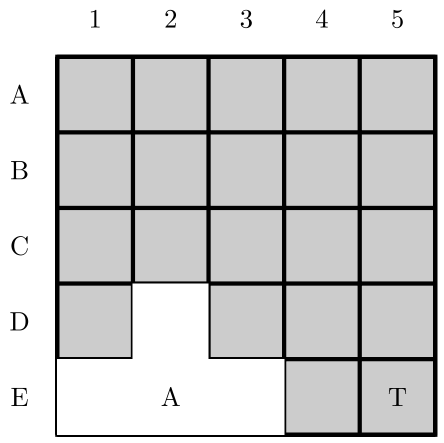
CS440 Report

Yeoun Chan Kim John Strauser Xuanang Wang

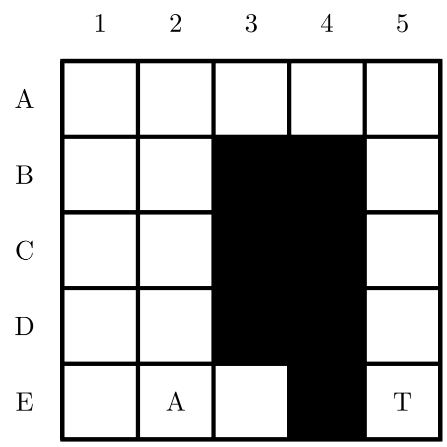
October 13, 2019

Part 1



a)Given the illustration above, Let (a, b) represent a unique state where a ∈ { 1-5 } and b ∈ { A-E }. We want to explain why the first move of the agent A in state (2, E) is to the east (3, E) rather than the north (2, D). Grey represents what agent cannot see and the white represents what agent can. According to the question, T in state (5, E) is the goal state. We calculate the Manhattan distance from A to T, which is 3 in the current situation: h(2, E) = 3. Let g(a, b)(c, d) represents the total distance cost from state (a, b) to state (c, d) in the A* searching path. Then in state (2, E), g(2, E)(2, E) = 0. Therefore, f(2, E) = 0 + 3 = 3. For all the state adjacent to (2, E) which are (1, E), (2, D), (3, E); the f value of each is (1, E): 4 + 1 = 5, (2, D): 4 + 1 = 5, (3, E): 2 + 1 = 3. According to the A* algorithm, we move the agent A to the position where the smallest f value holds. Therefore, the agent would move to the east (3, E) rather than the north(2, D) or (1, E).

b)Let S be the state in the finite gridworld, S(s) denotes the starting state, S(c) denotes the current state and S(g) denotes the goal state. For any S(c), let h(S(c)) be the Manhattan distance from S(c) to S(g), let g(S(c)) be the distance traveled from S(s) to S(c), let S(c') be any adjacent state of S(c) and h(S(c')) g(S(c')) be the corresponding h and g value. According to the A* algorithm, for any state S(c), as long as it's adjacent is not blocked which given value infinity, we calculate the f value using the formula f(c') = h(S(c')) + g(S(c')) and choose the one with the smallest f value. Because of the fact that for each iteration we always choose the $_{(\mathrm{min})}$f(c') to be the next step for the path, as long as one of the possible path from S(s) to S(g) is not blocked, we would guarantee to reach the target. To make the explanation more clear, we consider the illustration below where black cells represent blocked.



Because we are using the greedy approach, the next move of agent A at state (2, E) would be (3, E) (explained in part 1a)). However, when the state moves to (3, E), the dead-end situation occurred. Whenever we meet the situation like this, we would trace back to the parent cell of the dead ended cell, which is (2, E). After this, we would put (3, E) to closed list therefore the agent A would not able to go the (3, E) again even if f(3, E) is the smallest giving the situation the current state is (2, E).

However, if we put another block on (4, A), the agent would traverse from (2, E) vertically to (2, A); then go to (3, A) and backtrace to (2, A) because of the dead end.After that, the agent would go to (1, A) and vertically travel down to (1, E). In this situation, the open list is empty therefore the program returns false.

Part 2

We run Repeated Forward A* break ties in favor of cells with smaller g-values for 30 times in 30 different 101 * 101 mazes. The expanded cell (runtime) for each is (10159, 10, 25194, 24938, 11969, 6033, 5226, 17104, 14849, 528, 1300, 341, 3424, 1623, 19212, 7388, 5365, 40, 1248, 2744, 4735, 15193, 1233, 1274, 6011, 1296, 977, 9868, 3322, 774).

Then we run Repeated Forward A* break ties in favor of cells with larger g-values for 30 times in 30 different 101 * 101 mazes. The expanded cell (runtime) for each is (2619, 8, 1739, 2167, 2967, 410, 1105, 1236, 2448, 110, 243, 151, 1327, 630, 3278, 623, 2014, 29, 414, 548, 1520, 1633, 289, 386, 1417, 294, 762, 1287, 679, 241).

The average for running RFA* in favor of smaller g-value 30 times is approximately 6605. And the average for running RFA* in favor of larger g-value 30 times is approximately 987 .

Part 3

As stated in part 2, the expanded cell (runtime) for RFA* in favor of cells with larger g-values is (2619, 8, 1739, 2167, 2967, 410, 1105, 1236, 2448, 110, 243, 151, 1327, 630, 3278, 623, 2014, 29, 414, 548, 1520, 1633, 289, 386, 1417, 294, 762, 1287, 679, 241).

Then we run Repeated Backward A* in favor of cells with larger g-values for 30 times in the same 30 different 101 * 101 mazes. the expanded cell (runtime) for each is (7454, 6, 8924, 6978, 10492, 1755, 7668, 8889, 7977, 408, 648, 251, 4338, 1681, 16762, 6206, 5607, 25, 1321, 1447, 3738, 5504, 458, 929, 4814, 1579, 742, 7747, 1269, 266).

The average for running RFA* 30 times is approximately 987. And the average for running RBA* 30 times is approximately 4196.

Part 5

As stated in part 2, the expanded cell (runtime) for RFA* in favor of cells with larger g-values is (2619, 8, 1739, 2167, 2967, 410, 1105, 1236, 2448, 110, 243, 151, 1327, 630, 3278, 623, 2014, 29, 414, 548, 1520, 1633, 289, 386, 1417, 294, 762, 1287, 679, 241).

Then we run Adaptive A* in favor of cells with larger g-values for 30 times in the same 30 different 101 * 101 mazes. the expanded cell (runtime) for each is (2671, 8, 1846, 2109, 2958, 429, 1044, 1201, 2383, 212, 268, 149, 1338, 545, 3681, 637, 2079, 17, 413, 556, 1530, 1655, 281, 377, 1441, 297, 767, 1254, 657, 235).

The average for running RFA* 30 times is approximately 987. And the average for running Adaptive A* 30 times is approximately 1101.