Minh-Hoang HUYNH

Test 5 : Application serveur 2

I. Présentation

Le programme s'appelle **`Test5_BestFirst.py`**.

> Utilisation
>
> - Lancer le terminal de commande
>   Avec Python 3 d'installé, exécuter le fichier avec la commande :
>   **`python /cheminVersLefichier/Test5_BestFirst.py`**
> - Le programme demandera le chemin vers un fichier contenant une grille et affichera une simulation graphique de la grille avec la propagation du feu.
> - Pour la grille 4, il se peut que le terminal soit trop petit et retourne une erreur, ne pas hésiter alors à réduire la taille de la police en dézoomant et agrandir le terminal.
> - Une option **`'--nodisplay'`** peut être passée en argument si l'on veut uniquement le résultat dans la console.

J'ai choisi une bibliothèque qui me permet un affichage graphique directement dans le terminal :

- **`curses`** pour un affichage graphique dans le terminal, qu'il faut installer avec le gestionnaire de package python : **`pip install windows-curses`**

Le temps passé sur les programme est d'environ 35h.

II. Logique de développement

1. Analyse du besoin

Besoin

On souhaite un programme pour résoudre un problème donné :

- Dans une grille représentant une forêt, un feu **F** se propage et le personnage **S** doit trouver le chemin vers une sortie **E** avant qu'elle soit en feu
- À chaque pas de temps le feu se propage sur les quatre cases voisines et le personnage **S** peut se déplacer sur les 4 cases voisines **'.'** t est bloquée par des arbre **'#'**
- La solution doit être sous la forme d'une liste de caractères représentant les directions choisies : URDL pour up, right, down, left. Par exemple une solution pour le problème ci-contre est :
  RRDDDDDLLLLLDDDDD

```
####################
#............#.....#
#............#.....#
#.........#######..#
#......E.....#F....#
#.............#....#
#.............#....#
######........######
#...S.........#....#
#...........#.#....#
#........############
#...........#......#
######......#......#
#...........#......#
#..................#
#..................#
#..................#
#..................#
#E.......#######...#
####################
```

Interface technique

Le programme s'exécute en ligne de commande et demande directement un fichier texte. Il lance une simulation graphique directement dans le terminal et retourne à la fin la solution du problème dans la console,

Livrable

*On souhaite un programme qui trouve une trajectoire vers une sortie **E** sans que **S** brûle.*

2. Architecture

Stratégie de développement

Le but est de rechercher un chemin avec comme exemple des grilles textes comme ci-dessus.

Les algorithmes de recherche de chemin se distingue en deux catégories, soit la recherche à l'aveugle (par la force brute aléatoirement) soit la recherche heuristique (avec des heuristiques, une valeur d'estimation à partir d'informations sur le problème).

- On choisit la **recherche heuristique** plus rapide en générale que la recherche à l'aveugle et avec **l'algorithme Best-First** qui a l'avantage de n'utiliser que l'heuristique.
  On prend comme heuristique :

$$h(S) = \max(\{SF + EF - 2 \times SE \mid pour\ tout\ feu\ F\ et\ toute\ sortie\ E\})$$

  avec SF : la distance euclidienne entre le personnage S et un feu F
  EF : la distance euclidienne entre une sortie et un feu
  SE : la distance euclidienne entre le personnage et une sortie

  On souhaite que S s'éloigne du feu et que la sortie E soit la plus loin possible, donc on maximise SF et EF. À l'inverse on souhaite que S se rapproche d'une sortie d'où le coefficient – pour le minimiser. Le coefficient 2 est là pour ajuster le comportement de manière « homogène ».

Néanmoins, cet algorithme ne permet que de trouver un chemin pour un état statique du problème, et ainsi ne peut pas prendre en compte la propagation du feu dynamique.

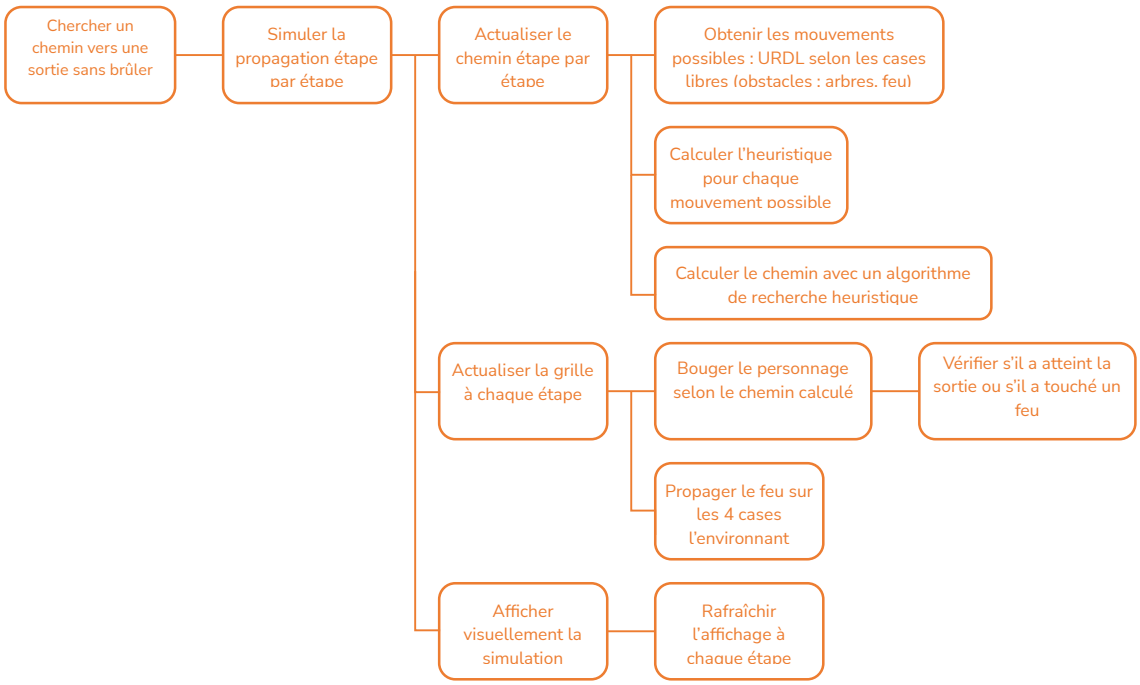
- C'est pourquoi on choisit de **simuler** la propagation du feu, étape par étape. À chaque pas de temps, la solution est recalculée pour prendre en compte l'état actuel de la propagation du feu et la position du personnage. Le personnage avance en fonction de la solution calculée.

Pour aider à la validation, on choisit d'afficher les grilles actualisées à chaque pas de temps

- On se donne un affichage des grilles avec la bibliothèque **`curses`** en Python qui permet un affichage direct dans le terminal de commande et permet d'afficher les grilles telles qu'elles sont dans le fichier texte.

Architecture fonctionnelle

Vue logique



Architecture technique

On choisit de coder le programme en Python 3, dans un seul fichier script par simplicité nommé **`Test5_BestFirst.py`**.

La bibliothèque **`curses`** utilise le terminal (powershell ou invite de commande) qui doit être redimensionné de façon à ce que la grille s'affiche.

III. Plan de validation

L'objectif est de trouver une trajectoire permettant au personnage d'atteindre une sortie sans brûler. Pour valider, on simule la propagation du feu et on affiche la solution en même temps que la propagation.

On effectue les tests de validation avec les 4 fichiers grids données. Il y a succès si le programme trouve une solution dans lequel le personnage atteint la sortie et échec si le personnage touche le feu.