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

Algoritmi e Strutture Dati - 31/01/18

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 1 – Punti ≥ 6 (Parte A)

Trovare i limiti superiore e inferiori più stretti possibili per l'equazione di ricorrenza qui a lato, utilizzando il metodo di sostituzione (o per tentativi). Prestare attenzione ai casi base.

$$T(n) = \begin{cases} T(\lfloor n/2 \rfloor) + T(\lfloor n/3 \rfloor) + n \log n & n > 1 \\ 1 & n \leq 1 \end{cases}$$

Esercizio 2 – Punti ≥ 6 (Parte A)

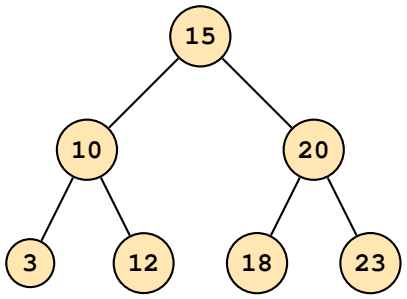
Dati un albero binario di ricerca T (contenente valori interi distinti) e due valori interi min, max, con $min \leq max$, scrivere un algoritmo

intoccurrences(TREE T, **int** min, **int** max)

che restituisca il numero di valori di T che sono compresi fra min e max, estremi inclusi.

Discutere informalmente la correttezza della soluzione proposta e calcolare la complessità computazionale. Soluzioni che non richiedano la visita dell'intero albero, pur restando $O(n)$ nel caso pessimo, verranno valutate maggiormente.

Nell'esempio a lato, occurrences$(T, 11, 18)$ deve restituire 3 (per i valori $12, 15, 18$).



Esercizio 3 – Punti ≥ 9 (Parte A)

Un'area geografica è suddivisa in $n \times n$ *celle*. Ogni cella può contenere alberi (1) oppure prati (0). Una *foresta* è un'area contigua di celle adiacenti contenenti alberi, dove dove due celle sono *adiacenti* se hanno un lato in comune (non in diagonale).

Scrivere un algoritmo **int** searchForest(**int**[][] A, **int** n)
che prenda in input una matrice $n \times n$ di interi e restituisca la dimensione della foresta più grande, dove la dimensione è il numero di celle contenenti alberi.

Discutere informalmente la correttezza della soluzione proposta e calcolare la complessità computazionale.

Nell'esempio a lato, la foresta più grande è quella ombreggiata in grigio; la funzione searchForest() deve quindi restituire 7.

0	0	**1**	**1**	**1**
0	0	0	0	**1**
1	0	**1**	0	**1**
1	**1**	**1**	0	0
1	**1**	0	**1**	**1**

Esercizio 4 – Punti ≥ 10 (Parte A)

Sia A un vettore *ordinato* contenente n interi, in cui tutti i valori presenti compaiono esattamente due volte tranne uno che compare una volta sola. Scrivere un algoritmo

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

che prenda in input A ed n, e restituisca il *valore* che compare una volta sola.

Discutere informalmente la correttezza della soluzione proposta e calcolare la complessità computazionale. Soluzioni con costo $O(n)$ daranno punteggio 20%, in quanto banali.

Esempio: per $A = [1, 1, 3, 3, 4, 4, 5, 5, 7, 7, 8]$, il vostro algoritmo deve restituire 8.