Esame di Algoritmi e Strutture di Dati (A)
Martedì 7 Gennaio 2003

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 20*)

Due insiemi si dicono disgiunti *se non posseggono elementi in comune. Ideare una struttura di dati per rappresentare insiemi che permetta di implementare l'operazione di unione tra insiemi disgiunti con complessità pessima costante* $\Theta(1)$. *Scrivere lo pseudocodice della procedura di complessità costante che implementa l'unione disgiunta.*

Soluzione

Rappresento un insieme mediante una lista circolare (il predecessore della testa è la coda, e il successore della coda è la testa). Ogni elemento della lista contiene un elemento dell'insieme. Date due liste circolari L e M che contengono insiemi disgiunti, la seguente procedura $DisjointUnion(L, M)$ ritorna un puntatore alla lista che contiene l'unione disgiunta degli insiemi contenuti in L e M. La procedura semplicemente concatena la lista M alla fine della lista L. Il costo della procedura è costante.

Algoritmo 1 DisjointUnion(L,M)

DisjointUnion(L,M)

```
1: if ListEmpty(L) then
2:    return head(M)
3: end if
4: if ListEmpty(M) then
5:    return head(L)
6: end if
7: y ← prev[head(L)]
8: z ← prev[head(M)]
9: next[y] ← head(M)
10: prev[head(M)] ← y
11: prev[head(L)] ← z
12: next[z] ← head(L)
13: return head(L)
```

Esercizio 2 (*Punti 10*)

Sia A un vettore. Si consideri la seguente procedura di ordinamento $TreeSort(A)$. La procedura consiste di due passi:

1. *gli elementi di A vengono inseriti in un albero binario di ricerca T usando la procedura $TreeInsert$;*
2. *l'albero T viene visitato in ordine intermedio e gli elementi di T vengono inseriti nel vettore A.*

Si calcoli la complessità ottima, media e pessima di $TreeSort$.

Soluzione

Sia n la lunghezza di A. Nel caso pessimo, l'albero T risultante dopo gli n inserimenti degli elementi di A è una sequenza. In tal caso, dopo i inserimenti, con $1 \leq i \leq n$, l'altezza dell'albero corrente è pari a i. La complessità di $TreeInsert$ è dell'ordine dell'altezza dell'albero in cui il nodo viene inserito. Dunque la complessità dell'inserimento degli n elementi di A nell'albero T è pari a

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

La complessità pessima della visita intermedia di T è $\Theta(n)$. Dunque la complessità pessima di $TreeSort$ risulta $\Theta(n^2 + n) = \Theta(n^2)$.

Nel caso ottimo, l'albero T risultante dopo gli n inserimenti degli elementi di A è un albero completo fino al penultimo livello. In tal caso, dopo i inserimenti, con $1 \leq i \leq n$, l'altezza dell'albero corrente è pari a $\lfloor \log i \rfloor$. La complessità di $TreeInsert$ è dell'ordine dell'altezza dell'albero in cui il nodo viene inserito. Dunque la complessità dell'inserimento degli n elementi di A nell'albero T è pari a

$$\sum_{i=1}^{n} \Theta(\log i) = \Theta(\sum_{i=1}^{n} \log i) = \Theta(\log \prod_{i=1}^{n} i) = \Theta(\log n!) = \Theta(n \log n).$$

La complessità ottima della visita intermedia di T è $\Theta(n)$. Dunque la complessità ottima di $TreeSort$ risulta $\Theta(n \log n + n) = \Theta(n \log n)$.

Nel caso medio, l'albero T risultante dopo gli n inserimenti degli elementi di A è un albero bilanciato con altezza logaritmica nel numero dei nodi. Dunque la complessità media è pari alla complessità ottima, cioè $\Theta(n \log n)$.

Esame di Algoritmi e Strutture di Dati (B)
Martedì 7 Gennaio 2003

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 20*)

Due insiemi si dicono disgiunti *se non posseggono elementi in comune. Ideare una struttura di dati per rappresentare insiemi che permetta di implementare l'operazione di unione tra insiemi disgiunti con complessità pessima costante* $\Theta(1)$. *Scrivere lo pseudocodice della procedura di complessità costante che implementa l'unione disgiunta.*

Soluzione

Rappresento un insieme mediante una lista circolare (il predecessore della testa è la coda, e il successore della coda è la testa). Ogni elemento della lista contiene un elemento dell'insieme. Date due liste circolari L e M che contengono insiemi disgiunti, la seguente procedura $DisjointUnion(L, M)$ ritorna un puntatore alla lista che contiene l'unione disgiunta degli insiemi contenuti in L e M. La procedura semplicemente concatena la lista M alla fine della lista L. Il costo della procedura è costante.

Algoritmo 2 DisjointUnion(L,M)

DisjointUnion(L,M)

```
1: if ListEmpty(L) then
2:    return head(M)
3: end if
4: if ListEmpty(M) then
5:    return head(L)
6: end if
7: y ← prev[head(L)]
8: z ← prev[head(M)]
9: next[y] ← head(M)
10: prev[head(M)] ← y
11: prev[head(L)] ← z
12: next[z] ← head(L)
13: return head(L)
```

Esercizio 4 (*Punti 10*)

Sia A un vettore. Si consideri la seguente procedura di ordinamento $TreeSort(A)$. La procedura consiste di due passi:

1. *gli elementi di A vengono inseriti in un albero binario di ricerca T usando la procedura $TreeInsert$;*
2. *l'albero T viene visitato in ordine intermedio e gli elementi di T vengono inseriti nel vettore A.*

Si calcoli la complessità ottima, media e pessima di $TreeSort$.

Soluzione

Sia n la lunghezza di A. Nel caso pessimo, l'albero T risultante dopo gli n inserimenti degli elementi di A è una sequenza. In tal caso, dopo i inserimenti, con $1 \leq i \leq n$, l'altezza dell'albero corrente è pari a i. La complessità di $TreeInsert$ è dell'ordine dell'altezza dell'albero in cui il nodo viene inserito. Dunque la complessità dell'inserimento degli n elementi di A nell'albero T è pari a

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

La complessità pessima della visita intermedia di T è $\Theta(n)$. Dunque la complessità pessima di $TreeSort$ risulta $\Theta(n^2 + n) = \Theta(n^2)$.

Nel caso ottimo, l'albero T risultante dopo gli n inserimenti degli elementi di A è un albero completo fino al penultimo livello. In tal caso, dopo i inserimenti, con $1 \leq i \leq n$, l'altezza dell'albero corrente è pari a $\lfloor \log i \rfloor$. La complessità di $TreeInsert$ è dell'ordine dell'altezza dell'albero in cui il nodo viene inserito. Dunque la complessità dell'inserimento degli n elementi di A nell'albero T è pari a

$$\sum_{i=1}^{n} \Theta(\log i) = \Theta(\sum_{i=1}^{n} \log i) = \Theta(\log \prod_{i=1}^{n} i) = \Theta(\log n!) = \Theta(n \log n).$$

La complessità ottima della visita intermedia di T è $\Theta(n)$. Dunque la complessità ottima di $TreeSort$ risulta $\Theta(n \log n + n) = \Theta(n \log n)$.

Nel caso medio, l'albero T risultante dopo gli n inserimenti degli elementi di A è un albero bilanciato con altezza logaritmica nel numero dei nodi. Dunque la complessità media è pari alla complessità ottima, cioè $\Theta(n \log n)$.