ESAME 27 GIUGNO 2022

Esercizio 1 (10 punti): Si consideri la seguente funzione:

```
def Exam(A, n):
    b=1   Θ(1)
    if n <= 2:  return b  Θ(1)  CASO BASE
    i = 1  Θ(1)
    while i ≤ 8:  8 VOLTE
        b=b*Exam(A, n//2) } T(A, n/2)
        i+=1
    for i in range(n-1):  n VOLTE
        A[i]+=A[i+1]    Θ(1)
    return b  Θ(1)
```

che viene richiamata la prima volta così: Exam(A, $len(A)$).

a) Si imposti la relazione di ricorrenza che ne 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.

$$T(A,n) = 8T\left(A, \frac{n}{2}\right) + \Theta(n)$$

$$T(A,1) = \Theta(1)$$

METODO PRINCIPALE

$$n^{\log_b a} = n^{\log_2 8} = n^3$$

$$f(n) = \Theta(n)$$

$$f(n) = O(n^3) \Rightarrow T(n) = \Theta(n^3)$$

METODO ITERATIVO

$$T(n) = 8^K T\left(A, \frac{n}{2^K}\right) + \sum_{i=0}^{K-1} 8^i \Theta\left(\frac{n}{2^i}\right) \quad \text{fino a } K = \log(n)$$

$$T(n) = 8^{\log_2(n)} \cdot \Theta(1) + \sum_{i=0}^{\log(n)-1} \Theta\left(\left(\frac{8}{2}\right)^i n\right)$$

$$* \; 8^{\log_2(n)} = n^{\log_2(8)} = n^3$$

$$T(n) = n^3 \Theta(1) + \Theta(n) \sum_{i=0}^{\log(n)-1} (4^i) = \Theta(n^3) + \Theta(n)\left[\frac{4^{\log(n)} - 1}{3}\right]$$

$$T(n) = \Theta(n^3) + \Theta(n)\left[\frac{1}{3}(n^2 - 1)\right] = \Theta(n^3)$$

Esercizio 2 (10 punti): Un array A ordinato di $n > 1$ interi distinti ha subito una rotazione di k posizioni verso sinistra, $1 \leq k < n$. Ad esempio, per $A = [5, 7, 9, 2, 3]$ il valore di k è 2 mentre per $A = [9, 2, 3, 5, 7]$ è 4.
Progettare un'algoritmo che, dato l'array A, in tempo $O(\log n)$ restituisca il valore di k.

Dell'algoritmo proposto:

a) si dia la descrizione a parole;

b) si scriva lo pseudocodice;

c) si giustifichi il costo computazionale.

COME PRIMA COSA, CERCO TRAMITE RICERCA BINARIA L'ELEMENTO DI INDICE i PER CUI $A[i-1] > A[i]$, LA ROTAZIONE FA SÌ CHE L'ARRAY SIA IN UN LATO CRESCENTE, E NELL'ALTRO DECRESCENTE, QUINDI, CONTROLLANDO L'ELEMENTO SUCCESSIVO AD i, SAPRÒ SE SIAMO NELLA PARTE CRESCENTE (VADO A DESTRA) O IN QUELLA DECRESCENTE (VADO A SINISTRA. SI NOTI CHE ESSENDO LA ROTAZIONE COMPRESA FRA 1 ED n, LA PARTE CRESCENTE SARÀ SEMPRE A SINISTRA. SE L'ARRAY È ORDINATO, $K = n$. UNA VOLTA TROVATO L'ELEMENTO, LA ROTAZIONE SARÀ LEN(A) - i.

IL PICCO DI INDICE i È $i > i-1$ E $i > i+1$

```
DEF ES2(A, i=0, J=LEN(A)-1):                                   — Θ(1)
    IF(i==J):                                                  — Θ(1)
        RETURN LEN(A); #ARRAY GIÀ ORDINATO                     — Θ(1)
    MID = (i+J)//2; #DIVISIONE INTERA                          — Θ(1)
    IF((A[MID-1] > A[MID]) AND (A[MID-1] > A[MID-2])):         — Θ(1)
        RETURN LEN(A)-MID;                                     — Θ(1)
    IF(A[MID] > A[MID-1]): #SIAMO NELLA CRESCENTE              — Θ(1)
        RETURN ES2(A, MID, J); #VADO A DESTRA                  — T(n/2)
    RETURN ES2(A, i, MID-1); #VADO A SINISTRA                  — T(n/2)
```

$$T(n) = T\left(\frac{n}{2}\right) + \Theta(1)$$

ESSENDO RICERCA BINARIA È $\Theta(\log(n))$

METODO PRINCIPALE

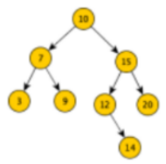
$f(n) = \Theta(1)$

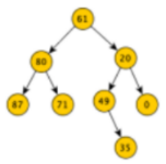
$n^{\log_b a} = \Theta(1)$ SECONDO CASO $T(n) = \Theta(1 \cdot \log(n))$

$n^{\log_b a} = n^0 = 1$

Esercizio 3 (10 punti): Progettare un algoritmo che, dato il puntatore alla radice di un **albero binario di ricerca** T, modifica il valore di ciascun nodo di T in modo che il nuovo valore del nodo risulti la somma di tutte le chiavi(che, in quanto tali, sono tutte distinte) che in T avevano un valore maggiore della sua chiave originaria.

Ad esempio l'albero sulla destra è il risultato dell'applicazione dell'algoritmo sull'albero binario di ricerca T riportato a sinistra.





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.

FARO' UNA VISITA IN PRE ORDER DEL TIPO FIGLIO DESTRO - NODO - FIGLIO SINISTRO, USANDO UNA VARIABILE GLOBALE CHE PRENDE TRACCIA DEL VALORE DA PROPAGARE.

```
DEF ES3(NODO):
    IF(NODO->RIGHT):
        ES3(NODO->RIGHT);
    TMP = NODO->KEY
    NODO->KEY = SOMMA
    SOMMA += TMP
    IF(NODO->LEFT):
        ES3(NODO->LEFT);
```

$$\begin{cases} T(n) = T(K) + T(n-K-1) + \Theta(1) \\ T(1) = \Theta(1) \end{cases}$$

K = NODI SOTTO ALB. SX

DIMOSTRO PER SOSTITUZIONE CHE $T(n) = \Theta(n)$

$an \leq T(n) \leq bn$ $\quad \begin{cases} T(n) = T(K) + T(n-K-1) + c \\ T(1) = d \end{cases}$

CASO BASE $\quad$ IPOT. INDUTTIVA: $\forall m < n \rightarrow am \leq T(m) \leq bm$

$a \leq d \leq b$

PASSO IND.

$an \leq T(K) + T(n-K-1) + c \leq bn$ i. IND. $\rightarrow$

$an \leq aK + a[n-K-1] + c \rightarrow \cancel{an} \leq \cancel{aK} + \cancel{an} - \cancel{aK} - a + c \rightarrow$

$bn \geq bK + b[n-K-1] + c \rightarrow \cancel{bn} \geq \cancel{bK} + \cancel{bn} - \cancel{bK} - b + c \rightarrow$

$a \leq c \leq b$