1 Problématique

Les protocoles de routage (RIP, OSPF) calculent les plus courts chemins entre les routeurs. Les algorithmes utilisés ont été découverts à la fin des années 60 et sont utilisés dans de nombreuses situations.

Comment fonctionnent les algorithmes de plus court chemin ?

2 Retour des graphes

2.1 Graphe orienté

Un graphe **orienté** possède les mêmes caractéristiques que celles déjà évoquées pour un graphe non orienté, mais dont les arcs ne peuvent être parcourus que dans un sens. On peut ainsi définir un graphe par *une matrice d'adjacence* ou *un dictionnaire d'adjacence*.

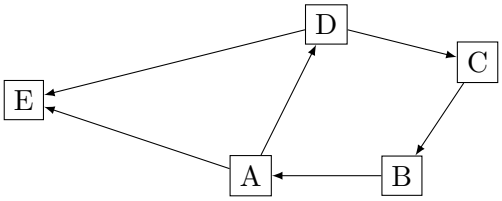

FIGURE 1 – graphe orienté

À retenir

Pour chaque nœud on peut définir ses **prédécesseurs** et ses **successeurs**.

- Le nœud A ne possède pas de *prédécesseur*.
- Le nœud E ne possède pas de *successeur*.

Activité 1 :

1. Établir la matrice d'adjacence du graphe figure 1.
2. Établir le dictionnaire d'adjacence du graphe figure 1.
3. Déterminer un cycle dans le graphe.

2.2 Graphe pondéré

Les arcs d'un graphe **pondéré** possèdent une *valeur ou poids*. Cette valeur peut être négative.

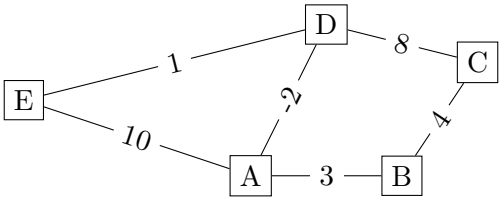

FIGURE 2 – graphe non orienté pondéré

Activité 2 :

Christophe Viroulaud

1. Établir la matrice d'adjacence du graphe figure 2.
2. Établir le dictionnaire d'adjacence du graphe figure 2.

3 Protocole RIP : algorithme de Bellman-Ford

3.1 Principe

Publié entre 1956 et 1958, cet algorithme calcule les plus courts chemins entre un *nœud source* et tous les autres nœuds d'un graphe orienté et pondéré. Le principe est le suivant :

La distance pour atteindre chaque nœud correspond à la distance pour atteindre son prédécesseur à laquelle on ajoute le poids de l'arête les séparant.

La démarche est récursive : on applique le même principe au prédécesseur. Enfin un nœud pouvant avoir plusieurs prédécesseurs, on gardera le chemin le plus court.

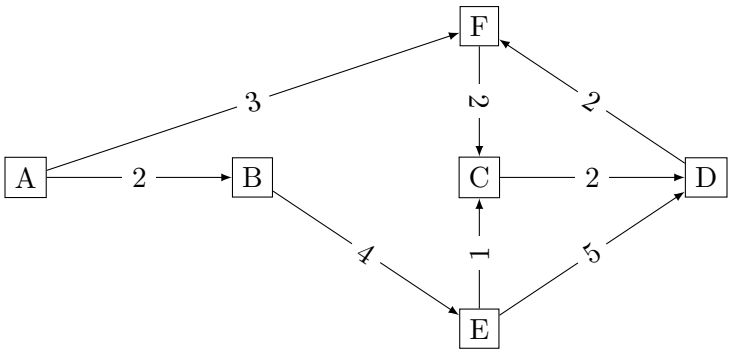

FIGURE 3 – graphe non orienté pondéré

Remarque

Dans le graphe figure 3 les pondérations représentent un nombre de routeurs traversés pour atteindre le nœud (routeur) suivant.

On obtient un tableau contenant la distance minimale entre le routeur de départ et chaque autre routeur.

3.2 Mise en application

Chaque routeur exécute l'algorithme pour calculer les plus courts chemins vers chaque autre routeur du réseau. L'algorithme détaillé pour le routeur A est le suivant :

```
1 Créer un tableau des distances entre A et les routeurs (A inclus),
     initialisées àl'infini.
2 Dans le tableau modifier la distance vers A à0.
3
4 Tant que (le nombre d'itérations) < (nombre de routeurs)
5     Pour chaque arc du graphe
6         Si (la distance du routeur) > (la distance de son prédécesseur +
             poids de l'arc entre les deux routeurs)
7             La distance du routeur est remplacée par cette nouvelle valeur
```


Christophe Viroulaud

Code 1 – Algorithme de Bellman Ford

Pour le graphe figure 3, le tableau est initialisé comme suit :

A	B	C	D	E	F
0	∞	∞	∞	∞	∞

On effectue une première itération sur les arcs.
La distance calculée pour atteindre B correspond à la distance (du tableau) de son prédécesseur (A) augmentée du poids de l'arc A-B :

$$0 + 2 = 2 < \infty$$

Le tableau devient :

A	B	C	D	E	F
0	**2 (A)**	∞	∞	∞	∞

Activité 3 : Continuer de dérouler l'algorithme sur le graphe figure 3.

3.3 Complexité

La complexité de l'algorithme (code 1) dépend :

— du nombre de sommets (notée S) : on visite chaque sommet (ligne 4) ;

— du nombre d'arcs (notée A) : pour chaque sommet on regarde tous les arcs du graphe (ligne 5).

$$O(S.A)$$

4 OSPF : algorithme de Dijkstra

4.1 Principe

Il a été publié en 1959 par le néerlandais Edsger Dijkstra. Le principe est de construire un sous-graphe en ajoutant à chaque itération un sommet de distance minimale.
Là encore on obtient un tableau contenant la distance minimale entre le routeur de départ et chaque autre routeur.

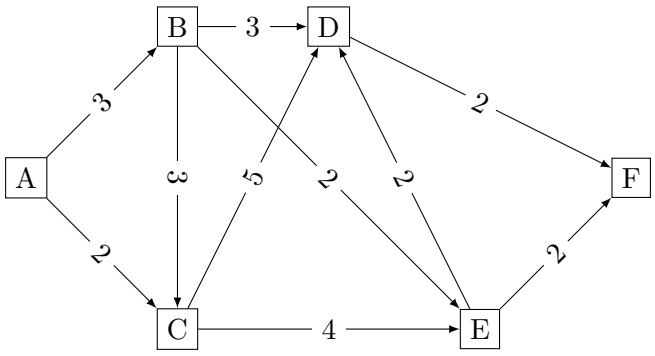

FIGURE 4 – graphe orienté et pondéré

Remarque

Les poids de chaque arc représentent le coût de chaque connexion.

Christophe Viroulaud

4.2 Mise en application

L'algorithme maintient également un tableau des distances du routeur de départ, vers tous les autres.

```
1 Créer un tableau des distances entre A et les routeurs (A inclus),
     initialisées àl'infini.
2 Dans le tableau modifier la distance vers A à0.
3
4 Tant qu'il reste des routeurs non sélectionnés
5     Parmi les routeurs non-sélectionnés, choisir le routeur (noté S)
         ayant la plus petite distance.
6     Pour chaque routeur adjacent àS (noté V) et non déjà sélectionné:
7         Si (la distance de V) > (la distance de S + poids S-V)
8             La distance de V est remplacée par cette nouvelle valeur
```

Code 2 – Algorithme de Dijkstra

Pour le graphe figure 4, le tableau est initialisé comme suit :

A	B	C	D	E	F
0	∞	∞	∞	∞	∞

Le routeur A possède la plus petite distance : il est sélectionné. La distance à ses voisins est mise à jour.

A	B	C	D	E	F
0	3	2	∞	∞	∞

nœuds visités
A

Activité 4 : Continuer de dérouler l'algorithme sur le graphe figure 4.

4.3 Complexité

Encore une fois la complexité dépend du nombre de sommets S et du nombre d'arcs A. Cependant le point clé de l'algorithme tient dans la recherche de la distance minimale (ligne 5).

$$O((A+S)\times \log S)$$

Christophe Viroulaud