Recherche du plus court chemin
ITU 2024
Rakotoarimalala Tsinjo

1 Problème de recherche du plus court chemin

Dans un problème de plus courts chemins, on possède en entrée un graphe orienté pondéré $G = (S, A)$, avec une fonction de pondération $w : A \to \mathbb{R}$ qui fait correspondre à chaque arc un poids à valeur réelle.

La longueur du chemin $p = (v_0, v_1, \ldots, v_k)$ est la somme des poids (longueurs) des arcs qui le constituent :

$$w(p) = \sum_{i=1}^{k} w(v_{i-1}, v_i)$$

Dans notre cas on suppose que tous les poids des arcs sont tous positifs.

On définit ***la longueur du plus court chemin*** entre u et v par

$$\delta(u, v) = \begin{cases} min\{w(p) | p : \text{ un chemin de } u \text{ vers } v\} & \text{s'il existe un chemin de } u \text{ vers } v \\ \infty & \text{sinon} \end{cases}$$

Un ***plus court chemin*** d'un sommet u à un sommet v est alors défini comme un chemin p de longueur $w(p) = \delta(u, v)$.

2 Applications

Le problème de recherche du plus court chemin:

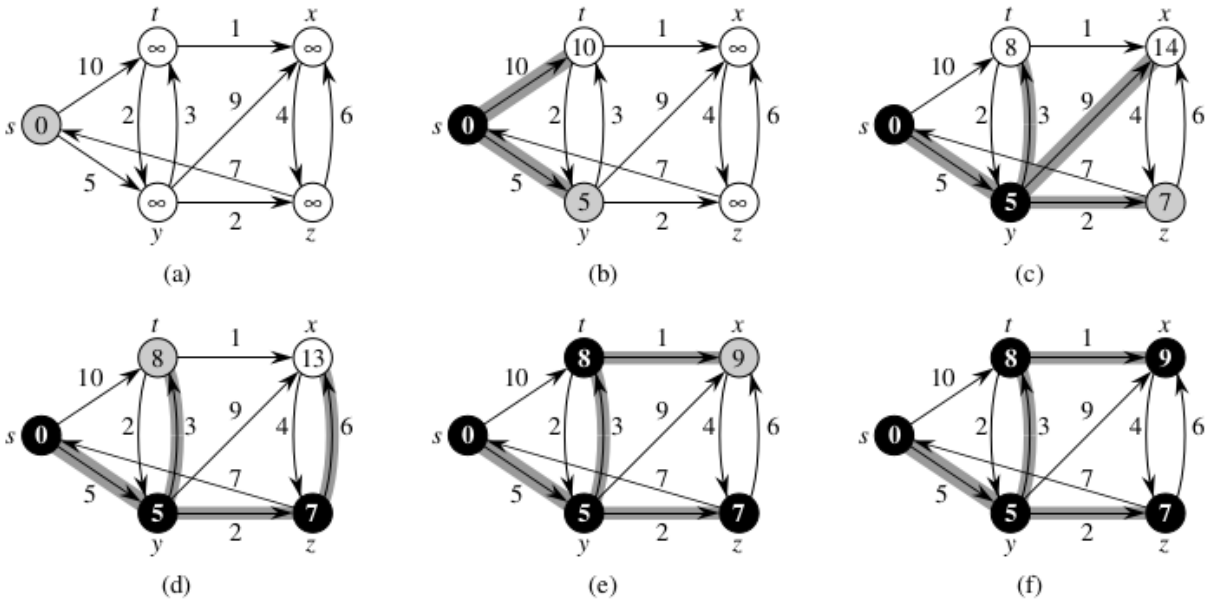
- trouve une utilité dans le calcul des itinéraires routiers. Le poids des arcs pouvant être la distance (pour le trajet le plus court), le temps estimé (pour le trajet le plus rapide), la consommation de carburant et le prix des péages (pour le trajet le plus économique).
- apparaît dans les protocoles de routage interne qui permettent un routage internet très efficace des informations en cherchant le parcours le plus efficace.

3 Algorithme de Dijkstra

Algorithme 1 Algorithme de Dijkstra

```
1: procedure PLUS-COURT-CHEMIN-DIJKSTRA(G = (S, A), s)
2:     pour tout sommet u de S faire
3:         u.distance ← ∞
4:         u.predecesseur ← NIL
5:     fin pour
6:     s.distance ← 0
7:     E ← ∅
8:     F ← S
9:     tant que F ≠ ∅ faire
10:        u ← EXTRAIRE-MIN(F)
11:        E ← E ∪ {u}
12:        pour chaque sommet v voisins de u faire
13:            si v.distance > u.distance + w(u, v) alors
14:                v.distance ← u.distance + w(u, v)
15:                v.predecesseur ← u
16:            fin si
17:        fin pour
18:    fin tant que
19: fin procedure
```

4 Exemple de déroulement de l'algorithme



5 Preuve de la validité de l'algorithme

L'ensemble des instructions des lignes $13 - 16$ est appelé ***relâchement***.

Propriété de convergence

:

Si $s \rightsquigarrow u \rightarrow v$ est un plus court chemin dans G pour un certain $u, v \in S$ et si $\texttt{u.distance} = \delta(s, u)$ à un certain instant antérieur au relâchement de l'arc (u, v), alors $\texttt{v.distance} = \delta(s, v)$ en permanence après le relâchement (ligne $13 - 16$).

□ Cette propriété peut être démontrée en faisant la remarque qu'après relâchement de l'arc (u, v), on a

$$\texttt{v.distance} \leqslant \texttt{u.distance} + w(u, v) = \delta(s, u) + w(u, v) = \delta(s, v) \Leftrightarrow \texttt{v.distance} = \delta(s, v)$$

car $\texttt{v.distance} \geqslant \delta(s, v)$

Théorème 1 *Si l'on exécute l'algorithme de Dijkstra sur un graphe orienté pondéré $G = (S, A)$, avec une fonction de pondération positive w et une origine s, après exécution l'on a $d[u] = \delta(s, u)$ pour tous les sommets $u \in S$.*

Pour démontrer ce théorème, il suffit de montrer que pour chaque sommet u de S, on a $\texttt{u.distance}= \delta(u, s)$ au moment où on a ajouté u dans E et la ligne $13 - 15$ nous garantira que cette égalité restera toujours vraie jusqu'à la fin de l'algorithme.

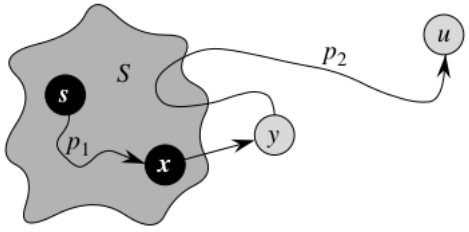
Raisonnons par l'absurde,

H : Soit u le premier sommet pour lequel $\texttt{u.distance} \neq \delta(s, u)$ quand on l'ajoute à E.

Cette hypothèse nous mène à dire que:

1. u ne peut pas être s car au début de l'algorithme $\texttt{s.distance}= \delta(s, s) = 0$ (en contradiction avec l'hypothèse H)
2. Il existe un chemin p de s à u sinon $\texttt{u.distance} = \delta(s, u) = \infty$ (en contraction à l'hypothèse). Et puisqu'il existe un chemin donc il existe un chemin plus court p, de s vers u.
3. p peut être décomposé comme suite $(p_1 : s \rightsquigarrow x, p_2 : y \rightsquigarrow u)$ avec
   (a) p_1 un chemin d'une suite d'éléments de E
   (b) x un élément de E
   (c) y un élément de $S - \{E\}$
      □ Cet élément y doit exister sinon p sera de la forme $p^i : s \rightsquigarrow x \rightarrow u$. Et d'après H, $\texttt{x.distance} = \delta(s, x)$ (car u est le premier qui ne vérifie pas cette égalité). Ainsi après relâchement de l'arc (x, u), forcement $\texttt{u.distance} = \delta(s, u)$ selon la propriété de convergence.
   (d) p_2 un chemin de y vers u pouvant contenir d'éléments de E et des éléments de $S - \{E\}$).

   (L'un ou l'autre des chemins p_1 ou p_2 peut ne pas avoir d'arcs.)



4. D'après l'hypothèse H u est le premier à être ajouté dans E tel que $\texttt{u.distance} \neq \delta(s, u)$. Donc $\texttt{x.distance} = \delta(s, x)$

5. D'après la propriété de convergence, $\texttt{y.distance} = \delta(s, y)$ après avoir relâcher les voisins de x. (ligne $13 - 16$)

6. On peut en déduire donc que

$$\texttt{y.distance} = \delta(s, y) \leqslant \delta(s, u) \leqslant \texttt{u.distance}$$

7. Mais puisque u fut choisit avant y pour être introduite dans E (ligne 10) donc à ce moment

$$\texttt{u.distance} \leqslant \texttt{y.distance}$$

8. Donc d'après les deux équations précédentes (sandwich)

$$\texttt{u.distance} = \delta(s, u) = \delta(s, y) = \texttt{y.distance}$$

   Ce qui est absurde par rapport à l'hypothèse H. (démonstration par l'absurde)

Donc l'hypothèse H est fausse. Ainsi pour tout u introduit dans E, $\texttt{u.distance} = \delta(s, u)$. Et une fois $\delta(s, u)$ atteint, il ne sera plus changé par un relâchement.