

---

***Faculty of Science, Engineering and Technology***

**COS30019 – Introduction to AI**

**Assignment-1**

**Tree Based Search**

**Option B – Robot Navigation Problem**

***Prepared by***

**Abdul Hamid Mahi**

**Student ID: 103521410**

# Table of Contents

<b>Table of Contents .....</b>	<b>3</b>
<b>1. Instructions.....</b>	<b>3</b>
<b>2. Introduction.....</b>	<b>3</b>
<b>3. Search Algorithms .....</b>	<b>5</b>
<b>4. Implementation .....</b>	<b>7</b>
<b>5. Features/Bugs/Missing .....</b>	<b>8</b>
<b>6. Conclusion .....</b>	<b>9</b>
<b>7. Acknowledgments/Resources.....</b>	<b>10</b>
<b>8. References .....</b>	<b>11</b>

## 1. Instructions

My whole program was written in C# using the dotNet framework. The recommended IDE for running my program is Visual studio Community, If there is access to Visual Studio 2013, 2015, 2017 or 2019 (version 16.0, version 15.0, version 14.0 and version 12.0 respectively) and if there is .Net (Dot Net) desktop development extension downloaded, one can easily run my program by making a new project in Visual Studio and then adding all the related files under that solution then, going to "Program.cs" file and typing in the path of the txt file where the "loadFile" object is initialized. One can also use "developer command prompt for VS" and then use the path where the files are downloaded to run the program. Upon starting the program, the users are given 4 search options and each option has a number assigned to it. For example, if the user wants to execute a Depth First search, then they have to press 2 as 2 is assigned to that search method. After selecting the number, the user has to press "enter" and then the program will run and show a map which will display the path taken by robot as asterisk symbols (\*), the directions taken and the total count of the nodes that were expanded in order to reach the solution will also be shown below the map. The lesser the count of the expanded node, the more efficient the search algorithm is. The map is simply built used dashes (|), walls represented by (|w), asterisks (\*) represent the path taken by robot, (g) is the goal and (i) is the initial node or starting point. No GUI was used in this program and the txt files which will be provided to be read by the program must follow the progression of "Map dimension" on the first line, then initial state, goal state and finally the position of the walls respectively.

## 2. Introduction

The Robot Navigation problem simply asks us to develop a program which will guide a robot from a random starting point to a specific goal all the while avoiding the walls that will obstruct the movement of the robot. The map on which the robot will navigate on has been divided up into NXM sized grid where N can be considered as the number of rows and M as the number of columns of the Map and both N and M has to be a positive integer. Each cell can be traced by its co-ordinates given by (x,y). The solution to this problem can be found by the implementation of a tree-based search. Some key terminologies related to my program and their meaning are explained below.

**Algorithm:** A process or set of rules to be followed in calculations or other problem-solving operations, especially by a computer.

**Graph Data Structure:** A graph is a non-linear data structure which contains a collection of nodes that contain data and are linked to other nodes via edges. Here any node can be connected to any edge and there is no hierarchical relation between the nodes.

**Node:** A node or vertex is a single unique value.

**Edge:** An Edge represents the connection between two nodes.

**Adjacency:** A vertex is said to be adjacent to another vertex if there is an edge connecting them.

**Path:** A sequence of edges that allows you to go from one specific vertex to another is called a path.

**Tree:** A tree is an undirected, linked, and acyclic graph in graph theory. In other terms, a tree is a linked network that does not include even a single cycle. A tree is a graphical representation of hierarchical organisation. Tree pieces are referred to as nodes, while the tree's edges are referred to as branches.

**Adjacency Matrix:** A 2D Array where we create a row and a column for every node and when two nodes have an edge or connection, we add a 1 at the point they intersect.

**Breadth First Search:** Breadth-first search, also known as BFS, finds shortest paths from a given source vertex to all other vertices, in terms of the number of edges in the paths. A Breadth First Search explores the nodes in a layered fashion. It does this by maintaining a queue of which node it should visit first. It starts at the tree root and explores all nodes at the present depth prior to moving on to the nodes at the next depth level.

**Depth First Search:** Depth First Search (DFS) algorithm traverses a graph in a depth ward motion and uses a stack to remember to get the next vertex to start a search, when a dead end occurs in any iteration.

**Greedy Approach:** A greedy algorithm is a method of solving problems that selects the best alternative available at the time. It is unconcerned with whether the current best outcome will result in the ultimate best result.

**Informed Search:** Informed Search: Algorithms in Informed Search include knowledge about the desired state, which aids in more effective searching. A function that assesses how near a state is to the desired state obtains this information.

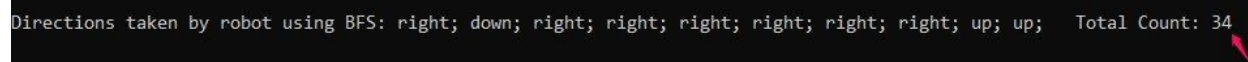
**Uninformed Search:** Uninformed search is a type of general-purpose search method that uses brute force to get results. Because uninformed search algorithms have no extra knowledge about the state or search space other than how to traverse the tree, it is sometimes referred to as blind search.

### 3. Search Algorithms

There are Four search algorithms implemented in my program to tackle the provided robot navigation problem. I have used two informed search methods which are: (i) The A Star Search, and (ii) Greedy Best First Search. The other two are uninformed search methods (i) Breadth First Search, and (ii) Depth First Search.

The efficiency of search methods used can be measured by observing how many nodes they expand in order to reach the goal state. In my code I have made a list called “expandedNodeList” which records the nodes that gets expanded. By returning the count of the total number of entries in this list one can compare the efficiency of the search algorithms. The lower the number of entries the lesser is the time required for the execution of the program.

#### (i). BFS ( Worst method)

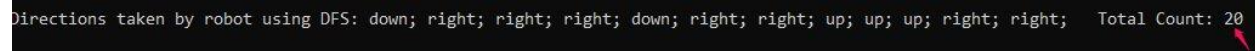


```
Directions taken by robot using BFS: right; down; right; right; right; right; right; right; up; up; Total Count: 34
```

Fig-1: Screenshot of Node expansion count from program terminal (using BFS)

For BFS we see that the greatest number of nodes were expanded in order to reach the goal state. This is because searching all the children's nodes before moving towards the grandchildren nodes is inefficient here since the number of columns here in the grid here is 11 opposed to the number of rows which is only 5 and since the goal state lies in the 3<sup>rd</sup> row it would take longer to find the goal following the layer-by-layer search executed by BFS.

### (ii). DFS (2<sup>nd</sup> best method)

A screenshot of a program terminal with a black background and white text. The text reads: "Directions taken by robot using DFS: down; right; right; right; down; right; right; up; up; up; right; right; Total Count: 20". A red arrow points to the number 20 at the end of the line.

```
Directions taken by robot using DFS: down; right; right; right; down; right; right; up; up; up; right; right; Total Count: 20
```

Fig-2: Screenshot of Node expansion count from program terminal (using DFS)

By executing the DFS method we see that the number of nodes expanded significantly reduces and goes down to only 20. This is because the goal state is relatively deeply positioned in the grid and since DFS expands the parents nodes until the very child node is reached the solution is found much faster compared to that of BFS.

### (iii). A Star (3<sup>rd</sup> best method)

A screenshot of a program terminal with a black background and white text. The text reads: "Directions taken by robot using AStar: right; down; right; right; right; right; right; up; right; up; Total Count: 23". A red arrow points to the number 23 at the end of the line.

```
Directions taken by robot using AStar: right; down; right; right; right; right; right; up; right; up; Total Count: 23
```

Fig-3: Screenshot of Node expansion count from program terminal (using A star)

Even though A Star search is generally relatively faster, Depth First search may outrun A star if the goal node is on the first few expanded branch and that is exactly the case here and that is why A Star is slightly inefficient compared to DFS.

### (iv). GBFS (Best method)

A screenshot of a program terminal with a black background and white text. The text reads: "Directions taken by robot using Greedy Best First: right; down; right; right; right; right; right; up; right; up; Total Count: 12". A red arrow points to the number 12 at the end of the line.

```
Directions taken by robot using Greedy Best First: right; down; right; right; right; right; right; up; right; up; Total Count: 12
```

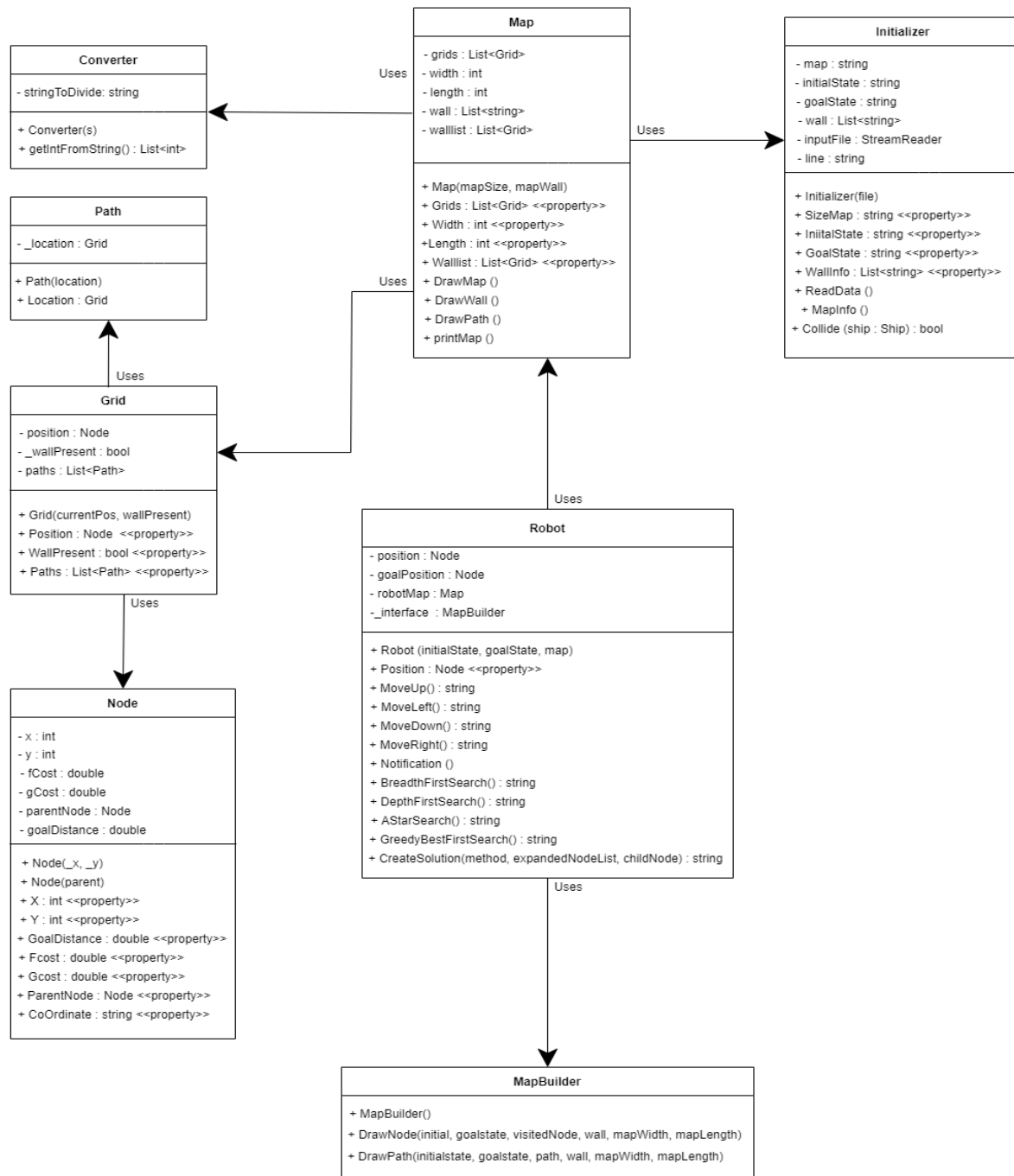
Fig-4: Screenshot of Node expansion count from program terminal (using GBFS)

GBFS seems to be the best search method because it uses the least amount of memory and a short amount of time as it expands only 12 Nodes to reach the goal state.

## 4.Implementation

A Class diagram shown below represents the relation between various classes of my program:

### Robot Navigation Problem (Abdul Hamid Mahi( 103521410))



**Fig 5: UML diagram of robot navigation problem**

My code has 8 Classes and several methods under those classes. Some of these classes will use the objects of other classes to execute some of its functions or methods. The Initializer class reads the provided txt file and then this information is used by my map class to build the map, paths and walls. The Map class uses a method from the class "Converter" called "intFromString()" which converts the string values of the txt file's list into integer values so that they can be used by the methods of the Map class. The methods of the map class like the drawMap(), drawPath() and drawWall() use this information to virtually build the map. The "PrintMap()" method can be used to check all the possible nodes of the grids and it all returns true for the nodes which contain walls. The Map class also uses the map information used by the methods of the class "Robot". These methods each individually execute the BFS, DFS, A Star and GBFS to find the suitable path for our robot. The Robot class also uses the "MapBuilder" class to show a rough map using dashes and other symbols and it also shows the path taken by a certain searching method which are highlighted by an Asterisk (\*).

## 5. Features/Bugs/Missing

**No Application of CUS1 and CUS2:** Unfortunately, due to ineffective time management and lack of enough research and knowledge I couldn't make any of the two custom search algorithms which I was supposed to do in addition with the DFS and GBFS search algorithms.

**Improper application of Object-Oriented Programming Principles:** In my program I have made several errors when it comes to following object-oriented programming principles. First off all there could have been ample opportunities for usage of inheritance, abstract methods, constructor overloading and other principles but I wasn't able to implement those properly and in the map class I also ended up mixing functional in my object oriented programming structure where I called one method into another method and then those methods into another one. Like how I called the "drawpath" and "drawWall" method in my "drawMap" method.

**No significant research, optimization, or extension of the program:** I couldn't do any extra significant research for my program which I can use to further optimize or develop my program. I have only added a simple program execution run time display which can be used to judge efficiency of certain algorithms. I didn't use a GUI (Graphical User Interface) but I used dashes and dots to make a simple map.



**Insufficient Comments and functionality explanation:** My code has insufficient comments to properly tell users how a certain method gets executed. I have added a few lines on certain methods about why they were added and how they get executed but significant explanation is lacking in my program which might make it difficult for someone to understand how the code executes some of its functions. Although I have given proper explanation about my search algorithms which are explained in the Robot.cs class.

## 6. Research

**(i) Program Execution Time Display:** I have made a simple extension to my program in the program.cs class. I have added an extra library called “System.Diagnostics” and this library allows me to use the methods of a class called “StopWatch”. One of this method called elapsed can be used to find out the run time of a certain search algorithm. This extension allows the users to collect more precise data to compare the efficiency of certain search algorithms.

```

D:\C#\RoboNav\RoboNav\RoboNav\bin\Debug\RoboNav.exe
| | w|w| | | g|w| |w| |
|i|*|w|w| | |*|*|w| |
|*|*|*|*|*|*| | |
| |w| | | | |w| |
| |w|w|w|w| | |w|w| |

Directions taken by robot using Greedy Best First: right; down; right; right; right; right; right; up; right; up; Total Count: 12
Algorithm Execution time: 00:00:00.7412648

```

Fig-6: Program execution time display for efficient data gathering

## 7. Conclusion

In summary, this report explains about the solution of the Robot Navigation problem. My solution program was written in C# and was run on Microsoft Visual Studio Community 2019 Edition (version 16.0). Various terminologies used in the explanation of my code has been briefly introduced and explained. Upon execution my program shows the users 4 different search algorithms the DFS, BFS, A star and GBFS and the user can select which search they want to execute. After execution the users are show a map which is made using dashes and other symbols and below the map the directions taken by the robot and the total number of nodes expanded to reach the solution is also shown. It has been seen after execution of my program that “Greedy Best First Search” is the most efficient approach and “Breadth First Search” is the worst approach which is measured by the number of nodes expanded to reach the goal state. My code has eight classes and various methods inside those classes, all of which are interdependent, and some classes will utilise the objects of other classes to execute some of its functions or methods. There are several flaws in my code, I couldn't follow the Object-oriented Principles properly in my code specially the usage of inheritance, abstract methods, and overloading constructors. I also didn't create any custom search solutions which are the CUS1 and CUS 2 in my code. I also didn't do any research or extend the program's functionalities and I also didn't optimize it. I took help from several resources for several things, and all these have been properly cited and acknowledged. Overall, this whole assignment was a great opportunity for me to practically apply and solidify my understanding of the graph and tree data structures and the concepts associated with these structures.

## 7. Acknowledgements/Resources

1) <https://www.youtube.com/watch?v=QRq6p9s8NVg>

Help with theory on BFS like how the nodes get expanded and the role of a queue data structure in accomplishing this.

2) <https://www.youtube.com/watch?v=oDqjPvD54Ss>

Help with idea on the data structure and implementation of BFS by following the provided pseudocode.

3) <https://www.youtube.com/watch?v=7fujbpJ0LB4>

Help with idea on the data structure and implementation of DFS by following the provided pseudocode.

4) [https://www.tutorialspoint.com/data\\_structures\\_algorithms/depth\\_first\\_traversal.htm](https://www.tutorialspoint.com/data_structures_algorithms/depth_first_traversal.htm)

Help with Theory and Understanding of DFS

5) <https://dotnetcoretutorials.com/2020/07/25/a-search-pathfinding-algorithm-in-c/>

Help with code structuring and implementation of A Star algorithm

6) <https://www.youtube.com/watch?v=qYP6fR8BoxY>

Help with concept of GBFS

7) <https://www.youtube.com/watch?v=dv1m3L6QXWs>

Help with idea on the data structure and implementation of GBFS by following the provided pseudocode.

## 8. References

[1] “Graph Theory Tree and Forest - javatpoint,” *www.javatpoint.com*, 2021.  
<https://www.javatpoint.com/graph-theory-tree-and-forest> (accessed Apr. 23, 2022).

[2] “Uninformed Search Algorithms - Javatpoint,” *www.javatpoint.com*, 2021.  
<https://www.javatpoint.com/ai-uninformed-search-algorithms> (accessed Apr. 24, 2022).

[3] “How to calculate the code execution time in C#?,” *Tutorialsteacher.com*, 2016.  
<https://www.tutorialsteacher.com/articles/how-to-calculate-code-execution-time-in-csharp>  
(accessed Apr. 24, 2022).

[4] “Greedy Algorithm,” *Programiz.com*, 2022. <https://www.programiz.com/dsa/greedy-algorithm> (accessed Apr. 24, 2022).

- [5] S. Lague, “A\* Pathfinding (E01: algorithm explanation),” *YouTube*. Dec. 16, 2014. Accessed: Apr. 24, 2022. [YouTube Video]. Available: <https://www.youtube.com/watch?v=-L-WgKMFuhE&t=422s>
- [6] thaihoangcfc, “robot-navigation/ConsoleApp1 at master · thaihoangcfc/robot-navigation,” *GitHub*, 2022. <https://github.com/thaihoangcfc/robot-navigation/tree/master/ConsoleApp1> (accessed Apr. 24, 2022).
- [7] benedictz, “Robot-Navigation-Assignment/Assignment1/SearchMethods at main · benedictz/Robot-Navigation-Assignment,” *GitHub*, 2022. <https://github.com/benedictz/Robot-Navigation-Assignment/tree/main/Assignment1/SearchMethods> (accessed Apr. 24, 2022).
- [8] Reducible, “Depth First Search (DFS) Explained: Algorithm, Examples, and Code,” *YouTube*. Jul. 05, 2020. Accessed: Apr. 24, 2022. [YouTube Video]. Available: <https://www.youtube.com/watch?v=PMMc4VsIacU>
- [9] “Data Structure - Depth First Traversal,” *Tutorialspoint.com*, 2022. [https://www.tutorialspoint.com/data\\_structures\\_algorithms/depth\\_first\\_traversal.htm](https://www.tutorialspoint.com/data_structures_algorithms/depth_first_traversal.htm) (accessed Apr. 24, 2022).