Algoritmi e Strutture Dati
17 giugno 2022

> *Note*
> 1. La leggibilità è un prerequisito: parti difficili da leggere potranno essere ignorate.
> 2. Quando si presenta un algoritmo è fondamentale spiegare l'idea e motivarne la correttezza.
> 3. L'efficienza e l'aderenza alla traccia sono criteri di valutazione delle soluzioni proposte.
> 4. Si consegnano tutti i fogli, con nome, cognome, matricola e l'indicazione *bella copia* o *brutta copia*.

Domande

Domanda A (6 punti) Si determini la soluzione asintotica della seguente equazione di ricorrenza:

$$T(n) = 4\,T(n/3) + 2n^2 + 1$$

Soluzione: Rispetto allo schema generale si ha $a = 4$, $b = 3$, $f(n) = 2n^2 + 1$. Si osserva che $\log_b a = \log_3 4 < 2$ quindi $f(n) = \Omega(n^{\log_b a+\epsilon})$ (per $0 < \epsilon \leq 2 - \log_3 4$). In aggiunta vale la condizione di regolarità, ovvero $af(\frac{n}{b}) \leq cf(n)$ per qualche $c < 1$. Infatti $af(\frac{n}{b}) = 4f(\frac{n}{3}) = 4(\frac{2n^2}{9} + 1) \leq 2cn^2 \leq c(2n^2 + 1)$, asintoticamente, quando $4/9 < c < 1$. Più precisamente, per avere

$$4(\frac{2n^2}{9} + 1) \leq 2cn^2$$

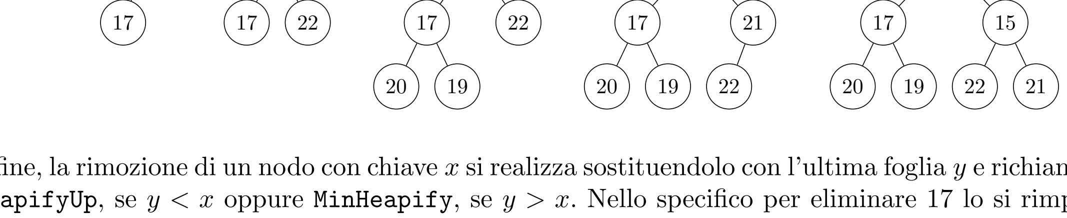
Occorre che $c \geq \frac{4}{9} + \frac{2}{n^2}$, che, per $n \geq 2$, si ottiene quando $c \geq \frac{4}{9} + \frac{2}{4} = \frac{15}{18} < 1$.

Domanda B (7 punti) Dare la definizione di min-heap. Data la sequenza di elementi `17, 15, 22, 20, 19, 21, 12,` si specifichi il min-heap ottenuto inserendo, a partire da uno heap vuoto, uno alla volta questi elementi nell'ordine indicato e infine rimuovendo 17. Si descriva sinteticamente come si procede per arrivare al risultato.

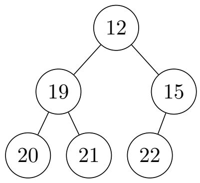
Soluzione: Un min-heap è un albero binario ordinato quasi-completo con la proprietà che per ogni nodo x, se x non è radice, il genitore di x ha chiave minore o uguale a quella di x, o equivalentemente ogni nodo x ha chiave minore o uguale a quella dei suoi successori.

Si procede inserendo nel min-heap i vari elementi con la procedura `HeapInsert` a partire da heap vuoto. La procedura inserisce l'elemento come prima foglia utile (ultimo elemento dell'array) e richiama la procedura `MinHeapifyUp` per ripristinare la proprietà di min-heap.

Gli inserimenti nell'ordine producono gli heap indicati in figura



Infine, la rimozione di un nodo con chiave x si realizza sostituendolo con l'ultima foglia y e richiamando `MinHeapifyUp`, se $y < x$ oppure `MinHeapify`, se $y > x$. Nello specifico per eliminare 17 lo si rimpiazza con l'ultima foglia 21 e si richiama `MinHeapify`, ottenendo il min-heap



che in forma di array è [12, 19, 15, 20, 21, 22].

Esercizi

Esercizio 1 (10 punti) Diciamo che un array senza ripetizioni $A[1, n]$ è *semi-ordinato* se esiste un indice k, con $1 \leq k < n$, tale che $A[k+1..n]A[1..k]$ sia ordinato, ovvero i sottoarray $A[k+1..n]$ e $A[1..k]$ sono ordinati e $A[n] < A[1]$. In questo caso l'indice k viene detto il centro dell'array. Ad esempio l'array che segue è semi-ordinato con centro $k = 4$.

1	2	3	4	5	6	7
4	9	12	18	-1	1	2

Scrivere una funzione `centre(A)` che dato un array A semi-ordinato ne restituisce il centro. Giustificare la correttezza dell'algoritmo e valutarne la complessità.

Soluzione: L'idea è quella di procedere con un algoritmo divide et impera. A tal fine osservazione fondamentale è la seguente: dato un sottoarry $A[p..r]$ semi-ordinato, se lo divido in due sottoarray $A[p..q]$ e $A[p..r]$, allora uno solo dei due è semi-ordinato (quello che contiene il centro), mentre l'altro è ordinato. Quando la dimensione del sottoarray $A[p..r]$ diventa 2, il centro è p.

Lo pseudo-codice è quindi il seguente:

```
centre(A)
    return centre-rec(A,1,A.length)

centre-rec(A,p,r)
    if r == p+1
        return p
    else
        q = (p+r)//2
        if (A[q]<A[p])
            return centre-rec(A,p,q)
        else
            return centre-rec(A,q,r)
```

La prova di correttezza procede per induzione sul numero di elementi n dell'array $A[p..r]$

- Se $n = 2$, ovvero $r = p + 1$, allora $A[p+1] < A[p]$ e chiaramente p è il centro.
- Se invece $n > 2$, l'array viene diviso in due parti, ovvero si definisce $q = \lfloor (p+r)/2 \rfloor$ e si considerano gli array $A[p..q]$ e $A[q..r]$. Detto k il centro, ci sono due possibilità.
  - ($q \leq k$) Questo accade se e solo se $A[p] > A[q]$, e correttamente in questo caso si ricorre sul sottoarray $A[q..r]$
  - ($q > k$) Questo accade se e solo se $A[p] < A[q]$, e correttamente in questo caso si ricorre sul sottoarray $A[p..q]$

Per quanto riguarda la complessità, la ricorrenza è del tipo $T(n) = T(n/2) + c$, dato che ad ogni chiamata si dimezza la dimensione dell'array e il costo della parte non ricorsiva è costante. Per il Master Theorem (con $a = 1$, $b = 2$, $f(n) = c = \Theta(n^{\log_b a}) = \Theta(n^0) = \Theta(1)$) si conclude $T(n) = \Theta(n^{\log_b a} \log n) = \Theta(\log n)$.

Esercizio 2 (9 punti) Sia n un intero positivo. Si consideri la seguente ricorrenza $M(i,j)$ definita su tutte le coppie (i,j) con $1 \le i \le j \le n$:

$$M(i,j) = \begin{cases} 2 & \text{se } i = j, \\ 3 & \text{se } j = i+1, \\ M(i+1,j-1) \cdot M(i+1,j) + M(i,j-1) & \text{se } j > i+1. \end{cases}$$

(a) Si scriva una coppia di algoritmi INIT_M(n) e REC_M(i,j) per il calcolo memoizzato di $M(1,n)$.

(b) Si calcoli il numero esatto $T(n)$ di moltiplicazioni tra interi eseguite per il calcolo di $M(1,n)$.

Soluzione:

(a)
```
INIT_M(n)
if n=1 then return 2
if n=2 then return 3
for i=1 to n-1 do
   M[i,i] = 2
   M[i,i+1] = 3
M[n,n] = 2
for i=1 to n-2 do
   for j=i+2 to n do
      M[i,j] = 0
return REC_M(1,n)


REC_M(i,j)
if M[i,j] = 0 then
   M[i,j] = REC_M(i+1,j-1) * REC_M(i+1,j) + REC_M(i,j-1)
return M[i,j]
```

(b)

$$T(n) = \sum_{i=1}^{n-2} \sum_{j=i+2}^{n} 1 = \sum_{i=1}^{n-2} n - i - 1 = \sum_{k=1}^{n-2} k = (n-2)(n-1)/2$$