ESAME 31 GENNAIO 2021

Esercizio 1 (8 punti):

Si consideri la seguente funzione:

TOT = n

```
funzione Exam(n):
    tot ← n; Θ(1)
    if n <= 1:  return tot; CASO BASE   Θ(1)
    j ← 512;
    while j >= 2 do: log2 512 VOLTE - 9 VOLTE
        k ← 0; Θ(1)
        while 3 * k <= n do:  k ← k + 1 {n/3 VOLTE}
        tot ← tot + Exam(k); T(n/3)
        j ← j/2; Θ(1)
    return tot Θ(1)
```

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 il **metodo principale** (o un altro metodo, ricordando che $\sum_{i=1}^{k} 3^i = \Theta(3^k)$) dettagliando i passaggi del calcolo e giustificando ogni affermazione.

$$T(n) = 9T\left(\frac{n}{3}\right) + \Theta(n)$$

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

TEOREMA PRINCIPALE

$$n^{\log_b a} = n^{\log_3 9} = n^2$$

$$f(n) = n$$

$$f(n) = O(n^{2-\varepsilon}) \rightarrow T(n) = n^2$$

METODO ITERATIVO

$$T(n) = 9T\left(\frac{n}{3}\right) + \Theta(n) = 9\left[9T\left(\frac{n}{3^2}\right) + \Theta\left(\frac{n}{3}\right)\right] + \Theta(n) =$$

$$= 9^K T\left(\frac{n}{3^K}\right) + \sum_{i=0}^{K-1} 9^i \Theta\left(\frac{n}{3^i}\right) \quad \text{fino a } n = 3^K \rightarrow K = \log_3 n$$

$$= 9^{\log_3 n}\Theta(1) + \sum_{i=0}^{\log_3 n - 1} \Theta(3^i \cdot n) = \Theta(n^2) + \Theta(n)\sum_{i=0}^{\log(n)-1} 3^i = \Theta(n^2)$$

Esercizio 2 (12 punti):

Sia A un array di dimensione n e B un array **ordinato** di dimensione m, contenenti entrambi numeri interi. Si vuole trovare il numero di interi di A che non sono presenti in B. Progettare un algoritmo ricorsivo che risolva il problema con un costo computazionale asintotico strettamente inferiore a $\Theta(nm)$.

Ad esempio: per $A = [8, 1, 2, 12, 10, 11, 20, 2]$ e $B = [3, 3, 4, 8, 10, 10, 13, 20, 21, 22]$ l'algoritmo deve restituire 5 (i numeri in A e non in B sono infatti $1, 2, 2, 11, 12$.

Dell'algoritmo proposto

a) si dia la descrizione a parole,

b) si scriva lo pseudocodice,

c) si giustifichi il costo computazionale.

8 4 6 3 1 10

1 2 4 8 9 10

COME PRIMA COSA ORDINO A CON UN MERGE SORT IN TEMPO $\Theta(n \log(n))$, DOPO DI CHE, CAMMINERO' CON 2 INDICI SU A E SU B, PER CONTROLLARE QUALI VALORI NON SONO PRESENTI.

```
DEF ES2(A,B):
    MERGESORT(A);                    Θ(n log(n))
    CNT=i=J=0;
    WHILE(i<LEN(A) AND J<LEN(B)):    AL PIU' MAX(m,n) VOLTE
        WHILE(A[i]<B[J]):
            i+=1;
            CNT+=1;
        IF(A[i]>B[J]):
            J+=1;
        ELSE:
            J+=1;
            i+=1;
    WHILE(i<LEN(A)):
        i+=1;
        CNT+=1;
    RETURN CNT;
```

$$T(n) = \Theta(n \log(n)) + \Theta(MAX(m,n)) \leq \Theta(m \cdot n)$$

Esercizio 3 (10 punti):

Si consideri una lista L, in cui ogni elemento è un record a due campi, il campo `val` contenente un intero ed il campo `next` con il puntatore al nodo seguente (`next` vale *None* per l'ultimo record della lista.

Bisogna contare i record della lista contenenti numeri pari. Si consideri ad esempio la lista L, per questa lista bisogna la risposta è 6

Progettare un **algoritmo ricorsivo** che, dato il puntatore r alla testa della lista effettui l'operazione di conteggio in tempo $\Theta(n)$ dove n è il numero di elementi presenti nella lista.
Dell'algoritmo proposto

a) si dia la descrizione a parole,

b) si scriva lo pseudocodice,

c) si giustifichi il costo computazionale risolvendo la ricorsione che viene fuori dall'algoritmo utilizzando uno dei metodi di soluzione visti a lezione.

```
DEF ES3(R, PARI):
    IF !R :                    Θ(1)
        RETURN PARI;           Θ(1)
    IF R->VAL % 2 == 0 :       Θ(1)
        PARI += 1;             Θ(1)
    RETURN (R->NEXT, PARI);    T(n-1)
```

SCORRO LA LISTA AGGIORNANDO IL CONTATORE DEI PARI E RITORNANDO QUANDO LA LISTA FINISCE.

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

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

METODO IT.

$$T(n) = T(n-1) + \Theta(1) = T(n-2) + \Theta(1) + \Theta(1) = T(n-k) + k\Theta(1)$$

FINO A $k = n-1 \rightarrow T(n-n-1) + (n-1)\Theta(1) = T(1) + \Theta(n) =$

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