Algorithmique
Plus courts chemins : Dijkstra

L'algorithme de Dijkstra détermine les plus courts chemins entre un sommet de départ (la source *src*) et tous les autres sommets du graphe accessibles depuis *src* dans les graphes orientés [1] dont les **coûts des arcs sont positifs ou nuls**.

Problème et solution

Rappels

— La *distance* entre deux sommets x et y est la somme des coûts des arcs qui constituent le chemin de x à y.

— Trouver le plus court chemin entre deux sommets (d'un sommet *src* à un sommet *dst*) nécessite de calculer les plus courts chemins depuis le sommet *src* jusqu'aux autres sommets du graphe.

Représentation de la solution

La solution est représentée par deux vecteurs :

— Le vecteur des distances, ici *dist* : $\forall x \in S$, $dist[x]$ représente la distance de la source à x.
Le vecteur de distances est initialisé à l'infini ($+\infty$) au départ, sauf pour la source qui est nulle !

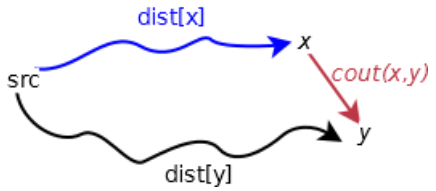
— Le vecteur *pred* : $\forall x \in S$, $pred[x]$ représente le prédécesseur de x dans le chemin du sommet source au sommet x.
Il suffira de "remonter" ce vecteur depuis un sommet jusqu'à atteindre la source pour retrouver le chemin.

La solution peut être représentée par un arbre ayant pour racine le sommet source. Cet arbre est représenté par le vecteur *pred* qui contient pour chaque sommet son père dans l'arbre.

Mises à jour des vecteurs : Le relâchement d'arcs

Soit x un sommet dont on a trouvé un plus court chemin depuis la source. Soit y un successeur de x.

La question : Est-ce que le chemin (depuis la source) pour arriver jusqu'à y est plus court en passant par x ?



relâchement de l'arc (x, y) :

```
si dist[x] + cout(x,y) < dist[y] alors
   dist[y] ← dist[x] + cout(x,y)
   pred[y] ← x
```

Dijkstra : le principe

Un ensemble contient au départ tous les sommets du graphe : à chaque itération, on **choisit parmi les sommets de l'ensemble un sommet x de distance minimale**. Les coûts étant positifs ou nuls, la distance de x depuis la source est définitive (plus aucun relâchement ne viendra minimiser cette distance). On enlève x de l'ensemble, et on relâche tous ses arcs sortants : on parcourt les successeurs de x et on met à jour leurs distances et prédécesseurs si nécessaire.

Le premier sommet choisi est donc la source (le seul sommet dont la distance est connue) : dans **l'algorithme donné en annexe**, le vecteur des distances est donc directement initialisé avec les coûts des arcs (x, y) pour tous les successeurs y (les autres sommets restant à $+\infty$).

L'algorithme s'arrête lorsque :

— Tous les sommets ont été traités.

— Il ne reste plus que des sommets non accessibles depuis la source.

Si la recherche est limitée à un plus court chemin vers une destination donnée, il n'est pas nécessaire de continuer l'algorithme lorsque la destination est choisie.

1. Dijkstra peut être utilisé sur des graphes non orientés.