SD per insiemi disgiunti

Una **struttura di dati per insiemi disgiunti** rappresenta una collezione $\mathcal{U} = \{S_1, S_2, \ldots, S_k\}$ di insiemi dinamici a due a due disgiunti. Cioè $S_i \cap S_j = \emptyset$ per ogni $i \neq j$.

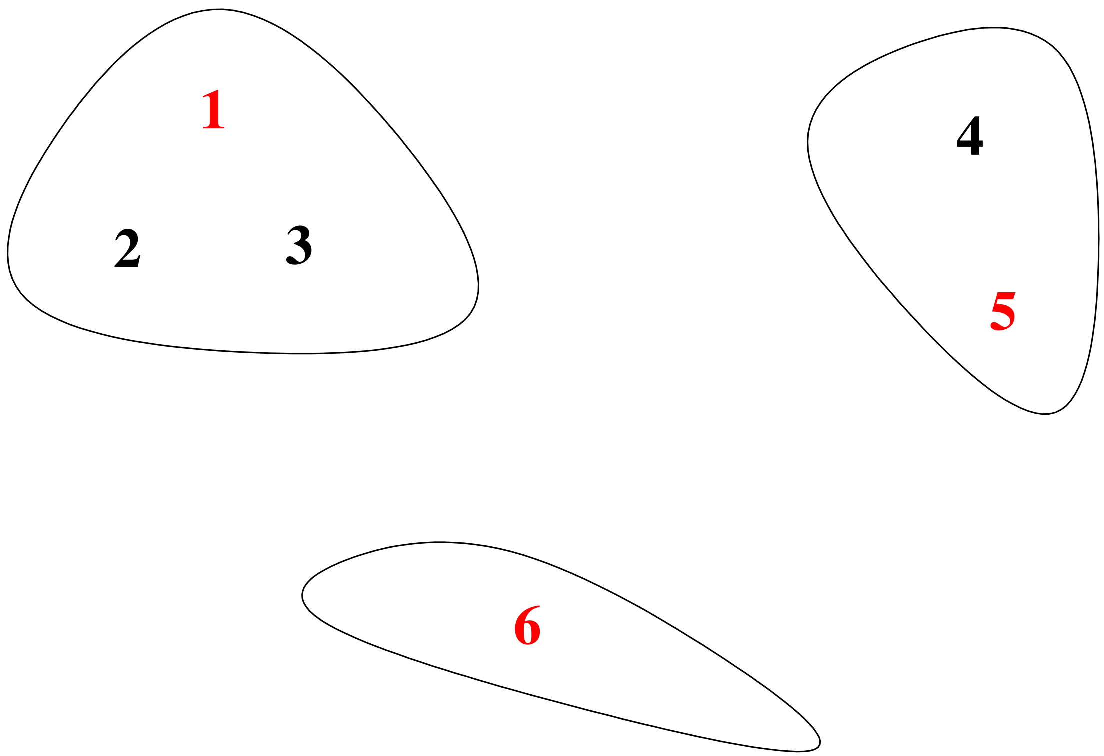
Ogni insieme viene identificato da un **rappresentante**, che è un membro dell'insieme.

SD per insiemi disgiunti

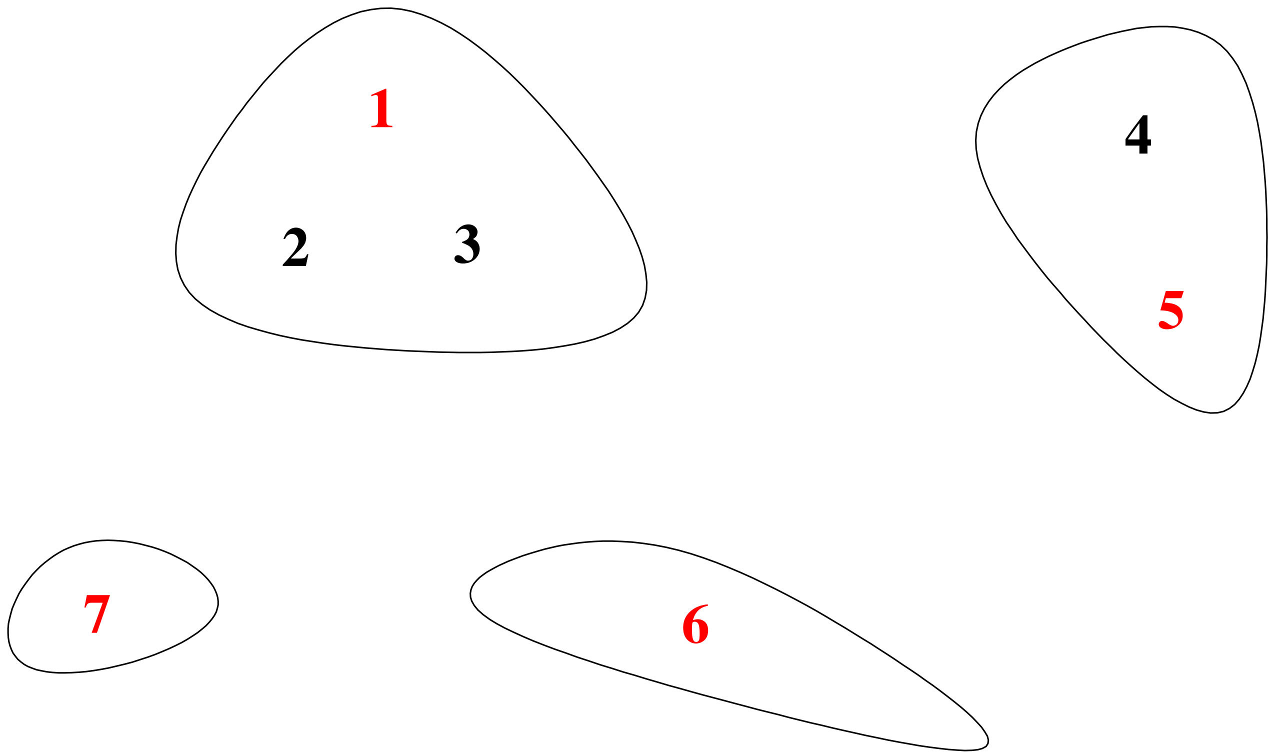
Una struttura di dati per insiemi disgiunti ammette le seguenti operazioni:

- $\mathit{MakeSet}(x)$ crea un nuovo insieme il cui unico membro e rappresentante è x;
- $\mathit{Union}(x, y)$ aggiunge alla collezione $\mathcal{U}$ un nuovo insieme S che è l'unione degli insiemi disgiunti S_x e S_y che contengono rispettivamente x e y. Il rappresentante di S è il rappresentante di S_x oppure quello di S_y. Gli insiemi S_x e S_y vengono rimossi dalla collezione $\mathcal{U}$;
- $\mathit{FindSet}(x)$ ritorna un puntatore al rappresentante dell'unico insieme che contiene x.

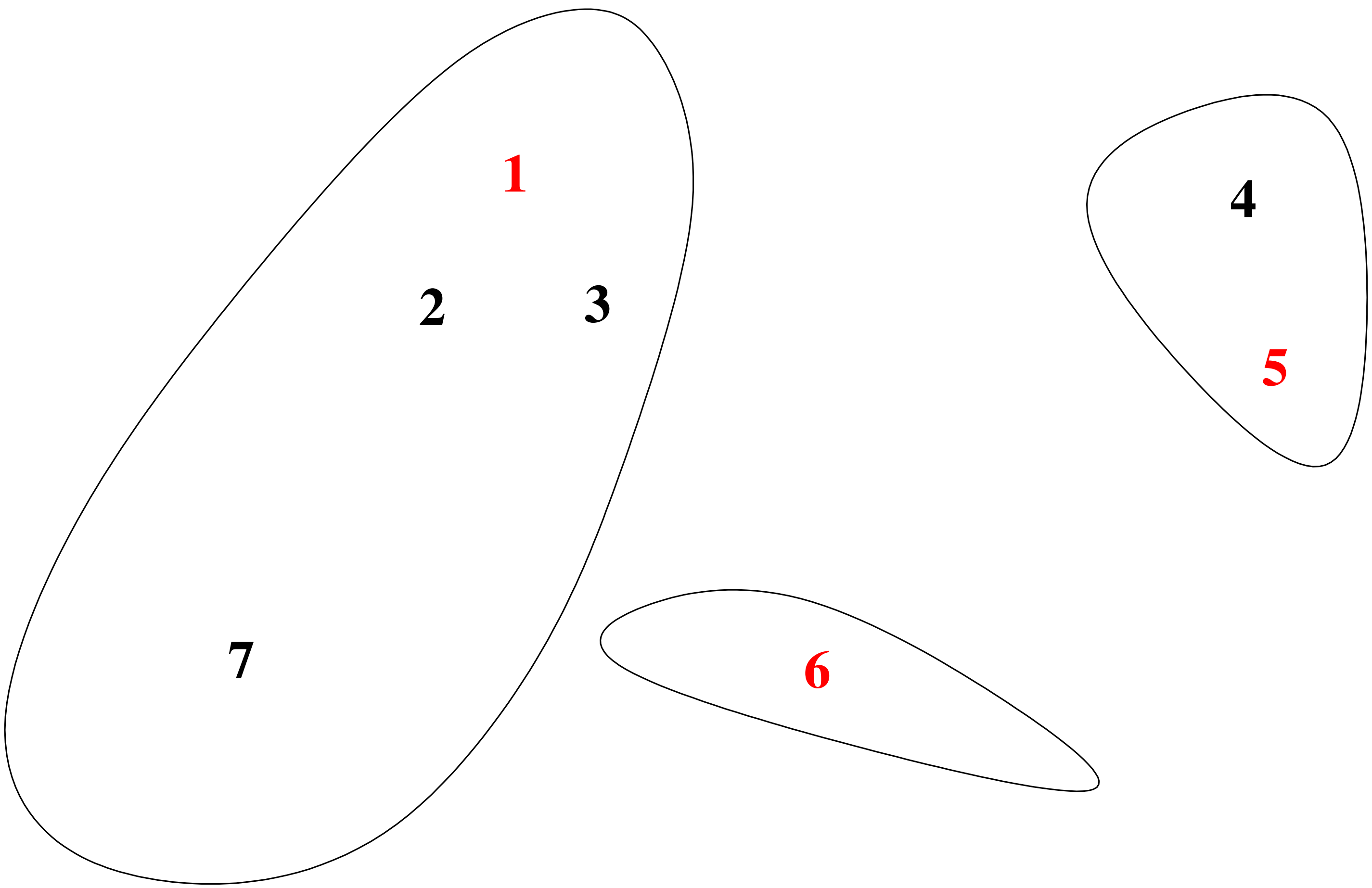
Insiemi disgiunti



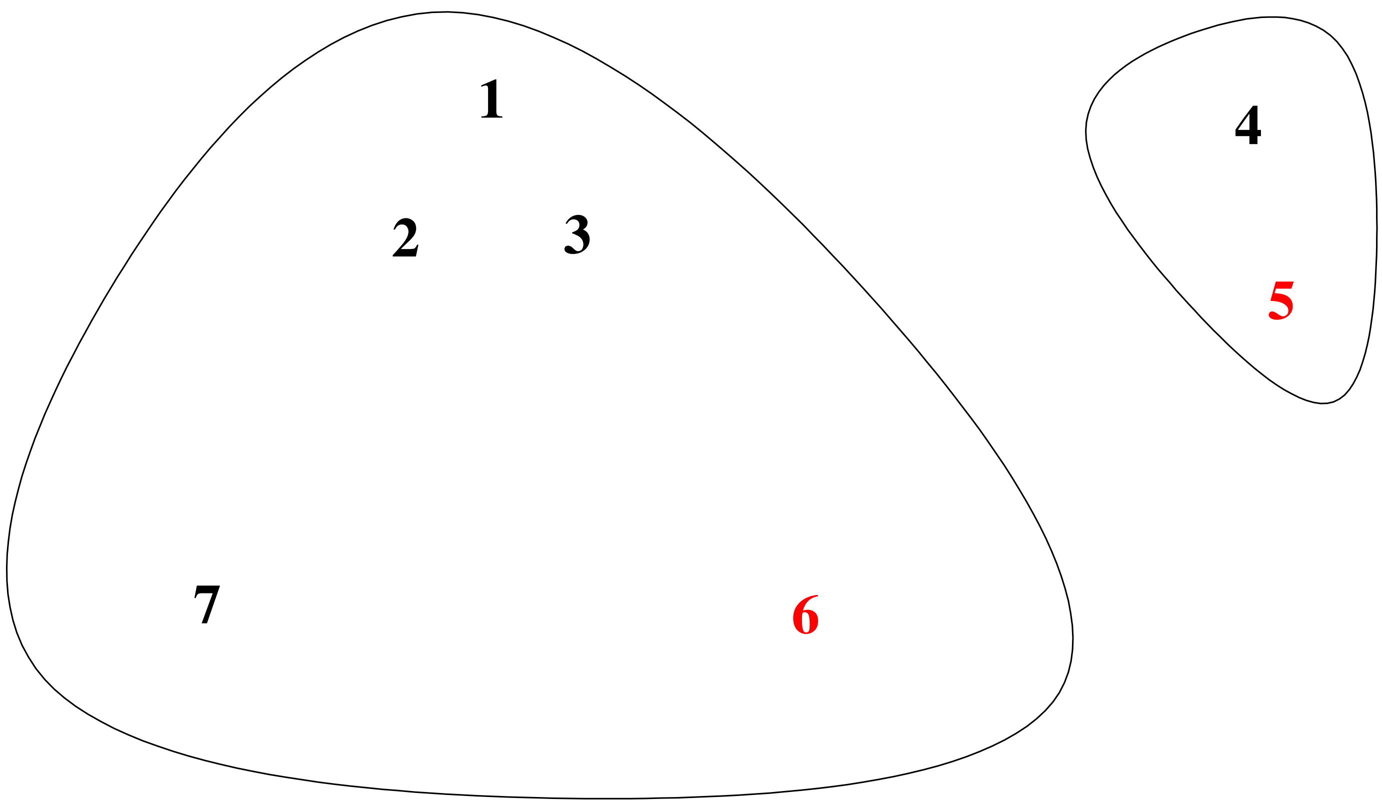
$MakeSet(7)$



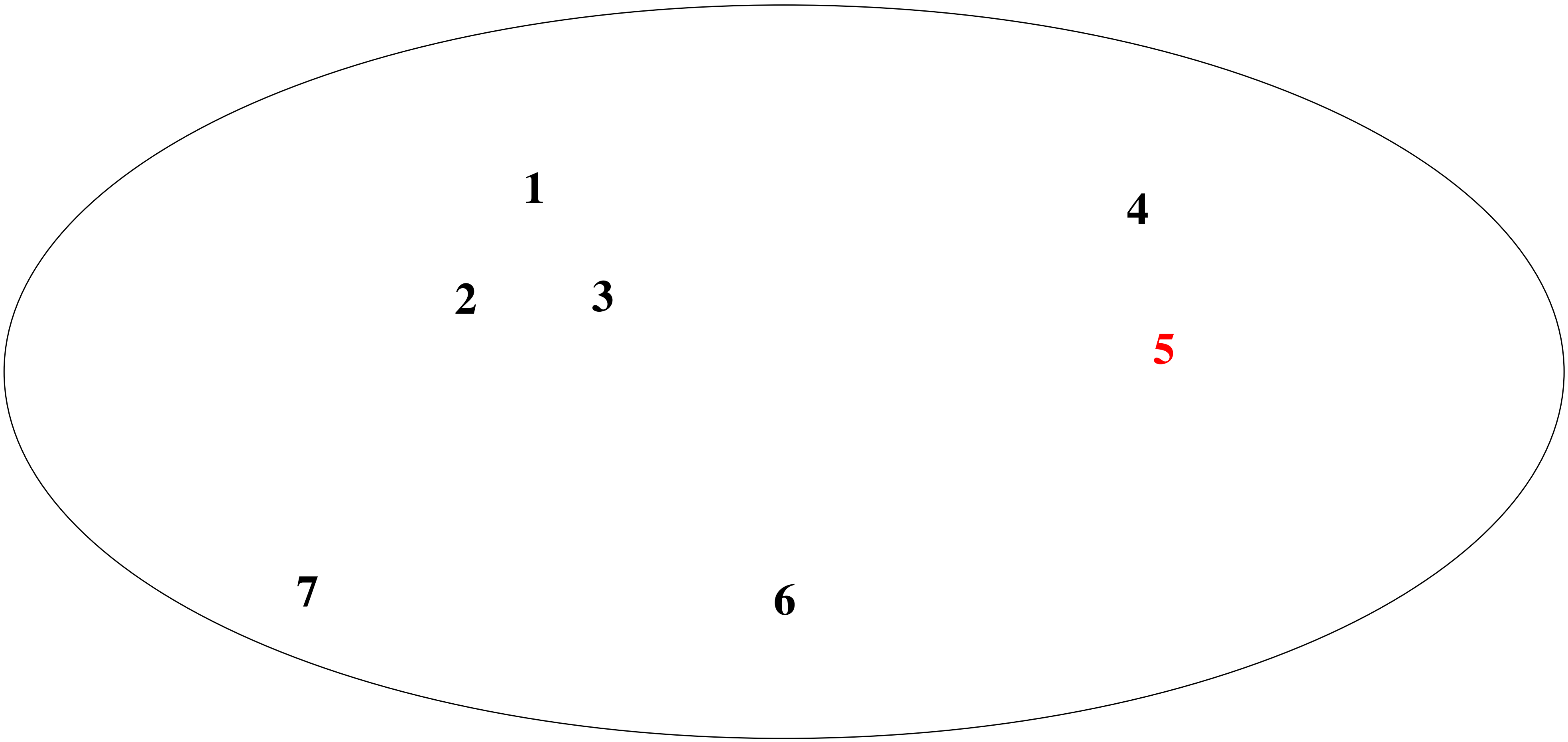
$$\boxed{Union(7, 2)}$$



$\boxed{Union(3,6)}$



$\boxed{Union(1,4)}$



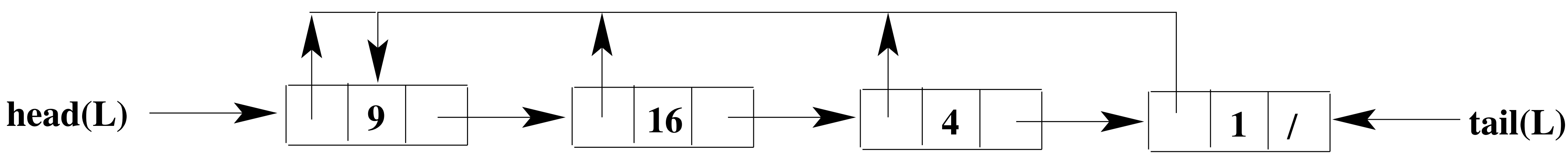
Rappresentazione a lista

Nella **rappresentazione a lista** ogni insieme della collezione viene rappresentato con una **lista**, e ogni elemento di un insieme è un oggetto della lista. Il primo oggetto della lista è il rappresentante dell'insieme.

Ogni oggetto della lista possiede un campo **next** che contiene un puntatore all'oggetto successivo nella lista, e un campo **rep** che contiene un puntatore al rappresentante della lista, cioè alla testa della lista.

La lista possiede gli attributi **head**, che punta alla testa della lista, e **tail** che punta alla coda della lista.

Rappresentazione a lista



Operazioni su insiemi disgiunti

Dato un oggetto x nella collezione, indicheremo con L_x la lista che contiene x.

MakeSet(x)

1: $rep[x] \leftarrow x$
2: $next[x] \leftarrow$ NIL
3: $head[L_x] \leftarrow x$
4: $tail[L_x] \leftarrow x$

FindSet(x)

1: **return** $rep[x]$

Queste procedura hanno complessità costante.

La procedura Union(x,y) appende la lista che contiene y alla fine della lista che contiene x. Essa deve aggiornare il rappresentante di ogni elemento nella lista di y con il rappresentante della lista di x.

Union(x,y)

$z \leftarrow head[L_y]$
while $z \neq$ NIL **do**
    $rep[z] \leftarrow rep[x]$
    $z \leftarrow next[z]$
end while
$next[tail[L_x]] \leftarrow head[L_y]$
$tail[L_x] \leftarrow tail[L_y]$
//Rimuovi L_y dalla collezione

La procedura di unione impiega un **tempo lineare** nella lunghezza della lista di y.

Teorema *Usando una rappresentazione a lista per gli insiemi disgiunti, una sequenza di operazioni delle quali* n *sono* MakeSet, $n-1$ *sono* Union *e* f *sono* FindSet *costa, nel caso pessimo,*

$$\Theta(n^2 + f).$$

Dimostrazione

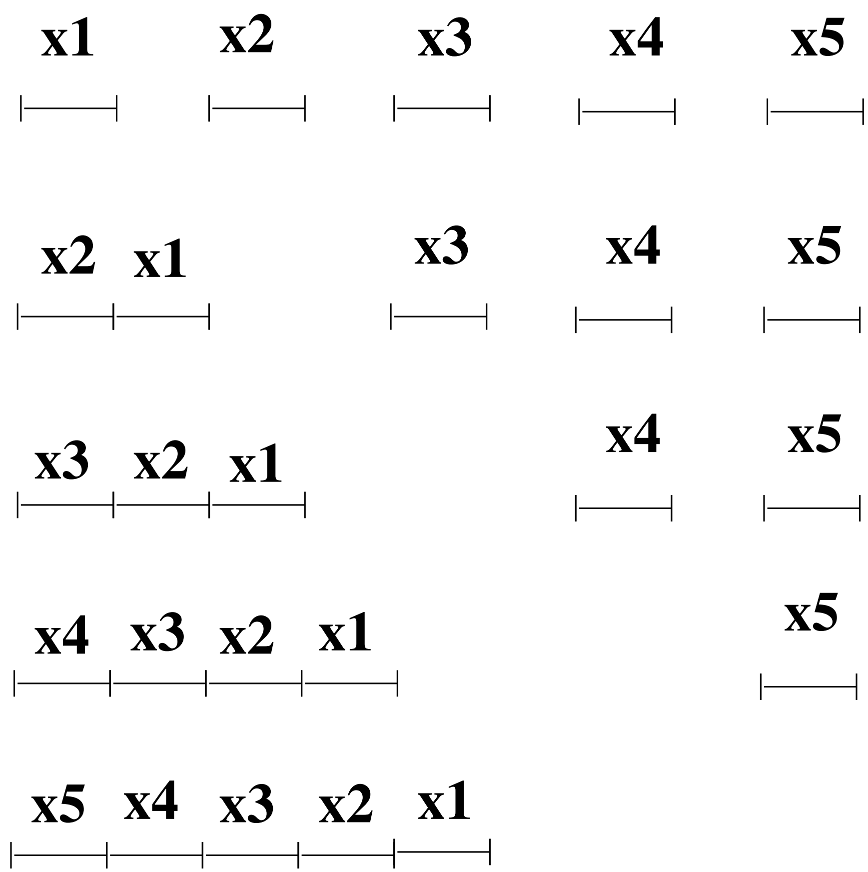
Consideriamo la sequenza:

$$MakeSet(x_1),\ MakeSet(x_2),\ \ldots,\ MakeSet(x_n)$$

seguita dalla sequenza

$$Union(x_2, x_1),\ Union(x_3, x_2),\ \ldots,\ Union(x_n, x_{n-1}),$$

seguita da una sequenza arbitraria di f operazioni *FindSet*.


x1
x2
x3
x4
x5
x2
x1
x3
x4
x5
x3
x2
x1
x4
x5
x4
x3
x2
x1
x5
x5
x4
x3
x2
x1

Le n operazioni *MakeSet* costano $\Theta(n)$ e le f operazioni *FindSet* costano $\Theta(f)$.

L'i-esima operazione $Union(x_{i+1}, x_i)$ appende la lista di x_i alla fine della lista di x_{i+1}. La lista di x_i è lunga i, e dunque la i-esima unione costa $\Theta(i)$.

Dunque le $n-1$ operazioni di unione costano

$$\sum_{i=1}^{n-1} \Theta(i) = \Theta(\sum_{i=1}^{n-1} i) = \Theta((n-1)n/2) = \Theta(n^2).$$

Dunque il costo totale risulta $\Theta(n + n^2 + f) = \Theta(n^2 + f)$.

Si noti che se la sequenza di unioni è invece la seguente:

$$Union(x_1, x_2),\ Union(x_2, x_3),\ \ldots,\ Union(x_{n-1}, x_n),$$

l'i-esima operazione $Union(x_i, x_{i+1})$ costa $\Theta(1)$ e quindi le $n-1$ operazioni di unione costano $\Theta(n-1) = \Theta(n)$.

Euristica di unione pesata

Questo costo elevato dipende dal fatto che, nel caso pessimo, l'operazione di unione appende sempre la lista più lunga alla fine della lista più corta.

L'**euristica di unione pesata** appende la lista più corta alla fine della lista più lunga. Occorre mantenere l'attributo $length[L]$ che contiene la lunghezza della lista L.

WeightedUnion(x,y)

if $length[L_x] > length[L_y]$ **then**
  $Union(x, y)$
else
  $Union(y, x)$
end if

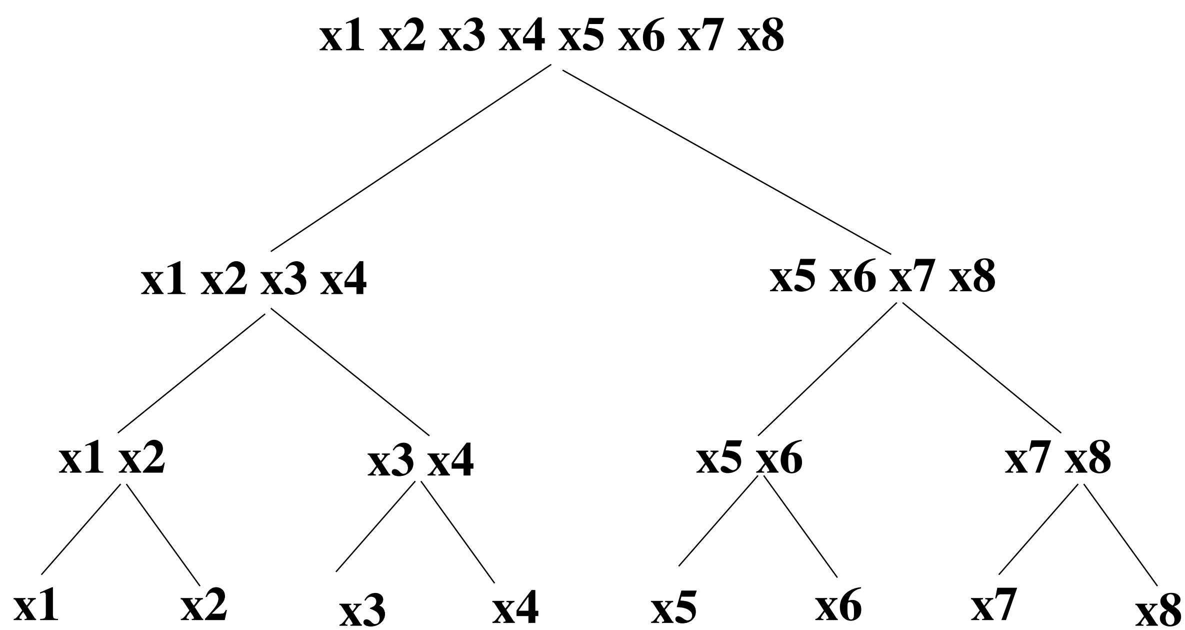
Teorema *Usando una rappresentazione a lista per gli insiemi disgiunti con l'euristica di unione pesata, una sequenza di operazioni delle quali n sono* MakeSet*, $n-1$ sono* Union *e f sono* FindSet *costa, nel caso pessimo,*

$$\Theta(n \log n + f).$$

Dimostrazione

Nel caso pessimo, ogni unione è di tipo $Union(x, y)$, ove la lunghezza della lista che contiene x è uguale alla lunghezza della lista che contiene y. Supponiamo che $n = 2^k$ sia una potenza di 2 (il caso generale segue facilmente da questo caso).

Possiamo costruire un albero binario completo con n foglie e $n-1$ nodi interni. Ogni foglia dell'albero è etichettata con una operazione di *MakeSet* e ogni nodo interno è etichettato con una operazione di *Union*. L'altezza dell'albero è $\log n$.


x1 x2 x3 x4 x5 x6 x7 x8
x1 x2 x3 x4
x5 x6 x7 x8
x1 x2
x3 x4
x5 x6
x7 x8
x1
x2
x3
x4
x5
x6
x7
x8

Per ogni $0 \leq i \leq \log n - 1$, ci sono:

- 2^i operazioni di unione che etichettano nodi a profondità i,
- ogni operazioni di unione a profondità i aggiorna $n/2^{i+1}$ rappresentanti.

Dunque il costo di tutte le unioni a profondità i è

$$2^i \cdot \frac{n}{2^{i+1}} = n/2$$

e dunque il costo di tutte le unioni dell'albero risulta

$$(n/2) \log n = \Theta(n \log n)$$

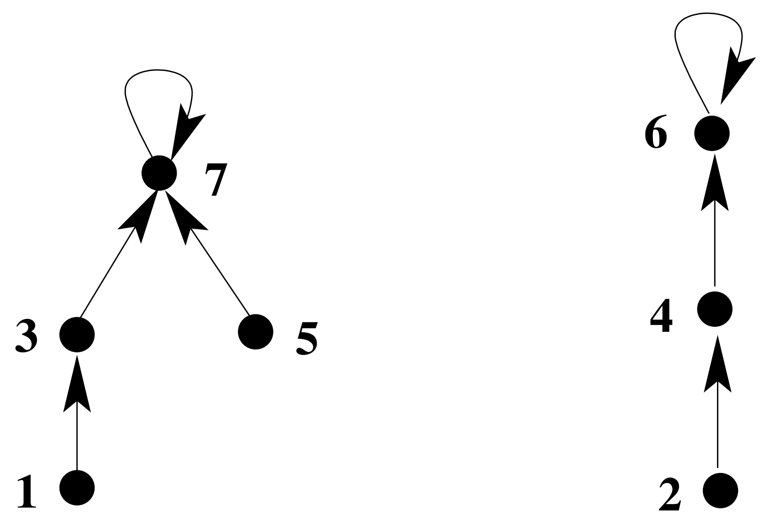
Le n operazioni di *MakeSet* costano $\Theta(n)$ e le restanti f operazioni di *FindSet* costano $\Theta(f)$.

Dunque il costo totale risulta $\Theta(n \log n + n + f) = \Theta(n \log n + f)$.

Rappresentazione ad albero

Rappresentiamo ogni insieme S della collezione come un albero radicato, in cui ogni nodo corrisponde ad un elemento di S e la radice è il rappresentante di S.

Ogni nodo x ha un unico puntatore $p[x]$ al padre di x, se x non è la radice, oppure a x stesso se x è la radice.



MakeSet(x)

$p[x] \leftarrow x$

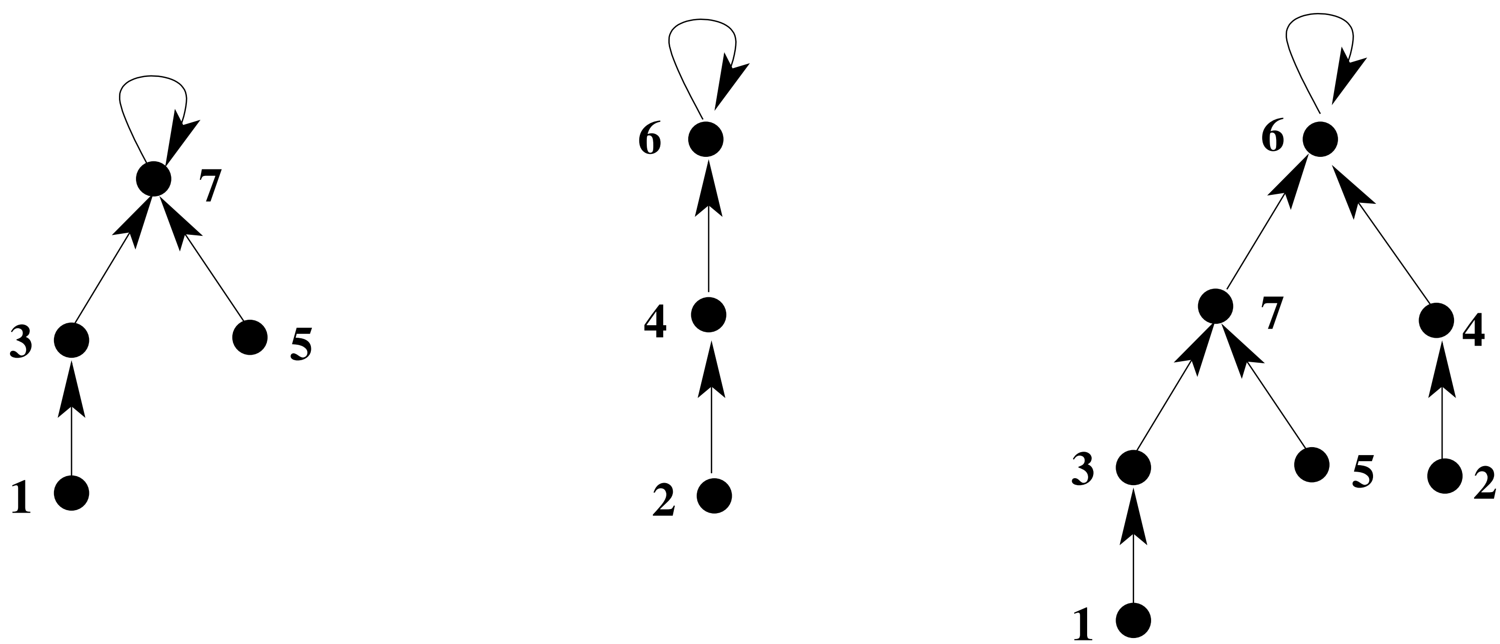
FindSet(x)

if $x = p[x]$ **then**
  return x
else
  return $FindSet(p[x])$
end if

Union(x,y)

$p[FindSet(y)] \leftarrow FindSet(x)$

Union(4,3)



Euristica di unione per rango

L'euristica **unione per rango** implementa la stessa idea della unione pesata: appendo l'albero con meno nodi all'albero con più nodi. In questo modo, **limito l'altezza** dell'albero risultante.

Invece di tenere traccia del numero di nodi di un albero, manteniamo per ogni nodo x il **rango** $rank[x]$ che corrisponde ad un limite superiore rispetto all'**altezza del nodo**.

Modifico le operazioni *MakeSet* e *Union* come segue.

MakeSet(x)

$p[x] \leftarrow x$
$rank[x] \leftarrow 0.$

Union(x,y)

$rep1 \leftarrow FindSet(x)$
$rep2 \leftarrow FindSet(y)$
if $rank[rep1] < rank[rep2]$ **then**
  $p[rep1] \leftarrow rep2$
else
  $p[rep2] \leftarrow rep1$
  if $rank[rep1] = rank[rep2]$ **then**
    $rank[rep1] \leftarrow rank[rep1] + 1$
  end if
end if

Teorema *Usando una rappresentazione ad albero per gli insiemi disgiunti con l'euristica di unione per rango, una sequenza di operazioni delle quali* n *sono* MakeSet, $n-1$ *sono* Union *e* f *sono* FindSet *costa, nel caso pessimo,*

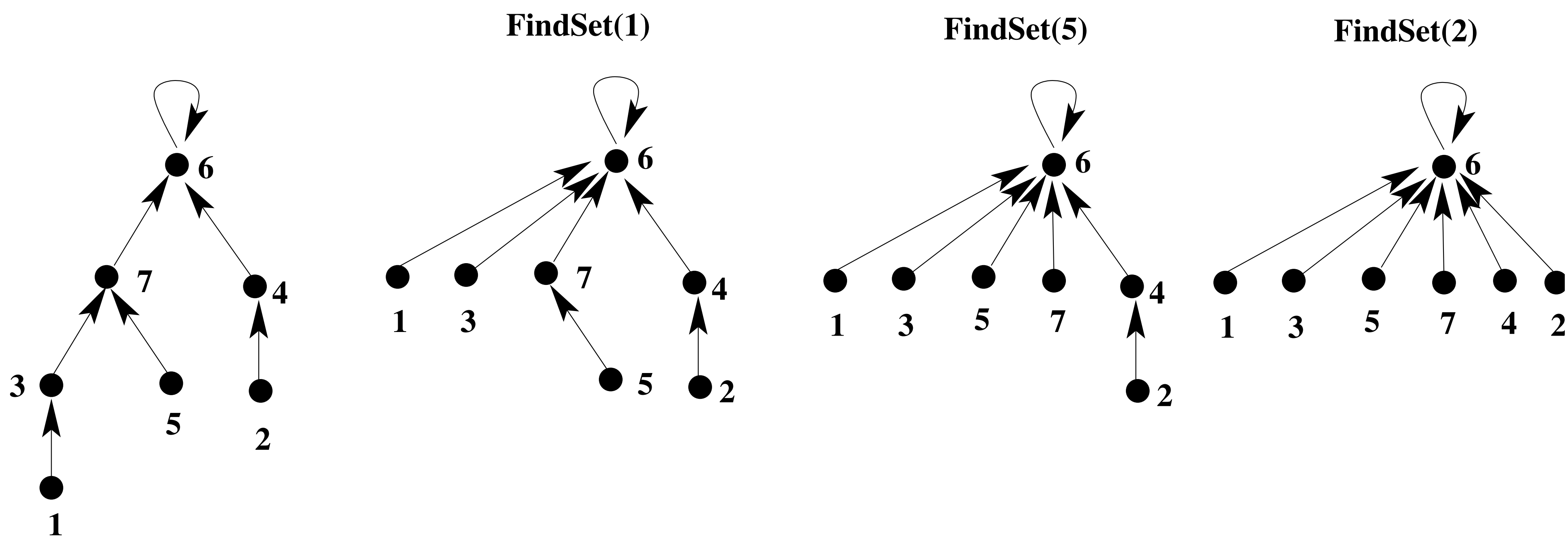
$$\Theta((n+f)\log n).$$

Euristica di compressione dei cammini

L'euristica di **compressione dei cammini** agisce sull'operazione *FindSet*. Quando $FindSet(x)$ viene invocata adottando questa euristica, essa aggancia direttamente alla radice tutti i nodi che incontra sul cammino da x alla radice.

FindSet(x)

if $x = p[x]$ **then**
    return x
else
    $p[x] \leftarrow FindSet(p[x])$
    return $p[x]$
end if


FindSet(1)
FindSet(5)
FindSet(2)
6
7
4
3
5
2
1

Teorema *Usando una rappresentazione ad albero con l'euristica di compressione del cammino, una sequenza di operazioni delle quali* n *sono* MakeSet, $n-1$ *sono* Union *e* f *sono* FindSet *costa, nel caso pessimo,*

$$\Theta(n + f \cdot (1 + \log_{2+f/n} n))$$

Ad esempio, se $f = n$, allora la complessità risulta

$$\Theta(2n + n \log_3 n) = \Theta(n \log n)$$

Funzione di Ackermann

Dati gli interi $k \geq 0$ e $j \geq 1$, sia

$$A_k(j) = \begin{cases} j+1 & se \quad k = 0 \\ A_{k-1}^{(j+1)}(j) & se \quad k \geq 1 \end{cases}$$

La notazione $f^{(n)}(x)$ denota l'iterazione per n volte della funzione f, cioè $f^{(0)}(x) = x$ e $f^{(n)}(x) = f(f^{(n-1)}(x))$, se $n \geq 1$.

La funzione $A_k(j)$ **cresce molto velocemente**. In particolare, $A_4(1) >> 10^{80}$, cioè $A_4(1)$ è molto maggiore del numero stimato di atomi nell'universo osservabile.

Definiamo la **funzione inversa** come

$$\alpha(n) = min\{k : A_k(1) \geq n\}$$

Si dimostra che per ogni costante $k > 0$

$$\lim_{n\to\infty} \frac{\alpha(n)}{k} = \infty$$

e

$$\lim_{n\to\infty} \frac{\alpha(n)}{\log n} = 0$$

La funzione $\alpha(n)$ **cresce più lentamente** di qualsiasi **funzione elementare**.

Teorema *Usando una rappresentazione ad albero per gli insiemi disgiunti con **entrambe** le euristiche di unione per rango e compressione del cammino, una sequenza di operazioni delle quali* n *sono* MakeSet, $n-1$ *sono* Union *e* f *sono* FindSet *costa, nel caso pessimo,*

$$O((n+f)\cdot\alpha(n))$$

Euristiche a confronto

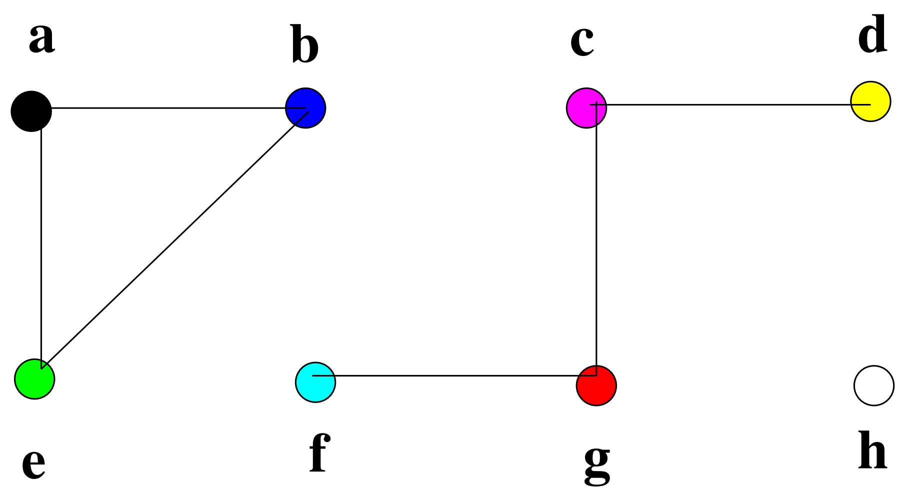
Supponiamo $f = n$.

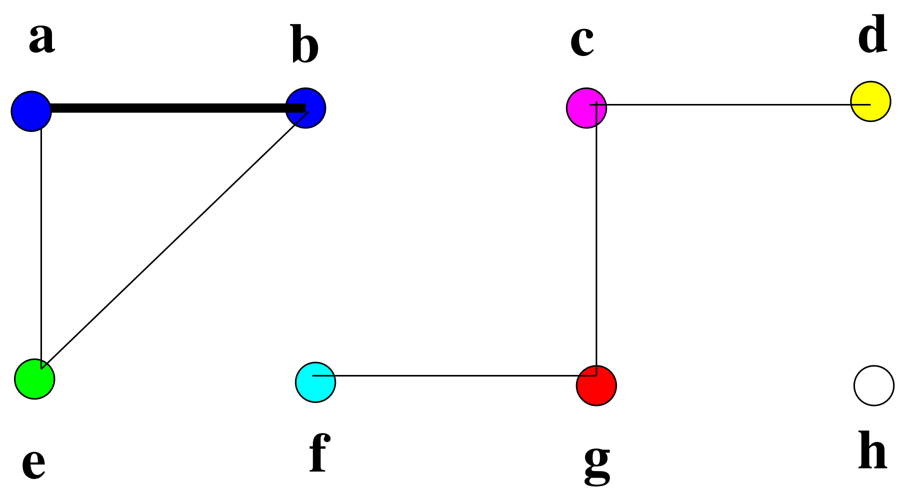
- Rappresentazione a lista
  - Unione semplice: $O(n^2)$
  - Unione pesata: $O(n \log n)$
- Rappresentazione ad albero
  - Unione per rango: $O(n \log n)$
  - Compressione del cammino: $O(n \log n)$
  - Unione per rango + Compressione del cammino: $O(n \cdot \alpha(n))$

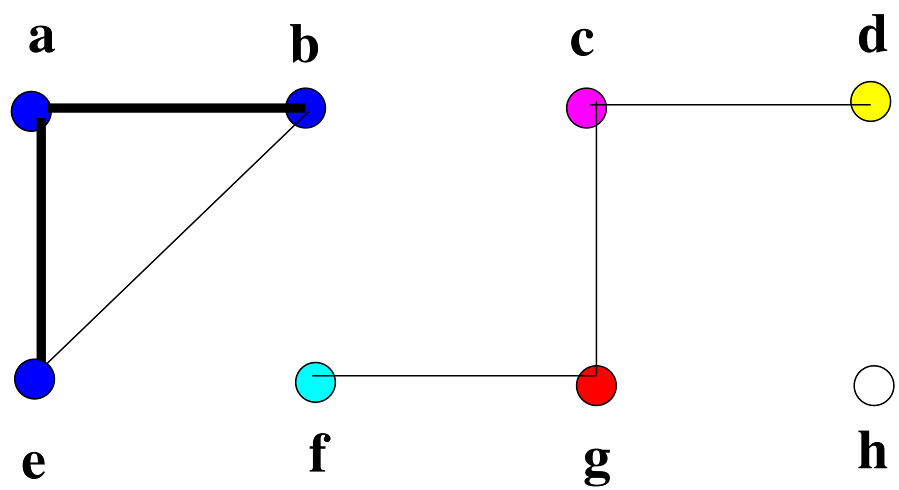
Componenti connesse di grafi indiretti

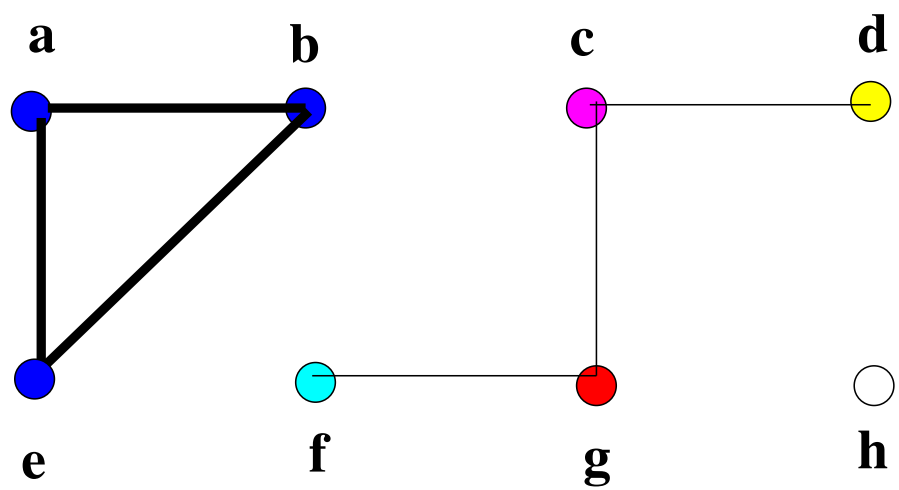
CC(G)

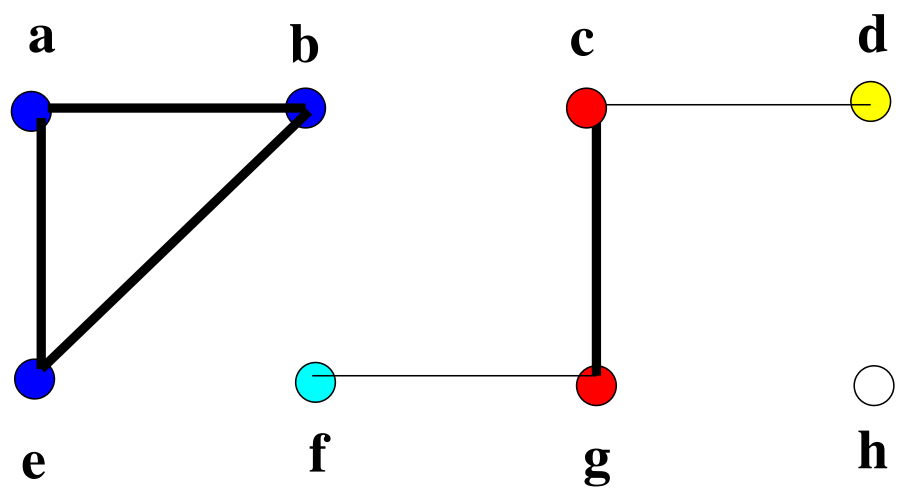
for each $v \in V[G]$ **do**
  $MakeSet(v)$
end for
for each $(u, v) \in E[G]$ **do**
  if $FindSet(u) \neq FindSet(v)$ **then**
    $Union(u, v)$
  end if
end for

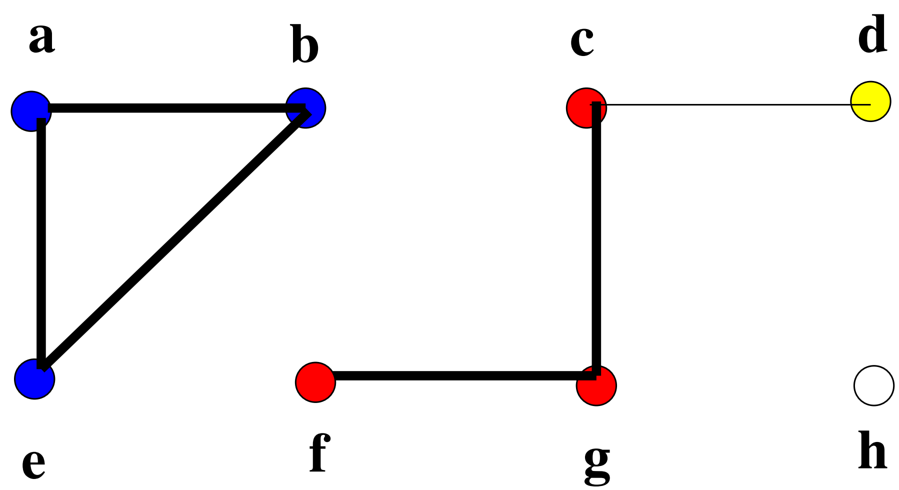

a
b
c
d
e
f
g
h

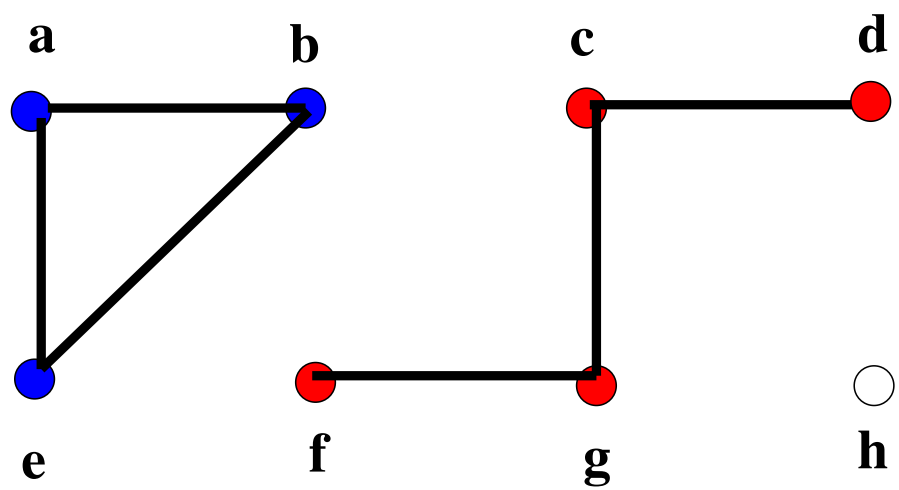

a
b
c
d
e
f
g
h


a
b
c
d
e
f
g
h


a
b
c
d
e
f
g
h


a
b
c
d
e
f
g
h


a
b
c
d
e
f
g
h


a
b
c
d
e
f
g
h

Esercizio *Si calcoli la complessità della procedura* CC *nei seguenti tre casi:*

1. *le componenti connesse sono rappresentate mediante liste senza l'ausilio di alcuna euristica;*
2. *le componenti connesse sono rappresentate mediante liste e adotto l'euristica di unione pesata;*
3. *le componenti connesse sono rappresentate mediante alberi e adotto entrambe le euristiche di unione per rango e compressione del cammino.*

Dato un grafo $G = (V, E)$ con k componenti connesse, sia $n = |V|$ e $m = |E|$. La procedura *ConnectedComponents* esegue:

- n operazioni di *MakeSet*;
- $2m$ operazioni di *Findset*;
- $n - k$ operazioni di *Union*. Infatti, il numero di unioni fatte su una singola componente connessa è uno in meno del numero di nodi della componente connessa. Dunque il numero totale di unioni è pari al numero di nodi del grafo meno il numero di componenti connesse. Nel caso pessimo, $k = 1$ e quindi faccio $n - 1$ unioni.

1. Nel primo caso, la complessità aggregata risulta $O(n^2 + f)$, e dunque la complessità della procedura ammonta a
$$O(n^2 + m)$$
2. Nel secondo caso, la complessità aggregata risulta $O(n \log n + f)$. Dunque la complessità della procedura vale
$$O(n \log n + m)$$
3. Nel terzo caso, la complessità aggregata risulta $O((n + f) \cdot \alpha(n))$. Dunque la complessità della procedura risulta
$$O((n + m) \cdot \alpha(n))$$
quindi **praticamente** uguale al costo di una visita in profondità del grafo.