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

# *Algoritmi e Strutture Dati – 24/08/2022 – 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.

## A1 – Complessità – Punti $\geq 8$

Trovare i limiti superiore e inferiore più stretti possibili, *non necessariamente uguali*, per la seguente equazione di ricorrenza:

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

## A2 – DAG massimale – Punti $\geq 11$

Scrivere un algoritmo:

**int** maxDag(Graph $G$)

che prende in input un grafo diretto aciclico (DAG) $G$ e restituisce il numero massimo di archi che è possibile aggiungere al grafo senza creare cicli.

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

## A3 – Somma su albero – Punti $\geq 11$

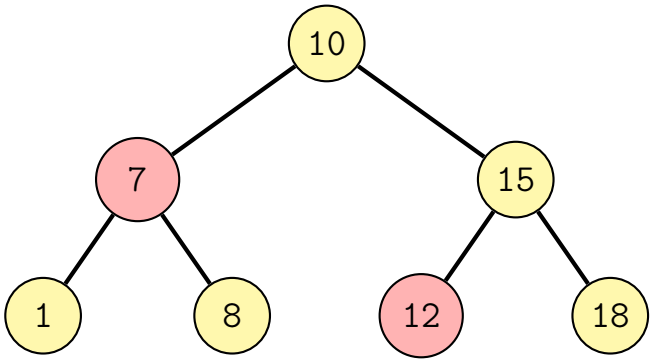
Scrivere un algoritmo

**boolean** containsSum(Tree $T$, **int** $k$)

che prende in input un albero binario di ricerca $T$ e un valore intero $k$, e ritorna **true** se esistono due nodi nell'albero la cui somma è $k$.

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

Per esempio, nell'albero seguente esistono due nodi (evidenziati in rosa) la cui somma è 19; non esiste una coppia di nodi la cui somma è 15.



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

# *Algoritmi e Strutture Dati – 24/08/2022 – 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.

## B1 – Esami – Punti $\geq 8$

L'ufficio Supporto alla Didattica dell'Università di Trento ti ha assunto per scrivere un algoritmo per programmare gli esami.

- Esistono $C$ corsi, per ognuno dei quali deve essere programmato un esame finale; ogni corso $c_i$ ha $cs_i$ studenti;
- esistono $A$ aule; ogni aula $a_j$ può contenere al più $as_j$ studenti
- esistono $O$ slot orari per gli esami; ogni slot $o_k$ può contenere al più un esame;
- esistono $S$ supervisori; ogni supervisore $s_l$ può supervisionare al massimo un esame alla volta;
- un esame del corso $c_i$ non può essere assegnato all'aula $a_j$, se $cs_i \geq as_j$;
- ogni esame deve essere supervisionato da esattamente un supervisore
- ogni supervisore $s_l$ è disponibile negli slot contenuto nell'insieme $T_l$;
- ogni supervisore $s_l$ può supervisionare al più $d$ esami nel periodo considerato;
- non possono esserci due esami contemporaneamente nella stessa aula.

Descrivere un algoritmo per programmare gli esami e discuterne la complessità.

## B2 – Permutazioni eleganti – Punti $\geq 10$

Una permutazione $a_1, a_2, \ldots, a_n$ dei numeri interi $1, 2, \ldots, n$ è *elegante* se ogni suo elemento $a_i$ soddisfa la seguente proprietà: $a_i$ è divisibile per $i$ oppure $i$ è divisibile per $a_i$.
Scrivere un algoritmo printElegant(**int** $n$) che stampa tutte le permutazioni elegante dei primi $n$ interi.

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

Per $n = 5$, le permutazioni eleganti sono le seguenti:

```
[1, 2, 3, 4, 5], [1, 4, 3, 2, 5],
[2, 1, 3, 4, 5], [2, 4, 3, 1, 5],
[3, 2, 1, 4, 5], [3, 4, 1, 2, 5],
[4, 1, 3, 2, 5], [4, 2, 3, 1, 5],
[5, 2, 3, 4, 1], [5, 4, 3, 2, 1]
```

## B3 – Connessioni – Punti $\geq 12$

Siano $X$ e $Y$ due vettori, ognuno dei quali contiene $n$ interi. Una *connessione* è una coppia di indici $(x, y)$ che contengono lo stesso valore nei due vettori: $X[x] = Y[y]$. Un insieme di connessioni $C = \{(x_1, y_1), (x_2, y_2), \ldots, (x_k, y_k)\}$ è detto *regolare* se:

- ogni elemento del vettore $X$ è connesso al massimo ad un elemento del vettore $Y$;
- ogni elemento del vettore $Y$ è connesso al massimo ad un elemento del vettore $X$;
- non esistono due connessioni intersecanti; due connessioni $(x_1, y_1)$ e $(x_2, y_2)$ in $C$ si dicono *intersecanti* se $x_1 < x_2 \wedge y_1 > y_2$ oppure $x_1 > x_2 \wedge y_1 < y_2$.

Scrivere un algoritmo

**int** maxConnections(**int**[] $X$, **int**[] $Y$, **int** $n$)

che prende in input i due vettori e la loro dimensione, e restituisce la dimensione dell'insieme di connessioni regolare e massimale, cioè il più grande insieme di connessioni che rispetti le regole descritte sopra.

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

La figura seguente riporta due insiemi di connessioni: quello a sinistra è regolare e massimale; quindi l'algoritmo deve restituire il valore 4. Quello a destra non è regolare poiché ci sono tre intersezioni e poiché il valore 8 del vettore $X$ è collegato a due elementi del vettore $Y$. L'algoritmo deve restituire 3 nel caso a destra.