Introduction

This project demonstrates algorithms that are used to transverse through graph by choosing node which could be based on the cost functions. The implementation of the algorithms was carried out using the python programming language.

A picture containing diagram

Description automatically generated

## Depth first search (DFS) Algorithm

In this project, we first apply the depth first search (DFS) algorithm which is an algorithm for exploring traversal tree or graph data. The algorithm involves searching for node to explore by starting from the root node and all branches recursively.

The procedure adopted for the depth first search algorithm is explained as follows:

1. Create an array that represents the road map. Each array index represents branches to visit
2. Identify the first nodes to be visited
3. Create recursive function to check node that has not been explored by traversing through neighbor nodes

The path obtained after applying the depth first search algorithm on the road map is describe below:

**A-B-S-K-Y-D-T-G-U-J-P-H-E-L-Q-Z-M-N-X-F**

# Breadth First Search (BFS) Algorithm

The objective of the BFS algorithm is similar to the DFS algorithm which is to travel through all the nodes in the road map. In BFS algorithm, boolean array is created, which is set to True when a node is visited and False by default. The visited nodes are explored by searching through the neighboring points of all branches.

The path obtained after applying the breadth first search algorithm on the road map is describe below:

**A-B-N-S-Q-X-K-H-Z-F-Y-J-E-M-D-P-L-T-U-G**

## Greedy best First Search Algorithm

While the BDFS and DFS explore all nodes without considering the cost function. The greedy best first search algorithm is heuristic search algorithm that uses the cost to decide the shortest adjacent nodes that should be explored. In this algorithm, we store the cost of nodes in a priority queue, which is arranged according to the closet nodes to a adjacent node.

The procedure

1. Create an array that represents the road map while including the cost
2. Create a priority queue
3. Insert the first node into the priority queue
4. Loop until the priority queue is empty
5. For the next node, loop through all the adjacent nodes for the node
6. Check if any adjacent node is empty
7. Add the adjacent node that has not been visited to priority queue
8. Select the adjacent node with the lowest cost

The path obtained after applying the greedy best first search algorithm on the road map is describe below:

**A-B-S-K-Y-D-T-G**

# Uniform Cost Search Algorithm

The uniform cost search algorithm has the same objective as the greedy best first search algorithm, it is an artificial intelligence based algorithm considered as the variant of the Dijikstra’s algorithm. The uniform cost search first inserts the source node into a pirorty queue and then insert other nodes into the priority queue one by one while checking if a node has already been inserted into the queue. Nodes are inserted into the priority queue by choosing the least costly nodes that has not been visited.

The path obtained after applying the greedy best first search algorithm on the road map is describe below:

# A-N-B-S-K-Q-X-H-Y-F-E-Z-J-D-L-T-M-U-G

# A \* Search Algorithm

The A\* search algorithm is an artificial intelligence based algorithm that considered as the best technique used to find the shortest path. It is commonly used to find the shortest path in 2D grid with the obstacles. The A\* search algorithm is the smartest traversal techniques that find shortest path.

The algorithm used to implement the A\* search algorithm is describes as follows

1. Create the heuristic cost function : this function is to obtain the cost value for a node
2. We create the open list, s to store visited nodes and we add the source node
3. We create the closed list, d to store the explored branch nodes
4. Loop until the s is empty
5. Loop through the neighboring nodes for s
6. Compute the total cost function using the heuristic cost function for all nodes
7. Select the next node, which is node with the lowest cost
8. If the parent node is not the open list, s and close list, d

* We add the next node to the list of path found

The path obtained after applying the A\* algorithm on the road map is describes below:

**A-B-S-H-J-D-T-G**