EXPERIMENT 5

Postlab

1. Explain the Time Complexity of the A* Algorithm.

2. What are the limitations of A* Algorithm?

3. Discuss A*, BFS, DFS and Dijkstra's algorithm in detail with examples.

Name : Namrata G. Joshi
ROLL NO: 9545
TE COMPS A

DATE

EXPT-5

(1) Explain the time complexity of A* algorithms?
The time complexity of A* algorithm depends on the heuristic accuracy and search space's size. In the worst-case scenario, it can be exponential. However with an admissible and consistent heuristic A* guarantees finding the optimal solution efficiently

(2) What are the limitations of A* algorithm?
① A* can consume significantly memory, ~~search is~~ especially in large spaces
② Worst case time complexity can be exponential particularly with ineffective heuristics or large search spaces
③ The quality of heuristic heavily influence A* efficiency and optimality
④ A* may encounter scenarios where it explores scenarios of search space intentionally

(3) Discuss A*, BFS, DFS, dijkstra's algorithm in detail with example
A* algorithm
A* is a widely used informed search algorithm that finds the shortest path from a short node to a goal node in graph. It combines the advantages of Dijkstra's algorithm and greedy best first search by using both the cost to reach a node
Algorithm Initialize an open list and add the start node
While the open list is not empty
select the node with lowest total cost
If the selected node is goal, terminate with succe

If the open list becomes empty without reaching the goal, terminates the failure

Example:- Find the shortest path from A to E

(2) Breadth first ~~Search~~ Search

① It is an uninformed search algorithm that systematically explores all neighbours nodes at present depth before moving on to nodes at the next depth level.

Algorithm: (2) Start with an initial node and enqueue it in a equeue

While the queue is not empty

Deque a node from the queue

If the dequeued node is the goal, terminate with successes

Enqueue all visited neighbour nodes of deque. node

If the queue becomes empty without reaching the goal, terminate the node

(3) Depth first search

DFS is an uninformed search algorithm that explores as far as possible along each branch before ~~branch~~ backtracking. It does not guarantee finding the shortest path and can get stuck in deep branches

Start with the initial node and push it into a stack.

While the stack is not empty

Pop a node from stack

If popped node is goal terminate success

Push all unvisited neighbour nodes of popped node of the stack

Dijkstra's Algorithm