ESAME 8 SETTEMBRE 2021

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

```
funzione Exam(n):
    tot ← n; Θ(1)                                   TOT = n
    if n <= 4: return tot; Θ(1)   CASO BASE         b = n/4
    b ← n//4;
    tot ← tot+Exam(b); T(n/4)
    j ← 1; Θ(1)
    while j * j <= n do: √n VOLTE
        tot ← tot + j;  } Θ(1)
        j ← j + 1;      }
    return tot+Exam(b). T(n/4)
```

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

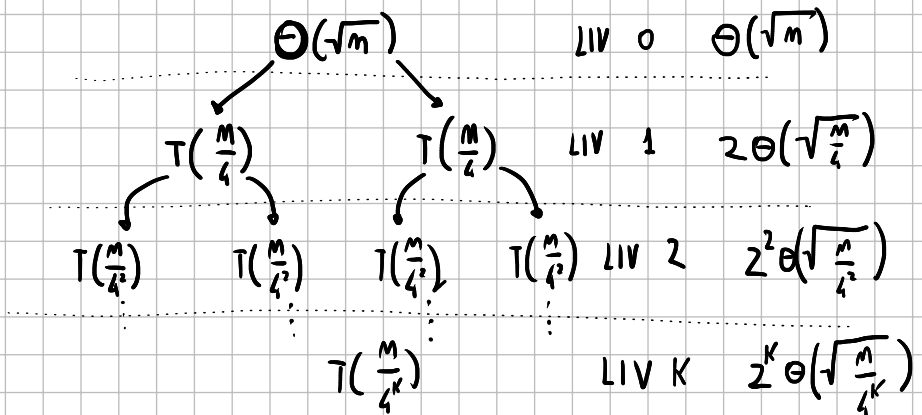
b) Si risolva l'equazione usando il **metodo dell'albero**, dettagliando i passaggi del calcolo e giustificando ogni affermazione.

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

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

AD OGNI PASSO RICORSIVO LA FUNZIONE SI RICHIAMA 2 VOLTE SU $b = \frac{n}{4}$, INOLTRE, ESEGUE UN CICLO WHILE FINCHE IL CONTATORE ALLA SECONDA NON È MAGGIORE DI n, J.J FINO AD $n \to J$ FINO A $\sqrt{n}$.

METODO DELL'ALBERO



la sua altezza sarà $\log n$

$$\sum_{i=0}^{\log(n)} 2^i \Theta\left(\sqrt{\frac{n}{4^i}}\right) = \sum_{i=0}^{\log(n)} 2^i \Theta\left(\frac{\sqrt{n}}{2^i}\right) = \sqrt{n} \sum_{i=0}^{\log(n)} \Theta(1) = \Theta(\sqrt{n}\log(n))$$

Esercizio 2 (10 punti):

Progettare un algoritmo che, dati tre array A, B e C **ordinati** e contenenti ciascuno n interi **distinti**, stampi in tempo $O(n)$ gli interi che compaiono nell'intersezione dei tre array. L'algoritmo proposto deve utilizzare spazio di lavoro $\Theta(1)$.
Ad esempio: per $A = [1, 2, 3, 4, 5, 6]$, $B = [1, 4, 5, 6, 8, 9]$ e $C = [2, 4, 6, 7, 8, 9]$ l'algoritmo deve stampare gli elementi 4 e 6.

Dell'algoritmo proposto:

a) si dia la descrizione a parole,

b) si scriva lo pseudocodice,

c) si giustifichi formalmente il costo computazionale.

d) si dia un'idea di quello che accadrebbe al costo computazionale se si volesse generalizzarlo a $\Theta(n)$ array.

USERO' 3 INDICI PER CAMMINARE SUI 3 ARRAY, INCREMENTANDO SEMPRE QUELLO COL VALORE MINORE, E STAMPANDO QUANDO I 3 VALORI SONO UGUALI.

```
DEF ESZ(A,B,C):                                                    Θ(1)
    INT i,j,k = 0;
    WHILE ( NOT ( i=LEN(A) AND j=LEN(B) AND K=LEN(C) )):           AL PIU' 3n VOLTE
        IF(A[i]==B[j]==C[k]):                                      Θ(1)
            PRINT(A[i]);                                           //
            IF ( i < LEN(A)-1 ): i++;                              //
            IF ( j < LEN(B)-1 ): j++;                              //
            IF ( k < LEN(C)-1 ): k++;                              //
        ELSE:                                                      //
            LESS = MIN(A[i],B[j], C[k]);                           //
            IF (A[i] == LESS AND i < LEN(A)-1):                    //
                i++;                                               //
            IF (B[j] = LESS AND j < LEN(B)-1):                     //
                j++;                                               //
            IF (C[k] == LESS AND K < LEN(C)-1):                    //
                K++;                                               //
```

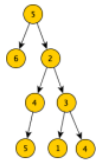
COSTO:

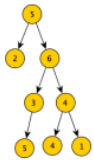
$$\Theta(1) + 3n\left[15\,\Theta(1)\right] = \Theta(n)$$

DOVREMMO SCORRERE AL PIU' OGNI ELEMENTO DEI 3 ARRAY, QUINDI, SE n È IL NUMERO DI ELEMENTI DI OGNI ARRAY, DOVREMMO ESEGUIRE $3n$ ITERAZIONI. SE VOLESSIMO GENERALIZZARLO AD UN NUMERO DI n ARRAY, E' CHIARO CHE DOVENDO SCORRERE OGNI ELEMENTO DEGLI n ARRAY, IL WHILE ESEGUIRA' n^2 CICLI, CONSIDERANDO POI CHE AD OGNI CICLO, IL VALORE MASSIMO SI TROVA IN TEMPO $\Theta(n)$, IL COSTO TOTALE SARA' $\Theta(n^3)$.

Esercizio 3 (10 punti):

Si consideri un albero binario radicato T, i cui nodi hanno un campo valore contenente un intero. Bisogna modificare l'albero in modo che i nodi fratelli scambino tra loro il valore. Si consideri ad esempio l'albero T in figura a sinistra, a destra viene riportato il risultato della modifica di T.





Progettare un algoritmo che, dato il puntatore r alla radice di T memorizzato tramite record e puntatori, effettui l'operazione di modifica in tempo $O(n)$ dove n è il numero di nodi presenti nell'albero. Ogni nodo dell'albero è memorizzato in un record contenente il campo *val* con il valore del nodo e i campi *left* e *right* con i puntatori ai figli di sinistra e destra, rispettivamente. Dell'algoritmo proposto:

a) si dia la descrizione a parole,

b) si scriva lo pseudocodice,

c) si giustifichi formalmente il costo computazionale.

BANALMENTE, AD OGNI PASSO RICORSIVO, SE IL NODO HA 2 FIGLI, SCAMBIO LE LORO CHIAVI, E POI RICHIAMO SU ESSE LA FUNZIONE.

```
DEF ES3(R):
    IF(!R): RETURN;                          Θ(1)
    IF(R->LEFT AND R->RIGHT):                Θ(1)
        TMP = R->LEFT->KEY);                 Θ(1)
        R->LEFT->KEY = R->RIGHT->KEY;        Θ(1)
        R->RIGHT->KEY = TMP;                 Θ(1)
    ES3(R->LEFT);                            T(K)
    ES3(R->RIGHT);                           T(N-K-1)
```

K = NODI SOTTO ALBERO SINISTRO

$$T(n) = T(k) + T(n-k-1) + \Theta(1) \qquad T(1) = \Theta(1)$$

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

$\Theta(n)$

$$\begin{cases} T(n) = T(k) + T(k-n-1) + c \\ T(1) = d \end{cases}$$

$T(n) \le an$

CASO BASE

$d \le a$

1. IND.

$\forall m < n \quad T(m) \le am$

P. IND.

$ak + a[n-k-1] + c \le an$

$c \le a$

$\Omega(n)$

$$\begin{cases} T(n) = T(k) + T(k-n-1) + c \\ T(1) = d \end{cases}$$

$T(n) \ge bn$

CASO BASE

$d \ge b$

1. IND.

$\forall m < n \quad T(m) \ge bm$

P. IND.

$bk + b[n-k-1] + c \ge bn$

$c \ge b$

$b \le c \le a$ ✓