Domanda A (7 punti) Si consideri la funzione ricorsiva `search(A,p,r,k)` che dato un array A, ordinato in modo decrescente, un valore k e due indici p, q con $1 \leq p \leq r \leq A.length$ restituisce un indice i tale che $p \leq i \leq r$ e $A[i] = k$, se esiste, e altrimenti restituisce 0.

```
search(A,p,r,k)
   if p <= r
      q = (p+r)/2
      if A[q] = k
         return q
      elseif A[q] > k
         return search(A,q+1,r,k)
      else
         return search(A,p,q-1,k)
   else
      return 0
```

Dimostrare induttivamente che la funzione è corretta. Inoltre, determinare la ricorrenza che esprime la complessità della funzione e risolverla con il Master Theorem.

Esercizio 2 (8 punti) Per $n > 0$, siano dati due vettori a componenti intere $\mathbf{a}, \mathbf{b} \in \mathbf{Z}^n$. Si consideri la quantità $c(i, j)$, con $0 \leq i \leq j \leq n-1$, definita come segue:

$$c(i,j) = \begin{cases} a_i & \text{se } 0 < i \leq n-1 \text{ e } j = n-1, \\ b_j & \text{se } i = 0 \text{ e } 0 \leq j \leq n-1, \\ c(i-1, j-1) \cdot c(i, j+1) & 0 < i \leq j < n-1. \end{cases}$$

Si vuole calcolare la quantità $m = \min\{c(i,j) : 0 \leq i \leq j \leq n-1\}$.

1. Si fornisca il codice di un algoritmo iterativo bottom-up per il calcolo di m.
2. Si valuti la complessità esatta dell'algoritmo, associando costo unitario ai prodotti tra numeri interi e costo nullo a tutte le altre operazioni.

Esercizio 2 (9 punti) Sia n un intero positivo. Si consideri la seguente ricorrenza $M(i, j)$ definita su tutte le coppie (i, j) con $1 \leq i \leq j \leq n$:

$$M(i,j) = \begin{cases} 2 & \text{se } i = j, \\ 3 & \text{se } j = i+1, \\ M(i+1, j-1) \cdot M(i+1, j) + M(i, j-1) & \text{se } j > i+1. \end{cases}$$

(a) Si scriva una coppia di algoritmi INIT_M(n) e REC_M(i, j) per il calcolo memoizzato di $M(1, n)$.

(b) Si calcoli il numero esatto $T(n)$ di moltiplicazioni tra interi eseguite per il calcolo di $M(1, n)$.

Domanda A (7 punti) Definire formalmente la classe $\Omega(f(n))$. Dimostrare che la seguente ricorrenza ha soluzione $T(n) = \Omega(n)$

$$T(n) = \frac{1}{3} T(n-1) + 2n + 1$$

Esercizio 1 (10 punti) Realizzare una funzione `Diff(A,k)` che, dato un array `A[1,n]` ordinato in senso decrescente, verifica se esiste una coppia di indici `i, j` tali che `A[i]` − `A[j]` = `k`. Restituisce la coppia di indici se esiste e (0,0) altrimenti. La funzione non deve alterare l'input e deve operare in spazio costante. Scrivere lo pseudocodice, provarne la correttezza e valutarne la complessità.

Domanda B (4 punti) Dato l'array $A = [60, 90, 49, 65, 46, 73, 88, 45, 43]$, mostrare in forma di albero, il max-heap prodotto dalla procedura `BuildMaxHeap`.

Esercizio 1 (9 punti) Realizzare una funzione `avgTree(T)` che dato un albero binario `T` con chiavi numeriche, verifica se, per ogni nodo che abbia discendenti, la chiave del nodo è maggiore o uguale della media delle chiavi dei discendenti e ritorna conseguentemente un valore booleano (la radice dell'albero è `T.root` e ogni nodo `x` ha i campi `x.left` `x.right` e `x.key`).

Esercizio 2 (9 punti) Lungo una strada ci sono, in vari punti, n parcheggi liberi e n auto. Un posteggiatore ha il compito di parcheggiare tutte le auto, e lo vuole fare minimizzando lo spostamento totale da fare. Formalmente, dati n valori reali $p_1, p_2, \ldots, p_n$ e altri n valori reali $a_1, a_2, \ldots, a_n$, che rappresentano

le posizioni lungo la strada rispettivamente di parcheggi e auto, si richiede di assegnare ad ogni auto a_i un parcheggio $p_{h(i)}$ minimizzando la quantità

$$\sum_{i=1}^{n} |a_i - p_{h(i)}|.$$

1. Si consideri il seguente algoritmo greedy. Si individui la coppia (auto, parcheggio) con la minima differenza. Si assegni quell'auto a quel parcheggio. Si ripeta con le auto e i parcheggi restanti fino a quando tutte le auto sono parcheggiate. Dimostrare che questo algoritmo non è corretto, esibendo un controesempio.

2. Si consideri il seguente algoritmo greedy. Si assuma che i valori $p_1, p_2, \ldots, p_n$ e $a_1, a_2, \ldots, a_n$ siano ordinati in modo non decrescente. Si produca l'assegnazione $(a_1, p_1), (a_2, p_2), \ldots, (a_n, p_n)$. Dimostrare la correttezza di questo algoritmo per il caso $n = 2$.