CSCI 390, Artificial Intelligence

Programming Assignment 2

In this assignment, we will consider the problem of finding a shortest path for a robot vacuum on an 8x8 grid. The rows are numbered from 1 to 8. The columns are also numbered 1 to 8. Figure 1 illustrates an example of this problem. Here, the start position is (1, 1) and the goal is at (8, 8). Movement of the robot is allowed by one square in any direction including diagonals, similar to the movement of a king in chess. The standard movement cost is 1. To make things more challenging, we set a barrier which occupies certain positions of the grid. Robot moving into any of the barrier positions will get an extremely high cost (i.e., 200).

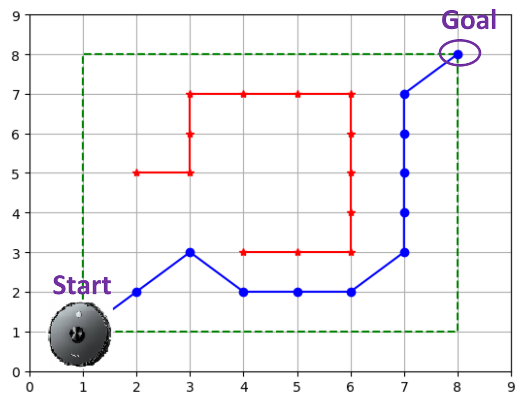

Figure 1: Example of finding a shortest path on an 8x8 grid. Green line denotes the border of the grid. Blue line is the optimal path. Red line is the barrier. In this example, the barrier occupies the positions (2,5), (3,5), (3,6), (3,7), (4,7), (5,7), (6,7), (6,6), (6,5), (6,4), (6,3), (5,3), (4,3).

Accordingly, we implement Uniform-cost search and A* search algorithms to find the shortest path with the lowest cost. Note that there might be multiple optimal solutions with these two algorithms, but they all have the same cost. Figure 2 shows two results obtained by using A* search and UCS search. You can see that two algorithms return different paths. However, the costs are both 11. Normally, UCS expands more nodes than A* does.

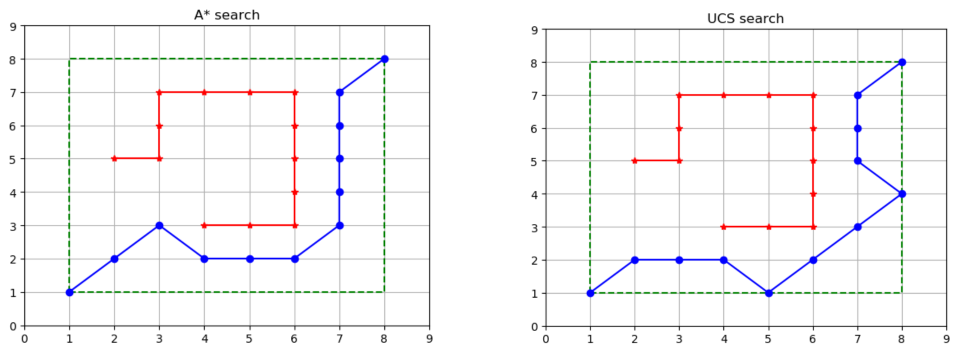

Figure 2: Solutions returned by A* search and UCS search.

For A* search algorithm, because the robot is allowed to move *left, right, up, down,* and *diagonal*, we use Chebysev distance as the heuristic function. Formally, the Chebysev distance between two positions, (x_1,y_1) and (x_2,y_2), on the grid is defined as follows:

$$D_{Chebysev} = max(\lceil x_2 - x_1 \rceil, \lceil y_2 - y_1 \rceil)$$

In summary, the problem of finding the shortest path on the grid is formulated as follows:

- **States**: positions (Cartesian coordinates) of the robot on the grid.
- **Initial state**: a start position of the robot (e.g., (1,1)).
- **Actions**: *left, right, up, down,* and *diagonal* → 8 possible movements.
- **Goal test**: check if the robot is in the goal (e.g., (8,8)).
- **Path cost**: Each standard movement cost is 1 but if the robot enters the barrier point, the cost is 200. Hence, the path cost is the sum of all movement costs.
- **Heuristic (**for A* search**)**: Chebysev distance between the current position of the robot and the goal.

Important Note**:** in the implementation of A* and UCS, you expand the current node to its neighbors and put the neighbors into the fringe, but there might be the case that one of the neighbors was already in the fringe. Then, we need to compare the current path cost with the same neighbor node's previous path cost. If the current path cost is smaller, we will update the cost and the route to the neighbor node. Otherwise, the neighbor will be ignored. Please take a look at the following figure for a better understanding of this case.

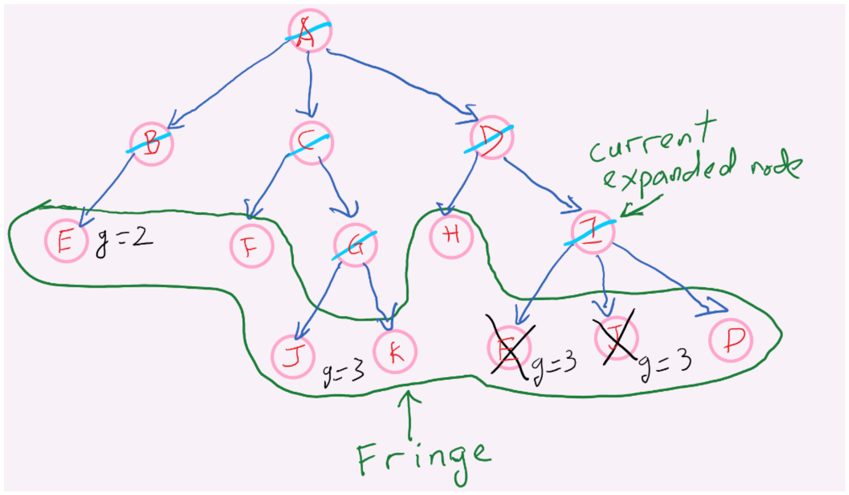

Figure 3: Given the search tree, we expand the current node I to three nodes E, J, P. However, node E and J were already in the fringe. The path cost of A→ D → I → E is greater than that of A → B → E. Meanwhile, the path cost of A→ C → G → J is equal to A→ D→I → J. Therefore, from the current expanded node I, we will not put E and J into the fringe. Note that, in this figure, nodes A, B, C, D, G, I (denoted with blue cross) are expanded and closed.

TASK:

- The code for this assignment includes the following files:
  - `PA2_search.py`: you will edit this file which contains methods for A* an UCS algorithms.
  - `PA2_test.py`: this file is used to visualize the results and test the A* and UCS algorithms.
  - **Files to Edit and Submit:** You will fill in portions of `PA2_search.py` only during the assignment, and submit it. You should submit this file with your code and comments. Please *do not* change the other files in this distribution or submit any of our original files other than this file.
- All the necessary instructions are included in python files that will guide you through the assignment.
- In `PA2 search.py` file, there are ###### START OF YOUR CODE HERE/#### END OF YOUR CODE#### tags denoting the start and end of code sections you should fill out. Take care to not delete or modify these tags. Otherwise, your assignment may not be properly graded.
- Complete `UCsearch()` and `aStarSearch()` functions in the `PA2_search.py` file.
- Although there are multiple solutions with A* and UCS, make sure that you achieve the same cost for these methods.
- To test your implementation, run your code with: `python PA2_test.py`