Inteligência Artificial 2021/2022 - FEUP

+

Exactly One Maze

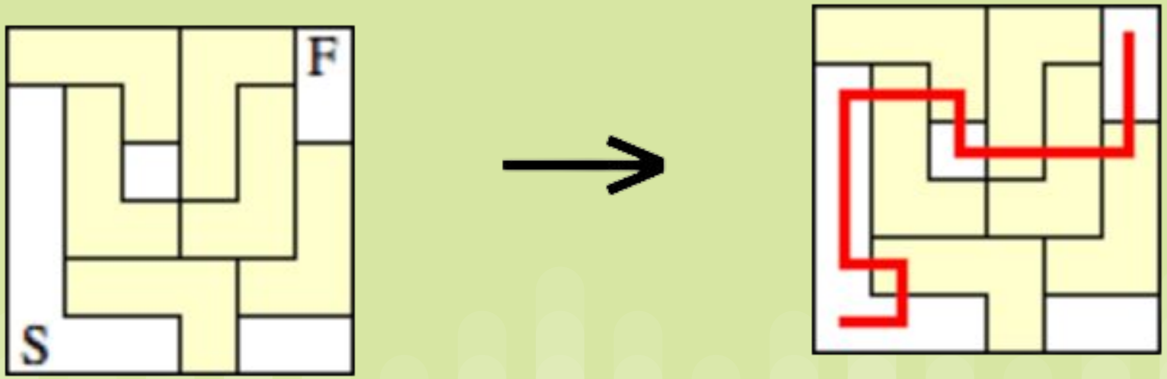
Duarte Sardão - 201905497

Edgar Lourenço - 201604910

Pedro Pereira - 201905508

Work to be performed

- Find a path that moves horizontally and vertically from the Start in the lower left to the Finish in the upper right.
- The path must pass through exactly one unit square of each L shape.
- A solved example is shown below.
- Each puzzle below has a unique solution.



Problem Formulation

State representation:

- 2D Vector {{pos11,..,pos1x},..{posx1,..,posxx}}
  Pos: A-Z | . (X represents path, other letters blocks, . empty space)
- Set of Crossed Blocks [A..Z]
- Set of Blocks to Cross [A..Z]
- Int current column
- Int current row

Problem Formulation

Initial State:

- 2D Vector {{pos11,..,pos1x},..{**X**,..,posxx}}
- Set of Crossed Blocks []
- Set of Blocks to Cross [A..Z]
- Int row = x
- Int col = 0

Problem Formulation

Initial State:

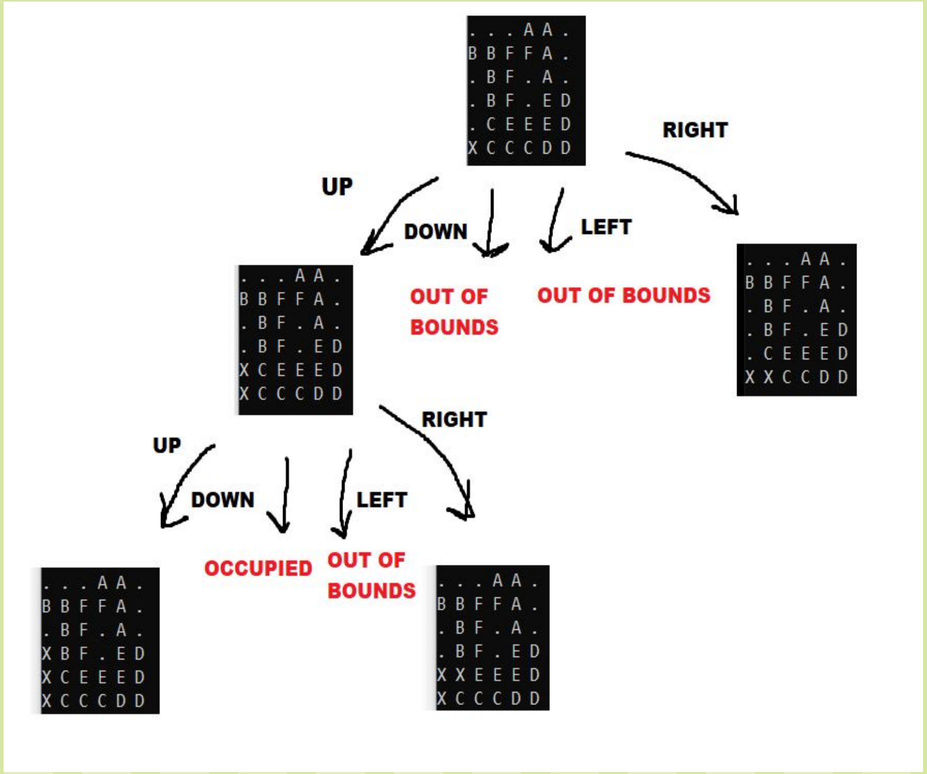
- 2D Vector: {{pos11,..,**X**},..{X,..,posxx}}
- Set of Crossed Blocks [A..Z]
- Set of Blocks to Cross [A..Z]
- Set of Crossed = Set to Cross
- Int row = 0
- Int col = x

Operations:

Name	Precond	Effects	Cost
UP	row > 0 && Grid[row-1][col] not in CrossedBlocks or X	row += 1 Grid[row][col] added to CrossedBlocks Grid[row][col] = ‘X’	1
DOWN	row < size-1 && Grid[row+1][col] not in CrossedBlocks or X	row -= 1 Grid[row][col] added to CrossedBlocks Grid[row][col] = ‘X’	1
LEFT	col > 0 && Grid[row][col-1] not in CrossedBlocks or X	col += 1 Grid[row][col] added to CrossedBlocks Grid[row][col] = ‘X’	1
RIGHT	row < size-1 && Grid[row][col+1] not in CrossedBlocks or X	col -= 1 Grid[row][col] added to CrossedBlocks Grid[row][col] = ‘X’	1

Problem Formulation

- **Search tree**
- **High amount of restrictions** (bounds, already crossed blocks, previously drawn line) means expansion rate will always be lower than 4



```
. . . A A .
B B F F A .
. B F . A .
. B F . E D
. C E E E D
X C C C D D
```

```
. . . A A .
B B F F A .
. B F . A .
. B F . E D
X C E E E D
X C C C D D
```

```
. . . A A .
B B F F A .
. B F . A .
. B F . E D
. C E E E D
X X C C D D
```

```
. . . A A .
B B F F A .
. B F . A .
X B F . E D
X C E E E D
X C C C D D
```

```
. . . A A .
B B F F A .
. B F . A .
. B F . E D
X X E E E D
X C C C D D
```

Implemented Algorithms (Uninformed)

C++ Implementation

Depth First Search

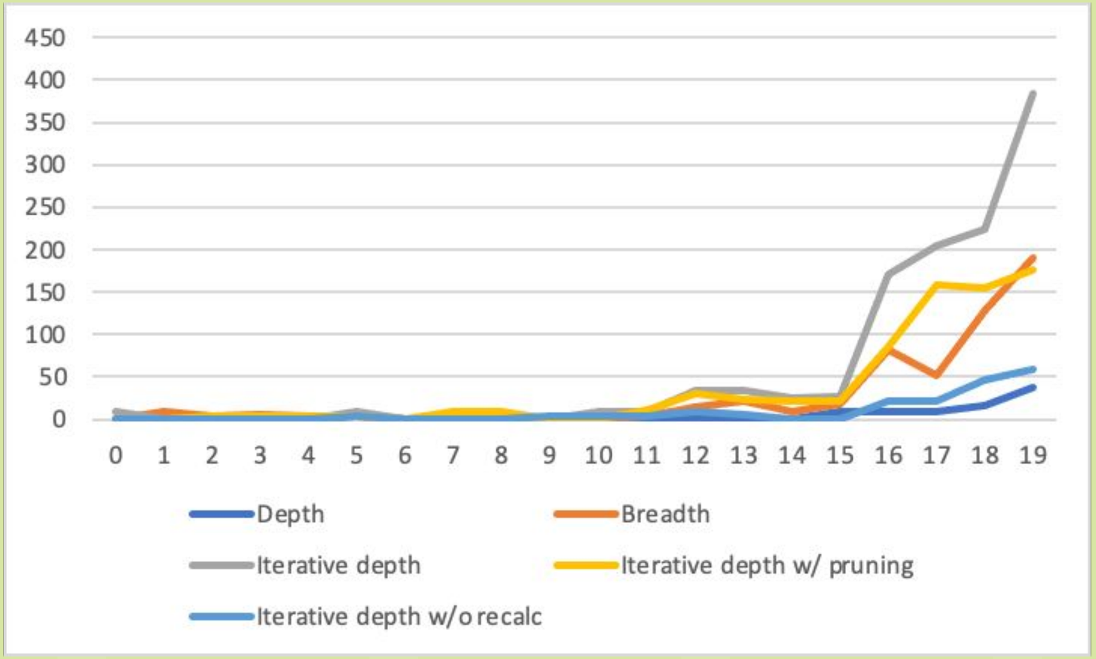
Breadth First Search

Iterative Deepening Search

Iterative Deepening Search with pruning of branches (we can calculate distance to target, so if distance is greater than the remaining depth we can explore, branch will never reach target and is cut)

Iterative Deepening Search without recalculations (branches that reach max depth or are pruned are saved and after deepening, explored without being recalculated from scratch)

Experimental Results



Grid sizes:

0-6 -> 6x6
7-11 -> 7x7
12-15 -> 8x8
16-19 -> 9x9

Average of 2 analysis, Y-Axis: Time in milliseconds, X-Axis: Tested grid

Heuristics tried for Informed Search and Results

Dist(s)

-Dist(s)

Blocks_Crossed(s)

-Blocks_Crossed(s)

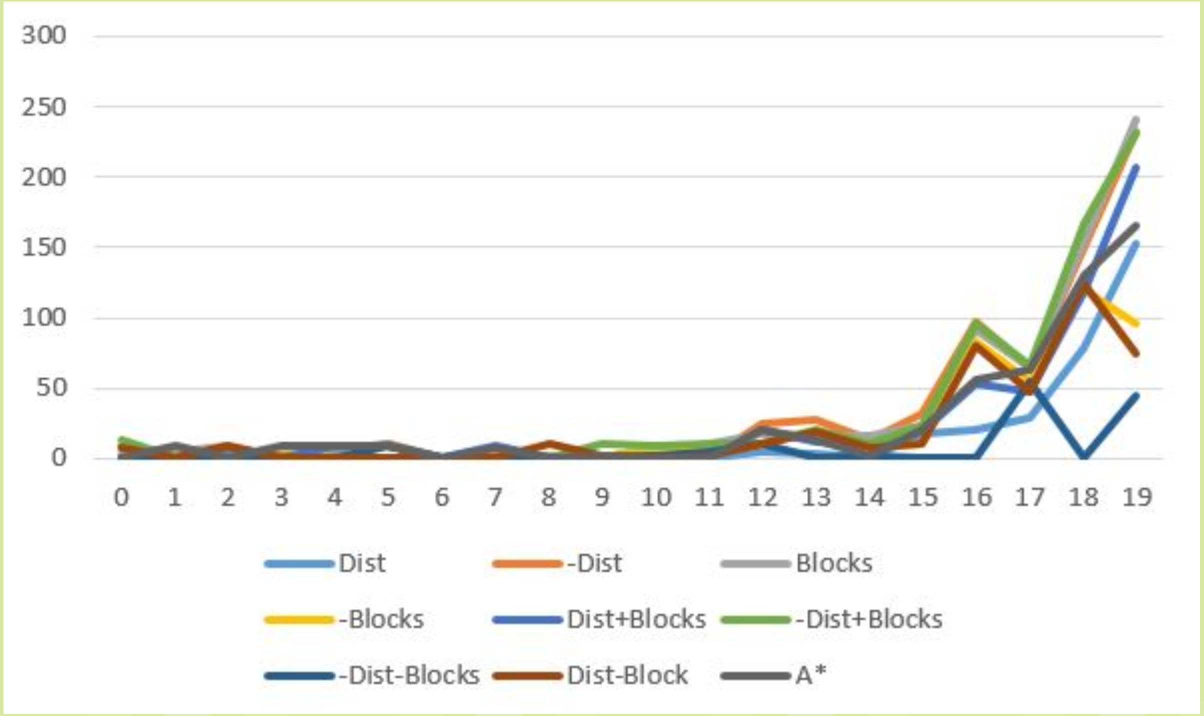
Dist(s)+Crossed(s)

-Dist(s)+Crossed(s)

-Dist(s)-Crossed(s)

Dist(s)-Crossed(s)

A*(s)



Average of 2 analysis, Y-Axis: Time in milliseconds, X-Axis: Tested Heuristic