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

Algoritmi e Strutture Dati – 17/01/2022 – Parte A

Esercizio A1 – Complessità – Punti ≥ 8

Trovare i limiti superiore e inferiore più stretti possibili per la seguente equazione di ricorrenza:

$$T(n) = \begin{cases} 1 & n < 8 \\ 2T(\lfloor n/2 \rfloor) + 31T(\lfloor n/8 \rfloor) + n^2 - n & n \geq 8 \end{cases}$$

Esercizio A2 – Alberi made up – Punti ≥ 10

Un albero binario *ma-come-se-li-inventa* è un albero in cui:

- i nodi che si trovano in livelli dispari hanno esattamente 1 figlio
- i nodi che si trovano in livelli pari hanno 0 o 2 figli.

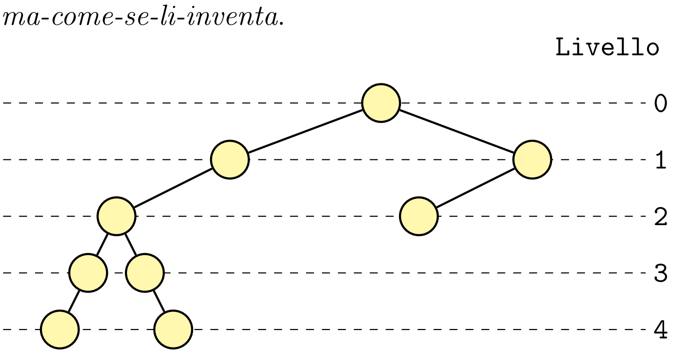
Vi ricordo che la radice si trova nel livello 0, i suoi figli si trovano nel livello 1, i suoi nipoti nel livello 2, ecc.

Scrivere un algoritmo

boolean isMakeup(Tree T)

che prende in input un albero binario T non vuoto e restituisce **true** se T è un albero *ma-come-se-li-inventa*, **false** altrimenti.

Per esempio, l'albero seguente è un albero binario *ma-come-se-li-inventa*.



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

Esercizio A3 – Serie aritmetica bucata – Punti ≥ 12

Si consideri una progressione aritmetica

$$x, x + d, x + 2d, \ldots, x + nd$$

composta da $n + 1$ termini, con $n \geq 2$ e $d > 0$.
Supponiamo che venga memorizzata, in ordine, in un vettore A di n elementi, omettendo di memorizzare uno dei termini all'interno della serie (in altre parole, x e $x + nd$ sono sicuramente presenti).
Scrivere un algoritmo

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

che prende in input il vettore A descritto sopra e restituisce il termine mancante.

Spiegare il funzionamento e discutere la complessità computazionale dell'algoritmo proposto. *Soluzioni in tempo lineare o superiore non verranno considerate.*

Per esempio, se $A = [3, 5, 7, 9, 11, 15, 17]$, l'algoritmo deve restituire 13.

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

Algoritmi e Strutture Dati – 17/01/2022 – Parte B

Esercizio B1 – Ottali – Punti ≥ 8

Scrivere una funzione

$$\mathsf{printOctals}(\mathbf{int}\ n)$$

che stampa tutti i numeri ottali (numeri in base 8, con cifre fra 0 e 7) di n cifre che non contengono due cifre uguali consecutive.

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

Per esempio, con $n = 2$ bisogna stampare i seguenti 56 numeri ottali (non necessariamente in quest'ordine):
10 20 30 40 50 60 70 01 21 31 41 51 61 71 02 12 32 42 52 62 72 03 13 23 43 53 63 73 04 14 24 34 54 64 74 05 15 25 35 45 65 75 06 16 26 36 46 56 76 07 17 27 37 47 57 67

Esercizio B2 – Longest repeating sequence – Punti ≥ 10

Scrivere un algoritmo

$$\mathbf{int}\ \mathsf{lrs}(\textsc{Item}[]\ T,\ \mathbf{int}\ n)$$

che prende in input una sequenza A e restituisce la lunghezza della sottosequenza ripetuta massimale contenuta in A. Una sottosequenza si dice ripetuta se è contenuta due volte nella sequenza originale, senza che nessun elemento della prima ripetizione sia usato per la seconda ripetizione.

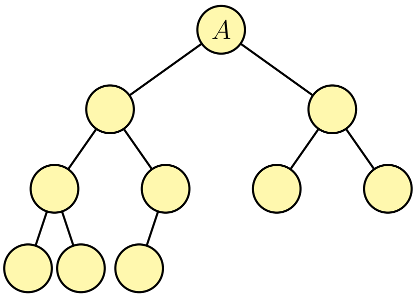
Discutere informalmente la correttezza dell'algoritmo e la sua complessità computazionale.
Per esempio,

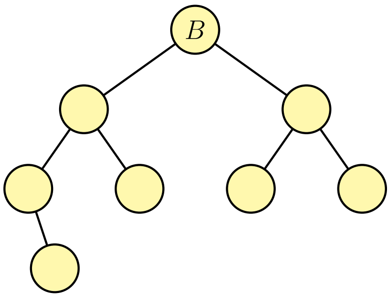
- la più lunga sottosequenza ripetuta in `"AABBCC"` è `"ABC"` e la funzione deve restituire 3;
- la più lunga sottosequenza ripetuta in `"ABCABC"` è `"ABC"` e la funzione deve restituire 3;
- la più lunga sottosequenza ripetuta in `"AAB"` è `"A"`, perché `"B"` non può essere usato in due sottosequenze; la funzione deve restituire 1;
- le più lunghe sottosequenze ripetute in `"ABCCBA"` sono `"A"`, `"B"`, `"C"` e l'algoritmo deve restituire 1.

Esercizio B3 – Ben bilanciato – Punti ≥ 12

Un albero binario T è ben-bilanciato se tutti i suoi nodi sono ben-bilanciati. Un nodo t è ben-bilanciato se il numero n_R di nodi nel sottoalbero destro di t e il numero n_L di nodi nel sottoalbero sinistro di t differiscono al più di 1 ($|n_L - n_R| \leq 1$).

Negli esempi successivi, l'albero binario con radice A non è ben-bilanciato (perché il sottoalbero sinistro della radice ha 6 nodi, mentre quello di destra ne ha 3), mentre quello con radice B è ben-bilanciato.





Scrivere un algoritmo

$$\mathbf{int}\ \mathsf{countWellBalanced}(\mathbf{int}\ n)$$

che prenda in input un intero n e restituisca il *numero* di alberi ben-bilanciati contenenti n nodi.

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

Il numero di alberi ben-bilanciati per n che va da 0 a 9 è pari a: 1, 1, 2, 1, 4, 4, 4, 1, 8, 16