IV

Dictionnaires 4 : Distance minimale entre les sommets d'un graphe orienté et pondéré - l'algorithme de Roy-Floyd-Warshall

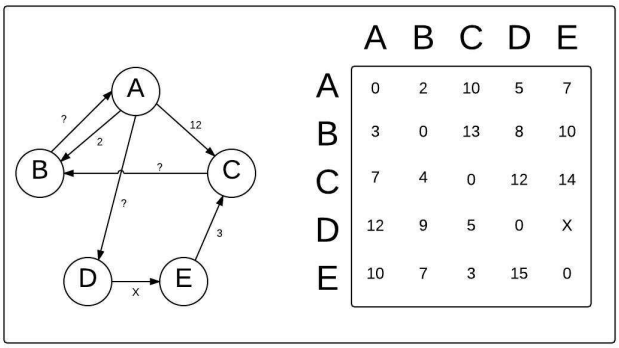


FIGURE IV.1 – Algorithme de Roy-Floyd-Warshall (1959)

En informatique, l'algorithme de Floyd-Warshall est un algorithme pour déterminer les distances des plus courts chemins entre toutes les paires de sommets dans un graphe orienté et pondéré, en temps cubique au nombre de sommets. Il est parfois appelé algorithme de Roy-Floyd-Warshall car il a été décrit par Bernard Roy en 1959 avant les articles de Floyd et Warshall datant de 1962. C'est à ce jour le plus performant. (Source : Wikipedia*/)*

PLAN DU CHAPITRE

Le cours de première année aborde la recherche de plus courts chemins dans un graphe orienté et pondéré par l'algorithme de Dijkstra (1959).
Nous rechercherons ici une méthode s'appuyant sur la programmation dynamique et les dictionnaires et permettant d'une part de donner le poids du plus court chemin **entre toutes les paires de sommets** (i, j) **du graphe** G, et d'autres part **d'exhiber ces plus courts chemins**.

Réponse :

I Notations et remarques préliminaires

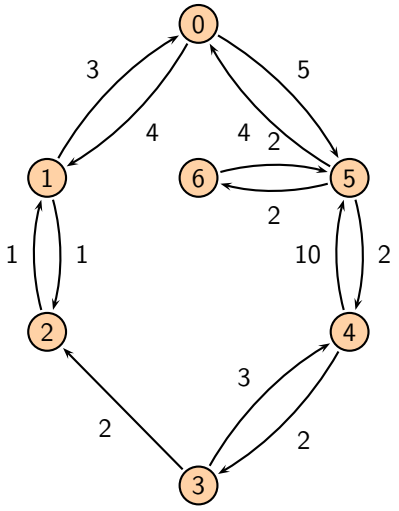
On notera $G = (V, A)$ le graphe orienté et pondéré de sommets (*Vertice*) $V = \{0, 1, \ldots, n-1\}$ et d'arcs $A = \{(i_0, j_0, w_{i_0,j_0}), (i_1, j_1, w_{i_1,j_1}), \ldots\}$.

Le graphe peut être défini par sa matrice d'adjacence $E_G = (e_{ij})$ telle que :

$$\left[\begin{array}{ll} e_{ij} = w_{ij} & \text{si } (i, j, w_{ij}) \in A \\ e_{ij} = 0 & \text{si } i = j \\ e_{ij} = \star & \text{sinon} \end{array}\right.$$

avec $\star$ qui peut être tout sauf un poids du graphe (on peut prendre la somme des poids+n ($n \in \mathbb{N}^*$), l'objet NAN, l'∞ etc... en tout cas quelque chose de repérable !)

On peut par exemple rappeler le graphe exploré en détail dans le cours de révisions sur l'algorithme de Dijkstra et sa matrice d'adjacence E_G :



$$E_G = \begin{pmatrix} 0 & 4 & \star & \star & \star & 5 & \star \\ 3 & 0 & 1 & \star & \star & \star & \star \\ \star & 1 & 0 & \star & \star & \star & \star \\ \star & \star & 2 & 0 & 3 & \star & \star \\ \star & \star & \star & 2 & 0 & 10 & \star \\ 4 & \star & \star & \star & 2 & 0 & 2 \\ \star & \star & \star & \star & \star & 2 & 0 \end{pmatrix}$$

Toujours dans l'idée d'assurer une efficacité maximale dans les primitives de type (rechercher, lire, insérer, supprimer : *cf chapitre Dictionnaire 1*), il est également possible de représenter les arcs du graphe par un dictionnaire d'adjacence dont les clés sont les origines i des arcs, et les valeurs les listes de tuples constitués du nœud extrémité j de l'arc et du poids correspondant à cet arc soit (j, w_{ij}) :

$$d[i] = [(j_0, w_{i,j_0}), (j_1, w_{i,j_1}), \ldots]$$

Exercice de cours: **(I.0) - n° 1.** *Rédiger le code d'implémentation du dictionnaire* `d` *correspondant au graphe ci-dessus.*

Réponse :

Exercice de cours: **(I.0) - n° 2.** *On considère le graphe $G = (V, A)$ orienté et pondéré, dont certains arcs peuvent avoir des poids négatifs.*

1. *Montrer par l'absurde que si le graphe présente un chemin de poids minimal entre deux de ses sommets quelconques, alors il ne peut posséder de circuit*[1] *de poids strictement négatif.*
2. *Montrer que si $p = \big((i, i_1, w_{i,i_1}), \ldots, (i_{l-1}, j, w_{i_{l-1},j})\big)$ est un chemin de poids minimal entre les sommets i et j, alors pour tout sommet intermédiaire k par lequel passe le chemin p, les chemins partiels de i à k et de k à j sont eux-mêmes minimaux.*

Réponse :

II Formulation du problème (Bottom-Up) et sous-structure optimale

Important : pour toute la suite, on notera $\mathcal{C}_{ij}$ l'ensemble des chemins de i à j.

On reprend toujours ici le graphe G de sommets $V = \{0, 1, \ldots, n-1\}$.

Posons la suite $(\mathcal{C}_{i,j,k})_{k\geq 1} \in \mathcal{C}_{i,j}$ où chaque $\mathcal{C}_{i,j,k}$ est formé par les chemins de i à j passant par les sommets intermédiaires, s'ils existent, présents dans $[\![0, k-1]\!]$; on remarquera par conséquent que $\mathcal{C}_{i,j,n} = \mathcal{C}_{i,j}$ (tous les chemins entre i et j) ; on notera $\mathcal{W}^k$ la matrice des $\mathcal{W}^k_{i,j}$ avec $\mathcal{W}^k_{i,j}$ **le poids minimal** d'un chemin de i à j appartenant à $\mathcal{C}_{i,j,k}$

On notera $\mathcal{C}_{i,j,0}$ l'ensemble des arcs reliant i à j (peut être $\varnothing$).

Exercice de cours: **(II.0) - n° 3.** *D'après ces notations, que représentent $\mathcal{W}^0$?*

Réponse :

Éligibilité à la programmation dynamique :

Comme pour le calcul de la distance de Levenshtein, il faut s'assurer ici que le problème possède la propriété de sous-structure optimale :

> Si p est un chemin de poids minimal $\mathcal{W}^k_{i,j}$ entre les sommets i et j dans le sous-ensemble $\mathcal{C}_{i,j,k}$ ($p \in \mathcal{C}_{i,j,k}$), alors deux situations peuvent se présenter :
>
> - **soit p n'emprunte pas le sommet $k-1$, et dans ces conditions il est aussi un chemin de poids minimum $\mathcal{W}^{k-1}_{i,j}$ dans $\mathcal{C}_{i,j,k-1}$ ($p \in \mathcal{C}_{i,j,k-1}$).**

1. Dans un graphe orienté, on appelle circuit une suite d'arcs consécutifs (chemin) dont les deux sommets extrémités sont identiques

- **soit p emprunte exactement une fois le sommet $k-1$ et alors p est la concaténation p_1+p_2 des chemins p_1 de i à $k-1$ et p_2 de $k-1$ à j dont les sommets intermédiaires sont dans $[\![0,k-2]\!]$ avec $p_1 \in \mathcal{C}_{i,k-1,k-1}$ et $p_2 \in \mathcal{C}_{k-1,j,k-1}$. On a alors $\mathcal{W}^k_{i,j} = \mathcal{W}^{k-1}_{i,k-1} + \mathcal{W}^{k-1}_{k-1,j}$.**

  CONCLUSION : le problème présente donc bien une sous-structure optimale.

Exercice de cours: **(II.0) - n° 4.** *A partir du constat ci-dessus, donner la relation de récurrence permettant de calculer la solution, c'est à dire les poids minimaux $\mathcal{W}^k_{i,j}$ par valeur croissante de k.*

RÉPONSE :

III Codes

A partir de la récurrence dégagée ci-dessus, on implémente sans peine l'algorithme avec la fonction `Floyd_Warshall(E:ndarray)` $\longrightarrow$ *`ndarray`* qui renvoie la matrice W des points minimaux. Le principe repose sur le calcul itératif de la matrice des poids minimaux W en intégrant à chaque itération un sommet supplémentaire parmi les sommets intermédiaires pouvant être visités. Cela donne donc :

Listing IV.1 –

```
def Floyd_Warshall(matadj):
  n=matadj.shape[0]
  W=matadj.copy()
  for k in range(1,n+1): #démarrage en intégrant le noeud 0 et fin avec le noeud n-1 (k=n donc noeud k-1=n-1)
    for i in range(n):
      for j in range(n):
        W[i,j]=min(W[i,j],W[i,k-1]+W[k-1,j])
  return W
```

Exercice de cours: **(III.0) - n° 5.** *Modifier la fonction précédente afin qu'elle reçoive en argument, non plus la matrice d'adjacence E, mais le dictionnaire d'adjacence d du graphe G, soit* `Floyd_Warshall(d:dict)` $\longrightarrow$ *`ndarray`* *:*

RÉPONSE :

Ci-dessous, la sortie obtenue pour la fonction `Floyd_Warshall(matadj)` en utilisant le graphe présenté en **II.2.a)** de matrice d'adjacence `matadj` :

```
>>> print(Floyd_Warshall(matadj))
[[ 0  4  5  9  7  5  7]
 [ 3  0  1 12 10  8 10]
 [ 4  1  0 13 11  9 11]
 [ 6  3  2  0  3 11 13]
 [ 8  5  4  2  0 10 12]
 [ 2  6  6  4  2  0  2]
 [ 4  8  8  6  4  2  0]]
```

IV Reconstruction des plus courts chemins

Il est possible de reconstruire par récurrence les plus courts chemins correspondants aux poids calculés à chaque étape de l'algorithme de Floyd-Warshall. L'algorithme va pour cela calculer une série de matrices carrées (dites *matrices de liaison*) $\Pi^{(0)}, \Pi^{(1)}, \ldots, \Pi^{(n)}$ où $\Pi^{(k)} = (\pi_{ij})^{(k)}$ avec $\pi_{ij}^{(k)}$ désignant le prédécesseur du sommet j sur un plus court chemin partant du sommet i et dont tous les sommets intermédiaires sont dans $\{0, 1, \ldots, k-1\}$.

On notera qu'avant toute itération, on a :

$$\pi_{i,j}^{(0)} = \begin{cases} NIL & \text{if } i = j \text{ ou } w_{i,j} = \infty \\ i & \text{if } i \neq j \text{ et } w_{i,j} \text{ fini} \end{cases}$$

Compte tenu de l'étude menée sur les matrices $\mathcal{W}$, la récurrence permettant la construction des matrices Π avec $k-1 \geq 0$ est assez évidente :

- si l'on emprunte le chemin $i \ldots k-1 \ldots j$ où $k-1 \neq j$, alors le prédécesseur de j à choisir sera le même que celui choisi sur le plus court chemin partant de $k-1$ dont tous les sommets intermédiaires se trouvent dans $\{0, 1, 2, \ldots, k-2\}$.
- sinon (i.e. si l'on n'emprunte pas $i \ldots k-1 \ldots j$), on prend le même prédécesseur de j que celui choisi sur le plus court chemin partant de i dont tous les sommets intermédiaires se trouvent dans $\{0, 1, \ldots, k-2\}$

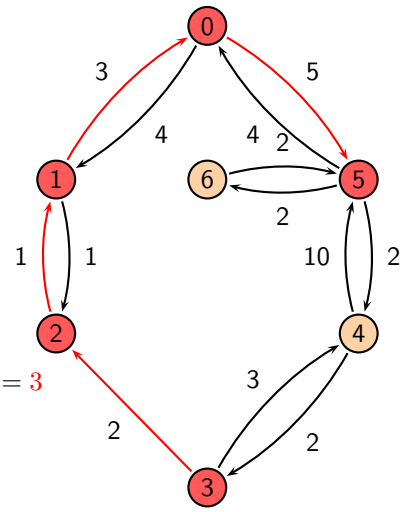
On en déduit la récurrence suivante pour $k - 1 \geq 0$:

$$\pi_{ij}^{(k)} = \begin{cases} \pi_{k-1,j}^{(k-1)} & \text{si } \mathcal{W}_{i,j}^{k-1} > \mathcal{W}_{i,k-1}^{k-1} + \mathcal{W}_{k-1,j}^{k-1} \\ \pi_{i,j}^{(k-1)} & \text{si } \mathcal{W}_{i,j}^{k-1} \leq \mathcal{W}_{i,k-1}^{k-1} + \mathcal{W}_{k-1,j}^{k-1} \end{cases}$$

Exercice de cours: **(IV.0) - n° 6.** *Modifier le code de la fonction* `Floyd_Warshall(matadj)` *afin qu'elle renvoie, en plus de la matrice des poids des plus courts chemins $\mathcal{W}$, la matrice de liaison finale Π permettant la reconstitution de tous les plus courts chemins entre deux noeuds du graphe.*

RÉPONSE :

PRINCIPE DE LECTURE DE LA MATRICE DE LIAISON Π :
sur chaque ligne i, on peut lire tous les plus courts chemins entre le noeud i et chaque noeud j du graphe ; par exemple, le plus court chemin du noeud 3 au noeud 5 se lit, à partir du noeud d'arrivée 5, et donne :



$5 \overset{\text{a pour prédécesseur}}{\underset{\text{prédécesseur de}}{\longleftarrow}} \pi_{3,5} = 0 \overset{\text{a pour prédécesseur}}{\underset{\text{prédécesseur de}}{\longleftarrow}} \pi_{3,0} = 1 \overset{\text{a pour prédécesseur}}{\underset{\text{prédécesseur de}}{\longleftarrow}} \pi_{3,1} = 2 \overset{\text{a pour prédécesseur}}{\underset{\text{prédécesseur de}}{\longleftarrow}} \pi_{3,2} = 3$