Le solitaire V2

# Présentation

Jeu de plateau à un joueur

Le plateau est constitué d’un ensemble de cases carrées jointives par les bords.

Il existe principalement deux configurations :

|  |  |
| --- | --- |
| Le plateau classique (33 cases) | Le plateau français (37 cases) |
| |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | |  |  |  |  |  |  |  | |  |  |  |  |  |  |  | |  |  |  |  |  |  |  | |  |  |  |  |  |  |  | |  |  |  |  |  |  |  | |  |  |  |  |  |  |  | |  |  |  |  |  |  |  | | |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | |  |  |  |  |  |  |  | |  |  |  |  |  |  |  | |  |  |  |  |  |  |  | |  |  |  |  |  |  |  | |  |  |  |  |  |  |  | |  |  |  |  |  |  |  | |  |  |  |  |  |  |  | |

Au départ, chaque case est occupée par une pierre, sauf au moins une case.

Dans les jeux classique et français, seule la case centrale est vide.

Les cases contigües sont celles qui partagent un même bord. Il n’y a pas de mouvements en diagonale.

Un mouvement consiste à choisir une pierre A et à la faire sauter par-dessus une pierre B contigüe pour autant que A atterrisse dans une case vide du plateau. Alors B est retiré. Cela revient à déterminer trois cases contiguës alignées, les deux premières contenant une pierre, la troisième à une des extrémités est vide, et d’inverser la présence des pierres dans ces trois cases.

Si on note x la présence d’une pierre, o une case sans pierre, les 4 mouvements de base sont :

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| x x o | → | o o x |  | o x x | → | x o o |  | x x o | → | o o x |  | o x x | → | x o o |

Chaque mouvement fait donc décroitre de exactement 1 unité le nombre de pierres sur le plateau.

L’objectif est d’arriver à ne conserver qu’une seule pierre sur le plateau.

La position de la dernière pierre n’est pas déterminante.

S’il y a N pierres sur le plateau initial, il faut exactement N-1 mouvements pour atteindre une solution (si elle existe).

# Objectif

Evidemment, l’intérêt du jeu est d’être résolu manuellement. Mais je ne suis pas doué. Et je me pose une question existentielle : y a-t-il une solution quelle que soit la pierre de départ retirée d’un plateau complet ?

Bon, le plateau classique ne m’offre que peu de résistance. Le Français par contre se révèle plus ardu.

Lançons-nous dans l’écriture d’un programme de recherche automatique de solutions.

Le volume des situations se révèle très élevé. La mémoire du PC explose.

En fait, la situation « classique » peut être résolue naïvement, et plus rapidement encore dès qu’on tient compte des symétries. Mais le plateau Français est plus coriace.

D’où la stratégie suivante :

* Tenter de gérer une mémoire des situations déjà rencontrées pour ne pas lancer de recherches redondantes.
* Exploiter les symétries pour encore diminuer les redondances.
* Minimiser l’empreinte de chaque situation pour optimiser l’utilisation de la mémoire.

Et surtout, devant les échecs répétés des premières tentatives, mettre au point la stratégie suivante :

* Pour un plateau donné, calculer la liste des situations (aux symétries près) où une pierre manque exactement, appelons cet ensemble EI.
* A partir des situations de EI, calculer toutes les situations (aux symétries près) possibles en jouant un nombre déterminé de coups, nombre nommé ND, par exemple 9. L’ensemble est nommé ED. Stocker cet ensemble dans un fichier réutilisable ED.dat .
* Calculer l’ensemble des situations gagnantes où il n’y a qu’une pierre sur le plateau (aux symétries près), ensemble qu’on nomme EG.
* Calculer à partir des situations de EG toutes les situations dites finales qui peuvent mener en un nombre donné de mouvements, nombre nommé NF, par exemple 12, à l’une de ces situations gagnantes (aux symétries près). L’ensemble est noté EF. Elles sont stockées dans un fichier EF.dat .
* Mettre au point un programme qui sait répondre à une sollicitation pour s’arrêter en sauvegardant son travail en cours. Ce programme peut reprendre à partir de cette position, mais en ayant libéré toute la mémoire des situations intermédiaires qu’il doit recalculer.
* Ce programme teste une à une les situations de ED. Et donc sait enregistrer la situation actuelle lors d’une demande d’arrêt pour pouvoir la reprendre au redémarrage.
* Pour chaque situation de ED, disons SD, s’il trouve un ensemble de mouvements qui mène à une situation SF de EF (aux symétries près), alors il recherche les situations de EI pour lesquelles une suite de mouvements existe qui mène à SD, ainsi qu’une suite de mouvements qui mènent de SF à une situation de EG (aux symétries près).
* En combinant ces trois suites de mouvements, (il faut tenir compte que passer de EI à ED peut mener à une situation absente de ED, seulement équivalente par symétrie, et donc que les mouvements doivent être retravaillés) on reconstitue une suite de mouvements qui mène au moins une, sinon d’autres, situation de EI à une situation de EG. Le programme enregistre ces situations de EI résolues et la suite de mouvements qu’il a trouvée pour sa résolution.
* Ces situations de EI résolues sont flaggées dans EI. Si toutes sont flaggées, alors le programme a trouvé une solution quelle que soit la situation initiale de l’ensemble EI d’origine.

EI

ED

EF

EG

ND mouvements

NF mouvements

N pierres initialement sur le plateau de N+1 cases, ou plus

N – ND pierres

dans les situations

de ED

NF + 1 pierres

dans les situations

de EF

N – ND – NF – 1

mouvements

séparent ED de EF

## Fiche de jeu

C’est un fichier XML décrivant une situation initiale et les deux paramètres ND et NF souhaités.

Exemple de contenu de fiche de jeu :

<solitaire>

<parametres nd="9" nf="12" />

<plateau>

xxx

xxxxx

xxxxxxx

xxxxxxx

xxxxxxx

xxxxx

xxx

</plateau>

</solitaire>

Une situation est une suite de lignes non vides constituées d’espaces ‘ ‘, de lettres ‘x’ ou ‘o’.

Le ‘x’ désigne une pierre, le ‘o’ une case vide.

L’algorithme calcule automatiquement la largeur du plateau L.

## Données persistantes

Le programme constitue initialement une banque de données, répertoire dans lequel il stocke une copie de la fiche de jeu, qu’il complète au fur et à mesure avec l’information lui permettant de reprendre le travail, et les solutions trouvées. Il y stocke aussi les fichiers ED.dat et EF.dat qu’il ne recalcule pas lors des reprises.

### Fichier Pilote

Exemple de contenu de « fichier pilote » :

Lors d’une reprise, les paramètres parametres/@nd et parametres/@nf, ainsi que la description du plateau initial doivent être identiques à ceux de la fiche de jeu.

Le paramètre reprise/@idx pointe sur la position de l’enregistrement de situation dans le fichier ED.dat à partir duquel reprendre les recherches.

<solitaire>

<parametres nd="9" nf="12" />

<plateau>

xxx

xxxxx

xxxxxxx

xxxxxxx

xxxxxxx

xxxxx

xxx

</plateau>

<reprise idx="999999999" />

<solution>

<plateau>

oxx

xxxxx

xxxxxxx

xxxxxxx

xxxxxxx

xxxxx

xxx

</plateau>

<mouvement c="2" dir="o" />

<mouvement c"16" dir="n" />

</solution>

</solitaire>

Une solution est une suite de mouvements à effectuer pour résoudre le jeu.

Un mouvement est un couple (c, dir). ‘c’ est l’indice de la pierre déplacée, dir est la direction dans laquelle est réalisé le mouvement. Les directions sont o, n, e, s pour ouest, nord, est et sud.

Lors d’une reprise, les situations de EI sont recalculées, et comparées aux situations des éventuelles situations déjà trouvées (aux symétries près) afin de déterminer quelles situations n’ont pas encore trouvé de solution.

## Description des situations

Le plateau est inscrit dans un rectangle minimal implicite, appelé rectangle encadrant. Ce rectangle est divisé en carrés dans certains desquels se situent les cases du plateau. Ces carrés sont numérotés à partir de 0, de gauche à droite, à partir du haut. Le nombre de cases du rectangle encadrant est inférieur à 128, aussi on utilise le byte pour coder les indices de ses cases.

Le plateau initial est décrit par

* La largeur de ce rectangle implicite.
* La suite des indices des cases qui le composent.
  + On ordonne cette suite pour faciliter les comparaisons.

Toute situation est décrite par

* La suite des indices des cases occupées par une pierre
  + On ordonne cette suite pour faciliter les comparaisons.

Voici les indices des cases des deux plateaux envisagés.

On y a fait figurer des indices de ligne et de colonne qui seront utilisées pour faciliter la mise au point du programme.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Les coordonnées et indices des cases du plateau   |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | |  |  | 0 | 1 | 2 | 3 | 4 | 5 | 6 |  |  |  | 0 | 1 | 2 | 3 | 4 | 5 | 6 | |  |  | -3 | -2 | -1 | 0 | 1 | 2 | 3 |  |  |  | -3 | -2 | -1 | 0 | 1 | 2 | 3 | | 0 | -3 | 0 | 1 | 2 | 3 | 4 | 5 | 6 |  | 0 | -3 | 0 | 1 | 2 | 3 | 4 | 5 | 6 | | 1 | -2 | 7 | 8 | 9 | 10 | 11 | 12 | 13 |  | 1 | -2 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | | 2 | -1 | 14 | 15 | 16 | 17 | 18 | 19 | 20 |  | 2 | -1 | 14 | 15 | 16 | 17 | 18 | 19 | 20 | | 3 | 0 | 21 | 22 | 23 | 24 | 25 | 26 | 27 |  | 3 | 0 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | | 4 | 1 | 28 | 29 | 30 | 31 | 32 | 33 | 34 |  | 4 | 1 | 28 | 29 | 30 | 31 | 32 | 33 | 34 | | 5 | 2 | 35 | 36 | 37 | 38 | 39 | 40 | 41 |  | 5 | 2 | 35 | 36 | 37 | 38 | 39 | 40 | 41 | | 6 | 3 | 42 | 43 | 44 | 45 | 46 | 47 | 48 |  | 6 | 3 | 42 | 43 | 44 | 45 | 46 | 47 | 48 | |

# Initialisation

Une fois la description textuelle du plateau effectuée, sa largeur est déterminée, sa suite de cases aussi.

Le programme détermine les symétries du plateau. Il prépare ainsi les calcules à effectuer pour dresser la liste des situations obtenues par symétrie de l’une d’elles pour déterminer si elle a déjà été rencontrée.

Le programme détermine aussi la liste des mouvements à envisager. Un mouvement « envisageable » est une suite ordonnée de 3 cases du plateau contiguës, alignées. Leur sens de parcourt est important. Il est représenté par le triplet des indices des cases. Par exemple le triplet (2,3,4) sera utilisé pour effectuer un mouvement à partir d’une situation où les cases 2 et 3 sont occupées par une pierre, et la case 4 est vide. Le triplet (4,3,2) sera utilisé pour effectuer un mouvement à partir d’une situation où les cases 4 et 3 sont occupées par une pierre, et la case 2 est vide.

Ces données sont rapides à calculer, elles sont recalculées à chaque reprise.

Les ensembles EI et EG sont eux aussi très rapides à calculer, et recalculés à chaque reprise.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Les situations initiales, une seule pierre retirée, avec mise en évidence des symétries.   |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | | Situation 1 | | | | | | |  | Situation 2 | | | | | | |  | Situation 3 | | | | | | |  | Situation 4 | | | | | | | |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | | Situation 5 | | | | | | |  | Situation 6 | | | | | | |  | Situation 7 | | | | | | |  | Situation 8 | | | | | | | |

On peut remarquer que les situations 2 et 8 autorisent chacune un seul mouvement, aux symétries près pour la situation 8, et que ce mouvement aboutit à la même situation. Donc s2 et s8 développent dès le premier mouvement exactement les mêmes situations.

Les situations gagnantes sont les duales de ces situations initiales où chaque pierre est remplacée par une case vide, et la case vide remplacée par une pierre. Sur le plateau Français il y en a donc aussi 8.

# Volume

Calcul grossier du nombre de situations différentes :

Sur un plateau de N cases, chaque case peut contenir ou non une pierre. Le nombre théorique de situations différentes est donc

Pour 33 cases, cela représente environ 9 milliards de cas.

Pour 37 cases, cela représente environ 140 milliards de cas.

On peut réduire ces cas d’un facteur 8 en considérant les symétries possibles.

Le nombre de cas dans les premières étapes est lui aussi réduit. Mais on peut penser que plus de mouvements sont effectués, et plus de situations sont possibles.

# Les symétries

Le plateau est composé de cases données par leur indice dans un rectangle englobant dont on connait la largeur L et la hauteur H. Pour un indice de case i donné, on lui calcule des coordonnées cartésiennes x=i%L, y = I/L ;

Les coordonnées de ces cases sont situées dans deux intervalles identifiés : [xMin, xMax], [yMin, yMax].

Ici, on est assuré que xMin = yMin = 0, xMax = L-1, yMax = H-1.

Une « symétrie » est une rotation ou une symétrie axiale qui conserve globalement toutes les cases du plateau. On envisage les 3 rotations de 90°, 180°, 270°, dont le centre reste à déterminer, et les symétries axiales selon des axes parallèles aux axes du repère ou à ses deux diagonales principales.

L’objectif est de dresser pour chaque symétrie trouvée un tableau T de taille L\*H, tel que T[i] fournit l’indice de la case image de la case d’indice i par la symétrie.

Ainsi, lorsqu’on voudra les symétries d’une situation donnée, situation elle-même inscrite dans le rectangle englobant, et fournissant la liste de ses cases occupées, il sera aisé d’exploiter ces tableaux.

Pour que le plateau soit conservé, il est nécessaire, mais pas suffisant, que son rectangle englobant le soit. Cela est le cas pour la rotation 180° et les deux symétries principales. Cela n’est le cas pour les autres rotations et symétries que si le rectangle est un carré.

Le centre du rectangle englobant joue un rôle essentiel dans la recherche de symétries. Mais dans le cas général, ses coordonnées peuvent ne pas être entières. Soient

ses coordonnées. Si le rectangle est un carré , les coordonnées de son centre sont soit toutes deux entières, soit toutes deux demi-entières. Leur somme ou leur différence est donc entière.

Si

Rotation 90° autour du centre du carré englobant

Rotation 180° autour du centre du rectangle englobant

Rotation 270° autour du centre du carré englobant

Symétrie du rectangle selon un axe vertical

Symétrie du rectangle selon un axe horizontal

Symétrie du carré selon une parallèle à la 1ère diagonale

Symétrie du carré selon une parallèle à la 2ième diagonale

# Résolution

## Fonctionnel

Choix d’une [fiche de jeu](#_Fiche_de_jeu) « chemin/nom.xml » qui décrit le plateau initial, et fournit des valeurs aux paramètres ND et NF.

Nous stockons les données dans un répertoire « chemin/nom »

Dans ce répertoire,

* un [fichier pilote](#_Fichier_Pilote) nom.xml, initialement égal à la fiche de jeu fournie. Et qui contiendra les éventuelles solutions trouvées et les informations de reprise en cas de suspension de la recherche.
* deux fichiers binaires ED.dat et EF.dat calculés lors de l’initialisation, et qui ne sont plus modifiés par la suite.

L’outil permet de choisir la fiche de jeu originelle « chemin/nom.xml » contenant une description du plateau et des valeurs pour les deux paramètres ND et NF

Si le répertoire n’existe pas, il le crée et l’initialise

S’il existe mais que la situation initiale du fichier diffère de celle sauvée dans le fichier de pilotage, ou l’un des paramètres ND ou NF, il le signale et refuse de continuer.

Si la situation décrite dans le fichier fourni n’a aucune case inoccupée, alors l’outil recherche une solution pour chaque situation issue de ce plateau complet, duquel il retire une pierre (aux symétries près). Sinon il recherche une solution pour la situation fournie uniquement (EI contient alors cette unique situation).

La valeur de reprise/@idx est initialisée à 0, et mise à jour avec l’indice de la situation en cours de test, et avec laquelle la recherche repart. Cet indice est l’index dans le fichier ED.dat de la situation à prendre en compte.

Si le plateau initial est complet et qu’il a NC cases (et donc NC pierres), alors ED.dat contient les situations accessibles après ND mouvements appliqués à des situations initiales ayant NP = NC – 1 pierres : le nombre de pierres dans les situations de ED.dat est NP – ND .

Si le plateau initial est incomplet, qu’il a NC cases et NP pierres ( NP < NC ), alors ED.dat contient les situations accessibles après ND mouvements appliqués à cette situation initiale ayant NP pierres : le nombre de pierres dans les situations de ED.dat est NP – ND .

Le fichier EF.dat contient les situations obtenues après NF mouvements inverses sur les situations à 1 pierre, ces situations ont donc NF + 1 pierres.

Ces deux fichiers contiennent des tableaux d’octets qui donnent les indices des pierres des situations calculées initialement.

Le programme sélectionne successivement chaque description de situation de départ stockée dans ED.dat (en sautant, lors d’une reprise, celles déjà testées) pour tenter de lui trouver une situation finale de EF.dat (à une symétrie près).

Supposons que pour la situation SD en cours d’évaluation, le programme trouve une situation SF’ équivalente à une situation SF de EF.dat. La liste des mouvements pour passer de SD à SF’ est mvtDF.

Le programme recherche alors les situations de EI qui permettent d’atteindre SD (à une symétrie près). Il opère par recherche inverse depuis SD pour trouver la ou les situations SI’ équivalentes à une situation SI de EI qui n’est pas encore résolue. La liste de mouvements pour passer de SI’ à SD est mvtID. Elle est l’inverse de celle ayant permis de passer par recherche inverse de SD à SI’.

Remarque : si SD existe, c’est qu’il existe une SI1 à partir de laquelle on peut trouver ce SD. Le fait est qu’il existe peut-être d’autres situations initiales SI’ (à une symétrie près) permettant de trouver SD. SI1 sera bien sûr trouvé, mais peut-être en existe-t-il d’autres dans EI (toujours à une symétrie près).

C’est alors SI’ qu’on enregistre comme situation initiale gagnante, alors qu’elle n’est éventuellement qu’une symétrie d’une SI initialement établie. Mais c’est SI de EI qu’on marque comme résolue.

Donc lors d’une reprise, lorsqu’oninitialise les données, on veut rapidement savoir quelles situations de EI sont déjà résolues. On les compare à la liste des solutions précédemment établies. Il faut les comparer à une symétrie près.

Ce n’est pas tout à fait terminé, il faut encore établir par recherche directe une liste de mouvements mvtFG qui permette, à partir de SF’, de trouver une situation gagnante. La situation gagnante n’est d’ailleurs pas nécessairement strictement égale à l’une de celles établies lors de l’initialisation. Elle est équivalente à une symétrie près à l’une d’elles. La solution enregistre la concaténation des suites de mouvements établies mvtID puis mvtDF puis mvtFG .

Une difficulté peut apparaitre :

on peut déjà avoir établi des stocks de situations entre le niveau initial et le niveau de départ. Et aussi entre le niveau final et le niveau gagnant. Lors du démarrage de la recherche, et même si le stock a été utilisé pour initialiser ED.dat et EF.dat, ce stock est remis à vide et les niveaux cités restent vides puisque la recherche se fait entre ceux de départ et ceux finaux.

Sauf si on trouve une première solution. Donc lors d’une découverte d’une seconde solution, si on n’y prend garde, on peut être influencé par les situations testées pour retrouver mvtID de la solution précédente. Trouver une telle situation déjà rencontrée n’assure pas qu’elle mène à une SI. Il faudrait effacer les situations issues précédemment d’une telle recherche.

Une autre difficulté peut apparaitre :

On risque de passer son temps à chercher des mvtID et mvtFG qui ne font pas découvrir de nouvelles SI résolues.

Piste : conserver non pas uniquement les situations accessibles de ED.dat, mais aussi pour chacune la liste des situations de EI qui y mènent, éventuellement à une symétrie près. Et tant qu’à faire, pourquoi pas la liste des mouvements qui y mènent.

Plutôt que constituer un fichier ED.dat, on aurait plus intérêt à constituer une base de données des situations depuis EI jusqu’à ED, avec liens avant et arrière.

Situation

Long Id

Descendants

Long IdPere

Long IdFils

1-n

Ascendants

Long IdFils

Long IdPere

1-n

Les Id seraient les positions des situations dans des fichiers

## Optimisations

Pour éviter de systématiquement allouer et désallouer (ou tout au moins demander au Garbage Collector de désallouer) des quantités d’objets, on alloue des objets tampon qu’on utilise pour les calculs.

Un premier objet, le Plateau, contient

* un bool[] de la taille du rectangle englobant permettant de savoir si une case du rectangle est ou non une case du plateau.
* la liste byte[] des indices des cases du plateau, de taille NP du nombre de cases du plateau.

Il contient aussi des informations géométriques

* Taille du rectangle englobant
* Informations sur les symétries du tableau
  + Le plateau Français offre 8 symétries
* Informations sur tous les mouvements potentiels (à condition que les pierres soient disposées selon les règles) : triplets d’indices (Pierre déplacée, Pierre enjambée, Case vide de réception)
  + Pour le plateau français, il y a 92 mouvements potentiels

Pour une situation donnée, décrite par la liste byte[] des indices des pierres de la situation, de taille P donnée entre 1 et NP – 1 , on va devoir

* Déduire chaque situation obtenue par application, quand il est possible, d’un des mouvements potentiels. Ces nouvelles situations ont P-1 pierres.
* Déduire de chaque nouvelle situation les images par les symétries
* Contrôler si ces symétries ont déjà été rencontrées ou non.

Donc

* On enregistre dans les fichiers ED.dat et EF.dat uniquement les suites d’indices (au format byte) des pierres de la situation
* On stocke dans des Hashset des objets Situation qui se résument à ces octets
* On manipule une situation dans un objet SituationEtude alloué initialement.

Situation et SituationEtude exposent tous deux des méthodes GetHashCode() et Equals(), qui donnent le même hashcode et rendent vrai quand ils sont comparés l’un à l’autre et contiennent les mêmes pierres, même si les descriptions de la liste de ces pierres diffèrent. Ainsi, une nouvelle situation établie dans SituationEtude (et ses symétries) peut être testée dans le Hashset des objets Situation déjà rencontrés, mais si la situation est nouvelle, alors l’objet de type SituationEtude produit un nouvel objet Situation qui sera ajouté à ce Hashset.

Pour pouvoir réaliser ceci en conformité avec les caractéristiques du C# et de sa collection Hashset, on a fait dériver Situation et SituationEtude d’une même classe abstraite SituationBase. Et les Hashset contiennent des objets (dérivés de) SituationBase.

L’objet de type SituationEtude contient deux tableaux, chacun de la taille du rectangle englobant :

* bool[] Pierres
* bool[] ImagePierres

Le premier, Pierres, traduit la liste des pierres de la situation qu’on souhaite étudier.

On le manipule pour effectuer un mouvement, et comme source pour calculer ses images par symétrie.

On restaure les pierres en effectuant le mouvement inverse une fois que le mouvement en cours a été étudié.

Le second, ImagePierres, est l’image du premier par l’une des symétries du plateau appliquée au second. C’est celui-ci qui est utilisé dans GetHashCode() et Equals().

Pour ajouter au stock une nouvelle situation, on pourrait aussi bien utiliser Pierres que la dernière ImagePierres calculée. On a choisi Pierres.

Le plateau est constitué de cases contenues dans un rectangle englobant. Ce tableau englobant est tel que les coordonnées minimales des cases de ce plateau sont 0, et les coordonnées maximales sont égales à la dimension – 1 du rectangle. Autrement dit, le rectangle englobant est ajusté aux cases du plateau. Une fois établies les dimensions (Largeur, Hauteur), on associe à chaque case du rectangle (et donc du plateau) un indice (de type byte) de 0 à Largeur\*Hauteur – 1. Le plateau est alors la suite (ordonnée) des indices de ses cases.

Chaque symétrie est décrite par un tableau byte[] d’indirections. Ce tableau a la taille du rectangle englobant mais seules les cases correspondant à une case du plateau sont renseignées, avec l’indice de la case du plateau obtenue par symétrie.

La situation étudiée est décrite par un tableau bool[] de la taille du rectangle englobant.

Soit le plateau décrit par byte[] Plateau, la symétrie décrite par byte[] Symetrie, une situation décrite par son tableau bool[] Pierres. Pour chaque indice idx tel que Pierres[idx] soit vrai, idx représente une case du plateau. Alors Symetrie[idx] est l’indice de la case du plateau obtenue par la symétrie correspondante. Il nous reste à renseigner ImagePierres [Symetrie[idx]] = true pour constituer l’image de la situation décrite par bool[] Pierres.

En fait, comme on effectue une recherche récursive en profondeur, on gère un tableau de SituationEtude, chaque élément étant dédié à l’étude des situations suivant leur nombre de pierres.

## Statistiques

Pour un plateau français, des paramètres ND=9, NF=12

* La génération des ED prend 20 secondes
* Il y a 1 251 993 situations différentes (aux symétries près),
  + qui occupent 33 803 811 octets sur le disque.
* La génération des EF prend 750 secondes, 12,5 minutes
* Il y a 32 876 213 situations différentes (aux symétries près),
  + qui occupent 427 390 769 octets sur le disque.

Lors de la reprise, le programme met 45 secondes à recharger les 32 876 213 situations finales en mémoire.

# Mode détail

Le mode précédent permet de découvrir de nombreuses situations SD de ED.dat qui aboutissent à une situation finale SF de EF.dat. La liste des mouvements est indiquée dans cette opération.

Il reste à résoudre les deux extrémités :

* Trouver les situations initiales SI qui permettent de rejoindre SD, et une liste de mouvements associée.
* Retrouver une liste de mouvements qui permette de résoudre SF

Donc quand un couple (SD,SF) est trouvé, on souhaite l’enregistrer avec sa liste de mouvements, et pouvoir le reprendre par la suite pour le compléter.