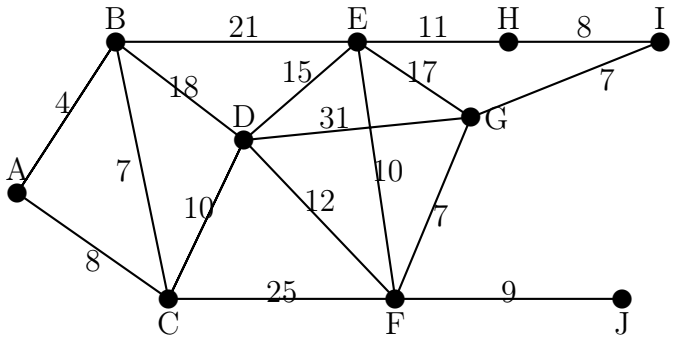
TP9 : GRAPHES II

Algorithme de Dijkstraa

On reprend le graphe du TP précédent, avec des pondérations :



Supposons que celui-ci représente le temps de trajet entre des villes. On souhaite connaître le trajet le plus court entre les villes A et I. L'algorithme de Dijkstraa va nous permettre de répondre efficacement à cette question.
Avant tout on définit la distance $d(x, y)$ entre deux sommets x et y comme le minimum des poids des chemins joignant x à y. Remarquons que ce minimum est atteint parmi les chemins élémentaires.

Description de l'algorithme

Données :
Un graphe pondéré X fini, connexe , poids ≥ 0, sans boucle. On note $p(x, y)$ le poids de l'arête joignant x à y, avec la convention $p(x, y) = +\infty$ si il n'y a pas d'arête joignant x à y.

Résultat attendu :
Étant donné un sommet A (départ) et un sommet Z (arrivée), trouver $d(A, Z)$ ainsi que le chemin qui réalise cette distance. En fait l'algorithme va trouver la distance entre A et n'importe quel autre sommet.
Principe :

- Variables :
  - Une liste S de sommets (ceux dont on connaît la distance à A)
  - Une liste S' de sommets (ceux qui, sans être dans S, ont un antécédent dans S)
  - Pour chaque sommet x, un réel $d_A(x)$ (appelé à être égal à $d(A, x)$)
  - Pour chaque élément $x \in S \cup S' \setminus \{A\}$, un élément de S : $\pi(x)$ (élément qui précède x sur le plus court chemin dont tous les éléments sont dans S (sauf éventuellement x) joignant A à x - on convient d'appeler un tel chemin un S-chemin).
- Initialisation :
  - $S = \{A\}$
  - $S' = \{x \in X,\ p(A, x) < +\infty\}$
  - $d_A(A) = 0$, pour $x \in S'$, $d_A(x) = p(A, x)$, pour les autres $d_A(x) = +\infty$.

  - Comme on a vu, $\pi(A)$ n'est pas défini ; pour $x \in S'$, $\pi(x) = A$.

- Progression :

  P1 Déterminer $x_0 \in X \setminus \{S\}$ tel que $d_A(x_0)$ est minimal (par conséquent $x_0 \in S'$)

  P2 On adjoint x_0 à S

  P3 Pour tous les $x \notin S$ tels que $p(x_0, x) < +\infty$, on adjoint x à S'. Pour chacun de ces x, on compare $d_A(x)$ et $d_A(x_0) + p(x_0, x)$ et si ce dernier est inférieur, on remplace.

- Condition d'arrêt : $S = X$

- Résultat : pour tout x, $d_A(x) = d(A, x)$, $\pi(x)$ le prédécesseur de x sur un plus court chemin de A à x.

Établir la trace de cet algorithme afin de déterminer le plus court chemin de A à J sur notre exemple. On présentera cela dans un tableau contenant à chaque étape, pour chaque sommet x, le couple $(d_A(x), \pi(x))$ dès qu'il est défini.

Implémentation

Un graphe pondéré d'ordre n est donné.
On suppose donné un tableau (liste de listes) `Table_poids` contenant les données relatives au graphe ayant servi d'exemple : en position `[i,j]` le poids de l'arrête reliant le sommet `i` au sommet `j` (`inf` si il n'y a pas d'arrête).
On souhaite calculer les distances d'un sommet fixé à tout autre sommet.
Pour cela, on va faire évoluer 3 listes :

- `listeS` : `listeS[x]` vaut 0, 1 ou 2 suivant que le sommet `x` n'a pas encore été traité, est dans S' ou S ;
- `listeDA` , contenant pour chaque sommet x la valeur de $d_A(x)$;
- `listeAnte` , liste donnant, pour chaque éléments de S ou S', son antécédant sur le plus cours chemin.

Compléter le programme fourni dans le fichier à compléter : `TP_graphes_2_Dijkstraa_Acompleter.py`.