Cognome: **Nome**: **# Matricola**: **Riga**: **Col**:

Algoritmi e Strutture Dati – 09/02/23 – Parte A

Esercizio -1 Iscriversi allo scritto entro la scadenza. In caso di inadempienza, -1 al voto finale.

Esercizio 0 Scrivere correttamente nome, cognome, numero di matricola, riga e colonna su tutti i fogli consegnati. Consegnare foglio A4 e foglio protocollo di bella. In caso di inadempienza, -1 al voto finale.

Esercizio A1 – Punti ≥ 8

Si consideri questo algoritmo:

```
int P(int[] A, int n)
    return R(A, 1, n)

int R(int[] A, int start, int end)
    if start == end then
        return A[start]
    int tot = 0
    int i = start
    int j = end
    while i ≤ j and A[i] == A[j] do
        tot = tot + A[i] + A[j]
        i = i + 1
        j = j − 1
    int mid = ⌊(end + start)/2⌋
    tot = tot + R(A, start, mid)
    tot = tot + R(A, mid + 1, end)
    return tot
```

Scrivere l'equazione di ricorrenza associata a questo algoritmo e calcolare la complessità computazionale che ne deriva, facendo particolare attenzione al caso pessimo e al caso ottimo.

Esercizio A2 – Broadcast – Punti ≥ 10

Si supponga che si debba inviare un messaggio a tutti i nodi di un albero binario. All'inizio, solo la radice conosce il messaggio. Ad ogni turno, ogni nodo che conosce il messaggio può inviarlo ad uno solo dei suoi figli (all'altro dovrà inviarlo al turno successivo); al termine del turno, i nodi ricevono il messaggio e possono spedirlo il turno successivo.

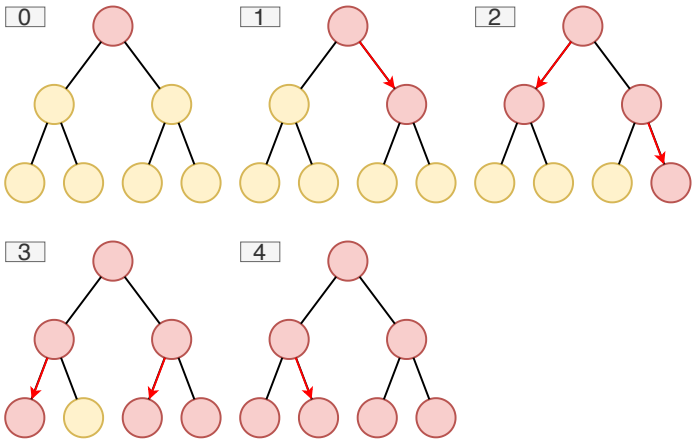
Scrivere un algoritmo

int minRounds(TREE T)

che restituisce il numero minimo di turni necessari affinché tutti i nodi di un albero binario T non vuoto vengano a conoscenza del messaggio.

Discutere informalmente la correttezza dell'algoritmo e la sua complessità computazionale.

Per esempio, nell'albero perfetto seguente, il numero di round necessari per raggiungere tutti i nodi è pari a 4. In rosa i nodi che hanno ricevuto il messaggio, in ocra quelli che non l'hanno ancora ricevuto; le frecce rosse indicano il messaggio che viene spedito durante un round; per ogni figura marcata i, la situazione illustrata è quella al termine del round i-esimo, tranne $i = 0$ che rappresenta la situazione iniziale.



Esercizio A3 – Local minima – Punti ≥ 12

All'interno di un vettore A contenente $n \geq 3$ interi distinti, un minimo locale è un elemento $A[i]$ tale che $A[i-1] > A[i] < A[i+1]$, con $1 < i < n$.

Scrivere un algoritmo

int findLocalMinimum(**int**[] A, **int** n)

che prenda in input un vettore A contenente n interi per cui vale la relazione:$A[1] > A[2]$ e $A[n-1] < A[n]$, e restituisca l'indice di un minimo locale in tempo $O(\log n)$. Algoritmi con complessità lineare o superiore non verranno considerati.

Si può dimostrare che se i primi due e gli ultimi due elementi di un vettore sono tali che $A[1] > A[2]$ e $A[n-1] < A[n]$, allora A contiene un minimo locale.

Dimostrare la correttezza dell'algoritmo proposto.

Per esempio, se $A = [17, 15, \mathbf{13}, 14, \mathbf{12}, 18]$, ci sono due minimi locali, 13 e 12. L'algoritmo può restituire 3 o 5 (indici che partono da 1).

Cognome: **Nome**: **# Matricola**: **Riga**: **Col**:

Algoritmi e Strutture Dati – 09/02/23 – Parte B

Esercizio -1 Iscriversi allo scritto entro la scadenza. In caso di inadempienza, -1 al voto finale.

Esercizio 0 Scrivere correttamente nome, cognome, numero di matricola, riga e colonna su tutti i fogli consegnati. Consegnare foglio A4 e foglio protocollo di bella. In caso di inadempienza, -1 al voto finale.

Esercizio B1 – MCS – Punti ≥ 8

I concetti di sottosequenza e sottosequenza comune si possono applicare ai vettori di interi, così come sono definiti sulle stringhe. Per esempio, $[1, 2]$ è *una* sottosequenza comune di $[1, -2, 2, 4]$ e $[1, 4, 2]$
Il *valore di una sottosequenza* è pari alla somma dei suoi valori. Per esempio, il valore di $[1, 2]$ è 3. Una *sottosequenza comune di valore massimale* (maximal common subsequence, MCS) è una sottosequenza comune a due vettori di interi, il cui valore è massimo fra tutte le sottosequenze comuni di tali vettori.

Scrivere un algoritmo:

$$\textbf{int } \mathsf{mcs}(\textbf{int}[\,]\ T,\ \textbf{int}[\,]\ U,\ \textbf{int}\ n,\ \textbf{int}\ m)$$

che restituisca il valore della MCS contenuta nei vettori T e U, di lunghezza n e m, rispettivamente.

Discutere informalmente la correttezza dell'algoritmo e la sua complessità computazionale.

Per esempio, se $T = [1, -2, 2, 4]$ e $U = [1, 4, 2]$, la sottosequenza comune di valore massimale è $[1, 4]$ e l'algoritmo dovrà restituire il suo valore, ossia 5.

Esercizio B2– Facoltiadi – Punti ≥ 10

L'organizione delle Facoltiadi vi ha incaricato di formare le squadre.
All'Università di Trento, ci sono $k = 14$ dipartimenti. Ognuno di essi deve formare esattamente una squadra. Una squadra è composta da 21 membri. Ci sono $n \geq 14 \cdot 21 = 294$ studenti che hanno chiesto di partecipare, quindi idealmente si potrebbero selezionare 294 di essi e inserirli nelle squadre.

Tuttavia, c'è un limite sul bilancio di genere e ogni squadra deve contenere almeno 7 studenti del genere meno rappresentato fra "maschi" e "femmine". Ogni studente i seleziona un genere g_i fra "maschio", "femmina" oppure "preferisco non rispondere". Uno studente identificato da "preferisco non rispondere" può ovviamente partecipare ma non viene contato nel bilancio di genere.

Ogni studente i-esimo può partecipare a solo una squadra, quella del suo dipartimento d_i o del dipartimento affine d'_i (a solo titolo di esempio, matematica e fisica sono affini, DISI e DII sono affini, CIBIO e CIMEC sono affini, etc.).

Descrivere un algoritmo che prenda in input gli n studenti, le loro preferenze di genere, i loro dipartimenti e dipartimenti affini e produca un assegnamento studenti - squadre, che rispetti le regole sul bilancio di genere e sull'appartenenza ai dipartimenti. Discutere informalmanente la correttezza e la complessità dell'algoritmo proposto.

Esercizio B3 – k-prodotto – Punti ≥ 12

Scrivere un algoritmo

$$\textbf{int } \mathsf{productK}(\textbf{int}[\,]A,\ \textbf{int}\ n,\ \textbf{int}\ m)$$

che prenda in input un vettore contenente n interi positivi e un intero positivo m, e restituisca il numero di sottosequenze di A il cui prodotto è inferiore o uguale a m.

Discutere informalmente la correttezza dell'algoritmo e la sua complessità computazionale.

Per esempio, sia $A = [1, 3, 2, 4]$, $n = 4$ ed $m = 10$; le sottosequenze possibili sono:

$[], [1], [3], [2], [4], [1, 3], [1, 2], [1, 4], [3, 2], [2, 4], [1, 3, 2], [1, 2, 4]$

e l'algoritmo deve restituire 12. Si noti che assumiamo che il prodotto di una sequenza vuota sia pari a 1, e quindi sia inferiore o uguale a qualunque intero positivo m.