## 1 - Généralités

Nous allons étudier les automates cellulaires. Nous nous limiterons au cas des automates cellulaires de dimension 1.

On note $A = \{0, 1\}$. Soit $r$ un entier non nul, on appellera *$r$-automate cellulaire (de dimension 1)* la donnée d'une *fonction de transition* $f : A^{2r+1} \to A$. On pose alors $F : A^{\mathbb{Z}} \to A^{\mathbb{Z}}$ la *fonction globale associée* définie par

$$\forall i \in \mathbb{Z}, [F(x)](i) = f(x(i-r), x(i-r+1), \ldots, x(i), x(i+1), \ldots, x(i+r)).$$

Soit $x^0 : \mathbb{Z} \to A$, on construit de proche en proche des « tableaux » $x^n$ pour $n \in \mathbb{N}$ en posant : $\forall n \in \mathbb{N}, x^{n+1} = F(x^n)$.

On définit le type `type automate = {degre : int ; transition = int array}` où le degré est l'entier $r$ et on code la fonction de transition $f$ d'un $r$-automate cellulaire par un tableau `f` de longueur $2^{2r+1}$ où `f.(i)` désigne l'image par $f$ de $(x_{-r}, x_{-r+1}, \ldots, x_0, \ldots, x_r)$ où $i = \sum_{k=0}^{2r} x_{r-k} 2^k$.

1) Quel est le tableau qui code l'automate $f$ défini par

$$\forall (a, b, c) \in A^3, f(a, b, c) = 1 \iff (a, b, c) \in \{(1, 1, 0), (0, 1, 1), (1, 1, 1)\}?$$

2) Écrire un fonction `log2 : int -> bool*int` telle que `log2 n` renvoie `true,r` si $n = 2^r$ avec $r$ entier naturel et `false,r` si $n$ n'est pas une puissance de 2 (la valeur de $r$ étant alors sans importance). En déduire une fonction `auto : int array -> automate` qui associe à un tableau l'automate associé. La fonction calculera le degré en s'assurant que la dimension du tableau est correcte.

3) Écrire une fonction `compose : int array -> int -> int -> int` telle que, si `x` est un tableau d'entiers, `compose x r debut` renvoie l'entier $i$ défini par $i = \sum_{k=0}^{2r} x_{debut+2r-k} 2^k$.

4) Soit $N \in \mathbb{N}$, on stocke $x(-N), x(-N+1), \ldots, x(0), \ldots, x(N)$ dans un tableau `x` de longueur $2N+1$. Pour quels $i \in \mathbb{Z}$ peut-on calculer $[F(x)](i)$ ?

5) a) Écrire une fonction `transition : automate -> int array -> int array` telle que `transition f x` renvoie le tableau qui correspond aux éléments de $F(x)$ que l'on peut calculer.

   b) Calculer `transition f x` où `f = {degre = 1 ; transition = [|0;1;1;0;1;1;1;0|]}` et `x = [|0;1;1;0;1;0;0;1;1;0;0;0;1|]` et vérifier que le résultat renvoyé est bien `[|0;1;1;1;1;1;0;1;1;0;1|]`.

6) a) En déduire une fonction `iteration : automate -> int array -> int -> int array` telle que `iteration f x n` renvoie le tableau qui correspond aux éléments de $x^n$ que l'on peut calculer où $x^0$ est donné par le tableau `x`.

   b) Calculer `iteration f x 3` où `f` et `x` sont ceux de la question 5.b et vérifier que le résultat obtenu est bien `[|0;0;1;0;1;1;0|]`.

7) Écrire une fonction `matrice : automate -> int array -> int -> int array array` telle que `matrice f x n` renvoie le tableau `t` tel que la première ligne correspond à $x^0$, la deuxième ligne correspond $x^1, \ldots$, la $(n-1)$ ème ligne correspond à $x^{n-1}$. On ferra attention à obtenir un tableau rectangulaire dont la largueur est donc déterminée par la taille de la $(n-1)$-ème itération.

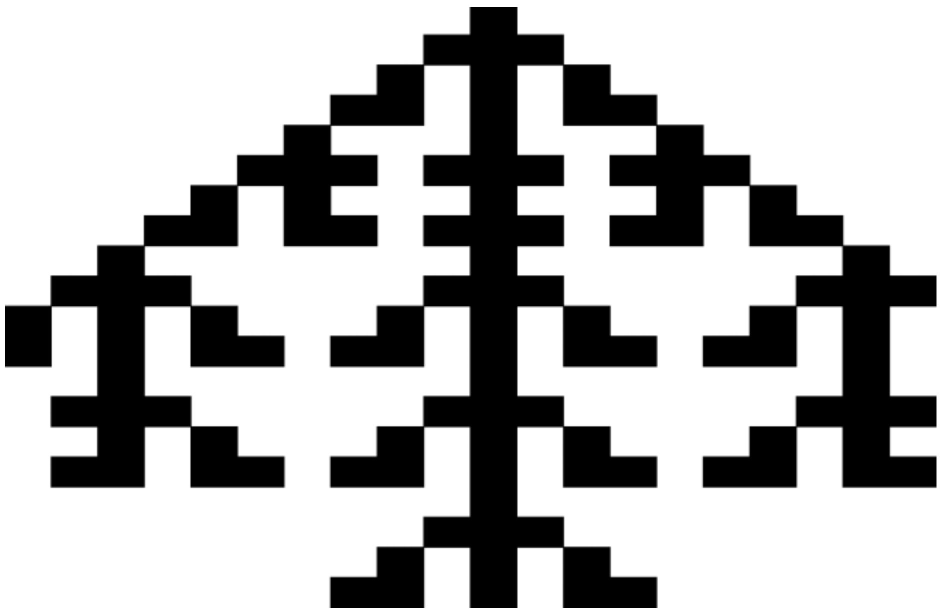
## 2 - Classification de Wolfram

Soit $f$ un 1-automate cellulaire, on lui associe l'entier $w(f) = \sum_{i=0}^{7} 2^i f(d(i))$ où $d : [0, 7] \to A^3$ associe à un entier $i$ sa décomposition en base 2 (poids fort en tête) éventuellement complété par des zéros (par exemple $d(3) = (0, 1, 1)$). On admet que cette fonction réalise une bijection entre l'ensemble des 1-automates cellulaires et $[0, 255]$.

8) Écrire une fonction `wolfram : int -> automate` qui associe à tout entier $i$ l'unique automate $f$ tel que $w(f) = i$. Vérifier que le résultat obtenu pour `wolfram 150` est bien $\{degre = 1; transition = [|0; 1; 1; 0; 1; 0; 0; 1|]\}$

9) Écrire une fonction `matriceWolfram : int -> int -> int array array` telle que `matriceWolfram i n` renvoie la matrice à $n$ lignes et $n$ colonnes représentant l'automate cellulaire $f$ tel que $w(f) = i$ où $x^0$ est pris tel que $x^0(k) = 1 \iff k = 0$.

10) À l'aide de la fonction `dessineMatrice` que vous trouverez dans l'espace partagé de votre classe, vérifier que le résultat obtenu pour `matriceWolfram 150 20` est bien l'image ci-dessous, puis tracer les automates 50, 126 et 30.



## 3 - Automate cellulaire probabiliste

On veut maintenant considérer des automates cellulaires probabilistes. Dans notre cas, la donnée d'un $r$-automate cellulaire probabiliste, sera la donnée d'une fonction de transition $f : A^{2r+1} \to A$ et d'un réel $p \in [0, 1]$. Si on se donne une configuration $x^0 \in A^{\mathbb{Z}}$ initiale, on considère alors la suite $(x^n) \in (A^{\mathbb{Z}})^{\mathbb{N}}$ définie par

$$\forall n \in \mathbb{N}, \forall i \in \mathbb{Z}, x^{n+1}(i) = \begin{cases} F(x^n)(i) & \text{avec la probabilité } p \\ x^n(i) & \text{avec la probabilité } 1-p \end{cases}$$

Cela peut se modéliser en prenant une suite de variables aléatoires $(X_{n,i})_{(n,i)\in\mathbb{N}\times\mathbb{Z}}$ indépendantes et suivant toutes la loi de Bernoulli de paramètre $p$. On adapte alors notre type :

`type automateProba = {degreProba : int ; transitionProba = int array ; proba : float}.`

11) Écrire les fonctions `transitionProba, iterationProba` et `matriceProba` qui généralisent les fonctions des questions 5,6 et 7 pour les automates probabilistes. On pourra utiliser l'instruction `random__float 1.` qui renvoie un nombre (flottant) au hasard entre 0 et 1.

12) On veut simuler un embouteillage.
    a) Déterminer un 1-automate (non probabiliste) qui modélise le fait qu'une voiture avance quand la case devant elle est libre.
    b) Dessiner l'évolution de cet automate en prenant au début une configuration $x^0$ où $x^0(i) = 1$ avec la probabilité $\frac{1}{2}$. Que constate-t-on ?
    c) Refaire la même chose mais en supposant que la voiture n'avance qu'avec une probabilité $p = \frac{1}{2}$.