Introduzione agli Algoritmi
9 gennaio 2015
Prof. Emanuela Fachini (canale 1) e Prof. Irene Finocchi (canale 2)

Parte I

Le risposte non motivate non saranno prese in considerazione.
Negli esercizi di progettazione, prima di passare allo pseudocodice descrivete l'idea algoritmica soggiacente. Per gli algoritmi progettati è necessario analizzare tempo di esecuzione e correttezza.

Esercizio 1

Siano dati due array $A[1..n]$ e $B[1..m]$, contenenti rispettivamente $n \geq 1$ ed $m \geq 1$ numeri reali. Gli array sono entrambi ordinati in senso crescente e non contengono valori duplicati; tuttavia, è possibile che uno stesso valore sia presente una volta in A e una volta in B.

1. Progettare un algoritmo efficiente che stampi in ordine i numeri reali che appartengono all'unione dei valori di A e di B. L'unione è intesa in senso insiemistico, quindi gli eventuali valori presenti in entrambi devono essere stampati solo una volta. Ad esempio, se A=[1, 3, 4, 6] e B=[2, 3, 4, 7], l'algoritmo deve stampare 1, 2, 3, 4, 6, 7.

2. Analizzare la correttezza e determinare il tempo di esecuzione dell'algoritmo proposto, in funzione di n e m.

Soluzione.

Si tratta di fare una minima variante della funzione fondi (merge) utilizzata nel mergesort.

```
Unione(A,B)
n = A.length
m = B.length
i=1
j=1
while (i≤n and j≤m):
        if (A[i] < B[j]): print(A[i]); i++
        else if (A[i] == B[j]): then print(A[i]); i++;j++
              else print(B[j]);j++
while i≤n do print(A[i])
while j≤m do print(B[j])
```

La correttezza si prova in modo analogo alla dimostrazione di correttezza per la funzione merge.

Il tempo di esecuzione in tutti i casi è $\theta(n+m)$. Nel caso peggiore vengono stampati esattamente n+m elementi perché non ci sono elementi comuni.

Esercizio 2

Si determini il tempo di esecuzione T(n) della seguente funzione:

```
analizzami(int n)
        c = 1
        m = n*n
        while m>1 do
                for j=1 to m do c++
                m=m/2
        if n>1 then analizzami(n/2)
```

Il ciclo while contiene un ciclo for che viene eseguito n^2 volte la prima volta, $n^2/2$ la seconda, … Quindi in totale il ciclo while comporta l'esecuzione di $\sum_{i=0,\ldots,i=\lg n} n^2/2^i = n^2\sum_{i=0,\ldots,i=\lg n} 1/2^i = O(n^2)$ istruzioni di costo costante, perché la sommatoria estesa all'infinito converge a una costante.
Quindi la relazione di ricorrenza è $T(n) = T(n/2) + O(n^2)$.
Per risolverla bisogna esplicitare le costanti e fissare il caso base:
$T(n) = c$ per $n \leq 1$,
$T(n) = T(n/2) + dn^2$, per costanti c,d>0
Ipotizziamo che $T(n) = O(n^2)$, quindi dobbiamo dimostrare che esistono un a e un n_0 tali che $T(n) \leq an^2$, per tutti gli $n \geq n_0$, dove a e n_0 sono costanti positive. Sia n>1.
Per ipotesi induttiva sia $T(n/2) \leq a(n/2)^2$ allora $T(n) \leq a(n/2)^2 + dn^2$ e si vuole determinare se esistono un a e un n_0 tali che $a(n/2)^2 + dn^2 \leq an^2$ per ogni $n \geq n_0$. $a/4n^2 + dn^2 \leq an^2 \iff a/4 + d \leq a \iff a + 4d \leq 4a \iff 4/3d \leq a$.
Si può concludere che prendendo a=2d e n_0 =2 la tesi è vera.

Introduzione agli Algoritmi
9 gennaio 2015
Prof. Emanuela Fachini (canale 1) e Prof. Irene Finocchi (canale 2)

Parte II

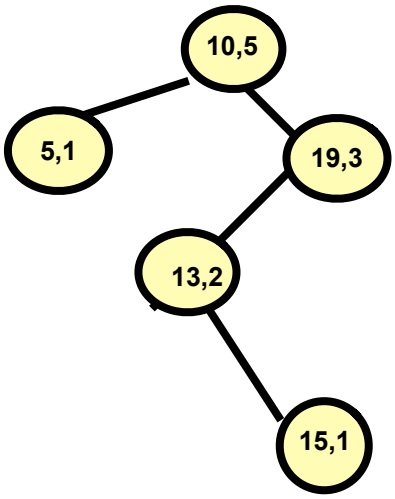
Esercizio 3

A. Qual'è il massimo numero di nodi che può avere un albero binario di altezza h?
B. Qual'è il minimo numero di nodi che può avere un albero binario di altezza h? Si motivi la risposta.
C. Se ne deducano i limiti inferiori e superiori per l'altezza, h, di un albero binario in termini del numero dei nodi, n.
D. Si dimostri che è possibile costruire un maxheap, a partire da un array di interi qualunque di n elementi, in O(n). Si descriva l'algoritmo e se ne analizzi la complessità.

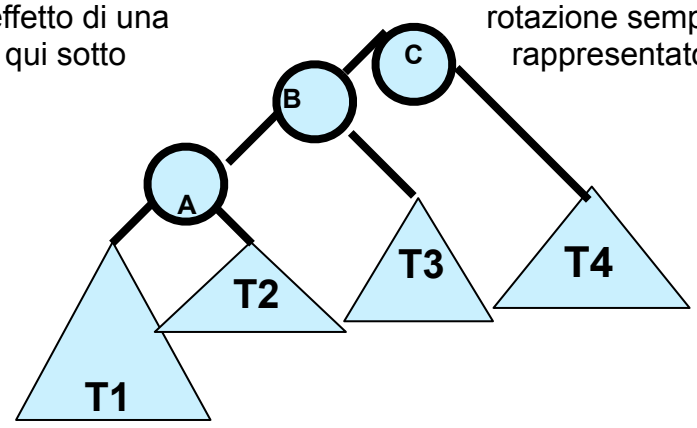
Vedere libro di testo.

Esercizio 4

Si supponga di avere un albero binario di ricerca in cui ogni nodo x ha un campo aggiuntivo, x.size, contenente il numero di nodi nel sotto albero radicato in x, quindi le foglie hanno il campo size a 1. Si sfrutti questa informazione per calcolare il numero di nodi di chiave minore di una chiave k, presente nell'albero, visitando il minor numero di nodi possibile. Nell'esempio i valori dei due campi sono separati da virgola



Si aggiungano allo pseudocodice delle rotazioni le istruzioni di aggiornamento dei campi size dei nodi coinvolti nella rotazione. Si premetta una descrizione grafica dell'effetto di una rotazione semplice a destra sul nodo C dell'albero T qui sotto rappresentato.



Introduzione agli Algoritmi
9 gennaio 2015
Prof. Emanuela Fachini (canale 1) e Prof. Irene Finocchi (canale 2)

Soluzione.

1) Poiché i nodi nel sotto albero sinistro di un nodo x hanno chiavi minori di quella di x, si può semplicemente evitare di visitare questo sotto albero quando la chiave di x è minore di k. In tal caso infatti la chiave k si trova nel sotto albero destro di T quindi basta aggiungere al risultato della chiamata sul sotto albero destro il numero dei nodi nel sotto albero sinistro, dato presente nel campo size del figlio sinistro, più 1 per il nodo stesso. Se invece la chiave è proprio quella del nodo x allora si restituisce direttamente il numero dei nodi nel suo sotto albero sinistro, dato presente nel campo size del figlio sinistro. In caso contrario, cioè se la chiave di x è maggiore di k allora si chiama la procedura sul sotto albero sinistro.

```
NumNodiMinK(T,k)
if (T==NIL) then return 0
if  (T.key == k) then if (T.left ≠ NIL ) return T.left.size else return 0
 if (T.key < k ) then
       if (T.left ≠ NIL ) then m = T.left.size else m = 0
       return NumNodiMinK(T.right,k) + m +1
else return NumNodiMinK(T.left,k)
```

Prova di correttezza.

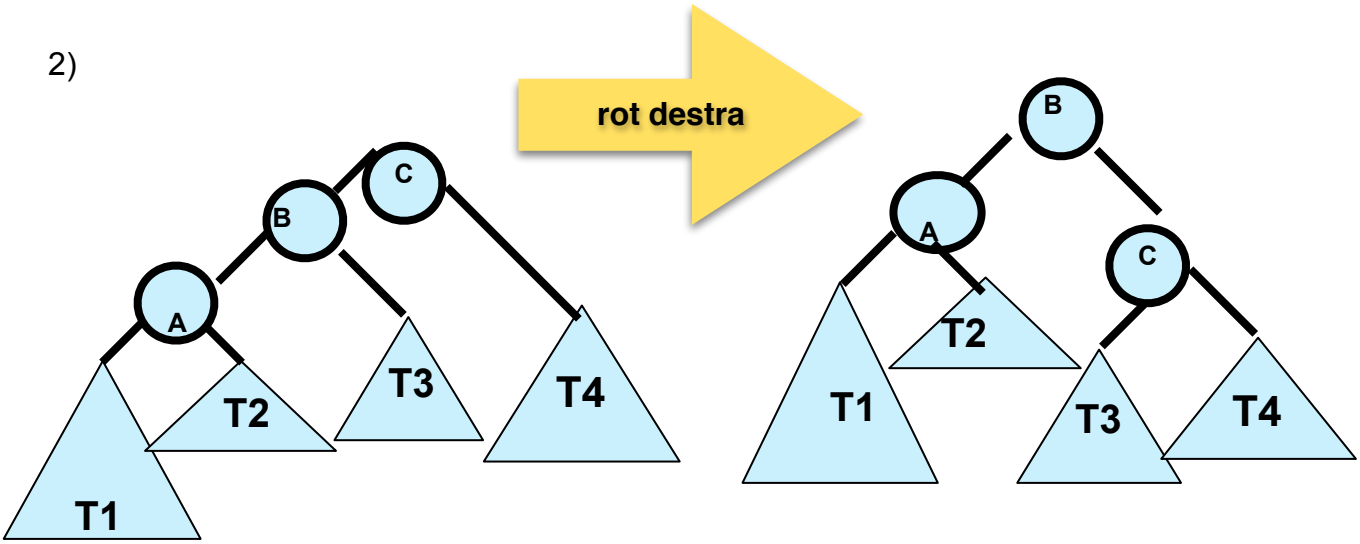
Per induzione strutturale.
Se T ha un solo nodo, allora per ipotesi deve avere chiave k e T.left == NIL, in tal caso NumNodiMinK(T,k) restituisce correttamente 0.

Sia T un albero con due sotto alberi sui quali la procedura lavora correttamente per ipotesi induttiva.
Se la chiave di T è k la procedura restituisce T.left.size cioè il numero dei nodi nel sotto albero sinistro di T, che sono tutti minori di k, oppure 0 se il sotto albero è vuoto.
Se invece la chiave di T è minore di k allora k è nel sotto albero destro e la procedura, chiamata sulla radice del sotto albero destro correttamente restituisce il numero dei nodi di chiave minore di k nel sotto albero destro, aggiungendo a questi il numero dei nodi nel sotto albero sinistro +1, per il nodo T.
Altrimenti il numero dei nodi di chiave minore di k è ottenuto direttamente dalla chiamata sul sotto albero sinistro.

Complessità.
Poiché ogni chiamata avviene o su un figlio sinistro o su un figlio destro al più si ha un numero di chiamate pari all'altezza dell'albero, quindi la complessità asintotica è O(h).

Introduzione agli Algoritmi
9 gennaio 2015
Prof. Emanuela Fachini (canale 1) e Prof. Irene Finocchi (canale 2)

2)



Gli unici nodi per i quali cambia il numero dei nodi nel sotto albero in essi radicati sono B e C, quindi le istruzioni sono semplicemente:
B.size = C.size
(perchè ora B è alla radice)
C.size = B.right.size + C.right.size +1