1 Jeu

1. a) Tout noir toit blanc

Un problème mathématique : le jeu du « tout noir, tout blanc ».

Le support : une grille à m lignes et n colonnes avec des cases pouvant prendre deux couleurs (noir ou blanc). Au départ, toutes les cases sont noires. En retournant une case, le joueur en change sa couleur ainsi que celle de ses quatre voisines (nord, sud, est, ouest). Le but du jeu est de rendre la grille toute blanche.

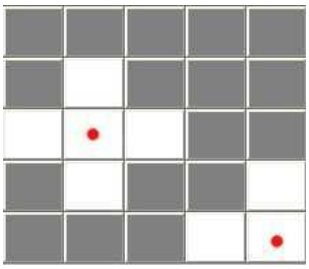
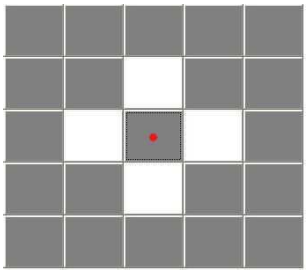


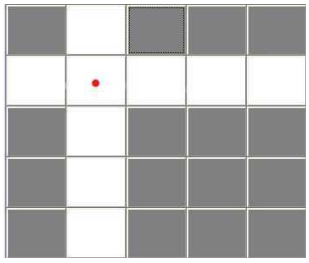
Figure 1: sur une grille 5*5

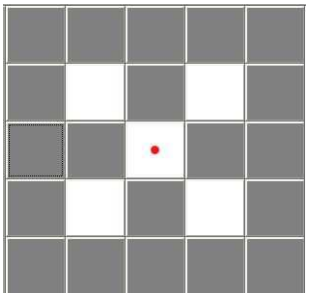
Projet : construire une telle zone de jeu, où l'utilisateur qui clique sur une case en change la couleur ainsi que celle des voisines. Proposer des variantes avec d'autres règles de jeu, ou des configurations initiales aléatoires. Programmer un algorithme de recherche des solutions de ce problème pour des tailles de grille convenables.

L'objectif est de vous laisser très libres quant à vos choix de programmation, vos créations de fonctions, et vos idées de résolution de ce problème. Ce papier a pour but de vous décrire le travail attendu, ainsi que de vous fournir un support pour guider vos premiers pas.

Extensions du jeu : On peut définir d'autres règles de retournement des cases. En voici trois autres , que vous pourrez proposer à l'utilisateur en début de programme :





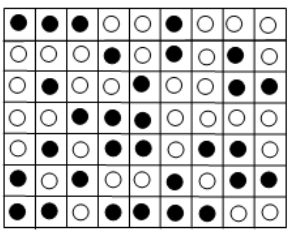


3ème partie : Recherche de solutions Au lieu du mode « jeu », votre programme pourra proposer à l'utilisateur de résoudre le problème lui-même. A vous de suggérer un algorithme le plus léger possible pour réduire le temps de calcul au maximum. L'affichage d'une solution pourra se faire comme précédemment en marquant d'un point rouge les cases à cliquer.

1. b) Clobber

Construction d'une interface de jeu « Solitaire Clobber » est un jeu qui a été étudié en 2002 par des chercheurs et qui présente des propriétés très intéressantes dans le domaine des jeux combinatoires. À l'heure actuelle, nous ne connaissons que très peu de résultats à son sujet.

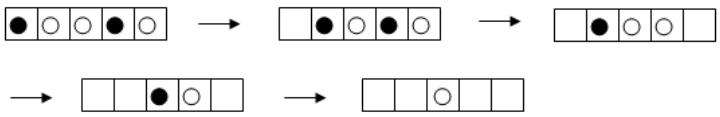
Ce jeu est défini de la façon suivante : sur une grille n*m, on dispose arbitrairement des pierres noires et blanches, comme l'illustre l'exemple ci-dessous.



Il y a un seul joueur et les règles sont les suivantes : le joueur choisir une pierre, et va capturer une pierre voisine (située horizontalement ou verticalement) de l'autre couleur. La pierre capturée est retirée du jeu, et l'autre pierre prend

sa place. Le joueur répète cette action tant que cela est possible. L'objectif est de minimiser le nombre de pierres restantes à la fin du jeu.

Exemple d'exécution sur une grille 1*5 :



Nous vous demandons dans un premier temps de créer une interface de jeu qui permette à un utilisateur de jouer.

L'utilisateur devra au préalable rentrer au clavier la taille (longueur et largeur) de la grille sur laquelle il souhaite jouer, puis votre programme génèrera une disposition aléatoire des pierres.

Ensuite, l'utilisateur pourra capturer les pierres de la façon suivante (ceci n'est qu'un exemple) :

- Cliquer sur la pierre que l'on veut choisir
- Cliquer sur la pierre que l'on veut capturer
- Si les pierres sont bien de couleurs différentes et situées sur des cases voisines, effectuer la capture. Sinon, annuler le coup et recommencer.

Lorsque l'utilisateur n'a plus la possibilité de jouer, afficher un message « fin du jeu ».

Votre programme devra ensuite afficher ensuite son score, c'est-à-dire le nombre de pierres restantes sur la grille.

Résolution du problème sur une seule ligne On considère le jeu sur une grille 1*m (cf. exemple ci-dessus). Etant donnée n'importe quelle disposition des pierres, le problème consiste à déterminer le nombre minimum de pierres que l'on peut laisser à la fin (et si possible, savoir comment jouer pour y arriver).

Améliorations du jeu initial.

- Permettre à l'utilisateur de recommencer le jeu et d'améliorer son score. La configuration initiale aléatoire doit donc être sauvegardée. Dans ce cas, afficher le meilleur score de l'utilisateur sur cette configuration.
- Changer de règles : la version originale du jeu prévoit que le joueur doit alternativement capturer une pierre blanche puis une noire. Votre programme doit maintenant incorporer cette règle. Il demandera initialement à l'utilisateur s'il souhaite jouer avec la première ou la seconde règle.

2 Image

2. a) Sténographie

Le principe La stéganographie est l'art de cacher une image (ou un texte) dans une image. Télécharger sur mon git : https://github.com/debimax/cours-debimax/tree/master/images l'images image_cachee2.png (c'est une de mes photos vacances en Bulgarie).

Effectivement j'ai caché une image dans image_cachee2.png. Vous voulez la voir ? Télécharger alors télécharger image_decodee.png et vous verrez l'image cachée décodé.

Figure 2: J'ai caché une photo dans une autre photo

Pour comprendre le principe regardez ce site

Le principe est relativement simple. Les pixels d'une image sont codés par 8 bits (un octet) de 0 à $2^8 = 256$ (soit en hexadécimal de 0 à FF.

Les 24 bits d'une couleur se décomposent en 3 fois 8 bits :

- 8 bits sont consacrés à la teinte primaire rouge ;
- 8 bits sont consacrés à la teinte primaire vert ;
- 8 bits sont consacrés à la teinte primaire bleu.

Par exemple si on a pour un pixel (rouge,vert,bleu) d'une image1 le code ***(F8,40,F1)*** et que l'on veuille « cacher » le pixel d'une image2 ***(51,A4,3B)*** on remplace les bits de poids faible de l'image1 par les bits de poids fort de l'image2.

$$(\mathbf{F}8, 40, \mathbf{F}1) + (\mathbf{5}1, \mathbf{A}4, \mathbf{3}B) \rightarrow (F5, 4A, F3)$$

En effet Il y a peu de différence entre (F8,40,F1) et (F5,44,F3).

Pour le décodage on transformera **(F5,44,F3)** en **(5F,44,3F)** (ou **(55,44,33)**) et de même il y a peu de différence entre les couleurs ce qui restitue l'image de départ cachée.

Projet Créer un programme qui cache une image dans une deuxième images avec la libraire PIL.

Extensions Crée un gui avec tkinter, ou un site internet qui fasse le codage et le décodage.

Coder du texte dans une image.

2. b) Codes barres

Principe rechercher des informations sur les codes barres EAN13 utilisés sur les produits de consommation courante.

- http://dsaurel.free.fr/Code_barre/cb.htm
- http://fr.wikipedia.org/wiki/Code-barres_EAN

Les éléments EAN se caractérisent par une succession de quatre barres (deux barres claires qui alternent avec deux barres sombres) dont la somme des largeurs vaut toujours 7 modules. Chacune de ces barres est elle-même constituée de la juxtaposition de 1 à 4 barres élémentaires de même couleur. Il y a donc au total 7 barres élémentaires dans un élément.

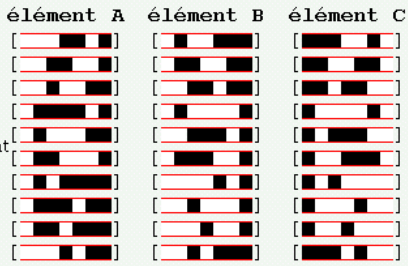
Chaque élément peut être représenté en binaire par une suite de 7 bits :

- Un X ou 1 correspondant à une barre élémentaire noire,
- Un _ ou 0 correspondant à une barre élémentaire blanche

Voici les représentations des 10 chiffres comme éléments A, B ou C :

	élément A	élément B	élément C
0	[___XX_X]	[_X__XXX]	[XXX__X_]
1	[__XX__X]	[_XX__XX]	[XX__XX_]
2	[__X__XX]	[__XX_XX]	[XX_XX__]
3	[_XXXX_X]	[_X____X]	[X____X_]
4	[_X___XX]	[__XXX_X]	[X_XXX__]
5	[_XX___X]	[_XXX__X]	[X__XXX_]
6	[_X_XXXX]	[____X_X]	[X_X____]
7	[_XXX_XX]	[__X___X]	[X___X__]
8	[_XX_XXX]	[___X__X]	[X__X___]
9	[___X_XX]	[__X_XXX]	[XXX_X__]

soit, graphiquement



Projet Construire un programme avec un codage et décodage :

- Générert une image du code barre au format PBM à partir du code numérique.
- Obtenir obtenir le code numérique à partir du code barre.

3 Audio

4 Internet