AI - ASSIGNMENT 1 REPORT

Author :- SHEKHAR SHRIVAS

I have taken five pairs of nodes mentioned below to analyse each Algorithm on a random connection city graph.

1. Number of nodes in complete graph = 45
2. Number of edges in complete undirected graph = 990
3. Dropout chosen to extract a random graph from a complete graph = 0.92
4. Number of edges in the random connected graph = 152
5. Number of nodes in the random connected graph = 45

I will analyse the same pairs for every Algorithm.

Five pair of nodes are:

1. Delhi to Agra
2. Agra to Bhopal
3. Sri Ganganagar to Lucknow
4. Patna to Kota
5. Baghpat to Rajsamand

1. Analysis for DFS:

Since DFS is the basic uninformed graph traversal algorithm used in computer science and mathematics. It will definitely give a path if the number of nodes in the graph is finite and both the nodes(start & goal) are present in the graph. It finds a path between two pairs of nodes by going in depth of search as much as possible before taking backtrack. If the goal is not found then it will backtrack and move to the next branch of the graph and again start searching for a goal node.

- Delhi to Agra
  Found a path with length 6. Including start and goal nodes.

  Path:- Delhi -> Bhagalpur -> Belagavi -> Gaya -> Bikaner -> Agra

- Agra to Bhopal
  Found a path with length 8. Including start and goal nodes.

  Path:- Agra -> Bundi -> Patna -> Baghpat -> Lucknow -> Ghazipur -> Faridabad -> Bhopal

- Sri Ganganagar to Lucknow
  Found a path with length 3. Including start and goal nodes.

  Path:- Sri Ganganagar -> Baghpat -> Lucknow

- Patna to Kota
  Found a path with length 27. Including start and goal nodes.

  Path:- Patna -> Baghpat -> Lucknow -> Ghazipur -> Faridabad -> Bhopal -> Daudnagar -> Sitamarhi -> Delhi -> Bhagalpur -> Belagavi -> Gaya -> Bikaner -> Agra -> Bundi -> Jehanabad -> Arrah -> Raebareli -> Madhepura -> Mahoba -> Jaipur -> Pakur -> Aligarh -> Durgapur -> Mirzapur -> Nawada -> Kota

- Baghpat to Rajsamand
  Found a path with length 13. Including start and goal nodes.

  Path:- Baghpat -> Lucknow -> Ghazipur -> Faridabad -> Bhopal -> Daudnagar -> Sitamarhi -> Delhi -> Bhagalpur -> Belagavi -> Gaya -> Bikaner -> Rajsamand

In case 4 it is not giving the optimal path because it goes in depth of the one branch and stops once it finds the goal node.

Pros:
1. Memory-efficient:- DFS takes less memory as compared to BFS because it goes as deep as possible in the branch of the start node and stops searching once the goal node is found.
2. Maze Solving:- DFS frequently used in Maze Solving. By using DFS we can solve Maze in an efficient way.

Cons:
1. It may not give an optimal path because another shortest path may exist. In case of Patna to Kota this is not an optimal path because the shortest exists.
2. Completeness:- DFS is not guaranteed to find a solution if one exists. In an infinite graph or graph with cycles. it can get stuck in an infinite loop and may not explore all possible paths.

2. Analysis for BFS:

Since BFS is the basic uninformed graph traversal algorithm. It can guaranteed give an optimal path if both the nodes(star, goal) exist in the graph. It searches the graph level by level; this is why it is called Breadth-first search.

- Delhi to Agra
  Found a path with length 3. Including start and goal nodes.

  Path:- Delhi -> Bikaner -> Agra

- Agra to Bhopal
  Found a path with length 3. Including start and goal nodes.

Path:- Agra -> Bikaner -> Bhopal

- Sri Ganganagar to Lucknow
  Found a path with length 3. Including start and goal nodes.

  Path:- Sri Ganganagar -> Baghpat -> Lucknow

- Patna to Kota
  Found a path with length 3. Including start and goal nodes.

  Path:- Patna -> Nawada -> Kota

- Baghpat to Rajsamand
  Found a path with length 2. Including start and goal nodes.

  Path:- Baghpat -> Rajsamand

In case 4 it gives an optimal path which has only 3 nodes in the path list but in case of DFS it is not possible because DFS may or may not give an optimal path.

Pros:

1. It is guaranteed to give an optimal path if both the nodes exist in the graph.
2. If there's more than one solution then BFS will give a Minimal one that requires less number of steps. In case 4 in DFS has 27 nodes in the path but in BFS we only have three nodes.

Cons:

1. BFS is not Memory-efficient, especially when the graph is large because it has to store all the potential paths.
2. It is the slow algorithm because it expands all the at each level.

3. Analysis for GBFS:

G-BFS is called the “Greedy” algorithm because it tries to get closer to the goal node after opening every node. In G-BFS we first open the node which has the lowest - evolution function value or lowest heuristic value.

- Delhi to Agra
  Found a path with length 3. Including start and goal nodes.
  Path:- Delhi -> Bikaner -> Agra

- Agra to Bhopal
  Found a path with length 3. Including start and goal nodes.
  Path:- Agra -> Bikaner -> Bhopal

- Sri Ganganagar to Lucknow
  Found a path with length 3. Including start and goal nodes.

Path:- Sri Ganganagar -> Baghpat -> Lucknow

- Patna to Kota
  Found a path with length 3. Including start and goal nodes.
  Path:- Patna -> Nawada -> Kota

- Baghpat to Rajsamand
  Found a path with length 2. Including start and goal nodes.
  Path:- Baghpat -> Rajsamand

Unfortunately I am not able to find any pair of nodes which leads to dead ends in G-BFS.

Pros:
1. Fast and Efficient algorithm because the only node opened with the lowest heuristic value.
2. Greedy algo tries to get closer to the goal node at every step.
3. Minimal algorithm because it does not open the node which is not in the solution path.

Cons:
1. G-BFS is not a complete algorithm because it may lead to dead ends.
2. G-BFS is not optimal because another shortest path may exist.

4. Analysis for A*:

In A* algorithm first we evaluate the g(n), the route distance from node one to node n and then evaluate the h(n) heuristic value from the n node to goal node and then opened the node which have the lowest f(n) = g(n) + h(n) value.

- Delhi to Agra
  Found a path with length 3. Including start and goal nodes.
  Path:- Delhi -> Bikaner -> Agra

- Agra to Bhopal
  Found a path with length 3. Including start and goal nodes.
  Path:- Agra -> Bikaner -> Bhopal

- Sri Ganganagar to Lucknow
  Found a path with length 3. Including start and goal nodes.
  Path:- Sri Ganganagar -> Baghpat -> Lucknow

- Patna to Kota
  Found a path with length 3. Including start and goal nodes.
  Path:- Patna -> Nawada -> Kota

- Baghpat to Rajsamand
  Found a path with length 2. Including start and goal nodes.
  Path:- Baghpat -> Rajsamand

Pros:

1. A* is an optimal algorithm. If both nodes exist in the graph then it will guaranteed an path.
2. A* is a complete algorithm.
3. A* is an admissible algorithm.

Cons:

1. A* is not the Memory-efficient algorithm because it is a backtracking algorithm so it has to keep track of all the potential paths.
2. The time complexity of A* is higher than G-BFS or it is the slower algorithm because it first calculates the f(n) value of all nodes at each level.

- I have taken help from ChatGPT for G-BFS & A* algorithms.