si effettuino poi le seguenti operazioni *nell'ordine dato* e *disegnando l'albero risultante dopo ogni operazione*.

- cancellazione: 15;
- inserimenti: 21, 5;
- cancellazioni: 9, 6, 18.

Esercizio 6. Compito B
Sia dato un BST *completo* T con n nodi contenenti chiavi naturali senza ripetizioni. Sia inoltre dato un intervallo di naturali $[a, b]$ con $a \leq b$, e si assuma $b - a$ *costante rispetto ad* n.
Si proponga tramite pseudo-codice una procedura *efficiente* che stampi tutte e sole le chiavi di T appartenenti all'intervallo $[a, b]$. Qual è la complessità della procedura proposta?

Esercizio 7. Compito B
I due informatici russi Nikolaj Gigantov e Andrej Vielskij devono memorizzare alcune date $g/m/a$ nell'intervallo 01/01/1900–31/12/1999 (cioè $g \in [1, 31]$, $m \in [1, 12]$, e $a \in [1900, 1999]$) in una tabella di hash $H[0, \cdots, \ell - 1]$ di dimensione ℓ, nella quale le collisioni vengono gestite mediante *chaining*. Si assuma per semplicità che tutti i mesi abbiano 31 giorni.
Quali delle seguenti funzioni di hash sono adatte allo scopo? Perché?

1. $h(g, m, a) = \Big((g - 1) + (m - 1) \cdot 31 + (a - 1900) \cdot 31 \cdot 12\Big) \mod \ell$
2. $h(g, m, a) = \Big((g - 1) + (m - 1) + (a - 1900)\Big) \mod \ell$
3. $h(g, m, a) = \Big((g - 1) \cdot 100 \cdot 12 \cdot 31 + (m - 1) \cdot 100 \cdot 12 \cdot 31 + (a - 1900) \cdot 100 \cdot 12 \cdot 31\Big) \mod \ell$

1.2 § Compitino di ASD del 23-06-17

Esercizio 1.
Sia T un red-black tree, con radice nera, in cui esiste un percorso dalla radice ad una foglia costituito da tutti nodi neri.

a. Scrivere lo pseudocodice di una procedura efficiente per determinare la **lunghezza** minima di un cammino radice-foglia in T. Dimostrare la correttezza e valutare la complessità della procedura proposta.

b. Utilizzare il risultato restituito dalla procedura al punto a. per fornire l'intervallo di valori in cui si trova l'altezza di T. Quale ipotesi su T garantirebbe la conoscenza dell'altezza di T?

Esercizio 2.
Sia T un B-tree di grado $t \geq 2$, contenente n chiavi distinte. Dato un nodo x di T ed un indice $1 \leq i < n[x]$ (si noti che $i \neq n[x]$), si consideri il problema di determinare (se esiste) la chiave successore dell'i-esima chiave di x.

a. Scrivere lo pseudocodice di una procedura, SUCCESSOR(T, x, i), per risolvere il problema proposto.

b. Si dimostri la correttezza della procedura e si calcoli la complessità in funzione di t ed n.

Esercizio 3.
Si consideri il problema di gestire una collezione di insiemi disgiunti, utilizzando la rappresentazione ad alberi e le euristiche di *union-by-rank* e *path-compression*. Si assuma che, qualora vengano uniti due alberi aventi lo stesso rango, l'albero rappresentato dall'elemento minore venga fatto puntare all'albero rappresentato dall'elemento maggiore.

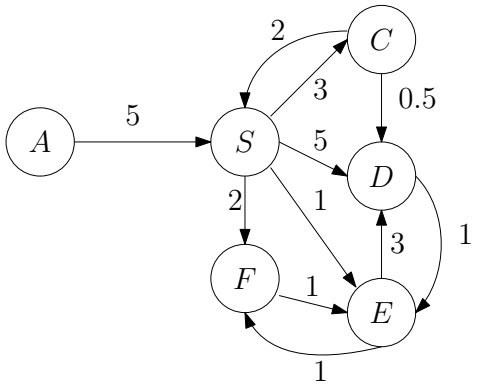
Si illustri, dopo ogni operazione, le collezioni che si ottengono a partire dai 10 insiemi $\{1\}, \{2\}, \ldots, \{10\}$, eseguendo: Union(1,2); Union(3,4); Union(5,6); Union(7,8); Union(9,10); Union(1,3); Union(6,7); Union(7,1); Union(3,9); Union(5,9).

Esercizio 4.
Sia $G = (V, E)$ un grafo orientato.

a. Si scriva lo pseudocodice di una procedura che dati G e $u \in V$ decide se ogni nodo di V raggiunge u. Si dimostri la correttezza e si calcoli la complessità della procedura proposta.

b. Dato G si consideri il problema di decidere se esiste un nodo $u \in V$ raggiunto da ogni altro nodo. Come si potrebbe utilizzare la procedura descritta al punto a. per risolvere il problema? Quale sarebbe la complessità di tale procedura? Si potrebbe risolvere il problema in modo più efficiente?

Esercizio 5.
Si simuli, mostrando ad ogni iterazione lo stato di Q, π e d, l'esecuzione dell'algoritmo di Dijkstra sul grafo in figura, utilizzando come nodo sorgente S.



Esercizio 6.
Si consideri un grafo $G = (V, E, \omega)$ non orientato, connesso e pesato in cui $\omega : E \to \mathbb{N}^+$ è iniettiva.

a. Si dimostri o si refuti tramite un controesempio, la seguente affermazione: esiste un *unico* MST (minimum spanning tree) per G.

b. Un *SbMST (second-best minimum spanning tree)* per G è uno spanning tree per G tale che il suo peso è *maggiore* di quello di un MST per G, ma *minore o uguale* a quello di un qualsiasi altro. Esiste un *unico* SbMST per G?

1.3 § Compito di ASD del 24-07-17

Esercizio 1.
Sia A un array di n numeri naturali. Si consideri il problema di stampare in ordine *crescente* tutti i numeri che compaiono in A almeno $\lceil \frac{n}{k} \rceil$ volte, dove $k > 0$ è una costante.

- Si scriva lo pseudo-codice di una procedura efficiente che, dati A, n e k, risolva il problema proposto.
- Si dimostri la correttezza e si calcoli la complessità dell'algoritmo proposto.

Esercizio 2.
Sia $G = (V, E, \omega)$ un grafo orientato, con funzione di peso degli archi $\omega : E \to \mathbb{R}^+$.

Si proponga un algoritmo efficiente che, *per ogni coppia* di vertici $u, v \in V$, calcoli la lunghezza (cioè, il numero di archi) di un percorso di peso minimo da u a v (si assuma $+\infty$ se u non raggiunge v).

Si calcoli la complessità e si dimostri la correttezza dell'algoritmo.

1.4 § Compito di ASD del 04-09-17

Esercizio 1.
Sia A un array di n numeri interi. Si consideri il problema di decidere se esistono 3 posizioni distinte x, y, z in A tali che $A[x] + A[y] + A[z] = 0$.

- Si scriva lo pseudo-codice di una procedura che, dati A e n, risolva il problema proposto. Calcolare la sua complessità.
- Sapreste fornire un algoritmo di complessità $\mathcal{O}(n^2)$ per risolvere il problema?
  (*Suggerimento:* si ordini A, poi si utilizzino 3 contatori i, j, k: i assume tutti i valori da 1 a $n-2$; poi per ogni valore di i, j viene inizializzato a $i+1$, e k ad n....) Provare la correttezza.

Esercizio 2.
Sia $G = (V, E)$ un grafo orientato e *aciclico*. Una *foglia* di G è un nodo $v \in V$ *senza archi uscenti*. Per ogni $v \in V$, definiamo il valore $pf_G(v)$ come il numero di percorsi in G da v ad una qualsiasi foglia di G.

Si risponda alle seguenti domande:

1. Se da un nodo v di G escono m archi distinti $(v, v_1), \ldots, (v, v_m) \in E$:
   - assumendo di conoscere $pf_G(v_1), \ldots, pf_G(v_m)$, quanto vale $pf_G(v)$?
   - in una visita DFS di G, cosa si può dire dei tempi di *fine* visita di $v_1, \ldots, v_m$, rispetto al tempo di *fine* visita di v? Perché?
2. Fornire un algoritmo *efficiente* per calcolare $pf_G(v)$, per ogni $v \in V$. Dimostrare la correttezza dell'algoritmo proposto, e calcolare la sua complessità.

Capitolo 2

ANNO 2015-2016

2.1 § Compitino di ASD del 22-02-16

Esercizio 1. Compito A
Sia A un vettore di interi di lunghezza n. Rispondere alle seguenti domande, motivando opportunamente le risposte.

a. Determinare un mediano di A ha complessità $\Theta(n \log n)$?

b. Si può ordinare A con CountingSort con complessità $\Theta(n)$?

c. Se ogni elemento compare in A ripetuto $\Omega(n)$ volte, allora QuickSort ordina A in $O(n \log n)$?

d. InsertionSort ha complessità $\Omega(n \log n)$?

Esercizio 2. Compito A
Sia A un vettore che memorizza una max-heap secondo le convenzioni viste a lezione con $n = length[A]$ e $m = heap\_size[A]$. Rispondere alle seguenti domande, motivando opportunamente le risposte.

a. È possibile che m sia maggiore strettamente di n?

b. È possibile che $A[m]$ sia maggiore strettamente di $A[1]$?

c. È possibile che $A[n]$ sia maggiore strettamente di $A[1]$?

d. Determinare una limitazione inferiore alla complessià per individuare un minimo di A.

e. Determinare una limitazione inferiore alla complessità per l'ordinamento in senso crescente di A.

Nota: le limitazioni proposte ai punti d. ed e. devono essere le più precise possibili.

Esercizio 3. Compito A
Sia A un vettore di interi di lunghezza n ed l, r due indici compresi tra 1 e n. Si consideri il seguente codice ricorsivo.

```
TriSort(A, l, r)
1: if l < r then
2:     m = ⌊(l + r)/2⌋;
3:     p = ⌊(l + m)/2⌋;
4:     q = ⌊(m + r)/2⌋;
5:     TriSort(A, l, m);
6:     TriSort(A, m + 1, r);
7:     TriSort(A, p + 1, q − 1);
8:     Merge(A, l, m, r);
9: end if
```

a. Determinare e risolvere l'equazione ricorsiva di complessità di TriSort, evidenziando i passaggi eseguiti.

b. TriSort$(A, 1, n)$ ordina il vettore A? Motivare opportunamente la risposta.

Esercizio 4. Compito A
Diciamo che in un vettore privo di ripetizioni un elemento è *spiazzato* se compare in una posizione diversa da quella in cui comparirebbe se il vettore fosse ordinato. Sia A un vettore di interi privo di ripetizioni di lunghezza n. Proporre tramite pseudo-codice due algoritmi efficienti per ordinare A nelle seguenti ipotesi:

a in A ci sono $O(\log \log n)$ elementi spiazzati;

b in A ci sono il 2% di elementi spiazzati e sono tutti compresi tra $2n$ e $3n$.

Dimostrare la correttezza e valutare la complessità delle soluzioni proposte.

Esercizio 5. Compito A
Sia T un BST. Si vogliono stampare le chiavi di T memorizzate in nodi il cui sottoalbero radicato nel figlio sinistro contiene più chiavi del sottoalbero radicato nel figlio destro.

a. Si rappresenti un BST la cui visita in pre-order ha come risultato $10, 5, 1, 20, 15, 25$. Si mostri quali chiavi verrebbero stampate in base alla condizione sopra descritta.

b. Proporre tramite pseudo-codice un algoritmo efficiente per risolvere il problema proposto. Valutarne la complessità e dimostrarne la correttezza.

Esercizio 6. Compito A
Supponiamo che in un linguaggio di programmazione ogni oggetto occupi in memoria RAM un multiplo di 8 byte e che quindi gli oggetti vengano mappati in indirizzi di memoria multipli di 8. Si vogliono memorizzare alcuni oggetti attualmente in RAM in una tabella di hash T di dimensione m in cui le collisioni vengono gestite con chaining. Giudicare se le seguenti funzioni, che associano all'indirizzo *address* di un oggetto una posizione in T, siano buone o cattive funzioni di hash.

a. $h_1(address) = address \mod m$

b. $h_2(address) = (address * 8) \mod m$

c. $h_3(address) = (address \mod 8) \mod m$

Motivare le risposte fornite.
In caso nessuna delle funzioni sopra elencate sia una buona funzione di hash, proporre una buona funzione di hash per il problema descritto.

Esercizio 1. Compito B
Sia A un vettore che memorizza una min-heap secondo le convenzioni viste a lezione con $n = length[A]$ e $m = heap\_size[A]$. Rispondere alle seguenti domande, motivando opportunamente le risposte.

a. È possibile che m sia maggiore strettamente di n?

b. È possibile che $A[m]$ sia minore strettamente di $A[1]$?

c. È possibile che $A[n]$ sia minore strettamente di $A[1]$?

d. Determinare una limitazione inferiore alla complessià per individuarei un massimo di A.

e. Determinare una limitazione inferiore alla complessità per l'ordinamento in senso decrescente di A.

Nota: le limitazioni proposte ai punti d. ed e. devono essere le più precise possibili.

Esercizio 2. Compito B
Sia A un vettore di interi di lunghezza n. Rispondere alle seguenti domande, motivando opportunamente le risposte.

a. InsertionSort ha complessità $\Omega(n \log n)$?

b. Determinare un mediano di A ha complessità $\Theta(n \log n)$?

c. Se ogni elemento compare in A ripetuto $\Omega(n)$ volte, allora QuickSort ordina A in $O(n \log n)$?

d. Si può ordinare A con CountingSort con complessità $\Theta(n)$?

Esercizio 3. Compito B
Diciamo che in un vettore privo di ripetizioni un elemento è *posizionato* se compare nella posizione in cui comparirebbe se il vettore fosse ordinato. Sia A un vettore di interi privo di ripetizioni di lunghezza n. Proporre tramite pseudo-codice due algoritmi efficienti per ordinare A nelle seguenti ipotesi:

a in A ci sono $O(\log \log n)$ elementi NON posizionati;

b in A ci sono il 98% di elementi posizionati e i restanti elementi sono tutti compresi tra $2n$ e $3n$.

Dimostrare la correttezza e valutare la complessità delle soluzioni proposte.

Esercizio 4. Compito B
Sia T un BST. Si vogliono stampare le chiavi di T memorizzate in nodi il cui sottoalbero radicato nel figlio sinistro contiene un numero pari di chiavi e il sottoalbero radicato nel figlio destro contiene un numero dispari di chiavi.

a. Si rappresenti un BST la cui visita in pre-order ha come risultato $10, 5, 1, 20, 15, 25$. Si mostri quali chiavi verrebbero stampate in base alla condizione sopra descritta.

b. Proporre tramite pseudo-codice un algoritmo efficiente per risolvere il problema proposto. Valutarne la complessità e dimostrarne la correttezza.

Esercizio 5. Compito B
Sia A un vettore di interi di lunghezza n ed l, r due indici compresi tra 1 e n. Si consideri il seguente codice ricorsivo.

a. Determinare e risolvere l'equazione ricorsiva di complessità di SortTri, evidenziando i passaggi eseguiti.

b. SortTri$(A, 1, n)$ ordina il vettore A? Motivare opportunamente la risposta.

```
SortTri(A, l, r)
1: if l < r then
2:     m = ⌊(l + r)/2⌋;
3:     p = ⌊(l + m)/2⌋;
4:     q = ⌊(m + r)/2⌋;
5:     SortTri(A, l, m);
6:     SortTri(A, m + 1, r);
7:     SortTri(A, p + 1, q − 1);
8:     Merge(A, l, m, r);
9: end if
```

Esercizio 6. Compito B
Supponiamo che in un linguaggio di programmazione ogni oggetto occupi in memoria RAM un multiplo di 8 byte e che quindi gli oggetti vengano mappati in indirizzi di memoria multipli di 8. Si vogliono memorizzare alcuni oggetti attualmente in RAM in una tabella di hash T di dimensione m in cui le collisioni vengono gestite con chaining. Giudicare se le seguenti funzioni, che associano all'indirizzo *address* di un oggetto una posizione in T, siano buone o cattive funzioni di hash.

a. $h_1(address) = address \mod m$

b. $h_2(address) = (address * 8) \mod m$

c. $h_3(address) = \lfloor (address \mod m)/8 \rfloor$

Motivare le risposte fornite.
In caso nessuna delle funzioni sopra elencate sia una buona funzione di hash, proporre una buona funzione di hash per il problema descritto.

2.2 § Compitino di ASD del 21-06-16

Esercizio 1.

a. Si scriva la definizione di B-tree.

b. Si consideri un B-tree di grado $t = 3$ vuoto. Si inseriscano nell'ordine le chiavi $1, 2, 3, 4, 5, 10, 6, 7, 8$. Si cancellino dal B-tree risultante nell'ordine le chiavi $1, 2, 3$. Illustrare il B-tree risultante al termine di ogni operazione di inserimento/cancellazione.

Esercizio 2. Sia T un RB-tree.

a. Si consideri il problema di trovare un nodo rosso a distanza minima dalla radice di T (nil se non esiste). Scrivere lo pseudo-codice di una procedura Min_Red(T) per risolvere il problema. Derminare la complessità e dimostrare la correttezza dell'algoritmo proposto.

b. Nell'ipotesi in cui la procedura al punto precedente su input T restituisca nil, scrivere lo pseudo-codice di una procedura efficiente per determinare l'altezza di T. Derminare la complessità e dimostrare la correttezza dell'algoritmo proposto.

Esercizio 3.
Si consideri il problema di gestire insiemi disgiunti di chiavi intere su cui si deve essere in grado di effettuare le seguenti operazioni:

1. data una chiave creare l'insieme contenente la chiave;
2. dati due puntatori a due chiavi, unire gli insiemi che contengono le due chiavi;
3. dato un puntatore a una chiave k, determinare la somma delle chiavi appartenenti allo stesso insieme a cui appartiene k.

a. Si proponga una struttura dati che permetta di implementare in maniera efficiente le operazioni sopra descritte.

b. Si implementino tramite pseudo-codice le operazioni sopra descritte valutandone complessità e correttezza.

c. Si determini il costo di m operazioni complessive di cui n operazioni di creazione.

Esercizio 4.
Sia $G = (V, E)$ un grafo non orientato e connesso. Sia $r \in V$ un nodo del grafo. Si consideri il problema di determinare il numero di nodi a distanza massima da r.

a. Descrivere tramite pseudo-codice una procedura efficiente per risolvere il problema.

b. Valutare la complessità e dimostrare la correttezza della procedura proposta.

Esercizio 5. Stabilire quali delle seguenti affermazioni sono vere, motivando brevemente la risposta.

a. L'algoritmo di Kruskal ha una complessità sempre inferiore a quello di Prim.

b. Se G è un grafo non orientato, pesato, non connesso, allora non esiste un MST di G.

c. Se $G = (V, E, W)$ è un grafo non orientato, connesso, aciclico e pesato, allora è possibile calcolare un MST di G in tempo $O(|V| + |E|)$.

d. Se $G = (V, E, W)$ è un grafo non orientato, connesso e pesato in cui tutti gli archi hanno peso $k > 0$, allora l'algoritmo di Prim e quello di Dijkstra a partire da uno stesso nodo sorgente s restituiscono lo stesso risultato.

e. Se $G = (V, E, W)$ è un grafo orientato, connesso e pesato con pesi positivi e l'algoritmo di Dijkstra a partire da un nodo sorgente s ha restituito $\pi[u] = v$ (v è il predecessore di u), allora non può esistere un cammino da v ad u di peso inferiore a $W((v, u))$.

f. Se $G = (V, E, W)$ è un grafo orientato, connesso e pesato con pesi positivi, allora un cammino di peso minimo da s a p seguito da un cammino di peso minimo da p a q costituisce un cammino di peso minimo da s a q.

Esercizio 6.
Sia $G = (V, E, W)$ un grafo orientato e pesato con pesi positivi. Si assuma inoltre che G sia fortemente connesso, ovvero ogni coppia $u, v \in V$ di nodi è mutuamente raggiungibile. Sia L una lista di nodi di G. Si consideri il problema di determinare il peso di un cammino di peso minimo che passi per tutti i nodi di L nell'ordine in cui questi compaiono in L

- Descrivere tramite pseudo-codice una procedura efficiente per risolvere il problema. Valutarne la complessità e dimostrarne la correttezza.
- Come affrontereste il problema se fosse richiesto il peso di un cammino di peso minimo che passi per tutti i nodi di L in un qualsiasi ordine con possibilità di passare più volte per un nodo?

2.3 § Compito di ASD del 21-07-16

Esercizio 1.
Sia T un albero binario completo contenente n chiavi intere distinte. In particolare, ogni nodo x di T è una struttura dati formata dai seguenti campi:

- `key`$[x]$ la chiave intera di x;
- `left`$[x]$ un puntatore al figlio sinistro di x (NIL se non esiste);
- `right`$[x]$ un puntatore al figlio destro di x (NIL se non esiste);

- `parent`$[x]$ un puntatore al padre di x (NIL se non esiste).

T è un puntatore alla radice dell'albero.

Si consideri l'ordine determinato sulle chiavi di T dalla visita per livelli (dalla radice verso le foglie, procedendo da sinistra verso destra su ogni livello dell'albero).

a. Si scriva lo pseudo-codice di una procedura di visita per livelli di T che stampi le chiavi contenute nei nodi. Si calcoli la complessità e si dimostri la correttezza dell'algoritmo proposto.

b. Dato un nodo x di T diverso dalla radice, si consideri il problema di stampare la chiave contenuta nel nodo che precede immediatamente x nella visita per livelli (nota: non il genitore di x). Si scriva lo pseudo-codice di una procedura *efficiente* che risolva il problema. Si calcoli la complessità e si dimostri la correttezza dell'algoritmo proposto.

Esercizio 2.
Sia $G = (V, E, w)$ un grafo orientato e pesato, con pesi positivi. Dato $v \in V$ si consideri il problema di determinare (se esiste) il ciclo di peso minimo che contiene v

a. Si scriva lo pseudo-codice di una procedura *efficiente* che risolva il problema. Si calcoli la complessità e si dimostri la correttezza dell'algoritmo proposto.

b. Si scriva lo pseudo-codice di una procedura che determini il ciclo di G di peso minimo.

2.4 § Compito di ASD del 05-09-16

Esercizio 1.
Si consideri il problema di gestire un'insieme di chiavi intere che varia dinamicamente nel tempo. In particolare si intendono gestire in modo efficiente le seguenti operazioni:

- `Insert(`S`,`k`)` Inserimento nell'insieme S della chiave k;
- `Max_Delete(S)` Ricerca e cancellazione del massimo da S;
- `Min_Delete(`S`)` Ricerca e cancellazione del minimo da S.

a. Si descriva dettagliatamente la struttura dati più opportuna per la gestione delle operazioni sopra elencate.

b. Si scriva lo pseudo-codice delle procedure `Insert(`S`,`k`)`, `Max_Delete(`S`)`, `Min_Delete(`S`)`. Si dimostri la correttezza e valuti la complessità delle procedure proposte.

c. Si consideri anche l'operazione `Create(`S`,`A`)` che permette di inizializzare l'insieme S inserendovi tutte le chiavi memorizzate nel vettore A. Che complessità avrebbe un'implementazione efficiente di questa operazione? È possibile modificare la struttura dati per migliorare l'efficienza di `Create(`S`,`A`)`, senza peggiorare la complessità delle altre operazioni? (motivare opportunamente le risposte)

Esercizio 2.
Sia $G = (V, E)$ un grafo non orientato e connesso. G è un *cactus* se e soltanto se ogni arco compare in al più un ciclo semplice (ciclo senza nodi ripetuti).

a. Dimostrare o refutare tramite un controesempio la seguente affermazione. *Essendo G non orientato, una visita in profondità (DFS) di G classifica gli archi di G in archi archi di visita e archi all'indietro (non vi sono archi in avanti e di attraversamento).*

b. Sfruttando il suggerimento fornito dal punto precedente, modificare opportunamente la visita in profondità per determinare se G è un cactus.

2.5 § Compito di ASD del 21-09-16

Esercizio 1.
Sia A un vettore di interi positivi di lunghezza n. Si consideri il problema di disporre gli elementi in A in modo che il valore assoluto della differenza tra due elementi adiacenti decresca scorrendo il vettore da sinistra verso destra. Formalmente, per ogni $i \in [1, n-2]$ dovrà valere che

$$|A[i] - A[i+1]| \geq |A[i+1] - A[i+2]|.$$

a. Quali elementi di A dovranno essere spostati in $A[1]$ e $A[2]$ per rispettare la condizione sopra presentata?

b. Si scriva lo pseudo-codice di una procedura efficiente per risolvere il problema descritto. Si dimostri la correttezza e valuti la complessità della procedura proposta.

c. Si scriva lo pseudo-codice di una procedura efficiente per risolvere il problema descritto nel caso in cui la differenza tra il massimo e il minimo di A sia inferiore a n. Si dimostri la correttezza e valuti la complessità della procedura proposta.

Esercizio 2.
Sia $G = (V, E)$ un grafo non orientato e connesso. Si consideri la visita in ampiezza (BFS) di G a partire da un nodo $s \in V$.

a. Scrivere lo pseudo-codice della procedura BFS(G,s) e valutarne la complessità.

b. Sia Q la coda utilizzata in BFS(G,s). Enunciare le proprietà degli elementi che si trovano contemporaneamente in Q durante BFS(G,s).

c. Scrivere lo pseudo-codice di una procedura che calcoli le distanze da s dei nodi di G facendo uso solo del vettore d delle distanze e di una quantità costante di memoria aggiuntiva. In particolare, la procedura non può sfruttare gli altri vettori e la coda utilizzate dalla BFS standard. Dimostrare la correttezza e valutare la complessità della procedura proposta.

2.6 § Compito di ASD del 25-01-17

Esercizio 1.
Sia T un B-tree di grado $t \geq 2$ contenente n chiavi distinte.

1. Scrivere lo pseudo-codice di una procedura che, ricevuto in input un nodo x di T, conti il numero di chiavi del sottoalbero di T che ha x come radice.
   a. Dimostrare la correttezza della procedura.
   b. Calcolare la complessità della procedura in funzione di n e di t.
2. Scrivere lo pseudo-codice di una procedura che, ricevuti in input T ed i, dove $1 \leq i \leq n$, determini la i-esima chiave più piccola di T.
   a. Dimostrare la correttezza della procedura.
   b. Calcolare la complessità della procedura in funzione di n e di t.

Esercizio 2.
Sia $G = (V, E, \omega)$ un grafo non orientato, connesso e con funzione di peso degli archi $\omega : E \to \mathbb{R}^+$ a valori positivi. Si consideri il problema di decidere se G *ammette (almeno) due MST (minimum spanning trees) distinti*.
Si consideri il seguente fatto.

Fatto *G ammette (almeno) due MST distinti se e solo se* ***per ogni*** *MST T per G* ***esiste*** *un arco $(u, v) \in E$ tale che:*

1. *(u, v) non appartiene a T, e*
2. *il percorso che collega u e v in T ha un arco di peso uguale a $\omega(u, v)$.*

Si risponda ai seguenti quesiti.

a. Scrivere lo pseudo-codice di una procedura che risolva il problema descritto utilizzando il fatto precedente.

b. Valutare la complessità della procedura nei casi in cui:

   1. $|E| = (1/k) \cdot |V|^2$,
   2. $|E| = k \cdot |V|$,
   3. $|E| = |V| + k$,

   dove $k \in \mathbb{N}^+$ è una costante.

c. Dimostrare la correttezza della procedura (cioè, dimostrare che la procedura risponde "sì" *se e solo se* G ammette (almeno) due MST distinti) sfruttando il fatto precedente.

Capitolo 3

ANNO 2014-2015

3.1 § Compitino di ASD del 02-02-15

Esercizio 1a.
Si consideri la seguente equazione di complessità ricorsiva:

$$T(n) = \begin{cases} \Theta(1) & \text{se } n \leq 1, \\ 4 \cdot T\left(\frac{n}{2}\right) + \Theta(n^2 \log n) & \text{se } n > 1. \end{cases}$$

a. Si risolva l'equazione ricorsiva, evidenziando i passaggi eseguiti.

b. Siano $g(n) = n^2(\log n)^2$ e $f(n) = n^4$. Si mostri che $g(n) \in O(f(n))$.

Esercizio 1b.
Si consideri la seguente equazione di complessità ricorsiva:

$$T(n) = \begin{cases} \Theta(1) & \text{se } n \leq 1, \\ 8 \cdot T\left(\frac{n}{4}\right) + \Theta(n\sqrt{n}) & \text{se } n > 1. \end{cases}$$

a. Si risolva l'equazione ricorsiva, evidenziando i passaggi eseguiti.

b. Siano $g(n) = n\sqrt{n}(\log n)$ e $f(n) = n^3$. Si mostri che $g(n) \in O(f(n))$.

Esercizio 2a.
Si vuole realizzare una struttura dati, denominata *MinStack*, tale per cui siano disponibili le seguenti operazioni:

- MakeMinStack(n): restituisce uno *MinStack* vuoto di dimensione massima n.
- Empty(S): restituisce `true` se non contiene elementi, `false` altrimenti.
- Pop(S): restituisce, se esiste, l'ultimo elemento inserito, rimuovendolo da S.
- Push(S,x): se S non è pieno, inserisce in S l'elemento x.
- FindMin(S): restituisce, se esiste, l'elemento minimo memorizzato in S (senza rimuoverlo).

Si proponga una possibile implementazione della struttura *MinStack* basata su vettori assieme allo pseudo-codice delle operazioni sopra descritte. Si valuti la complessità delle procedure proposte.

Esercizio 2b.
Si vuole realizzare una struttura dati, denominata *MaxStack*, tale per cui siano disponibili le seguenti operazioni:

- MakeMaxStack(n): restituisce uno *MaxStack* vuoto di dimensione massima n.
- Empty(S): restituisce `true` se non contiene elementi, `false` altrimenti.
- Pop(S): restituisce, se esiste, l'ultimo elemento inserito, rimuovendolo da S.
- Push(S,x): se S non è pieno, inserisce in S l'elemento x.
- FindMax(S): restituisce, se esiste, l'elemento massimo memorizzato in S (senza rimuoverlo).

Si proponga una possibile implementazione della struttura *MaxStack* basata su vettori assieme allo pseudo-codice delle operazioni sopra descritte. Si valuti la complessità delle procedure proposte.

Esercizio 3a.
Sia H una *min-heap* contenente n interi *distinti* ed implementata con un vettore (come visto durante il corso). Rispondere alle seguenti domande, motivando le risposte.

a. Assumendo che i nodi interni di H siano già ordinati, è possibile ordinare tutti gli elementi di H con complessità *strettamente* inferiore a $O(n \log n)$?

b. Sia k costante rispetto ad n. Quanto costa determinare la k-esima chiave più piccola di H?

c. Quanto costa eliminare la chiave $H[i]$?

Esercizio 3b.
Sia H una *max-heap* contenente n interi *distinti* ed implementata con un vettore (come visto durante il corso). Rispondere alle seguenti domande, motivando le risposte.

a. Assumendo che i nodi interni di H siano già ordinati, è possibile ordinare tutti gli elementi di H con complessità *strettamente* inferiore a $O(n \log n)$?

b. Sia k costante rispetto ad n. Quanto costa determinare la k-esima chiave più piccola di H?

c. Quanto costa eliminare la chiave $H[i]$?

Esercizio 4.
Sia T un BST contenente n chiavi *distinte*. Siano inoltre k_1, k_2 due chiavi contenute in T. Si consideri il problema di stabilire se k_1 è un antenato di k_2 in T.

a. Scrivere lo pseudo-codice di una procedura *efficiente* Antenato(T,k_1,k_2) per risolvere il problema.

b. Determinare la complessità e dimostrare la correttezza dell'algoritmo proposto.

Esercizio 5a.
Partendo da un BST T inizialmente vuoto, si consideri l'inserimento delle chiavi $8, 12, 9, 2, 4, 6, 20, 11, 3, 10, 1$ e la successiva cancellazione delle chiavi $9, 8, 2, 4$. Si illustri come varia T dopo ogni operazione di inserimento/cancellazione.
N.B: Le operazioni vanno eseguite nell'ordine dato.

Esercizio 5b.
Partendo da un BST T inizialmente vuoto, si consideri l'inserimento delle chiavi $10, 5, 7, 12, 2, 4, 15, 13, 3, 20, 14$ e la successiva cancellazione delle chiavi $10, 4, 20, 15$. Si illustri come varia T dopo ogni operazione di inserimento/cancellazione.
N.B: Le operazioni vanno eseguite nell'ordine dato.

Esercizio 6.
Il Dott. M.C., esperto programmatore C++, ha bisogno di implementare una funzione per ordinare un vettore di interi e propone il seguente pseudo-codice:
dove A è un vettore di n interi con indici in $\{1, \ldots, n\}$. Si consideri la chiamata McSort($A, 1, n$).

```
McSort(A, l, r)
1: if r − l + 1 < 4 then
2:     QuickSort(A, l, r);
3: else
4:     McSort(A, l, r − 3);
5:     McSort(A, l + 3, r);
6: end if
```

a. Determinare l'equazione ricorsiva di complessità e risolverla.

b. Dimostrare o refutare la seguente affermazione: al termine dell'esecuzione, il vettore A è ordinato.

Esercizio 7a.
Si consideri una tabella hash T di dimensione $m = 8$, basata su *open addressing* e *linear probing*, secondo la seguente funzione $h : \{1, 2, \dots |U|\} \times \{0, 1, \dots, m-1\} \to \{0, \dots, m-1\}$:

$$h(k, i) = (k + i) \mod m.$$

Partendo da T inizialmente vuota, si considerino (nell'ordine dato) le seguenti operazioni:

- inserimento delle chiavi 17, 1, 20, 22, 6.
- rimozione della chiave 22.
- inserimento delle chiavi 12 e 36.

Si disegni la tabella T risultante dopo ciascun inserimento/cancellazione.

Esercizio 7b.
Si consideri una tabella hash T di dimensione $m = 8$, basata su *open addressing* e *linear probing*, secondo la seguente funzione $h : \{1, 2, \dots |U|\} \times \{0, 1, \dots, m-1\} \to \{0, \dots, m-1\}$:

$$h(k, i) = (k + 2i) \mod m.$$

Partendo da T inizialmente vuota, si considerino (nell'ordine dato) le seguenti operazioni:

- inserimento delle chiavi 1, 18, 10, 22, 6.
- rimozione della chiave 18.
- inserimento delle chiavi 15 e 23.

Si disegni la tabella T risultante dopo ciascun inserimento/cancellazione.

3.2 § Compitino di ASD del 15-06-15

Esercizio 1.
Si consideri un generico B-tree T di grado t con n chiavi.

a. Si caratterizzi precisamente la complessità computazionale delle operazioni di ricerca e inserimento di una chiave in T rispetto a t ed n.

b. Si mostri un esempio di un B-tree T di grado $t = 4$ e di altezza maggiore di 1 tale per cui la cancellazione di una chiave decrementi l'altezza di T.

Esercizio 2.

a. Si scriva la definizione di RB-tree.

b. Sia T un RB-tree tale che ogni cammino radice-foglia contiene almeno un nodo nero (diverso dalla radice) tale per cui sia il nodo padre che i nodi figli sono colorati anch'essi di nero. Si consideri il problema di decrementare di 1 l'altezza nera di T.

Scrivere lo pseudo-codice di una procedura *efficiente* per risolvere il problema. Derminare la complessità e dimostrare la correttezza dell'algoritmo proposto.

Esercizio 3.
Sia $G = (V, E, w)$ un grafo non orientato, pesato e connesso. Si consideri il problema di determinare un albero di supporto T di peso minimo per G.

a. Dimostrare o refutare la seguente affermazione: se $|E| > |V| - 1$ ed esiste un unico arco $e \in E$ di peso massimo, allora e non appartiene a T.

b. Scrivere lo pseudo-codice dell'algoritmo di Kruskal per determinare T.

c. Annotando lo pseudo-codice di cui al punto precedente, si determini la complessità dell'algoritmo nel caso in cui l'operazione di UNION venga fatta facendo puntare, al contrario di quanto visto a lezione, la radice dell'albero di altezza *maggiore* a quello di altezza *minore* (senza path-compression).

Esercizio 4.
Sia $G = (V, E, w)$ un grafo non orientato e pesato con pesi positivi. Dato un vertice $s \in V$, si consideri l'algoritmo di Dijkstra visto a lezione per il calcolo dell'albero dei cammini di peso minimo da s.

a. Si caratterizzi la complessità dell'algoritmo nel caso in cui la coda di priorità venga implementata con una min-heap o con un array. Quali sono i vantaggi/svantaggi nell'adottare le due soluzioni?

b. Si supponga di interrompere l'algoritmo quando la coda di priorità Q contiene un solo elemento (anziché quando è vuota). Più precisamente, si rimpiazzi la riga "**while** $Q \neq \emptyset$" con "**while** $|Q| > 1$".

L'algoritmo così modificato rimane corretto? Motivare la risposta.

c. Si assuma che esista un *unico* MST T di G. È possibile che l'albero restituito da DIJKSTRA(G,w,s) non condivida alcun arco con T? Motivare la risposta.

Esercizio 5.
Il Dott. Nick Price, famoso informatico americano, ha implementato la seguente procedura NPALGO che prende in input un grafo $G = (V, E)$ orientato, dove $V = \{1, \ldots, n\}$.

Algorithm 1 NPALGO($G = (V, E)$)

```
1: Inizializza vettore globale color, impostando color[v] ← WHITE per ogni v ∈ V
2: for each v ∈ V do
3:    if color[v] = WHITE and NPVISIT(G,v) then
4:       return true
5:    end if
6: end for
7: return false
```

Algorithm 2 NpVisit(G,v)

```
1: color[v] ← GRAY
2: for each z ∈ Adj[v] do
3:    if color[z] = GRAY then
4:       return true
5:    else if color[z] = WHITE and NpVisit(G,z) then
6:       return true
7:    end if
8: end for
9: color[v] ← BLACK
10: return false
```

Qual è la complessità computazionale di una chiamata alla procedura NpAlgo(G)? Che cosa rappresenta l'output della procedura?

Esercizio 6.
Sia $G = (V, E)$ un grafo non orientato e sia $k \in \mathbb{N}$ una *costante*. Si consideri il problema di determinare se esiste un vertice $v \in V$ tale per cui da esso sia possibile raggiungere al più k vertici distinti.

Scrivere lo pseudo-codice di una procedura *efficiente* per risolvere il problema. Derminare la complessità e dimostrare la correttezza dell'algoritmo proposto.

3.3 § Compito di ASD del 21-07-15

Esercizio 1. Compito A
Sia T un B-tree di grado t con n chiavi ed altezza h e siano k_1 e k_2 due interi con $k_1 < k2$.

a. Si dia la definizione di B-tree e si dimostri che $h \leq \log_t \left(\frac{n+1}{2}\right)$.

b. Si consideri il problema di stampare tutte le chiavi $k \in T$ nel range $[k_1, k_2]$. Scrivere una procedura *efficiente* PrintInKeys(T,k_1,k_2) per risolvere il problema. Si calcoli la complessità e si dimostri la correttezza dell'algoritmo proposto.

Esercizio 2. Compito A
Sia $G = (V, E)$ un grafo non orientato e connesso. Sia $U \subseteq V$ un sottoinsieme non vuoto di V. Si consideri il problema di determinare, se esiste, un albero T di copertura di G in cui tutti i nodi di $V \setminus U$ siano foglie.

a. Scrivere lo pseudo-codice di un algoritmo *efficiente* che, dati G e U restituisca T, se esiste, e `false` altrimenti. Determinare la complessità e dimostrare la correttezza della procedura proposta.

b. Come si può modificare l'algoritmo descritto al punto precedente nel caso in cui G sia pesato e T, oltre a rispettare le specifiche precedenti, debba essere di peso minimo. Come varia la complessità computazionale?

Esercizio 1. Compito B
Sia T un B-tree di grado t con n chiavi ed altezza h e siano k_1 e k_2 due interi con $k_1 < k2$.

a. Si dia la definizione di B-tree e si dimostri che $h \geq \log_{2t} (n + 1) - 1$.

b. Si consideri il problema di stampare tutte le chiavi $k \in T$ con $k < k_1$ oppure $k > k_2$. Scrivere una procedura *efficiente* PrintOutKeys(T,k_1,k_2) per risolvere il problema. Si calcoli la complessità e si dimostri la correttezza dell'algoritmo proposto.

Esercizio 2. Compito B
Sia $G = (V, E)$ un grafo non orientato e connesso. Sia $U \subset V$ un sottoinsieme proprio di V. Si consideri il problema di determinare, se esiste, un albero T di copertura di G in cui tutti i nodi di U siano foglie.

a. Scrivere lo pseudo-codice di un algoritmo *efficiente* che dati G e U restituisca T, se esiste, e `false` altrimenti. Determinare la complessità e dimostrare la correttezza della procedura proposta.

b. Come si può modificare l'algoritmo descritto al punto precedente nel caso in cui G sia pesato e T, oltre a rispettare le specifiche precedenti, debba essere di peso minimo. Come varia la complessità computazionale?

3.4 § Compito di ASD del 01-09-15

Esercizio 1.
Si consideri un vettore A di dimensione n tale per cui $A[i] \in \mathbb{N}$ per ogni $i = 1, \ldots, n$. Si consideri il problema di determinare se esiste una coppia di indici (i, j) tale per cui $A[i] + A[j] = k$, dove $k \in \mathbb{N}$.

a. Si scriva lo pseudo-codice di una procedura *efficiente* CheckSum(A,k) per risolvere il problema. Si dimostri la correttezza e si valuti la complessità della procedura.

b. È possibile migliorare la complessità dell'algoritmo proposto al punto precedente nell'ipotesi che A contenga un numero costante di elementi maggiori di n? Motivare la riposta e, in caso affermativo, descrivere tramite pseudo-codice un algoritmo più efficiente, valutando la complessità e dimostrando la correttezza.

Esercizio 2.
Sia $G = (V, E)$ un grafo orientato in cui ad ogni nodo è associato un colore che può essere VERDE o ROSSO.

a. Si proponga una struttura dati adeguata a memorizzare il grafo G comprensivo dei colori. Un ciclo C di G è detto *alternante* se in C ogni nodo VERDE è seguito da un nodo ROSSO e ogni ROSSO è seguito da un VERDE. Si consideri il problema di determinare se in G c'è almeno un ciclo alternante. Scrivere lo pseudo-codice di un algoritmo *efficiente* che restituisca un ciclo alternante se G ne contiene almeno uno e `NIL` altrimenti. Determinare la complessità e dimostrare la correttezza della procedura proposta.

b. È possibile generalizzare la soluzione proposta al punto precedente al caso di grafi con nodi di colore VERDE, ROSSO e GIALLO e si cerchi un ciclo di lunghezza 3ℓ in cui il VERDE è seguito dal ROSSO, il ROSSO è seguito dal GIALLO e il GIALLO è seguito dal VERDE? Cosa succede nel caso di k colori (data la sequenza in cui si devono alternare)? Motivare opportunamente le risposte fornite.

3.5 § Compito di ASD del 16-09-15

Esercizio 1.
Si consideri la struttura dati `Min-Heap` implementata tramite un vettore (secondo lo schema visto a lezione). Si considerino le seguenti operazioni:

- PrintUnique(H): data una `Min-Heap` H contenente n interi positivi, stampa gli elementi di un cammino dalla radice di H ad una foglia (qualsiasi) senza ripetizioni (se un elemento è presente in più copie, dovrà essere stampato una sola volta).
- Intersect(H_1,H_2): date due `Min-Heap` H_1 e H_2 contenenti rispettivamente n_1 e n_2 interi positivi, ritorna in output una nuova `Min-Heap` contenente *tutti e soli* gli elementi che appartengono sia a H_1 che a H_2.

a. Si descriva tramite pseudo-codice una procedura *efficiente* che implementa PrintUnique(H). Si dimostri la correttezza dell'algoritmo proposto e se ne determini la complessità.

b. Si descriva tramite pseudo-codice una procedura *efficiente* che implementa Intersect(H_1,H_2). Si dimostri la correttezza dell'algoritmo proposto e se ne determini la complessità in funzione di n_1 e n_2.

Esercizio 2.
Sia $G = (V, E, w)$ un grafo non orientato, connesso e pesato, tale per cui $w : E \to \mathbb{Z}$ e $w(e) > 0$. Si consideri il problema di determinare un albero di copertura di peso *minimo* (MST).

a. Si scriva lo pseudo-codice dell'algoritmo di Prim per calcolare un MST di G. Si determini, inoltre, la complessità dell'algoritmo.

Dimostrare la validità (motivando opportunamente la risposta) oppure presentare un controesempio per ciascuna delle seguenti affermazioni:

b. L'algoritmo di Prim ritorna un MST anche se G contiene archi di peso negativo.

c. Se G contiene un ciclo C il quale ha un unico arco e di peso minimo, allora e appartiene ad ogni MST di G. (Osservazione: G può contenere altri cicli oltre a C)

d. Si definisce come r-cammino (con $r > 0$) un cammino costituito da soli archi $e \in E$ di peso $w(e) < r$. Se G contiene un r-cammino tra due vertici s e t, allora ogni MST contiene un r-cammino da s a t.

3.6 § Compito di ASD del 26-01-16

Esercizio 1.
Dato un vettore A di interi di lunghezza n privo di ripetizioni e dato $k \leq n$ si consideri il problema di determinare i k elementi di A aventi distanza minima dall'elemento mediano di A. Ricordiamo che il mediano di A è l'elemento che finirebbe in posizione $\lfloor n/2 \rfloor$ se A venisse ordinato e che la distanza tra due elementi $A[i]$ e $A[j]$ di A è definita come $|A[i] - A[j]|$.

a. Quale è la complessità del miglior algoritmo per determinare l'elemento mediano di A?

b. Sfruttare l'algoritmo di cui al punto a. per descrivere un algoritmo ottimale per risolvere il problema proposto.

c. Fornire lo pseudocodice dell'algoritmo proposto al punto b., dimostrarne la correttezza e valutarne la complessità.

Esercizio 2.
Sia $G = (V, E)$ un grafo orientato in cui i nodi sono etichettati con numeri interi (a due a due distinti).

a. Fornire la definizione di *ordinamento topologico di un grafo orientato aciclico.*

b. Dimostrare o refutare le seguenti affermazioni.

   - *Un grafo orientato aciclico ammette sempre un unico ordinamento topologico.*
   - *Un grafo orientato aciclico ammette sempre un unico ordinamento topologico che se due nodi non sono connessi da un cammino, mette prima quello con etichetta minore.*

c. Descrivere tramite pseudocodice un algoritmo che determini se G è aciclico e in tal caso fornisca in output un ordinamento topologico di G. Dimostrare la correttezza e valutare la complessità della soluzione proposta. Anche nel caso si faccia uso di algoritmi visti a lezione si riporti tutto lo pseudocodice delle procedure.

Capitolo 4

ANNO 2013-2014

4.1 § Compitino di ASD del 10-02-14

Domanda 1a.
Si consideri la seguente equazione di complessità ricorsiva

$$T(n) = \begin{cases} O(1) & \text{se } n = 1 \\ 2T(\frac{n}{4}) + c\sqrt{n} & \text{se } n > 1, \end{cases}$$

dove $c > 0$ è una costante.

a. Si forniscano le definizioni di $O(g(n))$ e $\Omega(h(n))$;

b. si dimostri per *induzione* che $T(n) \in O(\sqrt{n} \log n)$.

Domanda 1b.
Si consideri la seguente equazione di complessità ricorsiva

$$T(n) = \begin{cases} O(1) & \text{se } n = 1 \\ 3T(\frac{n}{2}) + cn^2 & \text{se } n > 1, \end{cases}$$

dove $c > 0$ è una costante.

a. Si forniscano le definizioni di $O(g(n))$ e $\Omega(h(n))$;

b. si dimostri per *induzione* che $T(n) \in O(n^2)$.

Domanda 2.
Sia A un vettore di interi distinti di dimensione n. L'ingegnere M.C. ci fornisce un algoritmo QuickSelect(A, p, q, r), dove p, q, r sono tali che $1 \leq p \leq r \leq q \leq n$.
Dopo l'esecuzione di QuickSelect(A, p, q, r) valgono le seguenti condizioni su A:

- $A[r]$ contiene l'elemento che si troverebbe in posizione r in A se il sotto-vettore $A[p..q]$ fosse ordinato;
- $A[p..q]$ è partizionato rispetto al pivot $A[r]$, precisamente:

  $A[i] < A[r]$ per ogni i tale che $p \leq i < r$
  $A[r] < A[j]$ per ogni j tale che $r < j \leq q$.

M.C. ci dice che QuickSelect(A, p, q, r) ha complessità $O(\log m)$, dove $m = q - p + 1$.

a. Si utilizzi QuickSelect per costruire un algoritmo *efficiente* per ordinare A. Si scriva lo pseudo-codice e si calcoli la complessità dell'algoritmo proposto.

b. Può esistere un algoritmo QuickSelect con le proprietà sopra descritte? Motivare la risposta.

Domanda 3a.

Si consideri la struttura dati `Max-heap` vista a lezione. Sia H una `Max-heap` contenente n chiavi e sia k un intero. Si considerino le seguenti operazioni:

- InsertKey(H, k): inserisce la chiave k in H;
- RetrieveMin(H): restituisce la chiave di valore minimo da H, senza cancellarla;
- DeleteMin(H): rimuove la chiave di valore minimo da H.

Si vogliono implementare le operazioni descritte sopra in maniera *efficiente* ed *in-place*.

a. Si scriva lo pseudo-codice di RetrieveMin(H) e se ne calcoli la complessità.

b. Si scriva lo pseudo-codice di DeleteMin(H) e se ne calcoli la complessità.

c. Qual è la complessità di DeleteMin(H) nel caso peggiore? Si fornisca un esempio di heap H con $n = 8$ elementi corrispondente al caso peggiore e si illustri l'esecuzione di DeleteMin(H) su H.

d. Sia $\max(H)$ il valore della chiave massima di H e sia m un intero tale che $m > \max(H)$. Qual è la complessità di InsertKey(H, m) nel caso migliore e nel caso peggiore? Motivare la risposta.

Domanda 3b.

Si consideri la struttura dati `Min-heap` vista a lezione. Sia H una `Min-heap` contenente n chiavi e sia k un intero. Si considerino le seguenti operazioni:

- Insert(H, k): inserisce la chiave k in H;
- RetrieveMax(H): restituisce la chiave di valore massimo da H, senza cancellarla;
- DeleteMax(H): rimuove la chiave di valore massimo da H.

Si vogliono implementare le operazioni descritte sopra in maniera *efficiente* e *in-place*.

a. Si scriva lo pseudo-codice di RetrieveMax(H) e se ne calcoli la complessità.

b. Si scriva lo pseudo-codice di DeleteMax(H) e se ne calcoli la complessità.

c. Qual è la complessità di DeleteMax(H) nel caso migliore? Si fornisca un esempio di heap H con $n = 8$ elementi corrispondente al caso migliore e si illustri l'esecuzione di DeleteMax(H) su H.

d. Sia $\min(H)$ il valore della chiave minima di H e sia m un intero tale che $m < \min(H)$. Qual è la complessità di Insert(H, m) nel caso migliore e nel caso peggiore? Motivare la risposta.

Domanda 4a.

Sia dato un albero binario di ricerca T contenente n chiavi distinte. Si consideri il problema di stampare in ordine *crescente* $\lfloor \log n \rfloor$ chiavi *distinte* di T.

a. Si scriva lo pseudo-codice di un algoritmo *efficiente* per risolvere il problema descritto;

b. qual è la complessità dell'algoritmo proposto al punto a.?

Domanda 4b.

Sia dato un albero binario di ricerca T contenente n chiavi. Dato un nodo x in T, sia $size(x)$ il numero di nodi nel sottoalbero di T radicato in x, compreso x. Sia dato un intero $c \in [1, n]$ e si consideri il problema di stampare in ordine *decrescente* le chiavi $key[x]$ realtive ai nodi x per i quali $size(x) = c$.

a. Si scriva lo pseudo-codice di un algoritmo *efficiente* per risolvere il problema descritto;

b. qual è la complessità della procedura proposta al punto a.?

Domanda 5a.

Sia T un albero binario di ricerca. Sia $\langle 1, 8, 12, 13, 15, 27, 30\rangle$ la lista delle chiavi ottenuta da una visita *inorder* di T. Sia $\langle 1, 13, 12, 8, 27, 30, 15\rangle$ la lista delle chiavi ottenuta da una visita *postorder* di T.

a. Si ricostruisca T.

b. Si inseriscano in T le chiavi $3, 2, 32, 29$, illustrando i passaggi eseguiti.

c. Si cancellino dall'albero ottenuto al punto b. le chiavi $15, 27, 3, 8$, illustrando i passaggi eseguiti.

N.B: Le operazioni di inserimento e cancellazione vanno eseguite nell'ordine dato.

Domanda 5b.

Sia T un albero binario di ricerca. Sia $\langle 4, 12, 16, 17, 19, 31, 34\rangle$ la lista delle chiavi ottenuta da una visita *inorder* di T. Sia $\langle 4, 17, 16, 12, 31, 34, 19\rangle$ la lista delle chiavi ottenuta da una visita *postorder* di T.

a. Si ricostruisca T.

b. Si inseriscano in T le chiavi $7, 6, 35, 32$, illustrando i passaggi eseguiti.

c. Si cancellino dall'albero ottenuto al punto b. le chiavi $19, 31, 7, 12$, illustrando i passaggi eseguiti.

N.B: Le operazioni di inserimento e cancellazione vanno eseguite nell'ordine dato.

Domanda 6a.

Sia A un vettore non vuoto contenente n interi a valori nell'intervallo $[1, k]$, con k costante rispetto ad n. Si consideri il seguente pseudo-codice:

```
ALGORITMO1(A, n, k)
 1: B ← vettore temporaneo di dimensione k
 2: for i ← 1 to k do
 3:     B[i] ← 0
 4: end for
 5: for i ← 1 to n  do
 6:     B[A[i]] ← B[A[i]] + 1
 7: end for
 8: x ← 1
 9: v ← B[A[1]]
10: for i ← 2 to n  do
11:     if B[A[i]] > v then
12:         x ← i
13:         v ← B[A[i]]
14:     end if
15: end for
16: RETURN A[x]
```

Si supponga che la creazione del vettore temporaneo B (linea 1) abbia complessità lineare nella sua dimensione.

a. Che cosa rappresenta l'output della chiamata ALGORITMO1(A, n, k)? Qual è la sua complessità rispetto ad n?

b. L'algoritmo è *in-place*? Motivare la risposta.

Domanda 6b.

Sia A un vettore non vuoto contenente n interi a valori nell'intervallo $[1, k]$, con k costante rispetto ad n. Si consideri il seguente pseudo-codice:

```
ALGORITMO1(A, n, k)
 1: B ← vettore temporaneo di dimensione k
 2: for i ← 1 to k do
 3:     B[i] ← 0
 4: end for
 5: for i ← 1 to n  do
 6:     B[A[i]] ← B[A[i]] + 1
 7: end for
 8: x ← 1
 9: v ← B[A[1]]
10: for i ← 2 to n  do
11:     if B[A[i]] < v then
12:         x ← i
13:         v ← B[A[i]]
14:     end if
15: end for
16: RETURN A[x]
```

Si supponga che la creazione del vettore temporaneo B (linea 1) abbia complessità lineare nella sua dimensione.

a. Che cosa rappresenta l'output della chiamata ALGORITMO1(A, n, k)? Qual è la sua complessità rispetto ad n?

b. L'algoritmo è *in-place*? Motivare la risposta.

Domanda 7a.

Si consideri l'universo U costituito da tutti i vettori v di lunghezza n contententi interi positivi compresi tra k e $k+3$ (estremi inclusi), con $n \geq 10$ e k costante. Sia $T[0..1023]$ una tabella hash gestita attraverso open addressing e scansione lineare (linear probing).

a. Quali sono le caratteristiche che devono essere soddisfatte da una buona funzione di hash?

b. Si proponga una ragionevole funzione di hash $h : U \times \mathbb{N} \to \{0, 1, \ldots, 1023\}$.

c. Considerando la funzione h definita al punto b. ed assumendo che T abbia almeno m posizioni libere, qual è la complessità di m inserimenti nel caso ottimo e nel caso pessimo? Motivare la risposta.

Domanda 7b.

Si consideri l'universo U costituito da tutti i vettori v di lunghezza n contententi interi positivi compresi tra k e $k+3$ (estremi inclusi), con $n \geq 10$ e k costante. Sia $T[0..4095]$ una tabella hash gestita attraverso open addressing e scansione lineare (linear probing).

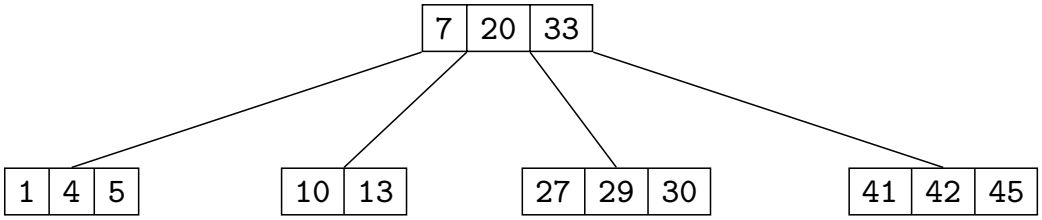
a. Quali sono le caratteristiche che devono essere soddisfatte da una buona funzione di hash?

b. Si proponga una ragionevole funzione di hash $h : U \times \mathbb{N} \to \{0, 1, \ldots, 4095\}$.

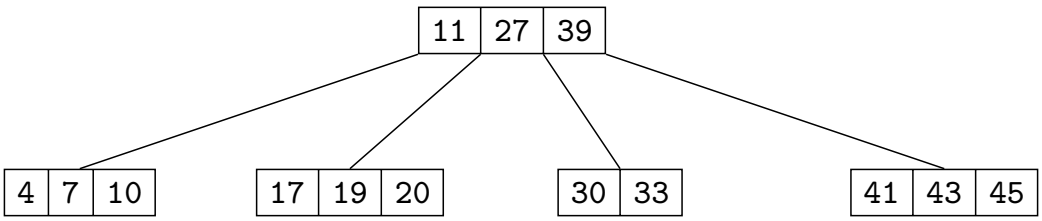
c. Considerando la funzione h definita al punto b. ed assumendo che T abbia almeno m posizioni libere, qual è la complessità di m inserimenti nel caso ottimo e nel caso pessimo? Motivare la risposta.

4.2 § Compitino di ASD del 16-06-14

Esercizio 1a.
Partendo dal B-tree sottostante di grado $t = 2$, illustrare (graficamente e con una breve spiegazione dei passaggi eseguiti) i B-tree risultanti dall'inserimento delle chiavi 50, 31, 28 e dalla cancellazione delle chiavi 41, 7. Le operazioni vanno eseguite nell'ordine proposto.



Esercizio 1b.
Partendo dal B-tree sottostante di grado $t = 2$, illustrare (graficamente e con una breve spiegazione dei passaggi eseguiti) i B-tree risultanti dall'inserimento delle chiavi 31, 1, 25 e dalla cancellazione delle chiavi 39, 11. Le operazioni vanno eseguite nell'ordine proposto.



Esercizio 2.
Sia dato un RB-tree T. Sia T' un albero ottenuto da T colorando di rosso un nodo nero diverso dalla radice e dalle foglie. Si consideri il problema di ricostruire T a partire da T'.

Scrivere lo pseudo-codice di una procedura *efficiente* FixRBT(T') per risolvere il problema. Determinare e risolvere l'equazione di complessità. Dimostrare, infine, la correttezza dell'algoritmo proposto.

Esercizio 3a.
Si illustri la famiglia di insiemi disgiunti che si ottiene eseguendo la seguente sequenza di operazioni sugli 8 insiemi singoletto contenenti gli elementi $1, 2, \ldots, 8$: *Union*(8,2); *Union*(3,7); *Union*(5,1); *Union*(4,6); *Union*(4,3); *Union*(1,4); *Find*(8); *Find*(4). Si utilizzi la rappresentazione con alberi con le euristiche di union-by-rank e path-compression. Si assuma che, qualora vengano uniti due alberi aventi lo stesso rango, l'albero rappresentato dall'elemento minore venga fatto puntare all'albero rappresentato dall'elemento maggiore.

Esercizio 3b.
Si illustri la famiglia di insiemi disgiunti che si ottiene eseguendo la seguente sequenza di operazioni sugli 8 insiemi singoletto contenenti gli elementi $1, 2, \ldots, 8$: *Union*(5,4); *Union*(6,7); *Union*(7,5); *Find*(4); *Union*(3,8); *Union*(2,1); *Union*(8,1); *Union*(4,2). Si utilizzi la rappresentazione con alberi con le euristiche di union-by-rank e path-compression. Si assuma che, qualora vengano uniti due alberi aventi lo stesso rango, l'albero rappresentato dall'elemento minore venga fatto puntare all'albero rappresentato dall'elemento maggiore.

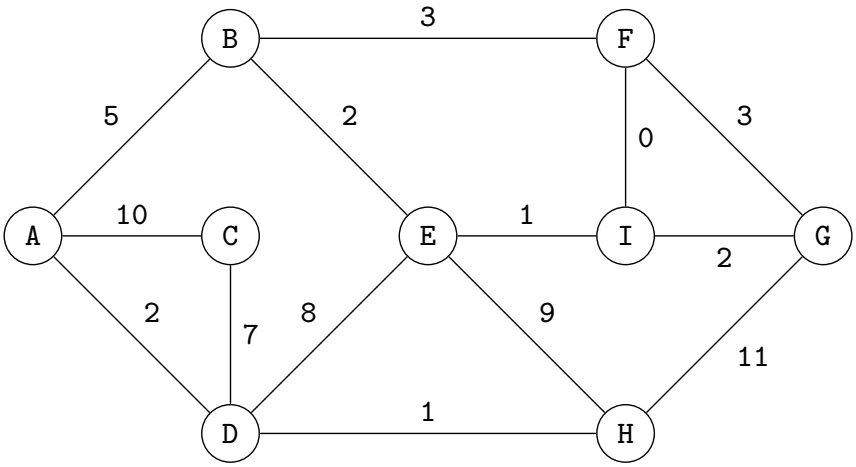
Esercizio 4.
Sia $G = (V, E, w)$ un grafo non orientato e connesso, dove $w : E \to \{k \mid k \in \mathbb{N} \wedge k > 0\}$ è una funzione di peso. Sia $G' = (V, E, w')$, dove $w'(e) = c \cdot w(e)$ per ogni $e \in E$, $c \in \mathbb{N}$ è una costante e $c > 0$. Dimostrare o refutare le seguenti affermazioni:

a. Se π è un cammino di peso minimo fra due nodi $s, t \in V$ in G, allora π è un cammino di peso minimo su G'.

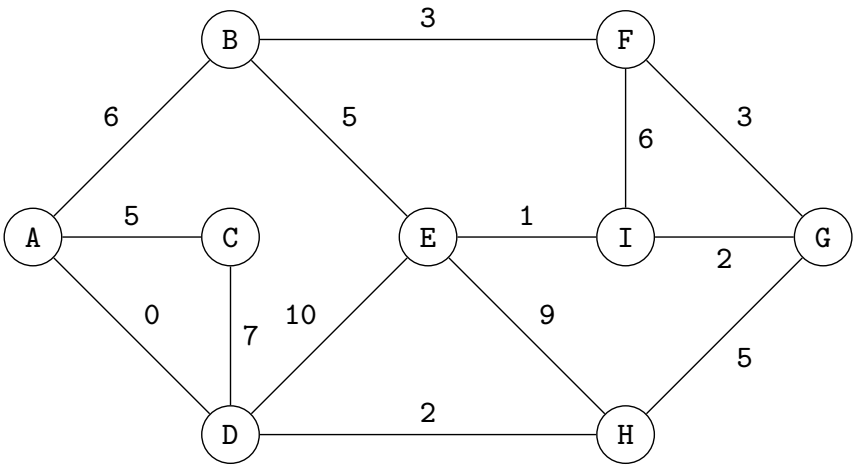
b. Se $s, t \in V$, T è un MST di G e π è un cammino fra s e t in T, allora esiste un MST di G' che contiene π.

c. Mst-Prim(G) e Mst-Prim(G') restituiscono lo stesso albero se e solo se w è iniettiva.

Esercizio 5a.
Si consideri il grafo G pesato e non orientato nella figura che segue. Si illustrino i passi compiuti dall'algoritmo di Dijkstra, applicato a G a partire dal nodo `C`. Si mostri, infine, l'albero risultante, associando ad ogni vertice il peso del cammino minimo calcolato.



Esercizio 5b.
Si consideri il grafo G pesato e non orientato nella figura che segue. Si illustrino i passi compiuti dall'algoritmo di Dijkstra, applicato a G a partire dal nodo `E`. Si mostri, infine, l'albero risultante, associando ad ogni vertice il peso del cammino minimo calcolato.



Esercizio 6.
Si consideri il seguente pseudo-codice di un algoritmo che prende in input un grafo $G = (V, E, w)$ non orientato e pesato. Sia inoltre $w : E \to \{1, \ldots, k\}$ una funzione di peso con $k \in \mathbb{N}$ costante.

```
ALGORITMO1(G = (V, E, w))
 1: A ← ∅
 2: for each v ∈ V do
 3:    MAKE-SET(v)
 4: end for
 5: ordina gli archi di E in senso decrescente rispetto al peso w usando CountingSort
 6: for (u, v) ∈ E preso in ordine di peso decrescente do
 7:    if FIND-SET(u) ≠ FIND-SET(v) then
 8:       UNION(u,v)
 9:    else
10:       A ← A ∪ {(u, v)}
11:    end if
12: end for
13: return A
```

a. Che cosa rappresenta l'insieme A restituito dalla procedura?

b. Assumendo che le operazioni MAKE-SET, UNION e FIND-SET gestiscano una collezione di insiemi disgiunti implementati con *liste pesate*, si calcoli la complessità della procedura.

Esercizio 7.
Un grafo è detto *hamiltoniano* se contiene un cammino che visita tutti i vertici *esattamente* una volta. Dato un grafo $G = (V, E)$ orientato e *aciclico*, si consideri il problema di determinare se G è hamiltoniano.

a. Scrivere lo pseudo-codice di una procedura *efficiente* ISHAMILTONIAN(G) che restituisce `true` se G è hamiltoniano, `false` altrimenti. Determinare la complessità e dimostrare la correttezza dell'algoritmo proposto.

b. Dimostrare o refutare la seguente affermazione: dato G orientato e aciclico, se ISHAMILTONIAN(G) restituisce `true`, allora un cammino che visita tutti i vertici di G esattamente una volta è unico.

4.3 § Esame di ASD del 16-07-14

Esercizio 1.
Sia dato un vettore A di interi (elementi di $\mathbb{Z}$) di dimensione n. Si consideri il problema di determinare un vettore B tale per cui, per ogni $i \in \{1, \ldots, n\}$, in $B[i]$ è memorizzato il *numero* di posizioni di elementi strettamente minori di $A[i]$ in $A[i+1, \ldots, n]$. In formule:

$$B[i] = |\{z \mid A[z] < A[i] \ \wedge \ z = i+1, \ldots, n\}| \,.$$

a. Dato il vettore di esempio $A = [2, -7, 8, 3, -5, -5, 9, 1, 12, 4]$, si illustri il vettore B relativo.

b. Si scriva lo pseudo-codice di una procedura $O(n^2)$ che risolva il problema. Si calcoli la complessità e si dimostri la correttezza dell'algoritmo proposto.

c. Nell'ipotesi che esista soltanto un numero costante di valori $A[i]$ tali per cui $A[i] \neq 0$, si descriva un algoritmo *efficiente* che risolva lo stesso problema.

Esercizio 2.
Sia $G = (V, E)$ un grafo non orientato e connesso con $V = \{1, 2, \ldots, n\}$. Siano V_1, V_2 due insiemi di vertici tali che $V_1 \subseteq V$, $V_2 \subseteq V$ e $|V_1| \in O(1)$. Si definisca la distanza $d_G(V_1, V_2)$ come la minima distanza in G fra tutte le coppie di vertici (w_1, w_2) tali per cui $w_1 \in V_1$ e $w_2 \in V_2$.

a. Si proponga una struttura dati adatta a gestire G, V_1 e V_2.

b. Si scriva lo pseudo-codice *completo* di una procedura *efficiente* che, dati G, V_1 e V_2 (implementati come è stato descritto nel punto a.), restituisce $d_G(V_1, V_2)$. Si calcoli la complessità e si dimostri la correttezza dell'algoritmo proposto.

c. Si riconsideri il problema senza assumere $|V_1| \in O(1)$.

d. Supponendo di avere a disposizione *soltanto* un albero di supporto T di G (oltre a V_1 e V_2). È ancora possibile calcolare $d_G(V_1, V_2)$? Motivare la risposta.

4.4 § Esame di ASD del 01-09-14

Esercizio 1.
Sia A un array di n interi che rappresenta una min-heap.

a. Si consideri il problema di determinare l'esistenza di una chiave k in A assumendo che al più $O(\sqrt{n})$ siano minori di k. Si scriva lo pseudo-codice di un algoritmo *efficiente* che risolva il problema proposto. Si dimostri la correttezza e si calcoli la complessità della procedura proposta.

b. Si calcoli la complessità dell'algoritmo HEAP-TO-BST(A) illustrato qui di seguito.

c. Dimostrare o refutare la seguente affermazione: l'altezza dell'albero T restituito da HEAP-TO-BST(A) è $O(\log n)$.

HEAP-TO-BST(A)
1: $T \leftarrow$NEWEMPTYTREE()
2: **for** $i = 1$ **to** $heapsize(A)$ **do**
3: BST-INSERT(T,$A[i]$)
4: **end for**
5: **return** T

Esercizio 2.
Sia $G = (V, E, w)$ un grafo orientato, dove $w : E \to \mathbb{N}$ è una funzione di peso definita sugli archi. Dato un cammino π fra due vertici di V, si definisce $M(\pi)$ come il *massimo* peso degli archi in π. Si consideri il problema di determinare, dati $s, t \in V$, il *minimo* valore $M(\pi)$ fra tutti i cammini π da s a t in G.

a. Si scriva lo pseudo-codice di una procedura *efficiente* che, dati G, s e t, risolva il problema proposto.

b. Si dimostri brevemente la correttezza e si calcoli la complessità dell'algoritmo proposto.

4.5 § Esame di ASD del 26-09-14

Esercizio 1.
Sia A un array di n interi, a due a due distinti, dove n è dispari. A è detto *alternante* se vale la seguente condizione:

$$A[1] > A[2] < A[3] > A[4] < \cdots > A[n-1] < A[n]$$

Si scriva lo pseudo-codice di un algoritmo *efficiente* AlternaArray(A,n) che renda A alternante. Si dimostri la correttezza e si calcoli la complessità della procedura proposta.

Esercizio 2.
Sia $G = (V, E)$ un grafo non orientato e connesso.

a. Si descriva tramite pseudo-codice l'algoritmo di visita in profondità DFS(G) e se ne analizzi la complessità;

Si consideri la seguente definizione: un nodo $v \in V$ è detto *punto di articolazione* di G se la sua rimozione, assieme agli archi ad esso incidenti, disconnette G.

b. Sia G_π un albero di visita ottenuto dopo una chiamata a DFS(G). Si risponda, motivando le risposte, alle seguenti domande.

   Qual è una condizione necessaria e sufficiente per cui la radice di G_π sia un punto di articolazione di G? Quando un vertice interno (diverso dalla radice) di G_π è un punto di articolazione di G? Che relazione c'è fra le foglie di G_π ed i punti di articolazione di G?

c. Si consideri il problema di determinare un vertice che <u>non</u> è un punto di articolazione per G.

   Si scriva lo pseudo-codice di un algoritmo *efficiente* che risolva il problema. Si dimostri la correttezza e si calcoli la complessità della procedura proposta.

4.6 § Esame di ASD del 26-01-15

Esercizio 1.
Si consideri un generico albero binario di ricerca T.

a. Si scriva la definizione di albero binario di ricerca, fornendo un limite superiore ed un limite inferiore all'altezza di T in funzione del numero di chiavi n in esso contenute.

b. Si consideri il problema di spostare il nodo con chiave k (se esiste) come radice di T. Si scriva lo pseudo-codice di una procedura LiftKey(T,k) che risolva il problema proposto in modo *efficiente* ed *in-place*. Si dimostri la correttezza e si calcoli la complessità dell'algoritmo proposto.
   (Suggerimento: si considerino le rotazioni sui BST)

c. Date tre chiavi k_1, k_2, k_3, si scriva lo pseudo-codice di una procedura HasPath(T,k_1,k_2,k_3) che ritorni `true` se e soltanto se esiste un cammino in T dalla radice ad una delle foglie che contiene k_1, k_2 e k_3 (non necessariamente in questo ordine). Si dimostri la correttezza e si calcoli la complessità dell'algoritmo proposto.

Esercizio 2.
Sia $G = (V, E, w)$ un grafo orientato, dove $V = \{1, \ldots, n\}$ e $w : E \to \{1, \ldots, kn\}$ è una funzione di peso sugli archi con $k \in \mathbb{N}$ costante. Si consideri la seguente procedura.
Essa restituisce un vettore A contenente *tutti e soli* gli archi di attraversamento (cross-edge) ordinati per peso crescente e definiti dalla visita in profondità DFS(G) alla riga 1.

a. Si scriva lo pseudo-codice di una procedura *efficiente* che implementa GetCrossEdges(G), la quale restituisce un vettore A contenente *tutti e soli* i cross-edge definiti dalla precedente visita DFS(G). Si dimostri la correttezza e si valuti la complessità della procedura.

ALGORITHM 1. SORTEDCROSSEDGES

input: $G = (V, E, w)$

1 DFS(G)

2 $A \leftarrow$GETCROSSEDGES(G)

3 SORTCE(A)

4 **return** A

b. Si scriva lo pseudo-codice di una procedura *efficiente* che implementa SORTCE(A), la quale ordina per peso crescente i cross-edge memorizzati nel vettore A. Si dimostri la correttezza e si valuti la complessità della procedura

Capitolo 5

ANNO 2012-2013

5.1 § Compitino di ASD del 28-01-13

Domanda 1.
Si consideri l'equazione ricorsiva:

$$T(n) = \begin{cases} \Theta(1) & \text{se } n = 1 \\ T(n-1) + \Theta(\log n) & \text{se } n > 1. \end{cases}$$

a. Si risolva l'equazione ricorsiva.

b. Si illustri una procedura ricorsiva che abbia tale complessità.

Domanda 2.
Sia A un vettore di interi di dimensione n.

- Sia $A = [3, 1, 6, 3, 7, 2]$ e si utilizzi `Quicksort(A,1,6)` per ordinare il vettore, illustrando i passaggi eseguiti. Quante volte, all'interno di `Quicksort(A,1,6)`, viene richiamata la funzione `Partition`?
- Sia T l'albero delle chiamate ricorsive di `Quicksort(A,1,n)` per n generico. A che cosa corrispondono le foglie di T, e quante sono (in funzione di n)?
- In quali nodi di T viene richiamata `Partition`?

Domanda 3.
Sia B un vettore di interi di dimensione n.

a. Si scriva lo pseudo-codice di una procedura che, presi in input B ed n, restituisca una `min-heap` H contenente tutti gli elementi di B.

b. Dato $B = [7, 3, 4, 6, 2, 5]$, si illustri l'output H della procedura proposta al punto a. sull'input $(B, 6)$, evidenziando i passaggi eseguiti.

c. Sul generico input (B, n), la procedura proposta al punto a. restituisce sempre un vettore ordinato?

Domanda 4.
Partendo da un BST T vuoto, si rappresenti il BST T risultante dall'inserimento delle chiavi 15, 10, 12, 30, 3, 45, 20, 90, 1, 22, 5. Si cancellino poi le chiavi 15, 30, 3, illustrando i passaggi eseguiti. Le operazioni di inserimento e cancellazione vanno eseguite dell'ordine dato.

Domanda 5.
Si consideri la struttura dati `Ordered_List` costituita da elementi l contenenti i seguenti campi:

- $l.key$ è una chiave intera;

- $l.next$ è il puntatore all'elemento successivo (eventualmente NIL).

Inoltre, ogni elemento l è tale che $l.next \neq NIL \Rightarrow l.key < l.next.key$. Rappresentiamo una `Ordered_List` con un puntatore al suo primo elemento.

Si considerino le seguenti operazioni implementate usando la struttura dati `Ordered_List` :

`Search(l,k)` ritorna 1 se k è una chiave di l, 0 altrimenti;

`Insert(l,k)` inserisce k nella `Ordered_List` l se k non vi appartiene, altrimenti non fa nulla;

`ExtractMin(l)` estrae l'elemento con chiave minima in l.

a. Si dica se la seguente affermazione è vera o falsa: Sia l una `Ordered_List` in cui sono state eseguite n operazioni di `Insert`; se k non occorre in l, allora `Search(l,k)` richiede $\Omega(n)$ operazioni.

b. Calcolare la complessità della procedura `Insert(l,k)` in funzione del numero n di elementi di l.

c. Scrivere lo pseudo-codice delle procedure `Search(l,k)` ed `ExtractMin(l,k)`.

Domanda 6.

Sia S un insieme finito di interi e sia $r \in \mathbb{R}$. Si consideri il problema di determinare se S contiene almeno un elemento $s < r$, in ciascuna delle seguenti ipotesi:

a. gli elementi di S sono gestiti tramite una `min-heap`;

b. gli elementi di S sono gestiti tramite un `BST`.

Per ognuna delle due ipotesi precedenti, si scriva lo pseudo-codice di un algoritmo *efficiente* per risolvere il problema descritto e se ne calcoli la complessità nel caso peggiore e nel caso migliore.

Si forniscano esempi di input corrispondenti al caso migliore e al caso peggiore per gli algoritmi proposti nel caso a. e nel caso b., rispettivamente.

Domanda 7.

Dati due interi $h, k \in \mathbb{Z}$, con $h \leq k$, siano T_1 e T_2 due `BST` tali che:

- per ogni $x \in T_1$ vale $key[x] < h \vee key[x] > k$;
- per ogni $y \in T_2$ vale $h \leq key[y] \leq k$.

Si consideri il seguente pseudo-codice:

a. L'output restituito da BSTMERGE$(T_1, T_2, root[T_1], h, k)$ è un `BST`? Motivare la risposta.

b. Si calcoli la complessità di BSTMERGE rispetto al numero di nodi di T_1.

Domanda 8.

Si consideri una tabella hash T basata su chaining con liste di tipo *single-linked*. Sia assuma che venga utilizzata una funzione di hash $h : \{0, \ldots, n-1\} \to \{0, \ldots, m-1\}$ (con $n > m$), il cui calcolo abbia complessità $\Theta(1)$.

a. Qual è la complessità della ricerca di una chiave k in T nel caso pessimo?

b. Quali sono i vantaggi/svantaggi di utilizzare degli alberi binari di ricerca al posto delle liste?

```
BSTMERGE(T_1, T_2, x, h, k)
 1: if key[x] > k then
 2:    if left[x] ≠ NIL then
 3:       return BSTMERGE(T_1, T_2, left[x], h, k)
 4:    else
 5:       left[x] ← root[T_2]    % inserisce tutto l'albero T_2 a sinistra di x
 6:       return T_1
 7:    end if
 8: end if
 9: if key[x] < h then
10:    if right[x] ≠ NIL then
11:       return BSTMERGE(T_1, T_2, right[x], h, k)
12:    else
13:       right[x] ← root[T_2]    % inserisce tutto l'albero T_2 a destra di x
14:       return T_1
15:    end if
16: end if
```

5.2 § Compitino di ASD del 13-06-13

Esercizio 1.
A partire dal RB-tree vuoto effettuare nell'ordine gli inserimenti delle chiavi $1, 2, 3, 4, 5$. Dal RB-tree ottenuto, cancellare nell'ordine le chiavi 2 e 3 utilizzando, ove necessario, il predecessore. Si illustrino brevemente i passaggi eseguiti.

Esercizio 2.
Diciamo che un B-tree è *minimale* se ogni nodo contiene il numero minimo possibile di chiavi.

a. Si illustri il B-tree T minimale di altezza 2 e grado 3 contenente le chiavi $1, 2, 3, \ldots, k$, dove k è il numero di chiavi di T.

b. Si cancelli la chiave contenuta nella radice di T.

Esercizio 3.
Rispondere brevemente alle seguenti domande:

a. Quanti B-tree di altezza 1 e grado 2 si possono costruire (considerare solo le diverse topologie indipendentemente dalle chiavi memorizzate)?

b. Sia T un RB-tree. La chiave mediana di T si trova nella radice?

c. Sia T un RB-tree. Trovare ed eliminare la chiave mediana da T ha costo $O(h)$?

d. Sia T un RB-tree. La chiave minima di T si trova in un nodo che ha due figli NIL?

Esercizio 4.
Calcolare lo spazio utilizzato dalla rappresentazione dei grafi tramite liste di adiacenza nei seguenti casi:

a. il numero di archi uscenti da ogni nodo è una costante;

b. il grafo è un albero;

c. il grafo è completo.

Esercizio 5.
Descrivere la complessità computazionale di Kruskal nei seguenti casi:

a. rappresentazione ad albero senza euristiche;

b. rappresentazione ad albero con *union-by-rank* (unione per rango);

c. rappresentazione ad albero con *union-by-rank* (unione per rango) e *path-compression* (compressione per cammini).

Esercizio 6.
Dato il grafo G rappresentato in Figura 5.1, si illustri l'output di Topological Sort partendo dal nodo A. Nel caso vi siano più soluzioni, si selezionino i nodi rispetto all'ordine alfabetico delle loro etichette.

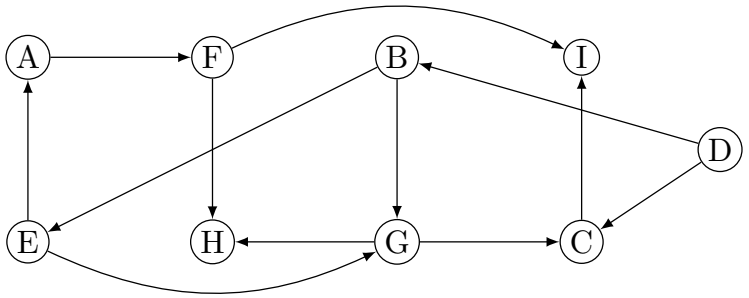


Figura 5.1: Esercizio 6

Esercizio 7.
Sia $G = (V, E)$ un grafo orientato, non pesato. Dati un nodo $s \in V$ e un intero non negativo $k \in \mathbb{N}$, sia L la lista dei nodi a distanza k da s (distanza = lunghezza del cammino minimo).

a. Si scriva lo pseudo-codice di un algoritmo per determinare L. Si calcoli la complessità e si dimostri la correttezza della soluzione proposta.

b. Dire se la seguente affermazione è vera o falsa: se la lista L restituita al punto a. non contiene $i \in V$, allora non esiste nessun cammino da s a i di lunghezza k.

c. Si supponga di avere a disposizione la lista I dei nodi a distanza $k-1$ da s. È possibile sfruttare I per ottenere una procedura più efficiente di quella proposta al punto a.? Motivare la risposta.

Esercizio 8.
Sia T un RB-Tree.

a. Scrivere la definizione di RB-tree.

b. Fornire lo pseudo-codice di una procedura *efficiente* per determinare l'altezza di T. Dimostrarne la correttezza e calcolarne la complessità.

c. Fornire lo pseudo-codice di una procedura *efficiente* per determinare l'altezza nera di T. Dimostrarne la correttezza e calcolarne la complessità.

5.3 § Esame di ASD del 16-07-13

Esercizio 1.
Sia T un RB-tree e k una chiave che *non* occorre in T. Si supponga che l'inserimento di k in T provochi un aumento dell'altezza nera di T. Si risponda brevemente alle seguenti domande:

a. Si fornisca la definizione di RB-tree e di altezza nera di un RB-tree.

b. Si fornisca un esempio per T e k, con T di altezza nera 2.

c. Se immediatamente dopo l'inserimento di k in T si effettua la rimozione di k, l'altezza nera di T decresce? In caso affermativo, fornire una dimostrazione; in caso negativo, esibire un controesempio.

d. Se invece di inserire in T la chiave k si fosse inserita una chiave k', l'altezza nera di T sarebbe sicuramente aumentata?

e. Si supponga che k sia più piccola di tutte le chiavi di T. Si scriva lo pseudo-codice della procedura INSERT(T, k) che inserisce k in T se l'inserimento di k fa aumentare l'altezza nera di T, altrimenti restituisce NIL.

N.B.1: Le risposte ai punti c. e d. devono essere fornite nel caso generale e non relativamente all'esempio fornito al punto b.
N.B.2: Al punto e. non è consentito richiamare procedure già viste a lezione, ma è necessario scrivere esplicitamente lo pseudo-codice di tutta la procedura.

Esercizio 2.
Sia A un vettore di interi *a due a due distinti*. Si consideri lo pseudo-codice che segue:

```
MeegerSort(A, p, q)
1: if p < q then
2:    m ← ⌊(p+q)/2⌋
3:    Scambia(A, m, q)
4:    MeegerSort(A, p, m)
5:    MeegerSort(A, m + 1, q − 1)
6:    Merge(A, p, m, q)
7: end if
```

a. Mostrare l'esecuzione di MeegerSort$(A, 1, 5)$ sul vettore A contenente i numeri da 1 a 5 in ordine decrescente.

b. Sia A un generico vettore di lunghezza n. Al termine di MeegerSort$(A, 1, n)$ il vettore A risulta ordinato? In caso affermativo, fornire una dimostrazione; in caso negativo, modificare gli indici delle chiamate ricorsive in modo da ottenere un algoritmo di ordinamento.

c. Valutare la complessità di MeegerSort descrivendo e risolvendo l'equazione ricorsiva di complessità.

5.4 § Esame di ASD del 16-09-13

Esercizio 1.
Si consideri il problema di gestire insiemi disgiunti di chiavi (interi) utilizzando la struttura dati `max-Heap`. In particolare, si vogliono implementare le seguenti operazioni:

`Make(x)` crea una max-Heap contenente il solo elemento x;

`Find(x)` restituisce il rappresentante della max-Heap contenente x;

`Union(x,y)` unisce le max-Heap contenenti gli elementi x e y;

`Extract-max(x)` estrae il massimo della max-Heap contenente x, lo inserisce in una nuova max-Heap, e ripristina la max-Heap di partenza.

Il rappresentante dell'insieme delle chiavi contenute nella max-Heap è la chiave massima.

a. Si scriva lo pseudo-codice delle procedure `Make(x)`, `Find(x)`, `Union(x,y)` and `Extract-max(x)` implementate tramite `max-Heap`.

b. Si calcoli la complessità di una sequenza di m operazioni di `Make`, `Find`, `Union` and `Extract-max`, di cui n di tipo `Make`, nell'ipotesi che esse siano implementate come al punto a.

Esercizio 2.
Sia $G = (V, E)$ un un grafo orientato.

a. Si proponga un algoritmo efficiente per determinare se G è ciclico e se ne determini la complessità.

b. Si disegni un algoritmo che, nell'ipotesi G sia ciclico, determini $G_1 = (V_1, E_1)$ e $G_2 = (V_2, E_2)$ aciclici con le seguenti proprietà:

   - $V_1 = V_2 = V$;
   - $E_1 \cap E_2 = \emptyset$;
   - $E_1 \cup E_2 = E$.

c. Si dimostri la correttezza e si valuti la complessità dell'algoritmo di cui al punto precedente.

5.5 § Esame di ASD del 27-09-13

Esercizio 1.
Siano $(a_1, b_1), \ldots, (a_n, b_n)$ n coppie ordinate di interi. Diciamo che le coppie sono *ordinate lessicograficamente* se, per ogni $i = 1, \ldots, n-1$, valgono le seguenti condizioni:

- $a_i \leq a_{i+1}$;
- $a_i = a_{i+1} \Rightarrow b_i \leq b_{i+1}$.

Si consideri il problema di ordinare lessicograficamente le n coppie e siano h, k costanti rispetto ad n.

a. Si proponga una struttura dati adatta a gestire le n coppie, si scriva lo pseudo-codice di un algoritmo efficiente per risolvere il problema e si calcoli la complessità della procedura proposta.

b. Si supponga che valga $a_i \in [1, k]$ per ogni $i = 1, \ldots, n$. In tale ipotesi, esiste una procedura più efficiente di quella proposta al punto a.? Motivare la risposta. In caso affermativo, descrivere brevemente tale procedura e calcolarne la complessità.

c. Si supponga che valga, oltre all'ipotesi del punto b., anche $b_i \in [1, h]$ per ogni $i = 1, \ldots, n$. In tali ipotesi, esiste una procedura più efficiente di quella proposta al punto a.? Motivare la risposta. In caso affermativo, descrivere brevemente tale procedura e calcolarne la complessità.

Esercizio 2.
Sia $G = (V, E, w)$ un grafo non orientato pesato, dove $V = \{1, \ldots, n\}$ e $|E| = m$. Sia $k \in \mathbb{Z}$.

a. Si scriva lo pseudo-codice di una procedura di complessità $O(n + m)$ che restituisca, se esiste, un albero di supporto per G (non necessariamente di peso minimo) contenente solo archi di peso k.

b. Si supponga che ScanEdges(G, n, k) abbia restituito TRUE e sia v il vettore ottenuto. Cosa rappresenta v? Calcolare la complessità di ScanEdges(G, n, k).

c. Si dica se la seguente affermazione è vera o falsa, fornendo una dimostrazione o esibendo un controesempio: ScanEdges(G, n, k) restituisce TRUE se e solo se esiste un albero di supporto per G contenente solo archi di peso k. È possibile ottenere una procedura che, sfruttando la conoscenza di v, risulti più efficiente di quella proposta al punto a.? Motivare le risposte.

```
ScanEdges(G, n, k)
 1: for i ← 1 to n do
 2:    v[i] ← 0
 3: end for
 4: for {i, j} ∈ E do
 5:    if w(i, j) = k then
 6:       v[i] ← v[i] + 1
 7:       v[j] ← v[j] + 1
 8:    end if
 9: end for
10: res ← TRUE
11: i ← 1
12: while i ≤ n and res do
13:    if v[i] = 0 then
14:       res ← FALSE
15:    end if
16:    i ← i + 1
17: end while
18: RETURN res
```

5.6 § Esame di ASD del 04-02-14

Esercizio 1.
Siano S e T due RB-tree tali che $s \leq t$ per ogni $s \in S$, $t \in T$.

a. Si fornisca la definizione di RB-tree e di altezza nera.

b. Assumendo che S e T abbiano la stessa altezza nera, si scriva lo pseudo-codice di una procedura *efficiente* RBT-Merge che, dati S e T, restituisca un RB-tree costituito da tutte le chiavi di S e T.

c. Si dimostri la correttezza e si valuti la complessità dell'algoritmo del punto precedente.

Esercizio 2.
Sia A un vettore di $n+1$ interi tutti distinti tranne uno, il quale è ripetuto esattamente due volte. Sia k tale numero.

a. Si scriva lo pseudo-codice di una procedura *efficiente* che restituisca k, nell'ipotesi che il numero di elementi minori o uguali a k sia compreso nell'intervallo $\left[\lfloor \frac{n}{3} \rfloor, \lfloor \frac{n}{3} \rfloor + c\right]$, dove c è costante rispetto ad n. Si dimostri la correttezza e si calcoli la complessità della soluzione proposta.

b. Si scriva lo pseudo-codice di una procedura *efficiente* che restituisca k, nell'ipotesi che $A[i] \in \{1, \ldots, n\}$ per ogni $i \in \{1, \ldots, n+1\}$. Si dimostri la correttezza e si calcoli la complessità della soluzione proposta.

c. L'algoritmo proposto al punto b. è *in-place*? Motivare la risposta e, nel caso in cui non lo fosse, si descriva un algoritmo *in-place* che risolva lo stesso problema con la stessa complessità computazionale.

Capitolo 6

ANNO 2011-2012

6.1 § Compitino di ASD del 02-02-12

[DOMANDA DI SBARRAMENTO]
Indicare quali tra le seguenti affermazioni sono vere:

a. $\log \sqrt{n} \in \Theta(\log n)$.

b. `CountingSort` è un algoritmo di ordinamento in-place.

c. L'estrazione del massimo da una `Max-heap` ha costo $\Omega(\log n)$

Domanda 2
Si consideri l'equazione ricorsiva

$$T(n) = \begin{cases} \Theta(1) & \text{se } n = 1 \\ 2T(\frac{n}{3}) + \Theta(n^2) & \text{altrimenti} \end{cases}$$

a. si risolva l'equazione ricorsiva;

b. si illustri una procedura ricorsiva che abbia tale complessità.

Domanda 3
Si consideri un vettore A di interi di dimensione n, in cui gli elementi assumono valori interi nell'intervallo $[1, k]$, con k costante. Dato un elemento r in A, si indichi con $m(A, r)$ la *molteplicità* di r in A, ovvero il numero di occorrenze di r in A. Si proponga un algoritmo efficiente che prenda in input A e k e restituisca in output un vettore B contenente gli elementi di A ordinati rispetto alla loro molteplicità, cioè tale che valga

$$i < j \Rightarrow m(A, B[i]) \leq m(A, B[j]).$$

Si valuti la complessità e dimostri la correttezza della soluzione proposta.

Domanda 4
Sia T un BST contenente n nodi, sia k una chiave che occorre in T e sia x un nodo di T. Si indichi quali tra le seguenti affermazioni sono corrette, motivando brevemente le risposte:

a. la ricerca di k in T ha costo $O(n)$;

b. il predecessore di x si trova nel sottoalbero radicato in $left[x]$;

c. la chiave massima si trova in una foglia;

d. eseguendo una `Post-visit` di T le chiavi vengono stampate in ordine decrescente.

Domanda 5

Si considerino n intervalli $[a_1, b_1], \ldots, [a_n, b_n]$ disgiunti e ordinati, ovvero $a_1 \leq b_1 < a_2 \leq b_2 < \ldots < a_n \leq b_n$. Si vuole determinare se un numero x è contenuto in uno degli intervalli. Si proponga una struttura dati per gestire gli n intervalli e si scriva lo pseudo-codice di un algoritmo che presa in input tale struttura restituisca in output l'indice i se x appartiene all'intervallo $[a_i, b_i]$, 0 altrimenti. Si valuti la complessità e dimostri la correttezza della soluzione proposta.

Domanda 6

Sia H un vettore i cui elementi costituiscono una `Max-heap` contenente n interi. Si indichi quali tra le seguenti affermazioni sono corrette, motivando brevemente le risposte:

a. la ricerca un intero k in H ha costo $O(\log n)$;

b. H può contenere chiavi ripetute;

c. la chiave massima si trova sicuramente in una foglia;

d. la chiave massima si trova sicuramente in $H[heapsize[H]]$.

Domanda 7

Sia H una `Max-heap` contenente n elementi. Si scriva lo pseudo-codice di una versione non ricorsiva della procedura `Heapify`. Si calcoli la complessità della procedura proposta.

Domanda 8

Sia T un BST di dimensione n e sia k una chiave di T tale che $key[root[T]] < k$. Si vuole costruire un BST T' contenente tutte e sole le chiavi di T appartenenti all'intervallo $[key[root[T]], k]$. Si scriva lo pseudo-codice di un algoritmo efficiente per risolvere il problema descritto. Si valuti la complessità e dimostri la correttezza della soluzione proposta.

Domanda 9

Sia U l'insieme di tutti i vettori di bit di 100 elementi, cioè ogni $k \in U$ è un vettore di dimensione 100 a valori in $\{0, 1\}$. Si vuole memorizzare un sottoinsieme di U in una tabella hash T. Si proponga una ragionevole funzione di hash $h \colon U \to \{0, 1, \ldots, 1023\}$. Si enuncino le caratteristiche che devono essere soddisfatte dalle funzioni di hash e si dimostri che la funzione proposta le soddisfa.

6.2 § Compitino di ASD del 13-06-12

Domanda 1

Sia T il BST rappresentato in Figura 6.1.

- Si assegni un colore (rosso oppure nero) a ciascun nodo di T in modo da ottenere un red-black tree.
- Si illustrino i red-black tree risultanti dall'inserimento della chiave 5 e dalla successiva cancellazione della chiave 14. Si fornisca una breve spiegazione dei passaggi eseguiti.

Domanda 2

Sia T un red-black tree e sia x un nodo di T. Denotiamo con $T[x]$ il sottoalbero di T radicato in x, con $h[x]$ l'altezza di $T[x]$ ($h[x] = 0$ se x è una foglia) e con $bh[x]$ l'altezza nera di $T[x]$. Si indichi quale delle seguenti affermazioni sono vere:

a. Vale sicuramente $h[left[x]] < 3h[right[x]]$.

b. Se $left[x]$ e $right[x]$ sono entrambi neri, allora $h[left[x]] = h[right[x]]$.

c. Se $left[x]$ e $right[x]$ sono entrambi neri, allora $bh[left[x]] = bh[right[x]]$.

Domanda 3

Sia $G = (V, E, w)$ un grafo non orientato pesato, siano $i, j \in V$ due nodi distinti e sia P un cammino minimo da i a j. Si indichi quali tra le seguenti affermazioni sono corrette, motivando brevemente le risposte:

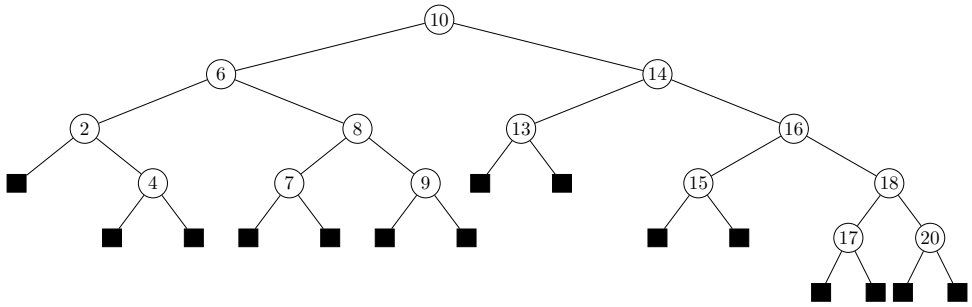


Figura 6.1: Esercizio 1

a. Se k è un nodo di P, allora il sotto-cammino di P che parte da i e termina in k è un cammino minimo da i a k.

b. Se T è un minimum spanning tree di G, allora T contiene almeno un arco di P.

c. Se T è l'albero dei cammini minimi da k, allora T contiene almeno un arco di P.

Domanda 4

Sia T un B-tree di grado $t \geq 2$ contenente almeno 2 chiavi. Si vuole determinare la seconda chiave più piccola di T, ovvero la chiave che finirebbe al secondo posto se si ordinassero in ordine crescente tutte le chiavi di T.

a. Descrivere tramite pseudo-codice un algoritmo efficiente per risolvere il problema proposto.

b. Dimostrare la correttezza e determinare la complessità dell'algoritmo proposto.

Domanda 5

Partendo dal B-tree di grado t = 3 rappresentato in Figura 6.2, illustrare (graficamente e con una breve spiegazione dei passaggi eseguiti) i B-tree risultanti dall'inserimento della chiave 32 e dalla successiva cancellazione della chiave 23. Le operazioni vanno eseguite nell'ordine proposto.

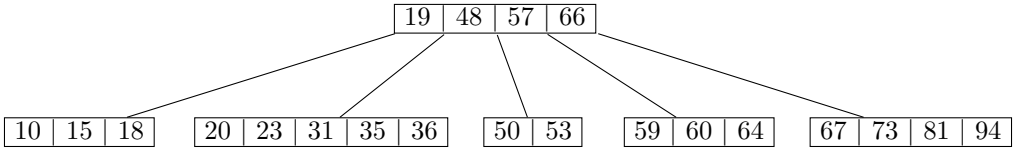


Figura 6.2: Esercizio 5

Domanda 6

Si illustri la famiglia di insiemi disgiunti che si ottiene eseguendo la seguente sequenza di operazioni sugli 8 insiemi singoletto contenenti gli elementi $1, 2, \ldots, 8$: Union(1,5); Union(2,8); Union(3,6); Union(4,7); Union(3,5); Union(1,2); Find(3). Si utilizzi la rappresentazione con alberi con le euristiche di union by rank e path compression. Si assuma che, qualora vengano uniti due alberi aventi lo stesso rango, l'albero rappresentato dall'elemento minore venga fatto puntare all'albero rappresentato dall'elemento maggiore.

Domanda 7 Si consideri il grafo G non orientato pesato rappresentato in Figura 6.3. Si illustri l'output di MST-Prim partendo dal nodo A.

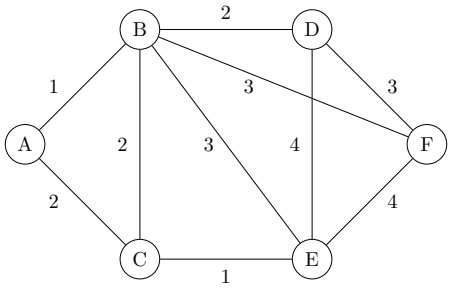


Figura 6.3: Esercizio 7

Domanda 8

Si consideri il seguente algoritmo che prende in input un red-black tree T e un nodo x.

```
BlackPath(T, x)
1: if x =nil then
2:     return true
3: end if
4: if color(x) =black then
5:     return BlackPath(T,left(x)) ∨ BlackPath(T,right(x))
6: end if
7: return false
```

a. L'algoritmo termina sempre? In caso affermativo, se ne calcoli la complessità.

b. Si calcoli l'output di BlackPath($T, root[T]$) sul red-black tree T rappresentato in Figura 6.1 come da voi colorato prima delle operazioni di inserimento e cancellazione.

Domanda 9

Dato un grafo orientato $G = (V, E)$, definiamo *distanza* fra due nodi di V la lunghezza del cammino minimo che li collega. Dato un nodo $s \in V$ si vuole determinare un cammino minimo che parte da s e termina in un nodo q a distanza massima da s.

a. Descrivere tramite pseudo-codice un algoritmo efficiente per risolvere il problema proposto. Dimostrarne la correttezza e determinarne la complessità.

b. Se P è il cammino determinato dall'algoritmo da voi proposto e P ha lunghezza n, può esistere in G un cammino che parte da s ed ha lunghezza $m > n$? Motivare brevemente la risposta.

6.3 § Esame di ASD del 19-07-12

Esercizio 1

Si consideri la struttura dati `RB*-tree` definita nel modo seguente: dato un `RB-tree` si aggiunga un campo `size[i]` contenente il numero di nodi del sottoalbero radicato nel nodo `i`.

Siano dati un `RB*-tree` T con n nodi e un intero k tale che $1 \leq k \leq n$. Si consideri il problema di determinare la k-esima chiave più piccola di T.

a. Si scriva lo pseudo-codice di un algoritmo efficiente per risolvere il problema descritto. Se ne calcoli la complessità e se ne dimostri la correttezza.

b. **(Facoltativo)** Si scriva lo pseudo-codice di un algoritmo efficiente per risolvere lo stesso problema, nell'ipotesi che il valore di `size[i]` non sia noto a priori (cioè T è un `RB-tree` normale). Se ne calcoli la complessità e se ne dimostri la correttezza.

Esercizio 2.
Sia $G = (V, E)$ un grafo orientato, con $V = \{1, \ldots, n\}$. Definiamo $G_2 = (V, E_2)$ come il grafo che ha lo stesso insieme di nodi di G e i cui archi sono definiti come segue:

$$(i, j) \in E_2 \Leftrightarrow \text{esiste in } G \text{ un cammino da } i \text{ a } j \text{ di lunghezza 2.}$$

a. Si scriva lo pseudo-codice di una procedura che, preso in input G, restituisca in output la matrice di adiacenza di G_2, ovvero M di dimensione $n \times n$ tale che

$$M[i][j] = \begin{cases} 1 & \text{se esiste in } G \text{ un cammino da } i \text{ a } j \text{ di lunghezza 2} \\ 0 & \text{altrimenti} \end{cases}$$

per ogni $i, j = 1, \ldots, n$.

b. Si valuti la complessità e si dimostri la correttezza della procedura proposta.

6.4 § Esame di ASD del 06-09-12

Esercizio 1.
Sia A un vettore di interi distinti di dimensione n. Definiamo *sequenza alternante in A di lunghezza m*, con $3 \leq m \leq n$, una sequenza di indici consecutivi $i, i+1, \ldots, i+m-1$ tale che

$$A[i] < A[i+1] > A[i+2] < A[i+3] > \ldots$$

oppure

$$A[i] > A[i+1] < A[i+2] > A[i+3] < \ldots$$

a. Scrivere lo pseudo-codice di un algoritmo che, presi in input A ed n, determini una sequenza alternante in A di lunghezza massima. Si calcoli la complessità e si dimostri la correttezza dell'algoritmo proposto.

b. Si supponga che A contenga esattamente una sequenza alternante e che tale sequenza abbia lunghezza k, con k costante rispetto ad n. Si proponga, tramite pseudo-codice, un algoritmo efficiente per ordinare il vettore A. Si calcoli la complessità e si dimostri la correttezza dell'algoritmo proposto.

Esercizio 2.
Sia $G = (V, E, w)$ un grafo non orientato, pesato e connesso. Siano $i, j \in V$ due nodi.

a. Si scriva lo pseudo-codice di un algoritmo efficiente che determini un cammino di lunghezza minima da i a j (numero minimo di archi attraversati). Si calcoli la complessità e si dimostri la correttezza dell'algoritmo proposto.

b. Si scriva lo pseudo-codice di un algoritmo efficiente che determini un cammino di peso minimo da i a j. Si calcoli la complessità e si dimostri la correttezza dell'algoritmo proposto.

Si dica se le seguenti affermazioni sono vere o false, dandone una dimostrazione o fornendo un controesempio:

c. Se p è un cammino di lunghezza minima, allora p è di peso minimo.

d. Se p è un cammino di peso minimo, allora p è di lunghezza minima.

6.5 § Esame di ASD del 20-09-12

Esercizio 1.

Sia A un vettore di interi di dimensione n, tale che $1 \leq A[i] \leq m$ per ogni $1 \leq i \leq n$. Sia B un vettore di lunghezza m tale che in $B[k]$ è memorizzato il numero di elementi di A con valore $\leq k$ per ogni $1 \leq k \leq m$.

a. Scrivere lo pseudo-codice di un algoritmo *efficiente* che, dati in input A, n, B ed m, ordini il vettore A. Calcolarne, inoltre, la complessità e dimostrarne la correttezza.

b. L'algoritmo proposto al punto precedente è stabile? Nel caso lo fosse, motivare la risposta. In caso contrario, scrivere lo pseudo-codice di un algoritmo stabile che risolva il problema al punto a. e motivare il fatto che sia stabile.

Esercizio 2.

Un *albero* è un grafo $T = (V_T, E_T)$ non orientato, connesso e aciclico. Un *albero radicato* è una coppia $\langle T, r \rangle$ dove $T = (V_T, E_T)$ è un albero ed $r \in V_T$ è un nodo di T, chiamato *radice*.

Dato un albero radicato $\langle T, r \rangle$ con $V_T = \{0, 1, \ldots, n-1\}$, siano d e h i vettori di dimensione n definiti, per ogni $i = 0, 1, \ldots, n-1$, nel modo seguente:

$d[i]$: distanza (numero di archi) del nodo i dalla radice r

$h[i]$: altezza del sotto-albero di $\langle T, r \rangle$ radicato in i.

a. Fornire lo pseudo-codice di un algoritmo *efficiente* e ricorsivo che calcoli d e h. Calcolarne la complessità e dimostrarne la correttezza.

b. Mostrare l'output della procedura proposta al punto a. sull'albero di Figura 6.4 (indicare i valori $d[i]$ e $h[i]$ per ogni nodo i).

c. È possibile modificare l'algoritmo di cui al punto a. per determinare anche il *diametro* (la massima distanza fra due nodi) di T?

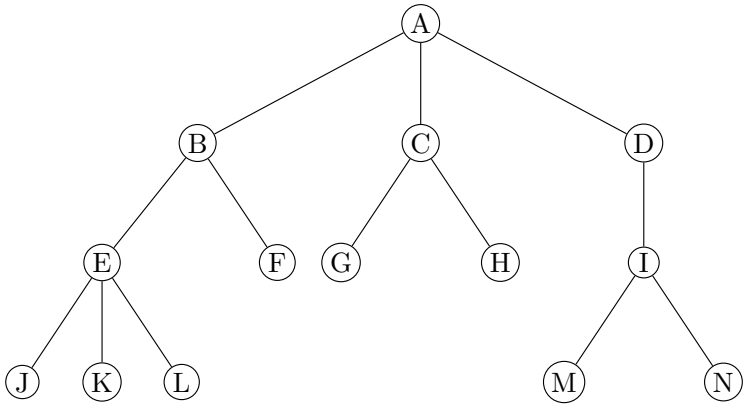


Figura 6.4: Esercizio 2.

6.6 § Esame di ASD del 21-01-13

Esercizio 1a.

Sia C un vettore di interi *non negativi*, di dimensione n, soddisfacente le seguenti condizioni:

- $C_{\max} \in O(n^2)$, dove $C_{\max}$ indica il massimo di C;
- per ogni $i, j = 1, \ldots, n$ tali che $i \neq j$, vale $|C[i] - C[j]| \in \Omega(n)$.

Si supponga di avere a disposizione un vettore D di dimensione n, tale che $D[i]$ è una lista per ogni $i = 1, \ldots, n$.

a. Si proponga un metodo per distribuire *uniformemente* gli elementi di C nella struttura D.

b. Si scriva lo pseudo-codice di un algoritmo *efficiente* che prenda in input D e restituisca il vettore C ordinato.

c. Si calcoli la complessità e si dimostri la correttezza della procedura proposta al punto b.

Esercizio 1b.

Sia C un vettore di interi di dimensione n. Si supponga che C sia costituito da k sottovettori ordinati, ovvero esistono k posizioni $i_1, \ldots, i_k$ in C (non note) tali che $C[i_j, \ldots, i_{j+1} - 1]$ è ordinato, per ogni $j = 1, \ldots, k$ (si assuma $i_1 = 1$ e $i_{k+1} = n + 1$).

a. Si scriva lo pseudo-codice di un algoritmo *efficiente* per ordinare C.

b. Si calcoli la complessità e si dimostri la correttezza della procedura proposta al punto a.

Esercizio 2.

Sia $G = (V, E, w)$ un grafo non orientato pesato e sia $k \in \mathbb{Z}$.

a. Si fornisca lo pseudo-codice di un algoritmo *efficiente* che prenda in input G, w e k e restituisca 1 se esiste un MST T di G contenente solo archi e di peso $w(e) \leq k$, 0 altrimenti.

b. Si fornisca lo pseudo-codice di un algoritmo *efficiente* che prenda in input G, w e k e determini, se esiste, un MST T di G contenente solo archi e di peso $w(e) \leq k$.

c. Si calcoli la complessità e si dimostri la correttezza delle procedure proposte.

Capitolo 7

ANNO 2010-2011

7.1 § Compitino di ASD del 28-01-11

[DOMANDA DI SBARRAMENTO]

Indicare quali tra le seguenti affermazioni sono vere:

a. Un BST con n nodi ha altezza $O(n)$;

b. `InsertionSort` ha complessità $\Theta(n \log n)$;

c. `MergeSort` ha complessità $\Omega(n)$;

Domanda 2

Si consideri l'equazione ricorsiva $T(n) = T(\frac{n}{3}) + O(\log n)$.

a. Si risolva l'equazione ricorsiva;

b. Si illustri una procedura ricorsiva (se possibile soluzione di un problema reale) che abbia tale complessità.

Domanda 3

Si consideri la struttura dati `DoubleHeap` formata da una min-Heap H_{min} e da una max-Heap H_{max} contenenti gli stessi valori. `DoubleHeap` supporta le seguenti operazioni:

- `Insert(x)`: l'elemento `x` viene inserito all'interno di `DoubleHeap`;
- `ExtractMin()`: estrae e restituisce l'elemento di valore minimo memorizzato in `DoubleHeap`;
- `ExtractMax()`: estrae e restituisce l'elemento di valore massimo memorizzato in `DoubleHeap`;
- `DescendingPrint()`: stampa tutti i valori memorizzati in `DoubleHeap` in ordine decrescente.

Si risponda alle seguenti domande, motivando le risposte:

a. Si descriva tramite pseudo-codice la procedura `Insert(x)`.

b. Sia H una `DoubleHeap` in cui sono stati effettuati n `Insert(x)`. La procedura `ExtractMin()` ha complessità $O(1)$?

c. `DescendingPrint()` può essere implementata con complessità $\Theta(n \log n)$?

Domanda 4

Si M una matrice $n \times n$ contenente numeri interi. Per ogni riga i viene fornita la coppia $\{min_i, max_i\}$, tale che ogni elemento x della riga i soddisfa $x \in [min_i, max_i]$. Gli n intervalli associati alle righe di M sono tutti disgiunti, cioè la loro intersezione è vuota ($\forall i \in \{1..n\}\ \forall j \in \{1..n\} \setminus \{i\} :\ [min_i, max_i] \cap [min_j, max_j] = \emptyset$). La matrice M è *ordinata* se il vettore risultante dalla concatenazione di tutte le righe dalla 1 alla n è ordinato.

a. Si proponga un algoritmo per ordinare la matrice M. Si dimostri la correttezza dell'algoritmo proposto e si valuti la sua complessità.

b. Si proponga un algoritmo per ordinare la matrice M nel caso in cui per ogni i si abbia $|min_i|, |max_i| \leq kn$. Si dimostri la correttezza dell'algoritmo proposto e si valuti la sua complessità.

c. Si proponga un algoritmo per determinare l'elemento mediano di M. Si dimostri la correttezza dell'algoritmo proposto e si valuti la sua complessità.

Domanda 5

Sia H una max-Heap contenente n chiavi intere, priva di ripetizioni. Indicare quali tra le seguenti affermazioni sono vere, motivando brevemente le risposte.

a. `BuildHeap(`H`)` ha complessità $\Omega(n)$;

b. Posso sempre ordinare H con un algoritmo avente complessità $O(n)$;

c. Posso sempre trovare il mediano di H in tempo $O(n)$;

d. Posso sempre trovare il mediano di H in tempo $O(\log n)$.

Domanda 6

A partire dal BST *completo* contenente le chiavi $10, 20, 30, 40, 50, 60, 70$, si inseriscano nell'ordine le chiavi $15, 35, 12$ e successivamente si cancellino nell'ordine le chiavi $60, 10, 40$. Si disegnino il BST di partenza ed i BST risultanti dopo ogni inserimento e cancellazione.

Domanda 7

Si consideri un BST *completo* e si proponga lo pseudocodice di un algoritmo che, dato in input un nodo x, restituisca il nodo y la cui chiave è minima tra quelle che soddisfano $key[y] \in (key[x], key[x]+8]$, se tale y esitste, mentre restituisce NIL altrimenti.

Si determini la complessità e si discuta la correttezza dell'algoritmo proposto.

Domanda 8

Si consideri la struttura dati `Circle` è costituita da $n \geq 0$ nodi ed un puntatore `Start` che punta a `NIL` se ci sono 0 nodi, altrimenti ad uno qualsiasi dei nodi eletto rappresentante. Ogni nodo ha un campo chiave e un puntatore `next` al prossimo nodo del cerchio e ogni nodo è puntato da un solo puntatore `next`. L'ultimo nodo della struttura punta con `next` al nodo rappresentante.

Si risponda vero o falso alle seguenti domande, motivando brevemente le risposte:

a. Dato un nodo x, determinare se x è l'ultimo del cerchio ha costo $O(n)$.

b. Inserire un nodo come nuovo rappresentante ha costo $O(1)$.

c. L'algoritmo ottimo per inserire un nodo come ultimo ha costo $O(n)$.

Domanda 9

Si risponda vero o falso alle seguenti domande, motivando brevemente le risposte:

a. la scelta dell'elemento pivot nella `Partition` non influenza la complessità asintotica di `Quick-Sort`.

b. `Partition` può restituire valori diversi se le vengono forniti elementi pivot diversi.

c. Data una max-Heap incrementare il valore di una chiave di un elemento in posizione data ha costo $O(n)$.

d. `Counting-Sort` è un algoritmo che ordina un qualunque array di n interi in tempo $O(n)$.

Domanda 10

Il signor Alano Turingo ha scritto una serie di programmi e vuole fare un po' di ordine dividendoli in 100 sottocartelle che numera da 0 a 99. Estrae il numero in base 2 codificante nel suo computer il testo dei singoli programmi (ovviamente ad ogni programma viene associato un numero binario enorme, scomodo da maneggiare). Vorrebbe quindi mappare ogni programma in una sottocartella da 0 a 99 utilizzando una ragionevole funzione di hash.

a. Lo potreste aiutare fornerndogliene una?

b. Per il problema descritto si consiglia l'utilizzo di una tabella hash basata su chaining o su open addressing? Motivare brevemente la risposta.

7.2 § Compitino di ASD del 14-06-11

[DOMANDA DI SBARRAMENTO]

Indicare quali tra le seguenti affermazioni sono vere:

a. Un RB-tree contenente n nodi ha altezza $\Omega(n)$.

b. Un grafo aciclico è un albero.

c. Un albero è un grafo aciclico.

Domanda 2

Sia $G = (V, E, w)$ un grafo connesso, pesato e non orientato. Indicare se le seguenti affermazioni sono vere o false, motivando la risposta.

a. Se T è un albero minimo di copertura (di supporto) di G e T' è un albero di copertura di G, allora $w(T) < w(T')$.

c. Se w è iniettiva, T è un albero minimo di copertura ed e è un arco di T, può esistere T' albero minimo di copertura che non contiene e.

Domanda 3

Sia $G = (V, E, w)$ un grafo connesso e pesato. Indicare se le seguenti affermazioni sono vere o false, motivando la risposta.

a. Se G è non orientato, T è un albero dei cammini minimi da un nodo s e T' è un albero dei cammini minimi da s', allora $w(T) = w(T')$.

b. Se G è orientato, s è raggiungibile da t e T è un albero dei cammini minimi da s, allora in T c'è un cammino da t ad s.

Domanda 4

Dato un RB-tree T ci si propone di incrementarne l'altezza nera trasformado T in un RB-tree T' procedendo ricorsivamente come segue. Si parte dalla radice di T. Arrivati a considerare un nodo x: (1) se x ha due figli rossi, allora si colorano i figli di x di nero; (2) si procede ricorsivamente su entrambi i figli di x.

a. Scrivere lo pseudo-codice della procedura sopra descritta e valutarne la complessità.

b. Dimostrare o confutare la correttezza della procedura in relazione alla seguente proprietà: T' è un RB-tree ed ha altezza nera maggiore o uguale dell'altezza nera di T.

Domanda 5

Sia $G = (V, E)$ e sia F una foresta di visita DFS di G. Sia u un nodo di G e sia T_i l'albero di F che contiene u, definiamo la profondità di u in F ($depth_F[u]$) come la distanza di u dalla radice di T_i. Definiamo quindi la profondità di F ($Depth(F)$) come la somma delle profondità di tutti i nodi di G (ovvero, $Depth(F) = \sum_{u \in V} depth_F[u]$).

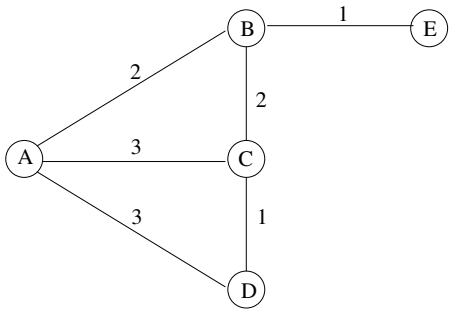


Figura 7.1: Esercizio 7.

a. Modificare opportunamente lo pseudo-codice di DFS (e DFS_visit) in modo che al termine la procedura restituisca oltre ad una DFS-foresta F anche la profondità di F.

b. Valutare la complessità e dimostrare la correttezza della procedura proposta.

Nota: non è richiesta la gestione dei tempi di inizio/fine visita.

Domanda 6

Si consideri il B-tree T di grado 2 contenente le chiavi $10, 20, 30, 40, 50, 60, 70$ ed in cui ogni nodo contiene il numero minimo di chiavi.

a. Rappresentare il B-tree T.

b. Inserire in T le chiavi $1, 2, 3$.

c. Cancellare da T la chiave 70.

Domanda 7

Si consideri il grafo non orientato e pesato rappresentato in Figura 1.

a. Si illustri un albero che si ottiene applicando l'algoritmo di Prim a partire dal nodo A. Si descrivano anche i passi intermedi dell'esecuzione.

b. Si illustri un albero che si ottiene applicando l'algoritmo di Dijkstra a partire dal nodo A. Si descrivano anche i passi intermedi dell'esecuzione.

Domanda 8

Si considerino i due alberi illustrati in Figura 2, che memorizzano due insiemi disgiunti (i numeri nei riquadri indicano i ranghi).

a. I due alberi possono essere stati ottenuti mediante una sequenza di operazioni di tipo *Make*, *Union*, *Find*, con euristiche *union-by-rank* e *path-compression* a partire da alberi contenenti una sola chiave? In caso affermativo si produca una tale sequenza di operazioni. In caso negativo si spieghi perché ciò non sarebbe possibile.

b. Si illustri l'effetto dell'esecuzione dell'operazione $Union(4, 8)$, utilizzando le euristiche *union-by-rank* e *path-compression*. Nel caso di unione di alberi aventi lo stesso rango l'albero avente rappresentante con chiave maggiore viene fatto puntare a quello avente rappresentante con chiave minore.

c. Si consideri l'albero risultante dopo l'unione di cui al punto precedente. Si illustri l'effetto dell'esecuzione dell'operazione $Find(5)$, nelle ipotesi delineate al punto precedente.

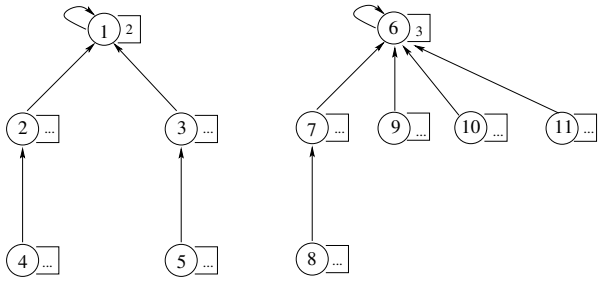


Figura 7.2: Esercizio 8.

7.3 § Esame di ASD del 14-07-11

Esercizio 1

Sia A un vettore in cui ogni elemento contiene due campi: $A[i].value$ contiene un numero intero ed $A[i].color$ contiene un colore (BIANCO o NERO). Gli elementi di A sono ordinati in ordine crescente rispetto al campo *value*.

1-a Si consideri il problema di ordinare gli elementi di A rispetto al campo *color* secondo l'ordinamento BIANCO<NERO, facendo in modo che gli elementi dello stesso colore siano ordinati rispetto al campo *value*. Si scriva lo pseudocodice di un algoritmo efficiente per risolvere il problema proposto. Si valuti la complessità e si dimostri la correttezza.

1-b Si consideri il problema di ordinare gli elementi di A rispetto al campo *value*, facendo in modo che gli elementi che hanno stesso *value* siano ordinati rispetto al campo *color* (sempre con la convenzione BIANCO<NERO). Si scriva lo pseudocodice di un algoritmo efficiente per risolvere il problema proposto. Si valuti la complessità e si dimostri la correttezza.

Esercizio 2 Sia $G = (V, E)$ un grafo non orientato connesso. Un orientamento di G è un grafo orientato G' ottenuto rimpiazzando ogni arco $\{u, v\}$ di G con (u, v) oppure (v, u).

2-a Si consideri il problema di determinare un orientamento aciclico di G. Si scriva lo pseudocodice di un algoritmo efficiente per risolvere il problema proposto. Se ne valuti la complessità e se ne dimostri la correttezza.

2-b Si consideri il problema di determinare un orientamento aciclico di G in cui esiste un solo nodo privo di archi entranti. Si scriva lo pseudocodice di un algoritmo efficiente per risolvere il problema proposto. Se ne valuti la complessità e se ne dimostri la correttezza.

7.4 § Esame di ASD del 06-09-11

Esercizio 1

Sia A un vettore di interi di lunghezza $2n + 1$, con $n \geq 0$. In A c'è un intero x che occorre una sola volta, mentre tutti gli altri elementi sono ripetuti esattamente 2 volte. Dato in input A si vuole determinare x.

1-a Si proponga un algoritmo *in-place ed* efficiente per risolvere il problema. Si scriva lo pseudocodice dell'algoritmo.

Si valuti la complessità e si dimostri la correttezza della soluzione proposta.

1-b Si supponga di avere a disposizione una funzione di hash

$$h_A : \{A[i] \mid i = 1, \ldots 2n + 1\} \longrightarrow \{0, 1, \ldots, m - 1\}$$

tale che per ogni $j \in \{0, 1, \ldots, m - 1\}$ vale che $f^{-1}(j)$ contiene al più 4 interi distinti. Si proponga un algoritmo efficiente per risolvere il problema utilizzando una tabella di hash T e la funzione di hash h_A.

Si scriva lo pseudocodice dell'algoritmo. Si valuti la complessità e si dimostri la correttezza della soluzione proposta.

Esercizio 2

Sia $G = (V, E)$ un grafo non orientato. Diciamo che G è un *caterpillar* (un *bruco*) se è connesso, aciclico e per ogni nodo u di G l'insieme dei nodi adiacenti ad u contiene al più due nodi non adiacenti solo ad u(tutti gli altri nodi adiacenti ad u, sono adiacenti *solo* ad u).

Si consideri il problema di determinare se G è un caterpillar.

2-a Si disegnino due caterpillar distinti con 17 nodi.

2-b Si proponga un algoritmo per risolvere il problema proposto.

Si scriva lo pseudocodice dell'algoritmo. Si valuti la complessità e si dimostri la correttezza della soluzione proposta.

7.5 § Esame di ASD del 20-09-11

Esercizio 1

Sia A un vettore di numeri naturali di dimensione n. Un sottointervallo ordinato è una coppia di puntatori $< i, j >$ tali che $A[i] \leq A[i+1] \leq ... \leq A[j]$. La dimensione di un intervallo $< i, j >$ è $L_{ij} = j - i + 1$. Un sottointervallo ordinato $< i, j >$ è massimale se non esiste nessun altro sottointervallo $< k, l >$ tale che $L_{kl} > L_{ij}$.

1-a Si fornisca lo pseudo-codice per identificare il sottointervallo ordinato massimale di A.

Si valuti la complessità e si dimostri la correttezza della soluzione proposta.

1-b Si proponga un algoritmo per odinare A sapendo che è composto da esattamente 4 sottointervalli ordinati.

Si valuti la complessità e si dimostri la correttezza della soluzione proposta.

1-c Facoltativo Si generalizzi il punto precedente al caso in cui il vettore A sia composto da esattamente k sottointervalli ordinati, con k costante.

Si valuti la complessità e si dimostri la correttezza della soluzione proposta.

Esercizio 2

Sia $G = (V, E, w)$ un grafo non orientato, pesato e connesso.

2-a Si fornisca lo pseudo-codice di un algoritmo che determina un *minimum spanning tree (MST)* di G.

2-b Si consideri l'algoritmo presentato al punto precedente e si dimostri o si refuti la seguente affermazione: due esecuzioni del precedente algoritmo possono produrre MST con archi di peso massimo distinti.

7.6 § Esame di ASD del 24-01-12

Esercizio 1 Siano H una max-heap di dimensione h e B un binary search tree di dimensione b, contenenti complessivamente $h + b$ interi distinti.

Si propongano algoritmi efficienti per risolvere i due seguenti problemi:

1-a la costruzione di un binary search tree contente gli $h + b$ elementi di H e di B;

1-b la costruzione di una max-heap contente gli $h + b$ elementi di H e di B,

nelle seguenti ipotesi:

1. $h = b = n$,
2. $h = b = \log(n)$,
3. $h = \log(n)$ e $b = n$,
4. $b = \log(n)$ e $h = n$.

Facoltativo. Si provi un limite inferiore alla complessità degli algoritmi per i problemi proposti, relativamente all'ipotesi (1).

Esercizio 2 Sia $G = (V, E)$ un grafo non orientato. Si proponga un algoritmo per determinare un insieme di archi di cardinalità minima che renda G connesso. Si provi la correttezza e si valuti la complessità dell'algoritmo proposto.

Capitolo 8

ANNO 2009-2010

8.1 § Compitino di ASD del 28-01-10

[DOMANDA DI SBARRAMENTO]
Indicare quali tra le seguenti affermazioni sono vere:

a. `InsertionSort` ha costo $\Omega(n)$

b. `QuickSort` ha complessità $\Theta(n \log n)$

c. `MergeSort` è un algoritmo di ordinamento in place

Domanda 2
Si consideri l'equazione ricorsiva $T(n) = 3T(\frac{n}{2}) + n \log n$.

a. Si risolva l'equazione ricorsiva;

b. Si illustri una procedura ricorsiva che abbia tale complessità.

Domanda 3
Il programmatore Pippo deve memorizzare dei numeri in base 10 molto grandi, composti da n cifre. Decide quindi di memorizzare un numero attraverso un array di dimensione n, memorizzando in posizione i-esima la cifra in posizione i partendo da sinistra (in posizione 0 verrà memorizzata la cifra più significativa, in posizione $n-1$ la cifra meno significativa). Pippo memorizza in questa maniera n^2 numeri. Pippo deve risolvere i seguenti problemi:

a. Fornire uno pseudocodice efficiente per ordinare questi numeri

b. Dimostrare la correttezza e la complessità della procedura

Si aiuti Pippo a risolvere questi due problemi.

Domanda 4
Si vogliono ordinare n numeri interi. Uno studente che ha appena frequentato il corso di ASD decide di inserirli uno alla volta all'interno di un BST, e di effettuare poi una visita `In-Order` del BST ottenuto. Si indichi quali delle seguenti affermazioni sono vere:

a. La procedura proposta ha costo $\Theta(n \log n)$

b. Esistono casi in cui la procedura non ordina correttamente

c. Il BST che si ottiene è sempre sbilanciato

d. L'algoritmo ha la stessa complessità del caso medio di `InsertionSort`

Domanda 5

Sia T un BST. Effettuando una visita `Pre-Order` di T l'output è il seguente: 20, 10, 40, 27, 55, 70. Si ricostruisca T e, successivamente, si inseriscano le chiavi $\{5, 42, 56, 23\}$. Infine, dall'albero così ottenuto, si cancellino le chiavi $\{42, 40, 20\}$. Per ogni inserimento e cancellazione si disegni il BST risultante.

Domanda 6

Si consideri una struttura dati *Set* che rappresenta un insieme di interi e che supporti le seguenti operazioni:

- `insert(S,x)` : restituisce l'insieme $S' = S \cup \{x\}$
- `member(S,x)` : restituisce *vero* se x appartiene ad S, $falso$ altrimenti
- `findMax(S)` : restituisce il massimo di S
- `deleteMax(S)` : restituisce l'insieme S privato del massimo, cioè $S' = S \setminus \{max(S)\}$

Sia S una istanza di *Set* contenente n interi. Dire quale delle seguenti affermazioni è corretta, motivando brevemente la risposta:

a. Se *Set* viene implementata con una linked-list ed `insert(S,x)` inserisce l'elemento x in testa alla lista, `findMax(S)` ha costo $\Theta(n)$

b. Se *Set* viene implementata attraverso una linked-list i cui elementi sono ordinati, la procedura `insert(S,x)` ha costo $\Theta(n)$

c. Se *Set* viene implementata attraverso una max-heap `deleteMax(S)` ha costo $\Theta(1)$

d. *Set* NON può essere implementata per mezzo di una tabella di hash

e. Se *Set* viene implementata attraverso una max-heap allora `member(S,x)` ha costo $O(n)$

Domanda 7

Si consideri la seguente struttura $TernarySearchTree$ (TST):

- Ogni nodo x ha due chiavi $x.k_1$ e $x.k_2$ tali che $x.k_1 \leq x.k_2$
- Due nodi distinti non possono avere alcuna chiave uguale
- Ogni nodo x può avere al più tre figli
- Sia x un nodo, il sottoalbero radicato nel figlio sinistro sarà indicato da $T_s[x]$, il sottoalbero radicato nel figlio centrale sarà indicato con $T_c[x]$, mentre il sottoalbero radicato nel figlio di destra sarà indicato con $T_d[x]$
- Sia x un nodo avente chiavi $x.k_1 \leq x.k_2$ allora $\forall\ y \in T_s[x]\ y.k_1 \leq y.k_2 < x.k_1$, $\forall\ y \in T_c[x]\ x.k_1 < y.k_1 \leq y.k_2 < x.k_2$ e $\forall\ y \in T_d[x]\ x.k_2 < y.k_1 \leq y.k_2$

Dati due valori v_1 e v_2 tali che $v_1 \leq v_2$ diciamo che l'intervallo $[v_1, v_2]$ occorre in un TST T, se esiste un nodo x tale che $x.k_1 = v_1$ e $x.k_2 = v_2$. Dato un TST T, il massimo è il nodo x che contiene il valore $x.k_2$ più alto. Un TST T contenente n nodi è completo se ogni nodo interno ha esattamente 3 figli e ha altezza $\log_3 n$. Sia T un TST completo contenente n nodi, si indichino quali tra le seguenti affermazioni sono corrette, motivando brevemente le risposte:

a. T ha altezza $O(\log n)$

b. Dato un intervallo $[v_1, v_2]$ con $v_1 \leq v_2$ verificare se l'intervallo appartiene o meno a T ha costo $\Theta(n)$

c. Il nodo x composto dalle chiavi $x.k_1$ e $x.k_2$ tali che $x.k_1 = x.k_2$ è una foglia

d. Il nodo massimo si trova nel sottoalbero T_c della radice

Domanda 8

Si consideri una max-heap A. Si indichino quali tra le seguenti affermazioni sono corrette, motivando brevemente le risposte:

a. l'elemento minimo si trova in una foglia;

b. l'elemento minimo si trova nell'ultima posizione dell'array A;

c. l'elemento mediano si può trovare in una foglia;

d. `Heapify(A,i)` viene eseguita sulla heap in tempo $O(1)$ per qualsiasi indice i;

Domanda 9

Data una max-heap A, di dimensione n, si vogliono ordinare i suoi elementi tramite la seguente procedura:

- si copino gli elementi di A in un vettore ausiliario B
- si costruisca la min-heap B
- si iteri $\lfloor n/2 \rfloor$ volte sulle due heap estraendo il massimo da A e il minimo da B (attraverso le procedure `Heap-Extract-Max` e `Heap-Extract-Min`, rispettivamente) e riportandoli in un terzo vettore C
- si gestisca l'eventuale caso con n dispari

Si richiede di:

a. fornire uno pseudocodice efficiente per ordinare i valori in input

b. dimostrare la correttezza e la complessità della procedura

Domanda 10

Il professor Mike Morgan, esperto genetista, e la sua giovane assistente Frida Catton vogliono preparare un nuovo esperimento di sequenziamento riguardante stringhe di DNA lunghe esattamente 100 caratteri (l'universo U). Una stringa di DNA è una stringa sull'alfabeto $\Sigma = \{\mathbf{A},\mathbf{C},\mathbf{G},\mathbf{T}\}$. Per velocizzare il proprio lavoro i due biologi vogliono combinare stringhe diverse di DNA in uno stesso esperimento che comprenda al più 256 stringhe ($256 = 4^4$). Chiedono aiuto al loro bioinformatico pasticcione C.D.F. il quale suggerisce alcune funzioni di hash. Si dica quali delle seguenti funzioni suggerite su una stringa S possono essere considerate ragionevoli funzioni di hash (considerando che una **A** viene decodificata con uno 0, una **C** con un 1, una **G** con un 2 e una **T** con un 3 tramite la funzione `DEC` e che S(i) rappresenti l'i-esimo carattere di S):

- $h_1(S) =$[`DEC`(S(1))+`DEC`(S(2))] *mod* 256;
- $h_2(S) =$[`DEC`(S(1))+`DEC`(S(2))+`DEC`(S(3))+1] *mod* 256;
- $h_3(S) =$[`DEC`(S(1))$*4^3$+`DEC`(S(2))$*4^2$+`DEC`(S(3))$*4^1$+`DEC`(S(4))$*4^0$] *mod* 256;

8.2 § Compitino di ASD del 23-06-10

[DOMANDA DI SBARRAMENTO]

Indicare quali tra le seguenti affermazioni sono vere:

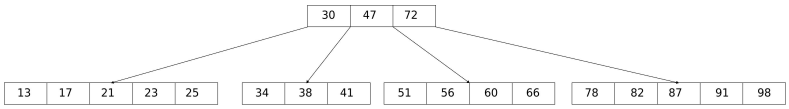
a. `BFS` ha costo $\mathcal{O}(|V|^2)$

b. Le operazioni di `MAKE, UNION` e `FIND` *non* possono essere implementate tramite liste

c. La procedura `Relax(u,v,w)` usata nell'algoritmo di Dijkstra ha costo $\mathcal{O}(1)$

Domanda 2

Si illustri il B-tree risultante dall'inserimento della chiave 92 nel B-tree di grado 3 illustrato nella figura sottostante. Si illustri quindi il B-tree che si ottiene in seguito alla cancellazione della chiave 47.



Domanda 3

Una implementazione degli Insiemi Disgiunti deve supportate le operazioni di $MAKE(x)$ (crea un insieme che contiene solamente x), $FIND(x)$ (restituisce il rappresentante dell'insieme che contiene x), $UNION(x, y)$ (unisce gli insiemi aventi come rappresentanti x ed y). Con opportuni cambiamenti alla struttura dati vista in classe, gli RB-tree possono essere utilizzati per rappresentare insiemi disgiunti:

a. Si illustri una variante degli RB-tree che esegua l'operazione $FIND(x)$

b. Utilizzando l'implementazione descritta al punto precedente si illustri l'algoritmo per effettuare $UNION(x, y)$ e se ne indichi la complessità

c. Qual è il costo di m operazioni di tipo $MAKE$ e di n operazioni di tipo $FIND/UNION$ utilizzando l'implementazione illustrata ai punti a e b?

Domanda 4

Si dica se le seguenti affermazioni sono vere o false, dimostrando quelle vere e fornendo un controsempio per quelle false:

a. Se durante una visita DFS di G non vengono prodotti *back edges* allora G è aciclico.

b. Per ogni grafo G e per ogni funzione di peso $E- > \mathcal{R}$ l'algoritmo di Dijkstra risolve il *Single Source Shortest Path Problem.*

c. Sia G un grafo con $|V|$ nodi ed $|E|$ archi tutti di peso 1. Allora il MST restituito da MST-Prim è identico al MST restituito da MST-Kruskal.

Domanda 5

Sia $G = (V, E)$ un grafo pesato ed orientato. Si propongano degli algoritmi efficienti per il calcolo del MST nei seguenti casi:

a. tutti gli archi di G hanno peso k;

b. $|E| = |V|$;

c. i pesi degli archi sono numeri interi compresi tra 0 e k con k costante;

Domanda 6

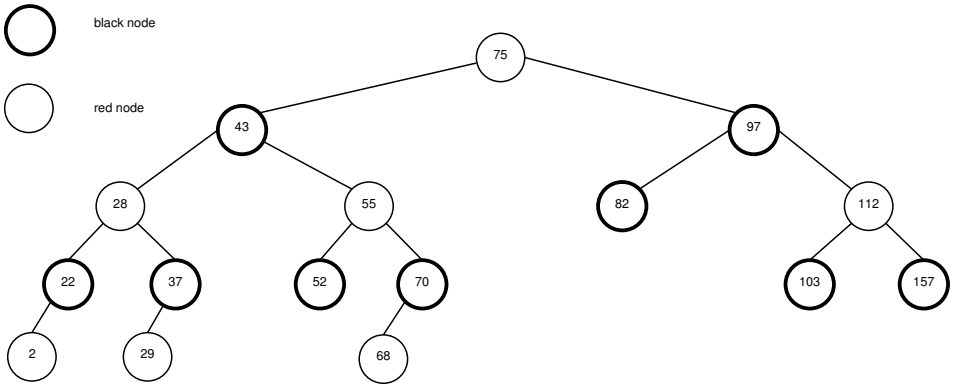
L'algoritmo di Dijkstra può operare anche su grafi non orientati. Sia G un grafo connesso non orientato e sia T l'albero dei cammini minimi costruito dall'algoritmo di Dijkstra. Si dica se le seguenti affermazioni sono vere o false, dimostrando quelle vere e fornendo un controsempio per quelle false:

a. T è uno spanning tree per G

b. T è un minimum spanning tree per G

Domanda 7

Si consideri il Red Black tree di seguito (i nodi `NIL` *non* sono stati disegnati). Si illustrino (graficamente e con una breve spiegazione dei passaggi eseguiti) i Red Black tree risultanti dagli inserimenti delle chiavi 71, 24 e 73 e dalla cancellazione della chiave 97. Le operazioni vanno eseguite nell'ordine proposto.



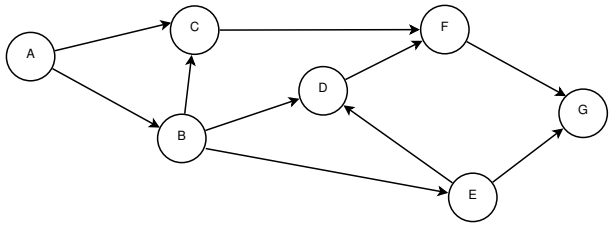
Domanda 8

Sia dato un B-tree di grado t in cui tutte le foglie contengono $t-1$ elementi. Si dica se la seguente affermazione è vera o falsa dandone una dimostrazione oppure fornendo un controsempio:

- È possibile che un inserimento faccia salire l'altezza dell'albero di 1.

Domanda 9

Si consideri il grafo orientato rappresentato nella figura seguente.



Si applichi su esso l'algoritmo Topological Sort prendendo nella visita DFS i nodi D, B, C, A nell'ordine dato (prima visito D e seguo tutti i cammini uscenti, poi riparto da B, etc.). Per ogni nodo del grafo si indichino i tempi di inizio e fine visita, partendo con tempo di inizio visita 1 sul primo nodo.

Domanda 10

Sia T un RB-Tree con radice nera, x nodo in T e k intero. Si consideri il seguente algoritmo.

TRAVERSING(T,x,k)
1: **if** $x =$NIL **then**
2: RETURN
3: **end if**
4: $k \leftarrow k+1$
5: **if** COLOR(x) =RED **then**
6: TRAVERSING(T,PARENT(x), k)
7: **else**
8: TRAVERSING(T,RIGHT(x), k)
9: **end if**
10: RETURN

La prima chiamata sia TRAVERSING(T,ROOT[T],0).

L'algoritmo proposto termina? In caso affermativo di dimostri formalmente che l'algoritmo termina, altrimenti si fornisca un controesempio.

8.3 § Esame di ASD del 23-07-10

Esercizio 1

Sia T un albero binario di ricerca. Si scriva un algoritmo il più efficiente possibile che a partire dalle chiavi in T costruisca un albero binario di ricerca completo in tutti i livelli tranne, al più, l'ultimo. Si determini la complessità e si provi la correttezza dell'algoritmo proposto.

Esercizio 2

Sia $G = (V, E)$ un grafo orientato e connesso con funzione di peso ω:

2.1 Sia $\omega : E \to \mathbb{N}$, e siano s, m ed f tre nodi del grafo. Si proponga un algoritmo che restituisca, se esiste, un percorso di peso minimo tra s ed f che *non passa* per il nodo m.

2.2 Sia $\omega : E \to \mathbb{N}$, e siano s, m ed f tre nodi del grafo. Si proponga un algoritmo che restituisca, se esiste, un percorso di peso minimo tra s ed f che *passa* per il nodo m.

2.3 Si valuti la correttezza e la complessità degli algoritmi proposti.

2.4 Si consideri ora $\omega : E \to \mathbb{R}$. Per risolvere il Single Source Shortest Path problem con l'algoritmo di Dijkstra su grafi con cicli negativi viene proposto il seguente algoritmo: sia -k il valore dell'arco di peso minimo, allora si sommi k a tutti gli archi di E e si chiami l'algoritmo di Dijkstra sul nuovo grafo. Tale soluzione è corretta? Si fornisca una dimostrazione in caso affermativo o un controesempio in caso negativo.

8.4 § Esame di ASD del 01-09-10

Esercizio 1

Si consideri il problema di gestire insiemi disgiunti di numeri interi. In particolare vogliamo implementare in maniera efficiente le seguenti operazioni:

- $Make(x)$: crea l'insieme che contiene solo l'elemento x;
- $Union(x, y)$: unisce gli insiemi a cui appartengono x ed y;
- $Find(x)$: restituisce il rappresentante dell'insieme a cui appartiene x;
- $Swap(x, y)$: scambia di insieme x ed y (x diventa elemento dell'insieme che contiene y e viceversa);
- $Sort(x)$: restituisce tutti gli elementi dell'insieme a cui appartiene x in ordine crescente;

1-a Si consideri di rappresentare gli insiemi disgiunti utilizzando i RB-tree. Si fornisca lo pseudocodice delle 5 operazioni e si valuti la complessità di ciascuna operazione.

1-b Si consideri di rappresentare gli insiemi disgiunti utilizzando le linked list. Si fornisca lo pseudocodice delle 5 operazioni e si valuti la complessità di ciascuna operazione.

1-c **FACOLTATIVO:** Sulla base della risposta data al punto **1-a** si valuti il costo di effettuare m operazioni di cui n di tipo $Make$, k di tipo $Union$ e d di tipo $Find$.

Esercizio 2

Si assuma che gli archi pesati di un grafo $\mathcal{G}$ siano memorizzati in una Min-Heap $\mathcal{H}$ e in un array $\mathcal{A}$ ordinato. Ogni nodo di $\mathcal{H}$ e ogni posizione di $\mathcal{A}$ ha come valore il peso dell'arco e come informazione ausiliaria i due nodi di $\mathcal{G}$ estremi dell'arco.

2-a Si scriva una variante dell'algoritmo di Kruskal che sfrutti unicamente $\mathcal{A}$ come struttura dati per la scelta degli archi.

2-b Si scriva una variante dell'algoritmo di Kruskal che sfrutti unicamente $\mathcal{H}$ come struttura dati per la scelta degli archi.

2-c Si discuta la correttezza e si valuti la complessità degli algoritmi proposti.

8.5 § Esame di ASD del 15-09-10

Esercizio 1

Sia H_1 un vettore di lunghezza $2n$ contenente una max-heap di interi, di dimensione n, secondo lo schema visto a lezione (e nel libro di testo). Sia H_2 un vettore di lunghezza n contenente una max-heap di interi di lunghezza n, secondo lo schema visto a lezione (e nel libro di testo). Si consideri il problema di trasformare il vettore H_1 in un vettore ordinato contenente tutti gli elementi delle heap H_1 ed H_2, senza allocare altri vettori ausiliari.

a. Fornire lo pseudocodice di un algoritmo efficiente per risolvere il problema proposto utilizzando, tra le procedure viste a lezione, solamente heapify.

b. Determinarne la complessità e dimostrarne la correttezza.

Esercizio 2

a. Si descriva tramite pseudocodice l'algoritmo di Dijkstra per la risoluzione del problema della ricerca dei cammini più brevi in un grafo orientato a partire da una sorgente singola, visto a lezione, discutendone brevemente la complessità e la correttezza.

b. Si dia la soluzione tramite pseudocodice ai seguenti problemi:

   1. ricerca dei cammini minimi verso un nodo destinazione singolo dato in input a partire da un nodo qualsiasi del grafo. Si discutano la complessità e la correttezza della soluzione proposta;
   2. ricerca dei cammini più brevi a partire da una sorgente singola per un DAG, Direct Acyclic Graph. Si discutano la complessità e la correttezza della soluzione proposta.

8.6 § Esame di ASD del 24-01-11

Esercizio 1 Definiamo un array k-`Alternato` se si può dividere in k blocchi di elementi adiacenti e se i blocchi sono ordinati alternativamente, uno dopo l'altro, in maniera crescente e descrescente.

Sia dato un array di n elementi k-`Alternato`. Si illustri tramite pseudocodice un algoritmo per ordinare l'array e se ne discutano correttezza e la complessità.

Esercizio 2

Sia $G = (N, E, w)$ un grafo orientato senza archi da un nodo a se stesso con funzione di peso $w : E \to \mathbb{N}$. Si considerino le seguenti definizioni:

- u è detto nodo pozzo se ha $|N| - 1$ archi entranti: $|in(u)| = |N| - 1$ dove $in(u) = \{v \neq u \in N | < u, v > \in E\}$;
- il peso del nodo u, indicato con $W(u)$, è la somma dei pesi degli archi entranti nel nodo u ($W(u) = \sum_{v \in in(u)} w(v, u)$);

2-a Si illustri tramite pseudocodice un algoritmo per la visita di G.

2-b Si descriva tramite pseudocodice un algoritmo che restituisca, se esiste un nodo pozzo del grafo $G = (N, E, w)$. Si calcoli la complessità dell'algoritmo proposto e se ne dimostri la correttezza.

2-c Si descriva tramite pseudocodice un algoritmo che restituisca il nodo u di peso massimo nel grafo, cioè $u \in N$ *t.c.* $\forall v \in N \setminus \{u\}$ $W(u) \geq W(v)$. Si calcoli la complessità dell'algoritmo proposto e se ne dimostri la correttezza.

Capitolo 9

ANNO 2008-2009

9.1 § Compitino di ASD del 06-02-09

Esercizio 1.[Domanda di Sbarramento] Indicare quali tra le seguenti affermazioni sono vere:

a. $5n + 2 \in \Theta(n)$

b. $n^2 \in O(n)$

c. $O(n) = \Theta(n)$

Esercizio 2.
Si consideri l'equazione ricorsiva $T(n) = T(n^{1/3}) + T(n^{2/3}) + \log_3 n$.

a. Si risolva l'equazione ricorsiva;

b. Si illustri una procedura ricorsiva che abbia tale complessità.

Esercizio 3.
Sia A un vettore di interi di lunghezza n tale che se x occorre in A ed x non è il massimo in A, allora in A occorre anche $x + 5$.

a. Fornire lo pseudocodice di un algoritmo efficiente per ordinare A.

b. Determinarne la complessità e dimostrarne la correttezza.

Esercizio 4.
Diciamo che una min-heap è *bella* se e soltanto se le chiavi ad una data profondità sono ordinate in modo crescente. Si indichino le proprietà vere nella seguente lista, motivando brevemente le risposte.

a. costruire una bella heap ha complessità $\Theta(n)$;

b. costruire una bella heap ha complessità $O(n^2)$;

c. rendere bella una heap ha complessità $\Omega(n \log n)$;

d. rendere bella una heap ha complessità $O(n)$.

Esercizio 5.
Si proponga un algoritmo lineare che dato in input un vettore A di interi di lunghezza n e due indici i e j tali che $0 < i, j \leq n$, restituisca la collezione degli elementi compresi tra l'i-esimo ed il j-esimo elemento della versione ordinata di A. Determinare la complessità e dimostrare la correttezza dell'algoritmo proposto.

Esercizio 6.
Si consideri una min-heap di dimensione n, con $n \geq 1$, priva di ripetizioni. Si indichino quali tra le seguenti affermazioni sono corrette, motivando brevemente le risposte:

a. il secondo elemento più piccolo si trova in uno dei figli della radice;

b. l'elemento più grande sta in una foglia;

c. l'elemento più grande sta nella foglia più a destra (nell'ultimo o nel penultimo livello) o nella foglia più a sinistra;

d. il massimo può stare nella radice;

Esercizio 7.
Si consideri un albero binario di ricerca (BST) con n nodi ed altezza h. Si indichino quali tra le seguenti affermazioni sono scorrette, motivando brevemente le risposte:

a. la complessità di BST-INSERT è $O(h)$;

b. il successore di x si trova sempre nel sottoalbero radicato in x;

c. la complessità di BST-INSERT è $O(n)$;

d. se y è un nodo nel sottoalbero sinistro del nodo x allora $\text{KEY}[y] < \text{KEY}[x]$;

e. il minimo del sottoalbero radicato in x non è mai x.

Esercizio 8.
Si consideri la struttura dati BST. A partire da un BST T la cui visita in pre-ordine è 5, 2, 4, 3, 6 si effettuino i seguenti inserimenti: 8, 7, 1. Per ogni inserimento si disegni l'albero ottenuto dopo l'inserimento.
Si effettuino le seguenti cancellazioni: 5, 4, 6. Per ogni cancellazione si disegni l'albero ottenuto dopo la cancellazione.

Esercizio 9.
Dati due BST T_1 e T_2 si vogliono unire i due alberi procedendo come segue:

- si scende in T_1 seguendo la strada che si seguirebbe per l'inserimento della radice di T_2;
- arrivati ad una foglia si aggancia opportunamente (a sinistra o a destra) tutto T_2.

a. Fornire lo pseudocodice della procedura sopra descritta.

b. Determinarne la complessità.

Quali tra le seguenti affermazioni sono corrette.

1. La procedura proposta restituisce sempre un BST.
2. Se tutte le chiavi di T_1 sono minori di tutte le chiavi di T_2, allora la procedura proposta restituisce sempre un BST.
3. Se i due alberi contengono complessivamente n chiavi, allora l'albero ottenuto non può avere altezza $O(\log n)$.

Si motivino opportunamente le risposte fornite.

Esercizio 10.
Il commissario Montalbano sta indagando sul dott. N.N. A tal scopo il commissario decide di utilizzare un metodo attualmente molto in voga, le intercettazioni telefoniche. L'universo dei numeri da controllare è composto da alcuni numeri a sei cifre, `U` = {100000..999999}. Il commissario, esperto informatico, decide di tenere in una tabella hash al più 1000 numeri riguardanti le ultime chiamate fatte all'interno dell'universo `U`. La tabella di hash è gestita mediante open addressing. Si dica quali delle seguenti funzioni h_i possono essere considerate funzioni ragionevoli di hash.

- $h_1(k, i) = (2 * (k + i))\ mod\ 1000$;
- $h_2(k, i) = (k + 2 * i)\ mod\ 1000$;
- $h_3(k, i) = (2 * k + 1 + i)\ mod\ 1000$;

9.2 § Compitino di ASD del 15-06-09

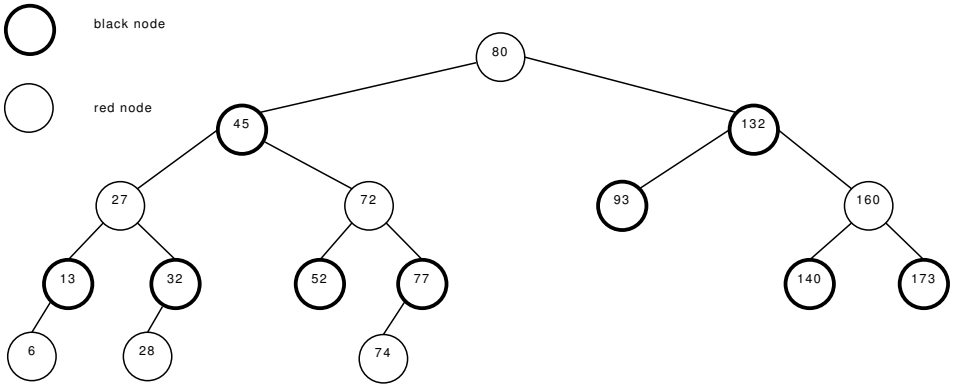
Esercizio 1.[Domanda di Sbarramento]
Indicare quali tra le seguenti affermazioni sono vere:

a. la complessità di BFS con liste di adiacenza su un grafo avente $|N|$ nodi e $|E|$ archi è $\Theta(|N| + |E|)$.

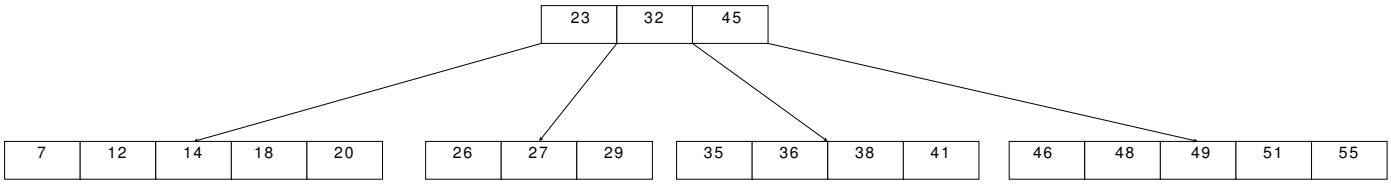
b. la complessità di DFS con liste di adiacenza su un grafo avente $|N|$ nodi e $|E|$ archi è $\Theta(|N| + |E|)$.

c. L'algoritmo di Prim e l'algoritmo di Dijkstra risolvono lo stesso problema, se i pesi sono tutti positivi.

Esercizio 2.
Si consideri il Red Black tree di seguito. Si illustrino (graficamente e con una breve spiegazione dei passaggi eseguiti) i Red Black tree risultanti dalla cancellazione della chiave 45 e dagli inserimenti delle chiavi 29, 18 e 22. Le operazioni vanno eseguite nell'ordine proposto.



Esercizio 3.
Partendo dal B-tree sottostante di grado $t = 3$, illustrare (graficamente e con una breve spiegazione dei passaggi eseguiti) i B-tree risultanti dall'inserimento della chiave 52 e dalla successiva cancellazione della chiave 27. Le operazioni vanno eseguite nell'ordine proposto.



Esercizio 4.
Si illustrino gli insiemi disgiunti che si ottengono, passo per passo, eseguendo la seguente sequenza di operazioni sugli 8 insiemi singoletto contenenti gli elementi 1, 2,. . . , 8: Union(3,5); Union(1,4); Union(5,1); Union(7,8); Union(2,6); Union(7,2), Union(5,7), Find(8). Si utilizzi la rappresentazione con alberi con le euristiche di union by rank e path compression. Si assuma che, qualora vengano uniti due alberi aventi lo stesso rango, l'albero rappresentato dall'elemento minore venga fatto puntare all'albero rappresentato dall'elemento maggiore.

Esercizio 5.
Sia $G = (N, E, W)$ un grafo non orientato, connesso e pesato tale che $W : E \to \mathbb{R}_{>0}$. Un arco di G è detto *ponte* se la sua eliminazione sconnette il grafo. Si supponga di disporre di una procedura PONTE(G) avente complessità $\Theta(|N| + |E|)$ in grado di determinare l'insieme E_p degli archi ponte di G.

a. Descrivere tramite pseudocodice una procedura per determinare un minimum spanning tree di G che faccia uso della procedura PONTE(G).

b. Determinare la complessità della procedura proposta in funzione di $|N|$, $|E|$ e $|E_p|$.

c. Dimostrare la correttezza della procedura proposta.

Esercizio 6.

Sia G un grafo orientato. Si consideri il problema di determinare un nodo v di G tale che v raggiunge il maggior numero possibile di nodi di G.

a. Descrivere tramite pseudocodice una procedura per risolvere il problema proposto.

b. Determinarne la complessità e dimostrarne la correttezza.

Esercizio 7.

Sia $G = \langle V, E \rangle$ un grafo aciclico con $V = \{1, \ldots, n\}$. Sia $flag$ un vettore di lunghezza n i cui elementi sono tutti inizializzati a 0. Sia s un nodo di G. Si consideri il seguente algoritmo.

```
PROVA(G,s)
1: if flag[s] = 0 then
2:     flag[s] ← 1
3:     for each v ∈ Adj[s] do
4:         PROVA(G,v)
5:     end for
6: end if
```

Si risponda alle seguenti domande, motivando brevemente le risposte:

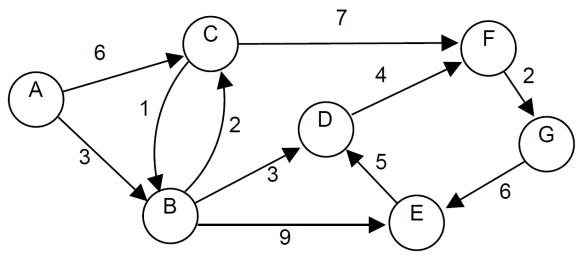
1. L'algoritmo sopra descritto termina?
2. Che complessità ha?
3. È vero che al termine dell'algoritmo tutti i campi $flag$ avranno valore 1?

Esercizio 8.

Si consideri il grafo orientato pesato rappresentato nella figura che segue.

a. Si illustri la foresta di alberi risultanti dall'esecuzione dell'algoritmo DFS, applicato a G, assumendo che i nodi vengano processati in ordine decrescente (rispetto alla loro etichetta). Per ogni nodo del grafo si indichino inoltre i tempi di inizio e fine visita, partendo con tempo di inizio visita 1 sul primo nodo.

b. Si illustri l'albero che si ottiene applicando l'algoritmo di Dijkstra a partire dal nodo A.



Esercizio 9.

Indicare quali tra le seguenti affermazioni sono corrette, motivando brevemente tutte le risposte. Sia $G = (N, E, W)$ un grafo non orientato, connesso e pesato con $W : E \to \mathbb{R}_{>0}$.

a. Sia T un minimum spanning tree di G. Si consideri il grafo G' ottenuto a partire da G, aggiungendo un nuovo nodo v ed alcuni archi con pesi positivi che connettono v a nodi di G. Sia $T' = T \cup \{\{v, x\}\}$, dove $\{v, x\}$ è l'arco con peso minore tra quelli aggiunti. T' è un minimum spanning tree di G'.

b. Sia T un minimum spanning tree di G e sia x un nodo di G. Si consideri il grafo G' ottenuto a partire da G, aggiungendo un nuovo nodo v ed un arco con peso positivo che connette v ad x. Sia $T' = T \cup \{\{v, x\}\}$. T' è un minimum spanning tree di G'.

c. Sia p un cammino minimo da s a v. Sia q un cammino minimo da v a w. Concatenando p e q si ottiene un cammino minimo da s a w.

d. Sia p un cammino minimo da s a v. Sia w un nodo che si trova all'interno del cammino p. Il cammino che si ottiene troncando il cammino p a w è un cammino minimo da s a w.

Esercizio 10.

Sia T un B-tree di grado t. In T ci sono due nodi che contengono $t - 1$ chiavi, mentre tutti gli altri nodi contengono almeno $t + 1$ chiavi. I due nodi contenenti $t - 1$ chiavi sono figli consecutivi di uno stesso nodo. Si consideri il problema di trasformare T in un B-tree in cui ogni nodo ha almeno t chiavi.

a. Descrivere tramite pseudocodice una procedura per risolvere il problema proposto.

b. Determinarne la complessità e dimostrarne la correttezza.

9.3 § Esame del 06-07-2009

Esercizio 1

Si consideri il problema Union-Find nel caso in cui si vogliano eseguire n operazioni m delle quali sono di tipo Make.

1.1 Si descriva la miglior implementazione possibile (migliore in termini di complessità) presentando: le strutture dati necessarie; lo pseudo-codice delle operazioni Make, Find e Union; la complessità di n operazioni di cui m Make.

1.2 Si consideri il caso in cui la quantità di operazioni di tipo Union è limitata da una costante k e si determini quale è la migliore implementazione possibile e che complessità raggiunge.

1.3 Si modifichi opportunamente l'implementazione proposta al punto 1.1 in modo da supportare l'operazione Find-Max(x) che restituisce l'elemento con chiave massima tra quelli dell'insieme a cui appartiene x.

Esercizio 2

Sia G un grafo orientato aciclico. Si consideri il problema di determinare, tra tutte le possibili foreste di visita DFS di G, quante radici ha una foresta DFS di G avente il minor numero possibile di radici.

2.1 Mostrare un esempio di grafo aciclico e di due visite DFS che producono su tale grafo foreste aventi un numero diverso di radici.

2.2 Dato un grafo orientato aciclico G ed una foresta di visita DFS di G, possono esistere archi di G che collegano due alberi distinti della foresta DFS? In caso di risposta negativa fornire una dimostrazione. In caso di risposta affermativa spiegare di che tipo di archi si tratta e descrivere come possono essere riconosciuti algoritmicamente.

2.3 Proporre tramite pseudocodice un algoritmo che dato G orientato aciclico restituisca in output il numero minimo di radici di una foresta DFS di G.

2.4 Dimostrare la correttezza dell'algoritmo proposto e determinarne la complessità.

9.4 § Esame del 02-09-2009

Esercizio 1

Si considerino alberi ternari in cui ogni nodo x ha i seguenti campi:

- key[x], chiave intera;
- left[x], puntatore al figlio sinistro;
- center[x] puntatore al figlio centrale;
- right[x] puntatore al figlio destro.
- p[x] puntatore al genitore.

Inoltre, per ogni nodo x vale che:

1. tutti i nodi del sottoalbero radicato in left[x] contengono chiavi minori di key[x];
2. tutti i nodi del sottoalbero radicato in right[x] contengono chiavi maggiori di key[x];
3. key[center[x]] è uguale a key[x].

Si descrivano tramite pseudocodice le procedure per ricerca, inserimento e cancellazione, discutendone la complessità (caso migliore e caso peggiore) e la correttezza.

Esercizio 2

Sia G un grafo non orientato, connesso ed s un nodo di G. Si descriva tramite pseudocodice la procedura BFS vista a lezione discutendone la complessità e la correttezza.

Si modifichi la procedura BFS in modo che il campo d per un generico nodo x, alla fine della visita, contenga la distanza da s del nodo x, utilizzando come uniche strutture dati ausiliarie il vettore delle distanze e quello dei colori (niente coda). Si proponga lo pseudocodice della procedura proposta, si discuta la complessità e la correttezza dello pseudocodice proposto.

9.5 § Esame del 04-02-2010

Esercizio 1

Si consideri la seguente variante di un RBT. T è detto *order-static tree* se ha tutte le proprietà di un RBT, e in ogni nodo x oltre ai campi $key[x], color[x], p[x], left[x]$ e $right[x]$ ha anche un ulteriore campo $size[x]$. Questo ulteriore campo contiene il numero di nodi nel sottoalbero radicato in x incluso x stesso. Per cui in un *order-static tree* con n nodi, le foglie hanno campo $size[x] = 1$, mentre la radice ha campo $size[x] = n$.
Un *order-static tree* può essere utilizzato per implementare in maniera efficiente una struttura dati che permetta le seguenti operazioni:

- `Insert(a)`: inserisce una nuova chiave
- `Extract_Min`: estrae l'elemento con chiave minima
- `Find(i)`: Trova, ma non estrae, l'i-esima chiave, cioè la chiave in posizione i se le chiavi fossero memorizzate in un array ordinato.

1.1 Si implementino le tre operazioni utilizzando un *order-static tree*

1.3 Si dimostri la correttezza e si valuti la complessità delle procedure proposte

Esercizio 2

Sia $G = (V, E)$ un grafo non orientato rappresentato con liste di adiacenza e siano dati tre vertici x, y e z in V. Si calcoli la distanza minima (numero di archi percorsi) di un percorso tra x e z che passi per y, se tale percorso esiste, e si ritorni un valore opportuno altrimenti.

2.1 Proporre sia in maniera informale che tramite pseudocodice una soluzione il più efficiente possibile al problema dato.

2.2 Dimostrare la correttezza e determinarne la complessità dell'algoritmo proposto, ponendo particolare cura alla dimostrazione di correttezza.

2.3 In base ai valori ritornati come soluzione al problema proposto, cosa si può inferire sulla connettività di G?

Capitolo 10

ANNO 2007-2008

10.1 § Compitino di ASD del 26-03-08

Esercizio 1.[Domanda di Sbarramento]
Sia T un BST (Binary Search Tree) contenente n nodi.

a. Quale è il costo nel caso peggiore di un'operazione di cancellazione?_____

b. Quale è il costo nel caso migliore di un'operazione di ricerca?_____

Esercizio 2.
Si consideri l'equazione ricorsiva $T(n) = 3 * T(n/4) + \Theta(n * \sqrt{n})$.

a. Si risolva l'equazione ricorsiva.

b. Si illustri, mediante il suo pseudocodice, una procedura che abbia ha complessità $T(n) = 3 * T(n/4) + \Theta(n * \sqrt{n})$.

Esercizio 3.
Sia A un vettore di interi di lunghezza n contenente un sottovettore ordinato. Inoltre gli elementi di A che non appartengono al sottovettore ordinato sono minori di $5n$. Formalmente riassumiamo le ipotesi su A nel seguente modo.
Esistono due indici p e q tali che: $1 \leq p \leq q \leq n$; $A[p..q]$ è ordinato; gli elementi in $A[1..p-1]$ e $A[q+1..n]$ sono minori di $5n$.
Nota: gli indici p e q NON sono noti!

a. Fornire lo pseudocodice di un algoritmo efficiente per ordinare A.

b. Determinarne la complessità e dimostrarne la correttezza.

Esercizio 4.
Sia H un vettore che contiene una Max-Heap di interi, priva di ripetizioni, di dimensione n, secondo lo schema visto a lezione. Sia k una costante intera ed $1 \leq i \leq n$ un indice. Si consideri il problema di incrementare di k l'elemento $H[i]$ e ripristinare la heap.

a. Fornire lo pseudocodice di un algoritmo efficiente per risolvere il problema proposto.

b. Determinarne la complessità e dimostrarne la correttezza.

Esercizio 5.
Sia A un vettore di lunghezza n. Riempire opportunamente gli spazi nelle seguenti affermazioni.

a. Se A è ordinato allora InsertionSort(A) ha complessità _____.

b. Se A contiene solo interi compresi tra 10 e $10n$, allora per ordinarlo uso _____.

c. Se ordino A con MergeSort(A), allora il costo in tempo è _____ ed in spazio è _____.

Esercizio 6.

Sia H una Max-Heap di dimensione n priva di ripetizioni. Indicare quali tra le seguenti affermazioni sono VERE e quali sono FALSE, motivando brevemente la risposta nella riga sottostante.

a. Se copio gli elementi di $H[p..n]$ in $A[1..n-p+1]$, allora A è una Max-Heap. _____

_____

b. Se eseguo l'istruzione $heapsize[H] \leftarrow heapsize[H] - 1$, allora H è ancora una Max-Heap._____

_____

c. Se elimino $H[1]$ spostando verso sinistra di una posizione tutti gli altri elementi e decremento $heapsize[H]$ di 1, allora H è ancora una Max-Heap. _____

_____

Esercizio 7.

Sia T un BST di dimensione n. Rispondere brevemente alle seguenti domande.

a. Osservando l'output di PostVisit(T) (Post_order_tree_walk), posso ricostruire univocamente T?

_____

b. Se il nodo x si trova nel sottoalbero destro di y, allora che relazione c'è tra $key[x]$ e $key[y]$?

_____

c. Se il nodo x si trova nel sottoalbero destro di y e $key[x] = key[y] + 1$, allora cosa posso dire sulla posizione di x?

_____

d. Quale è il costo nel caso peggiore di un'operazione di cancellazione?_____

e. Quale è il costo nel caso migliore di un'operazione di ricerca?_____

Esercizio 8.

Si consideri la struttura dati BST. A partire da un BST T vuoto si effettuino i seguenti inserimenti: 80, 37, 88, 61, 73, 43, 99, 91, 39, 12. Per ogni inserimento si disegni l'albero ottenuto dopo l'inserimento.

Si effettuino le seguenti cancellazioni: 88, 61, 80. Per ogni cancellazione si disegni l'albero ottenuto dopo la cancellazione.

Esercizio 9.

Sia T un albero binario di dimensione n ed A un vettore di lunghezza n Si consideri l'algoritmo Tree_to_array(T, A) illustrato nel seguito:

a. Determinare la complessità della procedura proposta.

```
TREE_TO_ARRAY(T, A)
 1: i ← 1
 2: Q ← ∅      % Q è una coda inizialmente vuota
 3: Enqueue(Q, root[T])      % inserisco nella coda Q la radice di T
 4: while Q ≠ ∅ do
 5:    x ← Dequeue(Q)      % x viene estratto dalla testa della coda
 6:    if x ≠ NIL then
 7:       A[i] ← key[x]      % copio la chiave di x in A[i]
 8:       i ← i + 1      % incremento l'indice i
 9:       Enqueue(Q, left[x])      % inserisco nella coda Q il nodo left[x]
10:       Enqueue(Q, right[x])      % inserisco nella coda Q il nodo right[x]
11:    end if
12: end while
```

b. Mostrare il funzionamento della procedura sull'albero binario completo T tale che INORDER(T) restituisce il seguente risultato: $1, 2, 3, 4, 5, 6, 7$. In particolare mostrare lo stato della coda Q e del vettore A all'inizio, durante ogni iterazione del ciclo while, ed al termine della procedura.

c. Dimostrare o refutare la seguente affermazione: Se T è una Max-Heap, allora al termine di TREE_TO_ARRAY(T, A) il vettore A contiene una Max-Heap.

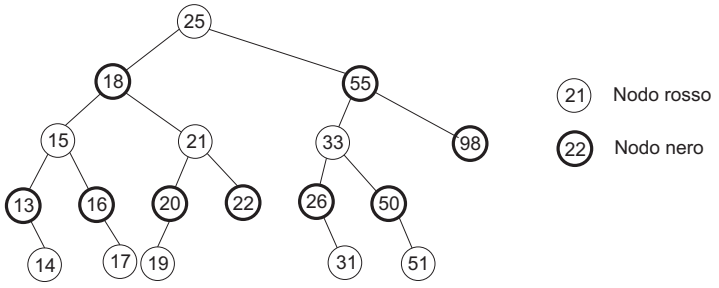
Esercizio 10.
Sia $T[0..m]$ una tabella di hash gestita attraverso open addressing e scansione lineare (linear probing).

a. Si descriva brevemente la procedura di cancellazione vista a lezione.

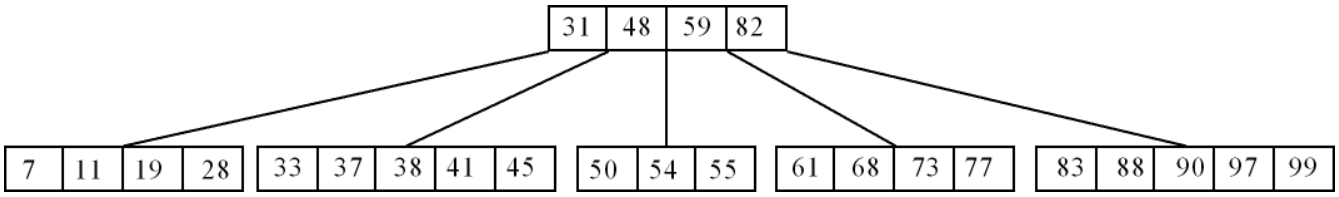
b. È possibile modificare la procedura precedentemente descritta per evitare la "marcatura" delle celle in cui avviene la cancellazione?

10.2 § Compitino di ASD del 23-06-08

Esercizio 1.
Si consideri il Red Black tree di seguito. Si illustrino (graficamente e con una breve spiegazione dei passaggi eseguiti) i Red Black tree risultanti dalla cancellazione della chiave 28 e dagli inserimenti delle chiavi 40 e 42. Le operazioni vanno eseguite nell'ordine proposto.



Esercizio 2.
Partendo dal B-tree sottostante di grado $t = 3$, illustrare (graficamente e con una breve spiegazione dei passaggi eseguiti) i B-tree risultanti dall'inserimento delle chiavi 40 e 58 e dalla successiva cancellazione delle chiavi 7 e 48. Le operazioni vanno eseguite nell'ordine proposto.

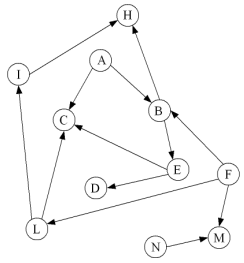


Esercizio 3.

Si illustri la famiglia di insiemi disgiunti che si ottiene eseguendo la seguente sequenza di operazioni sui 10 insiemi singoletto contenenti gli elementi 1, 2,. . . , 10: Union(1,3); Union(9,5); Union(7,1); Union(2,10); Union(5,2); Union(5,3). Si utilizzi la rappresentazione con alberi con le euristiche di union by rank e path compression. Si assuma che, qualora vengano uniti due alberi aventi lo stesso rango, l'albero rappresentato dall'elemento minore venga fatto puntare all'albero rappresentato dall'elemento maggiore.

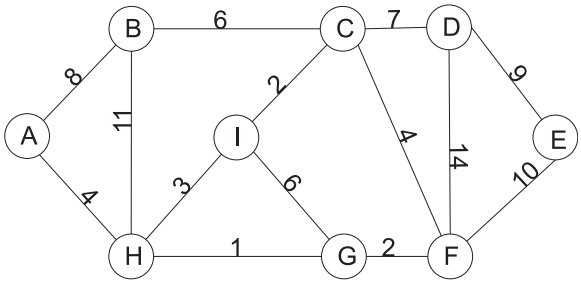
Esercizio 4.

Si consideri il grafo orientato rappresentato nella figura che segue. Si illustri l'albero risultante definito dall'algoritmo BFS, applicato a G, assumendo che i nodi vengano processati in ordine decrescente (rispetto alla loro etichetta), partendo dal nodo sorgente I. Per ogni nodo del grafo si indichino inoltre i tempi di inizio e fine visita.



Esercizio 5.

Si consideri il grafo pesato e non orientato rappresentato nella figura che segue. Si illustrino i passi compiuti dall'algoritmo di Prim, applicato a G a partire dal nodo A, e l'albero risultante.



Esercizio 6.

Dato $G = (V, E, w)$ un grafo non orientato, pesato e connesso, si vuole determinare $S \subseteq E$ tale che $G' = (V, E \setminus S)$ sia un albero.

a) Scrivere una procedura che dato G calcoli S. Determinare la complessità e dimostrare la correttezza della procedura proposta.

b) Si consideri inoltre il problema di determinare S' di PESO MINIMO. Scrivere una procedura che dato G calcoli S'. Determinare la complessità e dimostrare la correttezza della procedura proposta.

Esercizio 7.
Sia T un un RB-Tree di altezza nera hb tale che su ogni cammino radice-foglia si trova almeno un nodo rosso. Si consideri il problema di incrementare di 1 l'altezza nera di T.

Scrivere una procedura per risolvere il problema proposto. Derminare la complessità e dimostrare la correttezza della procedura proposta.

Esercizio 8.
Sia $G = (V, E)$ un grafo non orientato, connesso e pesato. Commentare le seguenti affermazioni.

a) Se G é un albero, allora durante la visita BFS quando estraggo il nodo u dalla coda Q, tutti i nodi della lista di adiacenza di u sono bianchi.

b) L'output di Prim é sempre diverso dall'output di Dijkstra.

c) Per ogni MST T esiste una coppia di nodi u e v di V tale che T contiene un cammino di peso minimo da u a v.

d) Esiste sempre un MST T tale che per ogni coppia di nodi u e v di V, T non contiene un cammino di peso minimo da u a v.

Esercizio 9.

Esercizio 10.
Sia $G = (V, E)$ un grafo non orientato tale che $V = \{1, 2, 3, \ldots, n\}$ e sia s un nodo di G. Si assuma che tutti i nodi siano stati precedentemente colorati di BIANCO e si consideri il seguente algoritmo.

VISITA(G,s)
1: $i \leftarrow n$
2: **while** ($i \notin Adj[s]$ or $color[i] != BIANCO$) **do**
3: $\quad i \leftarrow i - 1$
4: **end while**
5: **if** ($i > 0$) **then**
6: $\quad color[i] \leftarrow GRIGIO$
7: $\quad$ VISITA(G,i)
8: **end if**

Determinare l'equazione ricorsiva di complessità e risolverla.
Quante volte al più viene visitato un nodo? Fornire una breve dimostrazione.

10.3 § Esame di ASD del 21-07-08

Esercizio 1.
Sia V un vettore di lunghezza M arbitrariamente grande (NON NOTA!). Il vettore V contiene elementi non ripetuti e ordinati in modo crescente fino ad un certo punto, poi soltanto valori pari a 0.
In input viene fornita anche una chiave k.

1.1 Risolvere il problema della ricerca di k in V nel caso in cui gli elementi di V siano interi positivi.

2.1 Risolvere il problema della ricerca di k in V nel caso in cui gli elementi di V siano reali non nulli.

3.1 Dimostrare la correttezza degli algoritmi proposti e determinarne la complessità.

Esercizio 2.
Sia T un albero binario di ricerca (BST) contenente chiavi intere positive. Definiamo *interval BST* un BST tale che se le chiavi k e k' sono in T con $k < k'$, allora la chiave $k + 1$ è in T.

1.2 Proporre un algoritmo che permetta di marcare gli elementi facenti parte dell'intersezione tra due interval BST generici.

2.2 Proporre un algoritmo che permetta di marcare gli elementi facenti parte dell'intersezione tra due interval BST che siano anche RBT.

3.2 Proporre un algoritmo che permetta di marcare gli elementi facenti parte dell'intersezione tra due interval RBT nel caso in cui un interval RBT abbia dimensione logaritmica nella dimensione dell'altro.

4.2 Dimostrare la correttezza degli algoritmi proposti e determinarne la complessità.

10.4 § Esame di ASD del 05-09-08

Esercizio 1.
Si consideri la struttura dati pila su cui si possono effettuare le seguenti operazioni:

- PUSH(P,x). Inserisce l'elemento x in testa alla pila P;
- POP(P). Estrae dalla pila P l'elemento in testa;
- MULTIPOP(P,k). Estrae dalla pila P i primi k elementi che si trovano in testa.

1.1 Si scriva lo pseudocodice di una implementazione della struttura dati pila con un vettore (di dimensione M). Si implementino le operazioni PUSH, POP, MULTIPOP, segnalando errore in caso di "overflow" (pila contenente M elementi: piena) e "underflow" (pila vuota).

1.2 Si determini la complessità di: un'operazione di PUSH; un'operazione di POP; un'operazione MULTIPOP;

1.3 Si determini la complessità di m operazioni di PUSH, POP, MULTIPOP di cui n operazioni sono PUSH.

Esercizio 2.
Sia G un grafo orientato e pesato.

2.1 Si descriva tramite pseudocodice l'algoritmo di visita in ampiezza BFS(G,s) e se ne analizzi la complessità;

2.2 Si descriva un grafo $G = (N, E, w)$ tale che esiste $s \in N$ tale che non è possibile che BFS(G,s) determini l'albero dei cammini minimi da s;

2.3 Si descriva un grafo $G = (N, E, w)$ tale che esiste $s \in N$ tale che l'algoritmo di Dijkstra non determina i cammini minimi da s.

10.5 § Esame di ASD del 22-09-08

Esercizio 1.
Sia A un vettore di interi di dimensione m. Si assuma che nel vettore A siano stati inseriti $n \leq m$ numeri interi positivi provenienti da un binary search tree completo T. In particolare gli elementi di T sono stati inseriti in A usando la stessa convenzione che si usa normalmente per la memorizzazione di una heap in un vettore. Si vogliono eseguire sul vettore A le seguenti operazioni:

- Stampare le chiavi di A in ordine crescente;
- Determinare il massimo ed il minimo di A;

1.1 Si descriva tramite pseudocodice un algoritmo che dati A ed n stampa le chiavi memorizzate in A in ordine crescente. Si dimostri la correttezza dell'algoritmo proposto e se ne determini la complessità.

1.2 Si descriva tramite pseudocodice un algoritmo che dati A ed n stampa la chiave massima e la chiave minima memorizzata in A. Si dimostri la correttezza dell'algoritmo proposto e se ne determini la complessità.

1.3 Siano $A1$ ed $A2$ due vettori che memorizzano due binary seach tree completi $T1$ e $T2$ aventi ciascuno n elementi, con le convenzioni sopra fissate. Sia k una chiave intera tale che tutte le chiavi di $T1$ sono minori di k e tutte le chiavi di $T2$ sono maggiori di k. Sia A un vettore di dimensione $m \geq 2n+1$. Si vuole memorizzare in A il binary search tree completo T che si otterrebbe dalla fusione di $T1$, k, e $T2$. Si descriva tramite pseudocodice un algoritmo che dati $A1$, $A2$ ed n inserisce le chiavi nel vettore A. Si dimostri la correttezza dell'algoritmo proposto e se ne determini la complessità.

Esercizio 2.

Sia $G = (V, E)$ un grafo non orientato i cui nodi hanno un campo *color* che ha valore *rosso* oppure *verde* oppure *giallo* oppure *blu*. Diciamo che G è un grafo *ben colorato* se per ogni arco $\{u, v\} \in E$ si verifica che $color(u) \neq color(v)$.

2.1 Si descriva tramite pseudocodice un algoritmo che dato un grafo G determini se G è *ben colorato*.

2.2 Si dimostri la correttezza dell'algoritmo proposto e se ne valuti la complessità.

2.3 Si dimostri la correttezza o si refuti la seguente affermazione: Dato un grafo G è sempre possibile attribuire una colorazione ai nodi di G in modo che G sia *ben colorato*.

10.6 § Esame di ASD del 29-01-09

Esercizio 1.

Sia T un B-tree di grado t, sia x un nodo di T ed i un indice nell'intervallo $[1, n[x]]$. Sia inoltre k una costante. Si consideri il problema di determinare (se esistono) le k chiavi che precedono immediatamente la i-esima chiave di x, ovvero le k chiavi maggiori tra tutte le chiavi di T minori dell'i-esima chiave di x.

1.1 Si descriva tramite pseudo-codice un algoritmo che dati T, t, x, i e k risolva il problema sopra descritto;

1.2 Si dimostri la correttezza dell'algoritmo proposto e se ne determini la complessità.

1.3 Nel caso $k = 1$ si descriva un caso in cui l'algoritmo ha complessità pessima.

Esercizio 2.

Sia $G = (V, E)$ un grafo non orientato. Il grafo G si dice bipartito se esistono due insiemi V_1 e V_2 tali che $V_1 \cup V_2 = V$, $V_1 \cap V_2 = \emptyset$ e se $\{u, v\} \in E$, allora u e v non appartengono allo stesso V_i, con $i = 1, 2$.

2.1 Si consideri il problema di determinare se tutti i cicli di G contengono un numero pari di nodi. Che legame c'è tra questo problema ed il problema di determinare se un grafo è bipartito?

2.2 Si descriva un algoritmo che dato un grafo G determini se tutti i cicli di G hanno un numero pari di nodi ed in caso negativo restituisca un ciclo contente un numero dispari di nodi.

2.3 Si descriva tramite pseudo-codice un algoritmo che dato un grafo G determini se G è bipartito. Si dimostri la correttezza dell'algoritmo proposto e se ne calcoli la complessità.

Capitolo 11

ANNO 2006-2007

11.1 § Compitino di ASD del 17-04-07

Esercizio 1.[Domanda di Sbarramento]
Si indichi la complessità nel caso peggiore dei seguenti algoritmi:

- InsertionSort _____
- QuickSort _____
- HeapSort _____

Esercizio 2.
Sia A un vettore di lunghezza n i cui elementi sono numeri naturali. Si consideri il problema dell'ordinamento di A nell'ipotesi che in A ci siano al più k (un numero costante) elementi fuori posto. Si indichi la complessità nel caso peggiore, dei seguenti algoritmi applicati al problema proposto:

- QuickSort _____
- HeapSort _____

Proporre un algoritmo efficiente per il problema proposto.

Esercizio 3.
Sia A un vettore di interi di lunghezza $n > 3$ privo di ripetizioni. Si consideri il problema di partizionare il vettore A in 3 porzioni disgiunte A_1, A_2, A_3 in modo che gli elementi della porzione A_i siano tutti minori degli elementi della porzione A_j se $i < j$.

3.1) Fornire lo pseudocodice di un algoritmo efficiente per risolvere il problema proposto. Determinarne la complessità e dimostrarne la correttezza.

3.2) Modificare l'algoritmo proposto in modo che se n è multiplo di 3, allora A_1, A_2, A_3 abbiano la stessa cardinalità. Determinare la complessità della soluzione presentata.

Esercizio 4.
Sia A un vettore di dimensione $2^h - 1$, con $h \geq 0$. Si consideri l'algoritmo CheckHeap(A) illustrato nel seguito:

CheckHeap(A)
1: return RiCheckHeap($A, 1$)

4.1) Calcolare la complessità di CheckHeap(A).

RiCheckHeap(A, i)
1: **if** $i > heapsize(A)/2$ **then**
2: return TRUE
3: **else**
4: return $A[i] > (A[2i])^2 + (A[2i+1])^2$ and RiCheckHeap$(A, 2i)$ and RiCheckHeap$(A, 2i+1)$
5: **end if**

4.2) Dimostrare o refutare fornendo un controesempio la seguente affermazione: *se A contiene solo numeri positivi allora* CheckHeap(A) *termina restituendo TRUE se e soltanto se A è una Max-Heap.*

4.3) Se A contiene anche numeri negativi e CheckHeap(A) termina restituendo TRUE allora quale è il numero massimo di elementi negativi in A?

Esercizio 5.
Si consideri l'equazione ricorsiva $T(n) = T(n/9) + T(2n/3) + \Theta(n^2)$.

5.1) Si risolva l'equazione ricorsiva.

5.2) Si illustri, mediante il suo pseudocodice, una procedura che abbia ha complessità $T(n) = T(n/9) + T(2n/3) + \Theta(n^2)$.

Esercizio 6.
Sia H una Max-Heap di interi di dimensione n.

6.1) Se $H[i] < 0$ e $j > i$, allora cosa sappiamo di $H[j]$?

6.2) Se per ogni $i \leq heapsize[H]/2$ vale che $H[2i] < H[2i+1]$, allora H è anche un albero binario di ricerca?

6.3) Se H contiene solo potenze di 2, è privo di ripetizioni ed $H[1] = 2^k$, allora quanto vale al massimo *heapsize*?

6.4) Se $H[2] \neq H[1]$, quante ripetizioni di $H[1]$ ci sono al massimo in H?

6.5) Se $H[2] \neq H[1]$, quante ripetizioni di $H[2]$ ci sono al massimo in H?

Esercizio 7.
Sia T un albero binario di ricerca di dimensione n.

7.1) Se T è completo di altezza h ed il massimo si trova anche nella radice, quale è il numero minimo di ripetizioni del massimo in T?

7.2) Se T ha $\Omega(n)$ foglie, quanto tempo serve per trovare l'elemento con chiave minima?

7.3) Se la radice ha solo il figlio sinistro, quale è l'altezza minima di T?

7.4) Se la radice ha sia figlio sinistro, sia figlio destro, quale è l'altezza massima di T?

Esercizio 8.
Si consideri la struttura dati di albero binario di ricerca. A partire da un albero binario di ricerca T vuoto si effettuino i seguenti inserimenti: $20, 13, 4, 14, 13, 35, 40, 30, 80, 15, 32, 29$. Si effettuino le seguenti cancellazioni: $35, 20, 40$.

Esercizio 9.

Siano T_1 e T_2 due alberi binari di ricerca tali che tutte le chiavi memorizzate in T_1 sono minori delle chiavi memorizzate in T_2. Descrivere una procedura MERGEBST(T_1, T_2) che presi in input T_1 e T_2 restituisca un albero binario di ricerca T le cui chiavi sono tutte e sole le chiavi contenute in T_1 e T_2. Determinare la complessità della procedura proposta nel caso migliore e nel caso peggiore. Dimostrarne la correttezza.

Esercizio 10.

Si consideri l'universo U costituito da tutti i vettori di lunghezza 100 contententi numeri interi compresi tra 0 e 9 (estremi inclusi). Si determini quali tra le seguenti funzioni $h_i : U \mapsto \{0 \dots 999\}$ sono buone funzioni di hashing:

- $h_1(A) = max(A)\ mod\ 1000$;
- $h_2(A) = (\sum_{i=1}^{100} A[i])\ mod\ 1000$;
- $h_3(A) = (\sum_{i=1}^{100} A[i] * i)\ mod\ 1000$;
- $h_4(A) = \sum_{i=1}^{100}((A[i] * i)\ mod\ 1000)$;
- $h_5(A) = (\sum_{i=1}^{100}((A[i] * i)\ mod\ 1000))\ mod\ 1000$;

Motivare brevemente le risposte.

11.2 § Compitino di ASD del 28-06-07

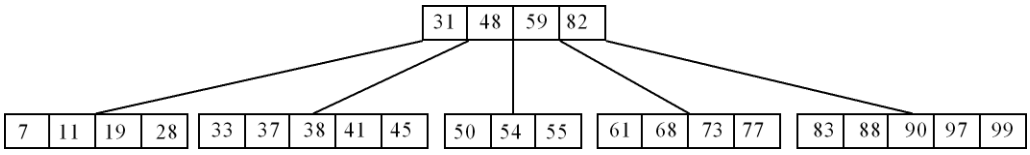
Esercizio 1. Domanda di sbarramento.

Si indichi la complessità, nel caso peggiore, dei seguenti algoritmi:

- DFS VISIT _____
- B-TREE INSERT _____

Esercizio 2.

Partendo dal B-tree sottostante di grado $t = 3$, illustrare (graficamente e con una breve spiegazione dei passaggi eseguiti) i B-tree risultanti dall'inserimento della chiave 34 e dalla cancellazione della chiave 48.



Esercizio 3.

Si illustri la famiglia di insiemi disgiunti che si ottiene eseguendo la seguente sequenza di operazioni sui 5 insiemi singoletto contenenti gli elementi 1, 2, 3, 4, 5. Si utilizzi la rappresentazione con alberi con le euristiche di union by rank e path compression. Si assuma che, qualora vengano uniti due alberi aventi lo stesso rango, l'albero rappresentato dall'elemento minore venga fatto puntare all'albero rappresentato dall'elemento maggiore.

UNION(1,2); UNION(3,4); UNION(1,4); UNION(1,5).

Esercizio 4.

Risolvere l'equazione di complessità $T(n) = 6T\left(\frac{n}{2}\right) + n^2$

Esercizio 5.

Sia $G = (V, E, w)$ un grafo connesso, pesato e non orientato, con un numero di archi pari al numero di nodi.

6.1 Si proponga un algoritmo efficiente che determini il Minimum Spanning Tree di G, valutandone la complessità e dimostrandone la correttezza.

6.2 Definiamo *Secondo Migliore* Minimum Spanning Tree di G uno Spanning Tree T' di G tale che $w(T) < w(T') < w(T")$ per ogni Minimum Spanning Tree T di G e per ogni Spanning Tree $T"$ di G. Si proponga un algoritmo efficiente che determini, se esiste, un Secondo Migliore Spanning Tree di G.

Esercizio 6a.

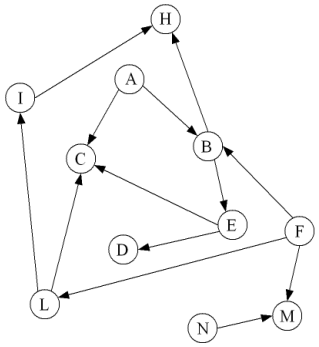
Sia $G = (V, E, w)$ un grafo connesso, pesato e non orientato, con un numero di archi pari al numero di nodi.

a Si proponga un algoritmo efficiente che determini il Minimum Spanning Tree di G, valutandone la complessità e dimostrandone la correttezza.

b Definiamo *Secondo Migliore* Minimum Spanning Tree di G uno Spanning Tree T' di G tale che $w(T) < w(T') < w(T")$ per ogni Minimum Spanning Tree T di G e per ogni Spanning Tree $T"$ di G. Si proponga un algoritmo efficiente che determini, se esiste, un Secondo Migliore Spanning Tree di G.

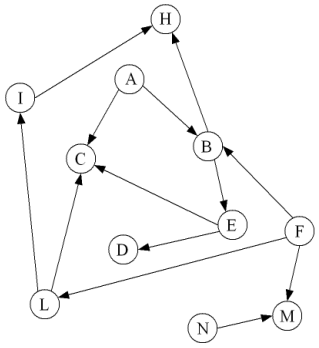
Esercizio 6b.

Si consideri il grafo orientato rappresentato nella figura che segue. Si illustri l'albero risultante definito dall'algoritmo BFS, applicato a G, assumendo che i nodi vengano processati in ordine crescente (rispetto alla loro etichetta), partendo dal nodo sorgente F. Per ogni nodo del grafo si indichino inoltre l'ordine di visita e la distanza dalla sorgente.



Esercizio 7.

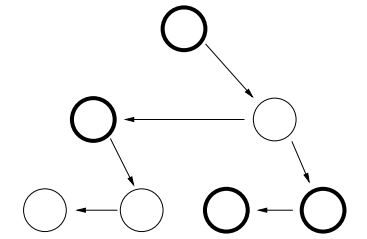
Si consideri il grafo orientato rappresentato nella figura che segue. Si illustri l'albero risultante definito dall'algoritmo BFS, applicato a G, assumendo che i nodi vengano processati in ordine crescente (rispetto alla loro etichetta), partendo dal nodo sorgente F. Per ogni nodo del grafo si indichino inoltre l'ordine di visita e la distanza dalla sorgente.



Esercizio 8a.

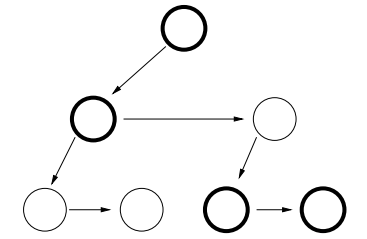
Un Red Black tree di fratelli è Red Black tree in cui ogni nodo NON ha i puntatori $left[x]$ e $right[x]$, ma al loro posto ha un puntatore al proprio figlio destro $rightson[x]$ ed un puntatore al proprio fratello sinistro $leftbr[x]$, come illustrato nella figura

sottostante. Ogni nodo ha anche il puntatore $parent[x]$. Scrivere lo pseudocodice delle procedure di ricerca del minimo e del massimo in un Red Black tree di fratelli. Dimostrarne la correttezza e calcolarne la complessità



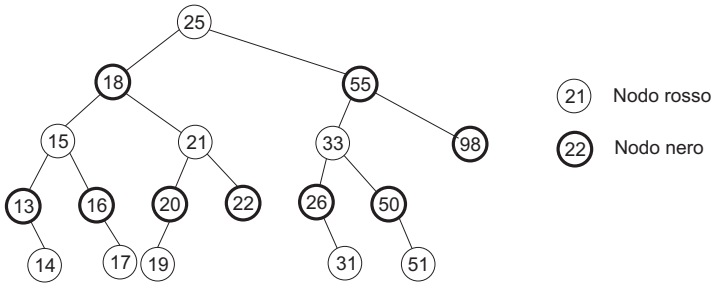
Esercizio 8b.

Un Red Black tree di fratelli è Red Black tree in cui ogni nodo NON ha i puntatori $left[x]$ e $right[x]$, ma al loro posto ha un puntatore al proprio figlio sinistro $leftson[x]$ ed un puntatore al proprio fratello destro $rightbr[x]$, come illustrato nella figura sottostante. Ogni nodo ha anche il puntatore $parent[x]$. Scrivere lo pseudocodice delle procedure di ricerca del minimo e del massimo in un Red Black tree di fratelli. Dimostrarne la correttezza e calcolarne la complessità



Esercizio 9.

Si consideri il Red Black tree di seguito. Si illustrino (graficamente e con una breve spiegazione dei passaggi eseguiti) i Red Black tree risultanti dall'inserimento della chiave 30 e dalla cancellazione della chiave 55. Le operazioni vanno eseguite nell'ordine proposto.



Esercizio 10.

Sia $G = (V, E, w)$ un grafo orientato, aciclico e pesato con pesi positivi. Sia s un nodo di G che raggiunge tutti gli altri nodi. Si consideri il seguente algoritmo per la determinazione dei cammini di lunghezza massima dalla sorgente s.

a Dimostrare che la procedura proposta è corretta oppure mostrare un controesempio.

b Calcolare la complessità della procedura nel caso in cui Q sia una lista concatenata.

```
AlgoFilaA(G,s)
1: Q ← V
2: d[s] ← 0
3: for (each v ∈ V \ {s}) do
4:    d[v] ← ∞
5:    π[v] ← NIL
6: end for
7: while (Q ≠ ∅) do
8:    v ← Min(Q)
9:    Q ← Q \ {v}
10:   for (each u ∈ Adj[v]) do
11:      if (d[u] = ∞ or d[u] < d[v] + w(v,u)) then
12:         d[u] ← d[v] + w(v,u)
13:         π[u] ← v
14:      end if
15:   end for
16: end while
```

11.3 § Esame di ASD del 16-07-07

Esercizio 1.

Si consideri un grafo orientato $G = \langle V, E\rangle$. Un cammino infinito in G è una sequenza infinita di nodi $v_1, v_2, \ldots, v_n, \ldots$ tali che $\forall i \geq 1$ si ha $\langle v_i, v_{i+1}\rangle \in E$.

1-a Si scriva lo pseudocodice di un algoritmo che stabilisca se in G c'è almeno un cammino infinito;

1-b Si scriva lo pseudocodice di un algoritmo che stabilisca se ogni nodo di G appartiene ad almeno un cammino infinito;

1-c Si dimostri la correttezza e si calcoli la complessità degli algoritmi proposti.

Esercizio 2.

Sia T un RB-tree e sia i un numero intero.

a Il nodo x più vicino ad i è il nodo di T che contiene la chiave $key[x]$ più vicina ad i, ovvero per ogni altra chiave k' di T vale $|k' - i| \geq |key[x] - i|$;

b Il nodo nero y più vicino ad i è il nodo nero di T che contiene la chiave $key[y]$ più vicina ad i tra tutte le chiavi contenute in nodi neri;

b Il nodo rosso z più vicino ad i è il nodo rosso di T che contiene la chiave $key[z]$ più vicina ad i tra tutte le chiavi contenute in nodi rossi;

2-a Si scriva lo pseudocodice di un algoritmo che determini il nodo x di T più vicino ad i;

2-b Si scriva lo pseudocodice di un algoritmo che determini il nodo nero y più vicino ad i;

2-c Si scriva lo pseudocodice di un algoritmo che determini il nodo rosso z più vicino ad i;

2-d Si dimostri la correttezza e si calcoli la complessità degli algoritmi proposti.

11.4 § Esame di ASD del 03-09-07

Esercizio 1.

Si consideri il problema di gestire insiemi disgiunti di numeri interi e si proponga una struttura dati per rappresentare gli insiemi che consenta di eseguire efficientemente le seguenti operazioni:

$Make(x)$ che preso in input x costruisce l'insieme contente solo x;

$May\_Union(x, y)$ che presi in input due elementi x ed y unisce gli insiemi a cui appartengono x ed y se e soltanto se tutti gli elementi dell'insieme di x sono minori o uguali degli elementi dell'insieme di y;

$Find(x)$ che restituisce il rappresentante dell'insieme a cui appartiene x;

$Sort(x)$ che restituisce tutti gli elementi dell'insieme a cui appartiene x in ordine crescente.

1-a Si descriva la struttura dati scelta e si fornisca lo pseudocodice delle operazioni sopra elencate;

1-b Si dimostri la correttezza delle procedure proposte e si valuti la complessità di una singola esecuzione di ogni procedura;

1-c Si valuti la complessità di n operazioni complessive di cui m operazioni $Make$ ed s operazioni $Sort$.

Esercizio 2.
Si consideri un grafo orientato $G = \langle V, E \rangle$ in cui ogni nodo ha un campo aggiuntivo *colore* che può assumere valore $Verde$ oppure *Rosso*. Un cammino *alternato* in G è un cammino i cui archi collegano nodi di colore diverso. Un cammino *quasi alternato* in G è un cammino i cui archi collegano nodi di colore diverso, tranne al più un arco che può collegare nodi dello stesso colore. Dati due nodi u e v di G si vogliono determinare:

- un cammino alternato di lunghezza minima da u a v;
- un cammino quasi alternato di lunghezza minima da u a v.

2-a si scriva lo pseudocodice di due algoritmi che risolvano i due problemi proposti;

2-b si dimostri la correttezza e si valuti la complessità degli algoritmi proposti.

11.5 § Esame di ASD del 20-09-07

Esercizio 1.
Sia $G = (V, E)$ un grafo orientato aciclico con n nodi. Dato un nodo $u \in V$, il rango di u è la massima distanza di u da una foglia (nodo di G che non ha archi uscenti).

1-a Dimostrare che esiste sempre almeno una foglia;

1-b Descrivere un algoritmo che calcola il rango di tutti i nodi di un grafo orientato aciclico;

1-c Dimostrare la correttezza dell'algoritmo proposto e determinarne la complessità.

Esercizio 2.
Sia A un vettore di interi di lunghezza n. Un gap in A è un indice i tale che $A[i] < A[i+1]$.

1-a Dimostrare che se A è un vettore di interi di lunghezza $n \geq 2$ e $A[1] < A[n]$, allora A contiene almeno un gap;

2-a Descrivere un algoritmo che dato un vettore A tale che $A[1] < A[n]$ determini un gap di A in tempo $o(n)$ (o piccolo);

3-a Dimostrare la correttezza dell'algoritmo proposto e determinarne la complessità.

11.6 § Esame di ASD del 14-12-07

Esercizio 1.

Sia $G = (V, E)$ un grafo orientato. Una componente connessa C di G è una componente connessa del grafo non orientato $G' = (V, E')$ che si ottiene non considerando l'orientamento degli archi di G (ovvero, l'arco non orientato $\{u, v\}$ appartiene ad E' se e soltanto se l'arco orientato $\langle u, v\rangle$ appartiene ad E).

1-a Descrivere un algoritmo che dato un nodo u di G determina tutti i nodi appartenenti alla componente connessa di u;

1-b Dimostrare la correttezza dell'algoritmo proposto e determinarne la complessità;

1-c Descrivere un algoritmo che data una componente connessa C di G determina se esiste un nodo v che raggiunge ogni nodo di C.

Esercizio 2.

Sia A un vettore di interi di lunghezza n privo di ripetizioni. Un elemento x di A si dice *fuori posto* se x si trova in posizione i in A e si troverebbe in posizione $j \neq i$ se ordinassimo A (per qualche valore di i e j).

2-a Descrivere un algoritmo per determinare quanti elementi sono fuori posto in A;

2-b Descrivere un algoritmo con complessità $\Theta(n)$ per decidere se in A ci sono almeno 2 elementi fuori posto;

2-c Dimostrare la correttezza degli algoritmi proposti e determinarne la complessità.

Capitolo 12

ANNO 2005-2006

12.1 § Compitino di ASD del 12-04-06

Esercizio 1.
Sia A un vettore di lunghezza n i cui elementi sono numeri naturali. Si consideri il problema dell'ordinamento di A nell'ipotesi che in A c'è k se e solo se in A c'è anche $n - k$. Si indichi la complessità nel caso peggiore, dei seguenti algoritmi applicati al problema proposto:

- Insertion Sort _____
- Quick Sort _____
- Heap Sort _____

Fornire un algoritmo efficiente.

Esercizio 2.
Si consideri il problema di ordinare il vettore di interi A contenente k elementi più grandi di k^2 e gli altri minori o uguali a k^2, con k costante. Fornire un algoritmo efficiente.

Esercizio 3.
Si consideri l'algoritmo HeapRiSort(A) illustrato nel seguito:

HeapRiSort(A)
1: BuildHeap(A)
2: **for** $i = 1$ to $length(A)$ **do**
3: $p =$ DeHeapify$(A, 1)$
4: $scambia(A, p, heapsize[A])$
5: $heapsize[A] = heapsize[A] - 1$
6: **end for**

3.1) Calcolare la complessità

3.2) L'algoritmo è corretto? _____ Si utilizzi il foglio allegato per provare formalmente (fornendo, eventualmente, un controesempio) quanto affermato.

```
DeHeapify(A, j)
1: if left(j) = NIL then
2:    return j
3: else
4:    if right(j) = NIL or A[left(j)] > A[right(j)] then
5:       m = left(j)
6:    else
7:       m = right(j)
8:    end if
9:    scambia(A, j, m)
10:   return DeHeapify(A, m)
11: end if
```

Esercizio 4.
Si consideri l'equazione ricorsiva $T(n) = 5 * T(n/2) + \Theta(n^2)$.

4.1) Si utilizzi lo spazio sottostante per risolvere l'equazione ricorsiva;

4.2) si utilizzi lo spazio sottostante per illustrare, mediante il suo pseudocodice, una procedura che abbia ha complessità $T(n) = 5 * T(n/2) + \Theta(n^2)$.

Esercizio 5.
Sia H una Max-Heap di dimensione n.

5.1) Se il successore del minimo non è in una foglia, in che posizione si trova?

5.2) Se H è completa, dove si trova il successore del minimo?

5.3) Se H è priva di ripetizioni, dove si trova il predecessore del massimo?

5.4) Se in $H[i]$ c'è il massimo, qual è il numero minimo di ripetizioni del massimo in H?

5.5) Se H è completa ed in posizione $H[j]$ con $j < \frac{n}{2}$ c'è il minimo, qual è il numero minimo di ripetizioni del minimo in H?

Esercizio 6.
Sia T un albero binario di ricerca di dimensione n.

6.1) Se T è completo e di altezza h, trovare l'elemento che finirebbe in posizione $\lfloor \frac{3}{8} \rfloor$ durante un'eventuale visita In-Order.

6.2) Se T ha solo due foglie, quanto tempo serve per trovare l'elemento con chiave minima? E quello con chiave media?

6.3) Se il minimo è nella radice il massimo non è in una delle foglie, qual è l'altezza massima di T? E l'altezza minima? Qual è la profondità massima e minima del nodo con chiame massima?

Esercizio 7.
Si consideri la struttura dati di *albero binario di ricerca* (BST). Inserimenti e cancellazioni... incompleto.

Esercizio 8.
Sia T un albero binario di ricerca avente n nodi. Considerare la procedura MODIFY_KEY(T, x, key) che modifica `key[x]` con key e ritorna T se T è ancora un BST, NULL altrimenti. Si utilizzi il foglio allegato per scrivere la procedura, dimostrandone brevemente correttezza e complessità

Esercizio 9.
Si consideri l'universo U delle date degli eventi storici dalla scoperta dell'America ai giorni nostri. Si supponga che una data sia espressa nel formato $x = D_1D_2M_1M_2Y_1Y_2Y_3Y_4$ (giorno–mese–anno). Si considerino le seguenti funzioni di hasing $h_i \colon U \to \{0, \ldots, 999\}$:

- $h_1(x) = (D_1 * 1000 + M_1 * 100 + Y_1 * 10 + Y_3) \mod (1000)$;
- $h_2(x) = (D_2 * 1000 + M_2 * 100 + Y_2 * 10 + Y_4) \mod (1000)$;
- $h_3(x) = (D_2 * 1000 + M_2 * 100 + Y_3 * 10 + Y_4) \mod (1000)$;

Si mostri se sono o meno delle buone funzioni di hashing motivando brevemente le risposte.

Esercizio 10.
Sia data la sequenza di interi, supposta memorizzata in un vettore: 32 45 55 61 21 29 39 38 37 33. La si trasformi in una min-heap, ipotizzando di usare un vettore come struttura dati. Si riportino i diversi passi della costruzione del vettore corrispondente alla min-heap, specificando le procedure utilizzate, ed il risultato finale.

12.2 § Compitino di ASD del 23-06-06

Esercizio 1.
Inserimenti e Cancellazioni in un RB-tree esempio. Fare in modo che la cancellazione sia di un nodo NERO con due figli avente predecessore ROSSO. Poi magari anche la cancellazione di un nodo NERO con un solo figlio ROSSO.

Esercizio 2.
Inserimenti e Cancellazioni in un B-tree esempio.

Esercizio 3.
Union-find su alberi con union by rank e path compression. Un po' di make, union e find.

Esercizio 4.
Un grafo orientato ciclico. Mostrare output di BFS e DFS.

Esercizio 5.
Risolvere l'equazione di complessità.

$$T(n) = 3T\left(\frac{n}{4}\right) + \sqrt{n}$$

Esercizio 6.
Per ognuna delle seguenti affermazioni dimostrare la validità oppure presentare un controesempio.

a Sia $G = (V, E, w)$ un grafo connesso non orientato e pesato, siano $x, y \in V$ due nodi, sia p un cammino MINIMO x ad y in G. Per ogni Minimum Spanning Tree T di G il cammino p è contenuto in T.

b Sia $G = (V, E, w)$ un grafo connesso non orientato e pesato, siano $x, y \in V$ due nodi, sia p un cammino MINIMO x ad y in G. Esiste sempre un Minimum Spanning Tree T di G tale che il cammino p è contenuto in T.

c Sia $G = (V, E, w)$ un grafo connesso non orientato e pesato tale che $w : E \longrightarrow \{k\}$, siano $x, y \in V$ due nodi, sia p un cammino MINIMO x ad y in G. Esiste sempre un Minimum Spanning Tree T di G tale che il cammino p è contenuto in T.

Esercizio 7.
Si consideri il problema di mantenere in memoria insiemi disgiunti ordinati (in ordine crescente). Proporre opportune modifiche alle strutture dati per insiemi disgiunti che consentano di risolvere in maniera efficiente il problema proposto. Si illustri lo pseudocodice di UNION e se ne calcoli la complessità.

Esercizio 8.
Sia A un vettore ordinato di interi di dimensione n. Si consideri il problema di costruire un B-tree T di grado 2 avente come chiavi tutti e soli gli elementi di A. Si illustri lo pseudocodice di un algoritmo per risolvere tale problema. Si dimostri la correttezza dell'algoritmo proposto e se ne valuti la complessità.

Esercizio 9.
Sia $G = (V, E, w)$ un grafo connesso, non orientato, aciclico e pesato, sia $s \in V$ un nodo di G. Si proponga un algoritmo per determinare i cammini di lunghezza massima da s ad ogni altro nodo di G. Si discuta la correttezza dell'algoritmo presentato e se ne valuti la complessità.

Esercizio 10.
Sia $G = (V, E, w)$ un grafo connesso, non orientato e pesato, sia T un Minimum Spanning Tree di G. Sia G' il grafo ottenuto da G eliminando un arco (u, v). Si consideri il problema di determinare un Minimum Spanning Tree T' di G'. Presentare lo pseudocodice di un algoritmo efficiente per determinare T'. Dimostrarne correttezza e complessità.

Esercizio 11.
Sia $G = (V, E, w)$ un grafo connesso, non orientato e pesato. Sia RB un RB-tree che mantiene in memoria gli archi di E con i rispettivi pesi. Si consideri la seguente variante dell'algoritmo di Kruskal.

RiKruskal(G,RB)

```
1: for (each v ∈ V) do
2:    Make(v)
3: end for
4: A ← ∅
5: while (|A| < |V| − 1) do
6:    (u, v) ← Min(RB)
7:    RBtree_Delete(RB,(u, v))
8:    if (Find(u)≠ Find(v)) then
9:       Union(u, v)
10:   end if
11: end while
12: return A
```

Dimostrare che la procedura proposta determina un Minimum Spanning Tree di G oppure mostrare un controesempio. Calcolare la complessità della procedura nei due casi seguenti:

a Gli insiemi disgiunti vengono gestiti con alberi ed euristiche di union by rank e path compression;

b Gli insiemi disgiunti vengono gestiti con alberi ed euristica di union by rank.

12.3 § Esame di ASD del 10-07-06

Esercizio 1.
Si consideri la un grafo orientato $G = \langle V, E \rangle$.

1-a Si scriva lo pseudocodice di un algoritmo che produca in output un ordinamento topologico di G (se esiste);

1-b si scriva lo pseudocodice di un algoritmo ricorsivo che produca in output *tutti* i possibili ordinamenti topologici di G;

1-c si dimostri la correttezza e si calcoli la complessità degli algoritmi proposti.

Esercizio 2.
Si consideri un insieme dinamico A di numeri interi e si proponga una struttura dati per rappresentarne gli elemnenti che consenta di eseguire efficientemente le seguenti istruzioni

$Build(A)$ che presa in input la lista degli elementi di A ne restituisce la rappresentazione (B);

$Ins(B, x)$ che preso in input B restituisce $B \cup \{x\}$;

$Del(B, x)$ che preso in input B restituisce $B \setminus \{x\}$;

$Mod(B, x, y)$ che preso in input B restituisce $(B \setminus \{x\}) \cup \{y\}$;

$M3(B)$ che preso in input B restituisce il valore $\lfloor \frac{[max(B)-min(B)]}{3} \rfloor$.

2-a Si dimostri la correttezza delle procedure disegnate rispetto alla rappresentazione proposta;

2-b si determini la complessità delle procedure proposte;

2-c si gestisca anche l'istruzione $[Med(B)]$ che preso in input B restituisce il valore mediano di B.

12.4 § Esame di ASD del 01-09-06

Esercizio 1.
Si consideri un vettore A di lunghezza $2n$. Le prime n posizioni del vettore A contengono numeri interi appartenenti all'intevallo $[0, n)$. Le ultime n posizioni di A contengono l'intero 0. Si vogliono ordinare gli elementi contenuti nelle prime n posizioni di A.

1-a Si descriva un algoritmo lineare ed in-place (non si possono utilizzare altri vettori ausiliari) per risolvere il problema proposto.

1-b Si descriva un algoritmo lineare, in-place e stabile per risolvere il problema proposto.

Esercizio 2.
Sia $G = (V, E, w)$ un grafo non orientato e pesato tale che $w : V \longrightarrow \mathbb{R}^+$. Si considerino i seguenti problemi:

2.1 determinare un Minimum Spanning Tree di G;

2.2 determinare l'albero dei cammini minimi da una sorgente $s \in V$.

2-a Si proponga lo pseudocodice completo degli algoritmi per risolvere i problemi di cui sopra ($Algo1$ e $Algo2$ rispettivamente) indicandone la complessità sia in termini di tempo che in termini di spazio;

2-b È possibile risolvere il problema [2.1] utilizzando un algoritmo che risolve il problema [2.2]? In caso affermativo si descriva un algoritmo che utilizzi come procedura l'algoritmo $Algo2$. Se ne dimostri la correttezza. In caso negativo si determini un controesempio e si descrivano delle ipotesi aggiuntive su G che rendano vera l'affermazione.

2-c È possibile risolvere il problema [2.2] utilizzando un algoritmo che risolve il problema [2.1]? In caso affermativo si descriva un algoritmo che utilizzi come procedura un algoritmo $Algo1$. Se ne dimostri la correttezza. In caso negativo si determini un controesempio e si descrivano delle ipotesi aggiuntive su G che rendano vera l'affermazione.

12.5 § Esame di ASD del 19-09-06

Esercizio 1.
Sia T un albero binario avente n nodi tale che:

- ogni nodo x ha una chiave intera $key[x]$, i puntatori $left[x]$ e $right[x]$ ai figli, il puntatore $parent[x]$ al genitore;
- per ogni nodo x tutte le chiavi contenute nel sottoalbero radicato in $left[x]$ sono minori di tutte le chiavi contenute nel sottoalbero radicato in $right[x]$.

1-a Si scriva lo pseudocodice di un algoritmo per ordinare le chiavi di T;

1-b Si dimostri la correttezza dell'algoritmo proposto e se ne calcoli la complessità;

Esercizio 2.
Sia $G = (V, E)$ un grafo orientato aciclico e siano s e p due nodi di G.

2-a Si descriva un algoritmo per determinare se tutti i cammini che partono da s **terminano** in p. Se ne dimostri la correttezza e se ne calcoli la complessità.

2-b Si descriva un algoritmo per determinare se tutti i cammini che partono da s **arrivano** a p. Se ne dimostri la correttezza e se ne calcoli la complessità.

12.6 § Esame di ASD del 09-01-07

Esercizio 1.
Si consideri una max-heap A di lunghezza n.

1-a Si descriva un algoritmo che dato un indice i ed un valore k determini quante occorrenze di k ci sono nella heap radicata in posizione i. Si dimostri la correttezza e si determini la complessità dell'algoritmo proposto.

1-b Sfruttando l'algoritmo presentato al punto precedente si presenti un algoritmo che determina quale è l'elemento maggiormente ripetuto in A. Si dimostri la correttezza e si determini la complessità dell'algoritmo proposto.

1-c Esiste un algoritmo più efficiente di quello descritto al punto **1-b** per determinare quale è l'elemento maggiormente ripetuto in A? Motivare la risposta opportunamente.

Esercizio 2.
Sia $G = (V, E)$ un grafo non orientato tale che $V = \{1, 2, \ldots, n\}$, sia s un nodo di G e sia $Dist$ un vettore di lunghezza n.

2-a si descriva un algoritmo ricorsivo che non utilizzi altre strutture dati e che per ogni nodo u di G memorizzi nel vettore $Dist$ in posizione u (ovvero in $Dist[u]$) la distanza di u da s.

2-b si dimostri la correttezza dell'algoritmo proposto.

Capitolo 13

ANNO 2004-2005

13.1 § Compitino di ASD del 14-04-05

Esercizio 1.
Sia A un vettore di lunghezza n che rappresenta una max-heap. Si consideri il problema di sostituire l'i-esimo elemento di A e ritrasformare A in una max-heap. Si indichi, con una crocetta, quali delle seguenti affermazioni sono corrette:

- l'operazione ha costo $O(\log n)$;
- l'operazione ha costo $\Theta(\log n)$;
- occorre utilizzare la procedura `BuildHeap`(A);
- occorre utilizzare $O(\log n)$ chiamate alla procedura `Heapify`$(A, 1)$.

Esercizio 2.
Si consideri il problema di ordinare vettori contenenti interi nell'intervallo $[1..c]$, dove c è una costante. Nelle ipotesi date si indichi, con una crocetta, quali delle seguenti affermazioni sono corrette:

- InsertionSort ha complessità $o(n^2)$ sui vettori delineati nelle ipotesi;
- SelectionSort ha complessità $\Theta(n^2)$ sui vettori delineati nelle ipotesi;
- QuickSort ha complessità $\Theta(n^2)$ sui vettori delineati nelle ipotesi;
- CountingSort ha complessità lineare sui vettori delineati nelle ipotesi.

Esercizio 3.
Si consideri l'algoritmo ricorsivo Algo$(A[1 \dots n])$ per l'ordinamento di un vettore contenente n numeri interi, illustrato nel seguito:

```
Algo(A[1...n])
1: if lenght(A) > 1 then
2:    pivotEl ←Select(A, n/3);      Utilizza l'algoritmo Select per determinare pivotEl
3:    q ← Partition(A, pivotEl);    Partiziona A[1...n] usando pivotEl come pivot
4:    Ordina(A[q + 1...n])
5:    Algo(A[1...q])
6: end if
```

Si definisca l'equazione di complessità, *nel caso migliore*, relativa all'algoritmo Algo in cui la procedura Ordina viene implementata mediante:

1. SelectionSort _____

2. InsertionSort _____

Si risolvano le relazioni di ricorrenza $(a), (b)$ definite al punto precedente e si completi opportunamente la tabella seguente, relativa alle complessità espresse dalle suddette equazioni $(a), (b)$.

	$\Theta(n^2)$	$\Theta(n^3 \log(n))$	$\Theta(n^3)$	$\Theta(n^2 \log(n))$	$\Theta(n \log(\log(n)))$	$\Theta(n \log(n))$
(a)						
(b)						

Esercizio 4.
Si consideri l'equazione ricorsiva $T(n) = 2 * T(n/2) + \Theta(n^2)$.

4.1) Si utilizzi lo spazio sottostante per risolvere l'equazione ricorsiva;

4.2) si utilizzi lo spazio sottostante per illustrare, mediante il suo pseudocodice, una procedura $\mathtt{P}(A[1 \dots n])$ che opera su un vettore $A[1 \dots n]$ ed ha complessità $T(n) = 2 * T(n/2) + \Theta(n^2)$.

Esercizio 5.
Siano a_1, a_2, a_3, a_4 quattro numeri naturali distinti. Quante max-heap si possono costruire con questi elementi? _____. Si utilizzi lo spazio sottostante per illustrare tali max-heap.

Esercizio 6.
Sia T un albero binario avente tutti i livelli completi ed i cui nodi sono etichettati mediante numeri naturali. Sia r la radice di tale albero e si consideri l'algoritmo SetUp(r), definito nel seguito, per la costruzione di una max-heap.

```
SetUp(r)
1: if r ≠ NIL then
2:    m ← left(r); n ← right(r);
3:    SetUp(m); SetUp(n);
4: end if
5: Heapify(r)
6: return r
```

6.1) L'algoritmo è corretto? _____ Si utilizzi il foglio allegato per provare formalmente (fornendo, eventualmente, un controesempio) quanto affermato.

6.2) Si definisca l'equazione ricorsiva di complessità relativa all'algoritmo SetUp _____ Si utilizzi quindi lo spazio sottostante per risolvere tale equazione di complessità

6.3) L'algoritmo Setup è efficiente? Si motivi brevemente la risposta, utilizzando lo spazio sottostante.

Esercizio 7.

Sia T un albero binario avente tutti i livelli pieni tranne, al più, l'ultimo. Si assuma inoltre che, ad ogni nodo x in T, sia associata una chiave $key(x) \in \mathbb{N}$. Si indichi con una crocetta quali delle seguenti affermazioni sono vere

- se in T ogni nodo x è tale che $\texttt{key}(\texttt{left}(x)) < \texttt{key}(x) < \texttt{key}(\texttt{right}(x))$, allora T è un albero binario di ricerca;
- se in T ogni nodo x è tale che tutti i nodi dei sottoalberi di sinistra e di destra di x hanno chiave minore di $\texttt{key}(x)$, allora T è una max-heap;
- se T è una max-heap, allora in T ogni nodo x è tale che $\texttt{key}(\texttt{left}(x)) \leq \texttt{key}(x)$ and $\texttt{key}(\texttt{right}(x)) \leq \texttt{key}(x)$ e i sottoalberi sinistro e destro di x sono due max-heap.

Esercizio 8.

Si consideri la struttura dati di *albero binario di ricerca* (BST). Si indichi con una crocetta quali, tra le seguenti affermazioni, sono corrette:

- l'operazione di ricerca del massimo in un BST ha costo $\Omega(\log(n))$;
- nel caso migliore, l'operazione di ricerca del minimo in un BST ha costo $\Theta(1)$;
- se ogni nodo interno ha due figli, allora l'albero è bilanciato (i.e. ha altezza $O(\log(n))$);
- se T è un BST bilanciato, allora il massimo elemento in T ha profondità $\Omega(\log(n))$.

Esercizio 9.

Sia T un albero binario di ricerca avente n nodi. Si utilizzi il foglio allegato per scrivere una procedura che trasforma T in una max-heap, dimostrandone brevemente correttezza e complessità.

Esercizio 10.

Si consideri l'universo U delle date di nascita degli iscritti al Corso di Laurea in Informatica, presso l'Università di Udine. Si supponga che una data di nascita sia data dalla terna (i, j, k), con $i \in \{1 \dots 31\}$, $j \in \{1 \dots 12\}$, $k \in \{30 \dots 95\}$ (assumiamo che vi possano essere iscritti "molto giovani" e "molto vecchi"). Si considerino le seguenti funzioni di hashing: $h_i : U \mapsto \{0 \dots 999\}$:

- $h_1(i, j, k) = (i + j + k) mod (1000)$;
- $h_2(i, j, k) = (i * 10^4 + j * 10^2 + k) mod (1000)$.

Si risponda alle domande che seguono, motivando brevemente le risposte.

10.1) h_1 è una buona funzione di hashing?

_____

_____

_____

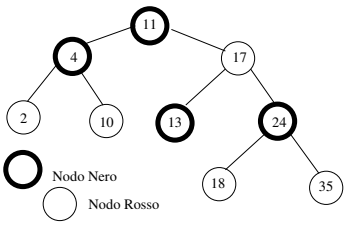
10.2) h_2 è una buona funzione di hashing?

10.3) si suggerisca una buona funzione di hashing per il problema dato, diversa da h_1, h_2

13.2 § Compitino di ASD del 30-06-05

Esercizio 1.

Si consideri il Red Black Tree, T, illustrato nel seguito. Si illustrino i Red Black Trees risultanti dalll'inserimento della chiave 40 in T, e dalla successiva cancellazione della chiave 11.



Esercizio 2.

Sia RB un RB-tree avente n nodi e radice r. Si consideri la seguente procedura.

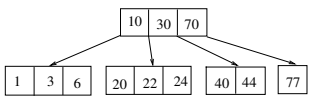
```
ALGO(r)
 1: if left(r)! = NIL and right(r)! = NIL then
 2:    ALGO(left(r))
 3:    ALGO(right(r))
 4:    if col(r) = RED then
 5:       if col(left(r)) = col(left(left(r))) = col(left(right(r))) = col(right(left(r))) = col(right(right(r))) then
 6:          col(r) ← BLACK
 7:          col(left(r)) ← RED
 8:          col(right(r)) ← RED
 9:       end if
10:    end if
11: end if
```

Si determini l'equazione ricorsiva di complessità $T(n)$ di ALGO(r) nel caso peggiore. Si risolva l'equazione.

Facoltativo: si dimostri che la procedura restituisce un RB-tree.

Esercizio 3.

Si illustri il B-tree risultante dall'inserimento della chiave 2 nel B-tree, di grado 2, illustrato nel seguito. Si illustri quindi il B-tree che si ottiene in seguito alla cancellazione della chiave 77.



Esercizio 4.

Mostrare un esempio di un B-tree T ottenuto applicando k volte la procedura di inserimento a partire da un B-tree vuoto e tale che ogni nodo di T è pieno. Che legame esiste tra k ed il grado t di T?

Esercizio 5.

Sia T un B-tree di grado t avente tutti i nodi pieni. Descrivere un algoritmo efficiente per determinare la chiave mediana di T. Dimostrare la correttezza dell'algoritmo proposto e valutarne la complessità.

Quale sarebbe il costo dell'eliminazione del mediano?

Esercizio 6.

Sia $G = (V, E, w)$ un grafo non orientato connesso e pesato. Sia T un minimum spanning tree di G. Si consideri il grafo G' ottenuto eliminando da E un arco (u, v) ed aggiungendo un arco (x, y) avente peso k, ovvero $G' = (V, E', w')$ in cui $E' = (E \setminus \{(u, v)\}) \cup \{(x, y)\}$ e

$$w'(a, b) = \begin{cases} w(a, b) & (a, b) \in E \\ k & \text{otherwise} \end{cases}$$

Descrivere un algoritmo efficiente per calcolare un minimum spanning tree di G'. Dimostrare la correttezza dell'algoritmo proposto e valutarne la complessità.

Esercizio 7.

Sia $G = \langle V, E \rangle$ un grafo *non orientato* e connesso, sui cui archi è definita la funzione di peso $\omega : E \mapsto \{1, 2\}$. Si consideri l'algortimo CYCLE2ALGO, illustrato nel seguito, per determinare se G contiene un ciclo con almeno un arco di peso 2.

1. L'algoritmo CYCLE2ALGO è corretto? Si motivi brevemente la risposta fornendo un'istanza su cui l'algoritmo non funziona correttamente, oppure una traccia di correttezza.
2. Si analizzi la complessità dell'algoritmo CYCLE2ALGO;

CYCLE2ALGO(G)

```
1: Let v to be a vertex of G;
2: A ← PRIM(G, ω, v);
3: for e ∈ E do
4:    if (e ∉ A ∧ ω(e) = 2) then return "Yes" endif
5: end for
6: return "No"
```

Esercizio 8.

Sia $G = (V, E, w)$ un grafo non orientato connesso e pesato tale che per ogni arco $(u, v) \in E$ vale $w(u, v) > 0$. Sia T un minimum spannig tree di G. Si consideri il grafo G' in cui ogni peso viene diminuito di 1 ovvero $G' = (V, E, w')$ con $w'(u, v) = w(u, v) - 1$ per ogni $(u, v) \in E$. Si dimostri o si trovi un controesempio alla seguente affermazione: T è un minimum spannig tree di G'.

Sia $G = (V, E, w)$ un grafo orientato e pesato tale che per ogni arco $(u, v) \in E$ vale $w(u, v) > 0$. Sia p un cammino di peso minimo da s a v in G. Si consideri il grafo G' in cui ogni peso viene diminuito di 1 ovvero $G' = (V, E, w')$ con $w'(u, v) = w(u, v) - 1$ per ogni $(u, v) \in E$. Si dimostri o si trovi un controesempio alla seguente affermazione: p è un cammino di peso minimo da s a v in G'.

Esercizio 9.

Si consideri il problema di mantenere in memoria insiemi disgiunti e di realizzare oltre alle operazioni MAKE, UNION e FIND l'operazione REMOVE che prende in input un elemento, lo rimuove dall'insieme a cui appartiene e lo mette in un insieme singoletto. Proporre opportune modifiche alle strutture dati per insiemi disgiunti che consentano di implementare in modo efficiente l'operazione REMOVE, senza modificare le complessità delle altre operazioni. Calcolare la complessità di REMOVE nell'implementazione proposta.

FACOLTATIVO... facile da fare ma difficile da spiegare REMOVE in cui il nodo porta con se tutti gli elementi che stavano nel suo insieme l'ultima volta che lui e' stato un rappresentante.

Esercizio 10.

Sia $G = \langle V, E \rangle$ un grafo *non orientato*. Si definisce *componente connessa* di G, un insieme massimale di nodi, $C \subseteq V$, tale che per ogni coppia di nodi $u, v \in C$, G contiene un cammino da u a v in G. Il grafo rappresentato in Figura 13.1, ad esempio, contiene 3 componenti connesse, come illustrato nella stessa Figura 13.1 (a destra).

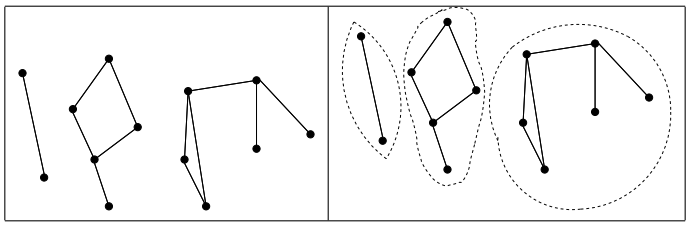


Figura 13.1:

L'algoritmo AlgoCC, illustrato nel seguito, utilizza le strutture dati per la gestione di insiemi disgiunti al fine di determinare le componenti connesse di un grafo non orientato G.

AlgoCC(G)

1: **for** each $v \in V$ **do**
2: Make-Set(v)
3: **end for**
4: **for** each $e \in E$ **do**
5: **if** Find-Set(u) $\neq$Find-Set(v) **then** Union(u, v) **endif**
6: **end for**

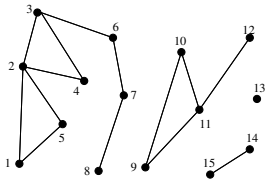
---

1. Si valuti la complessità[1] dell'algoritmo AlgoCC assumendo di implementare gli insiemi disgiunti mediante:

   (a) liste senza alcuna euristica;

   (b) liste con l'euristica di *weighted union*.

2. Si utilizzi lo spazio sottostante per delineare la foresta risultante dall'esecuzione dell'algoritmo AlgoCC sul grafo illustrato nel seguito (si assuma che gli archi vengano processati in ordine lessicografico i.e. nellordine:).



Esercizio 11.

Si illustri brevemente un algoritmo efficiente (i.e. avente complessità $O(|V| + |E|)$) per il problema della determinazione delle componenti connesse in un grafo non orientato $G = \langle V, E \rangle$.

[1] la complessità dev'essere naturalmente espressa rispetto ai parametri $|E|$ e $|V|$)

Esercizio 12.
Si definisca la foresta di alberi costruita dall'algoritmo DFS, eseguito sul grafo G, illustrato nell'esercizio (10) – (*b*). Si assuma che i nodi vengano processati in ordine crescente (rispetto alla loro etichetta), sia nell'analisi della lista di adiacenza di un vertice, che nel *loop* principale della procedura DFS(G).

13.3 § Esame di ASD del 27-07-05

Esercizio 1.
Si assuma che gli archi pesati di un grafo G siano memorizzati in un albero Rosso-Nero $\mathcal{T}$. Ogni nodo di $\mathcal{T}$ ha come chiave primaria il peso dell'arco e come informazione satellite i due nodi di G estremi dell'arco.

1-a Si scriva lo pseudocodice di una variante dell'algoritmo di Kruskal che sfrutta unicamente $\mathcal{T}$ come struttura dati (per per la scelta degli archi);

1-b si dimostri la correttezza e si calcoli la complessità della variante proposta.

Esercizio 2.
Si consideri l'albero binario $\mathcal{B}$ costruito dalla procedura CollectPivots, definita di seguito (Partition è la procedura utilizzata in QuickSort per partizionare il vettore $A[p \dots q]$ utilizzando $A[p]$ come elemento *pivot*).

2-a Si dimostri che l'albero $\mathcal{B}$ è un albero binario di ricerca;

2-b qual è l'altezza massima di $\mathcal{B}$? Quale l'altezza minima?

2-c (facoltativo) si determini il numero di nodi di $\mathcal{B}$.

Si motivi nel dettaglio ogni risposta fornita nei punti precedenti.

CollectPivots(A, p, r)
1: $T \leftarrow$ EmptyTree;
2: **if** $r \geq p$ **then**
3: $root[T] \leftarrow A[p]$;
4: **end if**
5: **if** $r > p$ **then**
6: $q \leftarrow$ Partition(A, p, r);
7: $T_1 \leftarrow$ CollectPivots(A, p, q);
8: $T_2 \leftarrow$ CollectPivots($A, q + 1, r$);
9: let T_1 to be the left subtree of $root[T]$;
10: let T_2 to be the right subtree of $root[T]$;
11: **end if**
12: **return** T

13.4 § Esame di ASD del 07-09-05

Esercizio 1. Scrivere una procedura che dato in input un albero rosso-nero T ed una chiave k appartenente a T produca in output un albero rosso-nero T' contenente tutte e sole le chiavi di T minori di k. Dimostrare la correttezza della procedura proposta e calcolarne la complessità.

Esercizio 2.
Scrivere una procedura che dato in input un grafo G non orientato connesso e pesato produca in output la risposta VERO se e soltanto se esistono due minimum spanning tree distinti per G. Dimostrare la correttezza della procedura proposta e calcolarne la complessità.

13.5 § Esame di ASD del 19-09-05

Esercizio 1.
Sia G una foresta di alberi in cui ogni nodo ha solo il puntatore al proprio genitore.

1.a Scrivere una procedura che dati in input due nodi u e v di G determini se u e v appartengono allo stesso albero. Dimostrare la correttezza della procedura proposta e calcolarne la complessità.

1.b Estendere la procedura proposta in modo che se u e v appartengono allo stesso albero venga prodotto in output l'antenato comune a u e v che si trova a profondità massima. Dimostrare la correttezza della procedura estesa e calcolarne la complessità.

Esercizio 2.
Sia $G = (N, E)$ un grafo non orientato e connesso.

2.a Descrivere un algoritmo che determini se la rimozione di un qualsiasi arco da G sconnette G. Dimostrare la correttezza e determinare la complessità dell'algoritmo proposto.

2.b Descrivere un algoritmo che determini tutti gli archi (u, v) tali che $G' = (N, E \setminus \{(u, v)\})$ è sconnesso. Dimostrare la correttezza e determinare la complessità dell'algoritmo proposto.

13.6 § Esame di ASD del 07-12-05

Esercizio 1.
Dati due numeri interi x ed y definiamo la distanza tra x ed y come $d(x, y) = |x - y|$.

1.a Sia T un albero binario di ricerca le cui chiavi sono numeri interi. Si descriva un algoritmo per determinare due elementi di T aventi distanza minima.

1.b Sia A un vettore di interi. Si descriva un algoritmo per determinare due elementi di A aventi distanza minima.

1.c Sia A un vettore di interi tale che la distanza tra due elementi di A è sempre minore o uguale di una costante k. Si descriva un algoritmo per determinare due elementi di A aventi distanza minima.

Si dimostri la correttezza e si determini la complessità degli algoritmi proposti.

Esercizio 2.
Sia $G = (N, E, W)$ un grafo orientato pesato e sia $s \in N$ un nodo di G tale che ogni altro nodo è raggiungibile da s. Dato un cammino $p = (v_0, v_1, \ldots, v_k)$ definiamo il peso di p come il massimo tra i pesi degli archi che compongono p, ovvero:

$$w(p) = \max(w(v_i, v_{i+1}))_{i \in [0,k-1]}$$

2.a Si descriva un algoritmo per deteminare l'albero dei cammini minimi radicato in s. Si dimostri la correttezza e si determini la complessità dell'algoritmo proposto.

2.b Nel caso in cui G sia aciclico si descriva un algoritmo per determinare l'albero dei cammini massimi radicato in s. Si dimostri la correttezza e si determini la complessità dell'algoritmo proposto.

Capitolo 14

ANNO 2003-2004

14.1 § Esame di ASD del 13-12-04

Esercizio 1.
Sia $A[n, n]$ una matrice quadrata di interi distinti. Si supponga che le righe di A siano ordinate in modo crescente da sinistra verso destra. Si assuma, inoltre, che le colonne di A siano ordinate in modo crescente dal basso verso l'alto. Si consideri il problema di determinare se, dato un intero k, k compare in A.

(1.a) Si definisca un algoritmo di complessitá strettamente inferiore ad $\mathcal{O}(n^2)$ per il problema proposto;

(1.b) si valuti la correttezza e la complessità dell'algoritmo definito al passo precedente.

Esercizio 2.
Sia $G = \langle V, E\rangle$ un grafo non orientato sui cui archi é definita la funzione di peso $w : E \mapsto \mathbb{N}$. Si consideri una versione modificata dell'algoritmo di Kruskal, che *non* esegue alcun ordinamento preliminare degli archi di G.

(2.a) che cosa restituisce in output l'algoritmo delineato?

(2.b) si produca lo pseudocodice della procedura proposta e se ne valuti la complessità, assumendo che la disjoint set forest, utilizzata nell'algoritmo, venga implementata mediante alberi con le euristiche di union by rank e path compression.

14.2 § Esame di ASD del 06-09-04

Esercizio 1.
Dato un array $A[1 \dots n]$, contenente n interi, si consideri il problema di determinare se A contiene più di $\lceil \frac{n}{2} \rceil$ ripetizioni di un suo elemento.

(1.a) Si definisca un algoritmo efficiente per il problema proposto;

(1.b) si valuti la correttezza e la complessità dell'algoritmo proposto al passo precedente.

Esercizio 2.
Sia $G = \langle V, E\rangle$ un grafo orientato e si assuma che i nodi in G siano colorati mediante il colore rosso oppure verde. Si consideri il problema di determinare se G possiede un ciclo semplice contenente *solo* nodi verdi.

(2.a) Si proponga un algoritmo efficiente per il problema proposto;

(2.b) si valuti la correttezza e la complessità della procedura delineata;

(2.c) **Facoltativo**: si riconsiderino i punti precedenti nell'ipotesi di determinare se G possiede un ciclo semplice contenente *almeno* un nodo verde.

14.3 § Esame di ASD del 20-07-04

Esercizio 1.
Sia $A[1 \dots n]$ un vettore contenente $n = 3m$ interi. Si consideri il problema di determinare gli elementi di A maggiori o uguali ad almeno $\frac{n}{3}$ interi in A e minori o uguali ad almeno $\frac{n}{3}$ interi in A.

- Si proponga un algoritmo lineare per risolvere il problema proposto;
- si discuta la correttezza e la complessità dell'algoritmo definito.

Esercizio 2.
Si consideri un grafo diretto e pesato $G = \langle V, E \rangle$. Si assuma che la funzione di peso, $w : E \mapsto \mathbb{N}$, assegni ad ogni arco $e \in E$ un peso $1 \leq w(e) \leq k$, dove k é una costante.

- Si proponga un algoritmo efficiente per determinare il peso di un cammino minimo da un nodo $s \in V$, ad un nodo $v \in V$;
- si analizzi la correttezza e la complessità dell'algoritmo proposto.
- **Facoltativo** Si proponga una versione modificata dell'algoritmo di Dijkstra che operi in modo lineare nelle ipotesi delineate sopra.

14.4 § Esame di ASD del 01-07-04

Esercizio 1.
Si consideri l'algoritmo di ordinamento Counting-Sort visto a lezione.

1-a Si proponga una versione di Counting-Sort che utilizzi soltanto un vettore ausiliario (il vettore C, nella terminologia del testo adottato a lezione), in luogo dei due vettori utilizzati nella versione classica (i vettori B e C).

1-b Rispetto a quale parametro l'algoritmo delineato peggiora le performance della versione di Counting-Sort vista a lezione?

Esercizio 2:
Si consideri la struttura dati *Binary Search Tree* (BST) ed il problema di determinare la chiave media mantenuta in un BST.

2-a Si proponga un algoritmo efficiente che, dato un BST T contenente n chiavi distinte, determini la chiave media in T (i.e. la chiave k tale che T contiene $\lfloor \frac{n}{2} \rfloor$ chiavi minori di k)

2-b Si proponga una struttura dati che permetta di mantenere un insieme di interi distinti e di gestire, in modo efficiente, le seguenti operazioni:

- inserimento di un intero k nella struttura dati;
- cancellazione di un intero k dalla struttura dati;
- determinazione dell'elemento medio mantenuto nella struttura dati.

Si descriva brevemente e si valuti la complessità di ognuna delle operazioni delineate sulla struttura dati proposta.

Capitolo 15

ANNO 2002-2003

15.1 § Esame di ASD del 09-12-03

Esercizio 1.
Si consideri il problema di definire delle funzioni di hash che mappano gli elementi dell'universo $\mathtt{U} = \{1 \dots N\}$ nelle celle di una tabella di hash $\mathtt{T}[0 \dots m]$. Dopo aver elencato le caratteristiche di una buona funzione di hash, per ognuna delle seguenti funzioni h_i, si dica se h_i può essere considerata una ragionevole funzione di hash, giustificando la risposta.

- $h_1(i) = (i \mod m) + 3$
- $h_2(i) = (2 * i) \mod m$
- $h_3(i) = (max(N - m, i)) \mod m$
- $h_4(i) = ((i \mod m) + 127) \mod m$

Esercizio 2.
Sia $G = \langle V, E \rangle$ un grafo non orientato, connesso e pesato. Si supponga di aver determinato un *minimum spanning tree* di G, T, avente radice v. Si assuma che il peso dell'arco $(u, v) \in E$ venga decrementato di 1 e si consideri il problema di determinare, dato T, un minimum spanning tree del grafo modificato.

- Si definisca un algoritmo efficiente per il problema proposto e se ne valutino correttezza e complessità.

(**Suggerimento**: si pensi ad un taglio che separi il nodo u dal nodo v ...)

15.2 § Esame di ASD del 02-09-03

Esercizio 1.
Dato un array ordinato contenente n interi, $A[1 \dots n]$, si consideri il problema di costruire un *Red Black Tree* contenente $A[1], \dots, A[n]$ come chiavi. Si proponga un algoritmo efficiente per il problema proposto e se ne dimostri la correttezza. Si valuti infine la complessitá della procedura proposta.

Esercizio 2.
Sia dato un grafo $G = \langle V, E \rangle$ non orientato e si considerino i seguenti problemi:

a) stabilire se G é un *albero*;
(**Suggerimento**: si osservi che, ovviamente, se esiste $r \in V$ tale che G é un albero di radice r, allora per ogni $r' \in V$, G é un albero di radice r')

b) dato $k \in \mathbb{N}$, stabilire se esiste un nodo $r \in V$ tale che G é un albero di altezza k e radice r.

Si definisca un algoritmo efficiente per ciascuno dei problemi proposti. Si dimostri la correttezza delle procedure delineate e se ne valuti la complessitá.

15.3 § Esame di ASD del 07-07-03

Esercizio 1.
Si consideri il problema di ordinare un array $A[1 \dots n]$ contenente al più k interi distinti, dove k è una costante.

- Si proponga un algoritmo efficiente per il problema proposto e se ne valuti la complessità.
- Si proponga un algoritmo efficiente ed *in place* per il problema proposto e se ne valuti la complessità.
- **Facoltativo** Si proponga un algoritmo *in place* e *stabile* per il problema proposto e se ne valuti la complessità.

Esercizio 2.
Si consideri un grafo, $G = \langle V, E \rangle$, orientato ed aciclico. Dato un vertice $v \in V$, sia *d-foglie*(v) il minimo numero di archi in un cammino da v ad una foglia del grafo. Si denoti invece mediante *D-foglie*(v), il massimo numero di archi in un cammino da v ad una foglia del grafo.

- Si proponga un algoritmo efficiente che, dato un grafo orientato aciclico G ed un nodo v, determini *d-foglie*(v). Si valuti la complessità della procedura proposta.
- Si proponga un algoritmo efficiente che, dato un grafo orientato aciclico G ed un nodo v, determini *D-foglie*(v). Si valuti la complessità della procedura proposta.
- **Facoltativo** Si proponga un algoritmo efficiente che associ ad *ogni* vertice di un grafo orientato ed aciclico G, i valori *D-foglie* e *d-foglie*.

15.4 § Esame di ASD del 02-12-02

Esercizio 1.
Si consideri un grafo non orientato e pesato $G = \langle V, E \rangle$ e si considerino gli algoritmi di Prim e Kruskal visti a lezione. Si supponga G non connesso.

1 Qual è la l'output prodotto da ognuno dei due algoritmi menzionati su G?

1-b Qual è la complessità dei due algoritmi menzionati su G?

Si giustifichino le risposte.

Esercizio 2.
Si supponga di rappresentare mediante una coppia di interi $[l, u]$ con $l \leq u$, l'intervallo di interi compresi tra l ed u (estremi inclusi). Sia dato l'insieme di n intervalli $S = \{[l_1, u_1], [l_2, u_2], \dots, [l_n, u_n]\}$.

2-a Si descriva un algoritmo efficiente per determinare se esistono (almeno) due intervalli disgiunti in S.

2-b Si delinei lo pseudocodice dell'algoritmo descritto al punto precedente e se ne studi la complessità.

2-c (Facoltativo.) Si riconsiderino i precedenti due punti nel caso il problema sia quello di determinare se esistono due intervalli non disgiunti.

Capitolo 16

ANNO 2001-2002

16.1 § Esame di ASD del 26-09-02

Esercizio 1.
Sia A un vettore di n interi e si consideri il problema di determinare se esistono due interi che occorrono in A lo stesso numero di volte.

1-a Si descriva un algoritmo efficiente per il problema proposto nel caso in cui in A occorrono c valori distinti, dove c è una costante intera positiva.

1-b Si descriva un algoritmo efficiente per il problema proposto.

1-c Si determini la complessità degli algoritmi proposti.

Esercizio 2.
Siano B_1 e B_2 due B-alberi di grado t, tali che per ogni chiave k_1 in B_1 e k_2 in B_2 si abbia

$$k_1 \leq k_2.$$

2-a Si descriva un algoritmo efficiente che, prendendo in input B_1 e B_2, esegua la *fusione* dei due B-alberi e ritorni in output un B-albero B contenente tutte le chiavi precedentemente in B_1 e B_2 (che vengono eliminati).

2-b Si determini la complessità computazionale dell'algoritmo proposto valutando, in particolare, il numero massimo di accessi alla memoria secondaria nell'ipotesi solo un numero costante di nodi di un B-albero di grado t possa risiedere contemporaneamente in memoria centrale.

16.2 § Esame di ASD del 02-09-02

Esercizio 1.
Sia A un vettore di n interi compresi tra 1 e una costante intera positiva c.

1-a Si descriva un algoritmo efficiente che ordini A.

1-b Si descriva un algoritmo efficiente *in place* che ordini A.

1-c Si determini la complessità degli algoritmi proposti in funzione sia di n che di c e si valuti la complessità di tali algoritmi nei casi in cui $c = O(\lg n)$ e $c = O(n)$.

Esercizio 2.
Sia $G = (V, E)$ un grafo non orientato e pesato, con funzione di peso $w : E \to \Re^+$.

2-a Si descriva brevemente un algoritmo efficiente (per esempio tra quelli visti a lezione) che determini un *Minimum Spanning Tree* per G e se ne valuti la complessità.

2-b Si provi o si refuti la seguente proprietà: dato un *Minimum Spanning Tree* per G questo contiene almeno un arco tra quelli aventi peso minimo in G.

2-c Si provi o si refuti la seguente proprietà: dato un *Minimum Spanning Tree* per G questo contiene un cammino minimo per ogni coppia di nodi in G.

16.3 § Esame di ASD del 18-07-02

Esercizio 1.
Sia A un vettore di $n = 2^k$ interi e si supponga di disporre di un algoritmo `Mediano`(A, i, j) che restituisce l'elemento mediano della sequenza $A[i..j]$, ovvero l'elemento di posizione $\lfloor (i + j)/2 \rfloor$ nella sequenza ordinata degli elementi di $A[i..j]$.

1-a Scrivere lo pseudo-codice di un algoritmo efficiente `Select`(A, i) che determini l' i-esimo piu' piccolo elemento di A.

2-a Si dimostri la correttezza e si valuti la complessita' dell'algoritmo proposto, supponendo che l' algoritmo `Mediano`(A, i, j) abbia complessita' $O(j - i)$.

Esercizio 2.
Sia $G = (V, E)$ un grafo orientato.

2-a Si proponga un algoritmo efficiente che determini se G è un albero, valutandone la complessità e dimostrandone la correttezza.

2-b (Facoltativo) Si proponga un algortimo efficiente che determini se G è trasformabile in un albero mediante l'eliminazione di un opportuno numero di archi. Si valuti la complessità e si dimostri la correttezza dell'algoritmo proposto.

16.4 § Esame di ASD del 18-06-02

Esercizio 1. Sia A un array contentente $n = 2m$ interi:

1-a Si proponga un algoritmo che, preso in input A, produca come output un array in cui gli elementi in posizione pari costituiscano una successione crescente e quelli in posizione dispari costituiscano una successione decrescente.

1-b Si discuta la correttezza e si valuti la complessità dell'algoritmo proposto.

1-b Si riconsideri il punto precedente e si proponga un algoritmo che svolga la stessa funzione ma che, in piú, operi *in-place*.

Esercizio 2.
Sia $G = (V, E)$ un grafo non orientato, connesso e con un numero di archi uguale al numero di nodi.

2-a Si proponga un algoritmo efficiente che determini un Minimum Spanning Tree di G, valutandone la complessità e dimostrandone la correttezza.

2-b Diciamo *Secondo Migliore* Minimum Spanning Tree di G, uno Spanning Tree T' di G tale che $T < T' \leq T''$ per ogni Minimum Spanning Tree T di G e per ogni Spanning Tree T'' di G.

Si proponga un algoritmo efficiente che determini, se esiste, un Secondo Migliore Minimum Spanning Tree di G valutandone la complessità e dimostrandone la correttezza.

Capitolo 17

ANNO 1999-2000

17.1 § Compitino di ASD del 01-06-00

Esercizio 1.
Sia $G = (V, E)$ un grafo pesato ed orientato, in cui la funzione peso $w : E \to \mathcal{R}$ assume un valore negativo su di un unico arco $(u, v) \in E$.

1-a Si proponga un algoritmo efficiente che determini se nel grafo esiste un ciclo negativo.

1-b Si discuta la correttezza e si calcoli la complessità dell'algoritmo proposto.

1-c Si considerino i punti precedenti nel caso in cui la funzione peso w assuma valore negativo su due archi $(u_1, v_1), (u_2, v_2)$ del grafo G.

1-d (Facoltativo) Si suggerisca l'idea per un algoritmo effeciente che determini l'esistenza di cicli negativi in grafi in cui il numero di archi negativi sia molto limitato, per esempio sia minore di $\ln n$ dove n e' il numbero dei nodi del grafo.

Esercizio 2.
Si consideri un *binary search tree* T.

2-a Si proponga un algoritmo che determini se è possibile "colorare" i nodi di T in modo da renderlo un *red-black tree*.

2-b Si proponga un algoritmo che dato in input T lo trasformi in un red-black tree.

2-c Si provi la correttezza e si valutino le complessità degli algoritmi proposti.

17.2 § Compitino di ASD del 22-2-00

Esercizio 1.
Definiamo *heap ternario* la naturale modifica dello heap (binario); ossia un albero ternario, (ogni nodo ha tre figli, eventalmente vuoti) in cui la chiave associata al padre è maggiore delle chiavi associate ai tre figli.

(a) Si proponga un implementazione dello heap ternario traminte un vettore. In particolare si scrivano le procedure: `Parent(i), Left(i), Heapify(A,i) e Build-Heap(A)`.

(b) (Facoltativo) Si discuta il punto (a) nel caso di un *heap esponenziale* ossia di un heap dove ogni nodo di altezza h, ha 2^h figli. Qual è la complessità di un algoritmo di heap-sort che utilizzi un heap esponenziale? Si motivino le risposte.

Esercizio 2.

Diciamo *propriamente bitonica* una sequenza S di numeri interi $a_1, \ldots, a_n$ per cui esista $k \in \{1, \ldots, n\}$ tale che $a_1, \ldots, a_k$ sia monotona crescente (decrescente) e $a_k, \ldots, a_n$ sia monotona decrescente (crescente). Diremo *bitonica* una sequenza di interi che possa essere resa propriamente bitonica mediante una permutazione ciclica degli indici.

Esempi: $\{3, 5, 7, 11, 12, 8, 4, -2, -11\}$ sequenza (propriamente) bitonica; $\{7, 11, 12, 8, 4, -2, -11, 3, 5\}$ sequenza bitonica.

Una sequenza bitonica è una sequenza con al più un *picco* (massimo locale interno) e al più una *valle* (minimo locale interno)

(a) Si proponga un algoritmo che, data in input una sequenza bitonica S, produca in output due sequenze bitoniche S_1 ed S_2 di dimensione $n/2$ tali che

$$S_1 \cap S_2 = \emptyset, \quad S_1 \cup S_2 = S, \quad \forall s_1 \in S_1, \forall s_2 \in S_2 (s_1 \leq s_2)$$

(b) Si provi la correttezza e si valuti la complessità dell'algoritmo proposto.

17.3 § Esame di ASD del 14-12-00

Esercizio 1.

Sia x un array $(x_1, \ldots, x_n)$ di interi e sia p una permutazione degli indici $\{1, \ldots, n\}$. Si consideri il seguente algoritmo:

```
for j := 1 to n do
  k := p(j);
  while k > j do
    k := p(k);
  end while
  if k = j then
    y := x[j]; l := p(k);
    while l ≠ j do
      x[k] := x[l]; k := l; l := p(k);
    end while
    x[k] := y;
  end if
end for
```

1-a Si commenti **Algoritmo 1** e si provi che ricevuto in input x produce in output l'array $(x_{p(1)}, \ldots, x_{p(n)})$.

1-b Si valuti la complessità di **Algoritmo 1**.

1-c (Facoltativo) Si produca un input relativamente al quale la complessità di **Algoritmo 1** sia pessima.

Esercizio 2.

2-a Si proponga un algoritmo che, dato un grafo G, pesato, orientato e aciclico, e un suo nodo v, determini, tra i cammini uscenti da v, quello di lunghezza massima.

2-b Si discuta la correttezza e si calcoli la complessità dell'algoritmo proposto.

(**Suggerimento**: modificando il peso degli archi, si riduca il proplema ad un problema di ricerca di cammino minimo)

17.4 § Esame di ASD del 29-11-00

Esercizio 1.

Un vettore S di n numeri interi si defisce *k-quasi ordinato* se esiste un vettore ordinato S' che differisce da S per al più k elementi, con k costante.

1-a Si proponga un algoritmo lineare per l'ordinamento di vettori k-quasi ordinati.

1-b Si discuta la correttezza dell'algoritmo proposto.

1-c (facoltativo) Si proponga un algoritmo lineare di ordinamento per il caso in cui il vettore S differisca da un vettore ordinato per al più $\sqrt{n}$ elementi.

Esercizio 2.

Si consideri il problema *Union-Find* nell'ipotesi gli elementi siano numeri interi e gli insiemi siano, quindi, insiemi disgiunti di numeri interi.

Si considerino le usuali operazioni **Make**, **Union** e **Find** e si consideri inoltre l'operazione

Split che prende in input un elemento x ed un insieme A tale che $x \in A$, e torna in output:

- i due insiemi $A' = \{y \in A \mid y \leq x\}$ e $A'' = \{y \in A \mid y > x\}$, se entrambi A' e A'' sono non vuoti,
- A altrimenti.

2-a Si propongano delle strutture dati atte a supportare le suddette operazioni.

2-b Si discuta la correttezza e si valutino le complessità degli algoritmi proposti.

17.5 § Esame di ASD del 14-09-00

Esercizio 1.

1-a si proponga un algoritmo che dati due insiemi di numeri interi A, B, determini la coppia di elementi $a \in A$ e $b \in B$ per cui l'espressione $\mid a - b \mid$ assuma il valore minimo.

1-b Si discuta la correttezza dell'algoritmo proposto e se ne valuti la complessità.

1-c Risolvere i punti precedenti nel caso l'algoritmo abbia come dato di ingresso tre insiemi distintinti A, B, C e debba trovare gli elementi $a \in A, b \in B$ e $c \in C$ per cui l'espressione $\mid a - b \mid + \mid b - c \mid$ assuma valore minimo.

Esercizio 2.

Sia $G = (V, E)$ un grafo orientato e pesato e sia $s \in V$ un nodo sorgente.

2-a Si proponga un algoritmo che risolva il *single source shortest path problem* a partire da s nell'ipotesi in cui i pesi degli archi possano essere solo 1 o 2.

2-b Si discuta la correttezza e si calcoli la complessità dell'algoritmo proposto.

2-c (Facoltativo). Si riconsideri il problema nel caso i pesi possano essere nell'insieme $\{1, \ldots, k\}$, con k costante fissa.

17.6 § Esame di ASD del 31-08-00

Esercizio 1.

Si consideri un array A contenente n interi (a due a due distinti) quale input per l'algoritmo di ordinamento `Heapsort`.

1-a Qual è la complessità di `Heapsort` quando gli interi in A costituiscono già una sequenza ordinata? Si giustifichi la risposta.

1-b Si supponga che A sia sempre costituito da un segmento finale di lunghezza almeno $\lfloor n/2 \rfloor$ già ordinato, e si ottimizzi `Heapsort` in modo da migliorarne la complessità su A.

Esercizio 2.

Sia $G = (V, E)$ un grafo non orientato.

2-a Si proponga un algoritmo che determini se ogni coppia di vertici in G sia collegabile con un cammino contenente al più quattro archi.

2-b Si proponga un algoritmo che determini l'esistenza di uno spanning tree T per il grafo G tale ogni coppia di vertici sia collegabile con un cammino, in T, contenente al più quattro archi. (Suggerimento: in un albero ogni coppia di vertici sono collegabili con un cammino contenente al più quattro archi se e solo se esiste un nodo raggiungibile da ogni altro nodo percorendo al più due archi.)

2-c Si discuta la correttezza e si calcoli la complessità degli algoritmi proposti.

17.7 § Esame di ASD del 13-07-00

Esercizio 1.
Sia $G = (V, E)$ un grafo pesato ed orientato, in cui la funzione peso $w : E \to \mathcal{R}$ assume solo valori positivi e sia $v \in V$ un nodo del grafo.

1-a Si proponga un algoritmo che determini se esistono cicli a cui v appartine e nel caso trovi quello di peso minimo.

1-b Si discuta la correttezza e si calcoli la complessità dell'algoritmo proposto.

1-c (Facoltativo) Si proponga un algoritmo che determini il ciclo di peso minimo in G.

Esercizio 2.
Sia T un albero binario di ricerca privo di nodi aventi la stessa chiave ($key[\cdot]$),

2-a Si proponga un algoritmo efficiente che dati due nodi a e b in T ritorni la lista dei nodi x tali che $key[a] \leq key[x] \leq key[b]$.

2-b Si discuta la correttezza e si calcoli la complessità dell'algoritmo proposto.

2-c (Facoltativo) Si modifichi l'algoritmo di cui sopra in modo che ritorni il nodo avente chiave mediana fra le chiavi relative ai nodi della lista ritornata in output.

17.8 § Esame di ASD del 22-06-00

Esercizio 1.
Si consideri una struttura dati S che gestisca un insieme di numeri naturali e implementi le tre seguenti operazioni:

- `Insert(n)`, inserisce il numero naturale `n` all'interno della struttura dati S,
- `Extract_min`, estrae l'elemmento minimo da S
- `Extract_min_even`, estrae da S il più piccolo numero pari.

1-a Si proponga un'implementazione per la struttura dati S.

1-b Si analizzi la complessità degli algoritmi proposti e se ne motivi la correttezza.

1-c Si riconsiderino i due punti precedenti nel caso in cui S implementi anche l'operazione `Add(i)` che somma il numero naturale `i` ad ogni elemento contenuto in S. Si cerchi di proporre un implementazione in cui `Add(i)` abbia costo costante.

Esercizio 2.
Sia $G = (V, E)$ un grafo orientato,

2-a Si proponga un algoritmo efficiente che determini se G è un albero.

2-b Si discuta la correttezza e si calcoli la complessità dell'algoritmo proposto.

2-c (Facoltativo) Si proponga un algoritmo che, nell'ipotesi G non sia un albero, stabilisca qual è il numero minimo di archi che vanno aggiunti/eliminati per trasformare G in un albero. Si discuta la correttezza e si calcoli la complessità dell'algoritmo proposto.

17.9 § Esame di ASD del 14-2-00

Esercizio 1.
In una sequenze di n elementi si definisce *elemento medio* l'elemento nella posizione $\lfloor n/2 \rfloor$.

(a) Proporre una struttura dati che permetta di realizziare, in maniera efficiente, le seguenti operazioni:

- `Insert(a)`: Inserisce un nuovo elemento all'interno della struttura
- `Extract_Min`: Estrae l'elemento con chiave minima.
- `Find_Med`: Trova (ma non estrae) l'elmento medio rispetto all'ordine sulle chiavi.

(b) Si discuta la correttezza degli algoritmi proposti e se ne valuti la complessità.

Suggerimento: Una possibile soluzione consiste nel far uso di alberi bilanciati.

Esercizio 2.
Sia $G = (V, E)$ un grafo orientato rappresentato mediante liste di adiacenza. Ricordiamo che un *ordinamento topologico* dei nodi di G è un ordinamento $\prec$ degli elementi di V tale che, per ogni $(u, v) \in E$, si abbia che $u \prec v$.

(a) Si proponga un algoritmo che preso in input G, produca un ordinamento topologico dei nodi di G.

(b) Si modifichi il precedente algoritmo in modo tale che, se esistono più ordinamenti topologici di G, due successive chiamate dell'algoritmo producano differenti ordinamenti.

(c) Si discuta la correttezza degli algoritmi proposti e se ne valuti la complessità.

17.10 § Esame di ASD del 24-1-00

Esercizio 1.
Dato un insieme S contenente n numeri interi, si considerino i seguenti due problemi:

- determinare la coppia di elementi s_1, s_2 in S tale che il valore $|s_1 - s_2|$ risulti minimo;
- determinare la tripla di elementi s_1, s_2, s_3 in S tale che la somma $|s_1 - s_2| + |s_2 - s_3| + |s_1 - s_3|$ risulti minima.

(a) Si scriva lo pseudo-codice di due algortimi che risolvano i i due problemi precedenti;

(b) si determini e si giusitfichi la complessità degli algoritmi proposti.

Esercizio 2.
Si consideri un B-albero T con di grado t con n chiavi:

(a) si proponga lo pseudo-codice di un algoritmo per l'inserimento di una chiave k in T che *minimizzi il numero di nuovi nodi generati* (**suggerimento**: si consideri una variante in due passi dell'algoritmo di inserimento visto a lezione);

(b) si discuta la correttezza e si valuti la complessità dell'algoritmo proposto in termini dei parametri t e k.

Capitolo 18

ANNO 1998-1999

18.1 § Compitino di ASD del 13-5-99

Esercizio 1.
Definiamo LRB-tree (*Loose* Red-Black tree) come un albero binario di ricerca in cui:

i) tutti i nodi sono etichettati come neri o rossi,

ii) ogni foglia (*nil*) è nera,

iii) se un nodo x è rosso ed il padre di x è rosso allora entrambi i figli di x sono neri,

iv) ogni cammino semplice dalla radice ad una foglia contiene lo stesso numero di nodi neri.

(a) Si mostri un LRB-tree che non sia un RB-tree.

(b) Qual e' l'altezza massima di un LRB-tree con n nodi?

(c) Se si applica l'algoritmo RB-Insert ad un LRB-Tree l'albero risultate e' un LRB-tree?

Si motivino le risposte.

Esercizio 2.
Si consideri un grafo pesato e orientato $G = (V, E)$ con funzione di peso $w : \Re \to E$

(a) Si proponga un'algoritmo che determini tutti i nodi x che si trovano su un ciclo negativo e, per ognuno di tali x, produca in output un ciclo negativo che contenga x.

(b) Si discuta la correttezza e si valuti la complessità dell'algoritmo proposto.

18.2 § Compitino di ASD del 23-2-99

Esercizio 1.
Sia dato un vettore $A[1..n]$ di interi positivi.

(a) Si scriva lo pseudo-codice di un algoritmo efficiente che trovi un indice k tale che:

$$| \sum_{i=1}^{k-1} A[i] - \sum_{j=k+1}^{n} A[j] \ | \leq \ A[k].$$

(b) Si giusitfichi e si determini la complessità dell'algoritmo proposto.

(c) Si riconsiderino i quesiti (a) e (b) nell'ipotesi di dover cercare un elemento A_k tale che:

$$| \sum_{A[i]<A[k]} A[i] - \sum_{A[k]<A[j]} A[j] | \leq A[k].$$

(d) (Facoltativo) Una algoritmo soddisfacente per il problema al punto (c) richiede tempo $\Theta(n \log n)$, nel casa peggiore. Si discuta l'esistenza di soluzioni con un limite di complessità media più basso.

Esercizio 2.

Si considerino gli algoritmi *build_heap*, *extract_min* e *insert* definiti per operare su una *min_heap* H di profondità h e contenente n nodi.

(a) Si proponga una variante della struttura dati *min_heap* atta a supportare le suddette procedure e anche una procedura *extract_max* di estrazione del massimo.

(b) Si discuta la correttezza e si valutino le complessità degli algoritmi proposti in funzione dei parametri h ed n.

18.3 § Esame di ASD del 30-9-99

Esercizio 1.

Sia S una stuttura dati, formata da un insieme di numeri naturali, su cui sia possibile eseguire le seguenti tre operazioni.

- Insert(n), inserisce il numero naturale n all'interno della struttura dati,
- Extract(k), estrae dalla struttura dati il più piccolo multiplo di k in essa contenuto.
- Delete(k), rimuove dalla struttura dati tutti i multipli di k contenuti in essa.

Si supponga inoltre che il valori di k sia limitato da una costante c.

a Si proponga una implementatizione per la struttura dati S.

b Si analizzi la complessita' degli algoritmi proposti e se ne discuta la correttezza.

Esercizio 2.

Sia $G = (V, E)$ un grafo non orientato pesato, tale che ogni arco abbia peso negativo.

a Si proponga un algoritmo che risolva il problema della determinazione dei cammini minimi da una data sorgente $s \in V$.

b Si discuta la correttezza e si valuti la complessità dell'algoritmo proposto.

c Si riconsideri il precedente problema nel caso G sia orientato

18.4 § Esame di ASD del 8-7-99

Esercizio 1.

Dato un numero naturale n, si consideri un vettore A contenente n valori compresi tra 1 ed 2^n, e tale che in ogni intervallo $[2^m, 2^{(m+1)}]$ ci siano al più k valori di A, con k costante.

(a) Si scriva lo pseudo codice di un algorimo che ordini il vettore A.

(b) Si discuta la correttezza dell'algoritmo proposto e se ne valuti la complessità, nell'ipotesi ogni operazione sugli interi abbia costo costante e nell'ipotesi i numeri siano rappresentati in base 2 e le operazioni fondamentali sugli interi abbiano un costo proporzionale al loro numero di cifre.

(c) Si confronti la complessità dell'algoritmo proposto con quella di merge-sort.

(d) Si riconsiderino i punti (a) e (b) nel caso in cui ogni intervallo $[2^m, 2^{(m+1)}]$ contenga al più $\ln n$ valori di A.

(e) Si riconsiderino i punti (a) e (b) nel caso in cui ogni intervallo $[2^m, 2^{(m+1)}]$ contenga al più $\sqrt{n}$ valori di A.

Esercizio 2.

Sia $G = (V, E)$ un grafo non orientato pesato (pesi positivi e negativi sono ammessi), tale che ogni ciclo non contiene archi negativi.

(a) Si proponga un algoritmo che risolva il problema della determinazione dei cammini minimi da una data sorgente $s \in V$.

(b) Si discuta la correttezza e si valuti la complessità dell'algoritmo proposto.

18.5 § Esame di ASD del 8-7-99

Esercizio 1.

Dati un numero naturale m e una matrice $A = (a_{i,j})$ quadrata di dimesione n, avente come valori numeri interi distinti e soddisfacente la sequente proprieta':

$$\forall i, i', j, j' \, (\, i < i' \Rightarrow a_{i,j} < a_{i',j'} \,),$$

(a) si scriva lo pseudocodice di un algoritmo che, considerando l'ordine sugli interi, determina l'm-esimo elemento della matrice A;

(b) si discuta la correttezza e si valuti la complessità dell'algoritmo proposto.

Esercizio 2.

Dato un RB-albero, si consideri il problema di mantenere un campo *black-height[x]* che, per ogni nodo x appartenente all'albero, contenga l'altezza nera di x.

(a) Si proponga lo pseudocodice commentato e dettagliato di opportune versioni delle procedure RB-TREE-INSERT e RB-TREE-DELETE atte a garantire il mantenimento del campo *black-height*;

(b) si discuta la correttezza e si valutino le complessità delle procedure proposte;

(c) si discuta brevemente il problema di mantenere un campo *height* che contenga l'altezza.

18.6 § Esame di ASD del 15-6-99

Esercizio 1.

Sia A una matrice $n \times n$ di numeri interi.

(a) Si dia lo psedocodice di un algoritmo che determina l'elemento di A che appartiene al maggior numero di righe, nell'ipotesi che ogni riga contenga elementi tutti distinti.

(b) Si risolva il punto precedente per una generica matrice.

(c) Si discuta la correttezza e si valuti la complessità degli algoritmi proposti.

Esercizio 2.

Dato un B-albero di grado t si consideri il problema di mantenere un campo *height[x]* che, per ogni nodo x appartenente all'albero, contenga l'altezza di x.

(a) Si proponga lo pseudocodice commentato e dettagliato di opportune versioni delle procedure B-TREE-INSERT e B-TREE-DELETE atte a garantire il mantenimento del campo *height*;

(b) si discuta la correttezza e si valutino le complessità delle procedure proposte.

18.7 § Esame di ASD del 10-2-99

Esercizio 1.
Sia dato un vettore A di interi, diversi da 0, di dimensione n.

(a) Si scriva lo pseudo-codice (commentandolo dettagliatamente) di un algoritmo efficiente ed *in–place* che modifichi il vettore A in modo tale che al termine il (sotto)insieme dei numeri positivi sia ordinato in modo crescente e quello dei numeri negativi in modo decrescente.

(b) Si provi la correttezza e si determini la complessità dell'algoritmo proposto.

(c) Si riconsiderino i quesiti (a) e (b) nell'ipotesi che venga richiesto anche che al termine dell'algoritmo nel vettore A ogni elemento positivo sia seguito da un elemento negativo, fino all'esaurimento dei numeri positivi o dei numeri negativi, e quindi compaiano tutti i numeri, positivi o negativi, rimanenti.

Esercizio 2.
Sia $G = (V, E)$ un grafo pesato non orientato in cui non ci sono due archi di peso uguale e sia T un *minimum spanning tree* di G.

(a) Dato un arco $e = (i, j) \in E$ si dimostri che e appartiene a T se e solo se ogni cammino di lunghezza maggiore o uguale a 2 da i a j contiene un arco di peso maggiore del peso di e. Inoltre si provi o si refuti l'unicità di T.

(b) Si descriva un algoritmo che dati G e T nelle ipotesi di cui sopra e dato un arco pesato $\bar{e} \notin E$ di peso diverso dal peso di ogni arco in E, ritorni un minimum spanning tree T' per $G' = (V, E \cup \{\bar{e}\})$?

(c) Si provi la correttezza e si determini la complessità dell'algoritmo di cui al punto precedente.

18.8 § Esame di ASD del 26-1-99

Esercizio 1.
Si considerino n numeri interi $a_1, ..., a_n$ la cui somma sia n.

1. Si disegni un algoritmo efficiente per ordinare $a_1, ..., a_n$.
2. Si provi la correttezza e si determini la complessità dell'algoritmo proposto.

Esercizio 2.
Sia H una heap contenente gli elementi $\{a_1, ..., a_n\}$ e dato A sottoinsieme di $\{a_1, ..., a_n\}$, sia $H(A)$ la più piccola sotto-heap di H contenente gli elmenti di A.

1. Si proponga un algoritmo efficiente che, dati $A_1, ..., A_k$ sottoinsiemi di $\{a_1, ..., a_n\}$, li ordini in modo tale che $A_i < A_j \Rightarrow H(A_i) \subseteq H(A_j)$
2. Si provi la correttezza e si determini la complessità dell'algoritmo proposto.

Capitolo 19

ANNO 1997-1998

19.1 § Compitino di ASD del 10-6-98

Esercizio 1.
Si consideri un grafo orientato e pesato $G = (V, E)$ e si supponga che G contenga una *radice* (cioè un nodo dal quale ogni altro nodo in V risulti raggiungibile).

(a) Si proponga un algoritmo efficiente per determinare se G contiene un ciclo negativo;

(b) si modifichi l'algoritmo proposto al passo precedente in modo da produrre in output i nodi di un ciclo negativo (quando ne esista uno);

(c) si provi la correttezza e si valuti la complessità degli algoritmi proposti.

19.2 § Compitino di ASD del 27-2-98

Esercizio 1.
Si considerino un insieme S di numeri naturali e le seguenti operazioni:

1. *extract_min*(S) che estrae (cancellandolo) l'elemento minimo da S;
2. *insert*(x, S) che inserisce l'elemento x in S;
3. *print_log_sequence*(S) che stampa una sequenza ordinata di $\log(|S|)$ elementi di S tale che il primo elemento della sequenza sia il minimo in S e l'ultimo sia il massimo in S.

(a) Si proponga una struttura dati atta a memorizzare S ed a supportare efficientemente le operazioni sopra elencate.

(b) Si provi la correttezza e si determini la complessità degli algoritmi proposti per supportare le operazioni considerate.

(c) Si discuta il problema di supportare anche l'operazione di estrazione del massimo.

19.3 § Esame di ASD del 12-02-98

Esercizio 1.
Sia A un vettore contenente n interi $a_1, \ldots, a_n$ a due a due distinti. Definiamo *spiazzamento* dell'elemento a_i l'intero k tale che $i + k$ sia la posizione di a_i nella versione ordinata di A.

(a) Si fornisca lo pseudo-codice di commentato dettagliatamente di un algoritmo che, preso in input il vettore A, produca come output il vettore $B = [b_1, \ldots, b_n]$ degli spiazzamenti di $a_1, \ldots, a_n$.

(b) Si provi la correttezza e si determini la complessità dell'algoritmo proposto.

Esercizio 2.
Dato un grafo non orientato $G = (V, E)$, diciamo che G è *bipartito* se esistono V_1 e V_2 tali che

- $V_1 \cap V_2 = \emptyset$;
- $V_1 \cup V_2 = V$;
- per ogni $e = (u, v) \in E$, $u \in V_1 \Leftrightarrow v \notin V_2$.

(a) Si fornisca lo pseudo-codice di commentato dettagliatamente di un algoritmo che determini se un grafo G è bipartito.

(b) Si provi la correttezza e si determini la complessità dell'algoritmo proposto.

19.4 § Esame di ASD del 28-9-98

Esercizio 1.
Sia dato un generatore di sequenze (localmente) ordinate di numeri naturali, di valore compreso tra 0 e c (con c costante prefissata), il cui comportamento può essere descritto come una successione infinita di passi che si articolano nelle seguenti due fasi:

(fase 1) generazione di sequenze ordinate di lunghezza esponenzialmente crescente 2^i, con $i = 0, 1, \ldots, k$;

(fase 2) generazione di sequenze ordinate di lunghezza esponenzialmente decrescente 2^i, con $i = k, k-1, \ldots, 0$.

Si chiede di:

(1) proporre lo pseudo-codice, adeguatamente commentato, di un algoritmo che riceva in input le $2 \cdot (k+1)$ sequenze ordinate generate in un singolo passo e restituisca in output un'unica sequenza ordinata;

(2) provare la correttezza e determinare la complessità dell'algoritmo proposto.

Esercizio 2.
Sia $G = (V, E)$ un grafo non-orientato e connesso. Diciamo che un vertice $v \in V$ è un *cut-vertice* se la rimozione da G di v (e di tutti gli archi che lo contengono) sconnette G.

(1) Proporre lo pseudo-codice, adeguatamente commentato, di un algoritmo che ricevuto in input G restituisca in output tutti i cut-vertici di G;

(2) provare la correttezza e determinare la complessità dell'algoritmo proposto.

Suggerimento: usare la visita in profondità e la classificazione degli archi di G che si ottiene eseguendo tale visita.

19.5 § Esame di ASD del 9-9-98

Esercizio 1.
Si considerino le due seguenti varianti dell'algoritmo *heap-sort*. A partire da una *min-heap*,

- dopo aver restituito in output il minimo corrente x, (anziché rimuoverlo sostituendolo con la foglia più a destra dell'ultimo livello) determina l'elemento minimo fra le foglie dell'ultimo livello, scambialo con la foglia più a destra e sostituisci ad x tale elemento;
- scambia il nodo x che si trova in posizione di radice (minimo corrente) con il minimo fra i suoi figli e ripeti l'operazionione finché x non raggiunge la posizione di una foglia; a quel punto restituisci x in output e rimuovilo.

(a) Si fornisca lo pseudo-codice commentato delle due varianti di heap-sort sopra descritte;

(b) si provi la correttezza e si determini la complessità degli algoritmi proposti.

Esercizio 2.
Sia $G = (V, E)$ un grafo diretto, $w : E \to \Re^+$ una funzione peso, $s \in V$ ed $(u, v) \in E$ un arco.

(a) Si proponga un algoritmo per determinare se esiste un cammino minimo da s a v che contiene l'arco (u, v);

(b) si proponga un algoritmo per determinare se esiste un cammino minimo da s a v che non contiene l'arco (u, v);

(c) si provi la correttezza e determini la complessità degli algoritmi proposti.

19.6 § Esame di ASD del 15-7-98

Esercizio 1.
Dato quale input un vettore V contenente $n = 3^k$ numeri interi distinti, si consideri il seguente algoritmo $\mathcal{A}$:

passo 1: se $k = 0$ l'unico elemento di V viene restituito come output, altrimenti il vettore V viene suddiviso in k gruppi contenenti 3 elementi ciascuno;

passo 2: per ognuno dei k gruppi viene determinato l'elemento avente valore mediano e viene costituito il vettore V' contenete i 3^{k-1} elementi determinati;

passo 3: $\mathcal{A}$ richiama se stesso con V' come input.

(a) Si determini la complessità di $\mathcal{A}$ in funzione di n;

(b) si descriva una implementazione *in-place* di $\mathcal{A}$;

(c) si provi o si refuti il seguente enunciato: l'elemento prodotto in output da $\mathcal{A}$ cui è stato dato in input V è l'elemento mediano di V.

Esercizio 2.
Dato $\Sigma = \{0, 1\}$, siano $a = a_0, a_1, \ldots, a_p$ e $b = b_0, b_1, \ldots, b_q$, due stringhe su Σ. Diciamo che la stringa a è lessicograficamente minore della stringa b se:

1. esiste un intero j, con $0 \leq j \leq min(p, q)$, tale che $a_i = b_i$, per $i = 0, 1, \ldots, j-1$, e $a_j < b_j$, oppure
2. $p < q$ e $a_i = b_i$ per $i = 0, 1, \ldots, p$.

La struttura dati *radix tree* è un albero binario ordinato i cui archi sinistri sono etichettati con 0, i cui archi destri sono etichettati con 1 e i cui nodi possono essere bianchi o neri. Inizialmente tutti i nodi sono bianchi; una stringa s viene rappresentata in un radix tree colorando di nero il nodo x tale che la concatenazione delle etichette nel cammino dalla radice ad x produca s.
Sia S un insieme di stringhe distinte su Σ, la somma delle cui lunghezze sia pari a n.

(a) Proporre un algoritmo efficiente che ordini lessicograficamente S utilizzando la struttura dati radix tree ;

(b) provare la correttezza e determinare la complessità dell'algoritmo proposto.

19.7 § Esame di ASD del 30-6-98

Esercizio 1.
Dato un albero k-ario τ (generalizzazione della nozione di albero binario in cui ogni nodo interno ha k, anzichè 2, figli) perfettaente bilanciato e un nodo interno $x \in \tau$, siano $\downarrow (x, 0), ..., \downarrow (x, k-1)$ rispettivamente il primo, il secondo, ... il k-esimo figlio di x.
Sull'insieme dei nodi di τ sia definita una relazione di ordinamento totale $<$ nel seguente modo: $<$ è la chiusura transitiva della relazione $\prec$, tale che:

i per ogni nodo interno x, $\downarrow(x,0) \prec x$ e $x \downarrow (x,j)$, con $j = 1, ..., k-1$;

ii per ogni nodo interno x, $\downarrow(x,j) \prec\downarrow (x, j+1)$, per $j = 1, ..., k-2$;

iii dati $x, x' \in \tau$, se $x \prec x'$ e x' non appartiene al sottoalbero radicato in x, allora $\downarrow(x, k-1) \prec x'$;

iv dati $x, x' \in \tau$, se $x \prec x'$ e x' non appartiene al sottoalbero radicato in x', allora $x \prec\downarrow (x', 0)$.

Si dimostri la verità o la falsità delle seguenti proposizioni (in caso di falsità si produca un controesempio):

1a dati $x, x' \in \tau$, se profondità(x) = profondità(x') e $x < x'$, allora per ogni nodo y appartenente al sottoalbero radicato in x, e per ogni nodo y' appartenente al sottoalbero radicato in x', $y < y'$;

1b per ogni nodo interno x è vero che, per ogni nodo appartenente al sottoalbero radicato in $\downarrow(x,0)$, $y < x$ e che, per ogni nodo y' appartenente al sottoalbero radicato in $\downarrow(x,1)$, $x < y'$;

1c (generalizzazione di 1a) dati $x, x' \in \tau$, se $x < x'$ allora per ogni nodo y appartenente al sottoalbero radicato in x e per ogni nodo y' apartenente al sottoalbero radicato in x', $y < y'$.

Esercizio 2.
Dato un grafo $G = (V, E)$ diretto aciclico:

2a si proponga un algoritmo efficiente che associ ad ogni nodo v la lunghezza del cammino più lungo da v ad una foglia (cioè un nodo privo di archi uscenti);

2b si provi la correttezza e si valuti la complessità dell'algoritmo proposto.

19.8 § Esame di ASD del 28-1-98

Esercizio 1.
Si assuma che la coppia di interi $[a_i, b_i]$, con $a_i \leq b_i$, rappresenti l'insieme degli interi compresi tra a_i e b_i (estremi inclusi). Si scriva un algoritmo efficiente che, ricevuto in ingresso un insieme di n coppie $[a_i, b_i]$, con $1 \leq i \leq n$, tali che $\forall i, j$ $(1 \leq i, j \leq n)$ $b_i - a_i = b_j - a_j$, restituisca un insieme di m coppie $[c_i, d_i]$, con $1 \leq i \leq m$, comprendenti tutti e soli gli interi che appartengono ad uno ed uno solo degli insiemi $[a_i, b_i]$.

(a) Si fornisca lo pseudo-codice dell'algoritmo, commentandolo dettagliatamente.

(b) Si provi la correttezza e si determini la complessità dell'algoritmo proposto.

(c) Si discutano gli effetti della rimozione della condizione $\forall i, j$ $(1 \leq i, j \leq n)$ $b_i - a_i = b_j - a_j$ sulle soluzioni proposte per i quesiti (a) e (b).

Esercizio 2.
Dato un grafo non orientato (non pesato) $G = (V, E)$ ed un insieme di nodi $F \subseteq V$:

(a) si proponga lo pseudocodice di un algoritmo efficiente che determini, se esiste, un albero di supporto per G in cui ogni nodo di F risulti essere una foglia;

(b) si provi la correttezza e si valuti la complessità dell'algoritmo proposto.

Capitolo 20

ANNO 1996-1997

20.1 § Esame di ASD del 1-10-97

Esercizio 1.
Si scriva un algoritmo efficiente che, ricevuto in ingresso un insieme di n intervalli $[a_i, b_i]$, con a_i, b_i numeri interi e $1 \leq i \leq n$, stabilisca se la loro unione è un intervallo (nel qual caso, restituisca tale intervallo) o meno.

(a) Si fornisca lo pseudo-codice commentato dell'algoritmo.

(b) Si provi la correttezza e si determini la complessità dell'algoritmo proposto.

Esercizio 2.
Si dia una risposta motivata ad ognuna delle seguenti domande:

1. Sia T un *minimum spanning tree* di un grafo indiretto G con pesi positivi. Se aggiungiamo a G esclusivamente un vertice v ed archi (di peso positivo) uscenti da v, il peso di un *minimum spanning tree* per il grafo ottenuto è sicuramente superiore al peso di T?
2. Tutti gli *spanning trees* (sia minimi che non) di un grafo G hanno lo stesso numero di archi?
3. Se aggiungiamo archi ad un grafo diretto aciclico, il grafo ottenuto è ancora un grafo diretto aciclico?
4. Se cancelliamo archi da un grafo diretto aciclico, il grafo ottenuto è ancora un grafo diretto aciclico?
5. Il grafo indiretto ottenuto eliminando la direzione degli archi da un grafo diretto aciclico, è aciclico?
6. Supponiamo che durante la visita in profondità (DFS) di un grafo diretto G vengano assegnati ad ogni vertice v due numeri: $pre(v)$ e $post(v)$. Tali valori corrispondano, rispettivamente, alle posizioni di v nelle liste in pre-ordine ed in post-ordine dei nodi nel DFS-albero/i ottenuto dalla visita. Sono vere le seguenti affermazioni?
   - se $pre(v) < pre(w)$ e $post(v) > post(w)$ allora c'è un cammino da v a w in G;
   - se $pre(v) < pre(w)$ e $post(v) < post(w)$ allora non c'è un cammino da v a w G;

20.2 § Esame di ASD del 18-9-97

Esercizio 1.
Si consideri un cammino dalla radice ad una delle foglie in un albero binario di ricerca. Siano A l'insieme delle chiavi associate ai nodi alla sinistra del cammino, B l'insieme delle chiavi associate ai nodi appartenenti al cammino, C l'insieme delle chiavi associate ai nodi alla destra del cammino. Sia inoltre x la chiave associata alla radice.
Si dimostri la verità o falsità delle seguenti proposizioni (in caso di falsità, si produca un controesempio):

1-a $\forall y, z((y \in B \wedge z \in C) \rightarrow y \leq z)$;

1-b $\forall y(y \in A \rightarrow y \leq x) \vee \forall y(y \in C \rightarrow y \geq x)$;

1-c $\forall y \in A \forall z \in C(y \leq z)$.

Esercizio 2.
Sia $G = (V, E)$ un grafo diretto e aciclico, e siano s e t due vertici distinti in V. Si consideri il problema di determinare un insieme *massimale* di cammini disgiunti (privi di nodi in comune) da s a t.

2-a Si producano G, s, e t tali che esiste un insieme massimale ma non di cardinalità massima, di cammini disgiunti da s a t in G.

2-b Si proponga lo pseudo codice commentato di un algoritmo che risolva il problema proposto.

2-c Si valuti la complessità dell'algoritmo di cui al punto precedente.

20.3 § Esame di ASD del 30-7-97

Esercizio 1.
Si scriva un algoritmo efficiente che, ricevuto in ingresso un multi-insieme (insieme con ripetizioni) $E \subseteq N \times N$ (con N insieme dei numeri naturali) di n coppie che coinvolgono $m \leq n$ elementi distinti di N tale che $\forall (x, y) \in E \; \exists z \in N \; ((y, z) \in E)$, restituisca l'insieme $E' = \{(f(x), f(y)) : (x, y) \in E\}$, dove f è una funzione che mappa gli m elementi distinti di N che compaiono nelle coppie di E nei primi m numeri naturali $1, \ldots, m$ in modo che $\forall (x, y), (z, w) \in E \; (x < z \rightarrow f(x) < f(z))$.

1-a Si fornisca lo pseudo-codice commentato dell'algoritmo.

1-b Si provi la correttezza e si determini la complessità dell'algoritmo proposto.

1-c Si riconsiderino i quesiti (a) e (b) nell'ipotesi aggiuntiva che valga la condizione $\exists k \in N \; \forall (x, y) \in E \; (x \leq k)$.

1-d E' possibile fornire un limite inferiore (lower bound) per la complessità dell'algoritmo di cui al punto (a)?

Esercizio 2.
Sia $G = (V, E)$ un grafo diretto e aciclico.

2-a Si proponga un algoritmo che determinini un ordinamento $<$ dei nodi di V tale che per $u, v \in V$, se esiste un cammino da u a v in G allora $u < v$.

2-b Si dimostri la correttezza e si valuti la complessitá dell'algoritmo proposto.

20.4 § Esame di ASD del 23-6-97

Esercizio 1.
Si descriva un algoritmo efficiente che, ricevuto in input un vettore A di n interi positivi, restituisca in output l'insieme degli elementi che occupano una delle posizioni comprese tra la k-esima posizione e la $(k + m)$-esima posizione (estremi inclusi) nella permutazione che ordina A, con $1 \leq k \leq k + m \leq n$.

1-a Si fornisca lo pseudo-codice commentato dell'algoritmo.

1-b Si provi la correttezza e si determini la complessità dell'algoritmo proposto.

Esercizio 2.
Sia $G = (V, E)$ un grafo indiretto e siano s, t ed u tre vertici distinti in V.

2-a Si proponga un algoritmo che determini se ogni cammino da s a t passa per u.

2-b Si proponga un algoritmo che determini se esiste un cammino da s a t passante per u.

2-c Si provi la correttezza e si valuti la complessità degli algoritmi proposti.

20.5 § Esame di ASD del 3-6-97

Esercizio 1.
Sia A un array di interi positivi o negativi, di dimensione n e tale che

$$A[1] < A[2] < \ldots < A[n].$$

1-a Scrivere lo pseudocodice commentato di un algoritmo che, preso in input A, produca come output un indice i tale che $A[i] = i$, se tale i esiste, o NIL altrimenti.

1-b Dimostrare la correttezza e determinare la complessità dell'algoritmo proposto.

Esercizio 2.
Si consideri la tecnica di implementazione di una min-heap mediante un array discussa a lezione. Si consideri l'operazione, che chiameremo $insert\_cut(x)$, che consiste nell'inserire l'elemento x nella heap e, contemporaneamente, cancellare dalla heap tutti gli elementi maggiori di x.

2-a Si dimostri che l'array (rappresentante la heap) ottenuto a partire dall'array vuoto dopo m operazioni di tipo $insert\_cut(x)$ è ordinato.

2-b Si proponga una implementazione atta a supportare esclusivamente operazioni di tipo $insert\_cut(x)$.

2-c Si discuta una modifica della tecnica usata per l'implementazione che permetta di trattare anche operazioni di tipo $delete\_min$, in modo che n operazioni $delete\_min$ ed m operazioni di tipo $insert\_cut(x)$ abbiano costo globale $\theta(m+n)$.

20.6 § Esame di ASD del 27-1-97

Esercizio 1.
Sia A un vettore contenente $2n$ numeri interi, di cui n positivi ed n negativi.

1-a Si scriva lo pseudo-codice commentato di un algoritmo efficiente che dati x, y con $x < y$ determina se esiste una coppia (i, j) tale che $A[i] < 0, A[j] > 0$ e $x < A[i] + A[j] < y$.

1-b Si dimostri la correttezza e si valuti la complessità dell'algoritmo proposto.

Esercizio 2.
Sia H una min-heap di altezza h contenente $2^{h+1} - 1$ elementi. Sia $H(p)$ l'insieme dei nodi di H aventi profondità p:

$$H(p) \quad = \quad \{a \in H \mid \ \text{depth}(a) = p\}.$$

Si assuma che H sia rappresentata mediante un vettore e che, per ogni p, il sotto-vettore contenente la lista dei nodi in $H(p)$ sia ordinata.

2-a Si dimostri che il vettore contenente H non è necessariamente ordinato.

2-b Si proponga lo pseudo-codice commentato di un algoritmo efficiente che produca la lista ordinata dei nodi in H.
[**Suggerimento**: Si consideri un algoritmo ricorsivo.]

2-c Si dimostri la correttezza e si valuti la complessità dell'algoritmo proposto.

Capitolo 21

ANNO 1995-1996

21.1 § Esame di ASD del 14-10-96

Esercizio 1.
Sia A un vettore di $n = 2^k$ interi. Una coppia (i, j) si dice *un'inversione di A* se $i < j$ e $A[i] > A[j]$.

1-a Si scriva lo pseudocodice commentato di un algoritmo efficiente che determina il numero di inversioni del vettore A.

1-b Si valuti la complessità e dimostrare la correttezza dell'algoritmo proposto.

Esercizio 2.
Sia $G = \langle V, E \rangle$ un grafo indiretto, connesso e aciclico con n nodi.

2-a Si dimostri che esiste sempre un cammino che percorre tutti gli archi del grafo G in entrambe le direzioni.

2-b Si proponga lo pseudo-codice commentato di un algoritmo che produca un cammino con le caratteristiche di cui al punto precedente.

2-b Si valuti la complessità dell'algoritmo proposto.

21.2 § Esame di ASD del 30-9-96

Esercizio 1.
Sia A un vettore di $n = 2^k$ interi. Una coppia (i, j) si dice *un'inversione di A* se $i < j$ e $A[i] > A[j]$.

1-a Si scriva lo pseudocodice commentato di un algoritmo efficiente che determina il numero di inversioni del vettore A.

1-b Si valuti la complessità e dimostrare la correttezza dell'algoritmo proposto.

Esercizio 2.
Sia T_{rb} un red-black tree di altezza h.

2-a Si disegni e si valuti la complessità di un algoritmo che trasformi T_{rb} in un B-albero T_B di grado 2 e di altezza $\Theta(h)$.

2-b Si dimostri che T_B è un B-albero.

21.3 § Esame di ASD del 2-9-96

Esercizio 1.
Sia k una costante intera maggiore di 1 e sia A un vettore di n interi tali che $0 \leq A[i] < kn$.

1-a Scrivere lo pseudo-codice commentato di un algoritmo che ordina il vettore A in tempo lineare e che usa non più di $3n + 3$ locazioni ausiliarie di memoria.

1-b Dimostrare la correttezza dell'algoritmo proposto.

Esercizio 2.
Si dica albero binario un albero in cui ogni nodo ha $0, 1$ o 2 figli, si dica inoltre albero binario *perfetto* un albero in cui ogni nodo ha 0 o 2 figli.

2-a Si dimostri che il numero di foglie f di un albero binario perfetto T è uguale ad $i + 1$, con i numero di nodi interni in T.

2-b Si proponga lo pseudo-codice commentato di un algoritmo che, dato un albero binario di ricerca, lo trasformi in un albero binario perfetto di ricerca, se questo è possibile, oppure ritorni NIL.

2-c Si provi la correttezza e si valuti la complessità dell'algoritmo proposto al punto precedente.

21.4 § Esame di ASD del 22-7-96

Esercizio 1.
Sia A un vettore di n interi e sia m il numero di posizioni i tali che $A[i] > A[i + 1]$.

1-a Scrivere lo pseudo-codice **commentato**, dimostrare la correttezza e determinare la complessità di un algoritmo efficiente che ordina il vettore A nel caso in cui m è una costante.

1-b Scrivere lo pseudo-codice **commentato**, dimostrare la correttezza e determinare la complessità di un algoritmo efficiente che ordina il vettore A nel caso in cui $m = O(\log n)$.

1-c Gli algoritmi proposti sono stabili? Motivare la risposta e, in caso negativo, indicare come rendere stabili i suddetti algoritmi.

Esercizio 2.
Si supponga di dover eseguire una sequenza di operazioni di tipo MAKE-SET, UNION e FIND. Si assuma che tutte le operazioni di tipo MAKE-SET precedano quelle di tipo UNION e che queste, a loro volta, precedano quelle di tipo FIND.

2-a Si propongano strutture dati ed algoritmi che permettano di implementare efficientemente le operazioni considerate.

2-b Si analizzino le complessità degli algoritmi proposti ed il costo asintotico di una sequenza generica di operazioni che soddisfi l'assunzione relativa all'ordine delle operazioni.

21.5 § Esame di ASD del 24-6-96

Esercizio 1.
Si consideri un albero binario di ricerca T e un nodo x in T con chiave associata $key[x]$.

1-a Scrivere lo pseudo-codice e dimostrare la correttezza di una procedura MODIFY-KEY(T, x, key) che modifica x assegnando a $key[x]$ valore key, e ritorna T se questo è ancora un albero binario di ricerca, altrimenti ritorna NIL. Si valuti la complessità dell'algoritmo proposto.

1-b Si riconsideri il precedente punto nel caso l'albero T sia un red-black tree.

Esercizio 2.

2-a Scrivere un algoritmo di complessità $O(n \cdot \log m)$ per fondere m vettori ordinati in un vettore A, dove n è il numero totale di elementi contenuti negli m vettori.

2-b Si consideri un vettore di $n = 2^{2^k}$ elementi. Utilizzando la soluzione al punto (2-a), scrivere lo pseudo-codice e l'equazione della complessità asintotica della seguente variante di MERGESORT:

- Suddividere il vettore in $\sqrt{n}$ sottovettori;
- ordinare (ricorsivamente) i sottovettori;
- fondere i sottovettori ordinati.

2-c Facoltativo: Valutare la complessità dell' algoritmo precedente. È possibile risolvere il punto (2-a) con complessità $O(n)$?