ISM MP, Informatique
Année 2025/2026

TP Informatique 09

Exercice 1

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

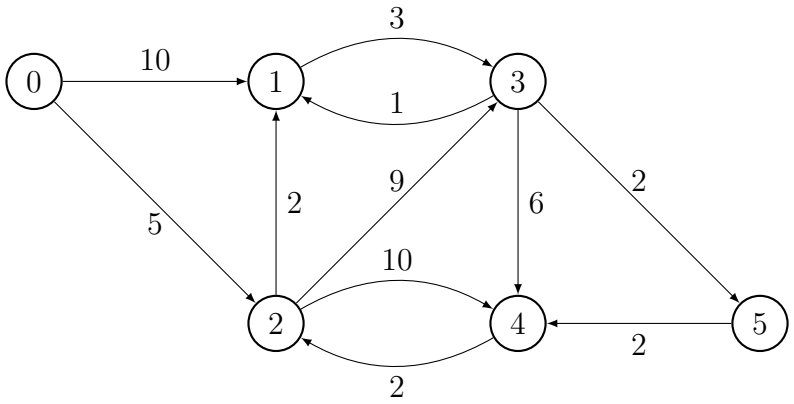
$$\forall k \in [\![0\,;\, n-1]\!] \qquad \mathrm{W}^{(k)} = \left(w_{i,j}^{(k)}\right)_{0 \leqslant i,j \leqslant n-1}$$

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

On a $$\forall (i,j,k) \in [\![0\,;\, n-1]\!]^3 \qquad w_{i,j}^{(k)} = \min\left(w_{i,j}^{(k-1)}, w_{i,k}^{(k-1)} + w_{k,j}^{(k-1)}\right)$$

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 $\mathrm{W}^{(n-1)}$ (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 $\mathrm{W}^{(n-1)}$ ainsi que la matrice $\mathrm{P} = \left(p_{i,j}\right)_{0 \leqslant i,j \leqslant n-1}$ des prédécesseurs où $p_{i,j}$ désigne le prédécesseur de j dans un plus court chemin de i à j avec $(i,j) \in [\![0\,;\, n-1]\!]^2$.

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



Exercice 2

Soit $\mathrm{S}_n = [c_1, \ldots, c_n]$ 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

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

On a
$$\mathrm{M}(\mathrm{S}_n, v) = 1 + \min_{1 \leqslant k \leqslant n} \mathrm{M}(\mathrm{S}_n, v - c_k)$$

avec
$$\mathrm{M}(\mathrm{S}_n, 0) = 0 \quad \text{et} \quad \forall v \in [\![1 \,;\, \min(\mathrm{S}_n) - 1]\!] \qquad \mathrm{M}(\mathrm{S}_n, v) = +\infty$$

1. Écrire une fonction `rendu1(S,v)` d'arguments `S` un système de monnaie et `v` un entier qui renvoie $\mathrm{M}(\mathrm{S}, v)$ 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 $\mathrm{M}(\mathrm{S}, v)$ 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 $\mathrm{M}(\mathrm{S}, v)$ 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) : $\mathrm{S} = [2, 3]$ et $v = 1$ (pas de rendu possible), $\mathrm{S} = [1, 3, 4]$ (système non canonique) et $v = 6$, puis $\mathrm{S} = [1, 2, 5, 10, 20, 50, 100, 200, 500]$ 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 $[\ \mathrm{M}(\mathrm{S}, k), \ k \in [\![0 \,;\, v]\!]\ ]$ et la liste $[\ p_k, \ k \in [\![0 \,;\, v]\!]\ ]$ 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.