

COMP3702/COMP7702 Artificial Intelligence

Semester 2, 2020

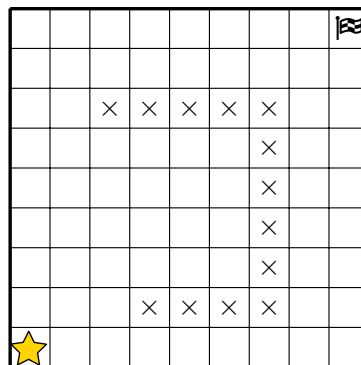
Tutorial 3

Before you begin, please note:

- Tutorial exercises are provided to help you understand the materials discussed in class, and to improve your skills in solving AI problems.
- Tutorial exercises will not be graded. However, you are highly encouraged to do them for your own learning. Moreover, we hope you get the satisfaction from solving these problems.
- The skills you acquire in completing the tutorial exercises will help you complete the assignments.
- You'll get the best learning outcome when you try to solve these exercises on your own first (before your tutorial session), and use your tutorial session to ask about the difficulties you face when trying to solve this set of exercises.

Exercises

Exercise 3.1. Consider the path planning problem on a 9×9 grid world in the figure below. The goal is to move from the star to the flag in as few steps as possible. Crosses indicate obstacles, and attempting to traverse either an obstacle or a boundary wall costs 1 time step and do not move your agent.





Uniform cost

- Develop a state graph representation for this search problem, and develop a `step()` method for finding the next legal steps this problem, i.e. for generating successor nodes (vertices).
- Implement BFS for this problem (using a *FIFO queue*) using your `step()` function.
- Implement iterative-deepening DFS (IDDFS) for this problem using a length-limited *LIFO queue*, and reusing `step()`.
- Compare the performance of BFS and IDDFS in terms of (i) the number of nodes generated, (ii) the number of nodes on the fringe when the search terminates (iii) the number of nodes on the explored set (if there is one) when the search terminates, and (iv) the run time of the algorithm (e.g. in units such as mins:secs). Discuss your findings.

Exercise 3.2. Now consider the path planning problem **with costly moves** on a 9×9 grid world in the figure below, where costs of arriving at a state are indicated on the board and the goal is to move from the star to the flag:

- Run BFS for this problem, reusing your answer from Question 1 (nb. it should not use the costs on the grid).

1	1	1	5	5	5	5	1	
1	1	1	5	5	5	5	1	1
1	1	10	10	10	10	10	1	1
1	1	1	10	10	10	10	1	1
1	1	1	1	1	10	10	1	1
1	1	1	1	1	10	10	1	1
1	1	1	1	10	10	10	1	1
1	1	1	10	10	10	10	1	1
	1	1	1	1	1	1	1	1

Costly moves

- Implement UCS for this problem using a *priority queue*.
- Compare the performance of BFS and UCS in terms of (i) the number of nodes generated, (ii) the number of nodes on the fringe when the search terminates (iii) the cost of the solution path. Discuss how and why these results differ from those for Question 1.
- Now derive an *admissible* heuristic for this path planning problem.
- Using your heuristic, implement A* search for solving this path planning problem.
- Compare the performance of UCS and A* search in terms of (i) the number of nodes generated, (ii) the number of nodes on the fringe when the search terminates (iii) the cost of the solution path. Explain these results.

Exercise 3.3. Implement A* for solving the 8-puzzle problem, assuming the goal of the 8-puzzle agent is to find the solution in as few steps as possible.

- Discuss the heuristics you could use with your classmates.
- Reuse the container in the program you created in the last tutorial, by modifying it to a priority queue. Along with this, revise the program, so as to use the cost from initial to the current node and the heuristics to identify which node to expand next.
- Implement the heuristic as part of your A* search algorithm.

Exercise 3.4. Let h_1 be an *admissible* heuristic, where the lowest value is 0.1 and the highest value is 2.0. Suppose that for any state s :

- $h_2(s) = h_1(s) + 5$,
- $h_3(s) = 2h_1(s)$,
- $h_4(s) = \cos(h_1(s) * \pi)$, and
- $h_5(s) = h_1(s) * |\cos(h_1(s) * \pi)|$.

Answer the following questions, and provide convincing arguments to support your answers:

- Can you guarantee that h_2 be admissible? How about h_3 , h_4 and h_5 ?
- Can we guarantee that A* with heuristic h_2 will generate an optimal path? How about A* with heuristic h_3 , h_4 or h_5 ?