# UNIVERSIDADE DE SÃO PAULO

**INSTITUTO DE CIÊNCIAS MATEMÁTICAS E COMPUTAÇÃO**


*Trabalho Prático 1*

*23/10/2017*


**Danilo Henrique Cordeiro - 6791651**



**São Carlos - SP**

# Introdução

O trabalho consiste na implementação de 4 algoritmos de busca para se encontrar o caminho em um labirinto. Os algoritmos implementados são: Busca em Profundidade, Busca em Largura, busca *Best-First Search* e o A*. Os dois primeiros são de busca cega e os 2 últimos são de busca informada.

O labirinto é uma entrada de um arquivo de texto da seguinte maneira: a primeira linha contem o número de linhas e o número de colunas respectivamente, a partir da segunda linha vem o labirinto, onde o caractere "*"é um espaço em que é possível se movimentar no labirinto, o caractere -"significa uma parede onde não se pode passar, o caractere "#" é a posição inicial e por fim o caractere "$" é o objetivo no labirinto. a Figura 1 mostra um exemplo de arquivo de entrada de um labirinto.

```
13 15
***#**********
**************
----------**
**************
**********|**
**************
**************
**_**********
**_**********
**_****_-------
**_**********
**_****_****$**
**_****_*******
```

Figura 1: Exemplo do arquivo de entrada.

# 1 Algoritmos

## 1.1 Busca em Profundidade

O algoritmo DFS é um algoritmo de busca não informada que pega o primeiro nó filho e se aprofunda até o fim dele, chegando ao fim e não encontrando o objetivo ele retrocede e começa pelo outro nó e vai assim até encontrar o objetivo ou percorrer todos os nós, é utilizada uma estrutura de dados de pilha.

O DFS não é ótimo e nem garante encontrar a melhor solução. Isso significa que ele não é uma boa escolha para se encontrar um caminho em um labirinto. Mas pode-se usar ele para se achar componentes conectados e até para geração de labirintos.

## 1.2 Busca em Largura

A BFS é um algoritmo de busca não informada que explora primeiro os nós vizinhos antes de seguir para o próximo nível de vizinhos. Para A BFS se utiliza a estrutura de dados de fila

BFS é ótimo e é garantido que ele irá encontrar a melhor solução que existir. No pior caso temos que a complexidade do BFS é de $O(N + A)$, onde N é o número de nós e A é o número de arestas no grafo.

## 1.3 Busca *Best-First Search*

O algoritmo *Best-First* é um algoritmo de busca informada, ou seja, ele faz uma investigação dos nós que "prometem"dirigir a busca mais rapidamente para o nó objetivo.

O *Best-First* é um caso especial do algoritmo $A^*$, que é apresentado a seguir, com $h(n) = 0$)

## 1.4 Busca em $A^*$

O algoritmo de busca $A^*$ é muito utilizado. Ele utiliza uma busca gulosa e encontra o caminho de menor custo dado um nó inicial até o nó objetivo. Como o $A^*$ percorre o grafo ele segue o caminho com o menor custo esperado, mantendo a ordem da fila de

prioridades. Ele usa heurísticas para calcular a distância do nó atual até o nó objetivo. Com isso temos:

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

sendo $g(n)$ o custo até o momento para se alcançar o nó $n$ e $h(n)$ o custo obtido pela heurística de se chegar até o nó objetivo a partir do nó $n$.

Mas esta heurística deve ser admissível, ou seja, ela não ultrapassa a distância real até o objetivo. A complexidade do $A^*$ depende da heurística utilizada.

# 2 Implementação e Execução

Os algoritmos foram implementados na linguagem Python, em executado em um notebook com sistema operacional Ubuntu instalado, um processador i5 de 2.20GHz e com 8GB de memória RAM.

Para facilitar o uso dos algoritmos primeiro o labirinto foi convertido para um grafo, sendo que para cada nó foi criado uma lista com cada vizinho válido para poder seguir.

Para a execução do programa estando na pasta dos arquivos executar o seguinte comando:

```
python labirinto.py labirinto.txt busca
```

Onde labirinto.py é o nome do programa feito, labirinto.txt é o arquivo de texto com o labirinto similar ao da Figura 1 e busca é o número do tipo de busca utilizada: 1 - Busca em Profundidade, 2 - Busca em Largura, 3 - Busca *Best-First Search* e 4 - Busca $A^*$.

Caso queira acompanhar o passo a passo de como foi percorrido o caminho basta adicionar mais um parâmetro com o valor de 1. Da seguinte maneira:

```
python labirinto.py labirinto.txt busca 1
```

Foi feito também um algoritmo para gerar labirintos. Ele funciona da seguinte maneira, primeiro se gera um labirinto com todos os espaços vazios para em seguida ir

colocando obstáculos em posições aleatórias e assim ele gera o labirinto. O código também foi incluído no trabalho, estando na pasta do trabalho basta executar o seguinte comando

python geraLabirinto.py labirinto.txt linhas colunas

onde linhas e colunas são os tamanhos para se criar o labirinto, se quiser é possível retirar esses dois parâmetros e assim o programa gera um labirinto de tamanho aleatório.

A Figura 2 é o labirinto de exemplo fornecido no enunciado do trabalho. Foi nele que foi realizado os primeiros testes dos algoritmos. A Figura 3 mostra os caminhos encontrados por cada algoritmo de busca.

Foi implementado duas heurísticas para se utilizar no algoritmo do $A^*$, a distância de Manhattan e a Euclidiana. Para todos os testes apresentados a seguir foi utilizado a distância Euclidiana.

A Figura 3 mostra os caminhos encontrados pelos algoritmos implementados e utilizados no labirinto da Figura 2. Já na Figura 4 foi usado os mesmos algoritmos mas desta vez foi liberado caminhar pelas diagonais nos labirintos.

Na Tabela 1 temos os tempos para a resolução de cada labirinto por cada algoritmo e a quantidade de expansões realizadas para se chegar na solução e na Tabela 2 temos os tempos e as expansões para os algoritmos com as diagonais liberadas.

Em especial ao liberar a possibilidade de se andar nas diagonais os algoritmos conseguem encontrar uma solução para o labirinto 8 da Figura 11, coisa que seria impossível andando somente para cima, para baixo ou para os lados.

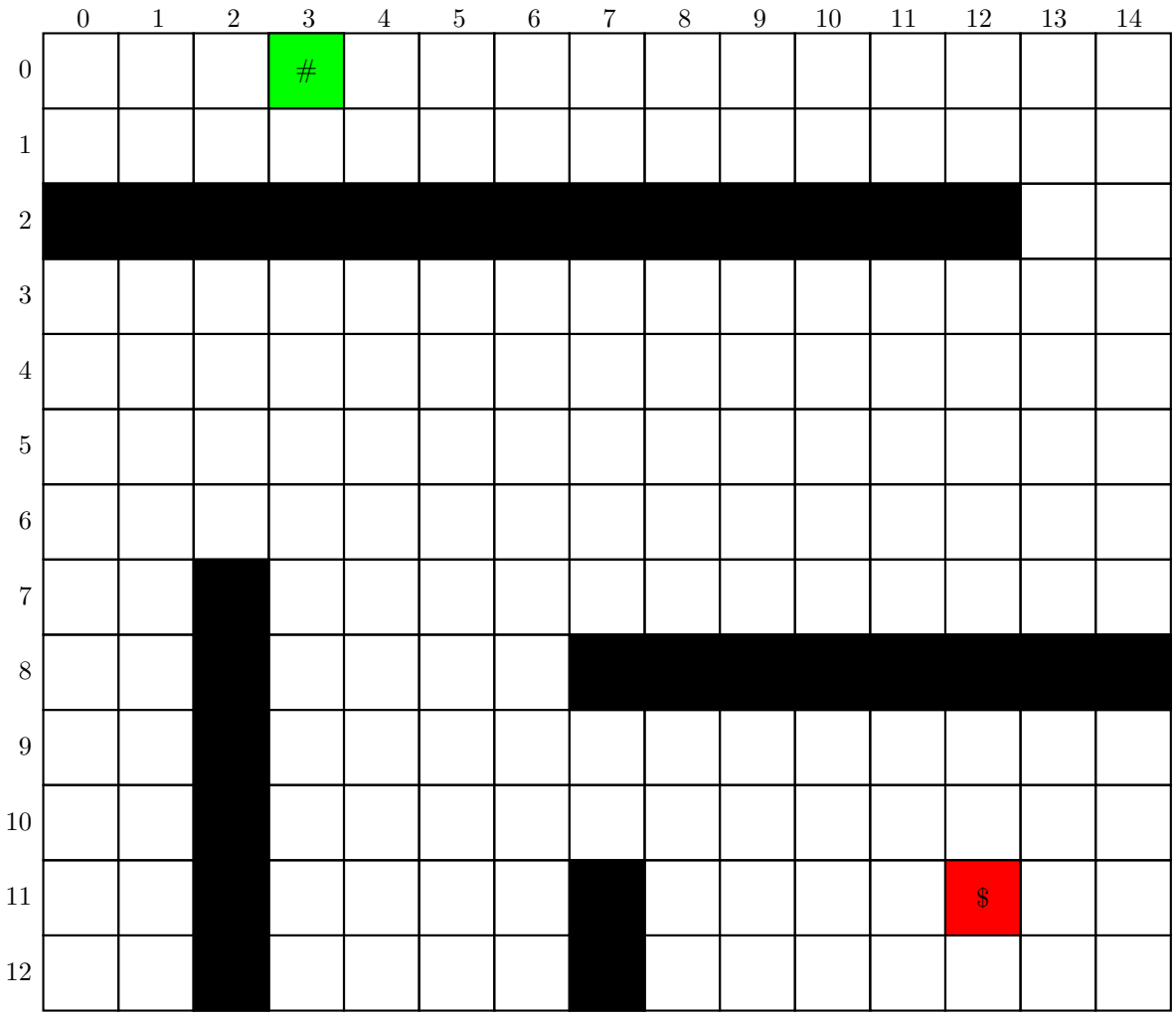
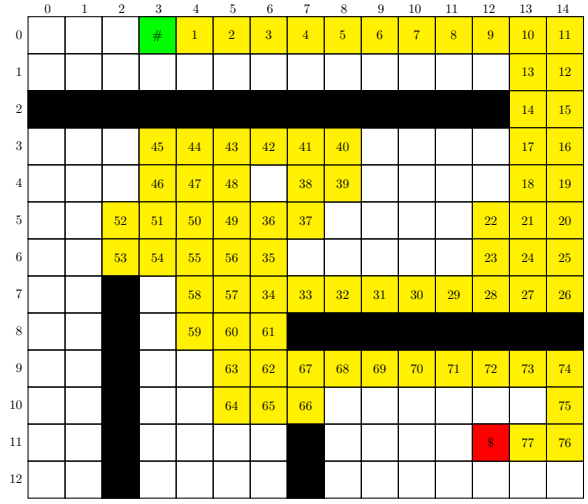
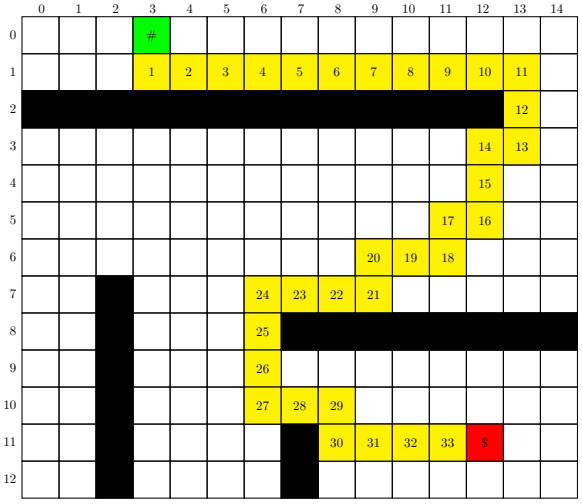


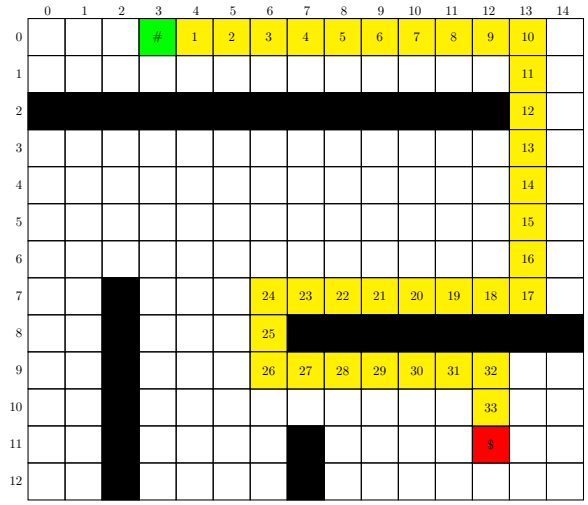

Figura 2: Labirinto 1 fornecido como exemplo.



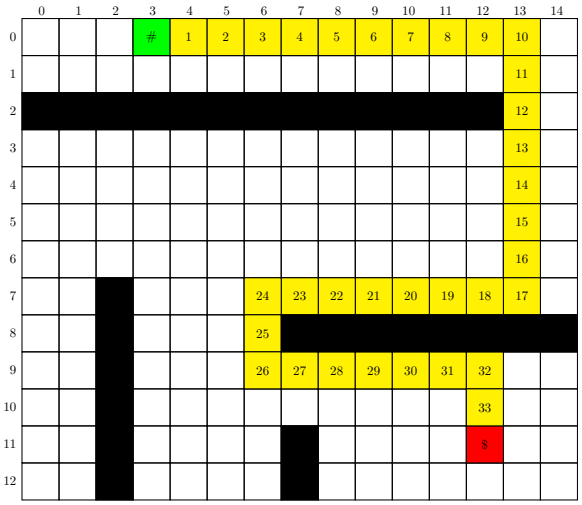

(a) Caminho encontrado pelo DFS.




(b) Caminho encontrado pelo BFS.

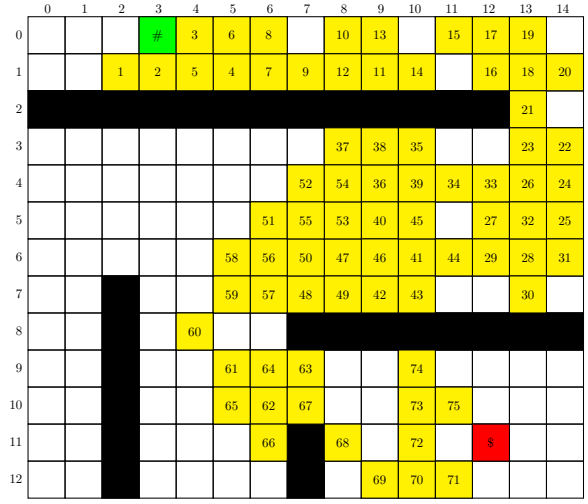



(c) Caminho encontrado pelo *Best-First Search*.

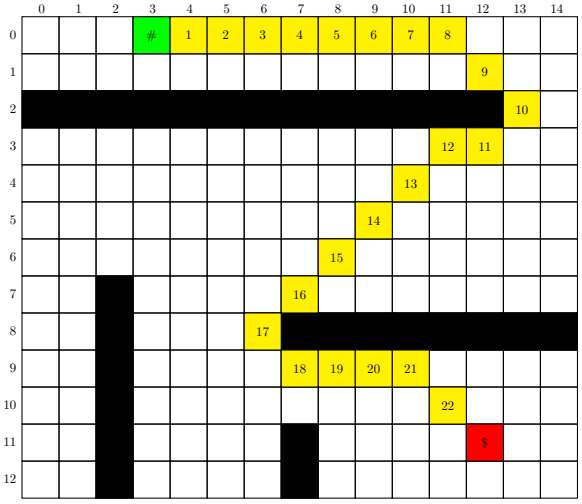



(d) Caminho encontrado pelo $A^*$.

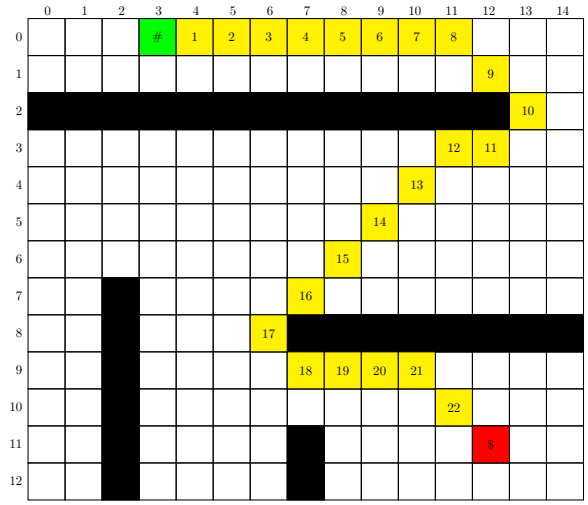
Figura 3: Caminhos percorridos.



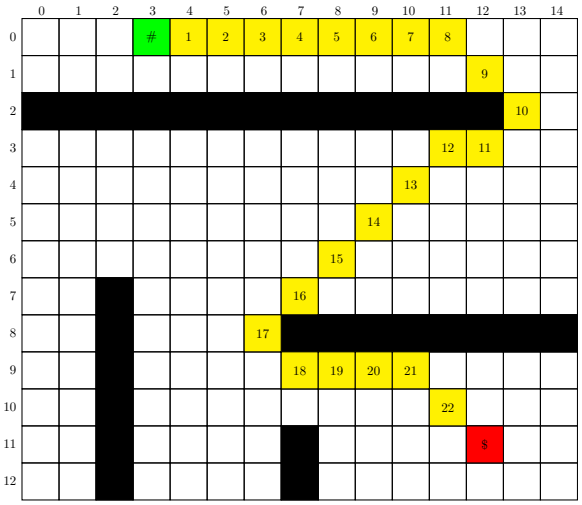

(a) Caminho encontrado pelo DFS.




(b) Caminho encontrado pelo BFS.




(c) Caminho encontrado pelo *Best-First Search*.




(d) Caminho encontrado pelo $A^*$.

Figura 4: Caminhos percorridos com diagonais habilitadas.

Tabela 1: Tabela de tempos.

| Labirinto | Algoritmo | Tempo(s) | expansões |
|---|---|---|---|
| 1 | DFS | 0.000605821609497 | 331 |
| | BFS | 0.000679016113281 | 515 |
| | *Best-First* | 0.00105595588684 | 518 |
| | $A^*$ | 0.00117707252502 | 506 |
| 2 | DFS | 0.000670909881592 | 118 |
| | BFS | 0.00043797492981 | 119 |
| | *Best-First* | 0.00104808807373 | 119 |
| | $A^*$ | 0.000692129135132 | 119 |
| 3 | DFS | 0.000885963439941 | 184 |
| | BFS | 0.00107216835022 | 526 |
| | *Best-First* | 0.00176095962524 | 526 |
| | $A^*$ | 0.00147581100464 | 526 |
| 4 | DFS | 0.000619173049927 | 111 |
| | BFS | 0.000465154647827 | 84 |
| | *Best-First* | 0.000593185424805 | 86 |
| | $A^*$ | 0.000762939453125 | 82 |
| 5 | DFS | 0.000746011734009 | 119 |
| | BFS | 0.000422954559326 | 119 |
| | *Best-First* | 0.00043797492981 | 119 |
| | $A^*$ | 0.000598907470703 | 119 |
| 6 | DFS | 0.294569015503 | 952 |
| | BFS | 0.00356316566467 | 1279 |
| | *Best-First* | 0.00460720062256 | 1281 |
| | $A^*$ | 0.00453209877014 | 1270 |
| 7 | DFS | 0.00776600837708 | 1066 |
| | BFS | 0.00875020027161 | 5983 |
| | *Best-First* | 0.0186059474945 | 5937 |
| | $A^*$ | 0.0207331180573 | 5783 |
| 8 | DFS | 0.000612020492554 | 122 |
| | BFS | 0.000588893890381 | 122 |
| | *Best-First* | 0.00122809410095 | 122 |
| | $A^*$ | 0.000687122344971 | 122 |

Tabela 2: Tabela de tempos, com as diagonais ativadas.

| Labirinto | Algoritmo | Tempo(s) | expansões |
|---|---|---|---|
| 1 | DFS | 0.00216388702393 | 492 |
| | BFS | 0.0022029876709 | 902 |
| | *Best-First* | 0.00309491157532 | 902 |
| | $A^*$ | 0.00303602218628 | 822 |
| 2 | DFS | 0.000586032867432 | 172 |
| | BFS | 0.000991106033325 | 179 |
| | *Best-First* | 0.000763177871704 | 176 |
| | $A^*$ | 0.00086522102356 | 176 |
| 3 | DFS | 0.00173783302307 | 169 |
| | BFS | 0.00225687026978 | 987 |
| | *Best-First* | 0.00298380851746 | 987 |
| | $A^*$ | 0.00300097465515 | 868 |
| 4 | DFS | 0.000754117965698 | 133 |
| | BFS | 0.00064492225647 | 133 |
| | *Best-First* | 0.000736951828003 | 133 |
| | $A^*$ | 0.000804901123047 | 131 |
| 5 | DFS | 0.000549077987671 | 145 |
| | BFS | 0.000499963760376 | 151 |
| | *Best-First* | 0.000594139099121 | 151 |
| | $A^*$ | 0.000648021697998 | 151 |
| 6 | DFS | 0.00580501556396 | 1144 |
| | BFS | 0.00430488586426 | 1828 |
| | *Best-First* | 0.00760817527771 | 1825 |
| | $A^*$ | 0.00567197799683 | 1806 |
| 7 | DFS | 0.0339920520782 | 12209 |
| | BFS | 0.020271062851 | 11569 |
| | *Best-First* | 0.0402519702911 | 11569 |
| | $A^*$ | 0.0382659435272 | 11101 |
| 8 | DFS | 0.00784087181091 | 560 |
| | BFS | 0.00210404396057 | 829 |
| | *Best-First* | 0.00322294235229 | 831 |
| | $A^*$ | 0.00413608551025 | 774 |

# 3 Labirintos Usados

Além do labirinto da Figura 2 foi utilizado outros labirintos como caso de testes que são exibidos a seguir. O labirinto da Figura 10 foi gerado aleatoriamente utilizando o gerador de labirintos mencionado anteriormente, já o labirinto da Figura 11 que também foi gerado aleatoriamente não possui um caminho até o destino, ele foi utilizado para saber se os algoritmos estão sabendo identificar se não possui solução para o labirinto. Todos estes labirintos estão na pasta do trabalho.

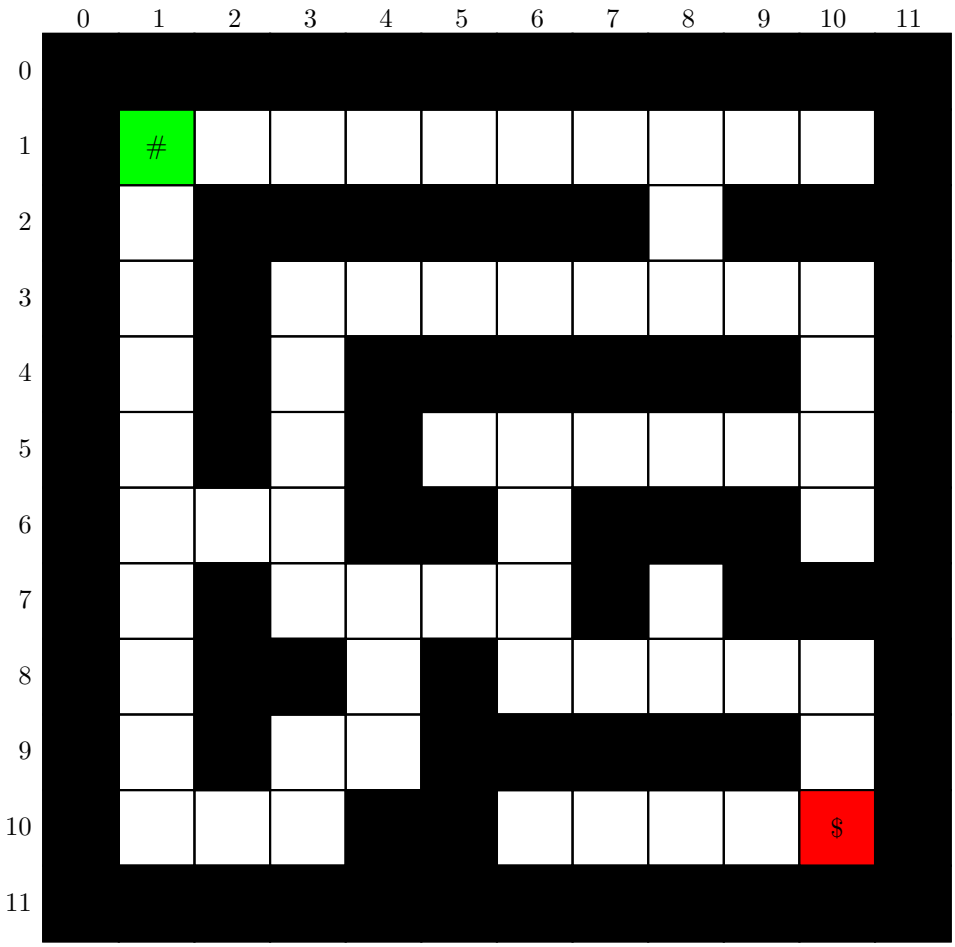



Figura 5: Labirinto 2 fornecido como exemplo.

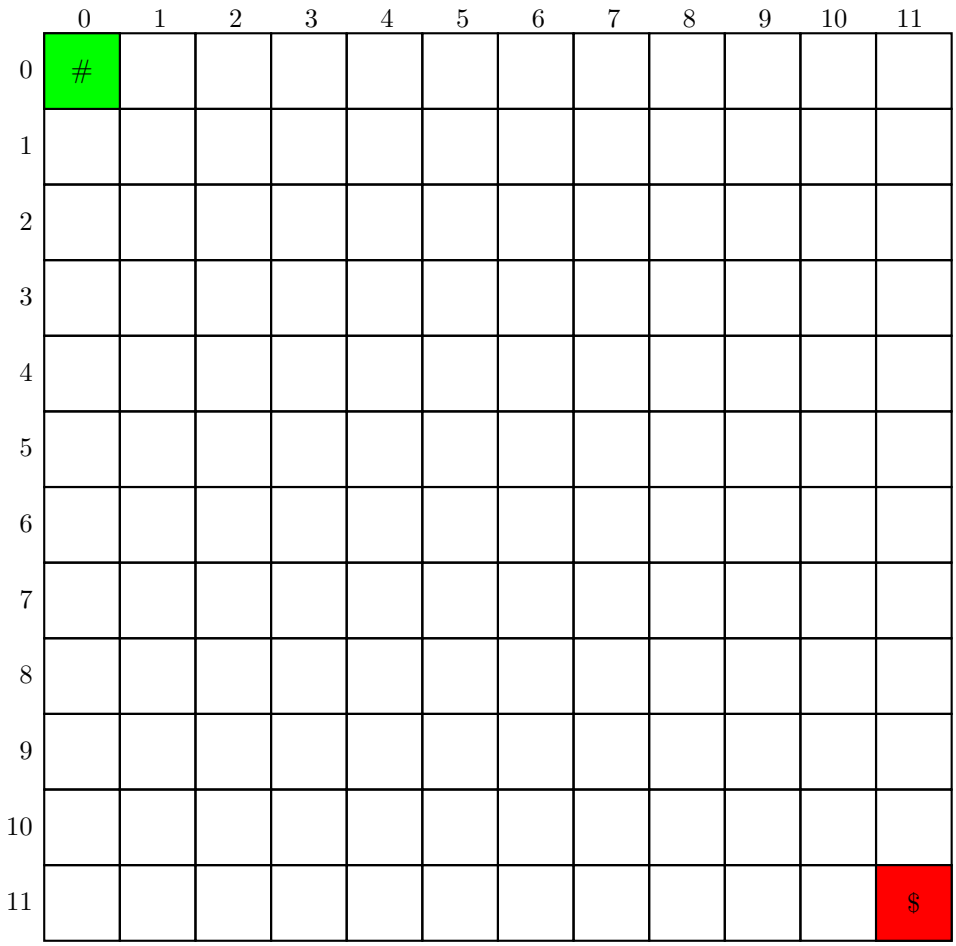



Figura 6: Labirinto 3 fornecido como exemplo.

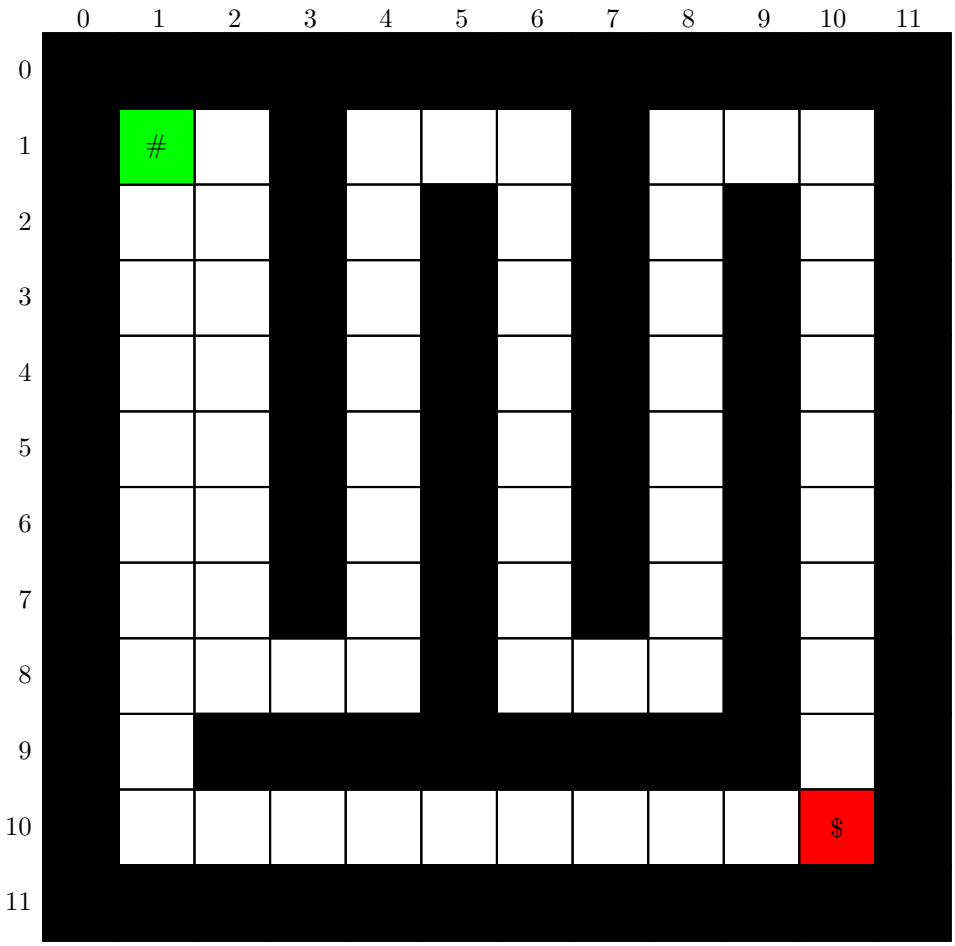



Figura 7: Labirinto 4 fornecido como exemplo.

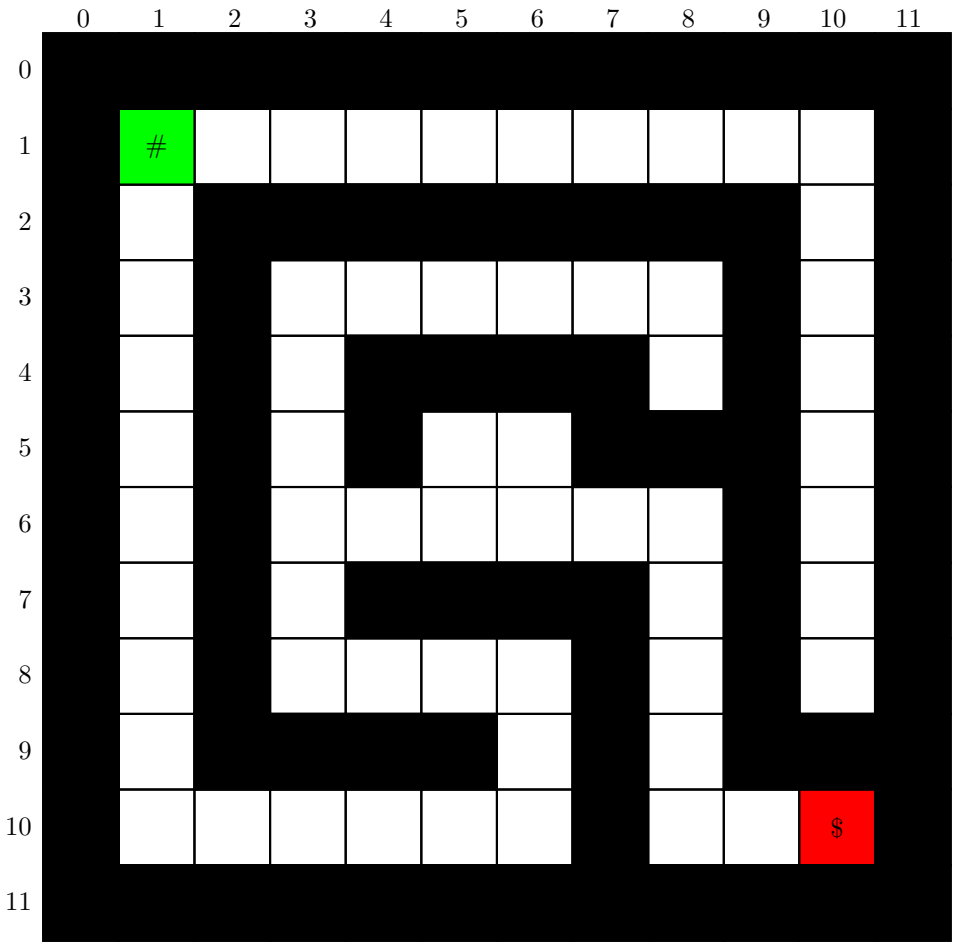



Figura 8: Labirinto 5 fornecido como exemplo.

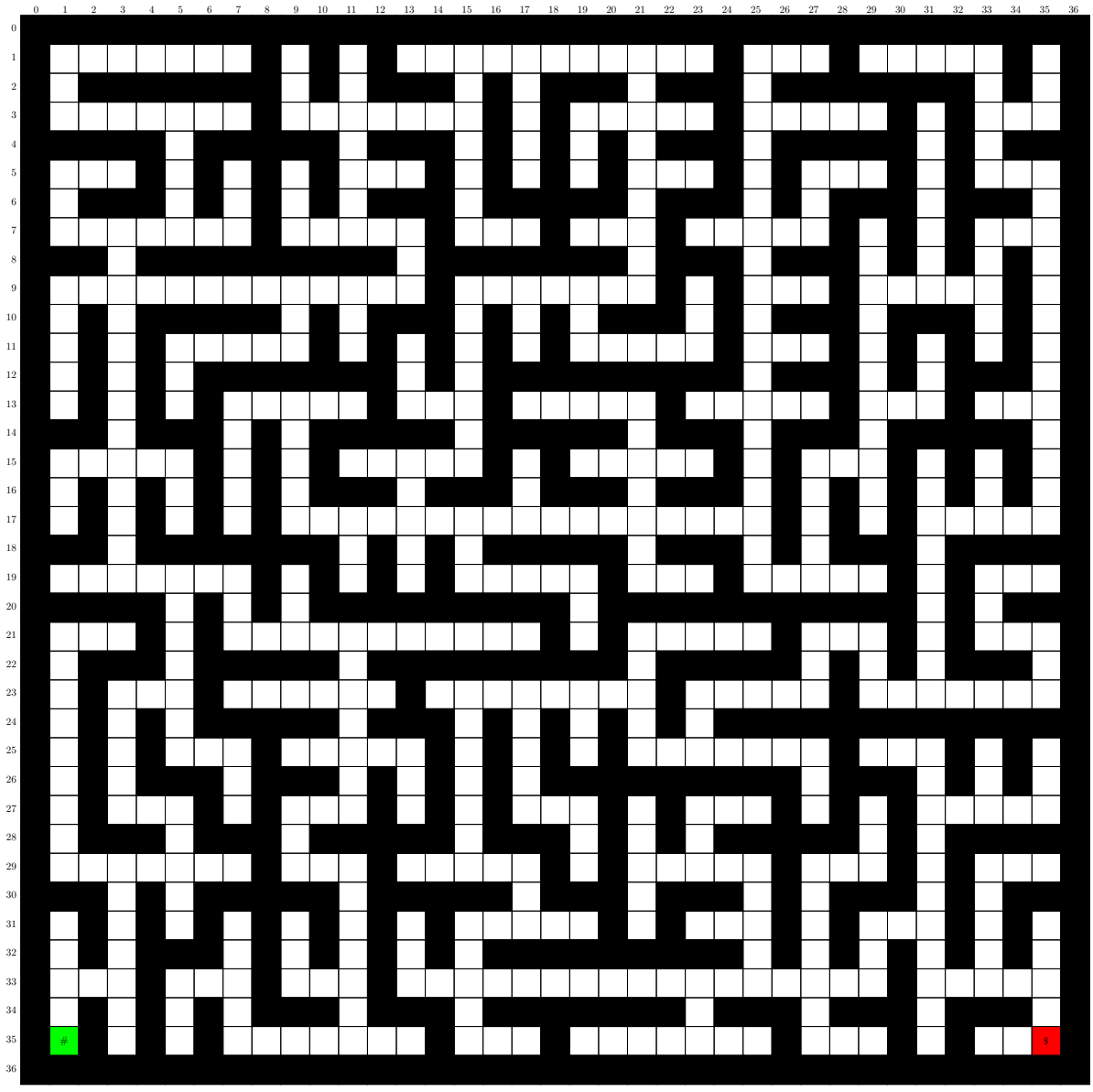



Figura 9: Labirinto 6 fornecido como exemplo.

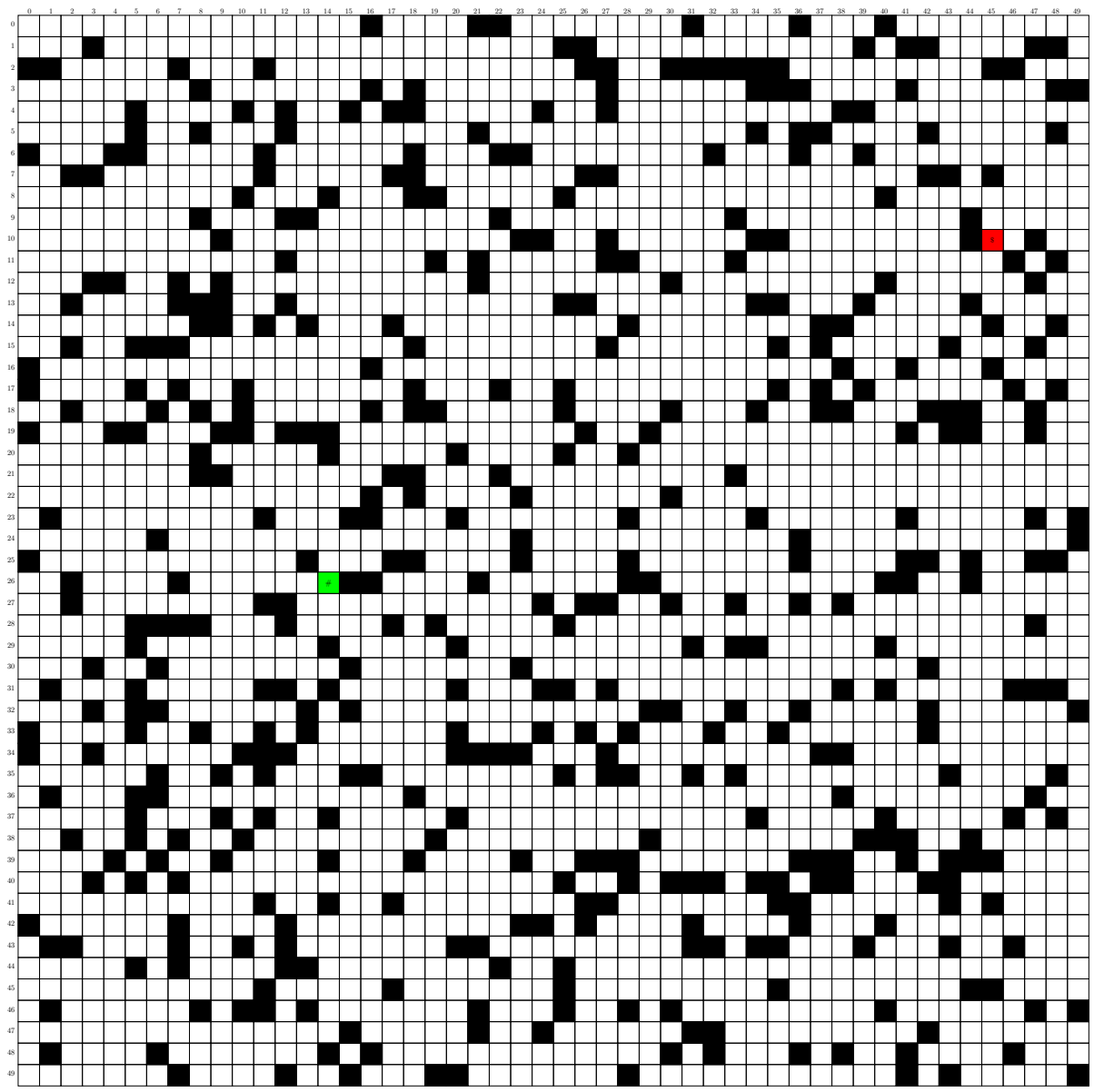


Figura 10: Labirinto 7 fornecido como exemplo.

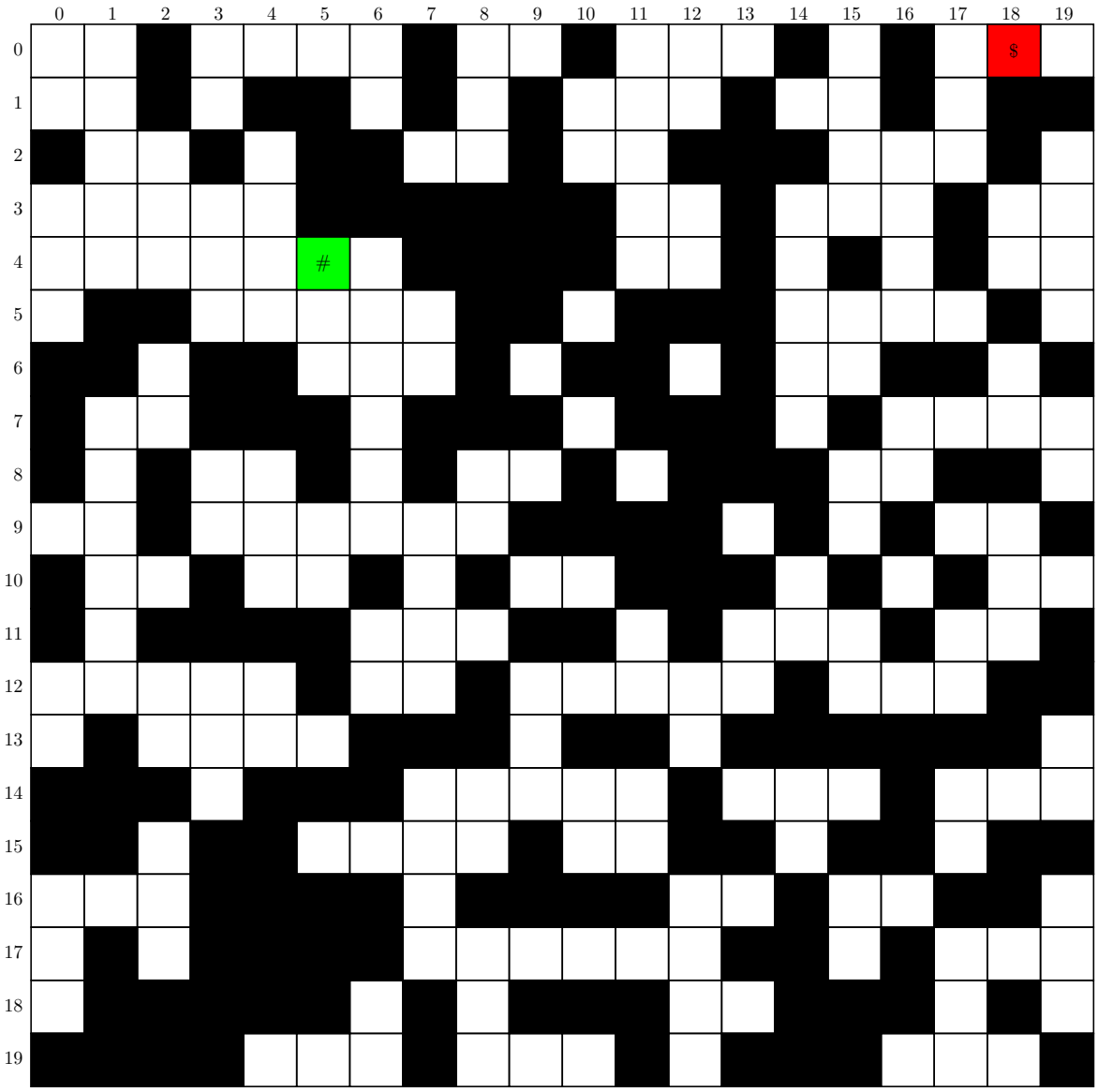



Figura 11: Labirinto 8 fornecido como exemplo.