ALGORITHMIQUE DES GRAPHES - DIJKSTRA, FORD FULKERSON - INGENIEURS 1 - EISTI

Ecole Internationale des Sciences du Traitement de l'Information

10 avril 2009

Plus court chemin d'un sommet vers les autres

Motivations

Plus court chemin d'un sommet vers les autres

Motivations

1. Graphes orientés
2. Temps minimal de transmission dans un réseau (télécom,...)

Plus court chemin d'un sommet vers les autres

Motivations

1. Graphes orientés
2. Temps minimal de transmission dans un réseau (télécom,...)
3. Coût minimal d'un trajet

Plus court chemin d'un sommet vers les autres

Motivations

1. Graphes orientés
2. Temps minimal de transmission dans un réseau (télécom,...)
3. Coût minimal d'un trajet
4. Meilleur routage dans un réseau

Principes de l'algorithme

Principes de l'algorithme

1. Soit un graphe orienté $G(S, A)$, on construit une arborescence $T(r, S')$ à partir d'un sommet r (racine)

Principes de l'algorithme

1. Soit un graphe orienté $G(S, A)$, on construit une arborescence $T(r, S')$ à partir d'un sommet r (racine)
2. Cette arborescence couvre un ensemble de sommets $\{s \in S'\} \subset S$ et donne le plus court chemin de r à chacun des sommets de S'.

Principes de l'algorithme

1. Soit un graphe orienté $G(S, A)$, on construit une arborescence $T(r, S')$ à partir d'un sommet r (racine)
2. Cette arborescence couvre un ensemble de sommets $\{s \in S'\} \subset S$ et donne le plus court chemin de r à chacun des sommets de S'.
3. On définit 2 fonctions sur S'
   - $\pi(s') \equiv$ distance de s' à r ($\equiv$ longueur du plus court chemin de r à s' à une itération donnée en utilisant seulement comme sommets intermédiaires, des sommets qui sont déjà dans S'
   - $pere(s')$ désigne (à une itération donnée), le prédecesseur de s' sur un tel chemin

Assertions

2 assertions à vérifier

Assertions

2 assertions à vérifier

1. Pour tout sommet $y \notin S'$ et qui est lié par un arc avec un sommet x déjà dans S' on a :

$$\pi(y) = min_{x \in S'} \pi(x) + p(x, y)$$

$$(x, y) \equiv arc$$

Assertions

2 assertions à vérifier

1. Pour tout sommet $y \notin S'$ et qui est lié par un arc avec un sommet x déjà dans S' on a :

$$\pi(y) = min_{x \in S'} \pi(x) + p(x, y)$$

$$(x, y) \equiv arc$$

2. $\pi(pivot) \equiv$ longueur d'un plus court chemin de r à $pivot$ et $pere(pivot)$ est le prédecesseur de $pivot$ dans ce plus court chemin.

Description de l'algorithme

Dijkstra

Description de l'algorithme

Dijkstra

1. $pivot \leftarrow r$

Description de l'algorithme

Dijkstra

1. $pivot \leftarrow r \quad S' \leftarrow \emptyset$

Description de l'algorithme

Dijkstra

1. $pivot \leftarrow r \quad S' \leftarrow \emptyset \quad \pi(r) \leftarrow 0$

Description de l'algorithme

Dijkstra

1. $pivot \leftarrow r \quad S' \leftarrow \emptyset \quad \pi(r) \leftarrow 0$
2. pour tout sommet $x \neq r$, faire $\pi(x) \leftarrow \infty$

Description de l'algorithme

Dijkstra

1. $pivot \leftarrow r \quad S' \leftarrow \emptyset \quad \pi(r) \leftarrow 0$
2. pour tout sommet $x \neq r$, faire $\pi(x) \leftarrow \infty$
3. pour j variant de 1 à $n-1$, faire

Description de l'algorithme

Dijkstra

1. $pivot \leftarrow r \quad S' \leftarrow \emptyset \quad \pi(r) \leftarrow 0$
2. pour tout sommet $x \neq r$, faire $\pi(x) \leftarrow \infty$
3. pour j variant de 1 à $n-1$, faire
   - pour tout sommet y non encore dans S' et successeur de $pivot$ faire
     Si $\pi(pivot) + p(pivot, y) < \pi(y)$ alors

$$\pi(y) \leftarrow \pi(pivot) + p(pivot, y)$$

$$pere(y) \leftarrow pivot$$

Description de l'algorithme

Dijkstra

1. $pivot \leftarrow r \quad S' \leftarrow \emptyset \quad \pi(r) \leftarrow 0$
2. pour tout sommet $x \neq r$, faire $\pi(x) \leftarrow \infty$
3. pour j variant de 1 à $n-1$, faire
   - pour tout sommet y non encore dans S' et successeur de $pivot$ faire
     Si $\pi(pivot) + p(pivot, y) < \pi(y)$ alors

$$\pi(y) \leftarrow \pi(pivot) + p(pivot, y)$$

$$pere(y) \leftarrow pivot$$

   - chercher parmi les sommets non dans S', un sommet y tel que $\pi(y)$ soit minimum

Description de l'algorithme

Dijkstra

1. $pivot \leftarrow r \quad S' \leftarrow \emptyset \quad \pi(r) \leftarrow 0$
2. pour tout sommet $x \neq r$, faire $\pi(x) \leftarrow \infty$
3. pour j variant de 1 à $n-1$, faire
   - pour tout sommet y non encore dans S' et successeur de $pivot$ faire
     Si $\pi(pivot) + p(pivot, y) < \pi(y)$ alors

$$\pi(y) \leftarrow \pi(pivot) + p(pivot, y)$$

$$pere(y) \leftarrow pivot$$

   - chercher parmi les sommets non dans S', un sommet y tel que $\pi(y)$ soit minimum
   - $pivot \leftarrow y$

Description de l'algorithme

Dijkstra

1. $pivot \leftarrow r \quad S' \leftarrow \emptyset \quad \pi(r) \leftarrow 0$
2. pour tout sommet $x \neq r$, faire $\pi(x) \leftarrow \infty$
3. pour j variant de 1 à $n-1$, faire
   - pour tout sommet y non encore dans S' et successeur de $pivot$ faire
     Si $\pi(pivot) + p(pivot, y) < \pi(y)$ alors

$$\pi(y) \leftarrow \pi(pivot) + p(pivot, y)$$

$$pere(y) \leftarrow pivot$$

   - chercher parmi les sommets non dans S', un sommet y tel que $\pi(y)$ soit minimum
   - $pivot \leftarrow y$
   - $S' \leftarrow S' \cup (pivot)$

Description de l'algorithme

Dijkstra

1. $pivot \leftarrow r \quad S' \leftarrow \emptyset \quad \pi(r) \leftarrow 0$
2. pour tout sommet $x \neq r$, faire $\pi(x) \leftarrow \infty$
3. pour j variant de 1 à $n-1$, faire
   - pour tout sommet y non encore dans S' et successeur de $pivot$ faire
     Si $\pi(pivot) + p(pivot, y) < \pi(y)$ alors

$$\pi(y) \leftarrow \pi(pivot) + p(pivot, y)$$

$$pere(y) \leftarrow pivot$$

   - chercher parmi les sommets non dans S', un sommet y tel que $\pi(y)$ soit minimum
   - $pivot \leftarrow y$
   - $S' \leftarrow S' \cup (pivot)$
   - $j \leftarrow j + 1$

Flot maximum et coupe de capacité minimum

Motivations

1. Résolution des problèmes de transports dans un réseau de télécommunications, de canalisations (débits), de transport de produits (capacités des camions, wagons, etc ...)
2. Etude de la connectivité des graphes (Résistance aux pannes dans un réseau)

Flot maximum et coupe de capacité minimum

Définitions

- Soit $G(S, A)$ un graphe orienté
  - La capacité $C : A \rightarrow \mathcal{R}$
  - Le flot $f : A \rightarrow \mathcal{R}$
- $\forall a \in A : f(a) \equiv$ flux de a
- 2 sommets privilégiés de S :
  - la source s_O
  - le puit p
- $S_0 \subset S$ contient s_0 mais pas p
- $(S_0, {S_0}^c) \equiv \{$ensemble des arcs dont l'origine est dans S_0 et l'extémité dans ${S_0}^c\}$
- $(S_0, {S_0}^c)$ est une coupe qui sépare la source du puits

Flot maximum et coupe de capacité minimum

Définitions

- Flot du réseau :
  - $\forall a \in A,\ 0 \leq f(a) \leq c(a)$
  - $\forall s \in S$

$$\sum_{a \ d'orig.\ s} f(a) = \sum_{a' \ d'extr.\ s} f(a')$$

(loi de Kirchoff, conservation du flot)

- Capacité d'une coupe $(S_0, {S_0}^c)$:

$$c(S_0, {S_0}^c) = \sum_{a \in (S_0, {S_0}^c)} c(a)$$

Valeur du flot

Valeur du flot

- valeur à la source :

$$\sum_{a \text{ quittant } s_0} f(a) - \sum_{a' \text{ entrant en } s_0} f(a')$$

- valeur au puits :

$$\sum_{a \text{ entrant}_e n \ p} f(a) - \sum_{a' \text{ quittant } p} f(a')$$

- valeur à la coupe :

$$\sum_{a \in (S_0, {S_0}^c)} f(a) - \sum_{a' \in ({S_0}^c, S_0)} f(a')$$

Flot maximum et coupe de capacité minimum

Propositions

Trouver un flot de valeur maximum et capacité minimum

1. Si f est un flot de G et $(S_0, {S_0}^c)$ une coupe, alors $f(S_0, {S_0}^c) - f({S_0}^c, S_0)$ est indépendant du choix de S_0, donc $val(f)$ est sans ambiguités.
2. Soit f un flot de G et $(S_0, {S_0}^c)$ une coupe qui sépare s_0 et p
   - 2.1 $val(f) \leq c(S_0, {S_0}^c)$
   - 2.2 si $val(f) = c(S_0, {S_0}^c)$ alors f est de valeur maximale et $(S_0, {S_0}^c)$ est de capacité minimale
   - 2.3 $val(f) = c(S_0, {S_0}^c)$ ssi :

$$\begin{cases} \forall a \in (S_0, {S_0}^c) & f(a) = c(a) \\ \forall a \in ({S_0}^c, S_0) & f(a) = 0 \end{cases}$$

Algorithme de Ford-Fulkerson

Principe

- Recherche d'une chaîne augmentante de s_0 à p sur un flot au hasard (augmentation du flux d'un arc a de la chaîne)
  - $c(a) - f(a) \geq \alpha$ pour les arcs à l'endroit (C^+)
  - $f(a') \geq \alpha$ pour les arcs à l'envers (C^-)
- Si il existe un tel α, $val(f)$est augmenté :

$$\alpha = min\{min_{a \in C^+}\{c(a) - f(a)\}; min_{a \in C^-} f(a')\}$$

- On augmente donc $f(a)$de α $\forall a \in C^+$ et on diminue $f(a')$ de α $\forall a' \in C^-$

Théorème de Ford-Fulkerson

Théorème de Ford-Fulkerson

1. f est un flot de valeur maximum ssi $\nexists$ de chaîne augmentante de s_0 à p
2. Pour un graphe G, la valeur maximum d'un flot dans G est égale à la capacité minimum d'une coupe de G

Algorithme : 1ère phase

Marquage

- Marquer s_0 par $(\Delta, +\infty)$
- Pour tout somment $x \neq s_0$, x non marqué
- Considérer qu'aucun sommet n'est examiné
- Tant que p est non marqué et qu'il reste un sommet marqué non examiné, faire :
  - Soit x un sommet marqué non examiné et soit α, la valeur absolue du second paramètre de la marque de x.
  - Pour tout successeur y de x qui n'est pas marqué, faire
    si $c(x,y) > f(x,y)$ alors
    $\beta \leftarrow min\{\alpha, c(x,y) - f(x,y)\}$; marquer y par $(x, +\beta)$
  - Pour tout prédécesseur z de x qui n'est pas marqué, faire
    si $f(z,x) > 0$,alors $\beta \leftarrow min\{\alpha, f(z,x)\}$; marquer z par $(x, -\beta)$
  - Considérer x comme examiné

Algorithme : 2eme phase

Description

- Définir un flot f, éventuellement le flot nul
- Répéter
  - Appliquer l'algorithme de marquage
  - Si p est marqué, alors
    - Soit C la chaîne augmentante de s à p
    - Soit α la valeur absolue du second paramètre de la marque p
    - Pour tout $arc(x, y) \in C$ parcouru à l'endroit, faire $f(x, y) \leftarrow f(x, y) + \alpha$
    - Pour tout $arc(x, y) \in C$ parcouru à l'envers, faire $f(x, y) \leftarrow f(x, y) - \alpha$
- jusqu'à ce que p ne soit plus marqué

A la sortie de cette boucle, le flot est de valeur maximum.