NSI - Projet 2 : automates cellulaires 1D

1 Automates cellulaires

Les automates cellulaires furent inventés à la fin des années 1940 par Stanislaw Ulam et John Von Neumann. Ils sont notamment utilisés pour modéliser les processus d'expansion des épidémies.

Un automate cellulaire peut se définir comme un ensemble de cellules pouvant prendre des états successifs en fonction des états des cellules voisines à chaque étape.

Pour créer un automate cellulaire, il faut donc deux éléments :

- une grille, ou matrice en 1, 2 ou 3 dimensions, où chaque cellule peut connaître un nombre fini d'états (au minimum 2)
- une règle qui définit l'évolution de chaque cellule lors de l'itération suivante, en fonction de l'état des cellules voisines

Les automates cellulaires peuvent être à :

- 1 dimension (cellules disposées sur une ligne, une ligne correspond à une génération)
- 2 dimensions (cellules disposées sur une grille, une grille correspond à une génération)
- 3 dimensions (cellules disposées sur une matrice cubique) - difficiles à visualiser.

En partant d'une configuration initiale de cellules (la génération 0), on applique les règles, ce qui donne lieu à une nouvelle configuration, sur laquelle les règles peuvent à nouveau s'appliquer, et ainsi de suite. En fonction des règles, certains automates peuvent donner lieu à des motifs très complexes, et d'autres s'éteindre rapidement.

Le plus connu des automates cellulaires est "le jeu de la vie" introduit par John Horton Conway dans les années 1960.

2 Automates cellulaires 1D

Les automates à 1 dimension sont décrits par Stephen Wolfram en 1982.

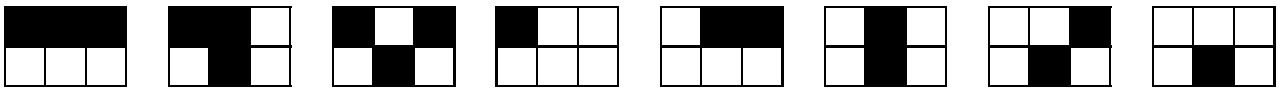
- Une cellule a deux états possibles : vivant (1), représenté par une case noire ou mort (0), représenté par une case blanche.
- Les cellules sont disposées sur une ligne, une ligne correspond à une génération.
- Les règles prennent en compte la cellule à gauche et celle à droite : v c v

Seules 256 règles sont possibles. Pour numéroter chaque règle, on utilise le codage en binaire. Les 8 configurations de voisinages possibles sont codées, puis le résultat donne le nom à la règle.

Par exemple, pour le tableau suivant, on parle de règle 103 - $(103)_{10} = (01100111)_2$:

111	110	101	100	011	010	001	000
0	1	1	0	0	1	1	1

correspondant à



Dans ce projet, nous ne regarderons pas l'évolution de la première et de la dernière cellule de chaque ligne.

Certaines règles donnent des fractales (règle 90 par exemple).

3 Ce qui est demandé

Il s'agit de compléter le code Python fourni.

1) Ecrire la fonction `afficheRegles(regles)` qui affiche le tableau à deux dimensions des règles. Par exemple, si `numeroRegle` vaut 103, `afficheRegles` doit afficher :

```
[1, 1, 1, 0]
[1, 1, 0, 1]
[1, 0, 1, 1]
[1, 0, 0, 0]
[0, 1, 1, 0]
[0, 1, 0, 1]
[0, 0, 1, 1]
[0, 0, 0, 1]
```

2) Ecrire la fonction `etatSuivant` qui retourne un booléen donnant l'état futur de la cellule.

3) Ecrire la fonction `initSimple(position)` qui retourne une liste de `NB_CELLULES` de booléens `False` sauf celui d'indice `position` qui est `True`.

   Nous appellerons `initSimple(49)`, c'est-dire que, pour démarrer, nous plaçons une cellule vivante au milieu de la première rangée.

4) Ecrire la fonction `generation(noRegle)` qui génère et dessine la colonie sur `NB_JOURS` suivant la règle `noRegle`.

   On utilisera la fonction `dessineColonie(colonie,numGen)`.

5) Pour les plus rapides, écrire la fonction `initialiseRegles(numRegle)` de manière générale.

Le code doit être clair, aéré et compréhensible, à déposer le 19/12 avant minuit dans l'application casier de l'ENT sous la forme d'un fichier nommé PR2 suivi du nom du groupe (7 lettres maximum) (par exemple PR2Fusees.py).

- Les noms des membres du groupe apparaissent en haut du fichier sous forme de commentaire.
- Les variables ont un nom significatif.
- Les fonctions sont documentées.

L'investissement dans le travail en classe peut également être évalué.

4 Compléments - Graphiques avec Tkinter

Pour afficher ceci :

Le code est :

```
import tkinter

def dessineFormes(canvas):
    canvas.create_rectangle((50,50),(150,100),fill="green")
    canvas.create_oval((50,110),(180,160),outline="red",width=3)
    canvas.create_line((10,50),(100,120),(150,200),fill="gray",width=3)
    canvas.create_text(250,75,text="Hello",font=("courier", 20,"bold"))

root=tkinter.Tk()
root.title("demo")
monCanvas=tkinter.Canvas(root,width=300,height=200,background="white")
monCanvas.pack()
dessineFormes(monCanvas)
root.mainloop()
```

Quelques explications :

- Le module `tkinter` nous permet de définir des variables `root` et `monCanvas` qui définissent une fenêtre graphique (ici de largeur 800 et de hauteur 600 pixels). On décrit ensuite tout ce que l'on veut ajouter dans la fenêtre. Et enfin la fenêtre est affichée par la commande `root.mainloop()` (tout à la fin).
- Le repère graphique de la fenêtre a son axe des ordonnées dirigé vers le bas. L'origine $(0, 0)$ est le coin en haut à gauche.
- `create_rectangle(x1,y1,x2,y2)` est une méthode permettant de tracer un rectangle : il suffit de préciser les coordonnées (x_1, y_1), (x_2, y_2) de deux sommets opposés. L'option `width` ajuste l'épaisseur du trait, `outline` définit la couleur de ce trait, `fill` définit la couleur de remplissage.
- `canvas.create_text(x,y,text="Mon texte")` est une méthode permettant d'afficher du texte en précisant les coordonnées (x, y) du point à partir duquel on souhaite afficher le texte.

5 Sources et références pour aller plus loin

http ://musiquealgorithmique.fr/automates-cellulaires/
http ://mathworld.wolfram.com/ElementaryCellularAutomaton.html
http ://exo7.emath.fr/cours/livre-python1.pdf
https ://python.developpez.com/cours/TutoSwinnen/ ?page=Chapitre8