Reecht Florian

3A IA2R

Documentation projet Sudoku

Certaines informations pourront être retrouvées dans les commentaires du code.

L'objectif de ce projet était d'établir un algorithme de résolution de sudoku.

1.Présentation des classes :

Pour réaliser ce projet Sudoku, j'ai programmé deux classes :

La classe Grille qui a comme attribut un tableau de cases (la seconde classe que j'ai programmée) Une taille ainsi qu'un entier représentant l'intervalle dans lequel les valeurs des cases se situent. Et la classe Case qui correspond à une case de la grille.

Une case à tout d'abord un attribut permettant de stocker sa valeur. Ainsi qu'un attribut grille permettant de stocker la grille à laquelle appartient la case.

Chaque case dispose de deux attributs permettant de retrouver la colonne et la ligne de la case dans la grille et pas conséquent sa position précise.

Chaque case dispose également d'une liste de type ArrayList d'entier qui correspond à l'ensemble des valeurs que la case pourra prendre sans rendre le sudoku incorrect.

Chaque case est aussi associée à un ordre qui permet de déterminer l'ordre de traitement des cases dans l'algorithme de résolution.

Pour construire le sudoku, le constructeur utilisé prend en paramètre un double tableau d'entiers.

Dans ce double tableau, il faut différencier deux types de valeurs, les 0 et les autres nombres.

Si la valeur d'indice[i][j] est égale à 0 alors la case doit pouvoir être modifiée par l'utilisateur et traitée par l'algorithme et son ordre est déterminé par le compteur ordre initialisé à 1 et incrémenté à chaque fois que la valeur de la case[i][j] est égale à 0.

Si la valeur de la case d'indice[i][j] n'est pas égale à 0, la case est une case de base et ne doit pas pouvoir être modifiée par l'utilisateur et ne doit pas être traitée par l'algorithme, sont attribut ordre prendra donc la valeur -1.

2.Présentation des Méthodes utilisées :

Dans la classe Grille on utilise tout d'abord trois méthodes returnLigne returnColonne et returnCarre

Qui prennent une case en paramètre et qui retournent l'ensemble des valeurs présentes sur la ligne,

La colonne et le carré de cette case.

Cette classe dispose également de trois méthodes valeurPossibleLigne, valeurPossibleColonne et valeurPossibleCarré qui prennent un entier ainsi que les coordonnées d'une classe en paramètre.

Ces méthodes retournent true sur la valeur peut être appliqué à la case sans rendre la ligne, la colonne ou le carré de cette case incorrect.

La dernière méthode utilisée est une méthode qui retourne le nombre de cases dont l'ordre est différent de -1 qui correspond au nombres de cases qui doivent être traitées par l'algorithme.

3.Algorithme de résolution

L'algorithme de résolution fonctionne de manière assez simple.On initialise un compteur CurrentOrdre et tant que ce compteur est inferieur au nombre de cases à traiter,on continue de traiter les cases.

Pour chaque case on récupère la liste des valeurs possibles (valeurs qui peuvent être ajoutées sans rendre la ligne, la colonne et le carré de cette case incorrecte)

On regarde la première valeur de cette case et si elle convient on l'ajoute dans le sudoku et on passe à la case d'ordre suivant.

Si cette valeur ne convient pas, on supprime cette valeur de la liste des valeurs possible et on passe à la suivante.

S'il n'y a plus de valeurs possibles dans la liste, on réinitialise cette liste et on retraite la valeur d'ordre précédent ainsi que la valeur de la case

L'algorithme se termine lorsque toutes les cases ont été traitées

4 : Partie jeu

La partie jeu est gérée par la classe jeu. La classe jeu contient des grilles prédéfinies sous la forme de double tableau d'entiers.

Le joueur peut entrer son nom et choisir une difficulté de sudoku.

Lorsque le joueur a choisi sa difficulté, une grille est choisie aléatoirement parmi les grilles de cette difficulté. Le joueur peut alors modifier les valeurs de la grille en entrant la ligne, la colonne et la valeur qu'il veut affecter dans la case. Le joueur peut également réinitialiser la grille à tout moment et aussi quitter le jeu. Si le joueur quitte le jeu la solution est affichée ainsi que le nombre de coups réalisé par l'algorithme de résolution présenté précédemment.

5 : Tests

Pour tester mon algorithme j'ai utilisé 10 grilles différentes pour chacune de ses grilles je regarde le temps de résolution ainsi que le nombre de coups effectués par l'algorithme (les tests se trouvent dans le fichier main)

6 : Pistes d'amélioration

L'algorithme présenté n'est pas optimal il est possible de l'optimiser en établissant un algorithme qui détermine l'ordre des cases en fonction des valeurs possibles. Prioriser les cases qui ont un minimum de valeurs possibles permettrai de limiter les appels et donc de raccourcir le temps d'exécution.

Concernant la partie jeu, il pourrait être intéressant d'établir un algorithme qui génère un sudoku de façon semi aléatoire.