INFO-F103 – Algorithmique 1
Backtracking Projet 1
Sequence

Stephane Fernandes Medeiros

Année académique 2012–2013

Énoncé

Nous vous demandons d'écrire un programme permettant de parcourir un labyrinthe. Ce parcours aura quelques particularités comme : le fait de commencer à une case donnée (pas nécessairement la case en haut à gauche), de se finir sur une autre case donnée (pas nécessairement la case en bas à droite), le chemin du point de départ au point d'arrivée devra passer par toutes les cases utilisables du labyrinthe sans passer deux fois sur la même case.

Votre programme chargera les grilles à résoudre à partir d'un fichier que l'utilisateur lui indiquera. Chaque ligne de ce fichier représentera une ligne du labyrinthe, de la même façon chaque colonne du fichier représentera une colonne du labyrinthe, les valeurs d'une ligne seront séparées par des virgules. La *source*, ou point de départ, sera le plus grand entier présent dans la grille. Le puits, ou point d'arrivée sera indiqué par un `0`. Les cases vides seront indiquées par des `x` tandis que les cases non utilisables seront indiquées par des `-`.

Le nombre dans la case source indique le début d'une séquence décroissante commençant par cette valeur et se finissant par 0. Une solution est trouvée s'il existe un chemin menant de la source au puits dont les valeurs successives seront la suite des nombres que représente la séquence décroissante du nombre indiqué à la source jusque 0.

Exemple :

Soit la grille suivante dans le fichier `grille1.txt` :

```
4 , x , x , x , 0
```

La séquence de code suivante :

```
S = Seq()
S.chargerGrille("grille1.txt")
S.trouvePuits()
S.afficherSolution()
```

Produira la solution suivante :

```
4 3 2 1 0
```

Les grilles fournies au programme pourront contenir une autre information : une valeur de passage imposée. Si une case de la grille (qui est utilisable) contient un autre nombre que 0, cela signifiera que votre chemin devra obligatoirement passer sur cette case avec la valeur imposée.

Exemple :

Soit la grille suivante dans le fichier grille1.txt :

```
8 , x , x
x , 0 , 5
x , x , x
```

La séquence de code suivante :

```
S = Seq()
S.chargerGrille("grille2.txt")
S.trouvePuits()
S.afficherSolution()
```

Produira la solution suivante :

```
8 7 6
1 0 5
2 3 4
```

Exemple avec cases non utilisables :

Soit la grille suivante dans le fichier grille1.txt :

```
17 , x , x , x
x , x , 0 , x
x , - , x , x
x , - , x , x
x , x , x , x
```

La séquence de code suivante :

```
S = Seq()
S.chargerGrille("grille2.txt")
S.trouvePuits()
S.afficherSolution()
```

Produira [1] la solution suivante :

```
17  16  15  14
2   1   0   13
3   -   11  12
4   -   10  9
5   6   7   8
```

Nous vous demandons d'écrire la classe `Seq`. Cette classe devra obligatoirement avoir les méthodes suivantes :

- `chargerGrille(self,nomDuFichier)` qui prend en paramètre un nom de fichier et va charger la grille correspondante,
- `trouvePuits(self)` qui va construire une solution menant de la source au puits et respectant les contraintes déjà mentionnées,
- `solutionExiste()` qui renverra `True` si une solution à la grille existe, `False` autrement,
- `afficheSolution()` qui affichera la solution si celle-ci existe, un message indiquant la non existence de celle-ci autrement.

Veillez à ne modifier ni le nom de la classe indiqué, ni le nom des méthodes imposées. En effet, votre programme sera testé à l'aide de test d'unités, dont une partie vous est fourni avec l'énoncé. Si pour une raison ou une autre votre programme ne devait pas fonctionner, une grosse pénalité sera appliquée à votre projet.

Nous attendons de votre part que vous appliquiez, lorsque nécessaire et justifié, les enseignements des chapitres vus en INFO-F101, INFO-F105 et INFO-F103.

Consignes

Les consignes quant à ce devoir sont celles reprises dans le document `http://www.ulb.ac.be/di/consignes_projets_2011-2012.pdf` à trois exceptions près : votre code doit faire bon usage des tests unitaires fournis, l'heure de remise est 15h30 **le lundi 25 mars 2013** et aucun retard n'est toléré. Tout manquement aux consignes ou à ces trois exceptions sera sanctionné directement d'un **0/10**. Le fichier des tests unitaires n'est pas à rendre avec votre projet, le correcteur aura sa propre version du fichier avec des tests additionnels. De plus un court rapport (2 pages de texte maximum) vous est demandé afin d'expliquer votre solution.

1. Notons que pour cet exemple, il existe plusieurs solutions.