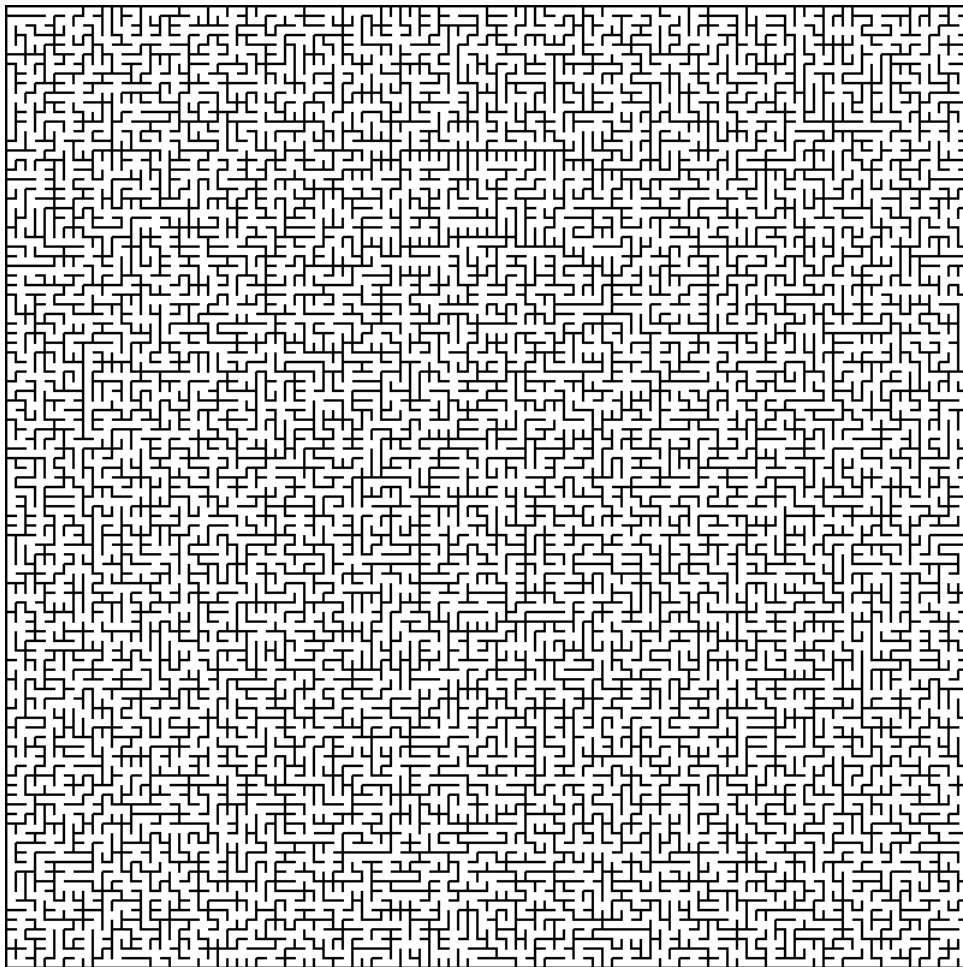


Bases de la Programmation

TP 10 – Labyrinthes

V. Padovani, PPS - IRIF



Un labyrinthe est dit *parfait* si, étant donnés deux emplacements quelconques dans celui-ci, il existe un et un seul chemin allant du premier vers le second. Le labyrinthe ci-dessus est parfait. Le but de ce TP est d'implémenter un algorithme générant aléatoirement des labyrinthes parfaits. L'algorithme est assez simple dans son principe. Son implémentation effective l'est un peu moins.

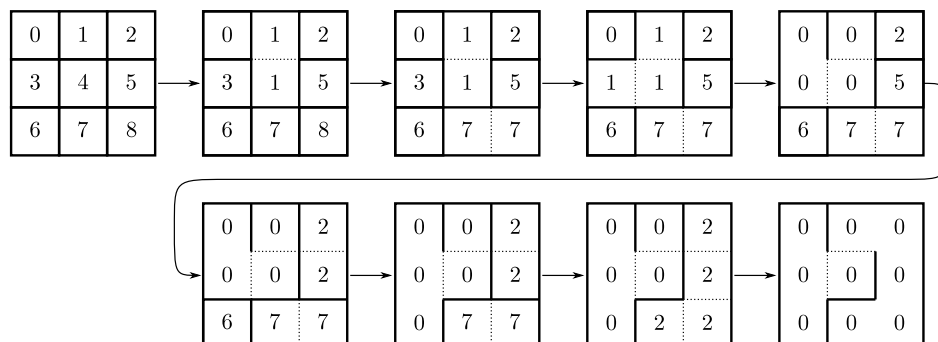
1 Algorithme de génération

L'algorithme fonctionne pour des labyrinthes de géométrie quelconque, mais je me limiterai ici au cas de murs alignés sur des lignes horizontales et verticales espacées de manière régulière, formant un quadrillage sur un rectangle. Chaque case de ce quadrillage est assimilée à une position dans le labyrinthe.

- Un *chemin* dans un labyrinthe est une suite de positions sans répétitions décrivant une suite de déplacements horizontaux et verticaux dans celui-ci, sans jamais traverser aucun mur.
- Une *zone* dans un labyrinthe est un ensemble maximal de positions tel que pour tout couple de positions dans cette zone, il existe un chemin allant de l'une vers l'autre. Cette zone est *parfaite* si, quel que soit le couple de positions choisi, ce chemin est unique. Un labyrinthe est *localement parfait* si chacune de ses zones est parfaite. Un labyrinthe *parfait* est un labyrinthe localement parfait d'au plus une zone.
- Un labyrinthe est *maximal* s'il n'est plus possible d'y construire aucun mur. Tout labyrinthe maximal est donc localement parfait - chaque zone est réduite à une position qui ne mène qu'à elle-même par le chemin vide.

L'algorithme part d'un labyrinthe maximal, dans lequel chaque zone reçoit un unique numéro. On itère ensuite le traitement suivant :

1. Choisir un mur séparant deux zones de numéros distincts. Si un tel mur n'existe pas, on interrompt le traitement.
2. Supprimer ce mur du labyrinthe.
3. Parmi les deux zones que ce mur séparait, renuméroter la zone de plus grand numéro par celui de la zone de plus petit numéro.



Le labyrinthe initial est maximal, donc localement parfait. Il n'est pas trop dur de voir qu'à chaque étape, le labyrinthe produit par suppression d'un mur est encore localement parfait, tout chemin dans la zone fusion des deux zones initiales et traversant l'ouverture créée ne pouvant passer que par cette ouverture¹. D'autre part, cette fusion fait strictement décroître le nombre de zones : le traitement termine donc nécessairement sur un labyrinthe localement parfait réduit à une unique zone, c'est-à-dire un labyrinthe parfait.

1. On peut le démontrer formellement.

2 Implémentation en C

Vous pouvez lire cette partie, ou non – il n’y a pas qu’une unique manière d’implémenter l’algorithme précédent.

Représentation du labyrinthe. On peut représenter un labyrinthe par une matrice d’entiers dont chaque case représente une case concrète de son quadrillage. Chaque case peut contenir deux informations : l’état de son mur Est (supprimé ou non), et l’état de son mur Sud (idem). Il suffit de définir deux constantes **EST** égale à 1, et **SUD** égale à 2, avec la convention suivante :

- Si les deux murs de la case ont été supprimés, sa valeur est 0.
- Si seul le mur Est est présent, la valeur de la case est **EST**.
- Si seul le mur Sud est présent, la valeur de la case est **SUD**.
- Si les deux murs sont présents, la valeur de la case est **EST** + **SUD**.

Avec cette convention :

- Le mur Est de `m[i][j]` est présent si et seulement si `m[i][j] % SUD` est non nul.
- Le mur Sud est présent si et seulement si `m[i][j]` est strictement supérieur à **EST**.

Le labyrinthe maximal initial de l’algorithme s’obtiendra en initialisant les cases d’une matrice à **EST** + **SUD**. La présence d’un mur au Nord d’une case sur le bord supérieur du labyrinthe, ou à l’Ouest d’une case sur son bord gauche, est implicite – ces murs ne peuvent pas de toute manière être détruits.

Tableau des murs destructibles. Il est inutile d’inscrire dans le labyrinthe maximal initial les numéros de ses zones élémentaires : on peut simplement convenir du fait que la case de coordonnées (i, j) recevra le numéro de zone $((i * L) + j)$ où **L** est la largeur de la matrice. Avec cette convention, au lancement de l’algorithme, l’ensemble des murs destructibles peut être collecté sous la forme d’un tableau de structures, chaque structure représentant un mur et contenant les informations suivantes :

- les coordonnées (i, j) de la case dont ce mur est le mur Est ou Sud,
- une valeur égale à **EST** ou **SUD** spécifiant s’il s’agit du mur Est ou Sud de cette case,
- le plus petit numéro de zone de part et d’autre de ce mur,
- le plus grand numéro de zone de part et d’autre de ce mur.

Il ne faut évidemment pas collecter les murs Est des cases sur le bord droit du labyrinthe, ainsi que les murs Sud des cases sur son bord inférieur, tous ces murs sont non destructibles. Le tableau sera couplé à un compteur : le nombre de structures à partir de la position 0 représentant les murs encore destructibles. La valeur initiale de ce compteur est 0. *Ajouter* un élément au tableau revient à :

- écrire un élément à la position égale à la valeur courante du compteur,
- incrémenter le compteur.

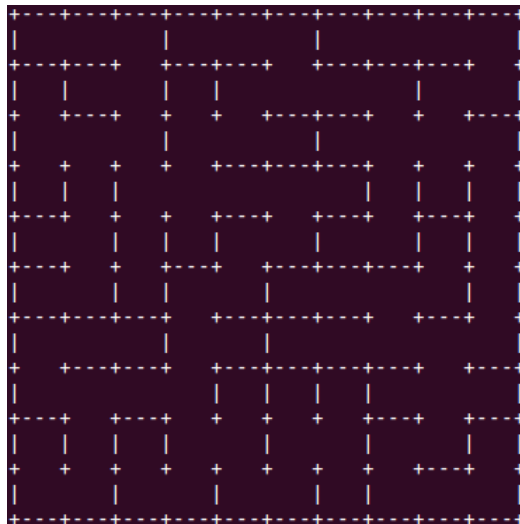
Supprimer du tableau un élément à la position **k** revient à :

- décrémenter le compteur.
- copier à la position **k** l’élément à la position égale à la valeur du compteur après décrément.

Implémentation d'une étape de l'algorithme.

1. On choisit aléatoirement dans le tableau des murs encore destructibles un mur w , qui est supprimé du tableau. Dans la matrice m du labyrinthe, à la position $(w.i, w.j)$, on soustrait de $m[i][j]$ la valeur `EST` si w est le mur Est de cette case, la valeur `SUD`, sinon – autrement dit, on abat le mur.
2. Soient z_inf , z_sup les numéros des zones que w séparait avant d'être abattu, où $z_inf < z_sup$. On parcourt le tableau en examinant tous les murs encore destructibles. Si l'un des numéros de zone d'un mur est égale à z_sup , on modifie sa valeur en z_inf . Si l'autre numéro de zone de ce mur est égal à z_inf , ce mur n'est plus destructible : il est supprimé du tableau.

Affichage du labyrinthe.



```
void afficher (int m[H][L]) {
    int i, j;
    for (j = 0; j < L; j++) {
        printf ("+---");
    }
    printf ("\n");
    for (i = 0; i < H; i++) {
        printf ("|");
        for (j = 0; j < L; j++) {
            printf ((m[i][j] % SUD) ? "  |" : "   ");
        }
        printf ("\n");
        for (j = 0; j < L; j++) {
            printf ((m[i][j] > EST) ? "+---" : "+   ");
        }
        printf ("\n");
    }
}
```