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

Algoritmi e Strutture Dati – Parte 1 - 07/02/2019

Esercizio A1 – Punti ≥ 8

Trovare un limite superiore, il più stretto possibile, per la seguente equazione di ricorrenza:

$$T(n) = \begin{cases} T(\lfloor n/2 \rfloor) + T(\lfloor n/4 \rfloor) + \\ T(\lfloor n/8 \rfloor) + 2T(\lfloor n/16 \rfloor) + 1 & n > 16 \\ 1 & n \leq 16 \end{cases}$$

Esercizio A2 – Punti ≥ 10

Sia A un vettore ordinato di n interi, con valori potenzialmente ripetuti e sia v un valore intero. Scrivere un algoritmo:

$$\textbf{int } \mathsf{first}(\textbf{int}[\,]\ A, \textbf{int}\ n, \textbf{int}\ v)$$

con complessità $\Theta(\log n)$ che restituisca la posizione della prima occorrenza di v in A. Per semplicità, si assuma che v sia sempre presente almeno una volta nel vettore.

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

Ad esempio, se $A = [1, 1, 2, 2, 2, 3, 4, 5, 6, 6]$ e $v = 2$, allora il vostro algoritmo deve restituire 3, perché il primo 2 compare in posizione 3; se $v = 4$, allora il vostro algoritmo deve restituire 7.

Esercizio A3 – Punti ≥ 12

Sia $G = (V, E)$ un grafo non orientato connesso contenente n persone e m relazioni simmetriche ("contatti Telegram").

Le elezioni sono alle porte ed è ufficialmente iniziata la campagna elettorale. All'inizio della campagna (giorno 0), ogni persona (nodo) fa parte o del Partito Elitario (`PE`), o del Partito Populista (`PP`), oppure è indeciso (`I`).

Il giorno 0, entrambi i partiti fanno partire una catena virale a proprio favore: tutti i membri del `PE` ricevono un messaggio m_e, tutti i membri del `PP` ricevono un messaggio m_p.

Ogni giorno $t \geq 0$, ogni nodo (indipendentemente dal partito di appartenenza) che riceve un messaggio m **per la prima volta** lo inoltra a tutti i propri contatti, che lo ricevono il giorno successivo. Nessun nodo inoltra lo stesso messaggio più di una volta.

Sia u un nodo che era indeciso all'inizio, e siano t_e e t_p i giorni in cui u ha ricevuto i messaggi m_e e m_p **per la prima volta**, rispettivamente:

- se $t_e < t_p$, u voterà per il partito `PE`
- se $t_e > t_p$, u voterà per il partito `PP`
- se $t_e = t_p$, u resterà indeciso.

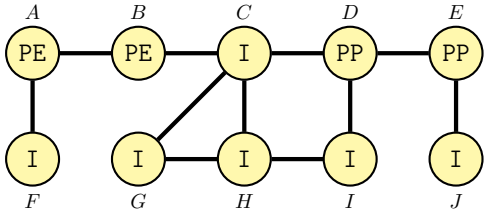
Supponendo che la campagna duri abbastanza affinché entrambi i messaggi raggiungano tutti i nodi, scrivere un algoritmo:

$$\textbf{int } \mathsf{election}(\textsc{Graph}\ G, \textbf{int}[\,]\ \mathit{status})$$

che prende in input un grafo G e un vettore *status* tale che $\mathit{status}[u]$ contiene, per ogni nodo u, un valore simbolico `PE`, `PP` o `I`, e restituisce il numero di nodi che resteranno indecisi al termine del processo.

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

Esempio – Giorno 0 (Situazione iniziale)



I giorni in cui i vari nodi ricevono un messaggio per `PE`/`PP` per la prima volta (t_e/t_p) sono riassunti nella tabella seguente. I nodi che vengono raggiunti nello stesso giorno sono C, G e H, quindi l'algoritmo deve restituire 3.

	A	B	C	D	E	F	G	H	I	J
t_e	0	0	**1**	2	3	1	**2**	**2**	3	4
t_p	3	2	**1**	0	0	4	**2**	**2**	1	1

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

Algoritmi e Strutture Dati – Parte 2 - 07/02/2019

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 – Punti ≥ 8

Scrivere un algoritmo: **int** bits(**int** n)
che dato un intero n, restituisca il numero di stringhe binarie di n bit che non contengono bit 1 consecutivi.

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

Ad esempio, per $n = 5$ le stringhe da contare sono le seguenti

```
00000 00001 00010 00100 00101
01000 01001 01010 10000 10001
10010 10100 10101
```

e l'algoritmo deve restituire 13.

Esercizio B2 – Punti ≥ 10

Si consideri un testo T di n caratteri e un pattern P di m caratteri. Abbiamo già visto problemi in cui si chiedeva quante volte il pattern P appare come **sotto-sequenza** di T (ricordate Lucia e i Promessi Sposi?). In questo compito, invece di contare bisogna **elencare** tutti i modi possibili.

Più formalmente, sia $I = [i_1, \ldots, i_m]$ una sequenza ordinata contenente m indici di T (ovvero compresi fra 1 ed n).
Diciamo che I *genera* P *da* T se e solo se:

$$\forall k, 1 \leq k \leq m : T[i_k] = P[k]$$

Ad esempio, data la stringa $T =$ "`gooddog`", la sequenza di indici $I = [4, 6, 7]$ genera il pattern $P =$ "`dog`".

Dati un testo T di n caratteri e un pattern P di m caratteri, scrivere un algoritmo:

printAll(Item[] T, **int** n, Item[] P, **int** m)

che stampa tutte le sequenze di indici di T che generano P.

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

Ad esempio, si consideri il testo "`gooddog`" e il pattern "`god`". L'algoritmo deve stampare (senza commenti):

```
1, 2, 4    # gooddog
1, 2, 5    # gooddog
1, 3, 4    # gooddog
1, 3, 5    # gooddog
```

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

Esercizio B3 – Punti ≥ 12

Si consideri una matrice binaria M di dimensione $n \times n$. Scrivere un algoritmo

$$\textbf{int } \mathsf{cross}(\textbf{int}[\,][\,]\ M, \textbf{int } n)$$

che restituisca la dimensione della più grande croce composta da valori `1` presente nella matrice. Una croce è composta da un bit `1` centrale e quattro braccia di uguale lunghezza, contenenti bit `1`. Non deve essere necessariamente circondata da bit `0`.

Discutere complessità e correttezza dell'algoritmo proposto. Si noti che esistono algoritmi $\Theta(n^2)$ e $\Theta(n^3)$.

Per esempio, la croce visibile nell'esempio è composta da 17 valori 1 e l'algoritmo dovrà restituire 17.

1	0	1	1	1	1	0	1	1	1
1	0	1	0	1	1	1	0	1	1
1	1	1	0	1	1	0	1	0	1
0	0	0	0	1	0	0	1	0	0
1	0	0	0	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1	0
1	0	0	0	1	0	0	1	0	1
1	0	1	1	1	1	0	0	1	1
1	1	0	0	1	0	0	0	0	1
1	0	1	1	1	1	0	1	0	0