Introduzione agli algoritmi
Proff. T. Calamoneri – S. Caminiti - E. Fachini
7 Maggio 2020

1. Si risolva per sostituzione la relazione di ricorrenza

$$f(n) = f(n/2) + \Theta(1) \qquad \text{ed} \qquad f(1) = \Theta(1)$$

mettendo in evidenza con chiarezza i passi seguiti.

Sol.

Esplicitando le costanti si ottiene

$$f(n) = f(n/2) + c \qquad \text{ed} \qquad f(1) = d$$

Sviluppando f(n), supponendo $n = 2^m$ per semplicità, si ottiene:

$f(n) = f(n/2) + c = f(n/2^2) + c + c = \ldots = f(n/2^i) + c + \ldots + c$ (i volte)

quindi $f(n) = f(n/2^m) + mc = d + mc$

Prevediamo dunque che f(n) = O(lg n).

Dimostrazione induttiva che f(n) = O(lg n). Si tratta di dimostrare che esistono due costanti positive k e n_0 tali che $f(n) \leq k \lg n$, per ogni $n \geq n_0$.

Passo induttivo.
Per ipotesi induttiva, preso un qualsiasi m < n la tesi è vera,
quindi $f(n/2) \leq k \lg(n/2)$ è vera.
Dunque $f(n) = f(n/2) + c \leq k \lg(n/2) + c$, bisogna trovare ora le due costanti positive k e n_0 che rendano vera la disuguaglianza:

$k \lg(n/2) + c \leq k \lg(n) \iff$

$k (\lg(n) - 1) + c \leq k \lg(n) \iff$

$k \lg(n) - k + c \leq k \lg(n) \iff$

$-k + c \leq 0 \iff k \leq c$

possiamo concludere che la disuguaglianza è vera per $k \geq c$ e $n_0 = 1$
Passo base.
Per n =1 si ha che $f(1) = d$ e $d \leq \lg 1 = 0$ è falso. Quindi consideriamo $f(2) = f(1) + c = d + c$ allora prendendo $k = c + d$ e $n_0 = 2$ otteniamo l'asserto.

2. Qual è l'algoritmo migliore da usare per riordinare un array contenente le prime *n* potenze di *3*, cioè una permutazione di $3^1, 3^2, \ldots, 3^n$ nel modo più efficiente possibile? Si motivi la risposta.

Sol. Il modo più semplice per ordinare le prime n potenze di 3, visto che non ci sono duplicati consiste semplicemente nello scrivere nell'array in input un'entrata alla volta le potenze $3^1,3^2,\ldots,3^{n-1}$. Il tempo di esecuzione è lineare nel numero di elementi, cioè in $\Theta(n)$.

3. Quali sono il minimo e il massimo numero di nodi in:

    a. un max-heap di altezza *h*

    b. un albero binario di ricerca di altezza *h*

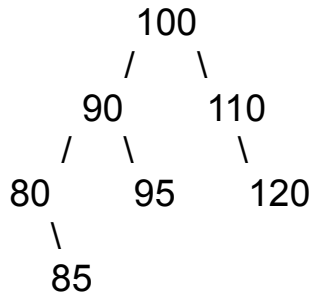
   Si motivi la risposta.

Sol.

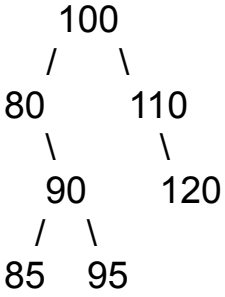
a) Un heap binario è un albero quasi completo ovvero un albero in cui tutti i livelli, tranne l'ultimo, sono completi e le foglie dell'ultimo livello sono tutte a sinistra.
Il numero massimo di nodi è dato dal numero di nodi dell'albero completo, $2^{h+1}-1$ mentre il minimo quando l'albero ha un solo nodo foglia al livello h, 2^h.

b) Il minimo numero di nodi in un ABR è dato dal numero dei nodi di un albero con un solo nodo su ogni livello, h+1 se l'albero ha altezza h. Mentre il massimo numero di nodi è $2^{h+1}-1$ che si ha quando l'albero è un albero completo di altezza h.

4. Dato l'albero binario disegnato sotto, si esegua una rotazione a destra intorno al nodo con chiave 90.



Sol. L'albero ottenuto è:



In una rotazione a destra intorno a un nodo n (90 nell'esempio), il suo figlio sinistro m (80 nell'esempio) prende il posto del padre, che ne diventa figlio destro, quindi il figlio destro di m (85 nell'esempio) deve essere reso figlio sinistro di n. Un eventuale figlio destro di n (95 nell'esempio) rimane figlio destro di n e un eventuale figlio sinistro di m (nell'esempio non c'è) rimane figlio sinistro di m. La rotazione così concepita applicata a un qualsiasi ABR conserva la proprietà di essere ABR.