TP INTELLIGENCE ARTIFICIELLE

MEMBRES

NESRINE KHIARI CIN : 15021295

HOUCEM HBIRI CIN : 11170377

PRÉSENTATION GÉNÉRALE DU PROJET

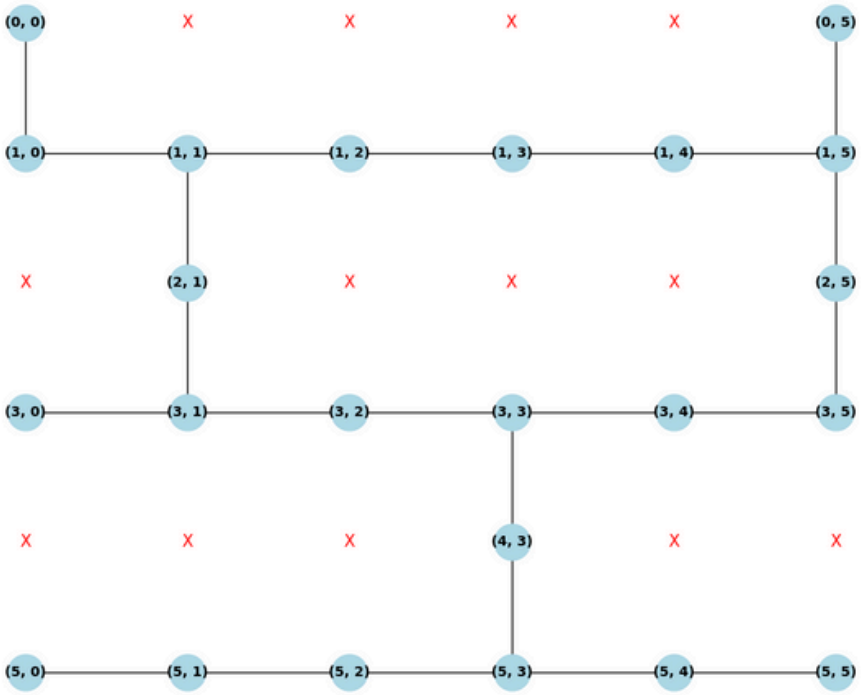
Ce projet vise à analyser et à comparer trois approches classiques de parcours dans un graphe d'états : la recherche en largeur, la recherche en profondeur et l'algorithme A*. Ces méthodes sont largement utilisées pour résoudre des problèmes complexes en informatique, tels que les jeux, l'optimisation ou encore la navigation. Le but principal est d'évaluer leur performance en termes de durée d'exécution et de nombre de nœuds visités, afin de mettre en lumière leurs points forts et leurs faiblesses.

DESCRIPTION DE LA PROBLÉMATIQUE

La problématique consiste à résoudre un labyrinthe représenté par une matrice en utilisant trois algorithmes de recherche : BFS (Breadth-First Search), DFS (Depth-First Search) et A (A-star)*. Le labyrinthe est défini comme une grille où les cases accessibles sont représentées par 0 et les obstacles par 1. L'objectif est de trouver le chemin le plus court (ou un chemin) du point de départ (0, 0) au point d'arrivée (5, 5) tout en explorant les nœuds de manière efficace.

PRÉSENTATION GÉNÉRALE DU PROJET

```python
# Labyrinthe représenté comme une matrice (0 = accessible, 1 = obstacle)
maze = [
    [0, 1, 1, 1, 1, 0],
    [0, 0, 0, 0, 0, 0],
    [1, 0, 1, 1, 1, 0],
    [0, 0, 0, 0, 0, 0],
    [1, 1, 1, 0, 1, 1],
    [0, 0, 0, 0, 0, 0],
]

start = (0, 0)  # Point de départ
end = (5, 5)  # Point d'arrivée
```



REMARQUES:

- En plus de trouver le chemin, le programme : Génère un graphe des connexions explorées à l'aide de NetworkX et Matplotlib.
- Dans les graphes à venir , le chemin solution est représenté en rouge, les autres nœuds représentent tous les nœuds générés.

ALGORITHME DE CHAQUE MÉTHODE DE RECHERCHE

1. BFS (Breadth-First Search)

La recherche en largeur explore un graphe en visitant les nœuds niveau par niveau. Elle examine d'abord tous les voisins d'un nœud avant de passer à ceux du niveau suivant.

Caractéristiques :

- Utilise une structure de file (queue) pour gérer les nœuds à explorer.
- Assure la découverte du chemin le plus court dans un graphe non pondéré.

Avantages :

- Idéal pour trouver des solutions optimales dans des graphes où les arêtes ont des coûts égaux.

Limites :

- Nécessite une quantité importante de mémoire pour stocker les nœuds à chaque niveau, ce qui peut poser problème pour des graphes de grande taille.

Étapes :

1. Initialiser une file pour suivre les nœuds à explorer.
2. Marquer le nœud de départ comme visité.
3. Pour chaque nœud, explorer ses voisins non visités et les ajouter à la file.
4. Si le nœud de destination est atteint, retourner le chemin.
5. Si la file est vide sans atteindre la destination, aucun chemin n'existe.

CAPTURES D'ECRAN D'EXECUTION

BFS

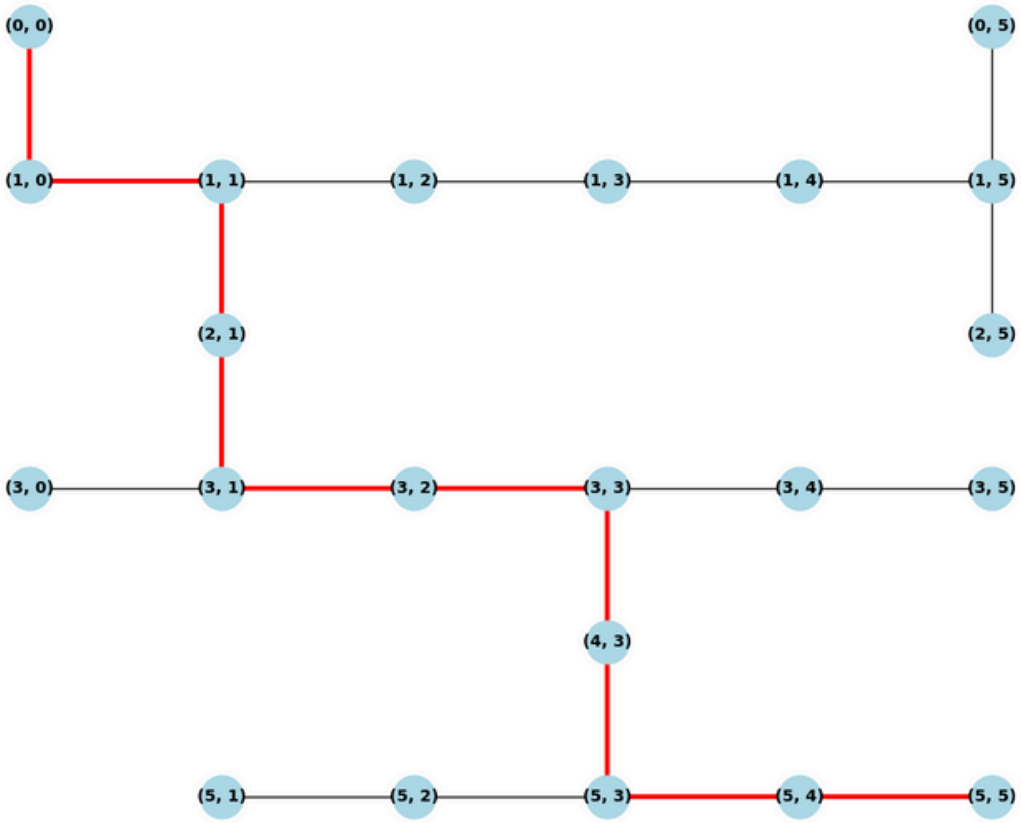
```
BFS :
_____________________________________________________________________________
((0, 0), [(0, 0)])
_____________________________________________________________________________
État courant: (0, 0)
Blocage pour la direction: (-1, 0)
Ajout du nouvel état: (1, 0)
Blocage pour la direction: (0, -1)
Blocage pour la direction: (0, 1)
_____________________________________________________________________________
((1, 0), [(0, 0), (1, 0)])
_____________________________________________________________________________
État courant: (1, 0)
Blocage pour la direction: (-1, 0)
Blocage pour la direction: (1, 0)
Blocage pour la direction: (0, -1)
Ajout du nouvel état: (1, 1)
_____________________________________________________________________________
((1, 1), [(0, 0), (1, 0), (1, 1)])
_____________________________________________________________________________
État courant: (1, 1)
Blocage pour la direction: (-1, 0)
Ajout du nouvel état: (2, 1)
Blocage pour la direction: (0, -1)
Ajout du nouvel état: (1, 2)
_____________________________________________________________________________
((2, 1), [(0, 0), (1, 0), (1, 1), (2, 1)])
((1, 2), [(0, 0), (1, 0), (1, 1), (1, 2)])
_____________________________________________________________________________
État courant: (2, 1)
Blocage pour la direction: (-1, 0)
Ajout du nouvel état: (3, 1)
Blocage pour la direction: (0, -1)
Blocage pour la direction: (0, 1)
_____________________________________________________________________________
((1, 2), [(0, 0), (1, 0), (1, 1), (1, 2)])
((3, 1), [(0, 0), (1, 0), (1, 1), (2, 1), (3, 1)])
_____________________________________________________________________________
État courant: (1, 2)
Blocage pour la direction: (-1, 0)
Blocage pour la direction: (1, 0)
Blocage pour la direction: (0, -1)
Ajout du nouvel état: (1, 3)
_____________________________________________________________________________
((3, 1), [(0, 0), (1, 0), (1, 1), (2, 1), (3, 1)])
((1, 3), [(0, 0), (1, 0), (1, 1), (1, 2), (1, 3)])
_____________________________________________________________________________
État courant: (3, 1)
Blocage pour la direction: (-1, 0)
Blocage pour la direction: (1, 0)
Ajout du nouvel état: (3, 0)
Ajout du nouvel état: (3, 2)
_____________________________________________________________________________
((1, 3), [(0, 0), (1, 0), (1, 1), (1, 2), (1, 3)])
((3, 0), [(0, 0), (1, 0), (1, 1), (2, 1), (3, 1), (3, 0)])
((3, 2), [(0, 0), (1, 0), (1, 1), (2, 1), (3, 1), (3, 2)])
_____________________________________________________________________________
État courant: (1, 3)
Blocage pour la direction: (-1, 0)
Blocage pour la direction: (1, 0)
Blocage pour la direction: (0, -1)
Ajout du nouvel état: (1, 4)
_____________________________________________________________________________
((3, 0), [(0, 0), (1, 0), (1, 1), (2, 1), (3, 1), (3, 0)])
((3, 2), [(0, 0), (1, 0), (1, 1), (2, 1), (3, 1), (3, 2)])
((1, 4), [(0, 0), (1, 0), (1, 1), (1, 2), (1, 3), (1, 4)])
_____________________________________________________________________________
État courant: (3, 0)
Blocage pour la direction: (-1, 0)
Blocage pour la direction: (1, 0)
Blocage pour la direction: (0, -1)
Blocage pour la direction: (0, 1)
_____________________________________________________________________________
((3, 2), [(0, 0), (1, 0), (1, 1), (2, 1), (3, 1), (3, 2)])
((1, 4), [(0, 0), (1, 0), (1, 1), (1, 2), (1, 3), (1, 4)])
_____________________________________________________________________________
État courant: (3, 2)
Blocage pour la direction: (-1, 0)
Blocage pour la direction: (1, 0)
Blocage pour la direction: (0, -1)
Ajout du nouvel état: (3, 3)
_____________________________________________________________________________
((1, 4), [(0, 0), (1, 0), (1, 1), (1, 2), (1, 3), (1, 4)])
((3, 3), [(0, 0), (1, 0), (1, 1), (2, 1), (3, 1), (3, 2), (3, 3)])
```

```
_____________________________________________________________________________
État courant: (1, 4)
Blocage pour la direction: (-1, 0)
Blocage pour la direction: (1, 0)
Blocage pour la direction: (0, -1)
Ajout du nouvel état: (1, 5)

_____________________________________________________________________________
((3, 3), [(0, 0), (1, 0), (1, 1), (2, 1), (3, 1), (3, 2), (3, 3)])
((1, 5), [(0, 0), (1, 0), (1, 1), (1, 2), (1, 3), (1, 4), (1, 5)])

_____________________________________________________________________________
État courant: (3, 3)
Blocage pour la direction: (-1, 0)
Ajout du nouvel état: (4, 3)
Blocage pour la direction: (0, -1)
Ajout du nouvel état: (3, 4)

_____________________________________________________________________________
((1, 5), [(0, 0), (1, 0), (1, 1), (1, 2), (1, 3), (1, 4), (1, 5)])
((4, 3), [(0, 0), (1, 0), (1, 1), (2, 1), (3, 1), (3, 2), (3, 3), (4, 3)])
((3, 4), [(0, 0), (1, 0), (1, 1), (2, 1), (3, 1), (3, 2), (3, 3), (3, 4)])

_____________________________________________________________________________
État courant: (1, 5)
Ajout du nouvel état: (0, 5)
Ajout du nouvel état: (2, 5)
Blocage pour la direction: (0, -1)
Blocage pour la direction: (0, 1)

_____________________________________________________________________________
((4, 3), [(0, 0), (1, 0), (1, 1), (2, 1), (3, 1), (3, 2), (3, 3), (4, 3)])
((3, 4), [(0, 0), (1, 0), (1, 1), (2, 1), (3, 1), (3, 2), (3, 3), (3, 4)])
((0, 5), [(0, 0), (1, 0), (1, 1), (1, 2), (1, 3), (1, 4), (1, 5), (0, 5)])
((2, 5), [(0, 0), (1, 0), (1, 1), (1, 2), (1, 3), (1, 4), (1, 5), (2, 5)])

_____________________________________________________________________________
État courant: (4, 3)
Blocage pour la direction: (-1, 0)
Ajout du nouvel état: (5, 3)
Blocage pour la direction: (0, -1)
Blocage pour la direction: (0, 1)

_____________________________________________________________________________
((3, 4), [(0, 0), (1, 0), (1, 1), (2, 1), (3, 1), (3, 2), (3, 3), (3, 4)])
((0, 5), [(0, 0), (1, 0), (1, 1), (1, 2), (1, 3), (1, 4), (1, 5), (0, 5)])
((2, 5), [(0, 0), (1, 0), (1, 1), (1, 2), (1, 3), (1, 4), (1, 5), (2, 5)])
((5, 3), [(0, 0), (1, 0), (1, 1), (2, 1), (3, 1), (3, 2), (3, 3), (4, 3), (5, 3)])
_____________________________________________________________________________
État courant: (3, 4)
Blocage pour la direction: (-1, 0)
Blocage pour la direction: (1, 0)
Blocage pour la direction: (0, -1)
Ajout du nouvel état: (3, 5)

_____________________________________________________________________________
((0, 5), [(0, 0), (1, 0), (1, 1), (1, 2), (1, 3), (1, 4), (1, 5), (0, 5)])
((2, 5), [(0, 0), (1, 0), (1, 1), (1, 2), (1, 3), (1, 4), (1, 5), (2, 5)])
((5, 3), [(0, 0), (1, 0), (1, 1), (2, 1), (3, 1), (3, 2), (3, 3), (4, 3), (5, 3)])
((3, 5), [(0, 0), (1, 0), (1, 1), (2, 1), (3, 1), (3, 2), (3, 3), (3, 4), (3, 5)])

_____________________________________________________________________________
État courant: (0, 5)
Blocage pour la direction: (-1, 0)
Blocage pour la direction: (1, 0)
Blocage pour la direction: (0, -1)
Blocage pour la direction: (0, 1)

_____________________________________________________________________________
((2, 5), [(0, 0), (1, 0), (1, 1), (1, 2), (1, 3), (1, 4), (1, 5), (2, 5)])
((5, 3), [(0, 0), (1, 0), (1, 1), (2, 1), (3, 1), (3, 2), (3, 3), (4, 3), (5, 3)])
((3, 5), [(0, 0), (1, 0), (1, 1), (2, 1), (3, 1), (3, 2), (3, 3), (3, 4), (3, 5)])
_____________________________________________________________________________
État courant: (2, 5)
Blocage pour la direction: (-1, 0)
Blocage pour la direction: (1, 0)
Blocage pour la direction: (0, -1)
Blocage pour la direction: (0, 1)

_____________________________________________________________________________
((5, 3), [(0, 0), (1, 0), (1, 1), (2, 1), (3, 1), (3, 2), (3, 3), (4, 3), (5, 3)])
((3, 5), [(0, 0), (1, 0), (1, 1), (2, 1), (3, 1), (3, 2), (3, 3), (3, 4), (3, 5)])
_____________________________________________________________________________
État courant: (5, 3)
Blocage pour la direction: (-1, 0)
Blocage pour la direction: (1, 0)
Ajout du nouvel état: (5, 2)
Ajout du nouvel état: (5, 4)

_____________________________________________________________________________
((3, 5), [(0, 0), (1, 0), (1, 1), (2, 1), (3, 1), (3, 2), (3, 3), (3, 4), (3, 5)])
((5, 2), [(0, 0), (1, 0), (1, 1), (2, 1), (3, 1), (3, 2), (3, 3), (4, 3), (5, 3), (5, 2)])
((5, 4), [(0, 0), (1, 0), (1, 1), (2, 1), (3, 1), (3, 2), (3, 3), (4, 3), (5, 3), (5, 4)])
_____________________________________________________________________________
État courant: (3, 5)
Blocage pour la direction: (-1, 0)
Blocage pour la direction: (1, 0)
Blocage pour la direction: (0, -1)
Blocage pour la direction: (0, 1)

_____________________________________________________________________________
((5, 2), [(0, 0), (1, 0), (1, 1), (2, 1), (3, 1), (3, 2), (3, 3), (4, 3), (5, 3), (5, 2)])
((5, 4), [(0, 0), (1, 0), (1, 1), (2, 1), (3, 1), (3, 2), (3, 3), (4, 3), (5, 3), (5, 4)])
```

```
--------------------------------------------------------------------------------
État courant: (5, 2)
Blocage pour la direction: (-1, 0)
Blocage pour la direction: (1, 0)
Ajout du nouvel état: (5, 1)
Blocage pour la direction: (0, 1)
--------------------------------------------------------------------------------
((5, 4), [(0, 0), (1, 0), (1, 1), (2, 1), (3, 1), (3, 2), (3, 3), (4, 3), (5, 3), (5, 4)])
((5, 1), [(0, 0), (1, 0), (1, 1), (2, 1), (3, 1), (3, 2), (3, 3), (4, 3), (5, 3), (5, 2), (5, 1)])
--------------------------------------------------------------------------------
État courant: (5, 4)
Blocage pour la direction: (-1, 0)
Blocage pour la direction: (1, 0)
Blocage pour la direction: (0, -1)
Ajout du nouvel état: (5, 5)
```

RÉSULTAT FINAL BFS

```
Chemin trouvé : [(0, 0), (1, 0), (1, 1), (2, 1), (3, 1), (3, 2), (3, 3), (4, 3), (5, 3), (5, 4), (5, 5)]
Nombre de nœuds explorés : 20
Temps d'exécution : 0.002512 secondes
```



2. DFS (Depth-First Search)

La recherche en profondeur suit un chemin dans le graphe jusqu'à ce qu'elle atteigne un nœud terminal ou une impasse, avant de revenir en arrière pour explorer d'autres chemins possibles.

Caractéristiques :

- Repose sur une pile (stack), souvent implémentée via la récursivité.
- Ne garantit pas toujours la découverte du chemin le plus court.

Avantages :

- Consomme moins de mémoire, surtout pour des graphes peu profonds.

Limites :

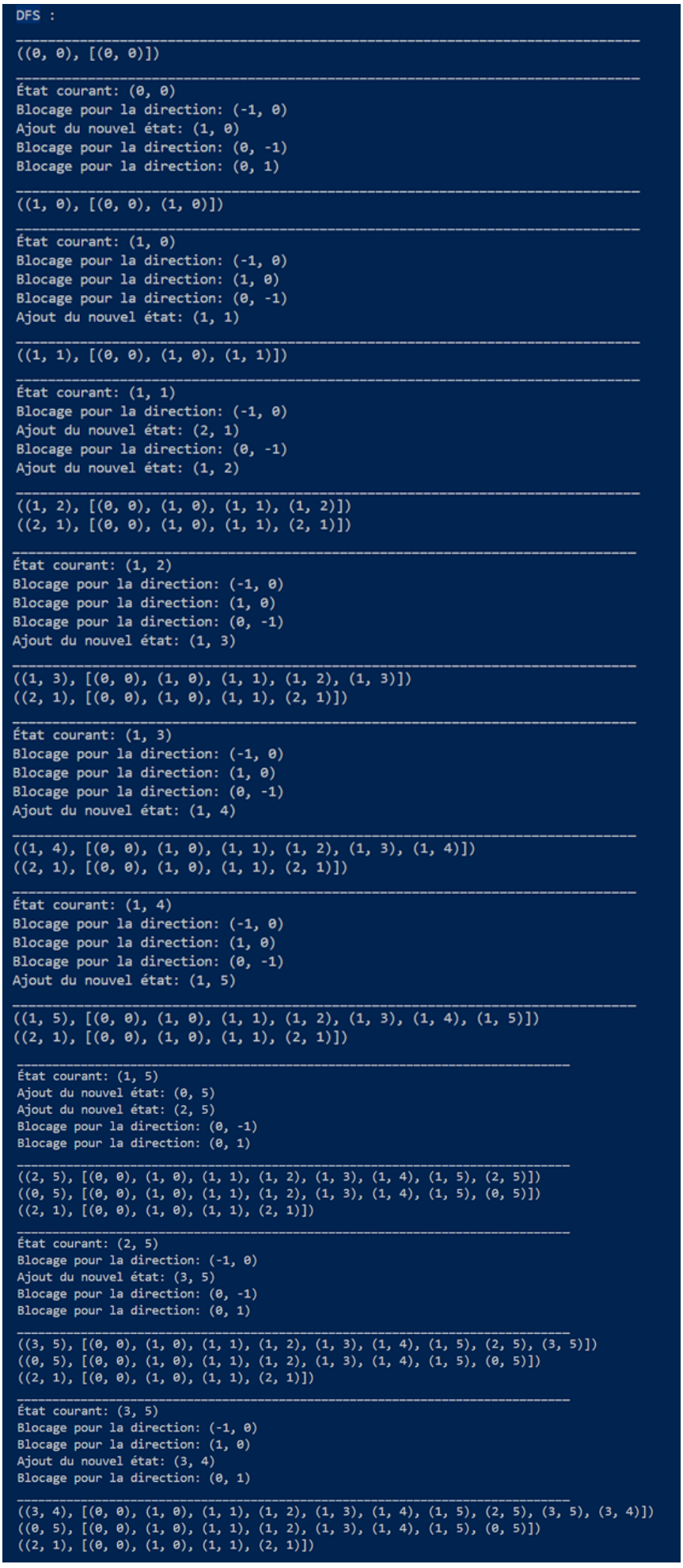
- Peu efficace dans des graphes très profonds ou contenant de nombreux chemins inutiles.

Étapes :

1. Initialiser une pile pour suivre les nœuds à explorer.
2. Marquer le nœud de départ comme visité.
3. Pour chaque nœud, explorer ses voisins non visités et l'ajouter à la pile.
4. Si la destination est atteinte, retourner le chemin.
5. Si la pile devient vide sans atteindre la destination, aucun chemin n'existe.

CAPTURES D'ECRAN D'EXECUTION

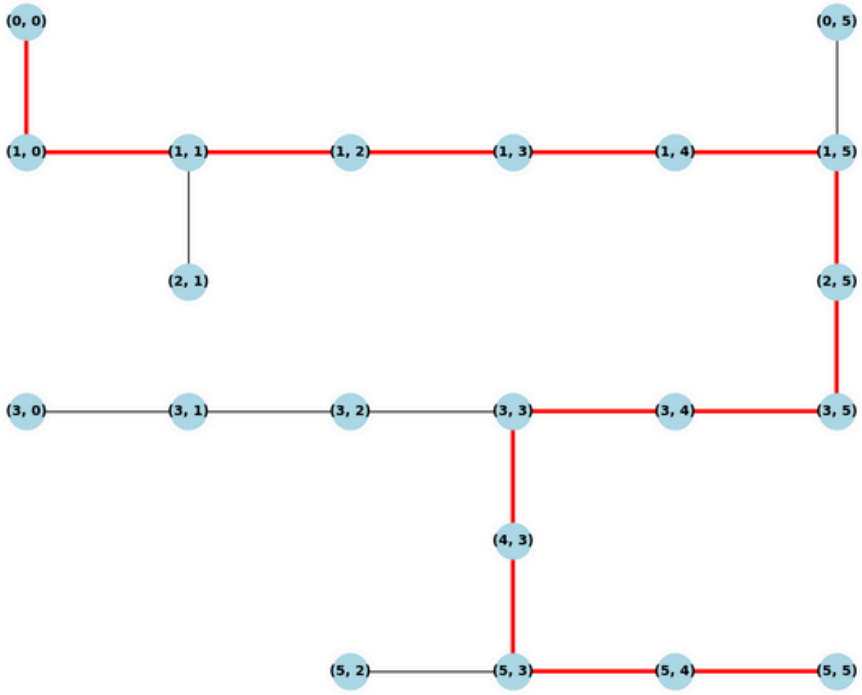
DFS



```
_____________________________________________________________________________
État courant: (3, 4)
Blocage pour la direction: (-1, 0)
Blocage pour la direction: (1, 0)
Ajout du nouvel état: (3, 3)
Blocage pour la direction: (0, 1)
_____________________________________________________________________________
((3, 3), [(0, 0), (1, 0), (1, 1), (1, 2), (1, 3), (1, 4), (1, 5), (2, 5), (3, 5), (3, 4), (3, 3)])
((0, 5), [(0, 0), (1, 0), (1, 1), (1, 2), (1, 3), (1, 4), (1, 5), (0, 5)])
((2, 1), [(0, 0), (1, 0), (1, 1), (2, 1)])
_____________________________________________________________________________
État courant: (3, 3)
Blocage pour la direction: (-1, 0)
Ajout du nouvel état: (4, 3)
Ajout du nouvel état: (3, 2)
Blocage pour la direction: (0, 1)
_____________________________________________________________________________
((3, 2), [(0, 0), (1, 0), (1, 1), (1, 2), (1, 3), (1, 4), (1, 5), (2, 5), (3, 5), (3, 4), (3, 3), (3, 2)])
((4, 3), [(0, 0), (1, 0), (1, 1), (1, 2), (1, 3), (1, 4), (1, 5), (2, 5), (3, 5), (3, 4), (3, 3), (4, 3)])
((0, 5), [(0, 0), (1, 0), (1, 1), (1, 2), (1, 3), (1, 4), (1, 5), (0, 5)])
((2, 1), [(0, 0), (1, 0), (1, 1), (2, 1)])
_____________________________________________________________________________
État courant: (3, 2)
Blocage pour la direction: (-1, 0)
Blocage pour la direction: (1, 0)
Ajout du nouvel état: (3, 1)
Blocage pour la direction: (0, 1)
_____________________________________________________________________________
((3, 1), [(0, 0), (1, 0), (1, 1), (1, 2), (1, 3), (1, 4), (1, 5), (2, 5), (3, 5), (3, 4), (3, 3), (3, 2), (3, 1)])
((4, 3), [(0, 0), (1, 0), (1, 1), (1, 2), (1, 3), (1, 4), (1, 5), (2, 5), (3, 5), (3, 4), (3, 3), (4, 3)])
((0, 5), [(0, 0), (1, 0), (1, 1), (1, 2), (1, 3), (1, 4), (1, 5), (0, 5)])
((2, 1), [(0, 0), (1, 0), (1, 1), (2, 1)])
_____________________________________________________________________________
État courant: (3, 1)
Blocage pour la direction: (-1, 0)
Blocage pour la direction: (1, 0)
Ajout du nouvel état: (3, 0)
Blocage pour la direction: (0, 1)
_____________________________________________________________________________
((3, 0), [(0, 0), (1, 0), (1, 1), (1, 2), (1, 3), (1, 4), (1, 5), (2, 5), (3, 5), (3, 4), (3, 3), (3, 2), (3, 1), (3,
 0)])
((4, 3), [(0, 0), (1, 0), (1, 1), (1, 2), (1, 3), (1, 4), (1, 5), (2, 5), (3, 5), (3, 4), (3, 3), (4, 3)])
((0, 5), [(0, 0), (1, 0), (1, 1), (1, 2), (1, 3), (1, 4), (1, 5), (0, 5)])
((2, 1), [(0, 0), (1, 0), (1, 1), (2, 1)])
_____________________________________________________________________________
État courant: (3, 0)
Blocage pour la direction: (-1, 0)
Blocage pour la direction: (1, 0)
Blocage pour la direction: (0, -1)
Blocage pour la direction: (0, 1)
_____________________________________________________________________________
((4, 3), [(0, 0), (1, 0), (1, 1), (1, 2), (1, 3), (1, 4), (1, 5), (2, 5), (3, 5), (3, 4), (3, 3), (4, 3)])
((0, 5), [(0, 0), (1, 0), (1, 1), (1, 2), (1, 3), (1, 4), (1, 5), (0, 5)])
((2, 1), [(0, 0), (1, 0), (1, 1), (2, 1)])
_____________________________________________________________________________
État courant: (4, 3)
Blocage pour la direction: (-1, 0)
Ajout du nouvel état: (5, 3)
Blocage pour la direction: (0, -1)
Blocage pour la direction: (0, 1)
_____________________________________________________________________________
((5, 3), [(0, 0), (1, 0), (1, 1), (1, 2), (1, 3), (1, 4), (1, 5), (2, 5), (3, 5), (3, 4), (3, 3), (4, 3), (5, 3)])
((0, 5), [(0, 0), (1, 0), (1, 1), (1, 2), (1, 3), (1, 4), (1, 5), (0, 5)])
((2, 1), [(0, 0), (1, 0), (1, 1), (2, 1)])
_____________________________________________________________________________
État courant: (5, 3)
Blocage pour la direction: (-1, 0)
Blocage pour la direction: (1, 0)
Ajout du nouvel état: (5, 2)
Ajout du nouvel état: (5, 4)
_____________________________________________________________________________
((5, 4), [(0, 0), (1, 0), (1, 1), (1, 2), (1, 3), (1, 4), (1, 5), (2, 5), (3, 5), (3, 4), (3, 3), (4, 3), (5, 3), (5,
 4)])
((5, 2), [(0, 0), (1, 0), (1, 1), (1, 2), (1, 3), (1, 4), (1, 5), (2, 5), (3, 5), (3, 4), (3, 3), (4, 3), (5, 3), (5,
 2)])
((0, 5), [(0, 0), (1, 0), (1, 1), (1, 2), (1, 3), (1, 4), (1, 5), (0, 5)])
((2, 1), [(0, 0), (1, 0), (1, 1), (2, 1)])
_____________________________________________________________________________
État courant: (5, 4)
Blocage pour la direction: (-1, 0)
Blocage pour la direction: (1, 0)
Blocage pour la direction: (0, -1)
Ajout du nouvel état: (5, 5)
_____________________________________________________________________________
_____________________________________________________________________________
((5, 5), [(0, 0), (1, 0), (1, 1), (1, 2), (1, 3), (1, 4), (1, 5), (2, 5), (3, 5), (3, 4), (3, 3), (4, 3), (5, 3), (5,
 4), (5, 5)])
((5, 2), [(0, 0), (1, 0), (1, 1), (1, 2), (1, 3), (1, 4), (1, 5), (2, 5), (3, 5), (3, 4), (3, 3), (4, 3), (5, 3), (5,
 2)])
((0, 5), [(0, 0), (1, 0), (1, 1), (1, 2), (1, 3), (1, 4), (1, 5), (0, 5)])
((2, 1), [(0, 0), (1, 0), (1, 1), (2, 1)])
_____________________________________________________________________________
État courant: (5, 5)
```

RÉSULTAT FINAL DFS

```
Chemin trouvé : [(0, 0), (1, 0), (1, 1), (1, 2), (1, 3), (1, 4), (1, 5), (2, 5), (3, 5), (3, 4), (3, 3), (4, 3), (5, 3), (5, 4), (5, 5)]
Nombre de nœuds explorés : 18
Temps d'exécution : 0.002287 secondes
```



3. A*

L'A* est une méthode de recherche combinant deux aspects :

- Le coût réel, g(n), qui correspond à la distance parcourue depuis le point de départ jusqu'à un nœud donné.
- Le coût estimé, h(n), qui représente une estimation de la distance restante pour atteindre la destination.

Le coût total, f(n), est défini comme la somme de ces deux composantes :

$f(n)=g(n)+h(n)$

Caractéristiques :

- Utilise une file de priorité pour explorer en priorité les nœuds avec le coût total f(n)le plus bas.
- Nécessite une fonction heuristique spécifique au problème, souvent basée sur une estimation réaliste du coût restant.

Avantages :

- Trouve un chemin optimal si la fonction heuristique est admissible (c'est-à-dire qu'elle ne surestime pas le coût).

Limites :

- Dépend fortement de la qualité de la fonction heuristique.
- Peut consommer beaucoup de mémoire pour des graphes de grande taille.

Étapes :

1. Initialiser une file de priorité avec le nœud de départ.
2. Calculer le coût total comme la somme du coût actuel et de l'heuristique (distance de Manhattan).
3. Explorer les voisins non visités, mettre à jour leurs coûts, et les ajouter à la file.
4. Si la destination est atteinte, retourner le chemin.
5. Si la file devient vide sans atteindre la destination, aucun chemin n'existe.

CAPTURES D'ECRAN D'EXECUTION

A*

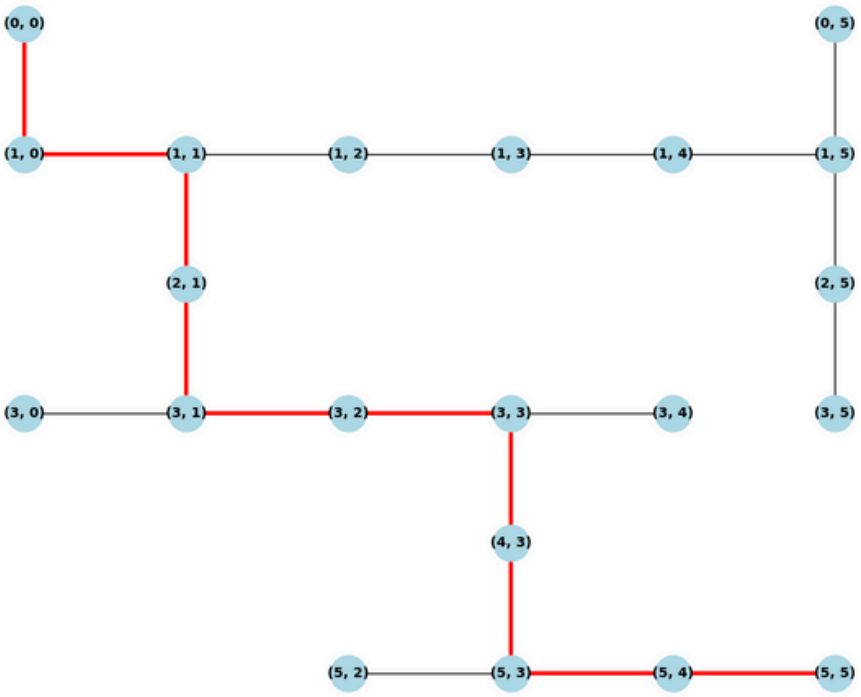
```
A* :
_____________________________________________________________________________
(0, (0, 0), [(0, 0)])
État courant: (0, 0)
Blocage pour la direction: (-1, 0)
Ajout du nouvel état: (1, 0)
Blocage pour la direction: (0, -1)
Blocage pour la direction: (0, 1)

_____________________________________________________________________________
(10, (1, 0), [(0, 0), (1, 0)])
État courant: (1, 0)
Blocage pour la direction: (1, 0)
Blocage pour la direction: (0, -1)
Ajout du nouvel état: (1, 1)

_____________________________________________________________________________
(10, (1, 1), [(0, 0), (1, 0), (1, 1)])
État courant: (1, 1)
Blocage pour la direction: (-1, 0)
Ajout du nouvel état: (2, 1)
Ajout du nouvel état: (1, 2)

_____________________________________________________________________________
(10, (1, 2), [(0, 0), (1, 0), (1, 1), (1, 2)])
(10, (2, 1), [(0, 0), (1, 0), (1, 1), (2, 1)])
État courant: (1, 2)
Blocage pour la direction: (-1, 0)
Blocage pour la direction: (1, 0)
Ajout du nouvel état: (1, 3)
```

```
(10, (1, 3), [(0, 0), (1, 0), (1, 1), (1, 2), (1, 3)])
(10, (2, 1), [(0, 0), (1, 0), (1, 1), (2, 1)])
État courant: (1, 3)
Blocage pour la direction: (-1, 0)
Blocage pour la direction: (1, 0)
Ajout du nouvel état: (1, 4)

_____________________________________________________________________________
(10, (1, 4), [(0, 0), (1, 0), (1, 1), (1, 2), (1, 3), (1, 4)])
(10, (2, 1), [(0, 0), (1, 0), (1, 1), (2, 1)])
État courant: (1, 4)
Blocage pour la direction: (-1, 0)
Blocage pour la direction: (1, 0)
Ajout du nouvel état: (1, 5)

_____________________________________________________________________________
(10, (1, 5), [(0, 0), (1, 0), (1, 1), (1, 2), (1, 3), (1, 4), (1, 5)])
(10, (2, 1), [(0, 0), (1, 0), (1, 1), (2, 1)])
État courant: (1, 5)
Ajout du nouvel état: (0, 5)
Ajout du nouvel état: (2, 5)
Blocage pour la direction: (0, 1)

_____________________________________________________________________________
(10, (2, 1), [(0, 0), (1, 0), (1, 1), (2, 1)])
(10, (2, 5), [(0, 0), (1, 0), (1, 1), (1, 2), (1, 3), (1, 4), (1, 5), (2, 5)])
(12, (0, 5), [(0, 0), (1, 0), (1, 1), (1, 2), (1, 3), (1, 4), (1, 5), (0, 5)])
État courant: (2, 1)
Ajout du nouvel état: (3, 1)
Blocage pour la direction: (0, -1)
Blocage pour la direction: (0, 1)
```

```
_____________________________________________________________________________
(10, (2, 5), [(0, 0), (1, 0), (1, 1), (1, 2), (1, 3), (1, 4), (1, 5), (2, 5)])
(10, (3, 1), [(0, 0), (1, 0), (1, 1), (2, 1), (3, 1)])
(12, (0, 5), [(0, 0), (1, 0), (1, 1), (1, 2), (1, 3), (1, 4), (1, 5), (0, 5)])
État courant: (2, 5)
Ajout du nouvel état: (3, 5)
Blocage pour la direction: (0, -1)
Blocage pour la direction: (0, 1)

_____________________________________________________________________________
(10, (3, 1), [(0, 0), (1, 0), (1, 1), (2, 1), (3, 1)])
(10, (3, 5), [(0, 0), (1, 0), (1, 1), (1, 2), (1, 3), (1, 4), (1, 5), (2, 5), (3, 5)])
(12, (0, 5), [(0, 0), (1, 0), (1, 1), (1, 2), (1, 3), (1, 4), (1, 5), (0, 5)])
État courant: (3, 1)
Blocage pour la direction: (1, 0)
Ajout du nouvel état: (3, 0)
Ajout du nouvel état: (3, 2)

_____________________________________________________________________________
(10, (3, 2), [(0, 0), (1, 0), (1, 1), (2, 1), (3, 1), (3, 2)])
(10, (3, 5), [(0, 0), (1, 0), (1, 1), (1, 2), (1, 3), (1, 4), (1, 5), (2, 5), (3, 5)])
(12, (0, 5), [(0, 0), (1, 0), (1, 1), (1, 2), (1, 3), (1, 4), (1, 5), (0, 5)])
(12, (3, 0), [(0, 0), (1, 0), (1, 1), (2, 1), (3, 1), (3, 0)])
État courant: (3, 2)
Blocage pour la direction: (-1, 0)
Blocage pour la direction: (1, 0)
Ajout du nouvel état: (3, 3)

_____________________________________________________________________________
(10, (3, 3), [(0, 0), (1, 0), (1, 1), (2, 1), (3, 1), (3, 2), (3, 3)])
(10, (3, 5), [(0, 0), (1, 0), (1, 1), (1, 2), (1, 3), (1, 4), (1, 5), (2, 5), (3, 5)])
(12, (0, 5), [(0, 0), (1, 0), (1, 1), (1, 2), (1, 3), (1, 4), (1, 5), (0, 5)])
(12, (3, 0), [(0, 0), (1, 0), (1, 1), (2, 1), (3, 1), (3, 0)])
État courant: (3, 3)
Blocage pour la direction: (-1, 0)
Ajout du nouvel état: (4, 3)
Ajout du nouvel état: (3, 4)

_____________________________________________________________________________
(10, (3, 4), [(0, 0), (1, 0), (1, 1), (2, 1), (3, 1), (3, 2), (3, 3), (3, 4)])
(10, (3, 5), [(0, 0), (1, 0), (1, 1), (1, 2), (1, 3), (1, 4), (1, 5), (2, 5), (3, 5)])
(10, (4, 3), [(0, 0), (1, 0), (1, 1), (2, 1), (3, 1), (3, 2), (3, 3), (4, 3)])
(12, (0, 5), [(0, 0), (1, 0), (1, 1), (1, 2), (1, 3), (1, 4), (1, 5), (0, 5)])
(12, (3, 0), [(0, 0), (1, 0), (1, 1), (2, 1), (3, 1), (3, 0)])
État courant: (3, 4)
Blocage pour la direction: (-1, 0)
Blocage pour la direction: (1, 0)

_____________________________________________________________________________
(10, (3, 5), [(0, 0), (1, 0), (1, 1), (1, 2), (1, 3), (1, 4), (1, 5), (2, 5), (3, 5)])
(10, (4, 3), [(0, 0), (1, 0), (1, 1), (2, 1), (3, 1), (3, 2), (3, 3), (4, 3)])
(12, (0, 5), [(0, 0), (1, 0), (1, 1), (1, 2), (1, 3), (1, 4), (1, 5), (0, 5)])
(12, (3, 0), [(0, 0), (1, 0), (1, 1), (2, 1), (3, 1), (3, 0)])
État courant: (3, 5)
Blocage pour la direction: (1, 0)
Blocage pour la direction: (0, 1)

_____________________________________________________________________________
(10, (4, 3), [(0, 0), (1, 0), (1, 1), (2, 1), (3, 1), (3, 2), (3, 3), (4, 3)])
(12, (0, 5), [(0, 0), (1, 0), (1, 1), (1, 2), (1, 3), (1, 4), (1, 5), (0, 5)])
(12, (3, 0), [(0, 0), (1, 0), (1, 1), (2, 1), (3, 1), (3, 0)])
État courant: (4, 3)
Ajout du nouvel état: (5, 3)
Blocage pour la direction: (0, -1)
Blocage pour la direction: (0, 1)

_____________________________________________________________________________
(10, (5, 3), [(0, 0), (1, 0), (1, 1), (2, 1), (3, 1), (3, 2), (3, 3), (4, 3), (5, 3)])
(12, (0, 5), [(0, 0), (1, 0), (1, 1), (1, 2), (1, 3), (1, 4), (1, 5), (0, 5)])
(12, (3, 0), [(0, 0), (1, 0), (1, 1), (2, 1), (3, 1), (3, 0)])
État courant: (5, 3)
Blocage pour la direction: (1, 0)
Ajout du nouvel état: (5, 2)
Ajout du nouvel état: (5, 4)

_____________________________________________________________________________
(10, (5, 4), [(0, 0), (1, 0), (1, 1), (2, 1), (3, 1), (3, 2), (3, 3), (4, 3), (5, 3), (5, 4)])
(12, (0, 5), [(0, 0), (1, 0), (1, 1), (1, 2), (1, 3), (1, 4), (1, 5), (0, 5)])
(12, (3, 0), [(0, 0), (1, 0), (1, 1), (2, 1), (3, 1), (3, 0)])
(12, (5, 2), [(0, 0), (1, 0), (1, 1), (2, 1), (3, 1), (3, 2), (3, 3), (4, 3), (5, 3), (5, 2)])
État courant: (5, 4)
Blocage pour la direction: (-1, 0)
Blocage pour la direction: (1, 0)
), (1, 5), (0, 5)])
(12, (3, 0), [(0, 0), (1, 0), (1, 1), (2, 1), (3, 1), (3, 0)])
(12, (5, 2), [(0, 0), (1, 0), (1, 1), (2, 1), (3, 1), (3, 2), (3, 3), (4, 3), (5, 3), (5, 2)])
État courant: (5, 5)
```

RÉSULTAT FINAL A*

```
Chemin trouvé : [(0, 0), (1, 0), (1, 1), (2, 1), (3, 1), (3, 2), (3, 3), (4, 3), (5, 3), (5, 4), (5, 5)]
Nombre de nœuds explorés : 18
Temps d'exécution : 0.001897 secondes
```



ÉTUDE COMPARATIVE

Algorithme	Chemin Trouvé	Noeuds Explotés	Temps d'éxecution
BFS	Oui	20	0.002512 secondes
DFS	Oui	18	0.002287 secondes
A*	Oui	18	0.001897 secondes

Observations :

- BFS : Explore de nombreux nœuds (explore plusieurs chemins avant de trouver une solution). Le nombre de nœuds explorés dépend de la profondeur du graphe.
- DFS : Explore tous les nœuds niveau par niveau jusqu'à atteindre le but. Le nombre de nœuds explorés est souvent plus élevé pour des solutions profondes.
- A* : Efficace grâce à la fonction heuristique qui priorise les nœuds proches de la destination.