CORRIGÉ DU DEVOIR À LA MAISON N^o 1

Description informelle
La solution la plus simple consiste à effectuer deux appels à l'algorithme de Dijkstra sur le graphe G', construit à partir du graphe G dont on a enlevé l'arête négative, une fois avec comme point de départ l'origine x_0, une seconde fois avec comme point de départ l'extrémité de l'arête négative t. La complexité est bien de l'ordre de grandeur requis, et la terminaison est garantie, puisqu'il s'agit de deux appels à l'algorithme de Dijkstra dans un graphe n'ayant que des arêtes positives. La correction viendra du fait qu'un plus court chemin de x_0 à x ne passant pas par s est détecté dans G', et que les deux parties d'un plus court chemin passant par l'arête négative auquel on a enlevé cette dernière sont détectées dans G'.

Pseudo-code On suppose que l'on dispose d'une fonction `Enlever`, qui prend en argument un graphe G et une arête a de ce graphe et renvoie le graphe G' construit à partir de G en enlevant l'arête a. On suppose en outre que l'on dispose d'une fonction `Dijkstra`, qui prend en entrée un graphe et un sommet origine, et renvoie deux tableaux `d` et `pere` comme dans le poly. On suppose enfin donnée l'arête $s \to t$ de longueur négative.

```
Plus_courts_chemins(G, s→ t, x_0)
G' ← Enlever(G, s→ t)
(d_1, pere_1) ← Dijkstra(G', x_0)
Si d_1[t] >= d_1[s] + l(s→t) alors
      (d_2, pere_2) ← Dijkstra(G', t)
      Si d_2[s] < - l(s→ t) erreur("Cycle de longueur < 0")
      Pour x sommet de G faire
           Si d_1[x] > d_1[s] + l(s→ t) + d_2[x] faire
                d_1[x] ← d_1[s] + l(s→ t) + d_2[x]
                pere_1[x] ← pere_2[x]
      finfaire
finsi
Renvoyer (d_1, pere_1)
```

Preuve On remarque qu'il y a deux possibilités pour chaque sommet x :

- Le plus court chemin de s à x passe par l'arête négative ; dans ce cas, il se décompose en un plus court chemin de x_0 à s, l'arête (s,t), et un plus court chemin de t à x.
- Le plus court chemin de s à x ne passe pas par l'arête négative ; dans ce cas, il est trouvé dans le premier appel à Dijkstra, sur le graphe ne contenant pas l'arête négative.

Pour que ce raisonnement soit valide, il faut garantir qu'il existe un plus court chemin contenant au plus une fois l'arête négative. Soit c un plus court chemin contenant l'arête négative un nombre de fois minimal. S'il contient plusieurs fois l'arête, il contient un cycle $s \to t \to \cdots \to s$. La suppression de ce cycle construit un chemin de x_0 à x contenant

Date: 7 février 2004.

l'arête négative une fois de moins que c, et qui est donc nécessairement strictement plus long que c ; dans ce cas, le cycle construit plus haut est de longueur strictement négative, et le plus court chemin est de longueur $-\infty$; le problème n'a donc pas de sens dans ce cas, sauf éventuellement pour les sommets qui ne sont pas accessibles depuis le cycle de longueur négative, donc depuis t ; la réponse est alors simplement pour ces sommets le résultat du premier appel à Dijkstra.

Ce problème est détecté en cours d'algorithme simplement en remarquant que l'existence d'un cycle de longueur négative dans G revient à affirmer l'existence d'un chemin de t à s dans G' de longueur inférieure à $|l(s,t)|$; or le chemin le plus court de t à s dans G' est calculé lors du second appel à Dijkstra.

Il reste à prouver que le tableau `pere` est correct. Dans le cas où le test correspondant est vrai, on sait qu'un chemin le plus court allant jusqu'à `x` se décompose en un chemin le plus court allant jusqu'à `pere_2[x]` via l'arête `s`$\to$ `t`, puis l'arête allant de `pere_2[x]` à `x`. Cette remarque prouve la validité du tableau `pere` par récurrence.

Analyse de complexité. Le calcul effectue deux appels à Dijkstra sur des graphes ayant $O(n)$ sommets et $O(m)$ arêtes ; le coût de chacun de ces appels est donc $O((n+m)\log n)$, et le coût global est donc du même ordre de grandeur. Le reste du calcul est linéaire en m, donc négligeable.