EpiReboot Lem-in - Dijkstra

14 avril 2015

Pour trouver un plus court chemin il convient d'utiliser, par exemple, l'algorithme de Dijkstra. Le but de cette séance est de vous faire implémenter cet algorithme en vous en donnant le principe.

Tout d'abord il est nécessaire de procéder à une étape d'initialisation. Cela se traduit par l'implémentation d'une fonction prototypée de la façon suivante.

```
void init_dijkstra(char** graph, int indice_begin, int* distance, int* previous);
```

Son pseudo-code est le suivant :

```
pour chaque sommet v du graphe G
   faire distance[v] = -1
         previous[v] = -1

distance[indice_begin] = 0
```

Par ailleurs, une fonction dite de relâchement sera utile pour l'algorithme de Dijkstra. Elle sera prototypée de la manière suivante.

```
void relax(char** graph, int* distance, int* previous, int u, int v);
```

Son pseudo-code est le suivant.

```
si distance[v] > distance[u] + graph[u][v]
    alors distance[v] = distance[u] + graph[u][v]
          previous[v] = u
```

Il est à noter que si il n'y a pas de chemin entre deux sommets dans le graphe, le poids est censé être infini.

Le prototype de l'algorithme de Dijkstra est le suivant.

```
void dijkstra(char** graph, int init_sommet);
```

Son pseudo-code est donné en suivant.

```
appel de init_dijkstra
E = {}
F = liste des sommets de G
tant que F != liste vide
    faire u = extraire_min(F)
          ajouter u dans E
          pour chaque sommet v voisin à u
               faire appeler relax avec u et v
```