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

Algoritmi e Strutture Dati – Parte A – 22/08/2019

Esercizio A1 – Punti ≥ 8

Durante un orale di Algoritmi, non ricordando come è implementata la funzione Merge() contenuta in MergeSort(), uno studente ha deciso di improvvisare e mi ha proposto questa versione:

```
Merge(ITEM A[], int primo, int ultimo, int mezzo)
for i = primo + 1 to ultimo do
    ITEM temp = A[i]
    int j = i
    while j > 1 and A[j − 1] > temp do
        A[j] = A[j − 1]
        j = j − 1
    A[j] = temp
```

Scrivere le equazioni di ricorrenza corrispondenti al costo computazionale nel caso *ottimo* e al caso *pessimo* dell'algoritmo MergeSort() così modificato e risolverle tramite Master Theorem.

Esercizio A2 – Punti ≥ 10

Sia T un albero binario di ricerca i cui nodi hanno un campo *color* che può assumere il valore RED o BLACK. Scrivere un algoritmo

boolean isRedBlack(TREE T)

che prenda in input un albero T e restituisca **true** se e solo se la colorazione rispetta le regole red-black, **false** altrimenti.

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

Esercizio A3 – Punti ≥ 12

In un corso di laurea sono previsti n esami, tutti obbligatori. Esistono m vincoli di propedeuticità, rappresentati tramite un DAG $G = (V, E)$. Se $(u, v) \in E$, l'esame u è propedeutico all'esame v; questo significa che v deve essere sostenuto in una sessione d'esame successiva a quella in cui si è svolto u. Se due o più esami non hanno vincoli di propedeuticità, allora possono essere sostenuti tutti all'interno di una stessa sessione.

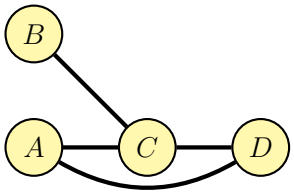
Scrivere un algoritmo

int minSessions(GRAPH G)

che prenda in input un grafo di propedeuticità G e restituisca il numero minimo di sessioni necessarie per completare tutti gli esami.

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

Ad esempio, nel grafo seguente, A, B possono essere dati nella prima sessione, C nella seconda, D nella terza. L'algoritmo deve quindi restituire il valore 3.



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

Algoritmi e Strutture Dati – Parte B - 22/08/2019

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

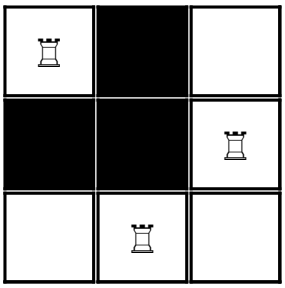
Si consideri una scacchiera $n \times n$, in cui alcune caselle sono colorate di bianco ed altre di nero. La scacchiera è rappresentata da una matrice booleana M di dimensione $n \times n$, dove **true** corrisponde al bianco e **false** corrisponde al nero. Avete un numero non limitato di torri, che possono essere piazzate nella matrice seguendo le seguenti regole:

- Le torri possono essere piazzate solo su caselle colorate di bianco
- Ci può essere al più una torre per ogni riga
- Ci può essere al più una torre per ogni colonna

Descrivere un algoritmo che prenda in input M e restituisca il numero massimo di torri che possono essere piazzate nella scacchiera seguendo le regole.

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

Nell'esempio seguente, è possibile posizionare tre torri come in figura.



Esercizio B2 – Punti ≥ 10

I costi degli skipass sono saliti alle stelle. Sugli abbonamenti giornalieri c'è scritto "non trasferibile", ma d'altronde non ci sono foto sui giornalieri...

Siete una comitiva di n persone in vacanza e il vostro obiettivo è quello di comprare il numero minimo di skipass da condividere, tenuto conto che non tutti vogliono sciare contemporaneamente. [1]

Ogni mattina, ogni persona dichiara l'intervallo di tempo in cui vuole sciare: dal minuto $start[i]$ (incluso) al minuto $end[i]$ (escluso), misurati a partire dalle 8.00 del mattino; ovviamente $start[i] < end[i]$.

Scrivere un algoritmo che prenda in input i vettori $start$, end (di dimensione n) e restituisca il numero minimo di skipass da comprare, in modo tale che lo stesso skipass non venga utilizzato contemporaneamente da più persone.

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

Per esempio, se lo sciatore 1 vuole sciare dalle 8.00 alle 9.00 ($[0, 60[$), lo sciatore 2 dalle 9.00 alle 10.00 ($[60, 120[$), lo sciatore 3 dalle 8.00 alle 10.00 ($[0, 120[$), è possibile comprare due skipass, uno dei quali verrà condiviso dagli sciatori 1 e 2.

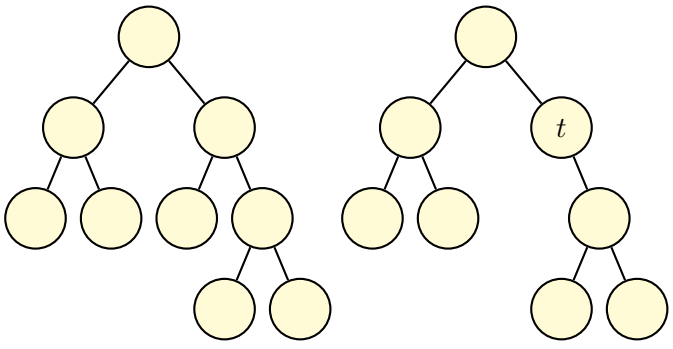
Esercizio B3 – Punti ≥ 12

Un albero binario è 1-bilanciato se e solo se, per ogni nodo dell'albero, la differenza fra l'altezza del suo sottoalbero destro e l'altezza del suo sottoalbero sinistro è al più 1.

Scrivere un algoritmo che prenda in input un'altezza h e restituisca il numero di alberi binari 1-bilanciati strutturalmente diversi di altezza h.

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

Nella figura sotto, l'albero di sinistra è 1-bilanciato, l'albero di destra no, in quanto il sottoalbero sinistro di t ha altezza 0 e il sottoalbero destro di t ha altezza 2; la differenza è quindi maggiore di 1.



[1] Don't try this in your favorite ski area!