Problem set 3: SUDOKU TESTING

Jonah Verly

Plan de présentation

- **Introduction**: Explication de ce en quoi consiste le problem set.
- **Code**: Explication du fonctionnement du code et des librairies utilisées.
- **Testing**: Explication de la stratégie de testing.
- **Conclusion**: Problèmes rencontrés et leur résolution.

Introduction

En quoi consiste le problem set?

Introduction

Créer un solveur de sudoku.

Le tester avec une stratégie de test.

Code

Créer un solveur de sudoku

- Pour créer ce solveur, il faut déjà que les paramètres en entrée respecte les préconditions. ➔ check_sudoku
- Solveur: programmation défensive, brute force avec backtracking. Duplicata de la grille afin de pouvoir être réutilisée pour les tests.

Test

Blackbox testing

- Exécution du solveur sur un sudoku dont on connait la valeur de retour.
- Comparaison avec ce qui est attendu.
- On ignore le fonctionnement du code ➔ quelqu'un d'autre teste.

Test

Random testing

- Utilisation de random: fonctionne avec le Twister Mersenne (algorithme)
- Fonction génératrice de grilles.
- Stocker les erreurs pour avoir le taux de grilles non résolubles.

Test

Fuzzer

- Utilisation de random pour changer la position des nombres, un nombre aléatoire de fois.
- La « seed » correspond aux grilles utilisées pour le blackbox testing.
- Copie des grilles en profondeur pour en garder la trace.

Conclusion

Conclusion

Erreurs rencontrées

- Python2 ➔ Python3
- Affichage des grilles non conformes.
- Dupliquer une grille pour pouvoir la retester par la suite.
- Stocker les opérations dans un fichier de logging.

Merci