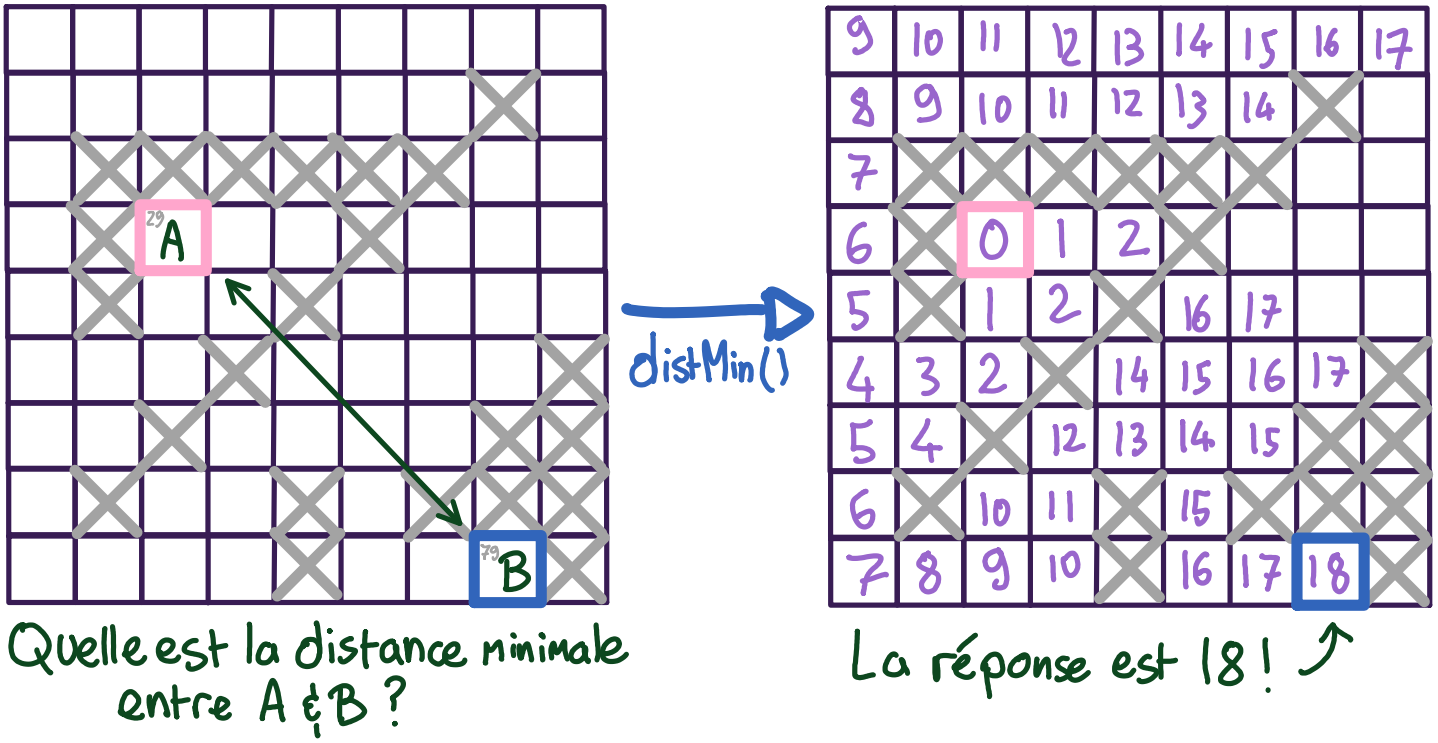
Algorithme de recherche de la distance minimale

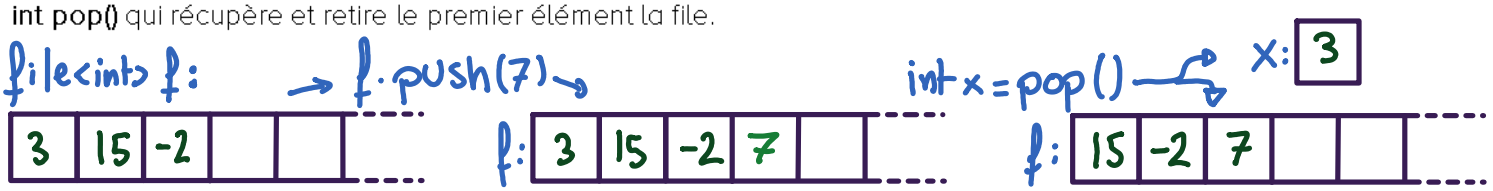
L'algorithme de recherche de la distance minimale (permettant de trouver la plus petite distance entre une cellule de départ et une cellule d'arrivée) s'effectue de manière itérative.
On associe à la cellule de départ la valeur 0. Ensuite, la valeur 1 à toutes les cellules vides voisines. Ensuite la valeur 2 à toutes les cellules vides voisines à celles ci, etc... Jusqu'à atteindre la cellule d'arrivée. La valeur associée à cette cellule est alors la distance minimale recherchée.

Exemple de fonctionnement :



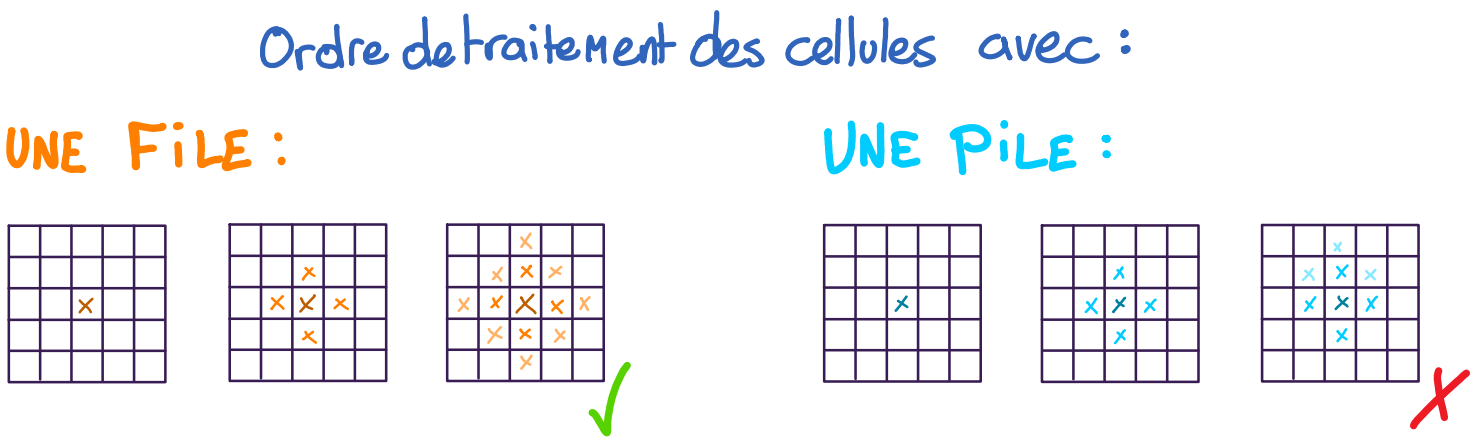
Structure de donnée utilisée

On utilisera une structure de donnée particulière : La file. Il s'agit d'une structure "FIFO", *last-in last-out*, ce qui signifie premier entré, premier sorti. La file est munie de deux méthodes : **push(int x)** qui ajoute un entier en dernière position, **int pop()** qui récupère et retire le premier élément la file.



Pourquoi utiliser une file et pas une autre structure, comme une pile? A chaque étape, on s'éloigne de plus en plus de la case de départ.
La file permet d'évoluer couche par couche autour de la case de départ car on traite les cellules dans leur ordre d'arrivée, on explore donc tous les chemins en même temps. Avec une pile, on traiterait d'abord un chemin, puis ensuite l'autre (ce qu'on ne souhaite pas).

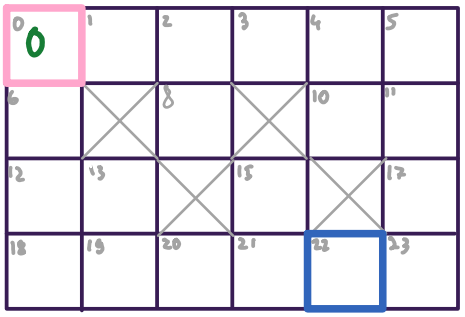
Ordre de traitement des cellules avec :



Traitement étape par étape

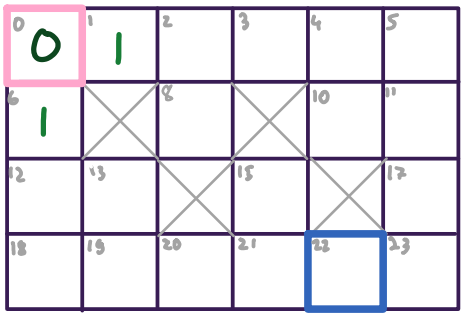
Pour commencer, on marque la case de départ d'un 0 et on push son id dans la file.
Ensuite, tant qu'on a pas marqué la case finale, on pop le premier élément de la file dans la variable *current*, marque toutes ses cases voisines qui ne sont pas un mur ou déjà marquées de sa valeur + 1 (dans l'ordre haut, droite, gauche puis bas), et on push leur id dans la file (dans le même ordre) -> Ordre arbitraire.
Si la case finale est marquée, on peut renvoyer la valeur qu'elle contient : Il s'agit de sa distance à la case de départ.

ETAPE 0



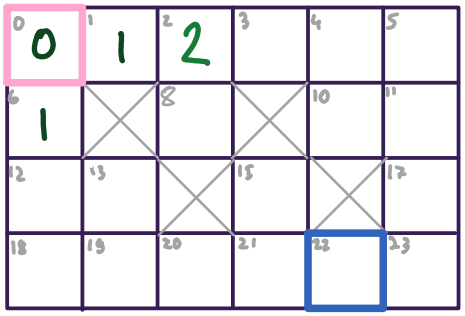
FILE: [0] CURRENT: ☐

1 :



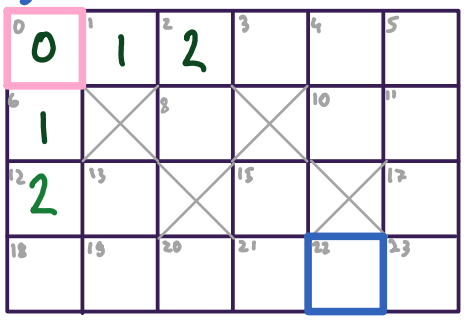
FILE: [1|6] CURRENT: 0

2 :



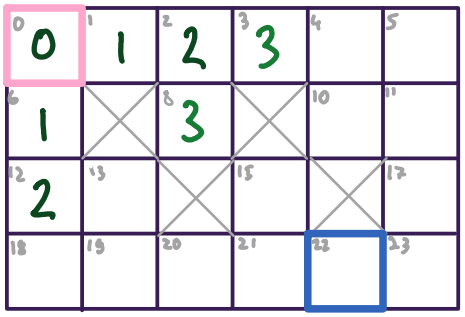
FILE: [6|2] CURRENT: 1

3 :



FILE: [21|2] CURRENT: 6

4 :



FILE: [12|3|8] CURRENT: 2

→ • • •

ETAPE N:

FILE: [17|15|22] CURRENT: 2

LA DERNIÈRE CASE EST MARQUÉE D'UN 7. C'EST LE RÉSULTAT !

• • • →

Pseudocode

```
1 Soit A la cellule de départ, B d'arrivée
2 valeur de A ← 0
3 pile.push(A)
4 tant que B n'est pas marquée
5 |   current = pile.pop()
6 |   voisins(current) ← valeur de current +1
7 fin tant que
8 retourner la valeur de B
```