ESAME GIUGNO 2022

Esercizio 1 (10 punti):
Per la soluzione di un certo problema disponiamo di un algoritmo iterativo con costo computazionale $\Theta(n^2)$. Ci viene proposto in alternativa un algoritmo ricorsivo il cui costo è catturato dalla seguente ricorrenza:

$T(n) = a \cdot T\left(\frac{n}{4}\right) + \Theta(1)$ per $n \geq 4$
$T(n) = \Theta(1)$ altrimenti
Dove a è una certa costante intera positiva con $a \geq 2$.

Determinare qual è il valore massimo che la costante intera a può avere perché l'algoritmo ricorsivo risulti asintoticamente più efficiente dell'algoritmo iterativo di cui disponiamo. **Motivare bene la vostra risposta.**

SE DOVESSIMO APPLICARE IL TEOREMA PRINCIPALE ALLA FUNZIONE RICORSIVA, NOTEREMMO CHE, ESSENDO $a \geq 2$, $n^{\log_b a} \geq \sqrt{n}$, QUINDI, $\forall a \geq 2$, $n^{\log_b a} = O(f(n))$, QUINDI, ESSENDO IL PRIMO CASO DEL TEOREMA PRINCIPALE, SI HA CHE $T(n) = \Theta(n^{\log_4 a})$ QUANDO ALLORA $\log_4 a < 3$? $\forall a \leq 15$, QUINDI, $\forall a \leq 15$, $n^{\log_4 a} < n^3$, RENDENDO L'ALGORITMO RICORSIVO PIU' EFFICENTE.

Esercizio 2 (10 punti):
Sia A un array di n interi. Con la *coppia ordinata* (i, j), $0 \leq i \leq j < n$, rappresentiamo il suo sottoarray che parte dall'elemento in posizione i e termina con l'elemento in posizione j, definiamo *valore* di un sottoarray come la somma dei suoi elementi.
Progettare un algoritmo che, dato un array A di interi positivi ed un intero positivo s, restituisce la coppia ordinata che rappresenta il sottoarray di A più a sinistra che ha valore s. Se un tale sottoarray non esiste, la funzione deve restituire $None$. L'algoritmo deve avere costo computazionale $O(n)$.

Ad esempio, per $A = [1, 3, 5, 2, 9, 3, 3, 1, 6]$

- con $s = 7$ l'algoritmo deve restituire la coppia $(2, 3)$ (ci sono infatti in A tre sottoarray con valore 7 le cui coppie nell'ordine da sinistra a destra sono $(2, 3)$, $(5, 7)$, $(7, 8)$).
- con $s = 21$ l'algoritmo deve restituire $None$ in quanto A non ha sottoarray con valore 21 .

Dell'algoritmo proposto:

a) si dia la descrizione a parole;

b) si scriva lo pseudocodice;

c) si giustifichi il costo computazionale.

PSEUDO CODICE

```
DEF ES3(A,S):
    i = 0;                                          Θ(1)
    j = 1;                                          Θ(1)
    SOMMA = A[i]+A[j];                              Θ(1)
    WHILE !(i==LEN(A)-2 AND j==LEN(A)-1):           AL PIU' 2M VOLTE
        IF SOMMA<S AND j<LEN(A)-1:                  Θ(1)
            j+=1;                                   Θ(1)
            SOMMA+=A[j];                            Θ(1)
        IF SOMMA>S AND j-i>1:                       Θ(1)
            SOMMA-=A[i];                            Θ(1)
            i+=1;                                   Θ(1)
        IF SOMMA==S:                                Θ(1)
            RETURN (i,j);                           Θ(1)
    RETURN NONE;                                    Θ(1)
```

$$\Theta(1) + 2M\Theta(1) = \Theta(m)$$

USEREMO 2 INDICI i E j INIZIALIZZATI COME 0 ED 1, FAREMO CAMMINARE j OGNI QUAL VOLTA IL SOTTO ARRAY AVRA' SOMMA INFERIORE AD S, SOMMANDO A[j], QUANDO INVECE LA SOMMA SARA' SUPERIORE, SOTTRARREMO A[i], FACENDO POI CAMMINARE i, SE LA SOMMA SARA' S, RITORNEREMO (i, j), SENNO', GIUNTI AL TERMINE CON i E j, TORNEREMO NONE;

Esercizio 3 (10 punti): Si consideri una lista concatenata dove ogni nodo ha 2 campi, il campo key contenente un intero ed il campo next con il puntatore al nodo seguente (next vale $None$ per l'ultimo nodo della lista).

Bisogna aggiornare i puntatori della lista in modo da creare una nuova lista priva dei nodi con valore superiore a 10 e in cui i nodi rimanenti appaiono in ordine inverso rispetto all'originale. Ad esempio per la lista di seguito a sinistra la funzione deve restituire la lista di seguito a destra:

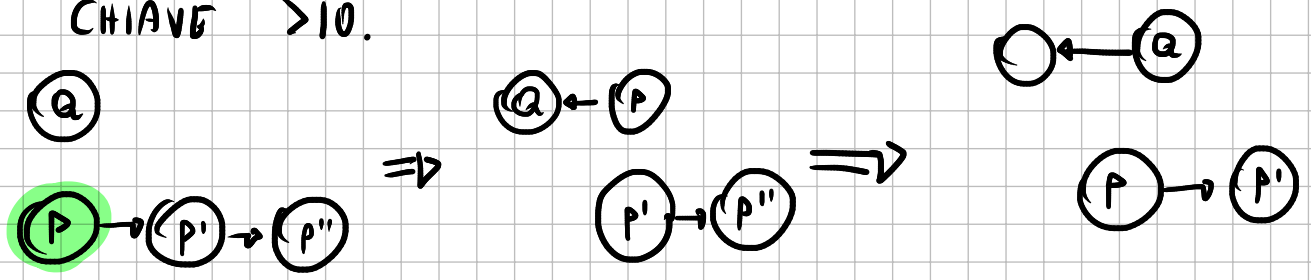
Progettare un **algoritmo** che, dato il puntatore p alla testa della lista, risolve il problema in tempo $\Theta(n)$ dove n è il numero di nodi della lista originaria. Lo spazio di lavoro dell'algoritmo proposto deve essere $\Theta(1)$ (in altri termini non è possibile definire e utilizzare altre liste o nodi).
Dell'algoritmo proposto:

a) si dia la descrizione a parole,

b) si scriva lo pseudocodice,

c) si giustifichi il costo computazionale.

ITERERO' PER I NODI COSTRUENDO UNA LISTA D'APPOGGIO ALLA QUALE COLLEGHERO' IN TESTA IL NODO. TAGLIERO' I VALORI CON CHIAVE >10.




```
DEF ES3(P):
    WHILE (P->KEY >10):
        P = P->NEXT;
    Q=NONE;
    WHILE (P):
        IF(P.NEXT.KEY >10):
            P->NEXT = P->NEXT->NEXT
            T = P.NEXT;
            P->NEXT = Q;
            Q=P;
            P=T;
    RETURN Q;
```

ESEGUE IL CICLO PER TUTTI I NODI DELLA LISTA, QUINDI IL COSTO È $\Theta(n)$