TP18 Graphes : Dijkstra.

ECG1

Juin 2023

I Graphes pondérés

I.1 Mise en place du problème et implémentations

Définition 1 (Graphe pondéré)**.**
*Un **graphe pondéré** est un graphe* (S, A) *munit d'une fonction de valuation* p *définie sur l'ensemble des arêtes et à valeur dans* $\mathbb{R}$ *(ou plus souvent* $\mathbb{R}_+$*)*

Remarque : Cette définition peut s'appliquer à un graphe orienté ou non.

Exemple :

- Réseau d'autoroute avec le prix des péages
- Réseau routier avec longueur des chemins
- Réseau routier avec temps de parcours
- Réseau de communication avec capacité maximale de chaque lien

Exemple :

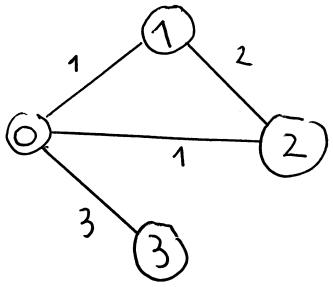


FIGURE 1 – Exemple 1

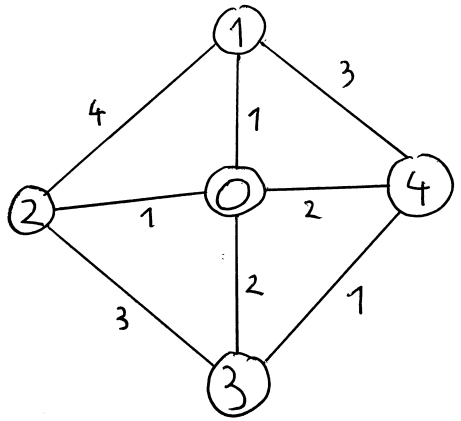
On peut représenter en mémoire un graphe pondéré de nombreuses façons :

1. Matrice des poids M, définie comme une matrice d'adjacence mais le coefficient $M_{i,j}$ vaut 0 si il n'y a pas d"arête entre i et j, et prend pour valeur le poids de l'arête dans le cas contraire
2. On peut choisir de numéroter les sommets par $\{0,\dots,n-1\}$ et représenter le graphe par une liste `adj` ou adj[i] est une liste (éventuellement vide)de couple de la forme (voisin, poids) donnant la liste des voisins de i dans le graphe et le poids de l'arête les joignant.
3. On donne une liste de triplet (depart,arrivée,poids) représentant chacune des arêtes (on pourra dans le cas d'un graphe non orienté ne faire apparaître qu'une fois une arête dans un sens arbitraire)

Exemple : En reprenant l'exemple précédent

1. La matrice [[0,1,1,3],[1,0,2,0],[1,2,0,0],[3,0,0,0]]
2. Les liste d'adjacences pondérées [[(1,1),(2,1),(3,3)],[(0,1),(2,2)],[(0,1),(1,2)],[(0,3)]]
3. La liste des arêtes [(0,1,1),(0,2,1),(0,3,3),(1,2,2)]

Exercice 1.
Donner les trois représentations pour le graphe suivant



Dans la suite on utilisera principalement les listes d'adjacences pondérées

Définition 2 (poids d'un chemin/longueur d'un chemin)**.**
Le poids (total) d'un chemin *est la somme des poids des arêtes qui le constitue, si on fait l'analogie entre poids d'un chemin, on utilisera aussi le terme* longueur (totale) du chemin

Exemple : Dans le graphe de l'exemple 1 le chemin (0,1,2,0,3) a un poids total de 7.

II Plus court chemin : Algorithme de Dijkstra

II.1 Mise en place

On donne un graphe pondéré où tous les poids sont **positifs**. On se donne un sommet de départ et on cherche à résoudre une classe de problèmes

- Trouver le chemin le plus court entre un point et un autre.
- Trouver le chemin le plus court entre un sommet de départ et tous les sommets accessibles.
- Trouver la **longueur** chemin le plus court entre un sommet de départ et tous les sommets accessibles.

II.2 L'algorithme

Cette version va donner la longueur du plus court chemin du sommet de départ à chacun des sommets accessibles.

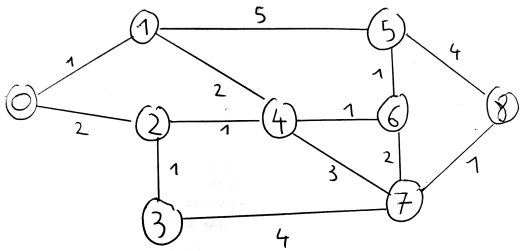
- À chaque étape chaque sommet est associé à une distance qui est la distance du plus court chemin trouvé jusqu'à cette étape. Un sommet pour lequel on n'a pas (encore) trouvé un chemin se voit attribué la distance ∞. On dispose aussi d'un moyen de connaître les sommets qu'il reste à examiner et les sommets déjà examinés.
- On initialise l'algorithme en attribuant 0 au sommet de départ et $+\infty$ à tous les autres sommets. Le sommet de départ est le seul à traiter.
- on choisit parmi les sommets à traiter celui qui a la distance la plus petite (on le note x dans la suite),
- Si x n'a pas déjà été marqué comme traité, on le marque. Dans ce cas là, pour tous les voisins y de x on calcule si le chemin passant par x arrivant à y est plus court que la distance actuellement trouvé, si oui on modifie cette distance. On place si nécessaire y dans les sommets à traiter
- L'algorithme se termine quand il ne reste plus de sommets à traiter

Exemple :

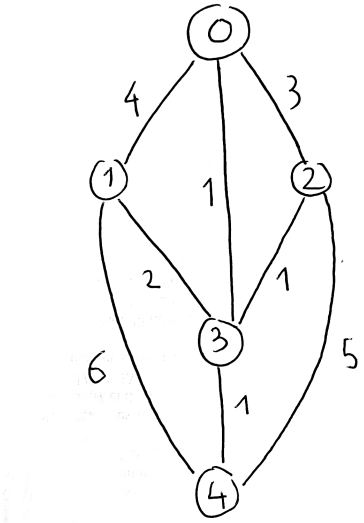
Exercice 2 (Application)**.**
Appliquer l'algorithme sur l'exemple 1

Exercice 3 (Application)**.**
Appliquer l'algorithme sur le graphe de l'exercice 1

Exercice 4 (Application)**.**
Appliquer l'algorithme sur le graphe



Exercice 5 (Application)**.**
Appliquer l'algorithme sur le graphe



II.3 L'implémentation

Exercice 6 (Réflexion)**.**
faire la liste des objets et des fonction dont l'on a besoin pour implémenter cet algorithme

Exercice 7 (fichier)**.**
Télécharger le fichier `dijkstra.py`, il contient des exemples de graphes et la fonction principale. Vous devez compléter ce fichier

Exercice 8 (Complétion)**.**
Compléter la liste d'adjacence `grand` pour quelle représente le graphe de la figure 2 sur la page 4.

Exercice 9 (fonction dminimal)**.**
Compléter la fonction `dminimal` qui prend pour arguments une liste d de flottants, une liste `atraiter` de sommets, et qui rend le sommet s de atraiter tel d[s] est minimal

Exercice 10 (manipulation de listes)**.**
recopier et tester les lignes suivantes

```
L=[ x for x in range(1,10)]
print(L)
L2=[x for x in range(1,10) if x%2==0 ]
print(L2)
L3=[x**2 for x in L2 ]
print(L3)
```

Exercice 11 (Manipulation de listes)**.**
utiliser les méthodes précédentes pour écrire une fonction `pair(L)` qui prend une liste d'entiers L et qui renvoie la lsite des termes de L qui sont pairs. Par exemple

- `pairs([1,2,1,3])` doit renvoyer `[2]`
- `pairs([1,2,1,2,4])` doit renvoyer `[2,2,4]`
- `pairs([1,2,1,2,4])` doit renvoyer `[2,2,4]`

Exercice 12 (Fonction principale)**.**
Compléter la fonction principale

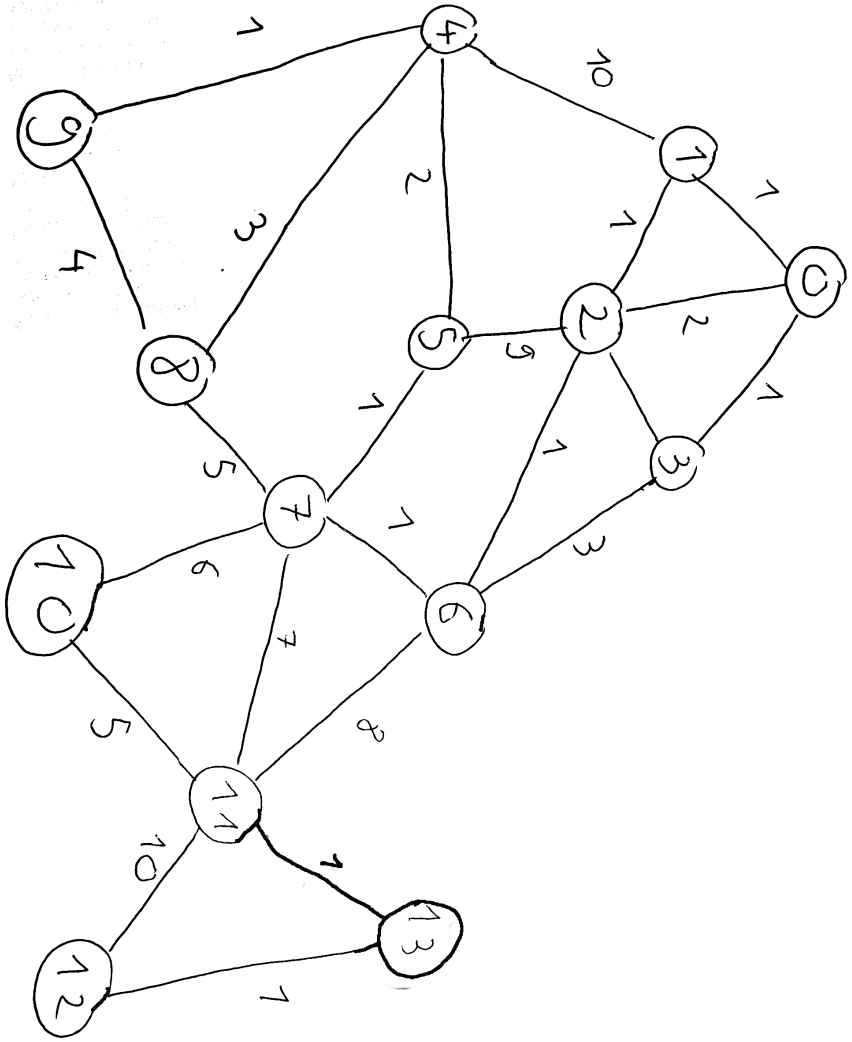


FIGURE 2 – Un graphe un peu grand

III Variations

Exercice 13 (chemins)**.**
Écrire une fonction `def dijkstra_chemin(ladj,s):` en modifiant le code de la fonction `dijkstra` de manière à ce qu'elle rende aussi un tableau `chemin_minimal` tel que, pour tout sommet x du graphe, `chemin_minimal[x]` donne un chemin de s à x de poids d[x] (un chemin sera représenté par la liste de ses sommets dans l'ordre, du début à la fin).