Inf-2101 Algorithms
Assignment 2
- You are likely to be eaten by a Grue!

Håvard Mathisen

October 4, 2012

Abstract

In the film "Labyrinth" Sarah (Jennifer Cornell) has to search through a maze to find her little brother, motivated by this problem we will in this paper present a method for searching through a maze using two recursive algorithms. A rectangle search algorithm and the depth-first search algorithm. We use the algorithm developed to play three games where we have to search through a maze. By studying what the algorithm does, we come up with some optimalizations that could have been used to improve the algorithm.

1 Introduction

In this assignment a maze-search algorithm is designed and implemented. The algorithm should be able to find a given tile in a square maze, using four different movement operations, go north, south, east or west. The algorithm is used to solve a labyrinth game, the game comes in three different versions.

1.1 Labyrinth

The first game called labyrinth has five operations: go north, south, west or east or look around. The four movement operations takes 3 minutes each and you have to complete the game before 13 hours have passed (780 minutes). The look around operation reviles what the 4 adjacent tiles contain, and does not cost any time. The point with the game is to find Toby.

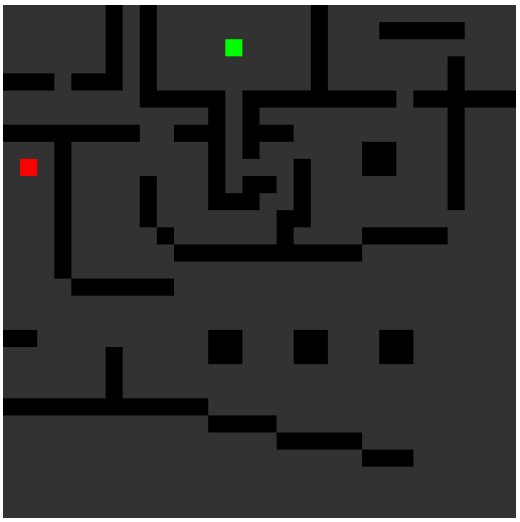

Figure 1: Labyrinth, the green tile is the player, the red tile is Toby and the black tiles are walls

1.2 Trapped Labyrinth

The second game called trapped labyrinth is the same game as labyrinth but it has traps placed around the map, if you go in a trap you lose one of your five lives. The look-around operation now reviles any adjacent traps, but now it costs 1 minute. You can take down an adjacent trap with the disarm operation, but it costs 2 minutes.

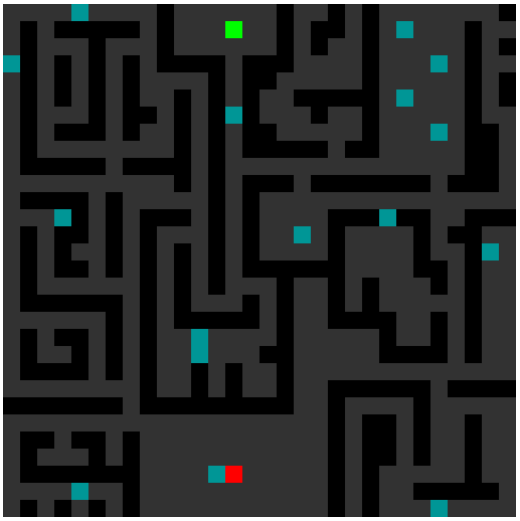

Figure 2: Trapped Labyrinth, the light-blue tiles are traps

1.3 LabyZork

The last game called labyzork is the same as trapped labyrinth but there are monsters called grues around the map, if a grue finds you it will chew on you so that you lose one life each turn until you light one of your 11 matches using the light match operation which costs one minute.

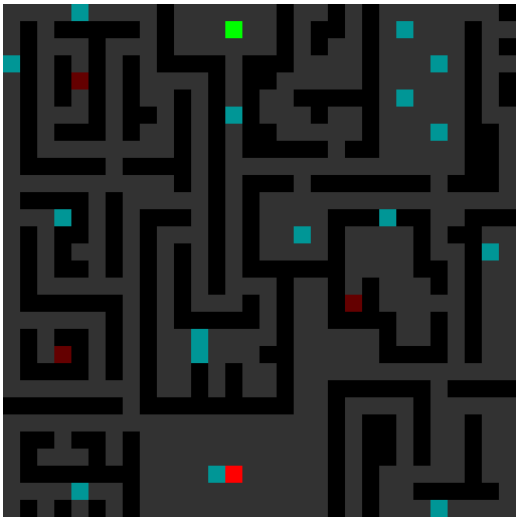

Figure 3: LabyZork, the dark-red tiles are grues

2 Design

The algorithm consists of two parts, the first part searches out a rectangle around a given point and sets it as a node in a graph, the second part of the algorithm is a depth first search, it takes a tile around the rectangle found goes to that tile and starts to search out a new rectangle.

2.1 Graph representation

To represent the graph we keep a two-dimensional map over all the tiles we have information about. The map contains different integers depending on what is known about the tile,

0. The contents of the tile is unknown.
1. The tile is a normal tile.
2. The tile is a wall.
3. The tile is a trap.
4. The tile is Toby.
5. The tile is a grue.
6. The tile is a normal tile, belonging to the first node.
7. The tile is a normal tile, belonging to the second node.
8. ...

2.2 Rectangle find

To search out a rectangle we use a modified spiral search algorithm. It starts searching out a rectangle by spiraling out from the center, when it hits an edge in the rectangle it reverses search direction.

2.3 Depth first search

When we have have found a rectangle we start checking which tiles around that rectangle we have not visited, if we find a tile we have not visited we start running the rectangle find algorithm on that tile, when the rectangle find algorithm is done we recursively run the depth first search on that square.

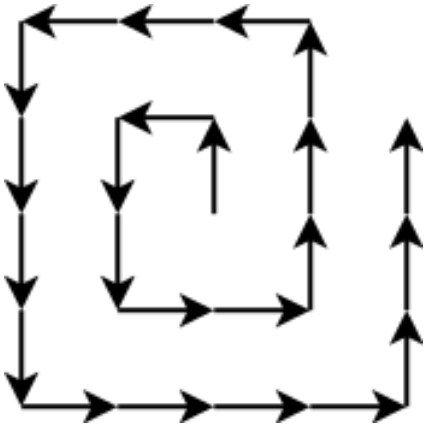

Figure 4: Spiral-search algorithm

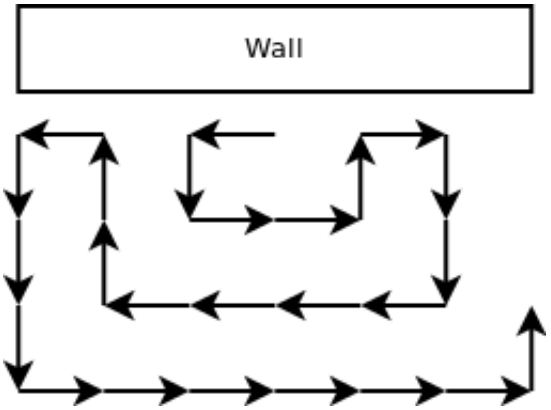

Figure 5: Spiral-search reversing direction

2.4 Trapped Labyrinth

To handle the traps the algorithm initially treats traps as walls, if the depth first search algorithm has searched all tiles not being traps it starts going through traps, removing them first if the player only have one life left.

2.5 LabyZork

If the player steps into a grue, the algorithm immediately lights a match to remove the grue. Note that it takes more time to remove a trap than to light a match, so the algorithm tries to use its lives on removing traps ratter than to remove attached grues.

3 Implementation

Implementation of the algorithms are done using recursive methods in an algorithm class. The class also contains methods for going to a given tile within a rectangle, for going in any direction, for looking at the adjacent tiles and storing what has been seen. There is also a class for the map that contain the graph-representation.

4 Discussion

The implementation is tested to run correct. There are many optimizations that could have been implemented.

4.1 Optimizing Rectangle-search

The implementation could have used the look operation in a better way, for-example the rectangle search algorithm could have used a wider search path to reduce the time needed to search out a rectangle, often the rectangle algorithm also search places where the contents of the tiles is already known.

4.2 Optimizing DFS

The depth-first algorithm could have found a shorter path for the player to go backwards to the next place the find-rectangle algorithm is used. This could have been implemented by making a weighted graph and finding the shortest path between the current node and the node we are going to.

5 Conclusion

The algorithm is tested to complete the run on all three different games, even if the algorithm could have been optimized it finds Toby on the test-maps within the test time.