

Martial DUVERNEIX, Florian GAUTIER,
Pierre LORSON, Teiki PEPIN

Solveurs et générateurs pour des jeux de logique II

Client : Emmanuel FLEURY
Chargé de TD : Celestin LANTERNE

Besoins fonctionnels

- Permettre l'affichage d'une grille de jeu ;
- Permettre la lecture de grilles à partir de fichiers ;
- Mettre en place un système de résolution de grilles ;
- Mettre en place un système de génération de grilles ;
- Possibilité d'écriture en sortie sur fichier ;
- Implémenter différentes options d'exécution.

Besoins non fonctionnels

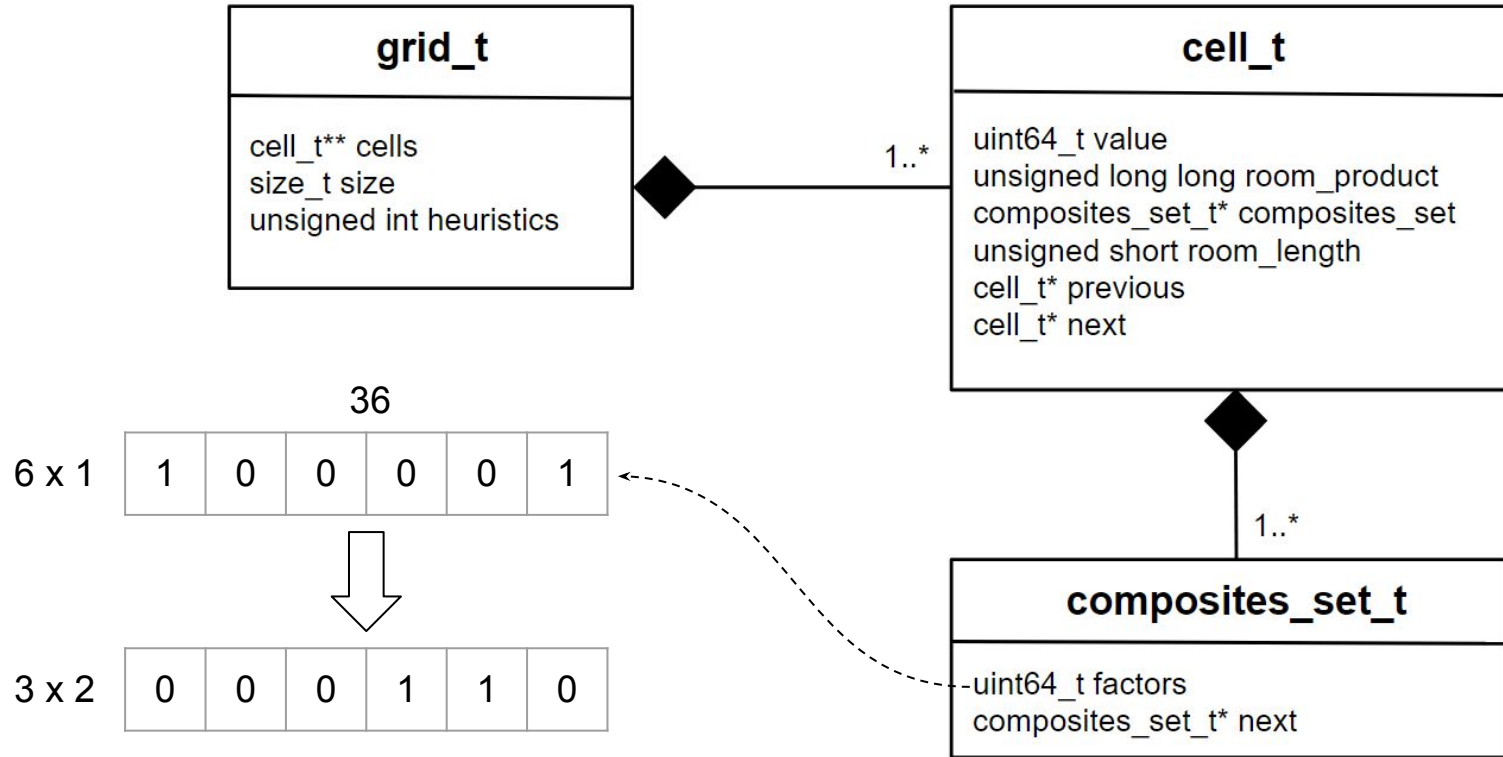
- Chaque jeu fera l'objet d'un exécutable ;
- Respect de règles de codage définies par le client ;
- Robustesse de la lecture ;
- Optimisation des performances : utilisation du C, algorithmes efficaces, structure de données permettant le stockage bit à bit, algorithmes SWAR, multithreading.

Présentation du jeu Inshi no Heya

- Une grille carrée avec des cases vides réparties en salles.
- Chaque salle a un nombre qui lui est associé.
- Remplir les cases avec des chiffres (1 à N) de telle façon à ce que le produit d'une salle soit égal au nombre associé.
- Aucun chiffre ne peut être répété en ligne ou en colonne.

| | | | | |
|----------------|-----------------|---|----|----|
| 6 | 15 | | 1 | 12 |
| | ²⁰ 5 | 4 | 8 | |
| 10 | | 6 | | |
| ⁴ 1 | ⁴ 4 | | 15 | |
| 4 | ¹ 1 | | 10 | |

Architecture du jeu Inshi no Heya



Fonctionnement du solveur d'Inshi no Heya

Étapes d'exécution du solveur :

- Lecture du fichier ;
- Décomposition des produits des salles ;
- Remplir les cases à possibilité unique ;
- Détermination des autres cases par backtracking ;
- Vérifier la validité de la grille ;
- Écriture de la grille en sortie.

| | | | | |
|---|----|----|---|----|
| | | 15 | 1 | |
| 6 | | 20 | | |
| | 10 | | 8 | 12 |
| | 4 | | | 15 |
| 4 | 1 | 6 | | 10 |

| | | | | | |
|-----|-------|--|-----|--|-----------|
| 254 | ----- | | | | |
| 255 | 2 | | 3 | | 5 1 4 |
| 256 | | | --- | | --- |
| 257 | 3 | | 5 | | 4 2 1 |
| 258 | | | --- | | --- |
| 259 | 5 | | 2 | | 1 4 3 |
| 260 | | | --- | | --- |
| 261 | 1 | | 4 | | 2 3 5 |
| 262 | | | --- | | --- |
| 263 | 4 | | 1 | | 3 5 2 |
| 264 | ----- | | | | |

```

1 5x5
2 6b2
3 15r2
4 1r1
5 12b3
6 20r2
7 8b2
8 10r2
9 6b3
10 4b2
11 4r1
12 15r2
13 1b1
14 10r2
    
```

Factorisation par le solveur d'Inshi no Heya

Factorisation du
produit

Sous
factorisation
récursive

Filtre par
longueur

Tri à bulles

Suppression
des doublons

36
36 x 1
18 x 2
18 x 2 x 1
12 x 3
12 x 3 x 1
9 x 4
9 x 4 x 1

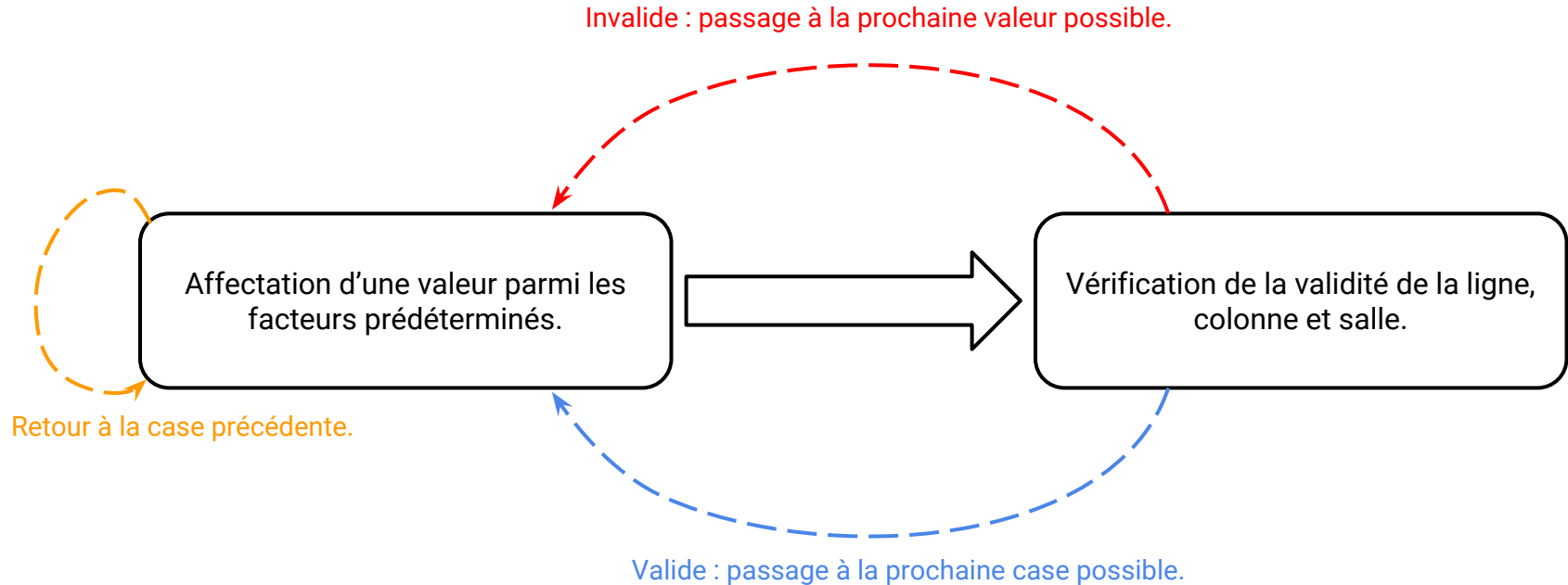
36
36 x 1
18 x 2
6 x 3 x 2
18 x 2 x 1
6 x 3 x 2 x 1
12 x 3
6 x 2 x 3
12 x 3 x 1
6 x 2 x 3 x 1
9 x 4
9 x 4 x 1

6 x 3 x 2
18 x 2 x 1
6 x 2 x 3
12 x 3 x 1
9 x 4 x 1

6 x 3 x 2
6 x 3 x 2
9 x 4 x 1
12 x 3 x 1
18 x 2 x 1

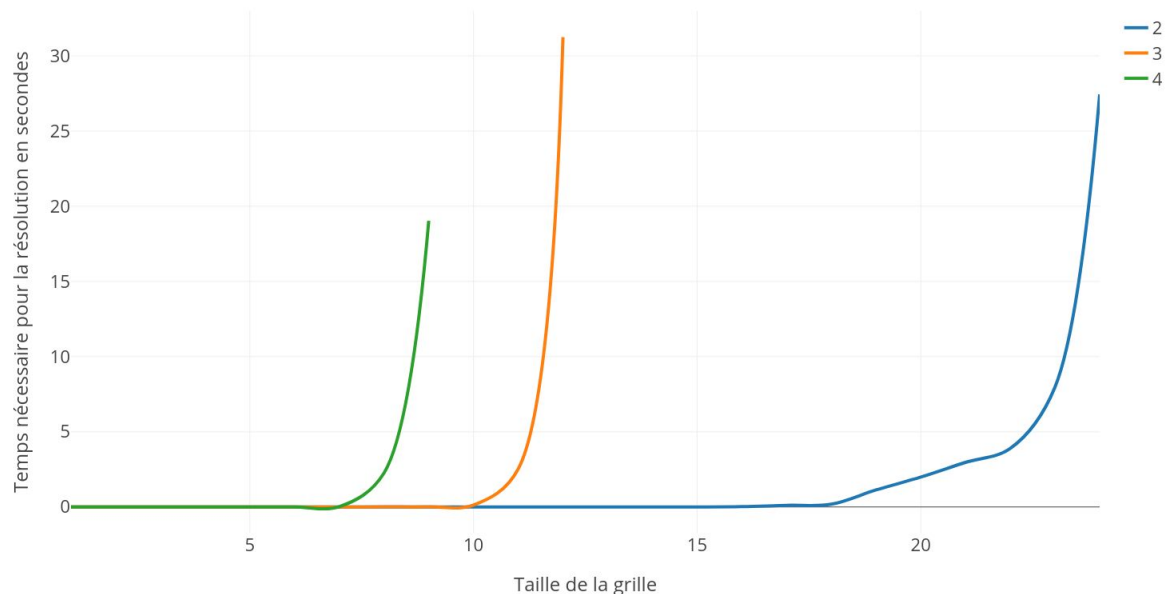
6 x 3 x 2
9 x 4 x 1
12 x 3 x 1
18 x 2 x 1

Backtracking par le solveur d'Inshi no Heya



Analyse du solveur d'Inshi no Heya

```
Command being timed: "./inshiNoHeya ../test/32x32_Generated"  
User time (seconds): 966.67  
System time (seconds): 0.00  
Percent of CPU this job got: 99%  
Elapsed (wall clock) time (h:mm:ss or m:ss): 16:06.96
```



Fonctionnement du générateur d'Inshi no Heya

Étapes d'exécution du générateur :

- Initialisation bit à bit de la grille ;
- Affectation de valeurs par backtracking ;
- Délimitation des salles ;
- Écriture en sortie.

```
tepepin@trelawney:~/PdP/logicgamessolver/trunk/InshiNoHeya/src$ ./inshiNoHeya -g 5
```

| | | | | |
|---|---|---|---|---|
| 4 | 1 | 5 | 3 | 2 |
| 1 | 2 | 4 | 5 | 3 |
| 2 | 5 | 3 | 4 | 1 |
| 3 | 4 | 2 | 1 | 5 |
| 5 | 3 | 1 | 2 | 4 |

```
tepepin@trelawney:~/PdP/logicgamessolver/trunk/InshiNoHeya/src$ ./inshiNoHeya -g 5 -i grille
tepepin@trelawney:~/PdP/logicgamessolver/trunk/InshiNoHeya/src$ cat grille
5x5
4r1
6r2
8b3
5r1
6b3
5b2
5r1
3b2
8b3
3r1
20r2
5r1
4r1
6r2
tepepin@trelawney:~/PdP/logicgamessolver/trunk/InshiNoHeya/src$
```

Analyse du générateur d'Inshi no Heya

Grande majorité du temps passé en sein de l'attribution des valeurs des cases par backtracking.

```
Each sample counts as 0.01 seconds.
```

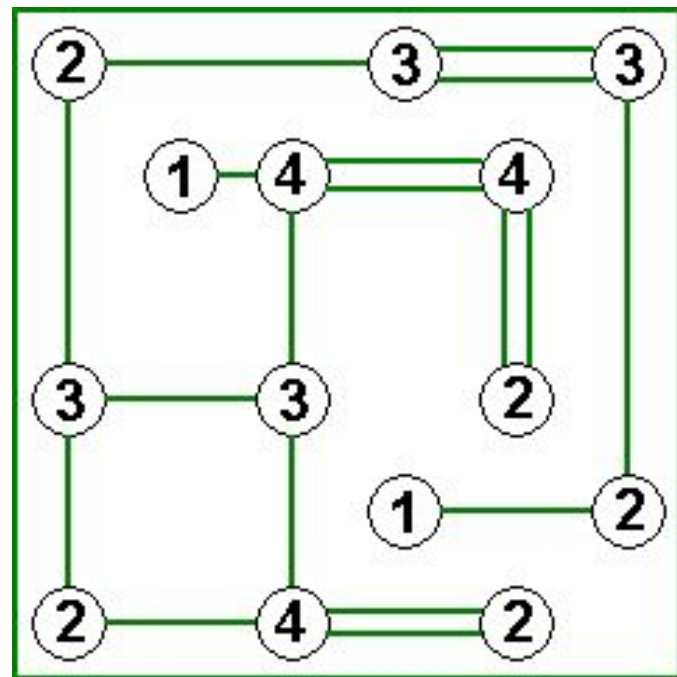
| % time | cumulative seconds | self seconds | calls | self ms/call | total ms/call | name |
|-----------|-----------------------|-----------------|----------|-----------------|------------------|----------------------|
| 58.05 | 0.47 | 0.47 | 1 | 470.23 | 770.37 | randomize_cell_value |
| 25.94 | 0.68 | 0.21 | 26513251 | 0.00 | 0.00 | check_bit |
| 8.65 | 0.75 | 0.07 | 2773836 | 0.00 | 0.00 | set bit |

Défauts pour Inshi no Heya

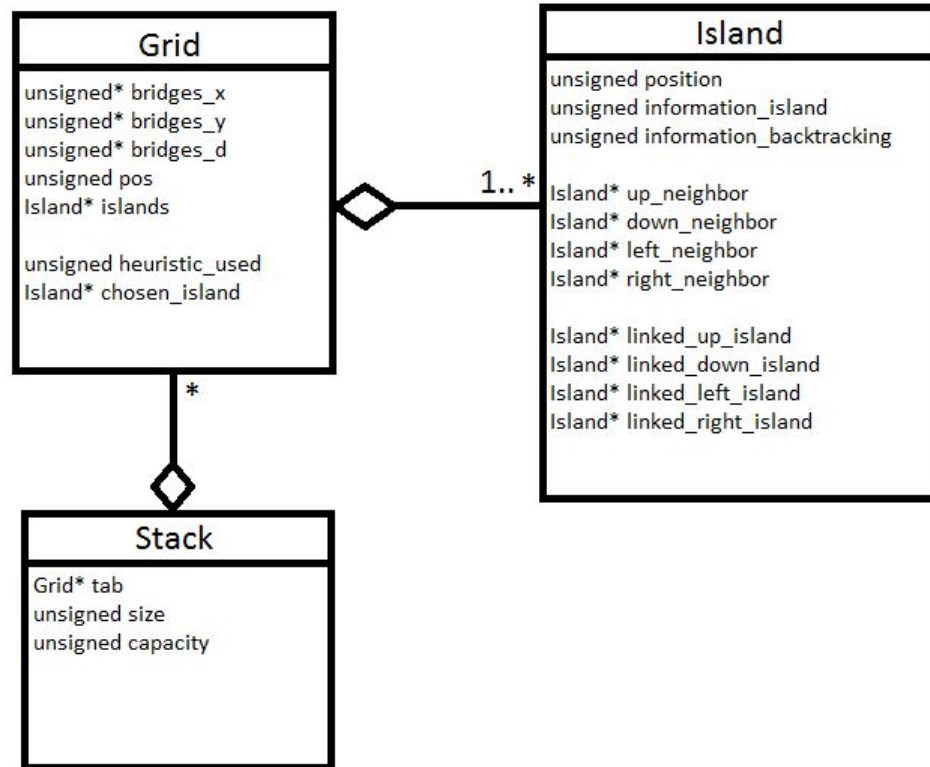
- Algorithme utilisé pour la décomposition en facteurs et leur tri insuffisant en performance pour des produits importants.
- Lenteur de la résolution sur des grilles de grande taille dû au nombre d'itérations au sein de la phase de backtracking.
- Taille limite de grille acceptée par notre architecture et nos algorithmes fixée à 64x64.

Présentation du jeu Hashiwokakero

- Les îles ont une valeur de 1 à 8.
- Les ponts relient les îles entre elles.
- Les ponts ne doivent pas se croiser.
- Les ponts sont tracés en ligne droite.
- Chaque île doit avoir un nombre de ponts équivalent à sa valeur.
- Les îles doivent toutes être reliées entre elles directement ou indirectement.



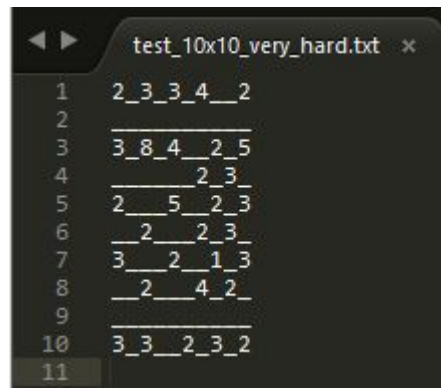
Architecture du jeu Hashiwokakero



Fonctionnement du solveur d'Hashiwokakero

Étapes d'exécution du solveur :

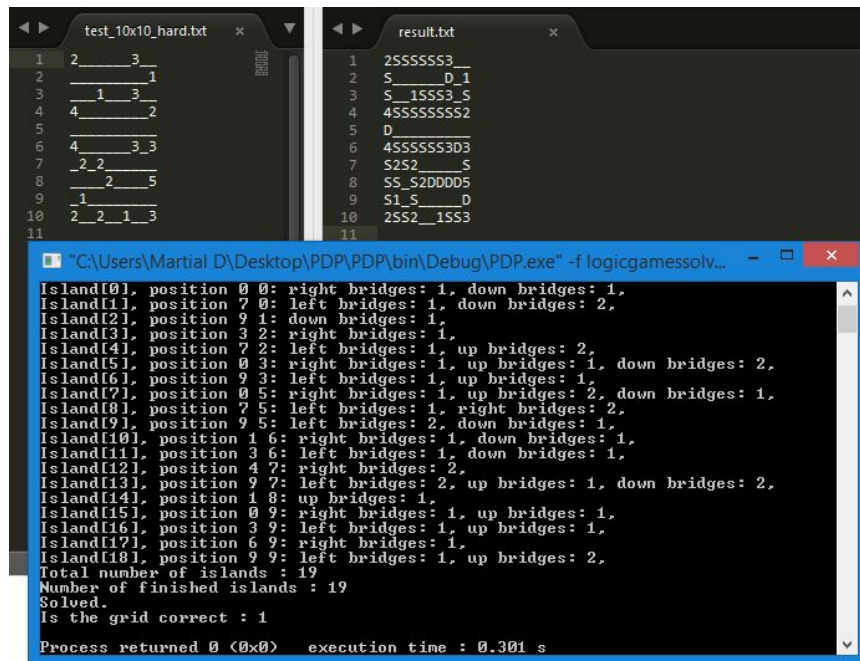
- Lecture du fichier ;
- Détermination des voisins de chaque île ;
- Utilisation des heuristiques ;
- Détermination des autres ponts par backtracking ;
- Écriture de la solution en sortie.



Résultat du solveur d'Hashiwokakero

Un exemple de fichier de grille :

- ‘_’ est une case vide ;
- Un chiffre symbolise une île avec sa valeur.



The screenshot displays three windows from a Hashiwokakero solver application:

- test_10x10_hard.txt**: Contains a 10x10 grid puzzle. Islands are represented by numbers (2, 3, 1, 4, 2, 3, 5, 1, 3) and empty cells by underscores.
- result.txt**: Shows the solved grid. 'S' represents a single bridge and 'D' represents a double bridge. For example, the first row is "2SSSSSS3", indicating a double bridge between the island of 2 and the island of 3.
- Console**: Displays the solver's internal logic and results. It lists 19 islands with their positions and the number of bridges connected in each direction (up, down, left, right). It confirms the puzzle is solved and the grid is correct.

Un exemple de fichier de solution :

- ‘S’ est un pont simple ;
- ‘D’ est un pont double.

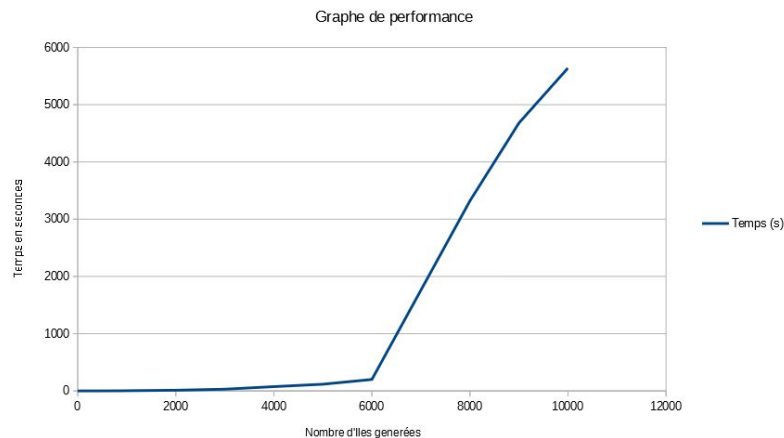
Un exemple de sortie sur la console.

Il faut spécifier le fichier d'entrée avec l'option -f.

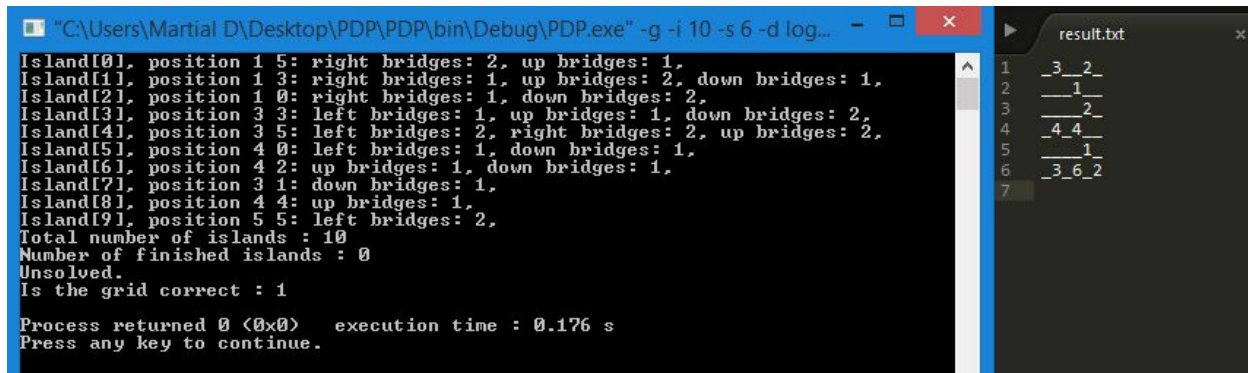
Fonctionnement du générateur d'Hashiwokakero

Étapes d'exécution du générateur :

- Choix aléatoire de la position de la première île ;
- Choix aléatoire d'une île, d'une direction et d'une distance pour tracer une nouvelle île ou un nouveau pont ;
- Si place insuffisante, on recommence en limitant la distance ;
- Écriture de la grille dans un fichier.



Résultat du générateur d'Hashiwokakero



```
"C:\Users\Martial D\Desktop\PDP\PDP\bin\Debug\PDP.exe" -g -i 10 -s 6 -d log...
Island[0], position 1 5: right bridges: 2, up bridges: 1,
Island[1], position 1 3: right bridges: 1, up bridges: 2, down bridges: 1,
Island[2], position 1 0: right bridges: 1, down bridges: 2,
Island[3], position 3 3: left bridges: 1, up bridges: 1, down bridges: 2,
Island[4], position 3 5: left bridges: 2, right bridges: 2, up bridges: 2,
Island[5], position 4 0: left bridges: 1, down bridges: 1,
Island[6], position 4 2: up bridges: 1, down bridges: 1,
Island[7], position 3 1: down bridges: 1,
Island[8], position 4 4: up bridges: 1,
Island[9], position 5 5: left bridges: 2,
Total number of islands : 10
Number of finished islands : 0
Unsolved.
Is the grid correct : 1

Process returned 0 (0x0)   execution time : 0.176 s
Press any key to continue.
```

result.txt

```
1  _3_2_
2  _ _1_
3  _ _2_
4  _4_4_
5  _ _1_
6  _3_6_2
7
```

Un exemple de sortie sur la console grâce à l'option verbose -v.

Il faut spécifier :

- le mode génération avec -g
- le nombre d'îles voulu avec -i
- la taille de la grille avec -s
- le chemin du fichier où sauvegarder la grille avec -d.

Défauts sur Hashiwokakero

- Le solveur accepte les îles côte à côte ce qui est incorrect. Le générateur, lui, n'en générera pas.
- Le générateur ne réussit pas toujours à trouver assez de place pour tracer le nombre d'îles demandé.
- Le backtracking ne fonctionne pas dans tous les cas et fait parfois une boucle infinie : le backtracking ne réussit pas à parcourir l'arbre des possibilités.

Conclusion

Extensions possibles pour **Inshi no Heya** :

- Ajout d'heuristiques pour améliorer les performances ;
- Mettre en place un algorithme de factorisation plus performant pour de larges valeurs ;
- Remplacer uint64_t par une nouvelle structure de donnée adaptée aux grilles de taille supérieure à 64.

Extensions possibles pour **Hashiwokakero** :

- Ajout d'heuristiques réutilisables afin d'améliorer la rapidité et le nombre de grilles résolubles ;
- Ajout du générateur unique : tracer seulement des ponts qui peuvent être trouvés par les heuristiques ;
- Ajout de la vérification que la solution forme un seul groupe.