

## 1 Création d'un labyrinthe

L'algorithme employé pour la génération de labyrinthes utilise des propriétés qui s'apparentent au *Quick Union Find*; algorithme de recherche et de construction de graphes de connexité dans un ensemble d'éléments donné.

Les labyrinthes créés ici sont au minimum 1-connexes<sup>1</sup>; à partir de ce type de labyrinthe<sup>2</sup>, nous pouvons à tout moment augmenter la connexité en cassant des murs, et de ce fait créer d'autres chemins possibles<sup>3</sup>.

L'embryon du labyrinthe est tel que toutes les positions de départ (les nœuds) sont entourées de 4 murs. Il se présente sous la forme d'une matrice d'entiers de dimensions  $(2N + 1, 2M + 1)$  où  $N \times M$  est le nombre de cases non-murées. Dans chacune de ces cases, un identifiant unique est placé (de 0 à  $N \times M - 1$ ). Nous posons la valeur -1 pour les cases murées. À l'initialisation, nous obtenons la matrice  $L$  pour  $N = M = 3$  :

$$L = \begin{pmatrix} -1 & -1 & -1 & -1 & -1 & -1 & -1 \\ -1 & \mathbf{0} & -1 & \mathbf{1} & -1 & \mathbf{2} & -1 \\ -1 & -1 & -1 & -1 & -1 & -1 & -1 \\ -1 & \mathbf{3} & -1 & \mathbf{4} & -1 & \mathbf{5} & -1 \\ -1 & -1 & -1 & -1 & -1 & -1 & -1 \\ -1 & \mathbf{6} & -1 & \mathbf{7} & -1 & \mathbf{8} & -1 \\ -1 & -1 & -1 & -1 & -1 & -1 & -1 \end{pmatrix}$$

Chacune des cases non-murées se trouve dans une des composantes d'un même graphe; chaque composante est réduite à un seul nœud. Pour  $N = M = 3$ , un graphe à 9 composantes connexes est créé.

Afin d'obtenir un graphe à une composante connexe, nous sélectionnons aléatoirement un mur séparant deux composantes disjointes; le mur est détruit et l'identifiant d'une des deux composantes est propagé sur les nœuds de l'autre. Nous choisissons de propager la valeur du plus petit identifiant.

Quand les  $N \times M$  nœuds de départ prennent<sup>4</sup> la valeur<sup>5</sup> 0, alors toutes les composantes du graphe sont jointes; l'unique graphe résultant est 1-connexe<sup>6</sup>.

### 1.1 Détail de l'algorithme

1. Initialisation :
  - Une matrice  $L[N][M]$  donnée;
  - $k \leftarrow 0$ ;
  - Pour  $i$  de 0 à  $N$  faire :
    - Pour  $j$  de 0 à  $M$  faire :
      - Si ( $i$  impair et  $j$  impair) alors :
        - $L[i][j] \leftarrow k$ ;
        - $k = k + 1$ ;
      - Sinon :  $L[i][j] \leftarrow -1$ ;
      - FIN "Si";
    - FIN "Pour";
  - FIN "Pour";
2. Pour  $NbCasesAZero$ , le nombre de cases à 0 :  $NbCasesAZero = 1$ ;
3. Tant que  $NbCasesAZero < N \times M$  faire :

- 
1. Pour tout point (emplacement non muré) dans ce labyrinthe, il existe un et un seul chemin le reliant à n'importe quel autre point.
  2. Assimilable à un graphe acyclique.
  3. Assimilable à un graphe cyclique.
  4. Ou dont la valeur était déjà à 0.
  5. Dans le cadre d'une représentation matricielle, pour un nœud du graphe, la valeur contenue dans une case à l'initialisation doit être différente de -1.
  6. Graphe à une composante 1-connexe.

- Prendre au hasard  $(x, y)$  tel que :  
 $L[x][y] = -1$  et  $(x$  impair ou  $y$  impair<sup>7</sup>) ;
- Si  $x$  impair alors :
  - $d \leftarrow L[x][y-1] - L[x][y+1]$  ;
  - Si  $d > 0$  alors :
    - $L[x][y] \leftarrow L[x][y+1]$  ;
    - Propager la valeur  $L[x][y+1]$  à partir du point  $(x, y-1)$  (en 4-connexité) pour les cases contenant la valeur  $L[x][y-1]$ <sup>8</sup> ;
  - Sinon, si  $d < 0$  alors :
    - $L[x][y] \leftarrow L[x][y-1]$  ;
    - Propager la valeur  $L[x][y-1]$  à partir du point  $(x, y+1)$  (en 4-connexité) pour les cases contenant la valeur  $L[x][y+1]$ <sup>8</sup> ;
- Sinon ( $y$  impair) :
  - $d \leftarrow L[x-1][y] - L[x+1][y]$  ;
  - Si  $d > 0$  alors :
    - $L[x][y] \leftarrow L[x+1][y]$  ;
    - Propager la valeur  $L[x+1][y]$  à partir du point  $(x-1, y)$  (en 4-connexité) pour les cases contenant la valeur  $L[x-1][y]$ <sup>8</sup> ;
  - Sinon, si  $d < 0$  alors :
    - $L[x][y] \leftarrow L[x-1][y]$  ;
    - Propager la valeur  $L[x-1][y]$  à partir du point  $(x+1, y)$  (en 4-connexité) pour les cases contenant la valeur  $L[x+1][y]$ <sup>8</sup> ;

4. FIN “Tant que”.

## 1.2 Exemple de propagation et résultat

+	+	+	+	+	+			+	+	+	+	+	+	+
+	0	+	<b>1</b>	<b>1</b>	<b>1</b>	+		+	0	+	<b>0</b>	<b>0</b>	<b>0</b>	+
+	0	+	+	+	<b>1</b>	+		+	0	+	+	+	<b>0</b>	+
+	0	<b>0</b>	<b>0</b>	+	<b>1</b>	+	→	+	0	<b>0</b>	<b>0</b>	<b>0</b>	<b>0</b>	+
+	+	<b>0</b>	+	+	+	+		+	+	<b>0</b>	+	+	+	+
+	<b>0</b>	<b>0</b>	<b>0</b>	<b>0</b>	<b>0</b>	+		+	<b>0</b>	<b>0</b>	<b>0</b>	<b>0</b>	<b>0</b>	+
+	+	+	+	+	+	+		+	+	+	+	+	+	+

$$NbCasesAZero \leftarrow NbCasesAZero + 3$$

Nous obtenons pour  $N = M = 200$  (matrice de  $401 \times 401$ ) le labyrinthe figure 1 :

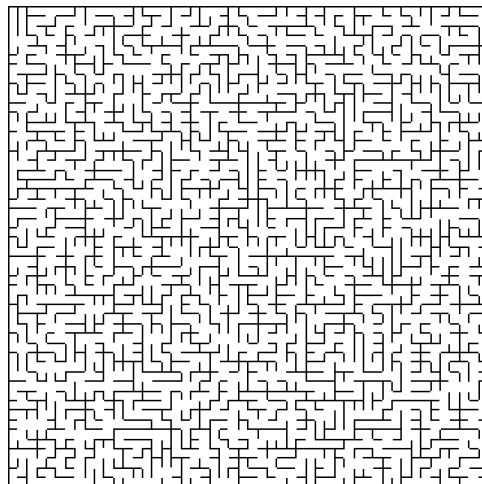


FIGURE 1 – Labyrinthe ( $N = M = 200$ ) généré aléatoirement.

7. Si  $x$  et  $y$  sont tous les deux impairs, alors  $L[x][y] \neq -1$

8. Pendant la propagation, incrémenter  $NbCasesAZero$  s’il y a lieu (comme décrit plus haut).

## 2 Le plus court chemin dans un labyrinthe

Nous allons décrire ici un algorithme pour calculer les plus courts chemins dans un labyrinthe ; cette méthode est basée sur un algorithme de recherche en profondeur d'abord dans un arbre. Il s'agit de trouver le plus court chemin dans un graphe orienté  $G = (S, A)$ . Dans ce graphe, les sommets sont reliés entre eux par des arêtes, chacune est notée  $a_i$  et possède un poids  $p_i$ . Le coût de parcours du chemin  $C = \{a_1, \dots, a_n\}$  équivaut à la somme des poids des arêtes qui le constituent. Dans le cas particulier du labyrinthe<sup>9</sup>, tous les  $p_i$  valent 1. Nous utilisons la matrice du labyrinthe pour stocker les informations liées au coût de déplacement, cf. figure 2.

### 2.1 Détail de l'algorithme

$Pcc(x_1, y_1, x_2, y_2, v)$

Début :

$v \leftarrow v + 1$  ;

$L[x_1][y_1] \leftarrow v$  ;

Si  $x_1 = x_2$  et  $y_1 = y_2$  alors :

retour ;

Pour chaque  $d(x_{dir}, y_{dir})$  pris parmi les quatre directions possibles :

Si  $v < L[x_1 + x_{dir}][y_1 + y_{dir}]$  ou  $L[x_1 + x_{dir}][y_1 + y_{dir}] = 0$  alors :

$Pcc(x_1 + x_{dir}, y_1 + y_{dir}, x_2, y_2, v)$  ;

Fin.

1. Initialiser le labyrinthe<sup>10</sup>  $M \times N$  ;
2. Pour un point de départ  $(x_d, y_d)$  et un point d'arrivée  $(x_f, y_f)$  faire :  
 $Pcc(x_d, y_d, x_f, y_f, 0)$  ;
3. Partir du point d'arrivée et reconstruire le chemin (pour un point à valeur  $v$ , prendre le voisin qui a pour valeur  $v - 1$ ) jusqu'au point de départ ;

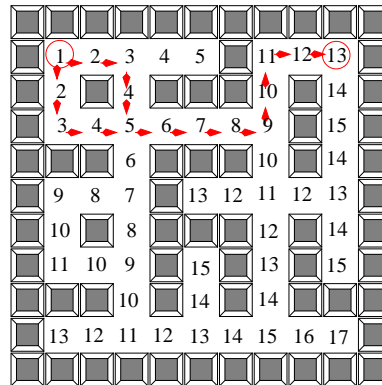


FIGURE 2 – Schéma de résolution du plus court chemin dans un labyrinthe.

9. Ici un graphe non pondéré

10. Nous obtenons une matrice  $L / \forall a_{i,j} \in L ; a_{i,j} \in \{-1, 0\}$ .