Résolveur de Sudoku

Introduction

Le Sudoku est un jeu de réflexion connu qui repose sur la logique et la déduction. Ce projet a pour objectif de créer un programme capable de résoudre des grilles de Sudoku en appliquant des règles de déduction logiques. En partant d'une grille incomplète, l'idée est d'utiliser des règles qui permettent de réduire progressivement les possibilités pour chaque cellule, en s'appuyant sur l'interdépendance des valeurs dans les lignes, colonnes et blocs 3x3. Le résolveur de Sudoku développé ici s'appuie sur trois règles principales : Naked Singles, Hidden Singles, et Naked Pairs, qui sont appliquées jusqu'à ce que la grille soit entièrement résolue ou que les déductions arrivent à leur limite. En cas de nécessité, l'utilisateur peut intervenir pour compléter la solution.

Conception et Stratégies de Résolution

Le cœur de la solution repose sur une approche de déduction logique à travers l'implémentation de plusieurs règles, chacune visant à simplifier la grille de Sudoku étape par étape. Pour structurer cette approche, j'ai choisi de diviser le processus en différentes "règles de déduction" qui s'appliquent successivement sur la grille. Chaque règle est conçue pour détecter des patterns dans la grille et réduire les possibilités dans les cellules sans jamais recourir à des essais et erreurs.

Les règles de déduction choisies

- **Règle 1 : *Naked Singles***

  Cette règle se concentre sur les cellules qui n'ont qu'un seul candidat possible, et donc le nombre doit être placé dans la cellule.

- **Règle 2 : *Hidden Singles***

  Contrairement à la règle précédente, celle-ci trouve des cellules dans lesquelles un chiffre est le seul candidat possible dans une ligne, une colonne ou un bloc 3x3, même si plusieurs candidats sont présents pour cette cellule.

- **Règle 3 : *Naked Pairs***

Cette règle identifie des paires de candidats dans deux cellules d'une même ligne, colonne ou bloc. Si ces paires de candidats sont les seules options possibles pour ces cellules, elles peuvent être éliminées comme options dans les autres cellules de la même unité possibles pour ces cellules, elles peuvent être éliminées comme options dans les autres cellules de la même unité.

Choix de Conception

- **Singleton pour la grille** :

La classe `SudokuGrid` utilise le pattern Singleton pour s'assurer qu'une seule instance de la grille existe à tout moment. Cela permet de gérer de manière centralisée toutes les opérations effectuées sur la grille, garantissant la cohérence des données tout au long du processus de résolution.

- **Héritage pour les règles de déduction** :

La classe abstraite `DeductionRule` sert de base pour chaque règle de déduction spécifique (DR1, DR2, DR3). Chaque règle implémente la méthode `apply()`, qui modifie la grille en fonction de la règle qu'elle applique. Ce système modulaire permet d'ajouter facilement de nouvelles règles de déduction si nécessaire.

- **Gestion des candidats** :

La classe `CandidateManager` se charge de la mise à jour des candidats pour chaque cellule. À chaque application de règle, cette classe ajuste les listes de candidats disponibles en fonction des nouvelles déductions.

- **Validation de la grille** :

La classe `Validator` est responsable de la vérification de la validité de la grille à chaque étape, en s'assurant que les lignes, les colonnes et les blocs respectent les règles du Sudoku.

Méthodologie de Résolution

Le processus commence par une mise à jour initiale des candidats pour chaque cellule. Ensuite, les règles de déduction sont appliquées itérativement. Si une règle modifie une cellule, la grille est mise à jour et les candidats sont recalculés. Ce processus est répété jusqu'à ce que la grille soit résolue ou qu'il ne soit plus possible de faire des déductions logiques. En cas de stagnation, l'utilisateur est invité à saisir une valeur manuellement.

Problèmes rencontrés

1. Limitation des règles de déduction pour des grilles complexes

Un autre défi rencontré a été la gestion des grilles de Sudoku plus complexes, où les règles de déduction ne suffisent pas à résoudre la grille. Certaines grilles nécessitent des combinaisons de plusieurs règles pour progresser, et dans certains cas, les règles de déduction ne peuvent pas suffire à remplir toutes les cellules.

Solution apportée :

Afin de contourner cette limitation, j'ai ajouté une phase dans le processus où l'utilisateur est invité à remplir manuellement les cellules lorsque plus aucune déduction ne peut être appliquée. Cela permet au programme de continuer à résoudre la grille même lorsque les règles logiques ne suffisent pas. Bien que cela ne soit pas optimal, cela garantit que la grille est toujours résolue, même si la solution ne peut être entièrement automatique.

2. Gestion des entrées utilisateurs incorrects

Lors de l'entrée de la grille initiale par l'utilisateur, des erreurs peuvent survenir, comme la saisie de valeurs invalides (par exemple, des chiffres en dehors de l'intervalle [1-9]) ou la création de grilles invalides. Cela aurait pu empêcher le programme de fonctionner correctement, car il n'était pas prévu de gérer ces cas.

Solution apportée :

J'ai mis en place des vérifications pour valider les entrées de l'utilisateur avant de les traiter. Cela inclut des contrôles pour s'assurer que chaque cellule contient un chiffre valide et que la grille initiale respecte les règles de base du Sudoku (pas de doublons dans les lignes, les colonnes et les blocs).

3. Interaction utilisateur et gestion des saisies

Le processus d'interaction avec l'utilisateur, en particulier lorsqu'une saisie manuelle est nécessaire, devait être intuitif et éviter toute confusion. Une gestion efficace des saisies était essentielle pour ne pas perturber l'utilisateur et garantir une bonne expérience, notamment lorsqu'il fallait lui demander de remplir des cellules à certains moments.

Solution apportée :

J'ai conçu un système simple pour demander à l'utilisateur de remplir des cellules lorsque le programme ne peut plus appliquer de règles. L'invite est claire, et l'utilisateur est informé des erreurs possibles, comme l'introduction de valeurs incorrectes, avec un message d'erreur précis pour chaque cas.

Conclusion

Ce projet a permis de créer un résolveur de Sudoku robuste basé sur des règles de déduction logique. Grâce à une approche modulaire et évolutive, le programme peut résoudre une large variété de grilles de Sudoku. Toutefois, certaines grilles complexes nécessitent l'intervention de l'utilisateur, ce qui limite l'efficacité totale du programme. Des améliorations futures pourraient inclure l'intégration d'un algorithme de backtracking pour gérer ces cas plus complexes ou optimiser davantage la gestion des candidats. Néanmoins, ce système offre une solution élégante et efficace pour résoudre des Sudoku tout en respectant les règles de déduction logique.