Esame di Algoritmi e Strutture di Dati (A)
Lunedì 8 Luglio 2002

Nome:
Cognome:
Matricola:

Per un buon esito dell'esame, si consiglia di:

- scrivere in forma leggibile il proprio nome, cognome e matricola sul testo del compito e su ogni foglio consegnato;
- provare gli esercizi prima in brutta copia. Ricopiarli in bella copia e consegnare quest'ultima oltre al testo del compito;
- non fatevi prendere dal panico, e neppure dalla noia. Un giusto livello di tensione è ciò che serve;
- svolgere il compito individualmente. Passare copiando è dannoso per voi, e irrilevante per il docente.

Esercizio 1 (*Punti 24*)

Si consideri la struttura matematica insieme. *Si rappresenti un insieme di numeri mediante una lista in cui il campo chiave di ogni oggetto della lista contiene un elemento dell'insieme. Si considerino i seguenti due casi:*

1. *la lista che rappresenta l'insieme non è ordinata (cioè gli oggetti nella lista sono inseriti in ordine qualsiasi);*
2. *la lista che rappresenta l'insieme è ordinata (cioè gli oggetti nella lista sono inseriti in ordine crescente di chiave).*

In entrambi i casi, si scriva la procedura $SetUnion(L_1, L_2, L_3)$ *che mette nella lista* L_3*, inizialmente vuota, l'unione degli insiemi contenuti nelle liste* L_1 *e* L_2*. Si calcoli in entrambi i casi la complessità computazionale della procedura.*

Soluzione

Nel caso della lista non ordinata abbiamo la seguente procedura:

Algoritmo 1 SetUnion($\mathbf{L_1}, \mathbf{L_2}, \mathbf{L_3}$)

SetUnion($\mathbf{L_1}, \mathbf{L_2}, \mathbf{L_3}$)

```
1:  x ← head(L_1)
2:  while x ≠ NIL do
3:      x' ← Copy(x)
4:      next[x'] ← NIL
5:      ListInsert(L_3, x')
6:      x ← next[x]
7:  end while
8:  x ← head(L_2)
9:  while x ≠ NIL do
10:     if ListSearch(L_1, key[x]) = NIL then
11:         x' ← Copy(x)
12:         next[x'] ← NIL
13:         ListInsert(L_3, x')
14:     end if
15:     x ← next[x]
16: end while
17: return head(L_3)
```

L'istruzione $x' \leftarrow Copy(x)$ crea una copia dell'oggetto x e la mette in x'. Tale istruzione ha costo costante. Siano n la lunghezza di L_1 e m la lunghezza di L_2. Dato che la complessità di *ListInsert* è costante e la complessità di *ListSearch* è lineare, la complessità di *SetUnion* risulta $\Theta(n + nm) = \Theta(nm)$.

Nel caso della lista ordinata abbiamo la seguente procedura. Siano n la lunghezza di L_1 e m la lunghezza di L_2. Dato che la complessità di $ListInsert$ è costante, la complessità della versione di $SetUnion$ che segue risulta $\Theta(n+m)$.

Algoritmo 2 SetUnion($\mathbf{L_1}, \mathbf{L_2}, \mathbf{L_3}$)

SetUnion($\mathbf{L_1}, \mathbf{L_2}, \mathbf{L_3}$)

```
1: x ← head(L_1)
2: y ← head(L_2)
3: while ((x ≠ NIL) or (y ≠ NIL)) do
4:    if x = NIL then
5:       y' ← Copy(y)
6:       next[y'] ← NIL
7:       ListInsert(L_3, y')
8:       y ← next[y]
9:    end if
10:   if y = NIL then
11:      x' ← Copy(x)
12:      next[x'] ← NIL
13:      ListInsert(L_3, x')
14:      x ← next[x]
15:   end if
16:   if (x ≠ NIL and y ≠ NIL) then
17:      if key[x] < key[y] then
18:         x' ← Copy(x)
19:         next[x'] ← NIL
20:         ListInsert(L_3, x')
21:         x ← next[x]
22:      end if
23:      if key[y] < key[x] then
24:         y' ← Copy(y)
25:         next[y'] ← NIL
26:         ListInsert(L_3, y')
27:         y ← next[y]
28:      end if
29:      if key[x] = key[y] then
30:         x' ← Copy(x)
31:         next[x'] ← NIL
32:         ListInsert(L_3, x')
33:         x ← next[x]
34:         y ← next[y]
35:      end if
36:   end if
37: end while
38: ListReverse(L_3)
39: return head(L_3)
```

Esercizio 2 (*Punti 6*)

Si consideri un'applicazione di una bacheca universitaria elettronica che deve ordinare i voti degli studenti espressi in trentesimi. Quale algoritmo di ordinamento usereste per implementare tale applicazione? Giustificare la risposta.

Soluzione

E' opportuno usare *CountingSort*. Infatti, gli elementi da ordinare sono numeri naturali minori o uguali a 30. Con queste ipotesi, *CountingSort* ha complessità lineare nel numero di elementi da ordinare.

Esame di Algoritmi e Strutture di Dati (B)
Lunedì 8 Luglio 2002

Nome:
Cognome:
Matricola:

Per un buon esito dell'esame, si consiglia di:

- scrivere in forma leggibile il proprio nome, cognome e matricola sul testo del compito e su ogni foglio consegnato;
- provare gli esercizi prima in brutta copia. Ricopiarli in bella copia e consegnare solo quest'ultima;
- non fatevi prendere dal panico, e neppure dalla noia. Un giusto livello di tensione è ciò che serve;
- svolgere il compito individualmente. Passare copiando è dannoso per voi, e irrilevante per il docente.

Esercizio 3 (*Punti 24*)

Si consideri la struttura matematica insieme. *Si rappresenti un insieme di numeri mediante una lista in cui il campo chiave di ogni oggetto della lista contiene un elemento dell'insieme. Si considerino i seguenti due casi:*

1. *la lista che rappresenta l'insieme non è ordinata (cioè gli oggetti nella lista sono inseriti in ordine qualsiasi);*
2. *la lista che rappresenta l'insieme è ordinata (cioè gli oggetti nella lista sono inseriti in ordine crescente di chiave).*

In entrambi i casi, si scriva la procedura $SetIntersection(L_1, L_2, L_3)$ *che mette nella lista* L_3*, inizialmente vuota, l'intersezione degli insiemi contenuti nelle liste* L_1 *e* L_2*. Si calcoli in entrambi i casi la complessità computazionale della procedura.*

Soluzione

Nel caso della lista non ordinata abbiamo la seguente procedura:

Algoritmo 3 $SetIntersection(L_1, L_2, L_3)$

```
SetIntersection(L1, L2, L3)
1: x ← head(L1)
2: while x ≠ NIL do
3:    if ListSearch(L2, key[x]) ≠ NIL then
4:       x' ← Copy(x)
5:       next[x'] ← NIL
6:       ListInsert(L3, x')
7:    end if
8:    x ← next[x]
9: end while
```

Siano n la lunghezza di L_1 e m la lunghezza di L_2. Dato che la complessità di $ListInsert$ è costante e la complessità di $ListSearch$ è lineare, la complessità di $SetIntersection$ risulta $\Theta(nm)$.

Nel caso della lista ordinata abbiamo la seguente procedura:

Algoritmo 4 SetIntersection(L_1, L_2, L_3)

SetIntersection(L_1, L_2, L_3)

```
1:  x ← head(L_1)
2:  y ← head(L_2)
3:  while ((x ≠ NIL) and (y ≠ NIL)) do
4:     if key[x] < key[y] then
5:        x ← next[x]
6:     end if
7:     if key[y] < key[x] then
8:        y ← next[y]
9:     end if
10:    if key[x] = key[y] then
11:       x' ← Copy(x)
12:       next[x'] ← NIL
13:       ListInsert(L_3, x')
14:       x ← next[x]
15:       y ← next[y]
16:    end if
17: end while
18: ListReverse(L_3)
```

Siano n la lunghezza di L_1 e m la lunghezza di L_2. Dato che la complessità di $ListInsert$ è costante, la complessità di $SetIntersection$ risulta $\Theta(n+m)$.

Esercizio 4 (*Punti 6*)

Si consideri un'applicazione di una bacheca universitaria elettronica che deve ordinare i voti degli studenti espressi in trentesimi. Quale algoritmo di ordinamento usereste per implementare tale applicazione? Giustificare la risposta.

Soluzione

E' opportuno usare $CountingSort$. Infatti, gli elementi da ordinare sono numeri naturali minori o uguali a 30. Con queste ipotesi, $CountingSort$ ha complessità lineare nel numero di elementi da ordinare.