Project Proposal (OS)

A star (A*) Search Algorithm

Introduction

1. What is a Search Algorithm?

Search algorithms are designed to search for or retrieve elements from a data structure, where they are stored. They are essential to access desired elements in a data structure and retrieve them when a need arises. A vital aspect of search algorithms is Path Finding, which is used to find paths that can be taken to traverse from one point to another, by finding the most optimum route.

What is an A* Algorithm?

It is a search algorithm that is used to find the shortest path between an initial and a final point.

It is a handy algorithm that is often used for map traversal to find the shortest path to be taken. A* was initially designed as a graph traversal problem, to help build a robot that can find its own course. It still remains a widely popular algorithm for graph traversal.

It searches for shorter paths first, thus making it an optimal and complete algorithm. An optimal algorithm will find the least cost outcome for a problem, while a complete algorithm finds all the possible outcomes of a problem.

Another aspect that makes A* so powerful is the use of weighted graphs in its implementation. A weighted graph uses numbers to represent the cost of taking each path or course of action. This means that the algorithms can take the path with the least cost, and find the best route in terms of distance and time.

A major drawback of the algorithm is its space and time complexity. It takes a large amount of space to store all possible paths and a lot of time to find them.

Methodology: Combinatorial optimization

Mathematical Optimization that consists of finding an optimal object from a finite set of objects, where the set of feasible solutions is discrete or can be reduced to a discrete set. Typical combinatorial optimization problems are the travelling salesman problem ("TSP") and the minimum spanning tree ("MST").

Initial condition - we create two lists - Open List and Closed List.

Now, the following steps need to be implemented -

- The open list must be initialized.
- Put the starting node on the open list (leave its f at zero). Initialize the closed list.
- Follow the steps until the open list is non-empty:

1. Find the node with the least f on the open list and name it “q”.
2. Remove Q from the open list.
3. Produce q's eight descendants and set q as their parent.
4. For every descendant:

i) If finding a successor is the goal, cease looking

ii)Else, calculate successor.

iii) Skip this successor if a node in the OPEN list with the same location as it but a lower f value than the successor is present.

iv) Skip the successor if there is a node in the CLOSED list with the same position as the successor but a lower f value; otherwise, add the node to the open list end (for loop).

- Push Q into the closed list and end the while loop.

  All graphs have different nodes or points which the algorithm has to take, to reach the final node. The paths between these nodes all have a numerical value, which is considered as the weight of the path. The total of all paths transverse gives you the cost of that route.