TP 17 : coût minimal d'un chemin

PTSI - 2025/2026

I Introduction

- Dans ce TP on va s'intéresser à des ***graphes pondérés***, c'est à dire des graphes pour lesquels, outre les sommet et les arêtes, un ***coût*** de celle-ci est connu.

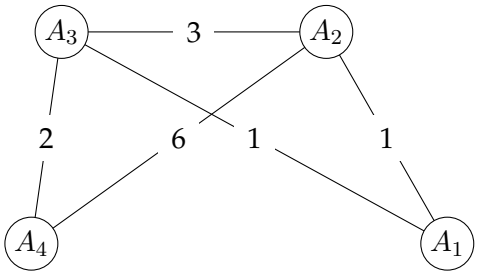
  EXEMPLE 1. L'exemple le plus simple est bien entendu celui d'un réseau routier pour lequel, outre le fait qu'une liaison routière existe entre deux villes, les distances ou les temps de parcours de cette liaison doivent être pris en compte.

- Le but de ce TP est de programmer un algorithme de calcul du "coût minimal" pour aller d'un sommet à un autre dans un graphe pondéré.
- La notion de ***coût minimal*** correspondant en fait à celle de la longueur d'un déplacement routier si les arêtes sont les routes et les pondérations les longueurs (ou temps de parcours) de celles-ci :

$$l(s_0, s_1, ..s_n) = \sum_{k=0}^{n} l(s_i, s_{i+1}))$$

où $l(s_i, s_{i+1})$ est le "coût" de l'arête.

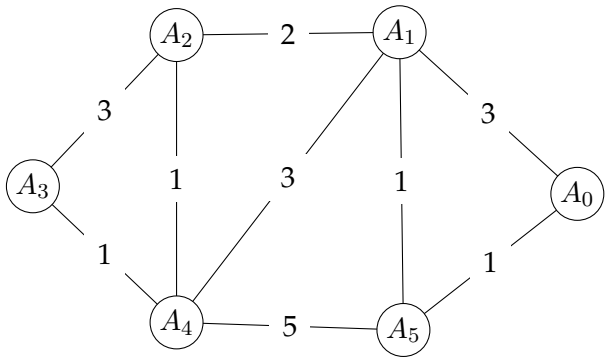
EXEMPLE 2. Sur la graphe ci-dessous, le coût du chemin $A_2 \to A_3 \to A_4$ est $3+2=5$ est-il le coût minimal ?



II Présentation de l'algorithme de DIJKSTRA (1930-2002)

II.1 Sur un exemple

On considère le graphe ci-dessous :



Auquel on va associer le tableau ci-après que l'on va compléter à chaque étape en appliquant l'algorithme de DIJSTRA depuis le sommet A_3 (choix arbitraire) :

- prendre le sommet dont le coût total mesuré jusque là est le plus faible. On le note S. Le marquer comme traité.
- Pour chacun de ses sommets adjacent, noté Ad, et qui n'a pas déjà été traité :
  - Vérifier si le coût du chemin obtenu jusque là pour aller de A_3 jusqu'à S ajouté au coût de l'arête (S, Ad) est inférieur au coût précédemment mémorisé pour aller de A_3 à Ad. Si oui on met à jour en conservant le coût le plus faible et on indique aussi qu'il a été obtenu en passant par S comme sommet précédent.
- Recommencer tant que le sommet sélectionné a des sommets adjacents.

Recopier et compléter le tableau ci-dessous en appliquant l'algorithme précédent :

	A_0	A_1	A_2	A_3	A_4	A_5
états des coûts	`inf`	`inf`	$3, A_3$	0	$1, A_3$	`inf`

Vérifier qu'il permet de donner le chemin le plus court en partant de A_3 dans les cas suivants :

- Chemin de A_3 à A_0
- Chemin de A_3 à A_5

II.2 Programme Python

On donne un squelette dans le cahier de prépa. Celui ci utilise utilise la constante `np.inf` qui simule une valeur flottante infinie

Complétez ce squelette pour le que la fonction `dijkstra` qui s'y trouve implémente le calcul du chemin à coût minimal par l'algorithme vu précédemment mais assurez vous avant : .

1. Que vous êtes capable d'écrire le contenu de `lst` après sa définition.
2. Que vous comprenez bien comment elle va évoluer dans l'algorithme (faire une étape vous même sur l'exemple).

II.3 Preuve de l'algorithme

On va utiliser les notations suivantes :

- s est le sommet de départ.
- $l(a, b)$ est le coût de l'arête (a, b) potentiellement infini (si il n'y a pas d'arête) et plus généralement $l(C)$ est le coût total du chemin C.
- $\min(u)$ est le coût du plus court chemin depuis s jusqu'à u dans le graphe fourni.
- à chaque itération de la boucle, T est l'ensemble des sommets traités.
- pour chaque sommet u, $d(u)$ est le coût obtenu de s à u par l'algorithme avant l'exécution du bloc de boucle.
- $\min\limits_{T}(u)$ est le coût du plus court chemin depuis s jusqu'à u en utilisant les points de T. (avec éventuellement $u \notin T$)

On donne également une version semi-formalisée de l'algorithme :

Entrées : Un ensemble de sommets $[0, 1.., N]$, un sommet s, un tableau des longeurs des aretes du graphe
Sorties : Un tableau des coûts minimaux entre s et les autres sommets

$T \leftarrow [s]$
$d \leftarrow [l(s, 0), .., l(s, s), ...] = [l(s, i)$ pour i dans $S]$
$n \leftarrow 0$
tant que *il existe un* $u \notin T$ *tel que* $d(u) < +\infty$ **faire**
    choisir sat tel que $d(sat) = \min\limits_{u \notin T}(d(u))$
    $T \leftarrow T \cup \{sat\}$
    pour $u \notin T$ **faire**
        si $d(sat) + l(sat, u) < d(u)$ **alors**
            $d(u) \leftarrow d(sat) + l(sat, u)$
        fin si
    fin pour
fin tant que
retourner d

1. Estimer la coût temporel de votre fonction Python.

2. Démontrer que l'assertion suivante est un invariant de la boucle principale dans la fonction :

$$A : \forall u \in T, d(u) = \min(u) \quad \wedge \quad \forall c \notin T, d(u) = \min_{T}(u)$$

On admettra que lorsque $d(u) < +\infty$ il existe un chemin $C = (s_0, ..., s_k, u)$ avec $s_0, .., s_k)$ dans T joignant s à u. (autrement dit un « chemin dans T).

3. Conclure sur la correction de l'algorithme implémenté par la fonction précédente.