Projet: Solitaire

November 1, 2021

1 Consignes (important)

Les projets seront faits en trinôme. Vous rendrez votre projet sur l'Université Virtuelle, dans l'espace prévu à cet effet, sous la forme d'un fichier zip qui sera intitulé:

NOM1_prénom1_NOM2_prénom2_NOM3_prénom3.zip.

Ce fichier zip qui contiendra à la fois le code et un rapport détaillé. Le code devra être fait en langage python version 3, et devra être commenté de telle manière à ce qu'il soit compréhensible facilement. Le rapport sera noté sur 10 et le code sur 10. La date limite du rendu final du projet est fixée au 10/12/2021. Chaque jour de retard entrainera une diminution de la note du projet de 2 points. Pour toute question à propos du projet, vous pourrez envoyer un mail à l'adresse edwin.hamel.de.le.court@ulb.be.

2 Le Solitaire

2.1 Une configuration du Solitaire

Le but de ce projet est d'utiliser un SAT-solveur pour résoudre le Solitaire. Le Solitaire est un jeu de plateau classique à un seul joueur. Le plateau de jeu comporte des trous dans lesquels on peut mettre des billes. Il prend traditionnellement différentes formes; certaines sont données dans la Figure 1.

Ainsi, pour pouvoir trouver des résultats sur toutes ces formes, nous nous intéresserons d'abord au cas général où le plateau est un carré de taille quelconque, et où les trous sont disposés de façon arbitraire sur le plateau. Ainsi, nous pouvons représenter une configuration du jeu par un entier positif n, et une matrice de taille $n \times n$ dont les éléments appartiennent à $\{-1, 0, 1\}$. Si la valeur de la case (i, j) de la matrice est égale à -1, alors il n'y a pas de trous sur le plateau à la position (i, j); si cette valeur est égale à 0, alors il y a un trou sur le plateau à la position (i, j), mais il n'y a pas de bille posée dessus; si cette valeur est égale à 1, alors il y a un trou sur le plateau à la position (i, j)

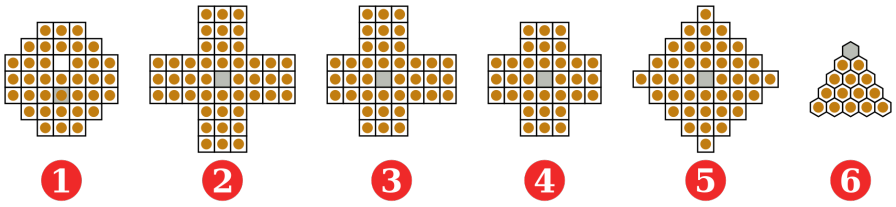

Figure 1: Différentes formes de plateau de Solitaire

avec une bille posée dessus. Par exemple, la configurations du plateau 1 de la figure 1 est représentée par la matrice suivante

$$\begin{pmatrix} -1 & -1 & 1 & 1 & 1 & -1 & -1 \\ -1 & 1 & 1 & 1 & 1 & 1 & -1 \\ 1 & 1 & 1 & 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ -1 & 1 & 1 & 1 & 1 & 1 & -1 \\ -1 & -1 & 1 & 1 & 1 & -1 & -1 \end{pmatrix}.$$

2.2 Un coup au solitaire

Pour jouer un coup au Solitaire, il faut choisir trois cases adjacentes, qui sont toutes les trois sur une même ligne ou une même colonne du plateau. De surcoît, exactement deux billes doivent être posées sur les trois cases, dont une posée sur la case du milieu. On fait "sauter" la bille qui se trouve sur une des extrémités par dessus la bille du milieu, pour la faire atterrir dans le trou qui se trouve à l'autre extrémité. On enlève alors la bille du milieu du plateau. La Figure 2 est une représentation des différents coups possible, en utilisant la correspondance entre matrice et configurations du jeu.

3 Questions

1. Soit n un entier et M et M' deux matrices de tailles $n \times n$ dont les éléments sont dans $\{-1, 0, 1\}$. Donner (dans votre rapport) une formule de logique propositionnelle en **forme normale conjonctive** qui est satisfaisable si et seulement s'il est possible de passer de la configuration du Solitaire qui correspond à M à celle qui correspond à M', en effectuant un ou plusieurs coups. La qualité des explications et le soin sont très importants: expliquer le choix de vos variables Booléennes, les différentes parties de votre formule, etc.

2. En déduire une fonction Python `solution` basé sur la librairie PySAT qui prend en entrée deux matrices (codées sous forme de listes de listes) M et

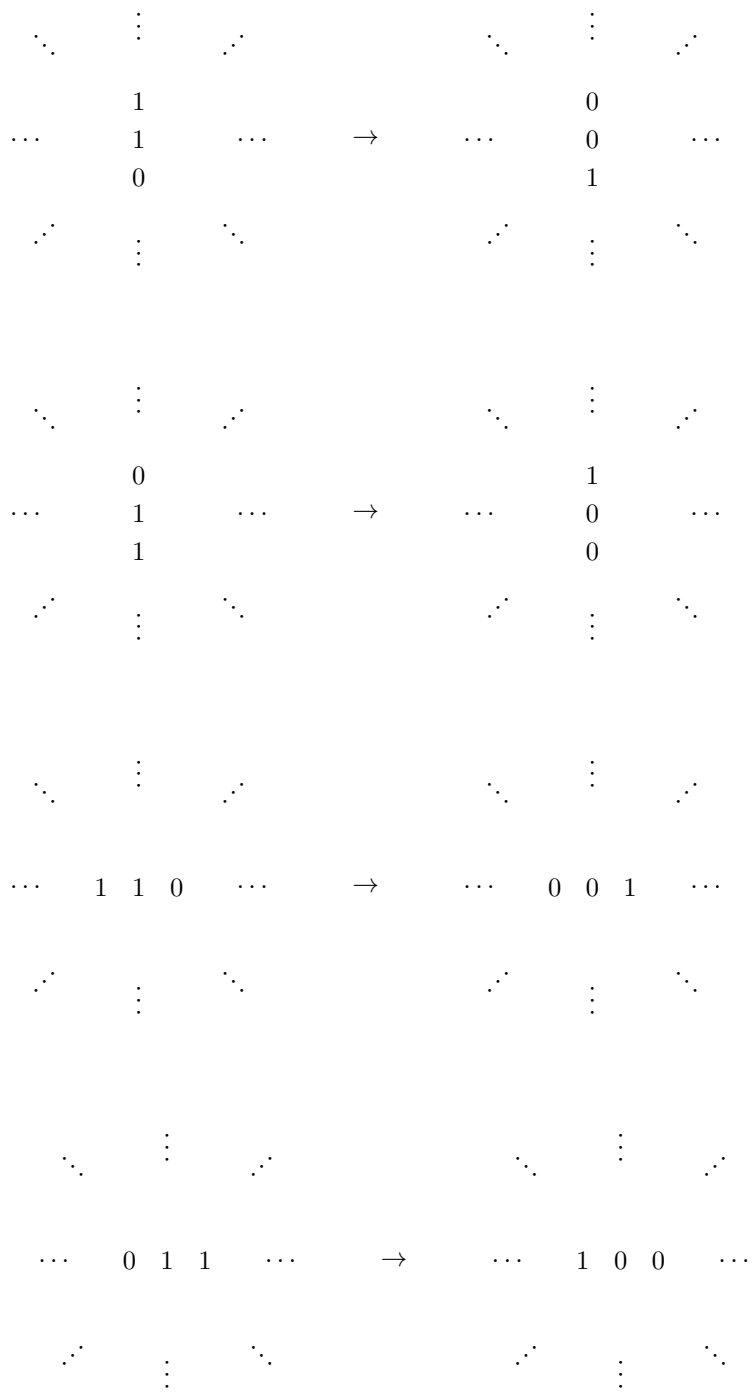

Figure 2: Les différents coups possibles au Solitaire

M' de même taille, dont les éléments sont dans $\{-1, 0, 1\}$, qui renvoie `True` s'il existe une suite de coups qui permettent de passer de la configuration du Solitaire qui correspond à M à celle qui correspond à M', et qui renvoie `False` sinon. Cette fonction `solution` devra construire une instance du problème SAT, et appeller un solveur SAT (vous pouvez par exemple utiliser MiniSAT22 ou Glucose3). Par ailleurs, cette fonction `solution` devra être écrite dans un fichier ayant pour nom `projet.py`. Pour que votre code soit considéré comme correct, le fichier `test.py` fourni avec le sujet devra écrire dans le terminal que votre programme marche sur tous les exemples testés (le fichier `test.py` devra se trouver dans le même répertoire que le fichier `projet.py`). Il est possible que cela prenne beaucoup de temps pour les deux derniers exemples (et cela sera pris en compte dans la correction). Bien sûr, le code que vous fournirez sera corrigé en détail, et passer tous les test fournis par les fichier `test.py` ne suffira pas à déterminer avec certitude la correction de votre code.

3. En vous basant sur le programme fait à la question précédente, construire un programme qui pour toute configuration de la Figure 1 (sauf la 6 et la 4), teste s'il est possible d'atteindre une configuration du même plateau mais qui contient une seule bille là où au départ il n'y avait pas de bille. Dans cette figure, les carrés gris correspondent à un trou sans bille posé dessus, les carrés blancs à une case du plateau où il n'y a pas de trous, et les carrés blancs avec un cercle à l'intérieur à des trous avec une bille posée par dessus). Donc, par exemple, il s'agira de tester s'il est possible de passer de la matrice donnée précédemment, à la matrice suivante:

$$\begin{pmatrix} -1 & -1 & 0 & 0 & 0 & -1 & -1 \\ -1 & 0 & 0 & 0 & 0 & 0 & -1 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ -1 & 0 & 0 & 0 & 0 & 0 & -1 \\ -1 & -1 & 0 & 0 & 0 & -1 & -1 \end{pmatrix}.$$

   Indiquer les résultats obtenus dans le rapport.

4. Même question que précédemment mais on ne demande pas que la dernière bille soit placée là où il y avait le trou de départ: on demande simplement qu'il y ait dans la configuration finale exactement une bille, peu importe sa position

5. Pour le premier plateau de la Figure 1, on voudrait tester pour chaque possibilité de placer la trou au début, s'il existe une stratégie pour n'avoir plus qu'une seule bille à la fin, placée sur le trou de départ. Mathématiquement,

considérons les matrices de la forme

$$\begin{pmatrix} -1 & -1 & b_1 & b_2 & b_3 & -1 & -1 \\ -1 & b_4 & b_5 & b_6 & b_7 & b_8 & -1 \\ b_9 & b_{10} & b_{11} & b_{12} & b_{13} & b_{14} & b_{15} \\ b_{16} & b_{17} & b_{18} & b_{19} & b_{20} & b_{21} & b_{22} \\ b_{23} & b_{24} & b_{25} & b_{26} & b_{27} & b_{28} & b_{29} \\ -1 & b_{30} & b_{31} & b_{32} & b_{33} & b_{34} & -1 \\ -1 & -1 & b_{35} & b_{36} & b_{37} & -1 & -1 \end{pmatrix}.$$

où chaque $b_i \in \{0, 1\}$. Pour tout $i \in \{1, \ldots, 37\}$, on pose M_i la matrice où $b_i = 0$ et $b_j = 1$, pour tout $j \in \{1, \ldots, 37\} \setminus i$. On pose également M_i' le dual de M_i: le 0 de M_i est reamplacé par 1, et tous les 1 par des 0.

On veut maintenant construire la matrice R suivante: pour tout $i \in \{1, \ldots, 37\}$, $b_i = 1$ s'il existe une stratégie pour passer de M_i à M_i', sinon $b_i = 0$. Faire un programme qui calcule et affiche la matrice R, en utilsant les questions précédentes. *Si le temps de calcul est trop long, vous pouvez considérer un plateau de dimension inférieure.*