

TP Informatique 09

Exercice 1

Soit $G = (S, A)$ un graphe valué, orienté, sans circuit absorbant avec $n = \text{Card } S$ l'ordre du graphe et $S = \llbracket 0; n - 1 \rrbracket$. Pour $(i, j, k) \in \llbracket 0; n - 1 \rrbracket^3$, on note $w_{i,j}^{(k)}$ le poids d'un plus court chemin du sommet i au sommet j de sommets intermédiaires dans $\llbracket 0; k \rrbracket$ s'il existe un tel chemin et $+\infty$ sinon. On pose

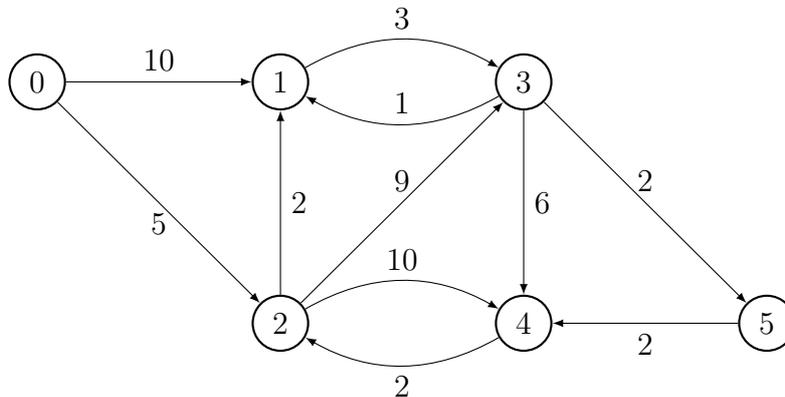
$$\forall k \in \llbracket 0; n - 1 \rrbracket \quad W^{(k)} = \left(w_{i,j}^{(k)} \right)_{0 \leq i, j \leq n-1}$$

Pour $k = -1$, la matrice $W^{(-1)}$ est la matrice d'adjacence du graphe G .

On a $\forall (i, j, k) \in \llbracket 0; n - 1 \rrbracket^3 \quad w_{i,j}^{(k)} = \min \left(w_{i,j}^{(k-1)}, w_{i,k}^{(k-1)} + w_{k,j}^{(k-1)} \right)$

1. Écrire une fonction `matr_adj(S, A)` d'arguments S une liste de sommets et A un dictionnaire contenant une liste d'adjacence et qui renvoie une liste de listes M correspondant à la matrice d'adjacence du graphe (S, A) .
2. Écrire une fonction `fw_cost(M)` d'argument M une matrice d'adjacence qui renvoie la matrice $W^{(n-1)}$ (matrice au format liste de listes) puis la tester sur le graphe fourni en exemple.
3. Modifier la fonction précédente en `fw_cost_path(M)` d'argument M une matrice d'adjacence pour que celle-ci renvoie la matrice $W^{(n-1)}$ ainsi que la matrice $P = (p_{i,j})_{0 \leq i, j \leq n-1}$ des prédécesseurs où $p_{i,j}$ désigne le prédécesseur de j dans un plus court chemin de i à j avec $(i, j) \in \llbracket 0; n - 1 \rrbracket^2$.

Les fonctions à compléter sont dans le fichier `TP09_EX01.py`.



Exercice 2

Soit $S_n = [c_1, \dots, c_n]$ un système de monnaie, les c_i désignant des montants entiers de *jetons*. Le problème du *rendu de monnaie* consiste, étant donné un montant entier v à rendre, à minimiser le nombre de jetons pour réaliser ce rendu, autrement dit à réaliser

$$M(S_n, v) = \min \left\{ \sum_{i=1}^n x_i : (x_1, \dots, x_n) \in \mathbb{N}^n, \sum_{i=1}^n x_i c_i = v \right\}$$

On a
$$M(S_n, v) = 1 + \underset{1 \leq k \leq n}{\text{Min}} M(S_n, v - c_k)$$

avec
$$M(S_n, 0) = 0 \quad \text{et} \quad \forall v \in \llbracket 1; \min(S_n) - 1 \rrbracket \quad M(S_n, v) = +\infty$$

1. Écrire une fonction `rendu1(S, v)` d'arguments S un système de monnaie et v un entier qui renvoie $M(S, v)$ en réalisant le calcul de manière descendante sans mémoïsation.
2. Écrire une fonction `rendu2(S, v)` d'arguments S un système de monnaie et v un entier qui renvoie $M(S, v)$ en réalisant le calcul de manière descendante avec mémoïsation.
3. Écrire une fonction `rendu3(S, v)` d'arguments S un système de monnaie et v un entier qui renvoie $M(S, v)$ en réalisant le calcul de manière ascendante.
4. Tester chaque fonction pour les configurations suivantes (en stoppant l'exécution si celle-ci s'avère trop lente) : $S = [2, 3]$ et $v = 1$ (pas de rendu possible), $S = [1, 3, 4]$ (système non canonique) et $v = 6$, puis $S = [1, 2, 5, 10, 20, 50, 100, 200, 500]$ avec $v = 29$, $v = 1989$ et $v = 111111$.
5. Modifier la fonction `rendu3(S, v)` en `rendu_tab(S, v)` pour que celle-ci renvoie la liste $[M(S, k), k \in \llbracket 0; v \rrbracket]$ et la liste $[p_k, k \in \llbracket 0; v \rrbracket]$ où p_k est la dernière pièce utilisée lors du calcul de minimisation.
6. Écrire une fonction `argmin_rendu(S, v)` d'arguments S un système de monnaie et v un entier et qui renvoie une liste de jetons permettant de réaliser le rendu de monnaie de v avec S . La fonction renverra `False` si le rendu n'est pas possible.