# Projet 1 : graphes et labyrinthes

Vincent Gripon

November 6, 2009

Charlie s'est perdu dans un labyrinthe, et doit trouver la sortie. On se propose d'utiliser plusieurs méthodes de la théorie des graphes pour l'aider à s'en sortir. Le fichier `laby.sci` contient le code source permettant la génération d'un labyrinthe aléatoire. Charlie commence sur la case (0,0) et doit atteindre la dernière case du labyrinthe pour s'en sortir.

## 1 Quelques manipulations pour commencer

**Question 1)**
Listez les éléments importants sur le graphe, quelle propriété vue en cours serait suffisante pour assurer à Charlie qu'il puisse s'en sortir ?

**Question 2)**
On se propose pour rendre les choses plus simples de colorer la position de Charlie en rouge. En s'aidant des fonctions `coordx`, `coordy` et `case`, codez des fonctions `haut`, `gauche`, `bas` et `droite` qui permettent de déplacer Charlie dans le graphe dans les directions indiquées si cela est possible. Si le déplacement n'est pas autorisé par le graphe, ces fonctions rendront 0, sinon 1.

Afin de visualiser ce déplacement, la nouvelle position devra être colorée en rouge à la place de l'ancienne sur l'affichage.

**Question 3)**
Charlie connait une bonne vieille méthode pour se sortir d'un labyrinthe : il suffit de toujours prendre le plus à gauche possible et de continuer d'avancer. Programmez cet algorithme pour que Charlie sorte du labyrinthe quand c'est possible. De quel type de parcours s'agit-il ?

**Question 3 bonus)**
Tracez la courbe de la proportion de labyrinthes dont la sortie est atteignable en fonction de la probabilité d'existence d'un mur.

## 2 Des parcours plus évolués

**Question 4)**
On se propose maintenant d'utiliser des parcours plus évolués de la théorie des graphes pour aider Charlie à s'en sortir. Commencez par programmer l'autre

type de parcours. Quel est le gros avantage de cette méthode par rapport à l'ancienne ?

**Question 5)**

On se place maintenant dans le cas où le labyrinthe est plus complexe: les transitions sont à présent orientées, et sont dotées d'un poids représentant le temps requis à Charlie pour se déplacer d'une case à l'autre. Transformez le programme de base pour qu'il permette à présent de prendre en compte ces modifications (on choisira des temps entiers compris entre 1 et 10).

**Question 6)**

Utilisez l'algorithme de Roy-Warshall pour déterminer le chemin le plus court entre votre position et l'arrivée.

**Question 7)**

Charlie possède maintenant un plan du labyrinthe, mais doit en sortir en un temps limité. Calculer toutes les plus courtes distances point à point serait beaucoup trop long. Proposez et programmez un autre algorithme qui permet de trouver le chemin le plus court à la sortie et qui soit moins couteux.

# 3 Pour les plus rapides

**Question 8)**

Le labyrinthe n'a maintenant plus de mur. Dix surprises ont été placées à différentes positions du labyrinthe et indiquées à Charlie. Programmez un algorithme qui lui permette d'aller toute les chercher en le moins de temps possible.