TD d'algorithmique avancée

Corrigé du TD 11 : Plus courts chemins pour tout couple de sommets

Jean-Michel Dischler et Frédéric Vivien

Nous nous intéressons ici à la recherche des plus courts chemins entre tous les couples de sommets d'un graphe (typiquement on cherche à élaborer la table des distances entre tous les couples de villes d'un atlas routier). On dispose en entrée d'un graphe orienté $G = (S, A)$ et d'une fonction de pondération w. Nous supposons que le graphe G peut contenir des arcs de poids négatifs mais pas des circuits de poids strictement négatifs. On note n le nombre de sommets de G ($n = |S|$), et on note $\{1, 2, ..., n\}$ les n sommets de G.

Programmation dynamique naïve

Comme nous l'avons remarqué en cours, tout sous-chemin d'un plus court chemin est lui-même un plus court chemin et le problème des plus courts chemins vérifie bien la propriété de sous-structure optimale : nous pouvons donc essayer de le résoudre par programmation dynamique.

1. On note $d_{i,j}^{(m)}$ le poids minimal d'un chemin d'*au plus* m arcs du sommet i au sommet j. Définissez récursivement $d_{i,j}^{(m)}$ (la récursion portera ici sur le nombre d'arcs d'un plus court chemin).
   Pour $m = 0$ il existe un plus court chemin sans arc de i vers j si et seulement si $i = j$:

   $$d_{i,j}^{(0)} = \begin{cases} 0 & \textit{si } i = j, \\ \infty & \textit{sinon.} \end{cases}$$

   Pour $m \geq 1$, $d_{i,j}^{(m)}$ est la longueur du plus court chemin de i à j contenant au plus m arcs. Soit un tel plus court chemin contient exactement m arcs et il est obtenu par concaténation d'un plus court chemin d'au plus $m-1$ arcs de i à un sommet k et de l'arc de k à j, soit il n'en contient au plus que $m-1$ et sa longueur est égale à $d_{i,j}^{(m-1)}$. Par conséquent :

   $$d_{i,j}^{(m)} = \min\left(d_{i,j}^{(m-1)}, \min_{1\leq k\leq n}\left\{d_{i,k}^{(m-1)} + w_{k,j}\right\}\right) = \min_{1\leq k\leq n}\left\{d_{i,k}^{(m-1)} + w_{k,j}\right\},$$

   la formule étant simplifiée grâce à la propriété : $w_{j,j} = 0$.
2. Construisez, au moyen de la formule de récurrence obtenue à la question précédente, un algorithme de calcul des longueurs des plus courts chemins pour tout couple de sommets, algorithme basé sur le paradigme de la programmation dynamique.
   On note $W = (w_{i,j})_{1\leq i,j\leq n}$ la matrice des poids et $D^{(m)} = \left(d_{i,j}^{(m)}\right)_{1\leq i,j\leq n}$ la matrice des poids des plus courts chemins contenant au plus m arcs. Le calcul de $D^{(m)}$ à partir de $D^{(m-1)}$ et de W se fait au moyen de l'algorithme ci-dessous :

   EXTENSION-PLUS-COURTS-CHEMINS(D, W)
   - $n \leftarrow$ *lignes*(D)
   - soit $D' = (d'_{i,j})_{1\leq i,j\leq n}$ une matrice carrée de taille n
   - **pour** $i \leftarrow 1$ **à** n **faire**
     - **pour** $j \leftarrow 1$ **à** n **faire**

$d'_{i,j} \leftarrow +\infty$
pour $k \leftarrow 1$ **à** n **faire**
$d'_{i,j} \leftarrow \min(d'_{i,j}, d_{i,k} + w_{k,j})$
renvoyer D'

À partir de cet algorithme, la résolution du problème est triviale :

PLUS-COURTS-CHEMINS(W)
$n \leftarrow$ *lignes*(W)
$D^{(1)} \leftarrow W$
pour $m \leftarrow 2$ **à** $n-1$ **faire**
$D^{(m)} \leftarrow$ EXTENSION-PLUS-COURTS-CHEMINS$(D^{(m-1)}, W)$
renvoyer $D^{(n-1)}$

3. Quelle est la complexité de votre algorithme ?
L'algorithme EXTENSION-PLUS-COURTS-CHEMINS *s'exécute en* $\Theta(n^3)$*, à cause des trois boucles imbriquées. Le coût total de résolution est donc en* $\Theta(n^4)$.
4. Exécutez votre algorithme sur le graphe de la figure 1.

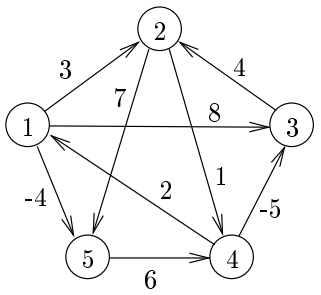

FIG. 1 – Exemple de graphe orienté.

La figure 2 présente un exemple d'exécution de cet algorithme.

$$D^{(1)} = \begin{pmatrix} 0 & 3 & 8 & \infty & -4 \\ \infty & 0 & \infty & 1 & 7 \\ \infty & 4 & 0 & \infty & \infty \\ 2 & \infty & -5 & 0 & \infty \\ \infty & \infty & \infty & 6 & 0 \end{pmatrix} \qquad D^{(2)} = \begin{pmatrix} 0 & 3 & 8 & 2 & -4 \\ 3 & 0 & -4 & 1 & 7 \\ \infty & 4 & 0 & 5 & 11 \\ 2 & -1 & -5 & 0 & -2 \\ 8 & \infty & 1 & 6 & 0 \end{pmatrix}$$

$$D^{(3)} = \begin{pmatrix} 0 & 3 & -3 & 2 & -4 \\ 3 & 0 & -4 & 1 & -1 \\ 7 & 4 & 0 & 5 & 11 \\ 2 & -1 & -5 & 0 & -2 \\ 8 & 5 & 1 & 6 & 0 \end{pmatrix} \qquad D^{(4)} = \begin{pmatrix} 0 & 1 & -3 & 2 & -4 \\ 3 & 0 & -4 & 1 & -1 \\ 7 & 4 & 0 & 5 & 3 \\ 2 & -1 & -5 & 0 & -2 \\ 8 & 5 & 1 & 6 & 0 \end{pmatrix}$$

FIG. 2 – Séquence des matrices calculées par PLUS-COURTS-CHEMINS.

Algorithme de Floyd-Warshall

L'algorithme de Floyd-Warshall est un autre algorithme conçu suivant le principe de la programmation dynamique.

1. Ici la récursion n'a pas lieu sur le nombre d'arcs d'un plus court chemin, mais sur les sommets intermédiaires de ces chemins, un sommet intermédiaire étant un sommet autre que les extrémités du chemin. Ici, $d_{i,j}^{(k)}$ est la longueur du plus court chemin de i à j n'utilisant comme sommets intermédiaires que des sommets parmi $\{1, 2, ..., k\}$.

   Définissez $d_{i,j}^{(k)}$ de manière récursive.

   De deux choses l'une, un plus court chemin de i à j n'ayant comme sommets intermédiaires que des sommets de $\{1, 2, .., k\}$ contient ou ne contient pas le sommet k :

   (a) *Si le plus court chemin p de i à j et n'ayant comme sommets intermédiaires que des sommets de $\{1, 2, .., k\}$ a effectivement comme sommet intermédiaire k, alors p est de la forme $i \overset{p_1}{\rightsquigarrow} k \overset{p_2}{\rightsquigarrow} j$ où p_1 (resp. p_2) est un plus court chemin de i à k (resp. de k à j) n'ayant comme sommets intermédiaires que des sommets de $\{1, 2, ..., k-1\}$.*

   (b) *Si le plus court chemin p de i à j et n'ayant comme sommets intermédiaires que des sommets de $\{1, 2, .., k\}$ ne contient pas k, alors c'est un plus court chemin p de i à j et n'ayant comme sommets intermédiaires que des sommets de $\{1, 2, .., k-1\}$.*

   La structure explicitée ci-dessus nous donne directement une récursion définissant $d_{i,j}^{(k)}$:

$$d_{i,j}^{(k)} = \begin{cases} w_{i,j} & \textit{si } k = 0, \\ \min(d_{i,j}^{(k-1)}, d_{i,k}^{(k-1)} + d_{k,j}^{(k-1)}) & \textit{sinon.} \end{cases}$$

2. Proposez un algorithme de calcul de la longueur des plus courts chemins basé sur la récursion précédente et le paradigme de la programmation dynamique.

   FLOYD-WARSHALL(W)
   - $n \leftarrow$ *lignes*(W)
   - $D^{(0)} \leftarrow W$
   - **pour** $k \leftarrow 1$ **à** n **faire**
     - **pour** $i \leftarrow 1$ **à** n **faire**
       - **pour** $j \leftarrow 1$ **à** n **faire**
         - $d_{i,j}^{(k)} \leftarrow \min(d_{i,j}^{(k-1)}, d_{i,k}^{(k-1)} + d_{k,j}^{(k-1)})$
   - **renvoyer** $D^{(n)}$

3. Quelle est la complexité de votre algorithme ?

   On remarque aisément que l'algorithme de Floyd-Warshall est de complexité $\Theta(n^3)$.

4. Proposez un algorithme de construction effective des plus courts chemins à partir de l'algorithme précédent.

   Tout comme on a défini récursivement les longueurs des plus courts chemins, on peut définir récursivement les prédécesseurs dans les plus courts chemins : $\pi_{i,j}^{(k)}$ représente ici le prédécesseur du sommet j dans le plus court chemin de i à j n'utilisant comme sommets intermédiaires que des sommets parmi $\{1, 2, ..., k\}$. Pour $k = 0$, un plus court chemin ne possède aucun sommet intermédiaire, donc :

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

   Dans le cas général, si le plus court chemin est de la forme $i \rightsquigarrow k \rightsquigarrow j$ le prédécesseur de j est le même que celui du plus court chemin de k à j et n'utilisant comme sommets intermédiaires que des sommets parmi $\{1, 2, ..., k-1\}$. Autrement, on prend le même prédécesseur de j que celui qui se trouvait sur le plus court chemin de i à j et n'utilisant comme sommets intermédiaires que des sommets parmi $\{1, 2, ..., k-1\}$. Nous avons donc, dans tous les cas :

$$\pi_{i,j}^{(k)} = \begin{cases} \pi_{i,j}^{(k-1)} & \textit{si } d_{i,j}^{(k-1)} \leq d_{i,k}^{(k-1)} + d_{k,j}^{(k-1)}, \\ \pi_{k,j}^{(k-1)} & \textit{si } d_{i,j}^{(k-1)} > d_{i,k}^{(k-1)} + d_{k,j}^{(k-1)}. \end{cases}$$

   Ce qui nous donne l'algorithme suivant :

```
FLOYD-WARSHALL(W)
  n ← lignes(W)
  D^(0) ← W
  pour i ← 1 à n faire
    pour j ← 1 à n faire
      si i = j ou w_{i,j} = ∞
        alors π^(0)_{i,j} = NIL
        sinon π^(0)_{i,j} = i
  pour k ← 1 à n faire
    pour i ← 1 à n faire
      pour j ← 1 à n faire
        si d^(k-1)_{i,j} ≤ d^(k-1)_{i,k} + d^(k-1)_{k,j}
          alors
            d^(k)_{i,j} ← d^(k-1)_{i,j}
            π^(k)_{i,j} ← π^(k-1)_{i,j}
          sinon
            d^(k)_{i,j} ← d^(k-1)_{i,k} + d^(k-1)_{k,j}
            π^(k)_{i,j} ← π^(k-1)_{k,j}
  renvoyer D^(n) et Π^(n)
```

5. Exécutez votre algorithme sur le graphe de la figure 1.

   La figure 3 présente le résultat de l'exécution de l'algorithme de Floyd-Warshall sur le graphe de la figure 1.

$$D^{(0)} = \begin{pmatrix} 0 & 3 & 8 & \infty & -4 \\ \infty & 0 & \infty & 1 & 7 \\ \infty & 4 & 0 & \infty & \infty \\ 2 & \infty & -5 & 0 & \infty \\ \infty & \infty & \infty & 6 & 0 \end{pmatrix} \qquad \Pi^{(0)} = \begin{pmatrix} \text{NIL} & 1 & 1 & \text{NIL} & 1 \\ \text{NIL} & \text{NIL} & \text{NIL} & 2 & 2 \\ \text{NIL} & 3 & \text{NIL} & \text{NIL} & \text{NIL} \\ 4 & \text{NIL} & 4 & \text{NIL} & \text{NIL} \\ \text{NIL} & \text{NIL} & \text{NIL} & 5 & \text{NIL} \end{pmatrix}$$

$$D^{(1)} = \begin{pmatrix} 0 & 3 & 8 & \infty & -4 \\ \infty & 0 & \infty & 1 & 7 \\ \infty & 4 & 0 & \infty & \infty \\ 2 & 5 & -5 & 0 & -2 \\ \infty & \infty & \infty & 6 & 0 \end{pmatrix} \qquad \Pi^{(1)} = \begin{pmatrix} \text{NIL} & 1 & 1 & \text{NIL} & 1 \\ \text{NIL} & \text{NIL} & \text{NIL} & 2 & 2 \\ \text{NIL} & 3 & \text{NIL} & \text{NIL} & \text{NIL} \\ 4 & 1 & 4 & \text{NIL} & 1 \\ \text{NIL} & \text{NIL} & \text{NIL} & 5 & \text{NIL} \end{pmatrix}$$

$$D^{(2)} = \begin{pmatrix} 0 & 3 & 8 & 4 & -4 \\ \infty & 0 & \infty & 1 & 7 \\ \infty & 4 & 0 & 5 & 11 \\ 2 & 5 & -5 & 0 & -2 \\ \infty & \infty & \infty & 6 & 0 \end{pmatrix} \qquad \Pi^{(2)} = \begin{pmatrix} \text{NIL} & 1 & 1 & \text{NIL} & 1 \\ \text{NIL} & \text{NIL} & \text{NIL} & 2 & 2 \\ \text{NIL} & 3 & \text{NIL} & 2 & 2 \\ 4 & 1 & 4 & \text{NIL} & 1 \\ \text{NIL} & \text{NIL} & \text{NIL} & 5 & \text{NIL} \end{pmatrix}$$

$$D^{(3)} = \begin{pmatrix} 0 & 3 & 8 & 4 & -4 \\ \infty & 0 & \infty & 1 & 7 \\ \infty & 4 & 0 & 5 & 11 \\ 2 & -1 & -5 & 0 & -2 \\ \infty & \infty & \infty & 6 & 0 \end{pmatrix} \qquad \Pi^{(3)} = \begin{pmatrix} \text{NIL} & 1 & 1 & \text{NIL} & 1 \\ \text{NIL} & \text{NIL} & \text{NIL} & 2 & 2 \\ \text{NIL} & 3 & \text{NIL} & 2 & 2 \\ 4 & 3 & 4 & \text{NIL} & 1 \\ \text{NIL} & \text{NIL} & \text{NIL} & 5 & \text{NIL} \end{pmatrix}$$

$$D^{(4)} = \begin{pmatrix} 0 & 3 & -1 & 4 & -4 \\ 3 & 0 & -4 & 1 & -1 \\ 7 & 4 & 0 & 5 & 3 \\ 2 & -1 & -5 & 0 & -2 \\ 8 & 5 & 1 & 6 & 0 \end{pmatrix} \qquad \Pi^{(4)} = \begin{pmatrix} \text{NIL} & 1 & 4 & 2 & 1 \\ 4 & \text{NIL} & 4 & 2 & 1 \\ 4 & 3 & \text{NIL} & 2 & 1 \\ 4 & 3 & 4 & \text{NIL} & 1 \\ 4 & 3 & 4 & 5 & \text{NIL} \end{pmatrix}$$

$$D^{(5)} = \begin{pmatrix} 0 & 1 & -3 & 2 & -4 \\ 3 & 0 & -4 & 1 & -1 \\ 7 & 4 & 0 & 5 & 3 \\ 2 & -1 & -5 & 0 & -2 \\ 8 & 5 & 1 & 6 & 0 \end{pmatrix} \qquad \Pi^{(5)} = \begin{pmatrix} \text{NIL} & 3 & 4 & 5 & 1 \\ 4 & \text{NIL} & 4 & 2 & 1 \\ 4 & 3 & \text{NIL} & 2 & 1 \\ 4 & 3 & 4 & \text{NIL} & 1 \\ 4 & 3 & 4 & 5 & \text{NIL} \end{pmatrix}$$

FIG. 3 – Séquence des matrice $D^{(k)}$ et $\Pi^{(k)}$ calculées par l'algorithme FLOYD-WARSHALL pour le graphe de la figure 1.