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

Algoritmi e Strutture Dati – Parte 2 - 06/06/2019

Esercizio B1 – Punti ≥ 6

Dato un vettore V contenente n interi distinti, scrivere una funzione

boolean checkPriority(**int**[] V, **int** n)

che ritorna **true** se il vettore V rappresenta un albero min-heap di n elementi memorizzato in un vettore, **false** altrimenti.

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

Esercizio B2 – Punti ≥ 12

ColorfulFlag.com produce bandiere contenenti n strisce colorate, ognuna delle quali prende uno di k colori, numerati da 1 a k. Ovviamente, come mi ricorda anche mia moglie ogni mattina, ci sono alcuni colori che non possono stare vicini perché sono un pessimo abbinamento.

Fortunatamente, GoodPantone.com ha fornito a ColorfulFlag.com (ma non a me) una matrice boolean simmetrica A di dimensione $k \times k$, tale che $A[c_1][c_2] = A[c_2][c_1] =$ **true** se e solo se i colori c_1 e c_2 sono abbinabili, **false** altrimenti. Per evitare strisce vicine di colore uguale, si assuma che $A[c][c] =$ **false** per ogni colore $1 \leq c \leq k$.

Scrivere un algoritmo:

printFlags(**boolean**[][] A, **int** n, **int** k)

che preso in input una matrice A di dimensione $k \times k$ e gli interi n e k, stampi tutte le possibili bandiere contenenti n strisce colorate, tali per cui ogni coppia di strisce contigue abbia colori abbinabili secondo la matrice A.

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

Per esempio, con tre strisce e con tre colori $1, 2, 3$ tali per cui il colore 1 non è abbinabile al colore 2, le bandiere possibili sono le seguenti:

$$[1,3,1], [1,3,2], [2,3,1], [2,3,2], [3,1,3], [3,2,3]$$

Esercizio B3 – Punti ≥ 12

Si consideri una scacchiera di dimensione $n \times n$. Si consideri un pedone che parte dalla cella $(1, 1)$ (in alto a sinistra) e si può muovere solo in basso (riga = riga + 1) o a destra (colonna = colonna + 1), fino ad arrivare alla cella (n, n) (in basso a destra). Si consideri una matrice di costi P di dimensione $n \times n$ che associa ad ogni cella (r, c) della scacchiera un costo $P[r][c]$ compreso fra 0 e 10. Per semplicità, si assuma che $P[1][1] = 0$.

- Il *costo* di un cammino da $(1, 1)$ a (n, n) è pari alla somma dei pesi delle sue celle.
- Dato un budget B, un cammino è *valido* se il suo costo è inferiore o uguale al budget.
- Due cammini si dicono *distinti* se gli insiemi delle celle che attraversano sono diversi, ovvero differiscono per almeno una cella.

Scrivere una funzione:

int countPaths(**int**[][] P, **int** n, **int** B)

che, data la matrice di pesi P di dimensione $n \times n$ e un budget B, restituisca il numero di cammini da $(1, 1)$ a (n, n) che siano validi e distinti.

Discutere informalmente la correttezza della soluzione proposta e calcolare la complessità computazionale. In particolare, si evidenzi se l'algoritmo è polinomiale, pseudopolinomiale o superpolinomiale.

Come esempio, si consideri la matrice di input proposta qui sotto e si assuma che il budget sia $B = 4$.

0	1	2
1	1	1
2	1	1

Esistono quattro cammini possibili, evidenziati in rosso e grassetto. La funzione dovrà restituire 4.

0	**1**	2
1	**1**	**1**
2	1	**1**

0	**1**	2
1	**1**	1
2	**1**	**1**

0	1	2
1	**1**	**1**
2	1	**1**

0	1	2
1	**1**	1
2	**1**	**1**