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

Algoritmi e Strutture Dati – Parte 1 - 28/06/2018

Esercizio A1 – Punti ≥ 8

Si consideri la seguente equazione di ricorrenza:

$$T_a(n) = \begin{cases} 7T_a(n/3) + T_a(n/a) + n^2 \log n & n > 1 \\ 1 & n \leq 1 \end{cases}$$

Per quali valori *interi* di a la funzione $T_a(n)$ è limitata superiormente da $O(n^2 \log n)$? Giustificare la risposta.

Esercizio A2 – Punti ≥ 10 (Soluzione mia: 6 righe)

Si consideri un albero binario T. Una *bi-colorazione* dell'albero è un assegnamento di colori identificati dai numeri $1, \ldots, 2$ ad ognuno dei nodi dell'albero, tale per cui due nodi adiacenti nell'albero (uno padre dell'altro) hanno colori diversi.

Il *costo della bi-colorazione* è dato dalla somma dei valori dei colori assegnati ai nodi.

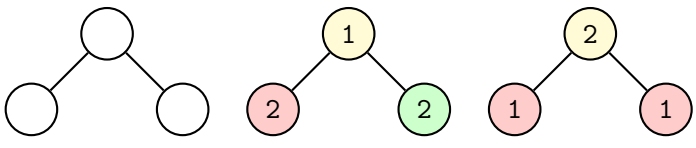
Scrivere un algoritmo

int minColoring(Tree T)

che prende in input un albero T e restituisce il più piccolo valore di costo fra tutte le bi-colorazioni possibili dell'albero T.

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

Si consideri l'esempio seguente. Sulla sinistra, un albero binario T ancora da colorare. In centro, una possibile bi-colorazione con costo $1 + 2 + 3 = 6$. A destra, una possibile bi-colorazione con costo $2+1+1 = 4$. Questa è anche la bi-colorazione di costo minimo, quindi il vostro algoritmo dovrà restituire 4.



Esercizio A3 – Punti ≥ 12 (Soluzione mia: 6 righe)

La Corea del Nord, forte delle recenti aperture di Trump, si è aggiudicata i mondiali di calcio del 2030[1]. Kim-Jong Un ha deciso di rinunciare alla tradizionale divisione in gironi e finali. Si adotterà il seguente metodo: vengono scelte due squadre e le si fa giocare. La squadra che perde viene fucilata sul campo, la squadra che vince viene rimessa nel gruppo. Si giocano $n - 1$ partite, al termine delle quali l'unica squadra rimasta viva è la vincitrice. I club hanno espresso qualche perplessità.

Ad esempio, si consideri un torneo con tre squadre. Francia batte Germania, la Germania viene eliminata. Poi Italia batte Francia, la Francia viene eliminata. L'Italia è l'unica squadra rimasta, vince il mondiale.

Siete tifosi dell'Italia e siete stati incaricati di scegliere l'ordine delle partite. Come tutti gli anni, c'è un qualche animale in qualche zoo europeo che azzecca tutti i pronostici[2]. Avete quindi a disposizione un grafo $G = (V, E)$, dove V è l'insieme delle n squadre. Per ogni coppia di nodi distinti u, v, sono dati due casi possibili:

- $(u, v) \in E$, $(v, u) \notin E$: u vincerà lo scontro fra u e v;
- $(v, u) \in E$, $(u, v) \notin E$: v vincerà lo scontro fra u e v.

Non è possibile pareggiare.

Scrivere un algoritmo

boolean canItalyWin(Graph G, **int** s)

che prende in input un grafo G contenente squadre e pronostici e un nodo s che rappresenta l'Italia, e restituisce **true** se e solo se esiste un ordine con cui effettuare le partite che permetta all'Italia di vincere il Mondiale 2030; restituisce **false** altrimenti.

Bonus – Solo se avete risolto tutto il resto del compito

Assumendo che esista un ordinamento che permetta all'Italia (nodo s) di vincere, scrivere un algoritmo

printOrder(Graph G, **int** s)

che stampi tale ordinamento.

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

[1] http://english.yonhapnews.co.kr/culturesports/2017/06/12/0702000000AEN20170612010851315.html
[2] https://it.wikipedia.org/wiki/Polpo_Paul

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

Algoritmi e Strutture Dati – Parte 2 - 28/06/2018

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, niente foglio di brutta. In caso di inadempienza, -1 al voto finale.

Esercizio B1 – Punti ≥ 8 (Soluzione mia: 8 righe)

Sapete bene cos'è una stringa palindroma. Alcune stringhe non sono palindrome a causa di qualche carattere di troppo o qualche carattere di differenza. Ad esempio, è possibile rendere palindroma "BAOBAB" eliminando la lettera "O" oppure cambiando la "O" in "B". Per rendere palindroma "ODORARONO", bisogna cambiare la "D" in "N" (ottenendo "ONORARONO"), o viceversa.

Scrivere un algoritmo **int** minPalindrome(ITEM[ ] S, **int** n) che restituisce il numero minimo di operazioni necessarie per rendere una stringa palindroma, dove le operazioni permesse sono la rimozione di un carattere e la sostituzione di un carattere con un altro in una delle due stringhe.

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

Esercizio B2 – Punti ≥ 10 (Soluzione mia: 15 righe, di cui 6 per stampare bene la sommatoria)

Per fare un esempio per un problema, ho copiato e incollato una somma di numeri da un sito. Come a volte capita, certi caratteri non vengono copiati; in particolare, non è stato copiato il + e questo è il risultato: "1151832075512". A quali somme possibili potrebbe corrispondere questa stringa?
Scrivere un algoritmo printAll(**int**[ ] S, **int** n) che prenda una stringa di numeri di lunghezza n (rappresentata da un vettore di interi compresi fra 0 e 9) e stampi tutti i modi in cui è possibile leggere la stringa come una sommatoria di valori interi.
Ad esempio, "3412" dà origine a:

```
3 + 4 + 1 + 2
34 + 1 + 2
3 + 41 + 2
341 + 2
3 + 4 + 12
34 + 12
3 + 412
3412
```

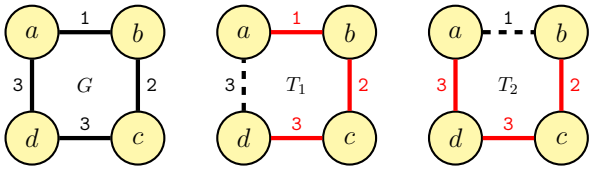
Discutere informalmente la correttezza della soluzione proposta e calcolare la complessità computazionale.

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

Esercizio B3 – Punti ≥ 12 (Soluzione mia: 14 righe)

Sia dato un grafo G e supponiamo che esista una funzione di pesi $w : E \to \mathbf{N}$. Un albero di copertura T è *k-minimale* per G se ha peso minimo fra tutti gli alberi di copertura di G che contengono solo archi di peso superiore o uguale a k.

Nell'esempio seguente, l'albero T_1 è 1-minimale per G e ha peso 6, ma non è 2-minimale (perché contiene un arco di peso 1). L'albero T_2 è 2-minimale per G e ha peso 8, perché tutti i suoi archi pesano 2 o più. Non esiste nessun albero 3-minimale, poiché i due archi con peso 3 o più non sono in grado di formare un albero di copertura.



La *snellezza* $S(T)$ *di un albero di copertura* T è data dalla differenza fra il peso più alto e il peso più basso dei suoi archi:

$$S(T) = \max_{e \in T} w(e) - \min_{e \in T} w(e)$$

Nell'esempio precedente, T_1 ha snellezza $2 = 3 - 1$ mentre T_2 ha snellezza $1 = 3 - 2$.

Dato un grafo G e un intero k, scrivere un algoritmo

int kMinimal(GRAPH G, **int** k) oppure
int kminimal(EDGE[] A, **int** n, **int** m, **int** k)

che restituisca la *snellezza* dell'albero di copertura k-minimale. Se non esiste un albero k-minimale, l'algoritmo deve restituire $+\infty$.

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

Bonus – Solo se avete risolto tutto il resto del compito

Un albero di copertura è *snellissimo* per G se ha snellezza minima fra tutti gli alberi di copertura k-minimali per G, per qualunque $k \geq 1$.

La *snellezza* $S(G)$ *di un grafo* G corrisponde alla snellezza di un albero di copertura snellissimo per G.

Nell'esempio precedente, G ha snellezza 1, perché T_2 è un albero di copertura snellissimo.

Dato un grafo G, scrivere un algoritmo **int** thinnest(GRAPH G) oppure **int** thinnest(EDGE[] A, **int** n, **int** m) (vettore di archi) che restituisca la snellezza del grafo.

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