EpiReboot Specialisation - Lem-in v2

7 avril 2015

1 Introduction

Bienvenue pour cette première journée consacrée au Lem-in !

2 Parsing

L'étape indispensable de ce projet est de parser le fichier donné sur l'entrée standard. Rappelons un exemple de fichier.

```
42
##start
1 23 3
2 16 7
#commentaire
3 16 3
4 16 5
5 3 9
6 1 0
7 4 8
##end
0 9 5
0-4
0-6
1-3
4-3
5-2
3-5
#autre commentaire
4-2
2-1
7-6
7-2
7-4
6-5
```

Une première possibilité, si vous désirez représenter votre fourmilière sous forme de liste de tubes, va donc être de :

1. stocker le nombre de fourmis dans une variable,
2. stocker l'ensemble des salles dans une liste, ainsi que de stocker le départ et l'arrivée dans 2 variables distinctes,
3. enfin de stocker les arêtes/tubes dans une liste.

Une autre possibilité concernant la troisième étape est aussi de créer directement votre graphe, par exemple sous forme de matrice, à partir des données d'entrée. Pour cela vous aurez besoin d'une petite lib sur les graphes.

3 Construction du graphe

Il s'agit de créer une petite librairie de manipulation de graphe. Elle permettra de :

1. créer une matrice à partir d'un ensemble de salle,
2. ajouter une arête entre 2 salles,
3. donner les salles voisines d'une salle donnée.

Créer une fonction qui renvoie une matrice carrée remplie de 0 à partir d'un nombre de salles. Elle sera prototypée de la manière suivante :

```
char** create_matrix(int nb_salles);
```

Créer une fonction qui à partir d'une matrice ajoute un chemin entre 2 salles. Elle sera prototypée de la manière suivante :

```
void add_edge(char** matrix, int node1, int node2);
```

Créer une fonction qui à partir d'une matrice renvoie les voisins d'une salle donnée. Elle sera prototypée de la manière suivante :

```
t_node* neighbor(char** matrix, int node1, int node2);
```

où t_node est la structure d'une liste chainée contenant un int correspondant au numéro de salle.

4 Parcours profondeur

5 Parcours largeur

6 Plus court chemin