E2 : CS 54 - Examen Machine Apprentis - 2022/202

TOP : Techniques and tOols for Programming – 1A

Novembre 2022

durée : 90 minutes.

Propos liminaires

Vous disposez de 2 heures pour réaliser tout ou partie de l'examen. Prenez le temps de lire les problèmes, de les comprendre, de réfléchir à des solutions. Il existe de multiples manières de résoudre les problèmes posés. Tentez, explorez. Aucune solution, si elle donne le bon résultat, n'est mauvaise !.

Veillez à respecter scrupuleusement les noms demandés pour chaque fichier python et assurez vous de bien déposer vos productions sur le dépôt git qui vous a été attribué pour l'examen, les évaluations étant automatiques.

Les seuls documents autorisés (en ligne ou sur papier) sont : une page web sur le projet git de l'examen, une page web sur la documentation officielle de python. Toute autre consultation sera considérée comme une fraude.

Ne copiez pas, ne trichez pas, ne récupérez pas de code d'internet. Les conséquences (conseil de discipline, exclusion, ...) sont trop lourdes pour s'y risquer.

Préparation de l'environnement de travail

Récupérez (par clonage) une version locale du projet git qui vous a été affecté pour cet examen.

À chaque étape de l'examen, nous vous demandons de déposer vos réalisations python sur le git du projet que vous venez de récupérer. Sur chaque exercice, nous vous imposons le nom du fichier python associé.

★ **Exercice 1:** Amazing ! (16 points)

Dans cet exercice, vous devrez écrire un certain nombre de fonctions python qui suivent chacune une signature imposée (nom de fonction et types des paramètres). **Toutes vos fonctions devront être implémentées dans un même fichier python dénommé : P1.py.**

En plein conflit mondial entre les rats des villes et les rats des champs, ces derniers ont miné des espaces afin de se protéger de l'exil des citadins. Ces derniers sont malins et ont volé les plans des espaces infectés. Vous devez aider ces derniers à traverser les champs hostiles. Vous disposez pour cela d'une photo satellitaire de chaque champs miné comme illustré dans la figure ci-dessous (celle-ci représente un champ de dimension 5 x 5 unités sur lequel les 1 représentent des cellules minées et les 0 des cellules "safe". La première ligne comprend les dimensions (lignes, colonnes) d'un champ ; la seconde ligne indique le point d'entrée dans le champ et la dernière, contient le point de sortie attendu (la cible).

```
5,5
1,1
3,5
0,1,1,1,1
0,0,0,1,0
1,0,1,0,0
0,0,0,0,1
1,1,1,1,1
```

FIGURE 1 – Exemple de représentation d'un champ miné dans un fichier de configuration

▷ **Question 1:** (1 pt) Ecrivez une fonction :

```
def loadField(fileName : str ) -> tuple[tuple[int,int], tuple[int,int], tuple[int,int], list[int]]:
```

qui renvoie :
— les dimensions du champs,
— le point d'entrée,
— le point de sortie,
— la modélisation du champ sous forme de liste d'entiers.

▷ **Question 2:** (1 pt) Ecrivez une fonction :

```
def nextOnes(position : tuple[int, int], longueur : int, largeur: int ) -> list[tuple[int, int]]:
```

qui, étant donnée une position courante d'un rat dans le champs (paramètre de la fonction) et des caractéristiques dun champs (longueur, largeur), calcule la liste des positions atteignables par celui-ci en un déplacement. On ne peut se déplacer que d'une case à chaque étape : vers le nord, le sud, l'est ou l'ouest et uniquement vers des cellules non minées.

▷ **Question 3:** (4 pts) Ecrire une fonction récursive optimale qui étant donné un champs, un point d'entrée et un point cible, renvoie *True* sur un chemin valide existe entre les deux points (les points d'entrée et de sortie sont fournis dans la description d'un champ.

Le profil de la fonction qui permet de réaliser cela est le suivant :

```
def cheminRat(champ: list[int], tailleChamps: tuple[int,int], pointEntree : tuple[int, int], cible: tuple[int, int]) -> bool :
```

▷ **Question 4:** (4 pts) Etendez votre première fonction pour qu'elle renvoie, lorsqu'une solution a été trouvée, le chemin parcouru par le rat depuis sa position de départ. Votre fonction aura le profil suivant :

```
def def cheminParcouruRat(echamp: list[int], position : tuple[int,int], PointEntree : tuple[int, int], Cible: tuple[int, int]) -> list[tuple[int, int]]
```

Le résultat renvoyé par la fonction est un tuple composé de la liste des positions successives prises par le rat pour atteindre la position finale.

▷ **Question 5:** (4 pts) Etendez votre première fonction pour qu'elle renvoie, le nombre de chemins possibles qui amènent à une solution :

```
def cheminsPossiblesRat(champ: list[int], position : tuple[int,int], PointEntree : tuple[int, int], Cible: tuple[int, int]) -> int:
```

▷ **Question 6:** (4 pts) Etendez votre première fonction pour qu'elle renvoie, si il existe, le chemin le plus court de la source à la destination :

```
def cheminPlusCourtRat(champ: list[int], position : tuple[int,int], PointEntree : tuple[int, int], Cible: tuple[int, int]) -> list[tuple[int, int]]
```

Votre programme doit se trouver dans le fichier `P1.py`. Votre programme doit être exécutable dans un terminal via la commande suivante :

`python3 P1.py playground<x>.txt`

ou x represente un entier quelconque, numéro du test

En entrée, votre programme doit lire le fichier `playground<x>.txt` conforme à la description donnée ci-dessus.

En sortie, il devra afficher successivement les résultats de la question 3,4 et 5 séparés par une ligne vierge.

★ **Exercice 2:** Complexité - (2 points)

Dans un fichier dénommé : complexite.txt indiquez la complexité de l'algorithme de la question 3.