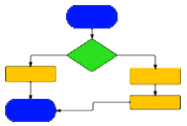
Fiche 9-7 (45 minutes)		
Les labyrinthes		

1. Introduction

Un **labyrinthe** (λαβύρινθος en grec ancien, *labyrinthus* en latin), est un tracé sinueux, muni ou non d'embranchements, d'impasses et de fausses pistes, destiné à perdre ou à ralentir celui qui cherche à s'y déplacer.

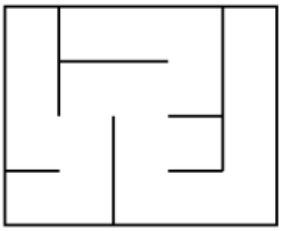
Ce motif, apparu dès la préhistoire, se retrouve dans de très nombreuses civilisations sous des formes diverses.

Le mot désigne dans la mythologie grecque une série complexe de galeries construites par Dédale pour enfermer le Minotaure. En latin, *labyrinthus* signifie « enclos de bâtiments dont il est difficile de trouver l'issue ».

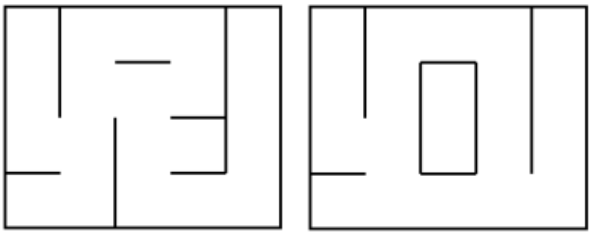
2. Construire un labyrinthe

On peut classer les labyrinthes, en deux catégories :

- celle des labyrinthes dits « parfaits » où chaque cellule est reliée à toutes les autres et, ce de manière unique ;
- celle des labyrinthes dits « imparfaits » qui sont tous les labyrinthes qui ne sont pas parfaits (ils peuvent donc contenir des boucles, des îlots ou des cellules inaccessibles).



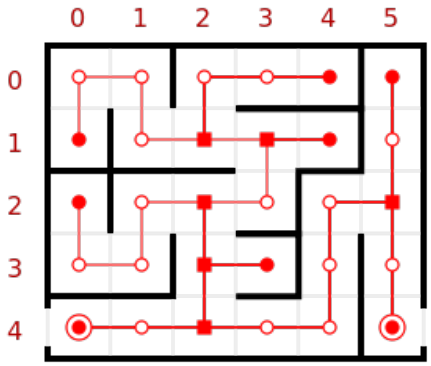
Labyrinthe « parfait »



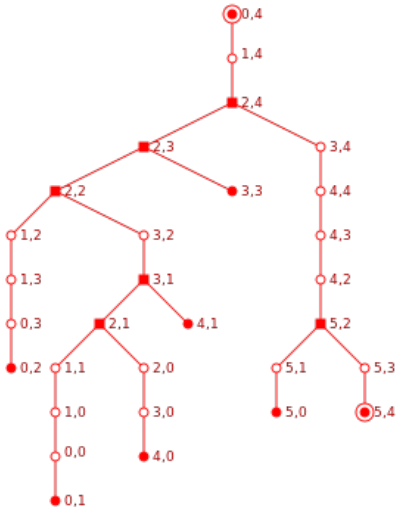
Labyrinthes « imparfaits »

2.1. Notions de graphes dans un labyrinthe

Les chemins des labyrinthes parfaits peuvent être représentés par un graphe orienté sans cycle (que l'on appelle aussi un arbre orienté) :

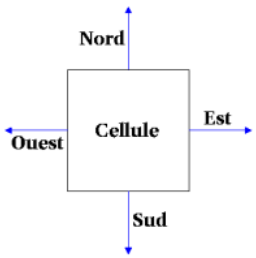


Les nœuds sont marqués selon qu'il s'agit d'entrées ou de sorties, d'intersections ou de cul-de-sac.

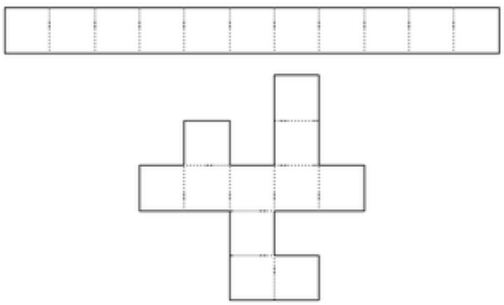


Représentation en arbre du labyrinthe

Les algorithmes proposés ici vont s'intéresser aux labyrinthes parfaits, mais quelques adaptations permettent de créer facilement des labyrinthes à îlots. Ils sont basés sur l'espace discrétisé dont les cellules carrées sont initialement remplies et séparées par des cloisons, selon les quatre directions (nord, sud, est et ouest).



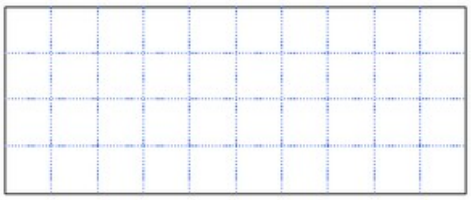
Un labyrinthe « rectangulaire parfait » de m colonnes par n lignes est un ensemble de $m \cdot n$ cellules reliées les unes aux autres par un chemin unique.



Ces deux surfaces sont équivalentes : chacune a 11 cellules et 10 murs internes (marqués en pointillés).

Le nombre de murs ouverts pour permettre un chemin unique dans un labyrinthe de $m \cdot n$ cellules est identique au nombre de murs ouverts pour un chemin droit de $m \cdot n$ cellules, soit $m \cdot n - 1$ murs.

Dans un rectangle m x n cellules, le nombre total de murs internes possible est $2m{\cdot}n{-}m{-}n$.. Pour obtenir ce résultat, on compte deux murs par cellule, par exemple celui du bas et celui de droite (on évite ainsi de recompter deux fois les mêmes) et on retire le nombre de murs limitant le rectangle en bas (m) et à droite (n).



Murs internes

Le nombre de murs fermés dans un labyrinthe parfait est donc $2m{\cdot}n{-}m{-}n{-}(m{\cdot}n{-}1) = (m{-}1)(n{-}1)$

2.2. Algorithmes de construction de labyrinthes

Il existe de nombreux algorithmes de construction de labyrinthes. Voici deux d'entre eux assez simples.

2.2.1. Fusion aléatoire de chemins

Cet algorithme utilise une propriété des labyrinthes parfaits précédemment énoncée telle quelle : « *Chaque cellule est reliée à toutes les autres, et ce, de manière unique* ».

Il fonctionne en fusionnant progressivement des chemins depuis la simple cellule jusqu'à l'obtention d'un chemin unique.

L'algorithme associe une valeur unique à chaque cellule (leur numéro de séquence, par exemple) et part d'un labyrinthe où tous les murs sont fermés.

À chaque itération, on choisit un mur à ouvrir de manière aléatoire.

Lorsqu'un mur est ouvert entre deux cellules adjacentes, les deux cellules sont liées entre elles et forment un chemin.

À chaque fois que l'on tente d'ouvrir un mur entre deux cellules, on vérifie que ces deux cellules ont des identifiants différents.

- Si les identifiants sont identiques, c'est que les deux cellules sont déjà reliées et appartiennent donc au même chemin. On ne peut donc pas ouvrir le mur.

- Si les identifiants sont différents, le mur est ouvert, et l'identifiant de la première cellule est affecté à toutes les cellules du second chemin.

Finalement, on obtient un chemin unique lorsque le nombre de murs ouverts est atteint., ce qui donne les étapes de l'exemple ci-après.

2.2.2. Exploration exhaustive

On part d'un labyrinthe où tous les murs sont fermés. Chaque cellule contient une variable booléenne qui indique si la cellule a déjà été visitée ou non (i.e. les cellules visitées sont celles qui appartiennent au chemin du labyrinthe en cours de construction).

Au départ, toutes les cellules sont marquées comme non visitées.

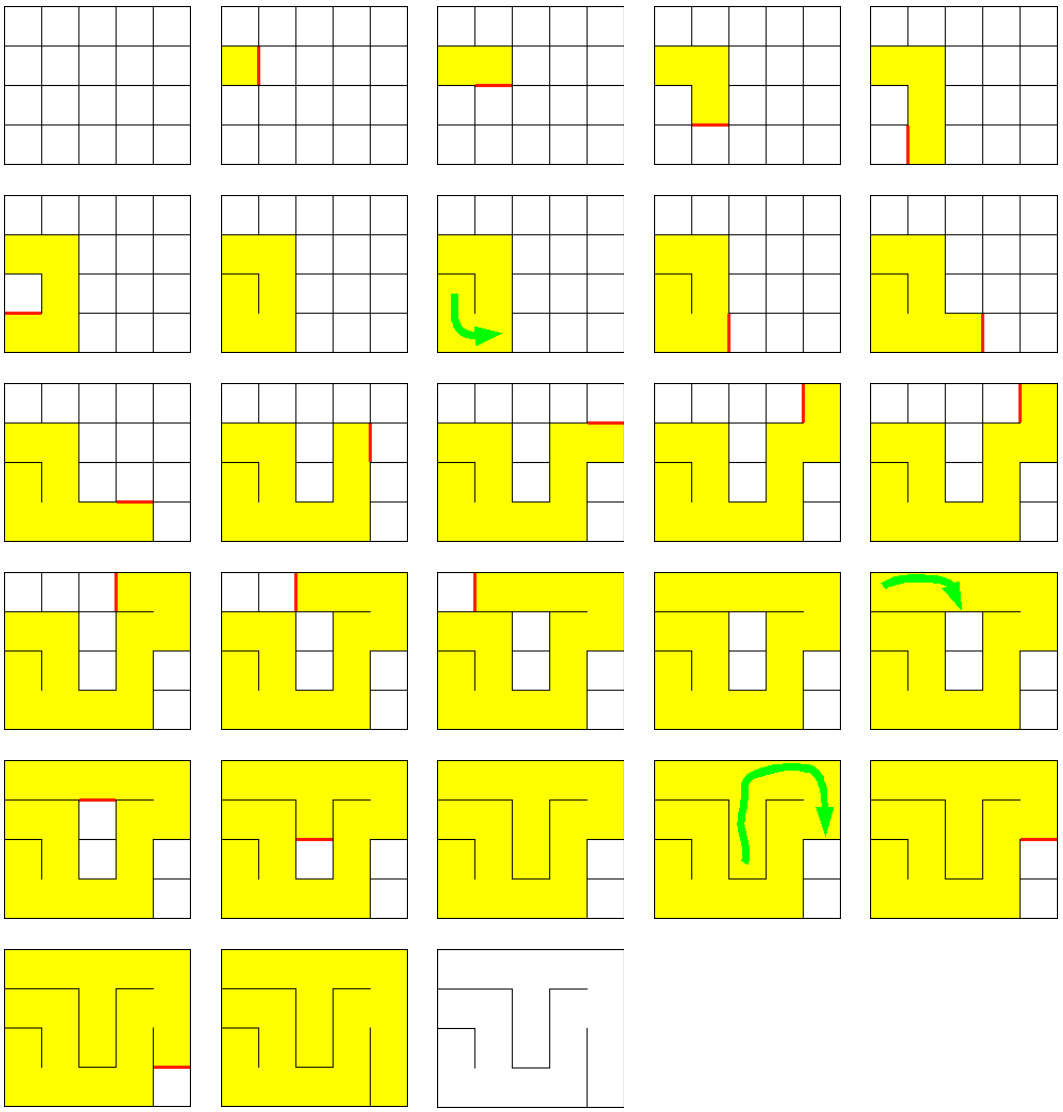
On choisit arbitrairement une cellule, on stocke la position en cours et on la marque comme visitée.

Puis on regarde quelles sont les cellules voisines possibles et non visitées.

S'il y a au moins une possibilité, on en choisit une au hasard, on ouvre le mur et on recommence avec la nouvelle cellule.

S'il n'y en pas, on revient à la case précédente et on recommence.

Lorsque l'on est revenu à la case de départ et qu'il n'y a plus de possibilités, le labyrinthe est terminé.



Le texte et les images de ce paragraphe proviennent de [3].

3. Sortir d'un labyrinthe

Vous êtes prisonnier d'un labyrinthe et vous voulez en sortir. Comment faire ? Vous n'avez évidemment aucun plan du labyrinthe. Voici quelques algorithmes, plus ou moins efficaces.

3.1. La méthode de la souris

L'algorithme de la souris est l'algorithme le plus simple pour se sortir d'un labyrinthe : il consiste à prendre à chaque intersection un chemin au hasard. Les théorèmes sur les marches aléatoires sont formels : en procédant de cette façon, vous finirez par sortir du labyrinthe... Cependant, le temps que vous mettrez pour trouver la sortie sera vraiment très long (proportionnel au moins au carré de la taille du labyrinthe).

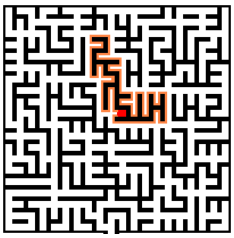
3.2. La méthode de la main sur le mur

Cet algorithme consiste à se déplacer en gardant la main contre un des murs, disons le droit. Ainsi, à chaque intersection, on tournera à droite. En notant les endroits où l'on est passé qu'une seule fois, on peut en déduire un chemin (pas forcément le plus rapide).

L'algorithme du mur en action : le point rouge est le point de départ, le chemin en orange est le chemin obtenu en suivant le mur droit. On peut en déduire le chemin le plus rapide, en bleu.

Cette méthode permet de sortir à coup sûr du labyrinthe en un temps raisonnable, mais on prend le risque de visiter l'ensemble du labyrinthe.

La méthode du mur pose un problème quand le labyrinthe contient des îlots.



L'algorithme du mur en action dans un labyrinthe avec îlots : en suivant toujours le mur (gauche ou droit), on finit par tourner en rond.

3.3. L'algorithme de Pledge

Le principe de base, c'est de longer les murs, mais en évitant de rester coincé sur un même îlot. Pour ça, il faut trouver le bon moment où lâcher son mur. Dans la pratique, il faut garder un compteur dans la tête, que l'on initialise à 0 : si le compteur indique 0, on va tout droit jusqu'au mur en face. À partir de ce mur, on tourne du côté que l'on préfère (mais toujours le même, disons gauche) et on suit le mur en ajoutant 1 au compteur dès que l'on tourne à droite et en soustrayant 1 dès que l'on tourne à gauche. Si le compteur indique 0, on lâche le mur, et on va tout droit.

L'algorithme de Pledge en action : on part du point rouge (compteur en position 0) et on file tout droit. On suit alors le mur par la gauche. Dès que l'on tourne à gauche (angle bleu), on incrémente le compteur), et si on tourne à droite (angle vert foncé) on le décrémente. Si le compteur atteint à nouveau 0 (angle vert clair), on lâche le mur, et on continue tout droit.

Évidemment, cet algorithme ne marche que dans le cas où le labyrinthe est orthogonal (tous les angles sont à 90°), mais on peut adapter l'algorithme aux autres labyrinthes. Au lieu d'incrémenter le compteur de 1 dès que l'on tourne, on l'incrémente par l'angle du virage (avec son signe + ou -).

Cet algorithme présente un problème si la sortie est une trappe située sur un îlot.

L'algorithme de Pledge en action sur un labyrinthe où la sortie (en jaune) se trouve sur un îlot : au bout d'un moment, on tourne en rond...

3.4. La méthode de Trémaux

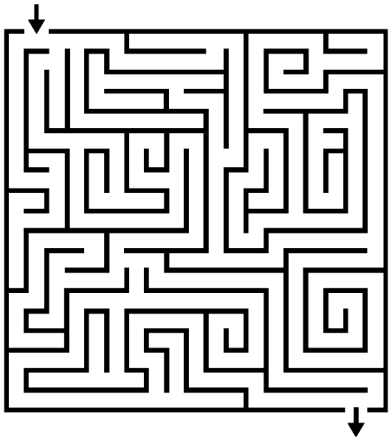
Marquez à la craie le chemin que vous suivez. Si vous tombez dans un cul-de-sac, faites simplement demi-tour. Si vous tombez sur une intersection que vous n'aviez encore jamais croisée, prenez le chemin que vous préférez. Si vous êtes déjà tombé sur cette intersection auparavant, faites comme si vous étiez tombé sur un cul-de-sac en faisant demi-tour, pour rejoindre la dernière intersection où vous avez fait un choix.

Si vous revenez à un moment où à un autre sur votre point de départ, c'est que vous avez visité la totalité du labyrinthe et qu'il n'avait aucune sortie... Par contre, si vous trouvez la sortie, rien ne vous dit que c'était le chemin le plus court !

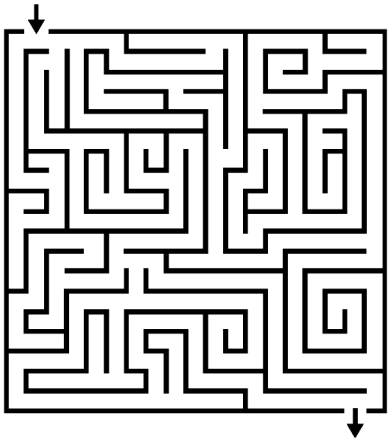
L'algorithme de Trémaux en action : au fur et à mesure, on peut marquer certains passages comme étant des culs-de-sac, ici en gris.

Exercice 2

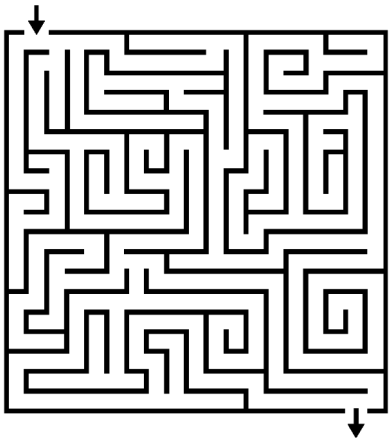
Utilisez les quatre méthodes ci-dessus pour sortir de ce labyrinthe, reproduit quatre fois à l'identique. Pour la méthode de la souris, utilisez un dé.



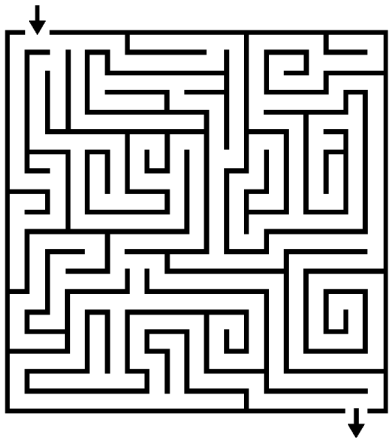
Souris



Main sur le mur



Pledge



Trémeaux

Sources

[1] Wikipédia : Labyrinthe, <https://fr.wikipedia.org/wiki/Labyrinthe>

[2] Wikipédia : Modélisation mathématique d'un labyrinthe, <https://fr.wikipedia.org/wiki/Modélisation_mathématique_d'un_labyrinthe>

[3] Le labyrinthe dont vous êtes le héros, <http://eljjdx.canalblog.com/archives/2011/02/20/20431821.html>