ESERCIZIO DI ASD DEL 1 DICEMBRE 2008

Fusione di Liste

Algoritmo per la Fusione di Liste. Siano L_1 ed L_2 due liste ordinate in cui ogni elemento x ha un campo $key[x]$ contenente una chiave intera ed un campo $next[x]$ che punta al successore di x. Consideriamo una lista ausiliaria M che all'inizio sarà vuota ed alla fine conterrà il risultato. Useremo anche due puntatori, $temp_1$ e $temp_2$ per scandire le due liste L_1 ed L_2. Ad ogni iterazione dovremo copiare in fondo alla lista M il più piccolo tra l'elemento puntato da $temp_1$ e quello puntato da $temp_2$, e spostare avanti il puntatore relativo all'elemento copiato. Per copiare in fondo alla lista M è opportuno che M abbia sia il campo $head[M]$ che punta all'inizio di M, che il campo $tail[M]$ che punta alla fine di M.

Algorithm 1 FONDI(L_1, L_2)

```
1: M ←CREA_LISTA_VUOTA()
2: temp_1 ← head[L_1]
3: temp_2 ← head[L_2]
4: while temp_1 ≠ NIL and temp_2 ≠ NIL do
5:    if key[temp_1] < key[temp_2] then
6:       INSERISCI_IN_FONDO(M, key[temp_1])
7:       temp_1 ← next[temp_1]
8:    else
9:       INSERISCI_IN_FONDO(M, key[temp_2])
10:      temp_2 ← next[temp_2]
11:   end if
12: end while
13: if temp_1 = NIL then
14:   while temp_2 ≠ NIL do
15:      INSERISCI_IN_FONDO(M, key[temp_2])
16:      temp_2 ← next[temp_2]
17:   end while
18: end if
19: if temp_2 = NIL then
20:   while temp_1 ≠ NIL do
21:      INSERISCI_IN_FONDO(M, key[temp_1])
22:      temp_1 ← next[temp_1]
23:   end while
24: end if
25: return M
```

Date: 1 Dicembre 2008.

Algorithm 2 CREA_LISTA_VUOTA()

1: $new(N)$
2: $head[N] \leftarrow NIL$
3: $tail[N] \leftarrow NIL$
4: return N

Algorithm 3 INSERISCI_IN_FONDO(N, i)

1: $new(x)$
2: $key[x] \leftarrow i$
3: $next[x] \leftarrow NIL$
4: $next[tail[N]] \leftarrow x$
5: $tail[N] \leftarrow x$

Correttezza.

Invariante 1. *All'inizio della i-esima iterazione del primo ciclo while la lista M contiene $i-1$ elementi, è ordinata, ed inoltre $temp_1$ e $temp_2$ puntano a due liste che contengono solo elementi maggiori o uguali rispetto agli elementi che compaiono in M.*

Dimostrazione. Procediamo per induzione su i.
BASE. $i = 1$. All'inizio della prima iterazione del ciclo while la lista M è vuota, quindi vale la tesi.
PASSO.

HpInd) all'inizio della j-esima iterazione del primo ciclo while lista M contiene $j-1$ elementi, è ordinata, ed inoltre $temp_1$ e $temp_2$ puntano a due liste che contengono solo elementi maggiori o uguali rispetto agli elementi che compaiono in M.

Ts) all'inizio della $j+1$-esima iterazione del primo ciclo while lista M contiene j elementi, è ordinata, ed inoltre $temp_1$ e $temp_2$ puntano a due liste che contengono solo elementi maggiori o uguali rispetto agli elementi che compaiono in M.

Dobbiamo esaminare ciò che accade durante la j-esima iterazione del ciclo while. Se $key[temp_1]$ è minore di $key[temp_2]$, allora $key[temp_1]$ viene copiata in fondo ad M e $temp_1$ viene spostato in avanti. Per ipotesi induttiva $key[temp_1]$ è maggiore o uguale di tutti gli elementi di M, ed M è ordinata, quindi alla fine dell'iterazione M continua ad essere ordinata e contiene j elementi. Siccome L_1 ed L_2 erano ordinate, $temp_1$ e $temp_2$ puntano ad elementi maggiori o uguali degli elementi di M. Quindi vale la tesi. Analogamente si dimostra che vale la tesi nel caso $key[temp_2] \leq key[temp_1]$. □

Invariante 2. *All'inizio della i-esima iterazione del secondo ciclo while la lista M è ordinata, ed inoltre $temp_1$ e $temp_2$ puntano a due liste che contengono solo elementi maggiori o uguali rispetto agli elementi che compaiono in M.*

Invariante 3. *All'inizio della i-esima iterazione del terzo ciclo while la lista M è ordinata, ed inoltre $temp_1$ e $temp_2$ puntano a due liste che contengono solo elementi maggiori o uguali rispetto agli elementi che compaiono in M.*

La dimostrazione di questi due invarianti è analoga a quella del primo invariante.

Theorem 1. *Se L_1 ed L_2 sono due liste ordinate, allora* FONDI(L_1, L_2) *termina sempre restituendo una lista ordinata che contiene tutti e soli gli elementi di L_1 ed L_2.*

Dimostrazione. FONDI(L_1, L_2) termina sempre in quanto ad ogni iterazione viene copiato in M un elemento di L_1 o un elemento di L_2 e l'elemento copiato non verrà più considerato.

La lista M contiene tutti e soli gli elementi di L_1 ed L_2 in quanto i puntatori $temp_1$ e $temp_2$ vengono spostati in avanti solo dopo che un elemento è stato copiato.

Infine la lista M è ordinata come segue dall'Invariante 3. □

Complessità. Dato che ogni elemento di L_1 ed ogni elemento di L_2 viene esaminato esattamente una volta, la procedura ha complessità $\Theta(|L_1| + |L_2|)$, dove $|L_1|$ è la lunghezza di L_1 e $|L_2|$ è la lunghezza di L_2.