Analisi e progettazione di algoritmi

(III anno Laurea Triennale - a.a. 2018/19)

Soluzioni prova scritta 27 giugno 2019

Esercizio 1 - Ordinamenti

1. Pseudocodice dei passi 1 e 2:

```
p = l.dequeue()
l1 = make_empty()
l2 = make_empty()
while (l.size()>0)
    x = l.dequeue()
    if (x <= p) l1.enqueue(x) else l2.enqueue(x)
```

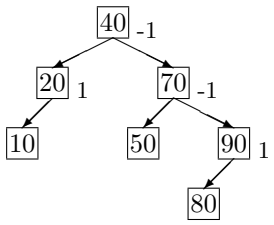
2. Sia $n = |l|$. Le istruzioni eseguite prima del ciclo hanno complessità temporale costante. Il ciclo è eseguito esattamente $n-1$ volte (tante volte quanti gli elementi di l, tranne p che è già stato tolto) e le istruzioni eseguite hanno costo costante. La complessità temporale è perciò in $\Theta(n)$.

   La complessità spaziale è $O(1)$ perché durante il ciclo il numero totale di nodi nelle tre liste l, l_1, l_2 è in ogni momento pari a $n-1$ (un elemento viene tolto da l ed aggiunto a l_1 oppure a l_2); le variabili aggiuntive sono solo p e x.

3. Esempio che non è stabile: $l = 10_A, 10_B, 20, 30$. Abbiamo $p = 10_A$. Si formano $l_1 = 10_B$, $l_2 = 20, 30$, che vengono ordinate ricorsivamente (e restano uguali perché già ordinate). Il risultato finale è $l = l_1 \cdot p \cdot l_2 = 10_B, 10_A, 20, 30$ in cui l'ordine reciproco dei due elementi di chiave 10 è stato invertito.

Esercizio 2 - Strutture dati

1. Fattori di bilanciamento dei nodi interni:

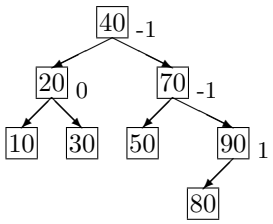


2. Una sequenza di inserimento: ad esempio 40,20,70,10,50,90,80. Non è unica (per es. posso scambiare 50 e 90).

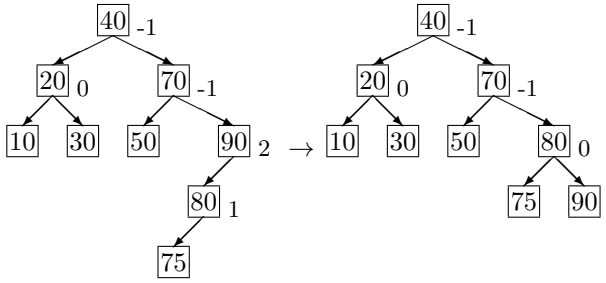
   Attenzione: se fosse un BST andrebbero bene anche sequenze del tipo 40,20,10... ma questo è un AVL: in tal caso l'inserimento di 10 provoca sbilanciamento e rotazione!

3. Inserimenti:

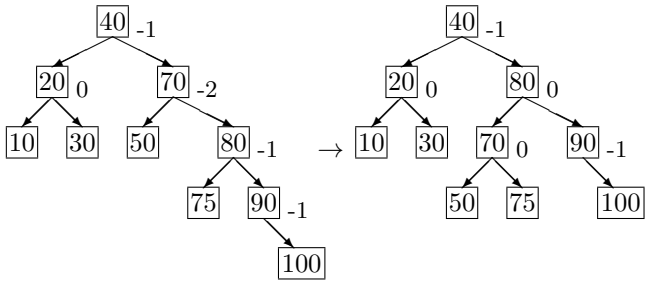
   Inserisco 30 come in un albero binario di ricerca, creando un nuovo nodo figlio destro di 10. Risalendo dal nuovo nodo verso la radice, ricalcolo i fattori di bilanciamento. I nuovi fattori sono tutti nell'intervallo accettabile (tra -1 e 1) per cui non occorrono rotazioni:



   Inserisco 75 come in un albero binario di ricerca, creando un nuovo nodo figlio sinistro di 80. Risalendo, ricalcolo i fattori di bilanciamento. Il nodo contenente 90 risulta sbilanciato (fattore 2). Faccio una rotazione semplice verso destra (che coinvolge i nodi 80,90):



   Inserisco 100 come in un albero binario di ricerca, creando un nuovo nodo figlio destro di 90. Risalendo ricalcolo i fattori di bilanciamento. Il nodo contenente 70 risulta sbilanciato (fattore -2). Faccio una rotazione doppia verso sinistra (che coinvolge i nodi 70,80,90):



Esercizio 3 – Grafi

Nella prima colonna indichiamo il nodo estratto.

	A	B	C	D	E	F	G
	0	∞	∞	∞	∞	∞	∞
A	-	7	∞	5	∞	∞	∞
D	-	7	∞	-	20	11	∞
B	-	-	15	-	14	11	∞
F	-	-	15	-	14	-	22
E	-	-	15	-	-	-	22
C	-	-	-	-	-	-	22
G	-	-	-	-	-	-	-

Esercizio 4 – Tecniche algoritmiche

1. Definizione induttiva di $\exists CS[i,j,k]$ per ogni $0 \leq i,j,k \leq n$:

   Base

   $\exists CS[i,j,0] = T$ per ogni $0 \leq i,j \leq n$
   (la sequenza vuota è sempre una sottosequenza comune)
   $\exists CS[0,j,k] = F$ per ogni $k > 0$ se $i = 0$ oppure $j = 0$
   (se una delle due stringhe è vuota non ci sono sottostringhe comuni di lunghezza > 0)

Passo induttivo per ogni $0 < i, j, k \leq n$

$\exists CS[i,j,k] = \begin{cases} \exists CS[i-1,j-1,k-1] & \text{se } X[i] = Y[j] \\ \exists CS[i-1,j,k] \vee \exists CS[i,j-1,k] & \text{se } X[i] \neq Y[j] \end{cases}$ (analogamente a quanto visto per LCS)

Analogamente a quanto visto per LCS la correttezza è basata sul principio di induzione forte.

2. Un corrispondente algoritmo di programmazione dinamica è il seguente.

```
for (i = 0; i <= n; i++)
  for (j = 0; i <= n; i++) Exists[i,j,0] = true

for (k = 1; k <= n; k++)
  for (j = 1; j <= n; j++) Exists[0,j,k] = false
  for (i = 1; i <= n; j++) Exists[i,0,k] = false

for (k = 1; k <= n; k++)
  for (i = 1; i <= n; i++)
    for (j = 1; j <= n; j++)
      if (X[i] = Y[j]) Exists [i,j,k] = Exists[i-1,j-1,k-1]
      else Exists[i-1,j,k] || Exists[i,j-1,k]
```

3. Per ottenere anche una delle sottosequenze di di $X[1..i]$ e $Y[1..j]$ di lunghezza (almeno) k, se ne esistono, si può utilizzare una tecnica analoga a quella vista per LCS, ossia memorizzare anche un puntatore alla casella adiacente in diagonale (se $X[i] = Y[j]$) oppure a sinistra o in alto (se $X[i] \neq Y[j]$, se il valore della casella è `true`.

Esercizio 5 - Analisi di complessità

1. Possiamo dire che il problema $\mathcal{P}$ è $O(n^2)$. Non possiamo dire nulla sul limite inferiore.

2. Dal teorema master sappiamo che:

   $a < 8 \;\; T(n) = \Theta(n^2)$

   $a = 8 \;\; T(n) = \Theta(n^2 \log_4 n)$

   $a > 8 \;\; T(n) = \Theta(n^{\log_4 a})$ (con $\log_4 a > 2$)

   Quindi, l'algoritmo iterativo è preferibile (asintoticamente) per $a \geq 8$.