ESAME 31 GENNAIO 2023

Esercizio 1 (10 punti):

Si consideri il seguente algoritmo ricorsivo che, dato un array A di dimensione n, verifica se esistono due indici diversi i e j compresi nell'intervallo $[0, n-1]$ tali che $A[i] = j$ e $A[j] = i$:

```
def IndiciValori(A, sx, dx):
    if (sx >= dx):  return False  Θ(1)
    else:
        trovato = False  Θ(1)
        centro = (sx + dx)//2  Θ(1)
        for i in range(sx, centro + 1): M/2 VOLTE
            for j in range(centro + 1, dx + 1): M/2 VOLTE
                if (A[i] == j) and (A[j] == i):  trovato = True  Θ(1)
    trovato1 = IndiciValori(A, sx, centro)  T(M/2)
    trovato2 = IndiciValori(A, centro + 1, dx)  T(M/2)
    return trovato or trovato1 or trovato2  Θ(1)
```

a) Si imposti la relazione di ricorrenza che 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.

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

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

METODO PRINCIPALE

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

$n^{\log_b a} = n$

ed $\exists\, c$ t.c. $f\left(\frac{n}{b}\right) \le c f(n)$ QUINDI $T(n) = \Theta(f(n))$

$\left[\frac{n}{2}\right]^2 \le c \cdot n^2 \Rightarrow c = \frac{1}{2}$ $\qquad T(n) = \Theta(n^2)$

METODO ITERATIVO

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

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

$$= 2^{\log(n)} \cdot \Theta(1) + \sum_{i=0}^{\log(n)} \left[\Theta(n^2)\,\frac{1}{2^i}\right] = n\Theta(1) + \Theta(n^2) \sum_{i=0}^{\log(n)} \left(\frac{1}{2}\right)^i =$$

= ESSENDO CHE $\sum_{i=0}^{\infty} \left(\frac{1}{2}\right)^i = 1 \Rightarrow \Theta(n) + \Theta(n^2) \cdot O(1) = \Theta(n^2)$

Esercizio 2 (10 punti): Scrivere un algoritmo `ElementoPiuFrequente` che, dato un array A di n interi, compresi tra 1 e $10n$ restiuisce il valore più presente all'interno dell'array, a parità di occorrenze va restituito il valore minimo.

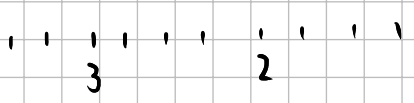
Ad esempio, se $A = [2, 6, 8, 5, 2, 3, 6, 8, 9, 5, 8, 1, 2]$, allora la risposta è 2 in quanto 2 ed 8 sono gli unici valori che compaiono 3 volte all'interno dell'array, mentre gli altri valori compaiono al più 2 volte.

Il costo computazionale dell'algoritmo proposto deve essere $\Theta(n)$.

Dell'algoritmo proposto:

a) si scriva lo pseudocodice opportunamente commentato,

b) si giustifichi il costo computazionale.




```
DEF ES2(A):
    INT B[ ] = [0] * MAX(A); # ARRAY LUNGO MAX(A)
    FOR i IN RANGE(LEN(A)-1): # SCORRO A
        B[A[i]] += 1; # CONTO LE OCCORRENZE DI A IN B.
    INT MAX = 0;
    INT i = LEN(B)-1;
    WHILE (i > 0): #SCORRENDO AL CONTRARIO, A PARITA' DI
        IF(B[i] >= MAX): OCCORRENZE TORNERO' IL MINIMO
            MAX = i;
        i -= 1;
    RETURN MAX;
```

ANALISI COSTO COMPUTAZIONALE:

```
DEF ES2(A):
    INT B[ ] = [0] * MAX(A);        Θ(1)
    FOR i IN RANGE(LEN(A)-1):       n VOLTE
        B[A[i]] += 1;               Θ(1)
    INT MAX = 0;                    Θ(1)
    INT i = LEN(B)-1;               Θ(1)
    WHILE (i > 0):                  K = MAX(n) VOLTE
        IF(B[i] >= MAX):            Θ(1)
            MAX = i;                Θ(1)
        i -= 1;                     Θ(1)
    RETURN MAX;                     Θ(1)
```

$$\Theta(1) + n\,\Theta(1) + 3\,\Theta(1) + K\left[2\,\Theta(1)\right] = \Theta(n) + \Theta(K) = \Theta(n)$$

$n > K$!

Esercizio 3 (10 punti): Sia L una lista concatenata semplicemente puntata data tramite il puntatore p alla sua testa e contenenti chiavi intere positive. Ogni record è composto da due campi: il campo *key* che contiene il valore del nodo ed il campo *next* che contiene il puntatore al nodo successivo della lista se questo esiste, il valore None altrimenti. Si progetti un **algoritmo ricorsivo** con costo computazionale $O(n)$ che restituisca un puntatore al primo elemento della lista la cui chiave sia esattamente uguale alla somma delle chiavi di tutti gli elementi precedenti; se un tale elemento non esiste, verrà ritornato None.

Ad esempio, per la lista $p \to 1 \to 2 \to 3 \to 6$ verrà restituito un puntatore al record contenente l'informazione 3; si noti che anche il record contenente l'informazione 6 soddisfa la richiesta di avere la chiave pari alla somma dei precedenti, ma il record contenente 3 lo precede.
Dell'algoritmo proposto:

a) si scriva lo pseudocodice opportunamente commentato,

b) si giustifichi il costo computazionale trovando e risolvendo l'equazione di ricorrenza.

ESSENDO LE CHIAVI INTERE E POSITIVE, È OVVIO CHE L'ELEMENTO CHE SODDISFA LA RICHIESTA PIU' A SINISTRA, SARA' ANCHE IL MINIMO.

```
DEF ES3(P, VAL=NONE):
    IF (P==NONE): RETURN NONE; # SIAMO A FINE LISTA        Θ(1)
    IF (VAL==NONE): # SE E' LA PRIMA ITERAZIONE            Θ(1)
        RETURN ES3(P->NEXT, P->KEY);                       T(n-1)
    IF (P->KEY==VAL):                                      Θ(1)
        RETURN P; # VALORE TROVATO                         Θ(1)
    RETURN (P->NEXT, P->KEY+VAL); # PROSSIMO PUNTATORE     T(n-1)
```

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

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

$$T(n) = [T(n-2) + \Theta(1)] + \Theta(1) = [[T(n-3) + \Theta(1)] + \Theta(1)] + \Theta(1)$$

$$T(n) = T(n-k) + k\Theta(1) \quad \text{FINO A } k = n-1:$$

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