Implémentation d'un besoin client

M.Adam – N.Delomez – JF.Kamp – L.Naert

11 octobre 2022

Objectifs de la SAÉ

- Développer des applications informatiques simples
- **AC 1** Implémenter des conceptions simples
- **AC 3** Faire des essais et évaluer leurs résultats en regard des spécifications

Jeu de Grundy

Le jeu de Grundy est une variante du jeu de Nim inventée en 1939 par Patrick Grundy.

Le jeu se joue à deux joueurs. La position de départ consiste en un unique tas d'objets (des allumettes ou des pions, par exemple). Le seul coup disponible pour les joueurs consiste à séparer un tas d'objets en deux tas de tailles distinctes. Les joueurs jouent à tour de rôle, jusqu'à ce que l'un d'entre eux ne puisse plus jouer.

L'objectif de cette SAÉ est d'écrire un programme java qui permet de jouer des parties du jeu de Grundy.

Partie 1 : joueur contre joueur

Dans une premier temps, le programme doit permettre à deux joueurs de jouer l'un contre l'autre.

Déroulement d'une partie

Une partie se déroule souvent le scénario suivant :

1. saisie du nombre d'allumettes, 7 dans notre exemple
2. saisie du nom du joueur qui joue en premier,
3. saisie du nom du joueur qui joue en deuxième,
4. affichage du jeu :

```
0 : | | | | | | |
```

5. affichage du nom du joueur qui joue,
6. saisie de la ligne, 0 dans notre exemple, et du nombre d'allumettes du premier tas, 3 dans notre exemple,
7. affichage du jeu :

```
0 : | | |
1 : | | | |
```

8. changement de joueur,
9. reprise de la séquence d'affichage du jeu et du nom du joueur, de saisie de la ligne et du nombre d'allumettes,
10. arrêt du jeu quand aucune séparation n'est possible,
11. affichage du nom du vainqueur.

Un exemple complet est :

```
Nombre d'allumettes : 7
Nom du premier joueur : Adam
Nom du second  joueur : Michel
0 : |  |  |  |  |  |  |

Tour du joueur : Adam
Nombre d'allumettes à séparer : 2
0 : |  |
1 : |  |  |  |  |

Tour du joueur : Michel
Ligne : -1
Ligne : 0
Ligne : 2
Ligne : 1
Nombre d'allumettes à séparer : 0
Nombre d'allumettes à séparer : 1
0 : |  |
1 : |
2 : |  |  |  |

Tour du joueur : Adam
Ligne : 2
Nombre d'allumettes à séparer : 4
Nombre d'allumettes à séparer : 2
Nombre d'allumettes à séparer : 1
```

```
0 : |  |
1 : |
2 : |
3 : |  |  |

Tour du joueur : Michel
Ligne : 3
Nombre d'allumettes à séparer : 1
Michel a gagné.
```

Dans la version présentée, lors du premier coup, le numéro de la ligne n'est pas demandé car il n'y a qu'un seul tas. Il est possible également de ne pas demander le numéro de ligne quand seul un tas peut est divisé en deux.

Le programme doit comporter plusieurs méthodes. Chaque méthode doit posséder sa méthode de test.

A rendre

- le fichier source : Grundy1.java,
- la trace de l'exécution des méthodes de test,
- la trace de l'exécution d'une partie.

Les trois fichiers doivent être compressés dans une archive Nom_Prénom_Partie1.zip, par exemple Le-Guen_Olivier_Partie1.zip.

Partie 2 : joueur contre ordinateur

Dans un deuxième temps, le programme doit permettre à un joueur d'affronter l'ordinateur. Idéalement, le programme doit chercher à gagner. Mais comme expliqué plus loin, trouver une solution gagnante n'est pas facile. Il est conseillé de procéder en plusieurs parties :

1. le programme choisit une décomposition au hasard ;
2. le programme ne cherche une solution gagnante qu'en fin de partie ;
3. le programme choisit quand c'est possible une stratégie gagnante.

Déroulement d'une partie

Une partie se déroule souvent le scénario suivant :

1. saisie du nom du joueur,
2. choix de celui qui joue en premier joueur ou ordinateur,
3. affichage du jeu :

```
0 : |  |  |  |  |  |  |  |  |  |  |  |  |  |
```

4. affichage du nom du joueur qui joue,
5. saisie de la ligne et du nombre d'allumettes à séparer s'il s'agit du joueur.
6. changement de joueur,
7. reprise de la séquence d'affichage du jeu et du nom du joueur, de saisie de la ligne et du nombre d'allumettes à séparer,
8. arrêt du jeu quand un des deux joueurs ne peut jouer,
9. affichage du nom du vainqueur.

Un exemple complet :

```
Nombre d'allumettes : 11
Type de partie :
- 0 joueur contre joueur
- 1 joueur contre ordinateur
1

Nom du joueur : Adam
Premier à jouer :
- 0 l'ordinateur
- 1 le joueur
1
0 : |  |  |  |  |  |  |  |  |  |  |

Tour du joueur : Adam
Nombre d'allumettes à séparer : 1
0 : |
1 : |  |  |  |  |  |  |  |  |  |

Tour du joueur : Ordi
0 : |
1 : |
2 : |  |  |  |  |  |  |  |  |

Tour du joueur : Adam
Ligne : 2
Nombre d'allumettes à séparer : 3
0 : |
1 : |
2 : |  |  |
3 : |  |  |  |  |  |
```

```
Tour du joueur : Ordi
0 : |
1 : |
2 : |
3 : |  |
4 : |  |  |  |  |  |

Tour du joueur : Adam
Ligne : 4
Nombre d'allumettes à séparer : 2
0 : |
1 : |
2 : |
3 : |  |
4 : |  |
5 : |  |  |  |

Tour du joueur : Ordi
0 : |
1 : |
2 : |
3 : |  |
4 : |  |
5 : |
6 : |  |  |

Tour du joueur : Adam
Ligne : 6
Nombre d'allumettes à séparer : 1
0 : |
1 : |
2 : |
3 : |  |
4 : |  |
5 : |
6 : |
7 : |  |
Adam a gagné.
```

A rendre

- un document pdf contenant l'explication de la stratégie utilisée par l'ordinateur,
- le fichier source : Grundy2.java,
- la trace de l'exécution des méthodes de test,

- la trace de l'exécution d'une partie.

Les quatre fichiers doivent être compressés dans une archive Nom_Prénom_Partie2.zip, par exemple Le-Guen_Olivier_Partie2.zip.

Critères d'évaluation

La SAÉ sera évaluée suivant les critères suivants :

- respect du format du rendu, nom du fichier, et compression en zip,
- respect de la date de rendu,
- respect des conventions java,
- décomposition en méthodes,
- méthodes de test,
- exécution correcte et sans erreur du programme,
- qualité du code rendu.

Cette liste n'est pas exhaustive et d'autres critères pourront être ajoutés au choix par les enseignants de la SAÉ.

Stratégie gagnante

Trouver une stratégie gagnante au jeu de Grundy n'est pas simple.

Évidemment, l'ordinateur pourrait jouer au hasard. C'est une solution, mais il serait préférable que l'ordinateur cherche le coup gagnant, s'il existe.

La solution dévéloppée par les enseignants emploie le principe suivant :

> à partir d'une position du jeu, le programme regarde toutes les décompositions possibles ;
>
> si une des décompositions est perdante pour l'adversaire, le programme joue ce coup ;
>
> si aucune des décompositions n'est perdante pour l'adversaire, il joue un coup au hasard.

Une position est dite perdante :

> l'autre joueur ne peut plus jouer ;
>
> quel que soit le coup de l'adversaire, il sera possible de rejouer pour le mettre dans une position perdante.

Par exemple la situation suivante est perdante :

```
0 : |
1 : |
2 : |  |
3 : |
4 : |  |
5 : |
6 : |  |
```

Par exemple la situation suivante est aussi perdante :

```
0 : |
1 : |
2 : |  |  |
3 : |  |  |  |  |  |
```

Evidemment c'est plus complexe à déterminer.

Soit J1 le joueur qui a effectué la dernière séparation et J2 le joueur qui doit jouer. Il n'existe que trois possibilités à J2 :

1. Séparer la ligne 2 :
   ```
   0 : |
   1 : |
   2 : |
   3 : |  |
   4 : |  |  |  |  |  |
   ```
   Le joueur J1 sépare la ligne 4 en deux tas de 2 et de 4. Le joueur J2 ne peut que séparer la ligne 4 en deux tas de 1 et de 3 qui est une situation perdante. Le Joueur J1 sépare le tas de 3 et bloque J2 qui perd.
2. Séparer la ligne 4 en deux tas de 1 et 5 :
   ```
   0 : |
   1 : |
   2 : |  |  |
   3 : |
   4 : |  |  |  |  |
   ```
   Le joueur J1 sépare la ligne 4 en deux tas de 2 et 3 qui est une situation perdante pour J2.
3. Séparer la ligne 4 en deux tas de 2 et 4 :
   ```
   0 : |
   1 : |
   2 : |  |  |
   3 : |  |
   4 : |  |  |  |
   ```
   Le joueur J1 sépare la ligne 4 en deux tas de 1 et 3 qui est une situation perdante pour J2.

Dans tous les cas le joueur J2 qui partira de cette position perdra si l'adversaire joue bien.

Par contre, la situation suivante est une position gagnante :

```
0 : |
1 : |
2 : |  |
3 : |
4 : |  |  |
5 : |
6 : |  |
```

Il suffit de jouer le seul coup jouable en séparant le ligne 4 et l'adversaire est bloqué.

Pour dévélopper cette solution, il est souhaitable de lire les deux documents suivants :

Le cours R1.01 sur la récursivité : `https://tinyurl.com/S101Doc01`. Ce cours présente la notion de recursité et surtout explique comment résoudre le problème de la recherche de la sous-somme maximale dans un tableau d'entiers. La recherche de la solution est récursive et peut servir de modèle

La page du site Maths en jeans sur la jeu de grundy : `https://tinyurl.com/S101Doc02`. Cette page explique les notions de situations perdantes et gagnantes, et leur composition. Par exemple une situation gagnante et une situation perdante donnent une situation gagnante. Une situation perdante et une situation perdante donnent une situation perdante. Par contre, une situation gagnante et une situation gagnante donnent une situation indéterminée, gagnante ou perdante.

Le programme de recherche de la meilleure solution est à classer dans la catégorie des programmes NP-complets. Le calcule de toutes les solutions possibles est très consommateur en temps. A titre d'exemple, sur les ordinateurs des enseignants, la recherche de la première solution avec 26 allumettes prend plus d'une minute et près de 5 minutes avec 27 allumettes. Le deuxième document doit permettre de ne pas devélopper toutes les branches quand elles sont perdantes et d'accélerer la prise de décisons.