Part A – Tree Based Search

- **Due** *11:59pm Sunday 13 April 2025 (Week 6)*
- **Contributes** 40% to Assignment 2 result.

Part A of Assignment 2 (2A) requires you to work as part of your group to get working tree based search algorithms.

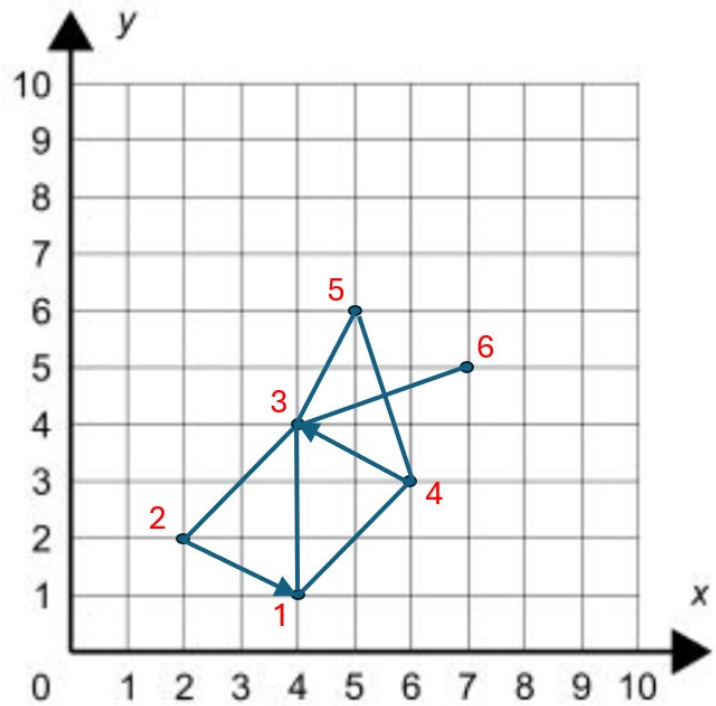
You need to implement tree-based search algorithms in software (from scratch) to search for solutions to the ***Route Finding*** problem. Both ***informed*** and ***uninformed*** methods will be required. You will also need to do some self-learning to learn several search methods (not covered in the lectures).

Implementation

You are encouraged to implement your software using Python. If you prefer to implement your assignment in one of the alternative languages Java, C++ or C#, you need to discuss with your tutor to seek their permission. **You must gain permission from the convenor before using anything else.** Assignment work will be tested on a standard **Microsoft Windows 10** system.

The Route Finding Problem

In this problem, our agent is tasked with finding optimal paths (i.e., lowest cost) between from an Origin node (O) to some Destination (D) nodes on a 2D graph. For example, the following graph represents one such problem:



In the above graph, some edges with an arrow indicate that they can only be traversed in one direction. For instance, one can traverse directly from Node 2 to Node 1 but one **cannot traverse directly** from Node 1 to Node 2. The above *graph* (also known as a *problem*) can be expressed by the following specification:

Nodes:
1: (4,1)
2: (2,2)
3: (4,4)
4: (6,3)
5: (5,6)

```
6: (7,5)
Edges:
(2,1): 4
(3,1): 5
(1,3): 5
(2,3): 4
(3,2): 5
(4,1): 6
(1,4): 6
(4,3): 5
(3,5): 6
(5,3): 6
(4,5): 7
(5,4): 8
(6,3): 7
(3,6): 7
Origin:
2
Destinations:
5; 4
```

File Format: The problems are stored in simple text files with the following format:

- A set of nodes following the keyword **"Nodes:"** ; each line following this keyword specifies a node and its coordinates; for example, **1: (4,1)** specifies a node **1** whose coordinate is **(4,1)** with **4** being the x-coordinate and **1** being the y-coordinate of this node;
- A set of edges connecting the nodes following the keyword **"Edges:"**; each line following this keyword specifies an edge and its cost; for example **(2,1): 4** specifies an edge allowing you to traverse from node **2** to node **1** with a cost of **4**;
- The origin node following the keyword **"Origin:"**;
- A set of destination nodes following the keyword **"Destinations:".** The destination nodes are separated by semi-colons (;). There can be 1, 2, or multiple nodes in the set of destination nodes.

Search Algorithms

The following describe a number of tree based search algorithms. DFS, BFS, GBFS and AS have been covered in the lectures and the tutorials. CUS1 and CUS2 are two algorithms you may learn by yourself (from the textbook, from the Internet or any other sources).

NOTE 1: The objective is to reach one of the destination nodes.

NOTE 2: When all else is equal, nodes should be expanded according to the ascending order, i.e., from the smaller to the bigger nodes. For instance, when all else are being equal between nodes 4 and 7, Node 4 should be expanded before Node 7. Furthermore, **when all else is equal**, the two nodes N_1 and N_2 **on two different branches** of the search tree should be expanded according to the chronological order: if node N_1 is added BEFORE node N_2 then N_1 is expanded BEFORE node N_2.

Search Strategy	Description	Method
Uninformed		
depth-first search	Select one option, try it, go back when there are no more options	DFS
breadth-first search	Expand all options one level at a time	BFS
Informed		
greedy best-first	Use only the cost to reach the goal from the current node to evaluate the node	GBFS
A* ("A Star")	Use both the cost to reach the goal from the current node and the cost to reach this node to evaluate the node	AS
Custom		
Your search strategy 1	An **uninformed** method to find a path to reach the goal.	CUS1
Your search strategy 2	An **informed** method to find a shortest path (with least moves) to reach the goal.	CUS2

Command Line Operation

Your program needs to operate from a DOS command-line interface to support batch testing. A DOS command-line interface can be brought up in Windows 7/8/10 by typing **cmd** into the search box at the ***Start*** button. However, please ensure that your program works on Windows 10 because we will be testing your program on Windows 10. This can be accomplished with a simple DOS .bat (batch) file if needed. Below are the three different arguments formats you need to support. Note the unique argument count for each.

```
C:\Assignments> search <filename> <method>
```

where **`search`** is your .exe file or a .bat (batch) file that calls your program with the parameters.

(if you program in Python, we can accept Python scripts and execute your scripts using the following command from the CLI:

```
C:\Assignments> python search.py <filename> <method> )
```

When a goal can be reached, standard output needs to be in the following format:

```
filename method
goal number_of_nodes
path
```

where **`goal`** is the goal node your search method reached and **`number_of_nodes`** is the number of nodes your program has created when perform this search strategy, and **`path`** is a sequence of moves in the solution that brings you from the start-configuration to the end-configuration. Line breaks are ignored (so use them if you want to).

Report file

You must also include a report which has to be either in Microsoft Word or in PDF whose name is your Team ID (for example, 123.PDF) containing your report. The aim of this report is for you to summarise your understanding of the problem and the algorithms you have used to solve the problem. We are most interested in the insights you have gained, especially those **using data obtained by running your software**.

Report Details: The report must be between 8 and 10 pages (excluding cover page and TOC).

- **Cover page:** including details of all group members (i.e., full names and student IDs), signed by all members of the group.
- **Table of contents (TOC)**.
- **Instructions:** Basic instructions of how to use your program. You can also include a **note** containing anything else you want to tell the marker, such as how to use the GUI version of your program, and something particular about your implementation.
- **Introduction:** Introduce the *Route Finding Problem*, basic graph and tree concepts and concise information about the algorithms the team have implemented.
- **Features/Bugs/Missing:** Include a list of the features you have implemented. Clearly state if a required feature has not been implemented. Failure to do this will result in penalties. Include a list of any known bugs. Also, anything else you want to tell the marker, such as how to use the GUI version of your program, and something particular about your implementation.
- **Testing:** Provide an overview of the test cases you have created to test your program (either manually or automatically). Report the results of testing your program.
- **Insights:** Present and discuss the qualities of the search algorithms used in your assignment. Which algorithms are better and why? **Use data collected to support your points.**
- **Research (if applicable):** If you managed to do some additional research to improve the program in some ways, please report it here.
- **Conclusion:** Conclude with a discussion about the best type of search algorithm you would use for this type of problem. Include thoughts about how you could improve performance.
- **Acknowledgements/Resources:** Include in your report a list of the resources you have used to create your work. A simple list of URL's is not enough. Include with each entry a basic description of how the person or website assisted you in your work.
- **References:** Cite the sources you referred to in your Assignment (implementation, report, etc.)

Tips:

- All figures and tables need to be properly captioned with sensible descriptions.
- Report presentation should include header/footer information (pages numbers, etc.)

Marking Scheme:

Requirements (or equivalent sections)	***Mark***
For each of the 6 methods Depth-First Search (DFS), Breadth-First Search (BFS), greedy best-first search (GBFS), A* (AS), CUS1 and CUS2, if you can get it work well	**10(x6)**
Testing: At least 10 test cases have been created to cover different problem scenarios. Test results have been checked and documented.	**10**
Report: Clear and provide sufficient information about the problem and your algorithm/solution AND the insights you have obtained through your program.	**20**
Research: If you show some initiatives in researching about the problem and solutions, or carrying out extensive tests to provide interesting data about the algorithms, or getting some clever optimization, etc. with a well-written **Research** section in the report to discuss and demonstrate these initiatives	**10**
Total	**100**
You need to follow good programming practice (e.g., well-designed, well-structure codes with clear and helpful comments). Failure to do so get penalty.	**Up to -10**

For each individual student: (the maximum mark a student can get for Assignment 2A is 100 and the minimum mark is 0)	***Mark***
Failure to provide in class updates of the assignment progress to your tutor.	**Up to -40**
Failure to contribute to the work performed by the team or to collaborate with other team members to achieve the desired outcomes for the project.	**Up to -80**

An idea for research initiatives

- Can you modify your program so that the robot will visit ALL destination nodes with the SHORTEST path? Please include in your report the challenges you had to overcome to address this requirement and how you solved them.
- For other ideas, please feel free to do the research yourselves and discuss with your tutors to get their approval first before doing the research component.