ESAME 20 MARZO 2023

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

```
def Es1(n):
    if n < 10:
        return n          CASO BASE
    s = Es1(n//2) * Es1(n//2)   ← 2T(n/2)
    for j in range(n):   n VOLTE
        k = n
        while k > 1:   log(n)
            s = s + 1
            k = k//2
    return s + Es1(n//2)   ← T(n/2)
```

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

b) Qualora sia possibile, risolvere la ricorrenza utilizzando il teorema principale, dettagliando il caso del teorema ed i passaggi logici. Se il teorema principale non è applicabile spiegarne il motivo.

$$T(n) = 3T\left(\frac{n}{2}\right) + \Theta(n\log(n))$$

$$T(9) = \Theta(1)$$

LA FUNZIONE RICHIAMA SE STESSA 3 VOLTE CON LA METÀ DELL'INPUT, DOPO DI CHE ESEGUE UN CICLO n VOLTE, AL SUO INTERNO UN ALTRO CICLO $\log(K)$ VOLTE, MA VISTO CHE $K = n$, OGNI PASSO RICORSIVO COSTERÀ $\Theta(n\log(n))$.

metodo principale

$n^{\log_b a} = n^{\log_2 3} \quad n \simeq n^{1.5}$

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

$f(n) = O(n^{\log_2 3 - \varepsilon})$ QUINDI $T(n) = \Theta(n^{\log_2 3})$

es. $\varepsilon = 0.1$

Esercizio 2 (10 punti):
Dato un array A di n interi non negativi distinti, si vuole determinare se esistono almeno tre numeri consecutivi di valore inferiore a 100.
Ad esempio, se $A = [101, 5, 9, 31, 33, 10, 100, 4, 8, 32, 500, 11, 99]$, gli elementi 8, 9 e 10 così come gli elementi 31, 32 e 33 rispettano la proprietà mentre 99, 100 e 101 no.
Progettare un algoritmo che, dato A, in tempo $\Theta(n)$ restituisce il valore dell'elemento centrale della terna se questa è presente, -1 altrimenti. Se esistono più terne allora bisogna restituire l'elemento centrale di valore massimo (nell'esempio sopra, l'algoritmo dovrebbe restituire 32; se nell'array ci fossero 4 numeri consecutivi andrebbe restituito il terzo, ad esempio se l'array contenesse soltanto 5, 6, 7 e 8, andrebbe restituito il 7).
Dell'algoritmo proposto:

a) si scriva lo pseudocodice opportunamente commentato;

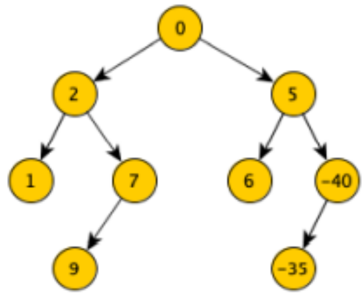
b) si giustifichi il costo computazionale.

SCORRO L'ARRAY, SENZA CONTROLLARE PER OVVI MOTIVI IL PRIMO E L'ULTIMO ELEMENTO. OGNI QUAL VOLTA CI SONO 3 ELEMENTI CONSECUTIVI, E QUELLO CENTRALE È MINORE DI 99, SE TALE ELEMENTO È MAGGIORE DELLO SCORSO ELEMENTO TROVATO, SARÀ QUELLO DA RITORNARE.

```
DEF ES2(A):
    i = 0;                                   Θ(1)
    RESULT = -1;                             Θ(1)
    WHILE (i < LEN(A)):                      n VOLTE
        IF ((i != 0) AND (i != LEN(A-1))):   Θ(1)
            IF A[i] < 99:                    Θ(1)
                IF ((A[i]-1 == A[i-1]) AND (A[i+1] == A[i]+1)):   Θ(1)
                    IF (A[i] > RESULT):      Θ(1)
                        RESULT = A[i];       Θ(1)
    RETURN RESULT;                           Θ(1)
```

$$T(n) = 2\Theta(1) + n[5\Theta(1)] = \Theta(1) + \Theta(n) = \Theta(n)$$

Esercizio 3 (10 punti): Un nodo di un albero a valori interi si dice *nodo valido* se la somma dei valori dei suoi antenati è uguale al valore del nodo. Ad esempio nell'albero binario in figura, risultano validi 3 nodi (quello con valore 0, quello con valore 9 e quello con valore -35).
Dato il puntatore r al nodo radice di un albero binario non vuoto, progettare un algoritmo *ricorsivo* che in tempo $\Theta(n)$ calcoli il numero di nodi validi dell'albero. L'albero è memorizzato tramite puntatori e record a tre campi: il campo key contenente il valore ed i campi $left$ e $right$ con i puntatori al figlio sinistro e al figlio destro, rispettivamente (questi puntatori valgono $None$ in mancanza del figlio).



Dell'algoritmo proposto:

a) si scriva lo pseudocodice opportunamente commentato;

b) si giustifichi il costo computazionale.

VISITERÒ OGNI NODO, PASSANDO AD OGNI PASSO RICORSIVO UN VALORE CHE SI SOMMERÀ ALLA CHIAVE DEL NODO, SE TALE VALORE SARÀ UGUALE ALLA CHIAVE DEL NODO, QUELLO SARÀ UN NODO VALIDO.

```
DEF ES3(R, VALIDITA'=0):   DI DEFAULT, VALIDITA' = 0.
    n = 0;
    IF R->KEY == VALIDITA':
        n += 1;
        VALIDITA' += R->KEY;
    IF R->RIGHT:
        n += ES3(R->RIGHT, VALIDITA');
    IF R->LEFT:
        n += ES3(R->LEFT, VALIDITA');
    RETURN n;
```

OGNI VISITA HA TEMPO COSTANTE. DEFINIAMO K I FIGLI SINISTRI:

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

RISOLVO CON METODO DI SOSTITUZIONE

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

IPOTIZZO $T(n) = \Theta(n) \rightarrow an \leq T(n) \leq bn$

CASO BASE:

$$a \leq d \leq b$$

IPOTESI INDUTTIVA

$$\forall m < n \to T(m) = \Theta(m) \Rightarrow an \le T(m) \le bn$$

PASSO INDUTTIVO

$$T(k) + T(n-k-1) + c \ge an \qquad T(k) + T(n-k-1) + c \le bn$$

essendo $k < n$, SI HA L'IPOTESI INDUTTIVA

$$ak + (n-k-1)a + c \ge an \qquad bk + (n-k-1)b + c \le bn$$

$$\cancel{ak} + an - \cancel{ak} - a + c \ge an \qquad \cancel{bk} + bn - \cancel{bk} - b + c \le bn$$

$$an - a + c \ge an \qquad bn - b + c \le bn$$

$$c \ge a \qquad c \le b$$

VERIFICATO