

UNIVERSITÉ DE LIÈGE

PROGRAMMATION AVANCÉE

INFO2050

Projet 2 : Structures de données

Noémie Lecocq
Andrew Sassoie

2017-2018



1 Analyse théorique

a)

b)

c) La structure de labyrinthe

La structure *maze* est composée de 4 éléments.

- *size* : contient la taille du labyrinthe. Pour une taille N , le labyrinthe sera un carré $N \times N$.
- *unionFind* : contient l'ensemble disjoint qui a été utilisé pour créer le labyrinthe.
- *neighbours* : est un vecteur contenant les paires de cellules étant voisines.
- *convert* : est une matrice permettant de retrouver le numéro d'une case du labyrinthe à partir de ses coordonnées.

En effet, nous avons choisi d'utiliser la plupart du temps des entiers pour identifier les cases afin de pouvoir utiliser la structure *UnionFind* préalablement définie (et qui contenait des entiers). Ainsi nous représentons chaque case par un chiffre de 0 à $N * N - 1$

d) Pseudocode

MZCREATE(*size*)

```
1  MAZE maze
2  maze.size = size
3  maze.unionFind = UFCREATE(size)
4  while UFCOMPONENTSCOUNT(maze.unionFind) > 1
5      (coord1, coord2) = a random pair of neighbours
6      if There is a wall
7          Remove it
8          UFUNION(maze.unionFind, coord1, coord2)
9  return maze
```

MZISVALID(*maze*)

```
1  if UFCOMPONENTSCOUNT(maze.unionFind) > 1
2      return false
3  else
4      return true
```

e) Complexité en temps avec UnionFindList

MZISVALID

MZISVALID est constant puisqu'il s'agit simplement d'aller lire la taille de l'*UnionFind*.

2 Analyse empirique

a)

JE SUIS OCCUPE A FAIRE CA :)

b)