National University of Computer and Emerging Sciences

Lab Manual 04
Artificial Intelligence Lab

Instructor: Mariam Nasim

1. What is Uniform Cost search

Uniform Cost Search is an algorithm used to move around a directed weighted search space to go from a start node to one of the ending nodes with a minimum cumulative cost. This search is an uninformed search algorithm since it operates in a brute-force manner, i.e. it does not take the state of the node or search space into consideration. It is used to find the path with the lowest cumulative cost in a weighted graph where nodes are expanded according to their cost of traversal from the root node. This is implemented using a priority queue where lower the cost higher is its priority.

2. What is A-star Algorithm (A*)?

- It is a searching 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.

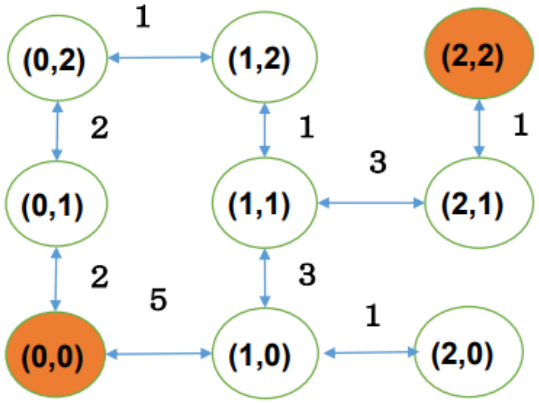
3. A* Algorithm Flow

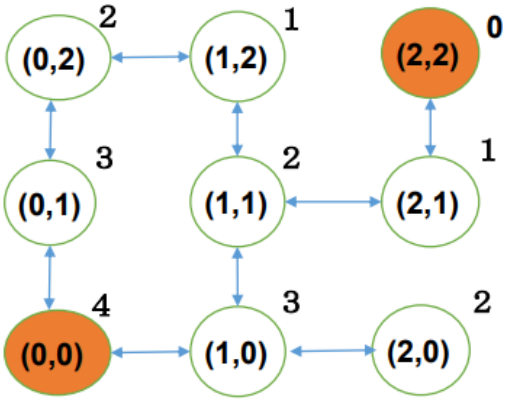
Step 1: Put the initial node x0 and its cost F(x0)=H(x0) to the open list.

Step 2: Get a node x from the top of the open list. If the open list is empty, stop with failure. If x is the target node, stop with success.

Step 3: Expand x to get a set S of child nodes. Put x to the closed list.

Step 4: For each x’ in S, find its cost – If x’ is in the closed list but the new cost is smaller than the old one, move x’ to the open list and update the edge (x,x’) and the cost. – Else, if x’ is in the open list, but the new cost is smaller than the old one, update the edge (x,x’) and the cost. – Else (if x’ is not in the open list nor in the closed list), put x’ along with the edge (x,x’) and the cost F to the open list.





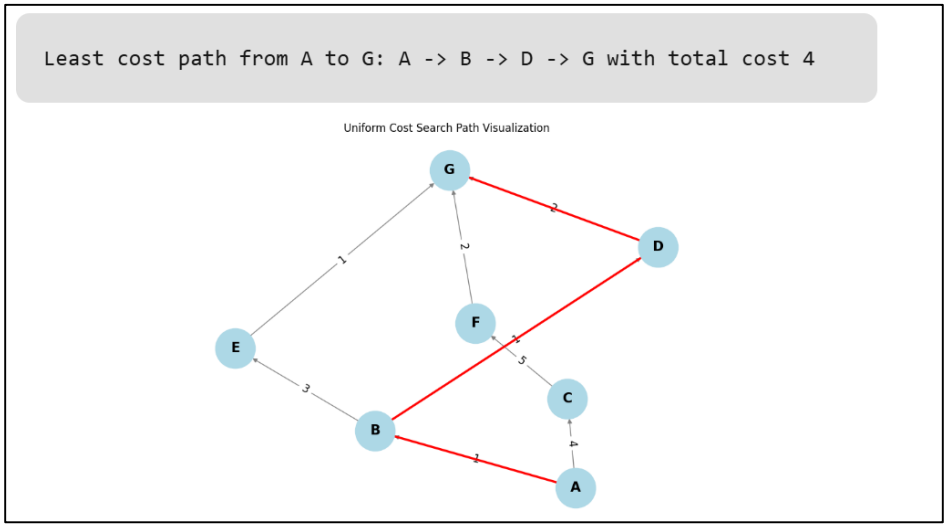
Steps	Open List	Closed List
0	{(0,0), 4}	--
1	{(0,1),5} {(1,0),8}	{(0,0),4}
2	{(0,2),6} {(1,0),8}	{(0,0),4} {(0,1),5}
3	{(1,2),6} {(1,0),8}	{(0,0),4} {(0,1),5} {(0,2),6}
4	{(1,0),8} {(1,1),8}	{(0,0),4} {(0,1),5} {(0,2),6} {(1,2),6}}
5	{(1,1),8} {(2,0),8}	{(0,0),4} {(0,1),5} {(0,2),6} {(1,2),6}} {(1,0),8}
6	{(2,0),8} {(2,1),10}	{(0,0),4} {(0,1),5} {(0,2),6} {(1,2),6}} {(1,0),8} {(1,1),8}
7	{(2,1),10}	{(0,0),4} {(0,1),5} {(0,2),6} {(1,2),6}} {(1,0),8} {(1,1),8} {(2,0),8}
8	{(2,2),10}	{(0,0),4} {(0,1),5} {(0,2),6} {(1,2),6}} {(1,0),8} {(1,1),8} {(2,0),8} {(2,1),10}
9	(2,2)=target node	{(0,0),4} {(0,1),5} {(0,2),6} {(1,2),6}} {(1,0),8} {(1,1),8} {(2,0),8} {(2,1),10}

Task 1: Uniform Cost Search

You are working for a logistics company that is responsible for delivering packages across a city. The company needs to optimize delivery routes for its fleet of vehicles, aiming to minimize time taken for each delivery. The delivery routes are represented as a weighted graph, where each node represents a location (such as a warehouse or delivery point), and each edge represents a road with an associated cost (e.g. time taken to reach from A to B).

Your task is to implement the **Uniform Cost Search (UCS)** algorithm to help the logistics company find the shortest path from the warehouse (start node) to a customer's address (goal node). The company wants to ensure that the route selected minimizes the total time.

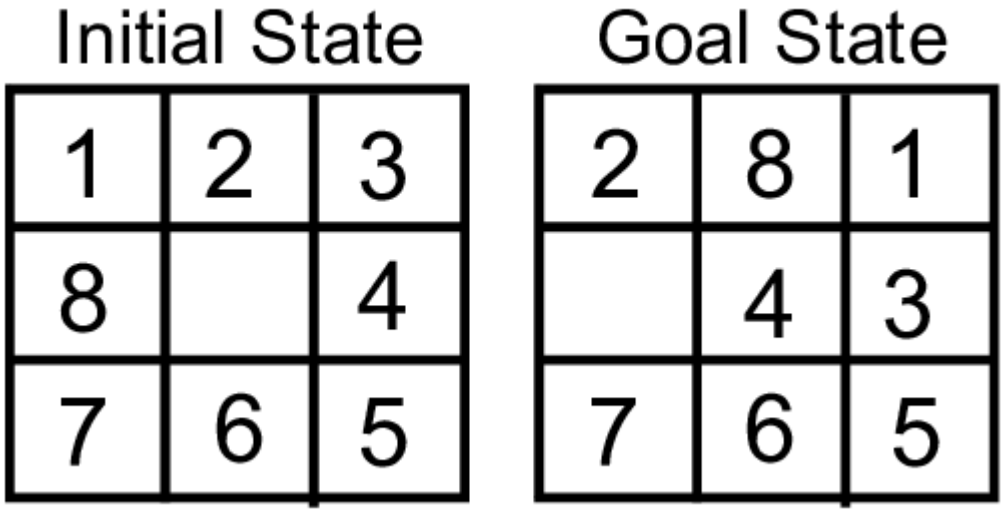
1. **Input:**
    - A **graph** representing the road network, with nodes (locations) and edges (roads with costs).
    - A **start node** (warehouse) and a **goal node** (customer's address).
2. **Output:**
    - The sequence of nodes representing the least costly path.
    - The total cost of this path.
3. **Edge Cases:**
    - Handle scenarios where there is no possible path from the warehouse to the customer.



Task 2: N-Puzzle using A* Algorithm

Exercise: You have to implement the 8-Puzzle problem using A* Algorithm.

N-Puzzle or sliding puzzle is a popular puzzle that consists of N tiles where N can be 8, 15, 24, and so on. In our example N = 8. The puzzle is divided into sqrt(N+1) rows and sqrt(N+1) columns. Eg. 15-Puzzle will have 4 rows and 4 columns and an 8-Puzzle will have 3 rows and 3 columns. The puzzle consists of N tiles and one empty space where the tiles can be moved. Start and Goal configurations (also called state) of the puzzle are provided. You have to solve the puzzle by moving the tiles one by one in the single empty space and thus achieving the Goal configuration.



Rules for solving the puzzle.

Instead of moving the tiles in the empty space, we can visualize moving the empty space in place of the tile, basically swapping the tile with the empty space. The empty space can only move in four directions,

1. Up
2.Down
3. Right or
4. Left

The empty space cannot move diagonally and can take only one step at a time (i.e. move the empty space one position at a time).

You can read more about solving the 8-Puzzle problem here.

Submission Instructions:

- Rename your Jupyter notebook to your “roll number_Name” and download the notebook as .ipynb extension.
- To download the required file, go to File->Download .ipynb
- Only submit the .ipynb file. DO NOT zip or rar your submission file
- Submit this file on Google Classroom under the relevant assignment.
- All outputs should be displayed properly.
- Late submissions will NOT AT ALL be accepted.