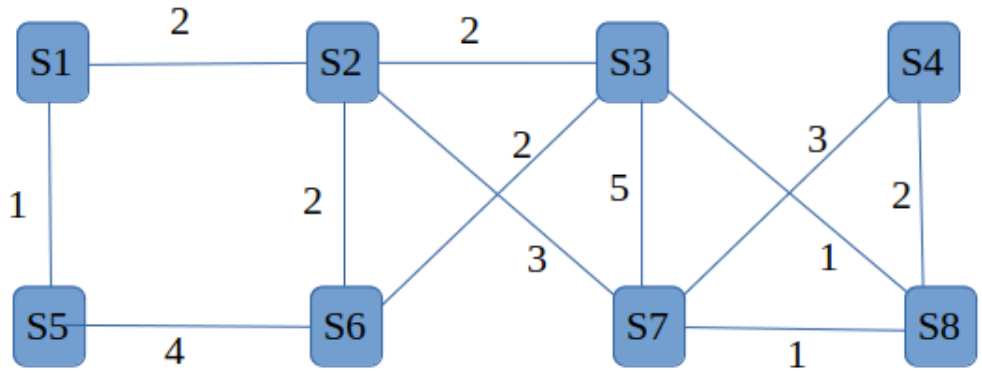
Lycée Jean Moulin **Draguignan**
NSI *Terminale* Année .....

TD - Recherche du plus court chemin

L'algorithme de Dijkstra

Considérons le graphe pondéré suivant :

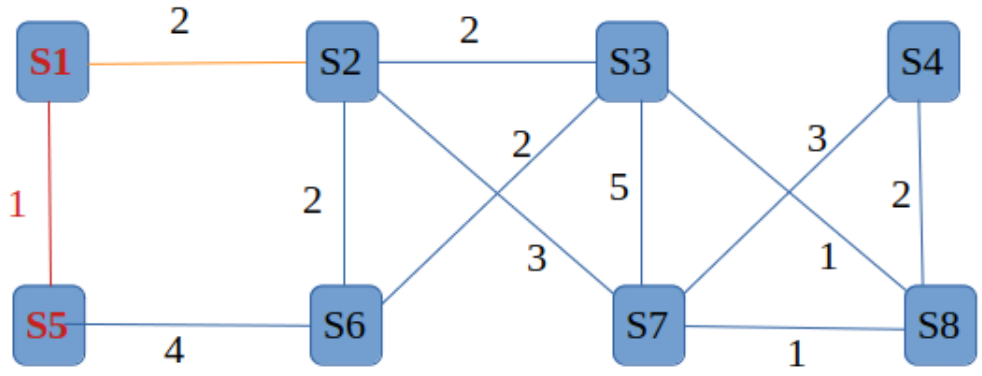


On recherche le plus court chemin entre le sommet S1 et chacun des autres sommets du graphe.
Il s'agit d'un algorithme glouton, où l'on fait un choix optimal à chaque étape.

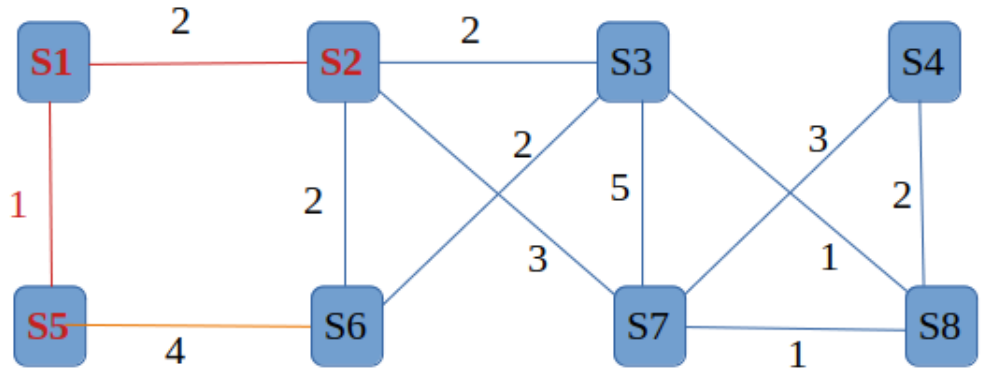
On commence par affecter une valeur très grande (∞) à chacun des sommets du graphe et la valeur 0 au sommet S1 (ces valeurs représenterons les longueurs des chemins entre S1 et les autres sommets du graphe).

Pris : (S1,0)
Non pris : (S2,∞),(S3,∞),(S4,∞),(S5,∞),(S6,∞),(S7,∞),(S8,∞)

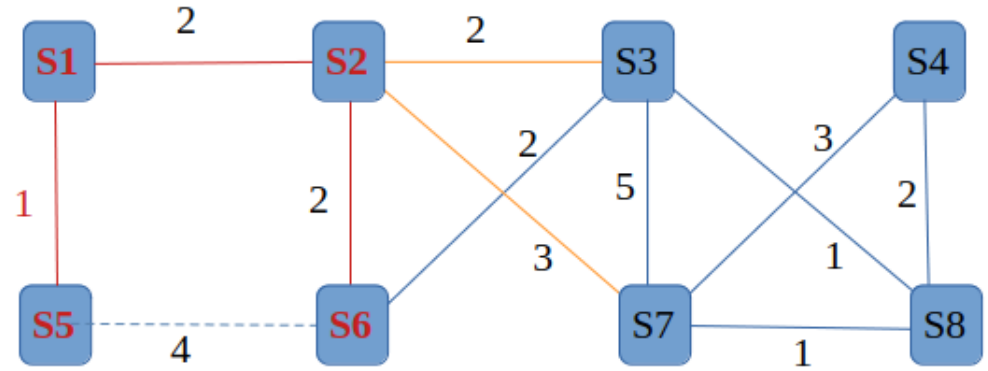
À partir de S1, on peut aller en S2 ou S5 avec les distances respectives 2 et 1. On garde celui correspondant à la valeur minimale S5 et on note la distance pour S2.
Pris : (S1,0),(S5,1)
Non pris : (S2,2),(S3,∞),(S4,∞),(S6,∞),(S7,∞),(S8,∞)



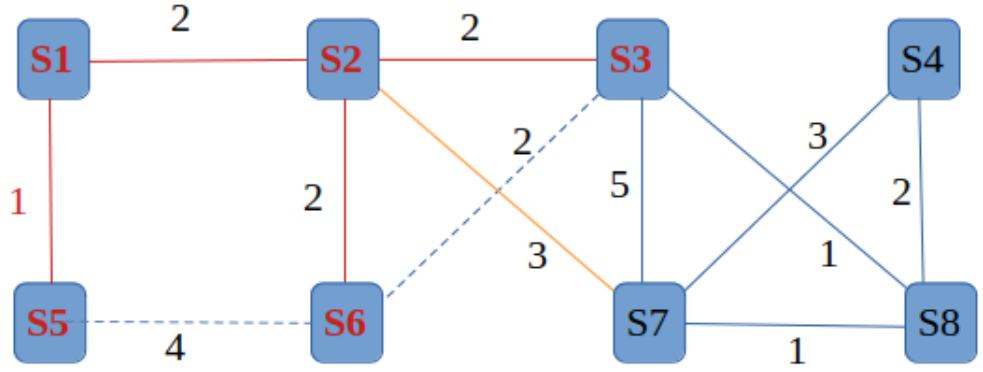
À partir de S5, on peut aller en S6 avec une distance 4, soit une distance totale depuis S1 de 5. On garde le sommet correspondant à la valeur minimale, soit S2
Pris : (S1,0),(S5,1),(S2,2)
Non pris : (S3,∞),(S4,∞),(S6,5),(S7,∞),(S8,∞)



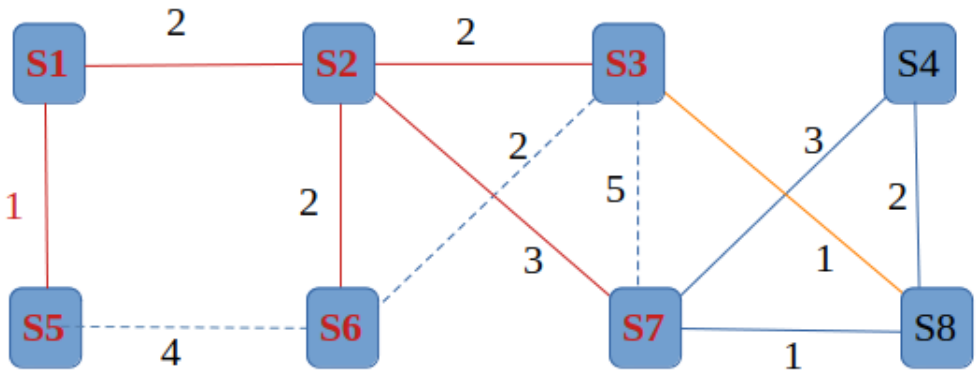
À partir de S2, on peut aller en S3, S6 et S7 avec les distances respectives 2,2 et 3 donc des distances totales respectives de 4,4 et 5. On garde le sommet non pris correspondant à la valeur minimale soit S6 (on aurait pu garder S3)
Pris : (S1,0),(S5,1),(S2,2),(S6,4)
Non pris : (S3,4),(S4,∞),(S7,5),(S8,∞)



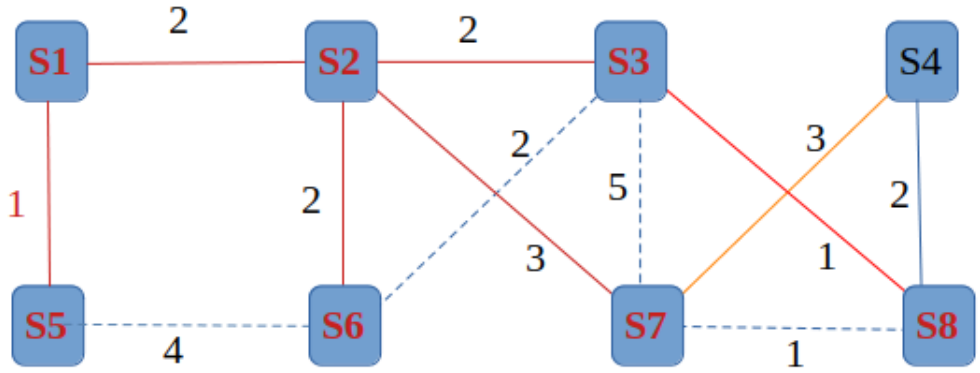
À partir de S6, on peut aller en S3 avec une distance totale de 6. On garde le sommet non pris correspondant à la valeur minimale soit S3.
Pris : (S1,0),(S5,1),(S2,2),(S6,4),(S3,4)
Non pris : (S4,∞),(S7,5),(S8,∞)



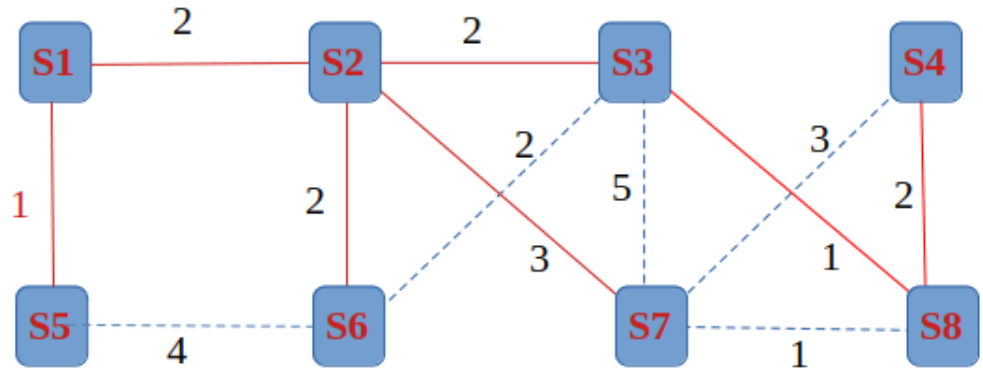
À partir de S3, on peut aller en S7 ou S8 avec des distances totales respectives de 9 et 5 (ce qui n'améliore pas la distance totale Pour S7. On garde donc S7 (on aurai pu choisir S8)
Pris : (S1,0),(S5,1),(S2,2),(S6,4),(S3,4),(S7,5)
Non pris : (S4,∞),(S8,5)



À partir de S7 on peut aller en S4 ou S8 avec des distances totales respectives de 8 et 6. Ce qui nous donne (la situation de S8 ne s'est pas améliorée) :
Pris : (S1,0),(S5,1),(S2,2),(S6,4),(S3,4),(S7,5), (S8,5)
Non pris : (S4,8)



À partir de S8 on peut aller en S4 avec une distance totale de 7 depuis S1, ce qui n'améliore pas sa situation. On a donc :
Pris : (S1,0),(S5,1),(S2,2),(S6,4),(S3,4),(S7,5), (S8,5),(S4,7)
Non pris :



À faire 1:
Représenter l'arbre enraciné en S1 ainsi obtenu.

Remarque :
On peut estimer le coût de cet algorithme comme étant de l'ordre de n^2, une étape par sommet, soit n étapes, et dans le pire des cas n opérations par étape pour calculer les distances avec les autres sommets.

Et si on faisait un programme...

Représentation du graphe
On représentera le graphe par un dictionnaire dont les clés seront les sommets et les valeurs seront des listes de couples (sommet adjacent à la clé, distance à la clé).

Nous aurons besoin d'une fonction `minimum(dico)` qui détermine et renvoie la clé correspondant au minimum des valeurs d'un dictionnaire.

À faire 2:
Écrire l'algorithme et son implémentation en Python de cette fonction. Il y a plusieurs façons de faire...

```
from math import inf # permet d'utiliser l'infini

def minimum(dico):




_
```

La fonction dijkstra:

À faire 3:

Voici la fonction dijkstra qu'il faudra commenter.

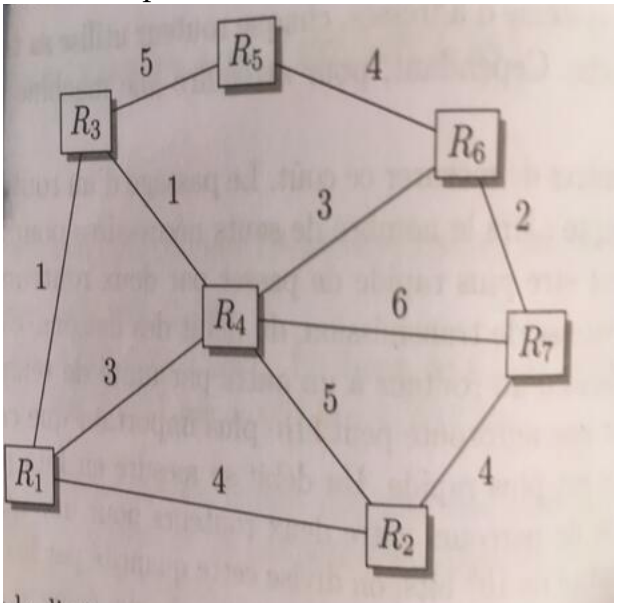
```
def dijkstra(G,s):
    D={}
    d={k:inf for k in G}
    d[s]=0
    while len(d)>0:
        k=minimum(d)
        for j in range(len(G[k])):
            v,c=G[k][j]
            if v not in D:
                d[v]=min(d[v],d[k]+c)
        D[k]=d[k]
        del d[k]
    return D
```

À faire 4:

Vérifier que vous obtenez bien :
{'S1': 0, 'S5': 1, 'S2': 2, 'S3': 4, 'S6': 4, 'S7': 5, 'S8': 5, 'S4': 7}

À faire 5:

Voici une partie d'un réseau constitué de 7 routeurs.



Déterminer "à la main" les longueurs des plus courts chemins entre le routeur R1 et les autres routeurs. Vérifier votre résultat avec le programme précédent.

.....

.....

.....

Prolongement possible

Rechercher les informations nécessaires pour implémenter l'algorithme de Bellman-Ford et l'appliquer aux graphes précédents.