ACT 2 : Trouver le meilleur chemin

1 Les graphes pondérés

Sur une carte autoroutière, on a lu que pour se rendre d'une ville D à une ville A, on peut passer par la ville B ou par la ville C. On se propose de déterminer, à l'aide d'un graphe, le trajet le plus court, ou le moins cher en péage pour aller de D à A.

Voici les distances séparant deux villes par les autoroutes existantes :

- De D à C : 210 km
- De D à B : 420 km
- De C à A : 420 km
- De B à A : 200 km
- De C à B : 105 km

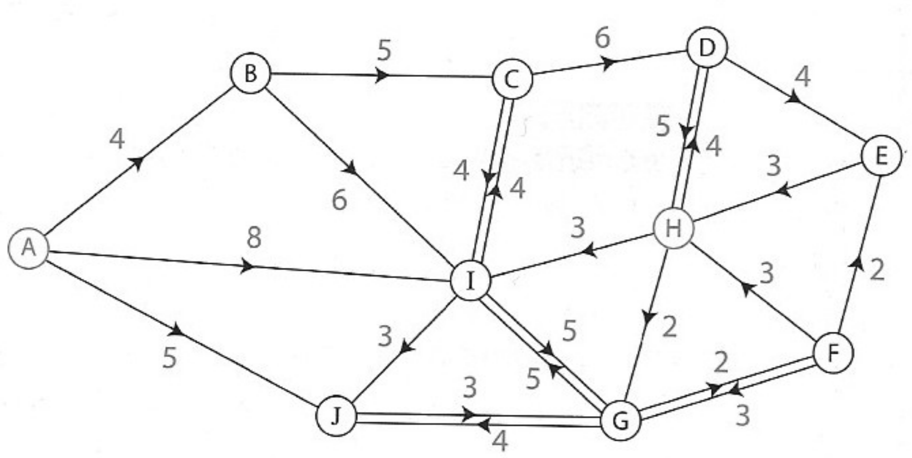
1) Réaliser un graphe étiqueté donc les sommets sont les villes et les étiquettes les distances entre les villes reliées. On dit qu'il s'agit d'un **graphe pondéré**.
2) Déterminer le trajet le plus court pour aller de D à A.
3) Réaliser le graphe pondéré de la situation précédente en remplaçant les distances par les prix des péages, qui sont les suivants :

- De D à C : 20 €
- De D à B : 45 €
- De C à A : 19 €
- De B à A : 40 €
- De C à B : 20 €

4) Quel est le trajet pour lequel la somme dépensée en péage est minimale ?

2. L'algorithme de Dijkstra

Le graphe pondéré ci-dessous indique la durée du trajet en minutes selon les rues empruntées dans un quartier d'une ville. Lorsqu'une ville à double sens rejoint deux lieux, il se peut qu'elle soit plus empruntée dans un sens que dans l'autre, d'où des durées différentes dans chaque sens.



*Le **poids d'une chaîne** est la somme des coûts inscrits sur les arcs qui la composent.*

En assimilant le coût à la durée dans notre exemple, donner le poids de chacune des chaînes ci-dessous :

A – B – J – I – H A – K – I – F – H

Pour déterminer le trajet de poids minimal, appelé plus courte chaîne, on peut utiliser un algorithme très performant : l'**algorithme de Dijkstra**.

Par exemple, pour trouver le trajet de poids minimal menant de A à H :

Étape	Tâches à effectuer
1	• Placer tous les sommets du graphe dans la 1re ligne d'un tableau. • Sur la 2e ligne du tableau, écrire le coefficient 0 sous le sommet de départ et le coefficient ∞ sous les autres sommets.
2	• Sur la dernière ligne écrite, repérer le sommet X de coefficient minimal. • Commencer une nouvelle ligne et rayer toutes les cases vides sous X.
3	• Pour chaque sommet Y adjacent à X, calculer la somme P du coefficient de X et du poids de l'arête reliant Y à X. • Si P est strictement inférieur au coefficient de Y, inscrire P_X dans la case correspondante de la colonne Y. • Sinon, inscrire le coefficient de Y et compléter la ligne par des coefficients de la ligne précédente.
4	• S'il reste des sommets non sélectionnés, recommencer à l'étape 2. • Sinon, passer à l'étape 5.
5	• La longueur minimale est le nombre lu sur la dernière ligne du tableau.

1) Recopier et compléter le tableau ci-dessous en suivant les étapes de l'algorithme.

A	B	C	D	E	F	G	H	I	J
0	∞	∞	∞	∞	∞	∞	∞	∞	∞
|	4_A	∞	∞	∞	∞	∞	∞	8_A	5_A

On pourra s'aider de la vidéo suivante :

[Dijkstra par Yvan Monca (https://www.youtube.com/watch?v=rHylCtXtdNs)](https://www.youtube.com/watch?v=rHylCtXtdNs)

2) En déduire la durée minimale du trajet pour aller de A à H et préciser ce trajet.

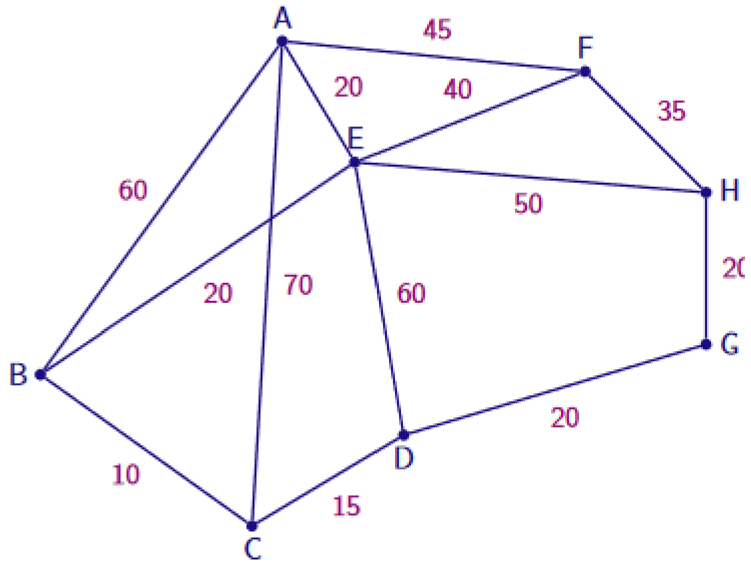
3. Applications

Exercice 1 : courir oui, mais pas trop non plus

Chuck est un sportif adepte du semi-marathon. Il a décidé de courir un semi-marathon. Pour améliorer sa préparation, il décide d'enchaîner les courses pédestres de 10 km dans différentes villes.

Le graphe pondéré ci-dessous représente les villes A, B, C, D, E, F et H organisant des courses de 10 km, et la ville G est celle organisant le prochain semi-marathon auquel Jonathan est inscrit.

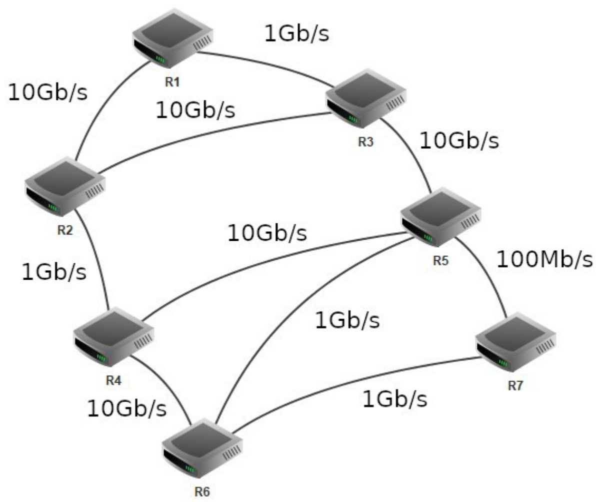
Le poids de chaque arête représente le temps, en minutes, nécessaire pour relier une ville à une autre grâce aux transports en commun.



Chuck vient de courir dans la ville A et souhaite se rendre dans la ville G pour repérer le parcours de son prochain semi-marathon. Déterminer le chemin permettant de relier le plus rapidement la ville A à la ville G, donner la durée de ce parcours en minutes, et les villes traversées.

Exercice 2 : application au protocole OSPF

Soit le réseau suivant :

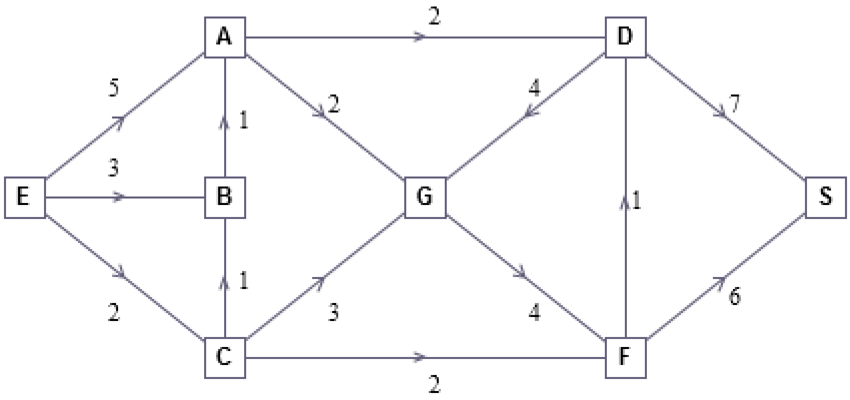


1) Créer un graphe pondéré où les étiquettes entre les sommets sont le coût de chaque liaison.
2) Déterminer à l'aide de l'algorithme de Dijkstra le chemin optimal de R1 à R 7.

Exercice 3 : Création du classe graphe

Pour utiliser l'algorithme de Dijkstra en python, il faut un « outil » permettant de gérer les graphes pondérés.

Soit le graphe dont la représentation est donnée ci-dessous :



1) Compléter le fichier **Classe_Graphe_pondere.py** pour obtenir les méthodes de base sur les graphes pondérés. On doit avoir pour chaque arête, son poids.
2) Créer, à l'aide de la classe précédente, le graphe ci-dessus.
3) Compléter le fichier **Dijkstra.py** afin de déterminer les plus courts chemins d'un sommet s à tous les autres sommets d'un graphe.
4) Le tester (après l'avoir fait à la main) sur le plus court chemin de E à S.
5) Vérifier à l'aide de ce programme les résultats de l'exercice 2 ci-dessus.