INF2010 :

Structure de données et Algorithmes

Bakashov, Marsel

Chowdhury, Rasel

27 novembre 2022

Partie 2.1 :

Implémenter la classe **Counter** dans chaque algorithme afin d'observer le nombre de *Tile* traversés pendant que l'algorithme cherche pour la sortie.

À voir dans le code remis !

Partie 2.2 :

Implémenter la classe **Counter** dans chaque algorithme afin d'observer le nombre de *Tile* ajoutés dans la pile pendant l'algorithme cherche pour la sortie. Afficher la taille de la file/pile/récursivité à chaque itération.

Conserver la valeur maximale d'élément dans la file/pile/récursivité dans l'attribut *maxStackSize*.

À voir dans le code remis !

Partie 2.3 :

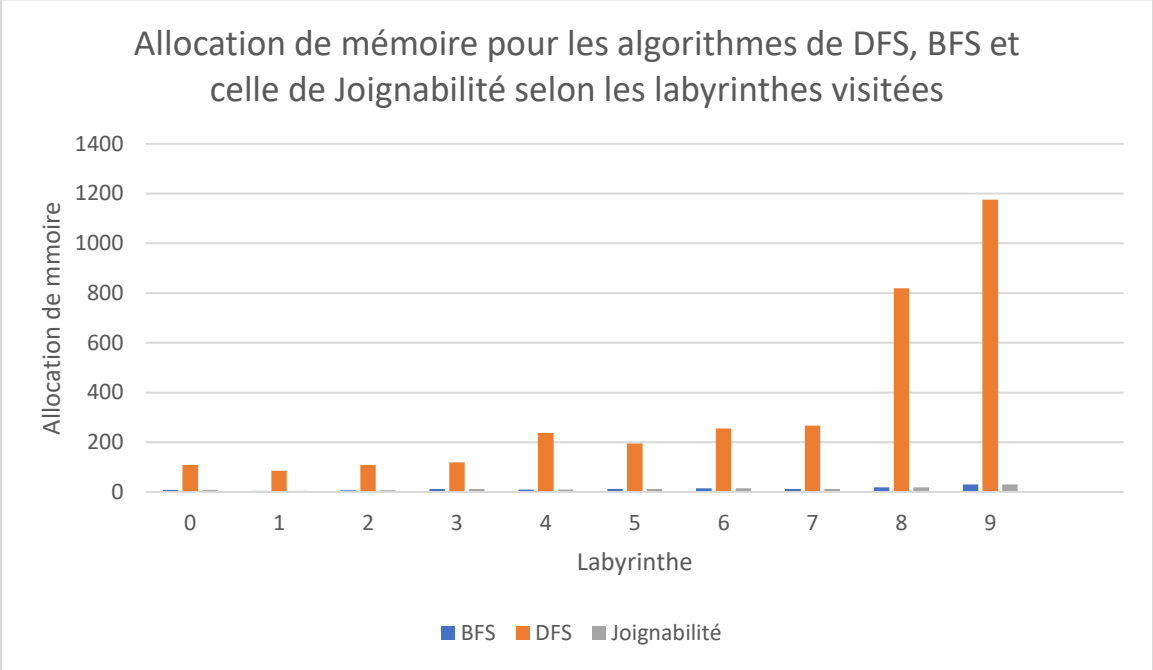
Démontrer les caractéristiques de vos algorithmes avec des graphiques visuels (exemple : graphique excel). Ensuite, noter vos observations.

(**)Les comportement moyens seront mesurés après avoir testé les algorithmes sur plusieurs labyrinthes partagés.*

Discuter critiquement des résultats obtenus pour les points 1,2,3, et 4. Partager vos observations et les conclusions que vous pouvez en sortir.

(---) : Valeur maximal d'élément dans la file / la pile

- L'algorithme de BFS alloue en moyenne tel nombre de mémoire:
  (8 + 5 + 7 + 12 + 10 + 12 + 15 + 12 + 19 + 30) / 10 = **13**
- L'algorithme de DFS alloue en moyenne tel nombre de mémoire:
  (109 + 85 + 109 + 119 + 237 + 195 + 255 + 267 + 819 + 1175) / 10 = **337**
- L'algorithme de Joignabilité alloue en moyenne tel nombre de mémoire:
  (8 + 5 + 7 + 12 + 10 + 12 + 15 + 12 + 19 + 30) / 10 = **13**



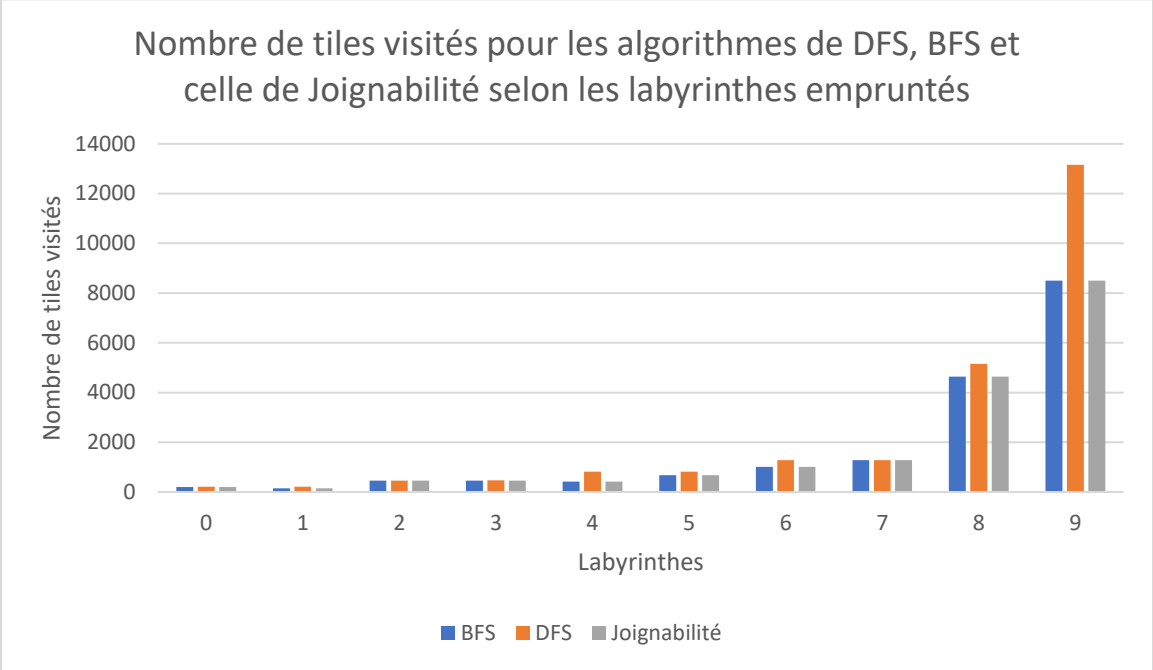
1. Quel algorithme alloue le moins de mémoire en moyenne. *

   Selon nos résultats, l'algorithme de **BFS et de Joignabilité** alloue le moins de mémoire.

2. Quel algorithme alloue le plus de mémoire en moyenne. *

   Selon nos résultats, l'algorithme de **DFS** alloue le plus de mémoire.

- L'algorithme de BFS visite tel nombre de Tile en moyenne :
  (197 + 150 + 457 + 461 + 422 + 673 + 1013 + 1285 + 4641 + 8500) / 10 = **1779.90**
- L'algorithme de DFS visite tel nombre de Tile en moyenne :
  (214 + 208 + 462 + 471 + 822 + 819 + 1279 + 1284 + 5150 + 13162) / 10 = **2387.1**
- L'algorithme de Joignabilité visite tel nombre de Tile en moyenne :
  (197 + 150 + 457 + 461 + 422 + 673 + 1013 + 1285 + 4641 + 8500) / 10 = **1779.90**



3. Quel algorithme visite le plus de *Tile* en moyenne. *

   L'algorithme(s) qui visite le plus de Tile en moyenne est celle de **DFS**.

4. Quel algorithme visite le moins de *Tile* en moyenne. *

   L'algorithme(s) qui visite le moins de Tile en moyenne est celle de **BFS et celle de Joignabilité**

Pour ce qui est de la théorie derrières ces aspect consultés durant ce TP, il faut savoir que l'algorithme de BFS (Breath-First Search) utilise une queue, alors que l'algorithme de DFS(Depth-First-search) utilise un stack. En gros, la manière dont un BFS parcours un arbre dépend du niveau de l'arbre, tandis que dans le cas d'un DFS, ce dernier parcourt un arbre dépendamment de sa profondeur. Une autre différence majeure est le fait qu'un BFS utilise une liste FIFO, au détriment du DFS qui lui utilise une LIFO.

Pour les questions 1 et 2, soit quel algorithme utilise le plus et le moins de mémoire, on a trouvé que DFS en utilise le plus, et BFS et l'algorithme de Joignabilité en utilise le moins. Ceci s'explique par le fait que dans le cas de traversée, BFS et l'algorithme de joignabilité prend le moins d'espaces possibles (parcours moins de cases), tandis que de l'autre côté, DFS prend le plus d'espaces possibles (parcours plus de cases).

Pour ce qui est des questions 3 et 4, DFS visite le plus de plus de Tile, suivi de BFS et l'algorithme de Joignabilité qui en utilise le moins. Ceci s'explique par le fait que le DFS est le plus rapide des trois algorithmes.

Bref, certes, ces trois algorithmes ont leurs propres avantages dépendamment de la situation ou elle se trouve, mais il peut être dit que concernant la rapidité de la traversé des éléments, il est préférable d'utiliser BFS au détriment des autres algorithmes. De l'autre côté, si l'on désire allouer le moins d'espace possible, il serait préférable d'utiliser un autre algorithme, outre que le BFS.