Problème 1. Le jeu de la vie

Un automate cellulaire est constitué d'une grille régulière de *cellules* étant chacune dans un *état* (parmi un nombre fini d'états possibles), et d'une règle d'évolution *locale* et *synchrone* : à chaque étape de temps (discret), l'état de chaque cellule est mis à jour en fonction de l'état de ses voisines (et du sien). Les automates cellulaires sont étudiés à la fois comme modèle de calcul parallèle, et comme système dynamique discret.

Le mathématicien John Conway a défini en 1970 un automate cellulaire particulier qu'il a nommé « jeu de la vie ». Il s'agit d'un automate dont la grille est une grille bi-dimensionnelle infinie, et chaque cellule peut prendre deux états 0 ou 1 (qu'on appelle habituelement « état mort » et « état vivant »). Les voisines d'une cellule sont les huit cellules qui l'entourent. Et la règle d'évolution est la suivante :

— une cellule vivante au temps t meurt au temps $t+1$ si elle n'est entourée que de 0 ou 1 cellule vivante (*isolement*), ou si elle est entourée d'au moins 4 cellules vivantes (*surpopulation*) ; elle *survit* dans les autres cas ;

— une cellule morte au temps t devient vivante au temps $t+1$ si elle est entourée d'exactement 3 cellules vivantes (*naissance*) ; elle reste morte sinon.

Certaines de ces règles sont illustrées par la figure 1.

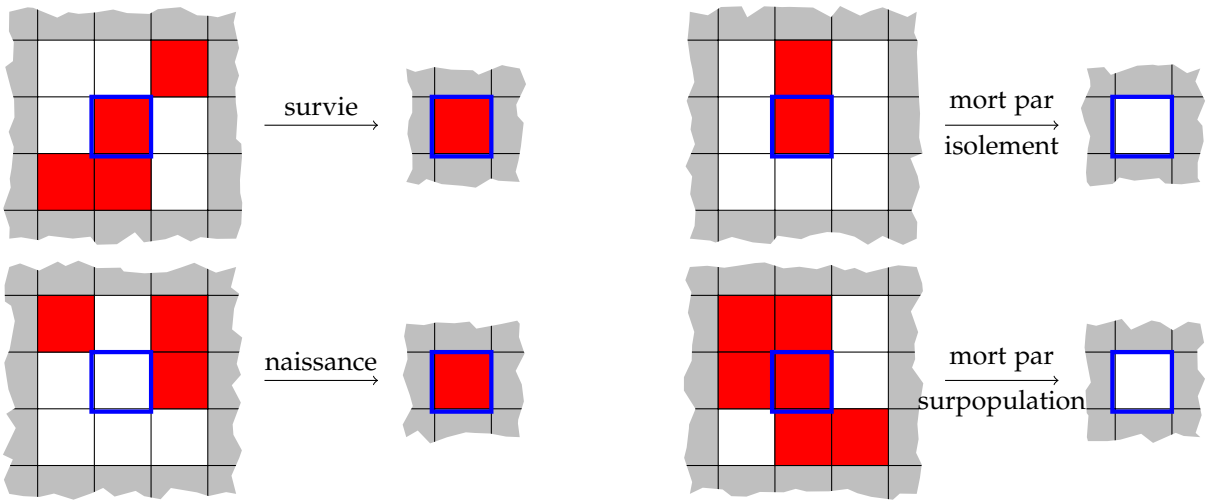


FIGURE 1 – Règles de survie, de mort et de naissance pour la cellule encadrée. Les cellules colorées sont les cellules vivantes. Le grisé qui entoure signifie qu'on ne connaît pas l'état des cellules voisines.

Le jeu de la vie a été très étudié, aussi bien par des mathématiciens et informaticiens professionnels que par des amateurs, car il présente des comportements très riches [1]

1.1 Simulation du jeu de la vie

Pour pouvoir représenter de manière finie le jeu de la vie, on modifie la grille pour en faire un *tore de dimension k* : au lieu d'être une grille infinie, c'est une grille $k \times k$ avec la colonne de gauche voisine de la colonne de droite, et la ligne du haut voisine de la ligne du bas. De cette manière, la grille n'a pas de *bord* et toute cellule possède exactement 8 voisines.

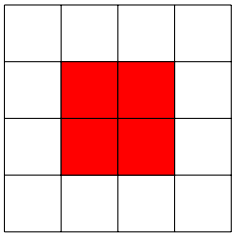
La représentation de la grille en Python sera une matrice (type `array` de `numpy`, décrit en annexe). On représentera l'état mort par un 0 et l'état vivant par un 1. La numérotation des lignes et des colonnes de la matrice ira de 0 à $(k-1)$.

1. Vous pourrez faire un tour sur votre plateforme préférée de vidéos pour admirer visuellement certains de ces comportements.

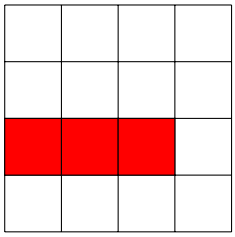
Question 1.1. Représentation de la grille

(a) On considère un tore de dimension 4. Quelle sont les coordonnées des cellules voisines de $(1,1)$? Et celles des voisines de $(0,3)$?

(b) Toujours pour le tore de dimension 4, donner la représentation matricielle des deux configurations suivantes, et donner les configurations obtenues au temps $t+1$ et $t+2$ pour chacune. Que remarque-t-on ?



Configuration A



Configuration B

(c) Écrire une fonction `grille_vide` qui prend en entrée un entier n et renvoie une grille torique de dimension n vide (toutes les cellules sont mortes).

(d) Écrire une fonction `est_vide` qui prend en entrée une grille et renvoie un booléen, qui vaut `True` si toutes les cellules de la grille sont mortes.

(e) Écrire une fonction `afficher_grille` qui prend en entrée une grille et l'affiche ligne par ligne en représentant les 0 par le caractère `-` et les 1 par le caractère `'*'`. La fonction ne renverra rien. *Par exemple, les grilles correspondant aux configurations A et B de la question (b) seront affichées sous la forme ci-dessous.*

```
- - - -     - - - -
- * * -     - - - -
- * * -     * * * -
- - - -     - - - -
```

(f) À l'aide du module `matplotlib` (voir en annexe), écrire une fonction `representation_graphique` qui prend en entrée une grille et la représente sous forme de carrés de différentes couleurs.

Question 1.2. Évolution de la grille

(a) Écrire deux fonctions `ind_suiv` et `ind_prec` qui prennent en entrée une grille et un indice i (de ligne ou de colonne) et renvoient l'indice de la ligne ou colonne suivante (resp. précédente) dans la grille. *Attention, il faut tenir compte du caractère torique de la grille !*

(b) Écrire une fonction `nombre_voisins` qui prend en entrée une grille et une cellule de la grille représentée par ses coordonnées (i,j), et renvoie le nombre de cellules vivantes parmi les voisines de (i,j), sans compter la cellule elle-même.

(c) Écrire une fonction `etat_suivant` qui prend en entrée une grille et une cellule, et renvoie l'état de la cellule (0 ou 1) au temps $t+1$.

(d) Écrire une fonction `configuration_suivante` qui prend en entrée une grille et renvoie la grille obtenue au temps $t+1$.

(e) Écrire une fonction `simulation` qui prend en entrée une grille et un entier N, et renvoie la grille obtenue après N étapes (au temps $t+N$).

(f) Modifier la fonction `simulation` pour qu'elle affiche, au cours de l'évolution, une représentation graphique de la grille à l'aide du module `matplotlib`.

Question 1.3. Sauvegarde d'une grille On peut sauvegarder une grille dans un fichier texte avec le format suivant. La première ligne contient un entier k représentant la dimension de la grille. Chacune des lignes suivantes contient deux entiers i et j séparés par un espace représentant les coordonnées d'une cellule vivante.

On donne ci-dessous des fonctions permettant de convertir une liste d'entiers en une chaîne de caractère contenant ces entiers séparés par des espaces, et inversement. Se référer à l'annexe pour les lectures et écritures dans un fichier.

```
def liste_vers_chaine(L):
    return ' '.join(str(n) for n in L)

def chaine_vers_liste(s):
    return [int(c) for c in s.split() if len(c)]
```

(a) Dessiner les configurations correspondant aux fichiers de la figure 2.

```
4
1 1
1 2
2 2
2 1
```

Automate1.txt

```
4
2 0
2 1
2 2
```

Automate2.txt

```
5
0 1
1 2
2 0
2 1
2 2
```

Automate3.txt

FIGURE 2 – Trois automates sauvegardés sous forme de fichiers texte

(b) Écrire une fonction `ouvre_grille` qui reçoit en entrée une chaîne de caractère contenant un nom de fichier, et renvoie la grille (sous forme de matrice) représentée dans le fichier.

(c) Écrire une fonction `sauve_grille` qui reçoit en entrée une grille et une chaîne de caractère contenant un nom de fichier, et sauvegarde la grille dans le fichier dans le format décrit ci-dessus. Si le fichier existe déjà, il est écrasé.

1.2 Motifs et évolution

Question 1.4. Insertion et recherche de motifs Un motif est n'importe quel rectangle de cellules, certaines vivantes et certaines mortes. L'ojectif est d'observer, pour des motifs particuliers, leur évolution au cours du temps. On appelle *cellule principale* d'un motif la cellule située en haut à gauche du motif. Des exemples de motifs sont représentés ci-dessous, avec la cellule principale encadrée..

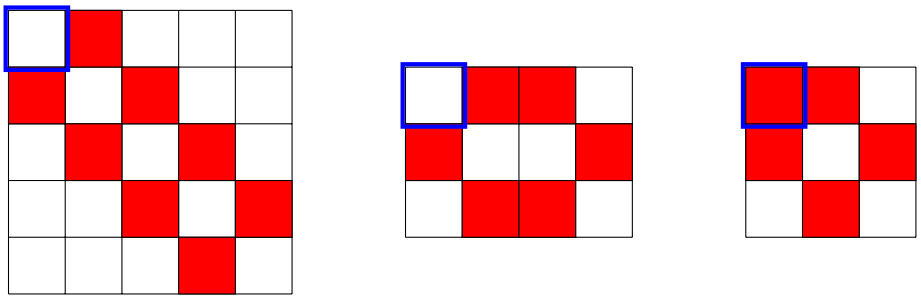


FIGURE 3 – Trois exemples de motifs

(a) Écrire une fonction `initialise` qui prend en entrée un motif et une dimension de grille m et renvoie la grille de dimension m contenant en son centre (approximatif) le motif. Si le motif est

trop grand pour rentrer dans la grille, une exception sera levée. *On détaillera le choix effectué pour placer le motif approximativement au centre de la grille.*

(b) Écrire une fonction `extraire_motif` qui prend en entrée une grille, une cellule (représentée par ses coordonnées (i, j)) et des dimensions (m, n) et renvoie le motif de dimensions $m \times n$ dont la cellule principale a pour coordonnées (i, j). *L'extraction tient naturellement compte de la nature torique de la grille.*

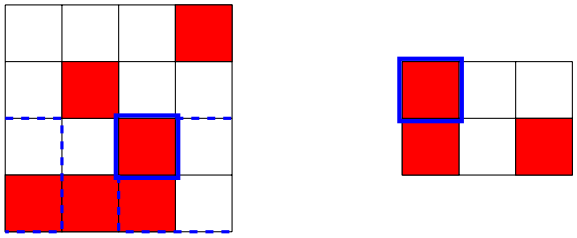


FIGURE 4 – Extraction d'un motif 2×3 depuis la cellule $(2, 1)$ d'une grille.

(c) Écrire une fonction `appartient_grille` qui prend en entrée un motif et une grille et teste si la grille contient le motif. La fonction renvoie une liste de coordonnées : pour chaque occurence du motif dans la grille, la liste contient les coordonnées de la cellule principale du motif dans la grille. Si le motif n'apparaît pas, la liste est naturellement vide. *Par exemple, avec en entrée le motif de taille* 1×1  *, la fonction renvoie les coordonnées de toutes les cellules vivantes.*

Question 1.5. Propriété des motifs et configurations Partant d'une grille G_0 au temps $t = 0$, on note G_i la grille obtenue au temps i. Une grille G_0 est un *point fixe* si $G_1 = G_0$. Une grille est dite *périodique de période* p (avec $p > 0$) si $G_p = G_0$.

(a) Vérifier que la grille du fichier `Automate1.txt` est un point fixe.

(b) Écrire une fonction `est_point_fixe` qui prend en entrée une grille et renvoie un booléen indiquant si la grille est un point fixe.

(c) Montrer que si une grille est périodique de période p, alors $G_{i+k\times p} = G_i$ pour $0 \leq i < p$ et pour tout $k \in \mathbb{N}$.

(d) Montrer que si une grille est périodique, elle admet une infinité de période.

(e) Vérifier que la grille du fichier `Automate2.txt` est périodique. Préciser sa (plus petite) période.

(f) Écrire une fonction `est_periodique` qui prend en entrée une grille et un entier N et teste si la grille est périodique de période p pour un entier $p \leq N$.

(g) On veut modifier la fonction précédente pour ne pas avoir à donner de borne supérieure N. Pour une grille de dimension n, donner une borne supérieure sur la longueur de la plus petite période de la grille, s'il en existe une. Modifier en conséquence la fonction `est_periodique` pour qu'elle ne prenne en entrée qu'une grille.

Question 1.6. Vaisseaux Une grille G_0 est appelé *vaisseau* s'il existe un temps $t > 0$ tel que G_t soit égale à G_0, à translation près. Autrement dit, s'il existe un *vecteur de translation* (a, b) tel que pour tout (i, j), la cellule de coordonnées $(i + a, j + b)$ de G_t soit égale à la cellule de coordonnées (i, j) de G_0 (fig. 5). Le temps minimal $p > 0$ tel que G_p soit une grille translatée de G_0 est la *période* du vaisseau. La *vitesse* d'un vaisseau de période p et de vecteur de translation (a, b) est $V = \max(|a|, |b|)/p$, où $|a|$ et $|b|$ sont les valeurs absolues de a et b.

(a) Dessiner les étapes intermédiaires du vaisseau de la figure 5. Quelle est la vitesse de ce vaisseau ?

(b) On considère un vaisseau de dimension n, de période p et de vecteur de translation (a, b). Montrer que la grille est périodique. Que peut-on dire de sa (plus petite) période ?

(c) Un vaisseau peut-il avoir plusieurs vecteurs de translation qui ne soient pas égaux *modulo* n ?

(d) Écrire une fonction `est_vaisseau` qui prend en entrée une grille et teste si la grille est un vaisseau. La fonction renvoie uni couple (p, V) où p est la période du vaisseau, et V la liste de ses vecteurs de translation. Si la grille n'est pas un vaisseau, la fonction renvoie le couple $(0, [])$.

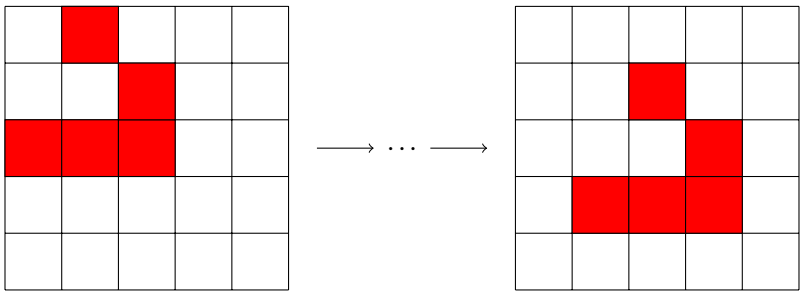


FIGURE 5 – Vaisseau de période 4 et de vecteur de translation $(1, -1)$.