Dédale It Yourself

Louis Vanhaelewyn

Table des matières

1 Définitions mathématiques

1.1 Graphe

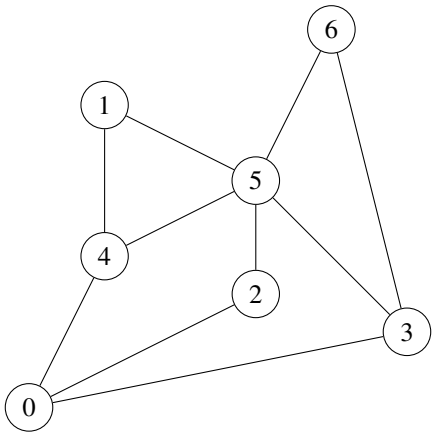


FIGURE 1 – Exemple de représentation de graphe

Définition 1. Graphe

*Un **graphe** G est la donnée d'un couple d'ensembles,* $\mathrm{G} = (\mathrm{V}, \mathrm{E})$.

- *l'ensemble* V *(vertices en anglais), représente les **sommets** du graphe,*
- *l'ensemble* $\mathrm{E} \subset \mathrm{V} \times \mathrm{V}$ *(edges en anglais) représente quant à lui, les relations entre les sommets du graphe, aussi appelées arêtes.*

On ne s'intéressera ici qu'aux graphes finis, c'est-à-dire les graphes qui ne comportent qu'un nombre fini de sommets. On représente généralement un graphe par un ensemble de points symbolisant les éléments de V. Un trait, symbolisant les arêtes, relie deux sommets en relation. Ainsi pour l'exemple de la figure 1, le graphe $\mathrm{G} = (\mathrm{V}, \mathrm{E})$ est défini par :

$$\mathrm{V} = \{0, \ldots, 6\} \quad \mathrm{E} = \{\{0,2\}, \{0,3\}, \{0,4\}, \{1,4\}, \{1,5\}, \{2,5\}, \{3,5\}, \{3,6\}, \{4,5\}, \{5,6\}\}$$

Pour un sommet u, on note $\deg(u)$, le *degré* de u, à savoir le nombre de voisins de u.

Définition 2. Chemin

Soit $\mathrm{G} = (\mathrm{V}, \mathrm{E})$ *un graphe.*
*Un **chemin** de longueur* $r \geqslant 3$ *dans le graphe* G *du sommet* x *au sommet* y *est une famille,* $(x = x_1, \ldots, x_r = y)$ *de sommets de* V*, tels que pour tout* $i \in [\![1, r-1]\!]$, $(x_i, x_{i+1}) \in \mathrm{E}$*, on le notera souvent* $x_1 \to \cdots \to x_r$.
Si un chemin de x *à* y *existe on dira que* x *est **relié** à* y.
Dans le cas où x *est relié à* y*, on peut définir la **distance** de* x *à* y *comme étant la longueur minimale d'un chemin de* x *à* y.

Définition 3. Connexe

*On dit qu'un graphe est **connexe** si pour tous sommets* x *et* y*,* x *est relié à* y.

Proposition 1. *Soit* $\mathrm{G} = (\mathrm{V}, \mathrm{E})$*, un graphe connexe. Alors* $|\mathrm{E}| \geqslant |\mathrm{V}| - 1$.

Démonstration.
On raisonne par récurrence sur $n = |\mathrm{V}|$ le nombre de sommets.
Si $n = 1$, le graphe est réduit à un sommet et est donc connexe.

Soit $n \in \mathbb{N}^*$, supposons que si le graphe est connexe, et possède $m \leqslant n$ sommets, alors il possède au moins $m-1$ arêtes. Soit $\mathrm{G} = (\mathrm{V}, \mathrm{E})$ un graphe connexe à $n+1$ sommets. Soit u un sommet de G quelconque, en supprimant u dans G, on obtient un graphe constitué de k composantes connexes. On obtient au plus une nouvelle composante connexe par arête sortant de u, donc $k \leqslant \deg(u)$. On applique alors l'hypothèse de récurrence sur chaque composante connexe. On en déduit que $\mathrm{G}\backslash\{u\}$, possède au plus $n-k$ arêtes. En ajoutant le sommet u, G possède au plus $n-k+\deg(u) \geqslant n = (n+1)-1$. Ce qui propage la récurrence, et clôt la démonstration. □

Définition 4. Cycle

*Un **cycle** est un chemin d'un sommet x à lui-même, de longueur strictement supérieure à 1, et passant par des sommets deux à deux distincts.*
*Lorsqu'un graphe ne possède pas de cycle, on dit qu'il est **acyclique**.*

On en fournit un exemple en figure 2, avec le cycle $0 \to 5 \to 1 \to 2 \to 0$.

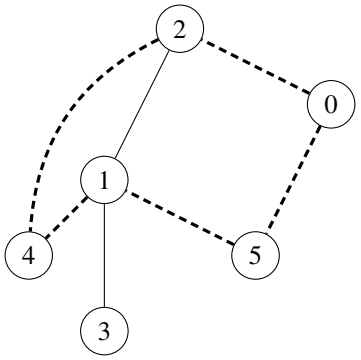


FIGURE 2 – Exemples de cycle dans un graphe : (0,2,1,5), (1,2,4) et (1,4,2,0,5)

Proposition 2. *Soit $\mathrm{G} = (\mathrm{V}, \mathrm{E})$, un graphe sans cycle. Alors $|\mathrm{E}| \leqslant |\mathrm{V}| - 1$.*

La contraposée de cette proposition est aussi intéressante, à savoir qu'un graphe $\mathrm{G} = (\mathrm{V}, \mathrm{E})$, tel que $|\mathrm{E}| \geqslant |\mathrm{V}|$ possède un cycle. C'est cette proposition que nous allons prouver.

Démonstration.

Raisonnons par récurrence. Tout d'abord pour que $|\mathrm{E}| \geqslant |\mathrm{V}|$, il nous faut $|\mathrm{E}| \geqslant |\mathrm{V}| \geqslant 3$. Auquel cas, G est un triangle, et possède donc un cycle.
Soit $n \in \mathbb{N}$, supposons que, pour tout graphe G de n sommets, s'il possède plus de n arêtes alors il possède un cycle. Soit G un graphe de $n+1$ sommets, et $n+1$ arêtes. Si G possède un sommet u de degré 1, on considère le graphe $\mathrm{G}\backslash\{u\}$, qui possède lui n sommets et plus de n arêtes et possède donc un cycle. Sinon G ne possède que des sommets de degré 2. En considérant un chemin $x_1 \to \cdots \to x_k$ de sommets deux à deux distincts de longueur maximale dans G, on remarque que x_1 qui ne peut être de degré 1, possède un voisin différent de x_2, mais qui comme le chemin est maximal, doit être égal à l'un des $(x_i)_{2 \leqslant i \leqslant k}$, notons le x_{i_0}, on obtient alors un cycle $x_1 \to \cdots \to x_{i_0} \to x_1$. Dans tous les cas, le graphe possède un cycle, et la récurrence se propage. □

1.2 Arbres

Définition 5. Arbre

*Un **arbre** est un graphe connexe acyclique.*
*On choisira parfois un sommet particulier appelé **racine** qui servira de base à l'arbre.*
*On nomme **profondeur** la distance d'un sommet à la racine.*

On représente parfois un arbre en dessinant sa racine en haut et les sommets de même profondeur à la même hauteur. En figure 3a, on peut observer une représentation d'arbre sous forme arborescente, la racine arbitraire est ici 0. Il est à noter qu'on aurait pu choisir d'enraciner l'arbre à partir de n'importe quel sommet.

La caractérisation suivante des arbres nous sera utile dans la suite de l'ouvrage. Elle découle immédiatement des propriétés 1 et 2 précédentes.

Proposition 3. *Soit* $\mathrm{G} = (\mathrm{V}, \mathrm{E})$ *un graphe à* n *sommets, les propositions suivantes sont équivalentes,*

i. G *est un arbre,*

ii. G *est connexe et possède* $n - 1$ *arêtes,*

iii. G *est acyclique et possède* $n - 1$ *arêtes.*

En particulier un arbre à n sommets possède $n - 1$ arêtes.

(a) Représentation d'un arbre sous forme arborescente

(b) Représentation d'un arbre sous forme de graphe quelconque

FIGURE 3 – Exemple de représentation du même arbre

La représentation arborescente laisse apparaître des sommets particuliers de degré 1, pour poursuivre la métaphore botanique, on nomme ces sommets **feuilles**.

Proposition 4. *Un arbre de* $n \geqslant 2$ *sommets possède au moins* 2 *feuilles.*

Démonstration.

Soit T l'arbre de $n \geqslant 2$ sommets considéré. En s'inspirant de la preuve de la proposition 2, on se donne un chemin de taille maximum dans T. On montre alors que les sommets aux deux extrémités du chemin de degré 1. En effet, soit u un tel sommet, relié à v à travers le chemin, et w un autre voisin de u. Si w n'appartient pas au chemin, on pourrait prolonger celui-ci ce qui contredirait sa maximalité. Si au contraire w appartient au chemin, alors T possède un cycle ce qui est impossible. En conclusion u est de degré 1. Chaque extrémité du chemin est donc une feuille. □

1.3 Arbre couvrant

Définition 6. Arbre couvrant

Soit G *un graphe, on nomme* ***arbre couvrant*** *tout sous-graphe de* G *qui est un arbre et dont les sommets sont exactement ceux de* G.

La figure 4, donne un exemple d'un tel arbre couvrant.

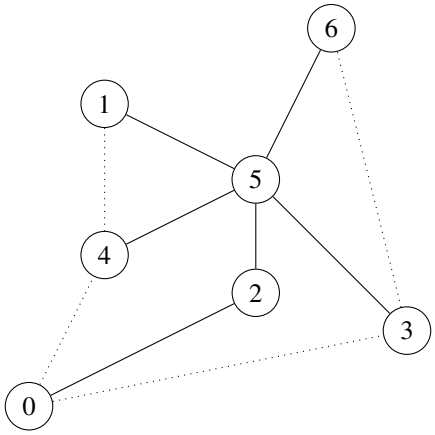


FIGURE 4 – Exemple d'arbre couvrant du graphe G de la figure 1

On remarque que tout graphe G qui possède un arbre couvrant est connexe. Il se trouve que c'est en fait une équivalence par la proposition suivante :

Proposition 5. *Un graphe est connexe si et seulement s'il possède un arbre couvrant.*

Démonstration.

On montre le sens direct de cette propriété par récurrence sur le nombre de sommets du graphe. Si le graphe est restreint à un sommet, la propriété est immédiate. Soit $n \in \mathbb{N}$, supposons que la propriété soit vraie pour tous les graphes de taille inférieure ou égale à n. Soit G un graphe connexe de $n+1$ sommets. On choisit un sommet u quelconque de G. On considère le graphe $\mathrm{G}\backslash\{u\}$, et en particulier ses composantes connexes, notons les $\mathrm{C}_1, \ldots, \mathrm{C}_k$, on remarque que chacune est relié à u par une arête de G. Pour chaque composante connexe C_i, comme $|\mathrm{C}_i| \leqslant n$, on peut appliquer l'hypothèse de récurrence et en déduire l'existence d'un arbre couvrant T_i de C_i. En notant e_i une arête quelconque reliant u à C_i, on s'intéresse au graphe dont l'ensemble des arêtes est :

$$\mathrm{T} = \bigcup_{i=1}^{k} (\mathrm{T}_i \cup \{e_i\})$$

On confondra ici, ensemble des arêtes et graphe lui-même. Montrons que ce graphe est un arbre, à savoir qu'il est connexe et acylique. Pour montrer qu'il est connexe, il suffit de montrer que chaque sommet est relié à u. Soit x un sommet de G, alors $x \in \mathrm{C}_i$ pour un certain i, de plus en étudiant l'arête e_i précédente, $e_i = (u, v_i)$ pour un certain sommet v_i. Comme C_i est connexe, il existe un chemin de x à v_i dans T_i, puis on prolonge le chemin à u par e_i. On a trouvé un chemin de x à u dans T.

Montrons maintenant que T est acyclique et raisonnons pour cela par l'absurde. Si le graphe possède un cycle, celui-ci ne peut être inclus dans une seule composante connexe C_i car T_i est acyclique. Alors en particulier si un sommet d'une composante C_i appartient au cycle, on peut s'intéresser à la première fois où le cycle quitte la composante C_i, dans un sens ou dans l'autre ; on dispose alors de deux arêtes e et e', dont l'une des extrémités est dans C_i et l'autre en dehors. Or C_i n'est connecté avec le reste du graphe que par u avec l'arête e_i, donc $e = e' = e_i$, ce qui est impossible car les sommets d'un cycle sont deux à deux distincts. On a montré que le graphe était acylique. Donc T est un arbre couvrant. □

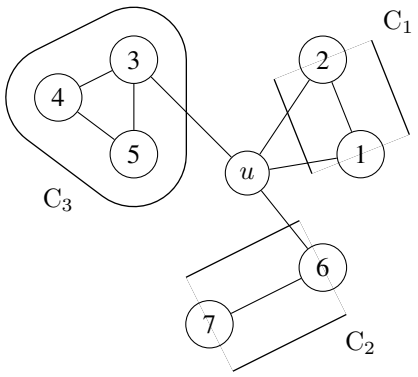


FIGURE 5 – Schéma de la récurrence de la preuve de la proposition 5

1.4 Modélisation

Les définitions précédentes nous permettent de modéliser mathématiquement la notion de labyrinthe. On peut modéliser la grille sur laquelle est construite le labyrinthe comme un graphe $\mathrm{G} = (\mathrm{V}, \mathrm{E})$, dont les sommets sont les cellules du labyrinthe, et les arêtes représentent la posibilité de passer d'une cellule à une autre.

Une grille carrée de taille 5 sera donc representée par le graphe ci-dessous en figure 6, où l'on a successivement numéroté les cellules puis représentée chacune par un sommet du graphe.

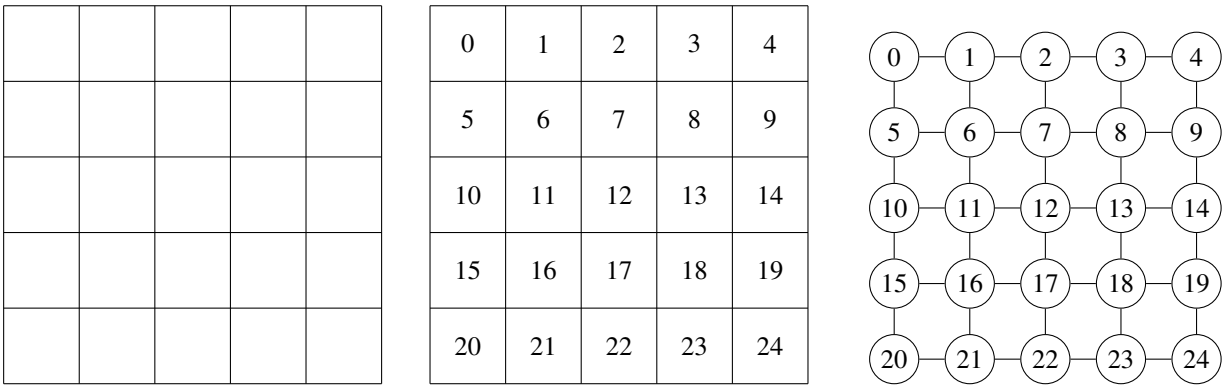


FIGURE 6 – Modélisation de la grille sous forme de graphe (on représente tout de même les murs pour mieux visualiser la structure de la grille)

Une fois cette première modélisation de la matrice du labyrinthe, il nous reste à modéliser le labyrinthe lui-même. La notion d'arbre couvrant, précédemment introduite est idéale pour cela. Comme un arbre couvrant notre grille nous fournit un sous-graphe, dans lequel chaque paire de cellules est connectée par un unique chemin, on obtient exactement la description de ce que l'on avait donné pour un labyrinthe ! à savoir que deux endroits de notre labyrinthe sont reliés par un unique chemin. Dans notre modèle, l'absence de mur entre deux cellules est modélisée par une arête. On peut dès lors observer un labyrinthe et sa modélisation dans la figure 7 suivante.

Notons au passage que la proposition 5 garantit l'existence d'un labyrinthe dès lors que la grille sur laquelle on souhaite le construire est connexe.

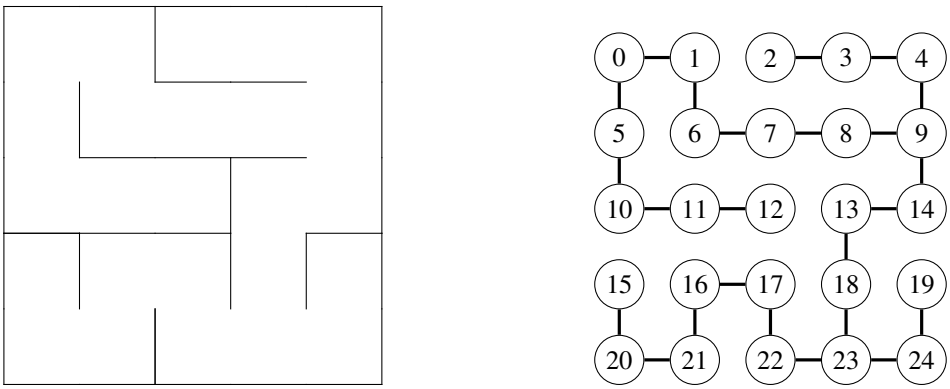


FIGURE 7 – Un labyrinthe et sa modélisation

2 Algorithme de division

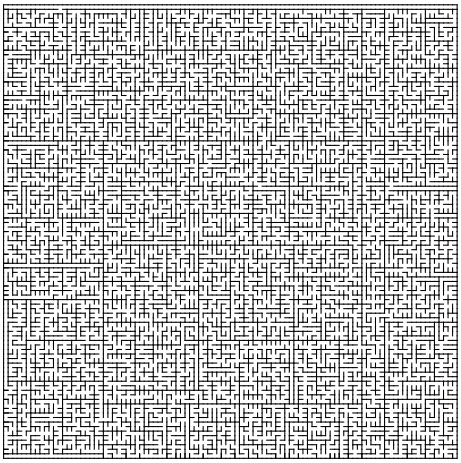
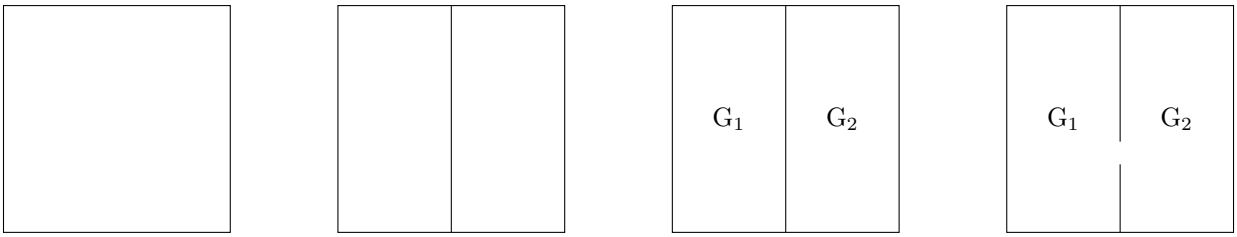


FIGURE 8 – Exemple de labyrinthe généré par l'algorithme de division



(a) Grille initiale (b) Création du mur (c) Application récursive (d) Ajout du passage

FIGURE 9 – Schéma explicatif de la récursion

Dans l'implémentation utilisée pour générer cette figure, on construit un mur de sorte à couper le rectangle en deux dans sa largeur.

2.1 Modélisation / Formalisation

Essayons de formaliser un peu cet algorithme. Initialement, le graphe considéré est la grille $n \times m$ complète. Pour modéliser la notion de "mur", coupant notre labyrinthe en deux, on introduit la notion de **coupe**.

Soit $G = (V, E)$ un graphe, une **coupe** de G est la donnée de deux sous-ensembles de V, U_1, U_2 disjoints non vides, tels que $V = U_1 \cup U_2$. Ce qui nous intéresse ici particulièrement est l'ensemble des arêtes qui relient U_1 à U_2 dans G, et que nous noterons $E_G(U_1, U_2)$.

Dans notre cas, on souhaite pouvoir appliquer récursivement notre algorithme à des sous-graphes induits de taille inférieure, afin qu'il nous renvoit un arbre couvrant. Pour cela, on demande aux sous-graphes induits par notre coupe d'être connexes, pour qu'ils possèdent des arbres couvrants (voir proposition 5).

On fixe désormais un graphe $G = (V, E)$. On suppose disposer d'une fonction φ qui à un sous-graphe induit H de G, renvoie $\varphi(H) = (U_1, U_2)$, une coupe de H.

On souhaite construire un labyrinthe Lab qui à un graphe en entrée, renvoie un arbre couvrant de ce graphe. En utilisant φ, on découpe notre graphe en entrée en deux sous-graphes induits connexes $G_1 = (U_1, E_1), G_2 = (U_2, E_2)$, sur lesquels on peut appliquer Lab et obtenir deux arbres couvrants T_1, T_2 de G_1, G_2 respectivement. On tire ensuite une arête aléatoire e dans $E_G(U_1, U_2)$. On retourne ensuite le graphe de sommets V et d'arêtes celles de T_1, T_2 auxquelles on ajoute e.

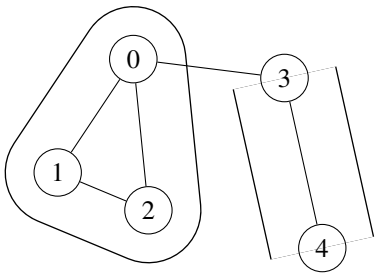


FIGURE 10 – Schéma d'une coupe avec $U_1 = \{0,1,2\}$, $U_2 = \{3,4\}$

Algorithme 1 Algorithme récursif de division

fonction LAB($G = (V, E)$)
  si $|V| = 1$ **alors**
    retourne G
  sinon
    $(U_1, U_2) = \varphi(G)$
    $T_1 = \text{Lab}(G[U_1])$
    $T_2 = \text{Lab}(G[U_2])$
    $e = \text{Aléatoire}(E_G(U_1, U_2))$
    retourne $(V, T_1 \cup T_2 \cup \{e\})$ ▷ on confond ici l'arbre et ses arêtes

2.2 Correction et terminaison

Pour chaque algorithme développé, il est nécessaire de s'intéresser à deux aspects : savoir si l'algorithme termine, et savoir s'il renvoie une solution correcte. Bien souvent la terminaison est prouvée par l'absurde, en considérant une suite d'entiers positifs strictement décroissante. Dans notre cas par exemple, le cardinal des ensembles de sommets U_1 et U_2 est strictement inférieur à celui de V. Si l'algorithme ne terminait pas, on obtiendrait à terme un ensemble d'arêtes de taille négative, ce qui est impossible. L'algorithme termine donc.

Ce premier algorithme nous permet d'introduire des démonstrations formelles de correction d'algorithmes. En particulier, on souhaite ici montrer que notre algorithme renvoit bien un arbre couvrant du graphe.

2.3 Biais

Cet algorithme, malgré sa simplicité, présente un défaut majeur, dont on se rend bien compte lorsque l'on en observe plusieurs itérations : il ne permet pas de générer tous les labyrinthes possibles. Le premier labyrinthe auquel on peut penser est la spirale, ou l'escargot (voir Figure 11). En effet la première étape de l'algorithme de division est de diviser en deux une pièce vide et d'y placer un mur ne contenant qu'une seule « porte ». Dans notre contre-exemple chaque ligne ou colonne possède au moins deux murs, ce qui rend impossible le fait d'obtenir ce labyrinthe avec cet algorithme.

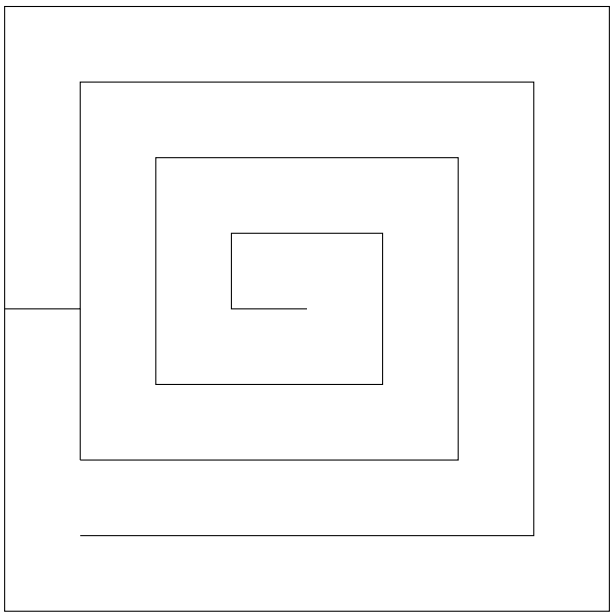


FIGURE 11 – Exemple de labyrinthe ne pouvant être atteint par l'algorithme

3 Algorithme d'Aldous-Broder

3.1 L'enjeu

On a remarqué dans simulations les deux précédents algorithmes des biais dans la génération des labyrinthes. Qu'est-ce qu'une loi sans biais ?

Sans introduire plus de formalisme, un algorithme génèrera un arbre sans biais lorsque *la distribution de l'arbre généré suit une loi uniforme*. Autrement dit, chaque arbre a la même probabilité d'apparaître. Si l'on souhaite générer un tel arbre, il suffit de prime abord de lister tous les arbres possibles et d'en piocher un au hasard. Oui mais...

Théorème 1. *Soit $a(n)$ le nombre d'arbres couvrants sur une grille de taille $n \times n$,*

$$a(n) = \frac{2^{n^2-1}}{n^2} \prod_{\substack{0 \leqslant n_1 \leqslant n-1 \\ 0 \leqslant n_2 \leqslant n-1 \\ (n_1,n_2) \neq (0,0)}} \left(2 - \cos\left(\frac{n_1 \pi}{n}\right) - \cos\left(\frac{n_2 \pi}{n}\right)\right)$$

Ce théorème nous donne le nombre exact de labyrinthes distincts dans une grille de taille n par n. Même si le produit est difficile à estimer, nous pouvons faire quelques applications numériques.

n	1	5	10	20
$a(n)$	1	557568000	$5{,}694 \times 10^{42}$	$7{,}897 \times 10^{186}$

En fait nous pouvons estimer le produit lorsque n tend vers l'infini,

$$a(n) \underset{n \to \infty}{\sim} \mathrm{C} \frac{(2e^{\mathrm{I}})^{n^2}}{n^2}$$

où C est une certaine constante et $\mathrm{I} = \dfrac{1}{\pi^2} \displaystyle\int_0^\pi \int_0^\pi \ln(2 - \cos(x) - \cos(y)) \, \mathrm{d}x \, \mathrm{d}y$

Ces remarques rendent impossible l'idée naïve de générer chaque labyrinthe et de piocher dedans.

3.2 L'algorithme

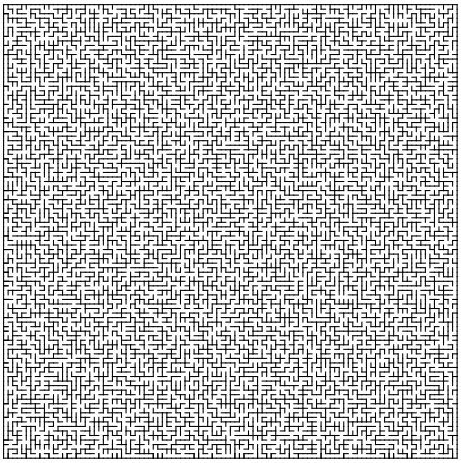


FIGURE 12 – Exemple de labyrinthe généré par l'algorithme d'Aldous-Broder

De manière intuitive l'algorithme peut s'imaginer comme une petite (nécessairement petite car c'est plus mignon) fourmi placée sur une case de la grille (peu importe la case), et se déplace de voisin en voisin au hasard.

Dès lors si la fourmi se déplace sur une case qu'elle n'avait pas précédemment visitée, on supprime, sur le labyrinthe, le mur séparant les deux cases. C'est ce déplacement aléatoire de proche en proche qui se nomme une *marche aléatoire*.

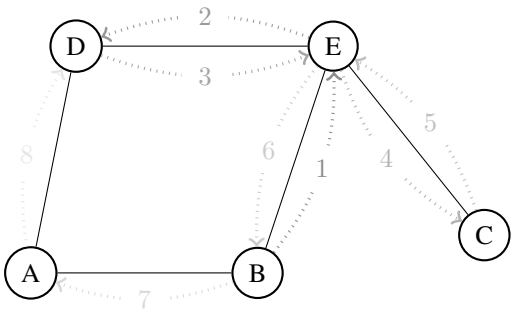


FIGURE 13 – Exemple de marche aléatoire sur un graphe. Les sommets parcourus sont B, E, D, E, C, E, B, A, D.

Cela nous conduit à la rédaction de l'algorithme formel suivant,

Algorithme 2 Algorithme d'Aldous-Broder

```
1:  fonction ALDOUS-BRODER(G = (V, E))
2:      u ← Aléatoire(V)                          ▷ on choisit un sommet u aléatoire
3:      M ← {u}
4:      T ← ∅
5:      tant que M ≠ V faire
6:          v ← Alatoire(Vois(u))                 ▷ on choisit un voisin de u aléatoire
7:          si v ∉ M alors
8:              M ← M ∪ {v}
9:              T ← T ∪ {(u, v)}
10:         u ← v
11:     retourne T
```

Mathématiquement, on obtient au cours de l'algorithme une suite $(X_n)_{n\in\mathbb{N}}$ de sommets de G, tels que $\{X_i, X_{i+1}\} \in E$.

3.2.1 Terminaison de l'algorithme

De prime abord l'algorithme peut ne pas terminer. Il termine en effet en un temps aléatoire, une fois avoir parcouru tous les sommets. Montrons que la probabilité que l'algorithme ne termine pas est nulle.

Soit $G = (V, E)$ le graphe sur lequel on fait tourner l'algorithme. Donnons-nous un *circuit* de G, à savoir un chemin qui passe par tous les sommets de G, notons le

$$\mathcal{C} = x_1 \to x_2 \to \cdots \to x_{r-1} \to x_r$$

L'idée de la preuve est la suivante, on découpe le temps en bloc de r étapes. La probabilité que durant une durée r la suite parcourt le circuit est strictement positive. Chaque bloc est *indépendant*, on peut donc faire le produit.

$$\begin{aligned}
\mathbb{P}(\text{l'algorithme ne termine pas}) &\leqslant \mathbb{P}\Big(\bigcap_{n\in\mathbb{N}} (X_{nr}, X_{nr+1}, \ldots, X_{(n+1)r-1}) \text{ ne suive pas } \mathcal{C}\Big) \\
&= \lim_{N\to+\infty} \prod_{n=1}^{N} \mathbb{P}((X_{nr}, X_{nr+1}, \ldots, X_{(n+1)r-1}) \text{ ne suive pas } \mathcal{C}) \\
&= \lim_{N\to+\infty} (1 - \mathbb{P}(X_1, \ldots, X_r) \text{ suive } \mathcal{C})^N \\
&= 0
\end{aligned}$$