Complementi di Algoritmi e Strutture Dati

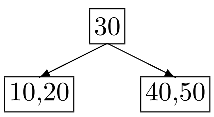
(III anno Laurea Triennale - a.a. 2017/18)

Soluzioni della prova scritta 28 giugno 2018

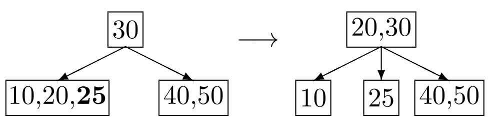
NB: I punteggi sono indicativi.

Esercizio 1 – Alberi (punti 6)

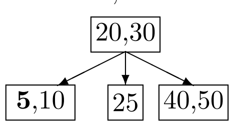
Inserimenti successivi nel seguente 2-3 albero iniziale:



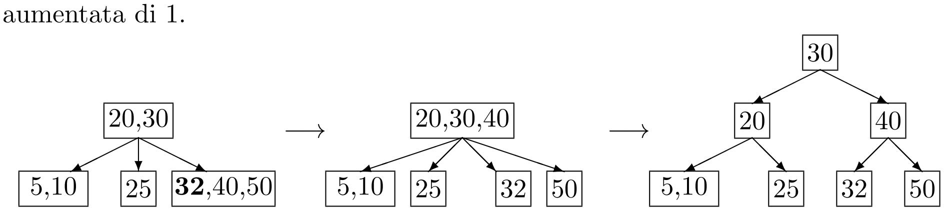
1. Inserisco 25, va nella foglia a sinistra, che passa da 2 a 3 elementi: overflow. La sorella ha già 2 elementi, quindi divido la foglia in overflow promuovendo l'elemento centrale.



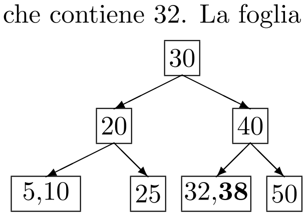
2. Inserisco 5, va nella foglia a sinistra, che passa da 1 a 2 elementi.



3. Inserisco 32, va nella foglia a destra, che passa da 2 a 3 elementi: overflow. Divido la foglia promuovendo l'elemento centrale (40). Adesso il nodo padre è in overflow. Lo divido promuovendo il suo elemento centrale (30). Avendo diviso la radice, l'altezza dell'albero è aumentata di 1.



4. Inserisco 38, che segue il percorso a destra di 30, a sinistra di 40, e va ad inserirsi nella foglia che contiene 32. La foglia passa da 1 a 2 elementi.



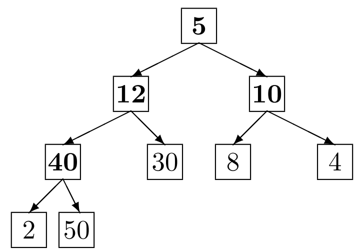
Esercizio 2 – Sorting (punti 6)

Prima fase dell'algoritmo heapsort sul seguente array:

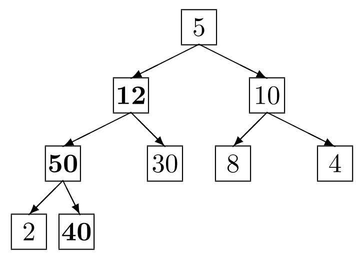
5	12	10	40	30	8	4	2	50

Versione con *heapify*

Pensiamo l'array come se fosse un albero. Le caselle dell'array che sono foglie (30,8,4,2,50) sono già heap fatti da solo un nodo. Devo esaminare gli altri nodi a ritroso (nell'ordine 40,10,12,5) e rendere heap l'albero di cui sono radice, eseguendo delle *moveDown*.

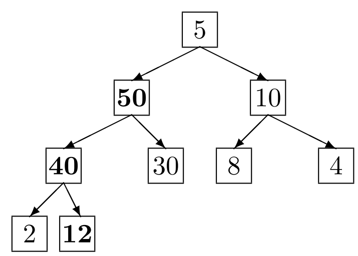


Eseguo *moveDown(40)*: scambio 40 con il figlio di chiave massima (50).



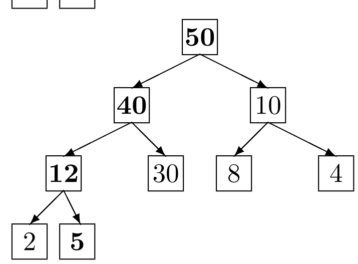
Eseguo *moveDown(10)*: 10 è maggiore di entrambi i suoi figli, non sono necessari scambi.

Eseguo *moveDown(12)*: scambio 12 con 50 e poi con 40.



Eseguo *moveDown(5)*: scambio 5 con 50, poi 40, poi 10. Risultato finale (array e albero):

50	40	10	12	30	8	4	2	5



Esercizio 3 – Grafi (Punti 7)

Indichiamo nella prima colonna il nodo via via estratto.

	1	2	3	4	5	6
	0	∞	∞	∞	∞	∞
0	-	7	1	∞	∞	∞
3	-	6	-	∞	3	8
5	-	5	-	8	-	8
2	-	-	-	8	-	6
6	-	-	-	8	-	-
4	-	-	-	-	-	-

Esercizio 4 – Tecniche algoritmiche (Punti 7)

1. Definiamo induttivamente $P[i,j]$ nel modo seguente.

    $P[1,1] = M[1,1]$ (siamo già arrivati)
    $P[1,j] = P[1,j-1] + M[1,j]$ per $1 < j \le n$ (se devo arrivare in una casella della prima riga posso solo spostarmi a destra)
    $P[i,1] = P[i-1,1] + M[i,1]$ per $1 < i \le n$ (se devo arrivare in una casella della prima colonna posso solo spostarmi in basso)
    $P[i,j] = \max(P[i-1,j], P[i,j-1]) + M[i,j]$ per $1 < i,j \le n$ (altrimenti ho due scelte possibili, e scelgo quella che mi fornisce il punteggio massimo)

2. Un algoritmo di programmazione dinamica basato sulla precedente definizione è il seguente.

```
P[1,1] = M[1,1]
for (j=2;j<=n;j++) P[1,j]= P[1,j-1]+M[1,j]
for (i=2;i<=n;i++) P[i,1]= P[i-1,1]+M[i,1]
for (i=2;i<=n;i++)
  for (j=2;j<=n;j++) P[i,j]= max(P[i-1,j],P[i,j-1])+M[i,j]
```

3. Per ottenere anche la sequenza di mosse corrispondente al massimo punteggio possiamo utilizzare un'altra matrice $D[2..n, 2..n]$ dove memorizziamo in ogni casella $\Leftarrow$ se previeniamo da destra, $\Uparrow$ se proveniamo dall'alto:

```
P[1,1] = M[1,1]
for (j=2;j<=n;j++) P[1,j]= P[1,j-1]+M[1,j]; D[1,j]= ⇐
for (i=2;i<=n;i++) P[i,1]= P[i-1,1]+M[i,1]; D[i,1]= ⇑
for (i=2;i<=n;i++)
  for (j=2;j<=n;j++)
    if (P[i-1,j]≥P[i,j-1]) P[i,j]= P[i-1,j]+M[i,j]; D[i,j]= ⇑
    else P[i,j]= P[i,j-1]+M[i,j]; D[i,j]= ⇐
```

Esercizio 5 – NP-completezza (Punti 7)

1. Non può essere INDEPENDENT-SET $\in$ P e VERTEX-COVER $\notin$ P. Infatti, essendo INDEPENDENT-SET $\in$ NP-C, sappiamo che ogni problema in NP è riducibile a INDEPENDENT-SET, in particolare VERTEX-COVER. Quindi, se fosse INDEPENDENT-SET $\in$ P, attraverso la riduzione otterrei anche un algoritmo polinomiale per VERTEX-COVER.

2. È facile vedere che, dato un grafo non orientato $G = (V, E)$ con n nodi e un $k \in \mathbb{N}$, il grafo ha un vertex cover V' di dimensione (almeno) k se e solo se $V \setminus V'$ è un insieme indipendente (di dimensione $n-k$). Infatti richiedere che ogni arco abbia almeno un estremo in V' equivale a richiedere che nessun arco unisca due nodi in $V \setminus V'$. Quindi, un input di VERTEX-COVER dato da $G = (V, E)$ (con n nodi) e k può essere trasformato in un input $G = (V, E)$ e $n-k$ di INDEPENDENT-SET.