TP - Labyrinthes, génération et résolution

Florian Bourse

1 Génération aléatoire de labyrinthes

Spécification du programme :

entrée : 2 entiers n et m donnés en ligne de commande correspondant à la largeur et la hauteur souhaitées pour le labyrinthe.

sortie : Un labyrinthe parfait généré aléatoirement et affiché en ascii art en suivant les conventions suivantes : un mur est représenté par "`#`", un espace vide par " ", le programme doit afficher $2m + 1$ lignes de $2n + 1$ caractères.
Exemple :

```
#####################
#           #   # # #
### ### ##### # # # #
#   #     # # # #   #
### ### ### # ##### #
#   # #         # # #
# ### ######### # # #
#           #       #
#####################
```

Structure de données :

- Un tableau `horizontal` gardant en mémoire le statut des murs représentés par "`-`" sur le schéma ci-dessous.
- Un tableau `vertical` gardant en mémoire le statut des murs représentés par "`|`" sur le schéma ci-dessous.
- Une structure de données unir & trouver contenant un élément pour chaque nœud du graphe, représenté par " " sur le schéma ci-dessous.

```
#####################
# - - - - - - - - - -
#|#|#|#|#|#|#|#|#|#|#
# - - - - - - - - - -
#|#|#|#|#|#|#|#|#|#|#
# - - - - - - - - - -
#|#|#|#|#|#|#|#|#|#|#
# - - - - - - - - - -
#|#|#|#|#|#|#|#|#|#|#
```

Remarque : Il y a $n \times m$ nœuds, $n \times (m-1)$ murs horizontaux potentiels et $m \times (n-1)$ murs verticaux potentiels, mais pour simplifier l'implémentation, nous allons numéroter tous ces éléments entre 0 et $n \times m - 1$:

```
#####
#0-1#
#|#|#
#2-3#
#|#|#
#4-5#
#|#|#
#6-7#
#####
```

La case située en ligne i et colonne j est numéroté $i \times n + j$. Chaque mur horizontal porte le numéro de la case à sa gauche. Chaque mur vertical porte le numéro de la case au dessus. En particulier, les murs horizontaux numérotés par un nombre dont le successeur est un multiple de n ne peut pas être enlevé car il n'appartient pas au graphe, de même pour les murs verticaux numérotés par un nombre supérieur à $n \times (m-1)$.

Tâches à accomplir :

- Lire n et m dans le tableau `argv`, afficher un message si le nombre d'arguments attendus n'est pas le bon (on pourra vérifier la valeur de `argc`)
- Générer le labyrinthe
- Afficher le labyrinthe
- *(Bonus)* Ajouter une troisième dimension. On représentera un chemin vers le haut par "/", un chemin vers le bas par "\" et un chemin où on peut aller vers le haut et vers le bas par "|".

2 Résolution de labyrinthe

Tâches à accomplir :

- Implémenter l'algorithme A* pour trouver le chemin le plus court entre deux cases en utilisant comme heuristique la distance de Manhattan. (on pourra séparer cette tâche en plusieurs tâches intermédiaires comme implémenter une file de priorité)
- Tirer aléatoirement deux cases du labyrinthe, et appliquer cet algorithme.
- Afficher la solution en même temps que le labyrinthe.