ESAME 15 SETTEMBRE 2022

Esercizio 1 (10 punti):

Si supponga di avere un algoritmo speciale in grado di eseguire la fusione di due sottoarray ordinati di $n/2$ elementi ciascuno in $O(\sqrt{n})$ operazioni. Quanto sarebbe, in questo caso, il costo computazionale dell'algoritmo di Merge Sort?

a) Si imposti la relazione di ricorrenza che definisce il tempo di esecuzione giustificando dettagliatamente l'equazione ottenuta.

b) Si risolva la ricorrenza usando due metodi a scelta, dettagliando i passaggi del calcolo e giustificando ogni affermazione.

$T(n) = 2T\left(\frac{n}{2}\right) + \Theta(\sqrt{n})$ il merge sort per ogni fusione impiega $O(n)$ OPERAZIONI.

ITERATIVO

$$T(n) = 2\left[2T\left(\frac{n}{2^2}\right) + \Theta\left(\sqrt{\frac{n}{2}}\right)\right] + \Theta(\sqrt{n})$$

$$T(n) = 2^K T\left(\frac{n}{2^K}\right) + \sum_{i=0}^{K-1} 2^i \Theta\left(\sqrt{n/2^i}\right) \quad \text{fino a } K = \log_2(n)$$

$$T(n) = 2^{\log_2 n}\Theta(1) + \sum_{i=0}^{\log_2(n)} 2^i \Theta\left(\frac{\sqrt{n}}{\sqrt{2^i}}\right) = \Theta(n) + \sqrt{n}\sum \frac{2^i}{2^{\frac{i}{2}}}$$

$$\Theta(n) + \sqrt{n}\sum_{i=0}^{\log_2(n)} \left(\frac{2}{\sqrt{2}}\right)^i = \Theta(n) + \sqrt{n}\sum_{i=0}^{\log_2(n)} (\sqrt{2})^i$$

$$\Theta(n) + \sqrt{n}\left[\frac{\sqrt{2}^{\log_2(n)} - 1}{\sqrt{2} - 1}\right] = \Theta(n) + \sqrt{n}\left[\frac{\sqrt{n} - 1}{\sqrt{2} - 1}\right] = \Theta(n)$$

PRINCIPALE

$n^{\log_b a} = n$ $\quad f(n) = O(n) \rightarrow T(n) = \Theta(n^{\log_2 2}) = \Theta(n)$

Esercizio 2 (10 punti): Sia dato un array A contenente n interi distinti e ordinati in modo crescente. Progettare un algoritmo che, in tempo $O(\log n)$, individui la posizione più a sinistra nell'array per cui si ha $A[i] \neq i$, l'algoritmo restituisce -1 se una tale posizione non esiste.

Ad esempio, per $A = [0, 1, 2, 3, 4]$ l'algoritmo deve restituire -1, per $A = [0, 5, 6, 20, 30]$ la risposta deve essere 1 e per $A = [-3, 1, 2, 3, 6]$ la risposta deve essere 0.

Dell'algoritmo proposto:

a) si dia la descrizione a parole,

b) si scriva lo pseudocodice,

c) si giustifichi il costo computazionale.

CERCO TRAMITE RICERCA BINARIA IL PRIMO VALORE PER LA QUALE A[i] != i & A[i-1] = i-1;

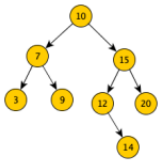
```
DEF ES2(A):
    IF(A[0]!=0): RETURN 0;                    Θ(1)
    IF(A[LEN(A)-1] == LEN(A)-1): RETURN -1;   Θ(1)
    i = 0;                                    Θ(1)
    J = LEN(A)-1;                             Θ(1)
    WHILE(i != J):                            AL PIU' log(n) VOLTE
        MID = (i+J)//2;                       Θ(1)
        IF(A[MID] == MID):                    Θ(1)
            i = MID;                          Θ(1)
        ELSE IF(A[MID-1] != MID-1):           Θ(1)
            J = MID;                          Θ(1)
        ELSE:                                 Θ(1)
            RETURN MID;                       Θ(1)
```

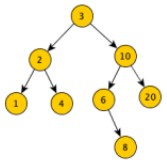
COSTO UGUALE = $4\Theta(1) + \log(n)[7\Theta(1)] = \Theta(\log(n))$

AD OGNI ITERAZIONE DIMEZZO LO SPAZIO DI CONTROLLO DELL'ARRAY, QUINDI AL PIU' $\log(n)$ VOLTE;

Esercizio 3 (10 punti): Progettare un algoritmo che, dato il puntatore alla radice di un albero binario T avente per chiavi degli interi, verifica se l'albero è un albero binario di ricerca.

Ad esempio, l'algoritmo per l'albero sulla sinistra deve restituire $True$ mentre per l'albero sulla destra deve restituire $False$ (infatti nel sottoalbero di sinistra del nodo con chiave 3 è presente un nodo con chiave 4)





Il costo computazionale dell'algoritmo proposto deve essere $\Theta(n)$ dove n è il numero di nodi dell'albero.
Dell'algoritmo proposto

a) si dia la descrizione a parole,
b) si scriva lo pseudocodice,
c) si giustifichi il costo computazionale.

TRAMITE UNA VISITA IN-ORDINE SI RIEMPIE UN ARRAY, SE ESSO NON E' ORDINATO, NON E' UN ABR.

```
DEF RIEMPI(R):
    IF(!R):
        RETURN
    RIEMPI(R->LEFT);
    A.APPEND(R->KEY);
    RIEMPI(R->RIGHT);
```

```
DEF CHECKABR(A):
    i=0;
    WHILE(i < LEN(A)-1):
        IF(A[i]>A[i+1]):
            RETURN FALSE;
        i++;
    RETURN TRUE;
```

FAI ATTENZIONE ALLE VISITE!