ESAME 13 LUGLIO 2021

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

```
funzione Exam(n):
    if n <= 2:  return 2 * n;
    b ← n/2;
    tot ← n * n;
    for i = 1 to n:
        for j = 1 to i:
            tot ← tot + i − j;
    for i = 1 to 4:
        for j = 1 to 4:
            if i = j:  tot ← tot + Exam(b)
            else:  tot ← tot + i − j;
    return tot.
```

CASO BASE $\Theta(1)$

$b = \frac{n}{2}$

$TOT = n^2$

$\sum_{i=1}^{n} \sum_{j=1}^{i} \Theta(1) = \Theta(n^2)$

$T(n) = 4T\left(\frac{n}{2}\right) + \Theta(n^2)$

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

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

b) Si risolva l'equazione usando il metodo iterativo, commentando opportunamente i passaggi del calcolo;

c) Si risolva l'equazione usando il metodo principale, specificando quale caso del teorema si applica e perché oppure per quale motivo non si può applicare il teorema.

METODO ITERATIVO

$$T(n) = 4\left[4T\left(\frac{n}{2^2}\right) + \Theta\left(\left[\frac{n}{2}\right]^2\right)\right] + \Theta(n^2) = 4^K T\left(\frac{n}{2^K}\right) + \sum_{i=0}^{K-1} 4^i \Theta\left(\left[\frac{n}{2^i}\right]^2\right)$$

fino a $T(1) \rightarrow \frac{n}{2^K} = 1 \rightarrow n = 2^K \rightarrow K = \log_2(n)$

$$4^{\log_2(n)} \cdot \Theta(1) + \sum_{i=0}^{\log_2(n)-1} 4^i \Theta\left(\frac{n^2}{4^i}\right) = \Theta(n^2) + \Theta(n^2) \sum_{i=0}^{\log_2(n)-1} 1$$

$$= \Theta(n^2 \log_2(n))$$

METODO PRINCIPALE

$n^{\log_b a} = n^{\log_2 4} = n^2$

$f(n) = \Theta(n^2)$

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

SECONDO CASO DEL TEOREMA.

Esercizio 2 (10 punti):

Progettare un algoritmo che, dato un array A di n interi distinti i cui elementi sono all'inizio in ordine crescente e da un certo punto in poi in ordine decrescente, restituisce in tempo $O(\log n)$ il massimo intero presente nell'array. Ad esempio: per $A = [8, 10, 20, 80, 100, 200, 400, 500, 3, 2, 1]$ l'algoritmo deve restituire il valore 500.

Dell'algoritmo proposto

a) si dia la descrizione a parole;

b) si scriva lo pseudocodice;

c) si calcoli il costo computazionale.

BISOGNA TROVARE TRAMITE RICERCA BINARIA L'ELEMENTO CHE HA SIA SUCCESSIVO CHE PRECEDENTE, MINORI A SE STESSO. QUANDO SIAMO NELLA PARTE CRESCENTE VADO A DESTRA, ALTRIMENTI SINISTRA.

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

ESSENDO RICERCA BINARIA, AVRA' COSTO $O(\log(n))$

Esercizio 3 (10 punti):

Dato un albero binario T, radicato e con n nodi, definiamo un nodo u di T *equilibrato* se il sottoalbero sinistro di u e il sottoalbero destro di u hanno entrambi lo stesso numero di nodi.

Progettare un algoritmo che, dato il puntatore r alla radice di un albero binario memorizzato tramite record e puntatori, restituisca in tempo $O(n)$ il numero dei suoi nodi equilibrati.

Dell'algoritmo proposto

a) si dia la descrizione a parole;

b) si scriva lo pseudocodice;

c) si motivi il costo computazionale.

Qual è il numero minimo e qual è il numero massimo di nodi equilibrati che l'albero T può avere? Motivare la risposta.

```
IF !R: RETURN 0
IF !R->RIGHT AND !R->LEFT :
    U+=1
    RETURN 1;
IF R->RIGHT :
    FIGLIDX = 1+ES(R->RIGHT)
IF R->LEFT:
    FIGLISX = 1+ES(R->LEFT)
IF FIGLIDX == FIGLISX :
    U+=1;
RETURN FIGLIDX+FIGLISX;
```

USERO' UNA VARIABILE GLOBALE "EQ" PER MEMORIZZARE I NODI EQUILIBRATI.
FARO' UNA VISITA IN POST-ORDINE, CALCOLANDO PER OGNI NODO, IL NUMERO DI FIGLI DESTRI E SINISTRI, RITORNANDO AL PADRE LA SOMMA DI ESSI. LE FOGLIE SONO EQUILIBRATE.

```
DEF ES3(R):
    GLOBAL EQ;                          Θ(1)
    FIGLISX=FIGLIDX=0;                  Θ(1)
    IF(R->LEFT):                        Θ(1)
        FIGLISX = 1+ES3(R->LEFT);       T(k)
    IF(R->RIGHT):                       Θ(1)
        FIGLIDX = 1+ES3(R->RIGHT);      T(n-k-1)
    IF(FIGLIDX==FIGLISX):               Θ(1)
        EQ+=1;                          Θ(1)
    RETURN FIGLIDX+FIGLISX;             Θ(1)
```

```
DEF MAIN(R):
    GLOBAL EQ;
    ES3(R);
    RETURN EQ;
```

$T(n) = T(k)+T(n-k-1)+\Theta(1)$ CONTROLLO OGNI NODO IN $\Theta(1)$, IL COSTO È $\Theta(n)$.

IL NUMERO MINIMO DI NODI EQUILIBRATI È 1, DATO CHE, OGNI FOGLIA È EQUILIBRATA, E SE L'ALBERO FOSSE UN SOLO NODO, SAREBBE EQUILIBRATO, SE AVESSE UN SOLO FIGLIO FOGLIA, ESSA SAREBBE EQUILIBRATA, IL NUMERO MASSIMO È, TUTTI I NODI, UN ALBERO BINARIO COMPLETO, HA TUTTI I NODI EQUILIBRATI.