Projet de maths S4 – Année 2016-2017
IUT de Villetaneuse - Université Paris 13
Tuteur : M. Bonino (bonino@univ-paris13.fr)

# RECHERCHE D'UN PLUS COURT CHEMIN DANS UN GRAPHE

**Contexte.** Le problème abordé ici est celui de la recherche algorithmique d'un plus court chemin entre deux sommets d'un graphe orienté. Il apparait naturellement dans la modélisation de nombreuses situations concrètes (circulation sur un réseau routier, routage de données informatiques,...). On présente ici l'algorithme de Dijkstra traitant de cette question.

---

**Objectifs.** - Analyser et programmer l'algorithme de Dijkstra ;
- produire une interface conviviale pour la saisie du graphe, donnant aussi une représentation graphique (lorsque le nombre de sommets est raisonnable) et permettant de visualiser un plus court chemin entre deux sommets.

---

**Logiciels, langages de programmation.** Langage de votre choix.

---

On considère un graphe fini orienté $\mathcal{G} = (X, U)$ où $X$ désigne l'ensemble des sommets et $U \subset X^2$ l'ensemble des arcs (ou arêtes). De plus, on associe à chaque arc $a \in U$ une « longueur » $L(a) \geq 0$ ; on dit que le graphe est valué. Dans les applications concrètes, ce nombre $L(a)$ peut représenter bien sûr une distance, mais aussi un coût, une durée etc. Si l'on considère maintenant un chemin $\mu = [x_0, x_1, \cdots, x_n]$ dans ce graphe, on définira naturellement la longueur de $\mu$ comme la somme des longueurs des $n$ arcs $(x_{i-1}, x_i)$, autrement dit on posera

$$L(\mu) = \sum_{i=1}^{n} L(a_i).$$

où $a_i$ désigne l'arc de $x_{i-1}$ à $x_i$, c'est à dire $a_i = (x_{i-1}, x_i)$. On notera bien que la longueur $L(\mu)$ ne compte pas le nombre d'arcs dans le chemin $\mu$ (sauf si $L(a) = 1$ pour tout arc $a \in U$).

Étant donné un sommet $s \in X$, on cherche pour tout $x \in X$ la longueur d'un plus court chemin $[s, \cdots, x]$ joignant $s$ à $x$. On notera $d(x)$ cette longueur en convenant que $d(x) = \infty$ s'il n'y a pas de chemin de $s$ à $x$ et que $d(s) = 0$. Ainsi $d(x)$ peut s'interpréter comme la « distance » de $s$ à $x$.

**Partie I.** Écrire un programme permettant d'entrer un graphe valué $\mathcal{G} = (X, U)$, de choisir un sommet $s \in X$, et réalisant l'algorithme suivant :
• Au début, aucun sommet n'est marqué ; on définit $\delta(s) = 0$ et $\delta(x) = \infty$ pour tout sommet $x \neq s$.
• On choisit parmi les sommets non marqués un sommet $x$ qui minimise $\delta$ et on marque ce sommet (en convenant bien sûr que $r < \infty$ pour tout réel $r$). On considère chaque successeur $y$ de $x$ qui n'est pas marqué et on change la valeur de $\delta(y)$ en $\delta(x) + L(x, y)$ si ce nombre est inférieur à $\delta(y)$.
• On recommence au second point tant que c'est possible.

**N.B :**
- Cet algorithme sera décrit sous forme de pseudo-code ou de logigramme dans votre rapport.
- La fonction $\delta : X \to \mathbb{R}^+ \cup \{\infty\}$ est mise à jour à chaque itération du second point de l'algorithme et, à la sortie de l'algorithme, elle est égale à la fonction $d$. Votre programme devra montrer (lorsque le nombre de sommets du graphe est raisonnable), pour chaque itération, la liste des sommets marqués et la valeur de $\delta(x)$ pour tout $x \in X$.

**Partie II. Analyse de l'algorithme de Dijkstra.**
1) Donner, en fonction des nombres de sommets et d'arcs de $\mathcal{G}$ :

a) le nombre $N$ d'itérations du second point de l'algorithme.

b) Une majoration du nombre d'opérations élémentaires (tests, opérations arithmétiques) effectuées par l'algorithme.

**Notations :**

— On désignera par $m_i$ le sommet marqué lors de la $i^{eme}$ itération ($i = 1, 2, \cdots, N$). Ainsi $\mathcal{M}_i = \{m_1, \cdots, m_i\}$ est l'ensemble des sommets marqués au cours des $i$ premières itérations.

— On écrira $\delta_i(x)$ pour désigner la valeur de $\delta(x)$ à la fin de la $i^{eme}$ itération. On notera que, avec cette convention, on a $\delta_i(m_i) = \delta_{i-1}(m_i)$ puisque $m_i$ est marqué lors du $i^{eme}$ passage dans la boucle et avant la mise à jour de $\delta$.

2)

a) Quel est l'ensemble $\mathcal{M}_1$ ? Combien vaut $\delta_1(m_1)$ ? Quels sommets $x$ vérifient $\delta_1(x) < \infty$ ?

b) Étant donné $x \in X$ que peut-on dire de la suite $\delta_1(x), \delta_2(x), \cdots, \delta_N(x)$ ?

c) Si $\delta_i(m_i) = \infty$ pour un certain $i \in \{1, \cdots, N\}$ que peut-on dire des $\delta_j(m_j)$ pour $j \geq i$ ?

3) Soient $i \in \{1, \cdots, N\}$ et $x \in X$.

a) Montrer que si $\delta_i(x) < \infty$ alors il existe un chemin $\mu = [s, \cdots, x]$ joignant $s$ à $x$ tel que

(i) $\mu$ ne passe que par des sommets dans $\mathcal{M}_i$, sauf éventuellement pour le dernier sommet $x$ ;

(ii) $L(\mu) = \delta_i(x)$.

b) Réciproquement, montrer que s'il existe un chemin $\mu = [s, \cdots, x]$ vérifiant la propriété (i) ci-dessus alors $\delta_i(x) < \infty$.

Indication : on pourra raisonner par récurrence sur $i$.

4) Le but de cette question est de montrer par récurrence sur $i$ que $d(m_i) = \delta_i(m_i)$, qui est la propriété centrale de l'algorithme.

a) Vérifier que ceci est vrai pour $i = 1$ (initialisation de la récurrence).

b) Supposons cette égalité vraie pour tout $i \in \{1, \cdots, n\}$, où $1 \leq n \leq N - 1$, et montrons qu'elle est alors vraie aussi pour $i = n + 1$ (hérédité).

— Premier cas : $d(m_{n+1}) = \infty$. Déduire du 3)a) que l'on a alors aussi $\delta_{n+1}(m_{n+1}) = \infty$.

— Second cas : $d(m_{n+1}) < \infty$. Considérons un chemin $\mu$ quelconque de $s$ à $m_{n+1}$ et parcourons le à partir de $s$ ; on note $y$ le premier sommet rencontré qui n'est pas dans $\mathcal{M}_n$ et $x$ le sommet précédant $y$ sur $\mu$ ; autrement dit

$$\mu = [s, \cdots, x, y, \cdots, m_{n+1}]$$

où les sommet de $s$ à $x$ sont dans $\mathcal{M}_n$ mais pas $y$.

Justifier chacune des inégalités

$$L(\mu) \geq L([s, \cdots, x, y]) \geq d(x) + L(x, y) \geq \delta_{n+1}(m_{n+1})$$

et en déduire que $d(m_{n+1}) = \delta_{n+1}(m_{n+1})$.

5) Déduire des questions précédentes que l'algorithme réalise bien ce que l'on souhaite.

6) Améliorer cet algorithme afin qu'il donne, pour chaque sommet $x \in X$, un plus court chemin de $s$ à $x$ (et non pas seulement la valeur de $d(x)$).

7) Interprétation des fonctions $\delta_i$. En utilisant les résultats des questions 3) et 4) montrer que $\delta_i(x)$ est la plus petite longueur possible pour un chemin de $s$ à $x$ vérifiant la propriété (i) du 3)a), si un tel chemin existe.

**Partie III. Une variation autour de l'algorithme de Dijkstra.**

On suppose dans cette partie que le graphe $\mathcal{G} = (X, U)$ possède une source $s$ et qu'à chaque arc $a \in U$ est associé une « capacité » $C(a) \geq 0$. Cette situation se rencontre naturellement dans les problèmes de maximisation de flux à travers un réseau (voir par exemple l'algorithme de Ford-Fulkerson). La capacité d'un chemin $\mu = [x_0, x_1, \cdots, x_n]$ dans ce graphe est définie comme la plus petite des capacités des $n$ arcs $(x_{i-1}, x_i)$, autrement dit on pose

$$C(\mu) = \min_{1 \leq i \leq 1} C(a_i).$$

où $a_i$ désigne l'arc de $x_{i-1}$ à $x_i$, c'est à dire $a_i = (x_{i-1}, x_i)$.

En vous inspirant des idées développées dans les parties précédentes, trouver un algorithme donnant, pour chaque sommet $x \in X$, la capacité maximale d'un chemin joignant $s$ à $x$.

Indication : comme dans l'algorithme de Dijkstra, on marquera un sommet et on mettra à jour une fonction $\gamma$ convenable à chaque passage dans la boucle de l'algorithme. À la différence de la fonction $\delta$ du I, les valeurs de $\gamma(x)$ augmentent au cours du déroulement de l'algorithme.