INFO-F-203 : PROJET D'ALGORITHMIQUE 2
PARKING ESCAPE

Bernard Fortz Fabio Sciamannini Luciano Porretta Nikita Veshchikov

version du 4 novembre 2015

1 Le problème

Après avoir passé une journée au centre commercial, Mademoiselle Goal est impatiente de rentrer chez elle. Malheureusement, une mauvaise surprise l'attend dans le parking. D'autres voitures ont été garée de telle manière que sa voiture semble coincée dans le parking. Heureusement, tous les autres conducteurs ont oublié leurs clés dans leur voiture. Mlle Goal a donc une idée brillante : elle va essayer de déplacer les autres véhicules dans le bon ordre pour pouvoir sortir du parking avec sa voiture.

Il est à noter que la voiture de Mlle Goal ne peut se déplacer qu'à gauche ou à droite, tandis que les autres voitures ne peuvent bouger que suivant l'axe de leur orientation. De plus, pour déplacer un véhicule, la position vers laquelle il se déplace doit être libre.

Pouvez-vous aider Mlle Goal à sortir du parking ?

```
Situation de départ:
+---+---+---+---+---+
|     c1  c1        |
+   +   +   +   +   +
|         c2        |
+   +   +   +   +   +
| G   G   c2  c3    
+   +   +   +   +   +
|             c3    |
+   +   +   +   +   +
|                   |
+---+---+---+---+---+
```

FIGURE 1 – Fichier contenant la situation initiale

2 Objectif du projet

Nous vous demandons de produire un algorithme efficace pour résoudre ce problème. L'algorithme en question reçoit en entrée la carte du parking et l'emplacement de la voiture G et des autres voitures dans le parking.

L'algorithme doit produire :

- un fichier qui représente la situation initiale (voir Figure 1) ;
- la séquence des déplacements de la voiture Goal ;
- un fichier qui représente la situation finale et la liste des déplacements de la voiture Goal et des autres voitures (voir Figure 3) ;

Dans le cas où aucune solution ne permet à la voiture Goal de sortir du parking, nous vous demandons d'expliciter la cause pour laquelle elle est empêchée de sortir.

2.1 Une petite aide pour la résolution

Pour résoudre ce problème, nous vous conseillons de construire un graphe dans lequel chaque sommet représente une configuration valide de la grille de parking (sans véhicule qui se chevauchent), et les arêtes représentent un déplacement possible (un déplacement consiste en le déplacement d'une voiture dans une des deux directions autorisées, sans aller dans une position occupée). Dans ce graphe, deux sommets sont donc connectés si et seulement si ils représentent deux configurations qui diffèrent par un seul mouvement d'une voiture dans le parking.

3 Détails techniques

```
Parking: 5 fois 5
+---+---+---+---+---+
|                   |
+   +   +   +   +   +
|                   |
+   +   +   +   +   +
|
+   +   +   +   +   +
|                   |
+   +   +   +   +   +
|                   |
+---+---+---+---+---+
Elements du Parking:
         voiture Goal: 1
         Autres voitures: 3
Emplacements:
         voiture Goal: [(2,0), (2,1)]
         voiture 1: [(0,1), (0,2)]
         voiture 2: [(1,2), (2,2)]
         voiture 3: [(2,3), (3,3)]
```

FIGURE 2 – Format du fichier d'entrée

Le programme devrait être implémenté en JAVA en suivant au maximum le paradigme Orienté Objet (OO). De plus, il est conseillé de bien concevoir votre code afin de minimiser le code redondant en pensant aux responsabilités de chaque classe et de chaque méthode et à l'utilité de l'héritage et de la composition. *Un conseil : ne faites jamais de copier-coller.*
Vous pouvez utiliser les structures de données des libraires standard fournies par JAVA.

Le données décrivant le parking et les positions de tous les éléments présents dans le parking (voiture Goal et autres voitures) seront transmises à votre programme via la lecture d'un seul fichier (dont le nom sera passé en argument de votre programme). Le fichier contiendra la description du parking, composée :

- de la dimension du parking 5 fois 5 ;
- d'une représentation graphique du parking ;
- d'une section contenant les nombres des éléments du parking (nombre de la voiture Goal et autres voitures) ;
- d'une section contenant les emplacements des éléments du parking (position de la voiture Goal et des autres voitures) ;

Un exemple de fichier est montré dans la Figure 2.

Effectuez le *parsing* dans une ou plusieurs méthodes (pas dans le main) et placez-les dans les classes qui vous paraissent les plus à même d'effectuer ce travail (pensez aussi aux méthodes statiques). Vous pouvez supposer que le fichier d'entrée ne comporte pas d'erreur.

Votre programme doit recevoir le nom du fichier en argument et écrire sur l'output standard la solution de la manière montrée en Figure 4. En Figure 5 et Figure 6 vous avez un exemple de fichier et output, respectivement, à produire dans le cas où il n'y a pas de sortie au parking.

Utilisez les exceptions dans vos algorithmes en cas de problèmes (et évitez les retours de booléens). Votre code source sera évidemment accompagné d'un `makefile` permettant de compiler facilement votre projet.

Le projet peut être réalisé individuellement ou par groupe de deux (ce que nous recommandons).

Tout cela sera naturellement présenté dans un *rapport* à la présentation soignée. Pour rappel, un rapport est un *texte suivi en français* correct qui *décrit* votre travail. Cela va sans dire, le rapport fait partie *intégrale* de votre travail et une partie importante de la note finale portera sur le rapport (y compris sa présentation).

```
Situation finale:
+---+---+---+---+---+---+
| c1  c1  c2        |
+   +   +   +   +   +   +
|         c2        |
+   +   +   +   +   +   +
|             G   G
+   +   +   +   +   +   +
|             c3    |
+   +   +   +   +   +   +
|             c3    |
+---+---+---+---+---+---+
Une façon de sortir du Parking en 6 mouvements a été trouvée.

Déplacements car1:
[(0,1), (0,2)] -> [(0,0), (0,1)]
Déplacements car2:
[(1,2), (2,2)] -> [(0,2), (1,2)]
Déplacements car3:
[(2,3), (3,3)] -> [(3,3), (4,3)]
Déplacements voiture Goal:
[(2,0), (2,1)] -> [(2,1), (2,2)] -> [(2,2), (2,3)] -> [(2,3), (2,4)]
```

FIGURE 3 – Fichier contenant la situation finale

```
~>java  parking.txt

Le parking a un dimension 5 fois 5
Il contient 1 Goal car et 3 autres voitures
La voiture Goal se trouve en position: [(2,0), (2,1)]
La voiture 1 se trouve en position: [(0,1), (0,2)]
La voiture 2 se trouve en position: [(1,2), (2,2)]
La voiture 3 se trouve en position: [(2,3), (3,3)]

Déplacements effectués par la voiture 1:
1. [(0,1), (0,2)] Départ
2. [(0,0), (0,1)] ouest

Déplacements effectués par la voiture 2:
1. [(1,2), (2,2)] Départ
2. [(0,2), (1,2)] nord

Déplacements effectués par la voiture 3:
1. [(2,3), (3,3)] Départ
2. [(3,3), (4,3)] sud

 Déplacements effectués par la voiture Goal:
1. [(2,0), (2,1)] Départ
2. [(2,1), (2,2)] est
3. [(2,2), (2,3)] est
3. [(2,3), (2,4)] Sortie!


Une façon de sortir en 6 mouvements a été trouvée.
```

FIGURE 4 – Exemple d'execution ayant comme entrée le fichier en Figure 2

```
Situation finale:
+---+---+---+---+---+
|     c1  c1        |
+   +   +   +   +   +
|         c2        |
+   +   +   +   +   +
| G   G   c2  c3  c3
+   +   +   +   +   +
|                   |
+   +   +   +   +   +
|                   |
+---+---+---+---+---+
Il n'y a pas moyen de sortir du parking.

Déplacements effectués par la voiture 1: [(0,1), (0,2)]  [(0,0), (0,1)]

Déplacements effectués par la voiture 2: [(1,2), (2,2)]  [(0,2), (1,2)]

Déplacements effectués par la voiture Goal: [(2,0), (2,1)]  [(2,1), (2,2)]
```

FIGURE 5 – Fichier contenant la situation finale quand il n'y a pas de solution valide

```
~>java  parking.txt

Le parking a un dimension 5 fois 5
Il contient, 1 voiture Goal et 3 autres voitures.
La voiture Goal se trouve en position: [(2,0), (2,1)]
La voiture 1 se trouve en position: [(0,1), (0,2)]
La voiture 2 se trouve en position: [(1,2), (2,2)]
La voiture 3 se trouve en position: [(2,3), (3,3)]

Il n'y a pas moyen de sortir du parking
car la voiture 3 nous empêche de sortir.
```

FIGURE 6 – Exemple d'exécution quand il n'y a pas de solution valide

Consignes pour la remise du projet

À respecter scrupuleusement !

1. Les modalités pour la remise du rapport sont :
   - Date : **le 18 Décembre 2015** (vous pouvez ne pas remettre votre projet dans le cas de fin du monde).
   - Lieu : **Université Virtuelle (**`uv.ulb.a.c.be`**)** et **au Secrétariat « étudiants » du Département d'Informatique, local 2N8.104**.
   - Heure : **Avant 13h**.

   Après 13h, les projets seront considérés comme **en retard**, et vous perdrez **la moitié des points** sur votre note finale (plus un point par jour de retard).
2. Les modalité pour la remise du code sont :
   - Tous les fichiers dans **une seule archive** au format **ZIP** (ni tar.gz, ni rar, ni autre chose !).
   - L'archive doit porter **le nom** `AG2.NomEtudiant1-NomEtudiant2.zip`, si deux étudiants ont réalisé le projet ; `AG2.NomEtudiant.zip` dans le cas contraire.

   Si vous ne respectez pas ces critères, nous considèrerons que vous n'avez pas rendu de code. Si votre projet ne compile pas avec la commande `make`, on ne corrige pas votre projet.

Bon travail !