Project 1

Patrick Meng and Ethan Chang

Note: We made out own heap for the extra credit

Question 1

A. Since the square to the east is closer to the goal (using a direct path), the heuristic for the eastern square would be lower. Since there is no varying cost, the travel cost for going 1 square north or east will be the same. Therefore A* would pick the block with the better heuristic.

B. The agent would take heuristics into account and always move in the direction of the goal if there is nothing blocking it. Thus it must always reach the goal. If it doesn't, then it will eventually realize there is no goal. If the robot reaches every unblocked tile, it would run out of states in its open list. Then it would stop and say the target is unreachable.

For each planned path, the worst case scenario is if the robot needs to move to every unblocked tile. Similarly, in the worst case the number of moves in each path is the number of unblocked tiles. Therefore, the maximum number of moves the agent needs to reach the target or discover that it doesn't exist is the number of unblocked tiles squared. This is the same for both if the target is reachable and if it isn't because in the worst case when the robot reaches the last unblocked cell it either reaches the goal or realizes there is no goal

Question 2

We found that on average 16904.96 nodes expanded for smaller g after 50 runs. 555.7 nodes expanded on average when favoring larger g. When f is the same, larger g signifies a smaller h. Smaller h is a better rule of thumb because it means the cell is absolutely closer to and in the general direction of the goal and a smaller g does not signify that. Being closer to the goal usually allows the agent to make fewer searches and thus fewer nodes are expanded. Another way to think of this is likening smaller g with Uniform-Cost Search and smaller h with Greedy-Best. Uniform-Cost is more accurate, but Greedy-Best is faster

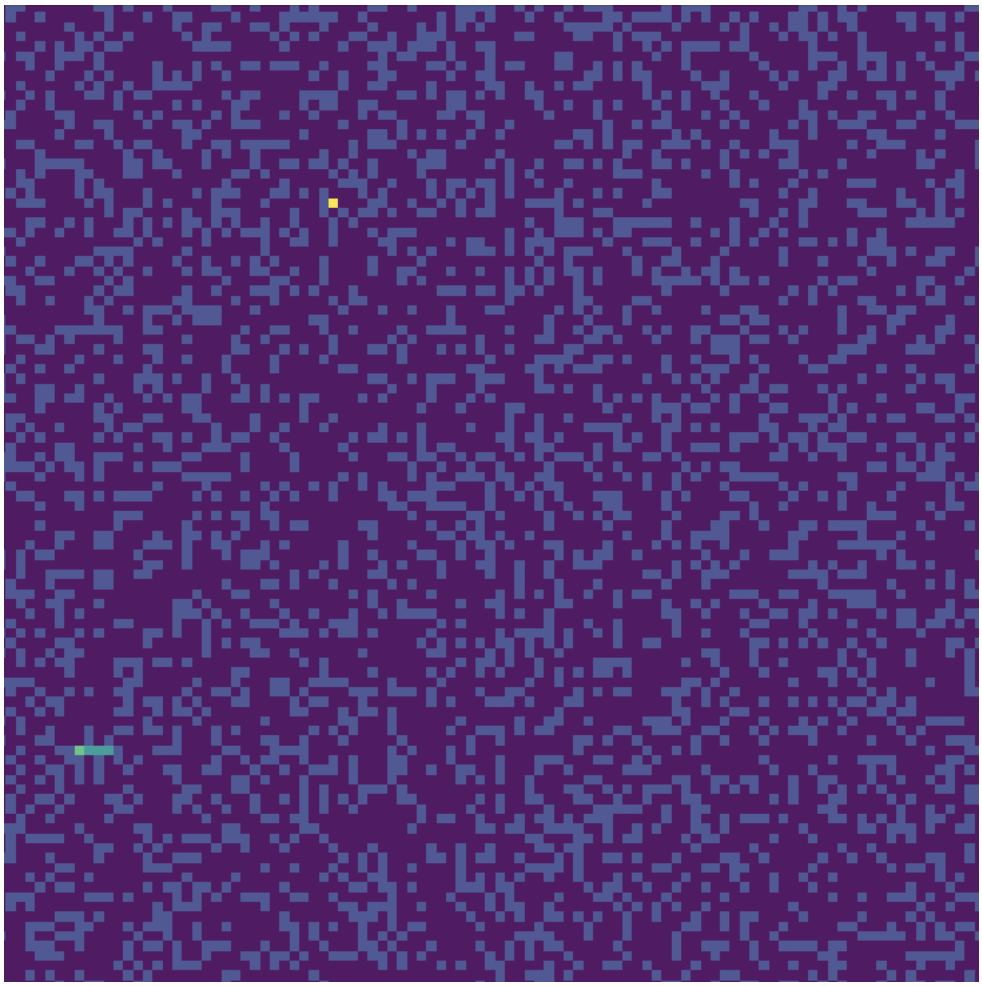
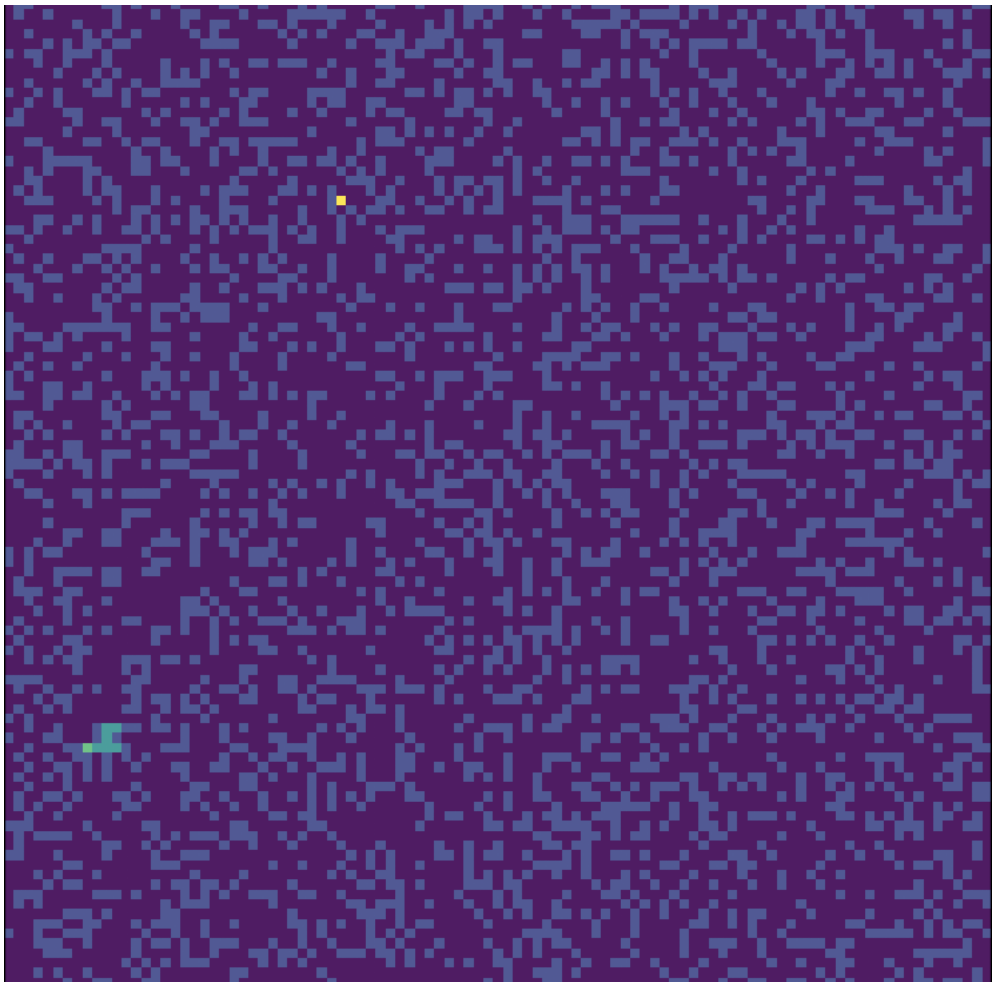
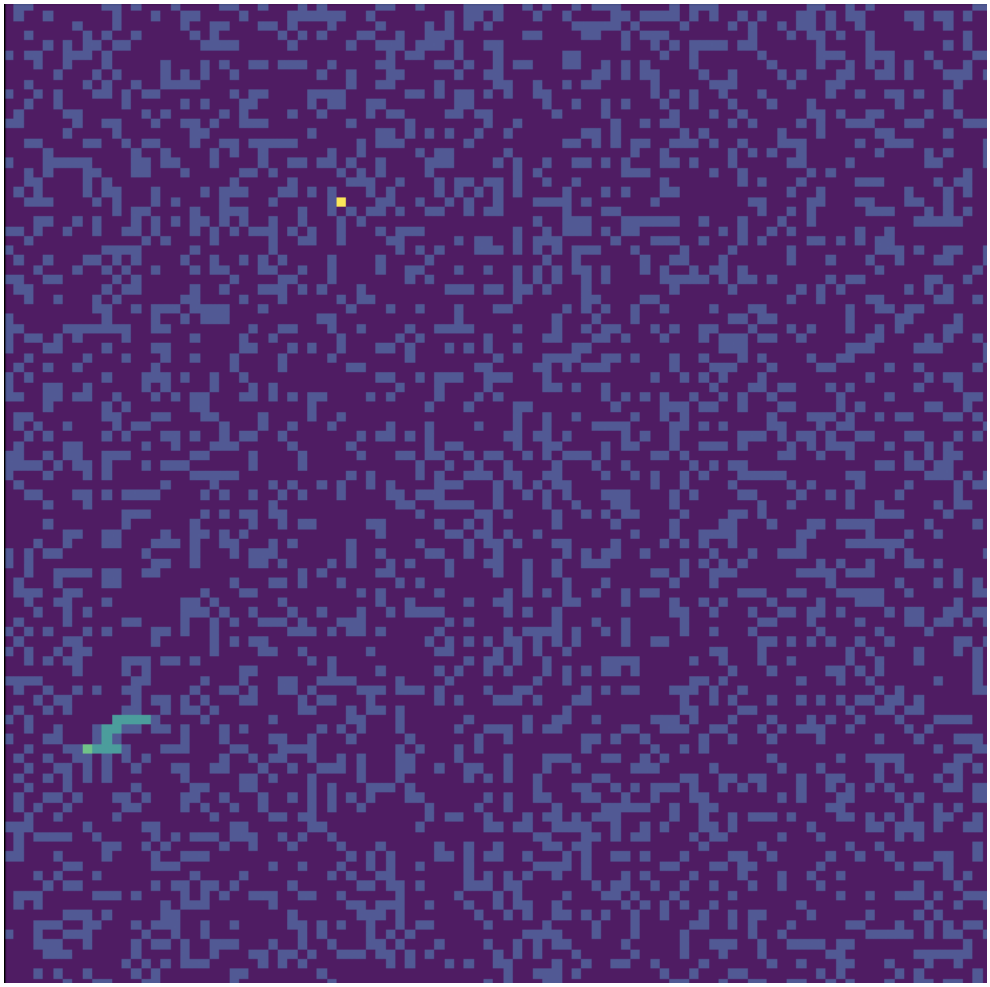
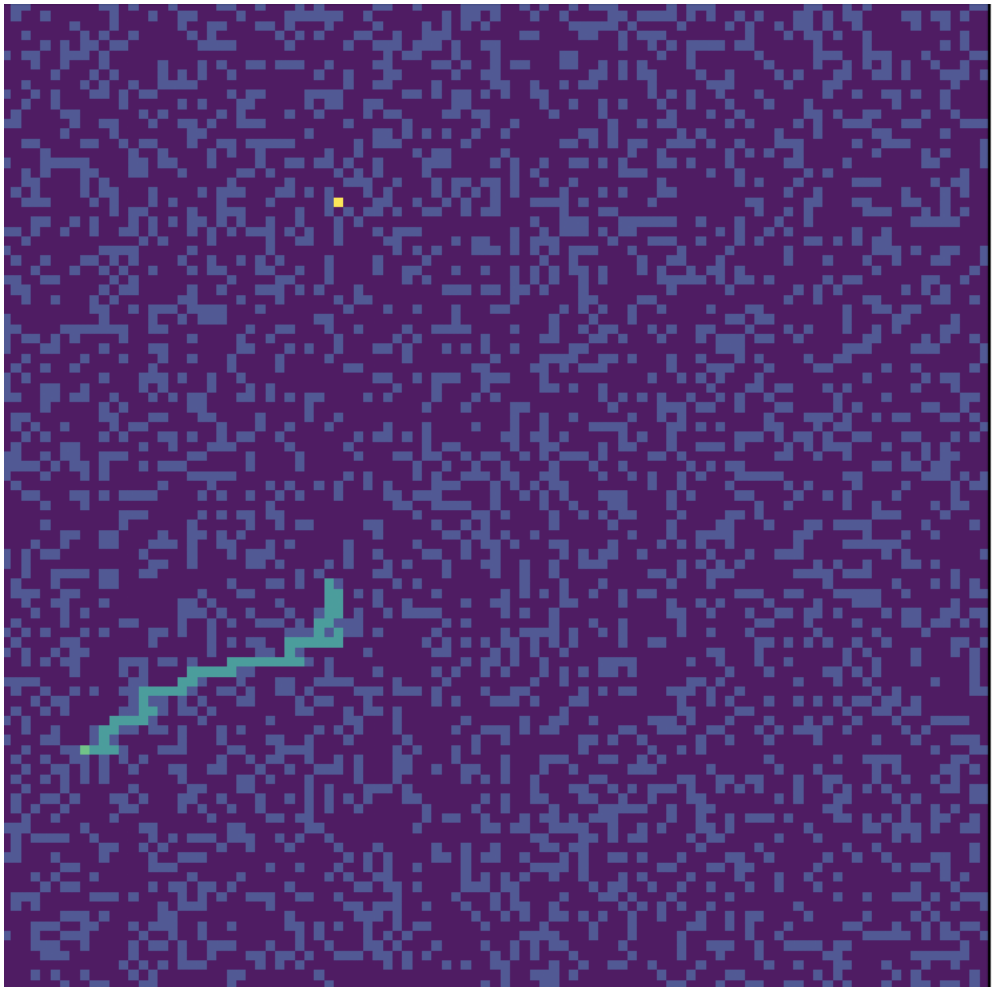
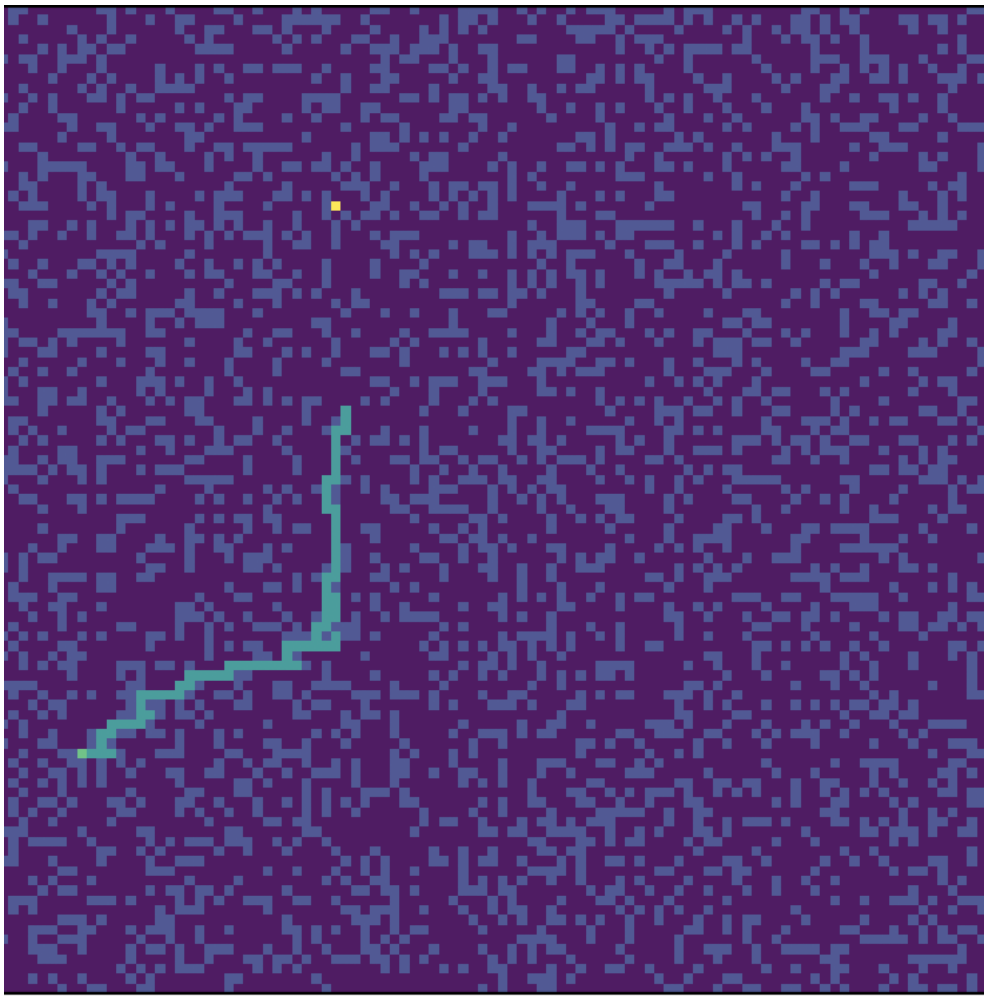


Figure 1: Forward A* favoring smaller h. Yellow is the goal









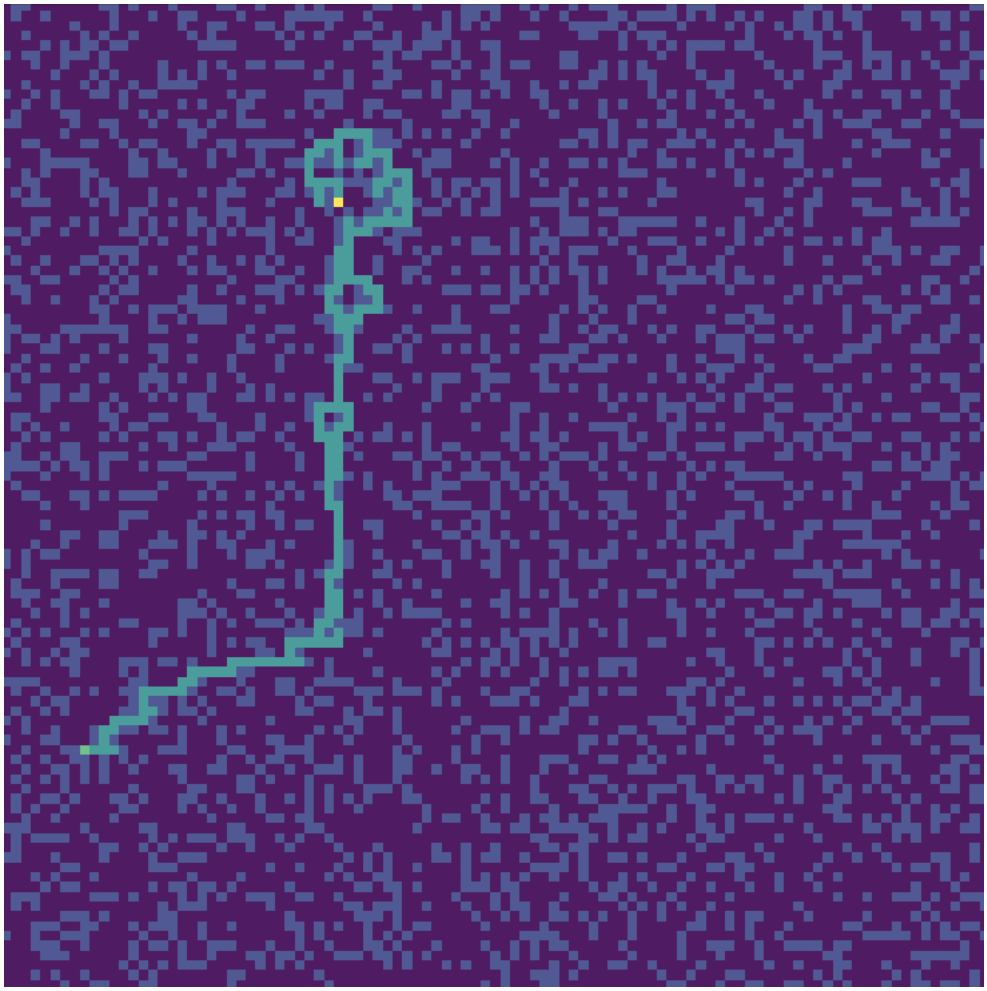


Figure 2: Goal Found

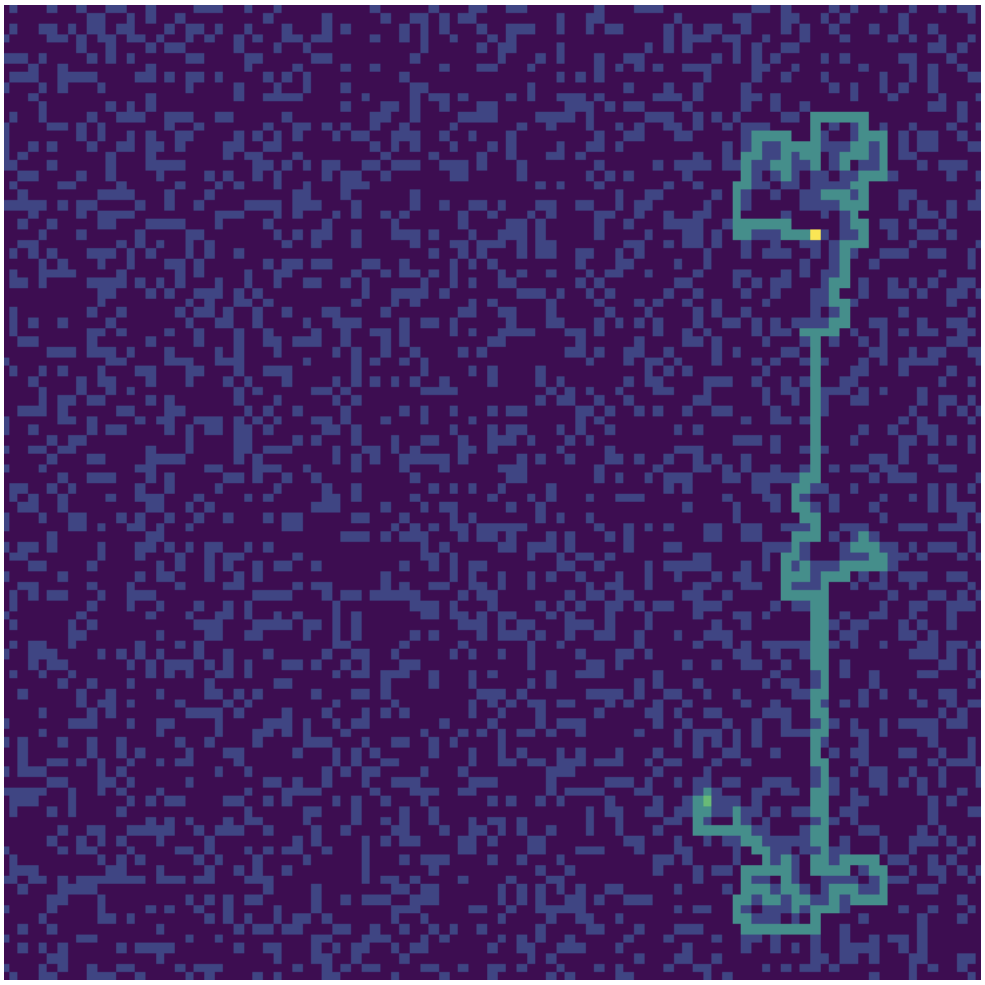


Figure 3: The following are different searches implemented on the same maze. This is Forward A* with larger g tie breaker

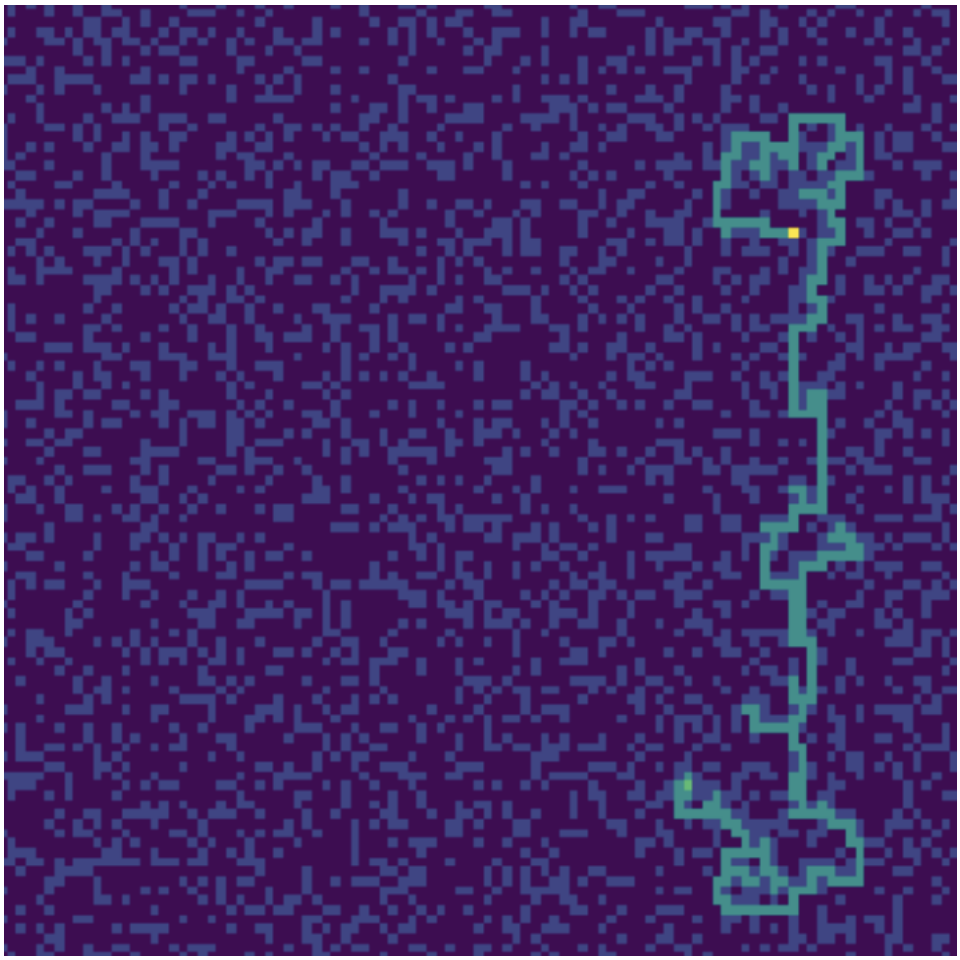


Figure 4: Forward A* with smaller g tie breaker

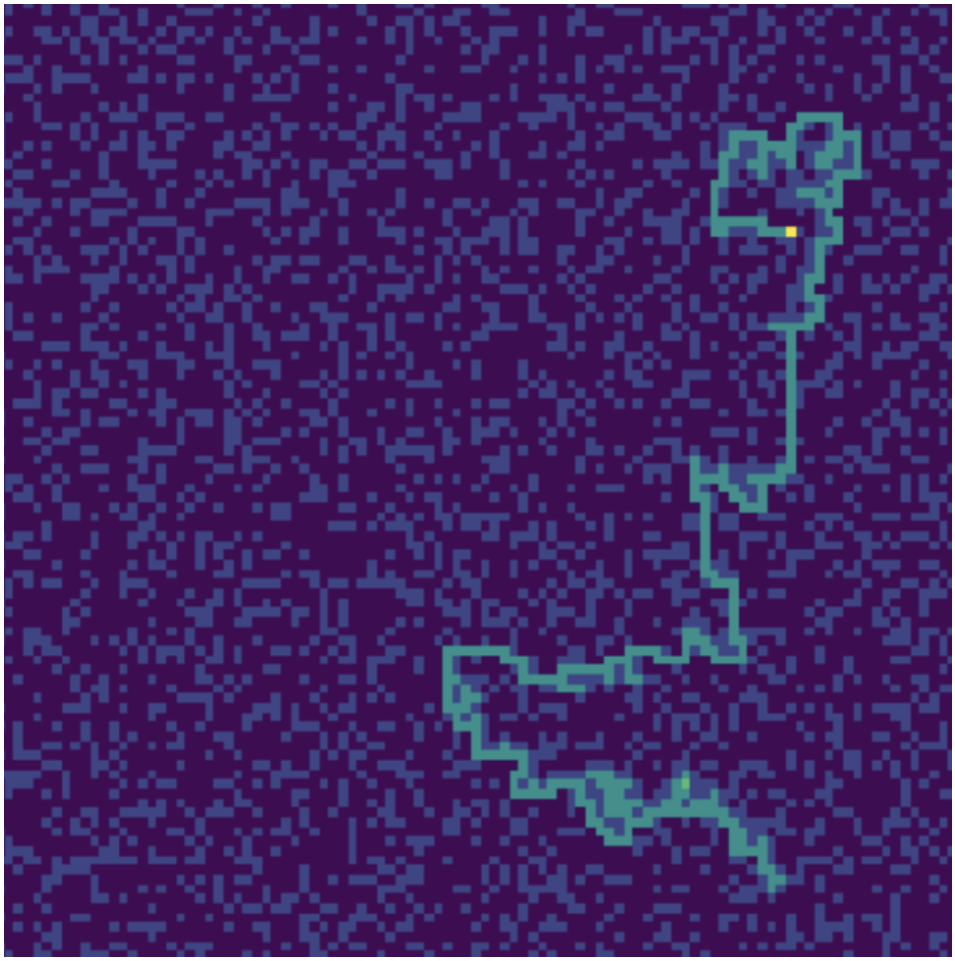


Figure 5: Backwards A*

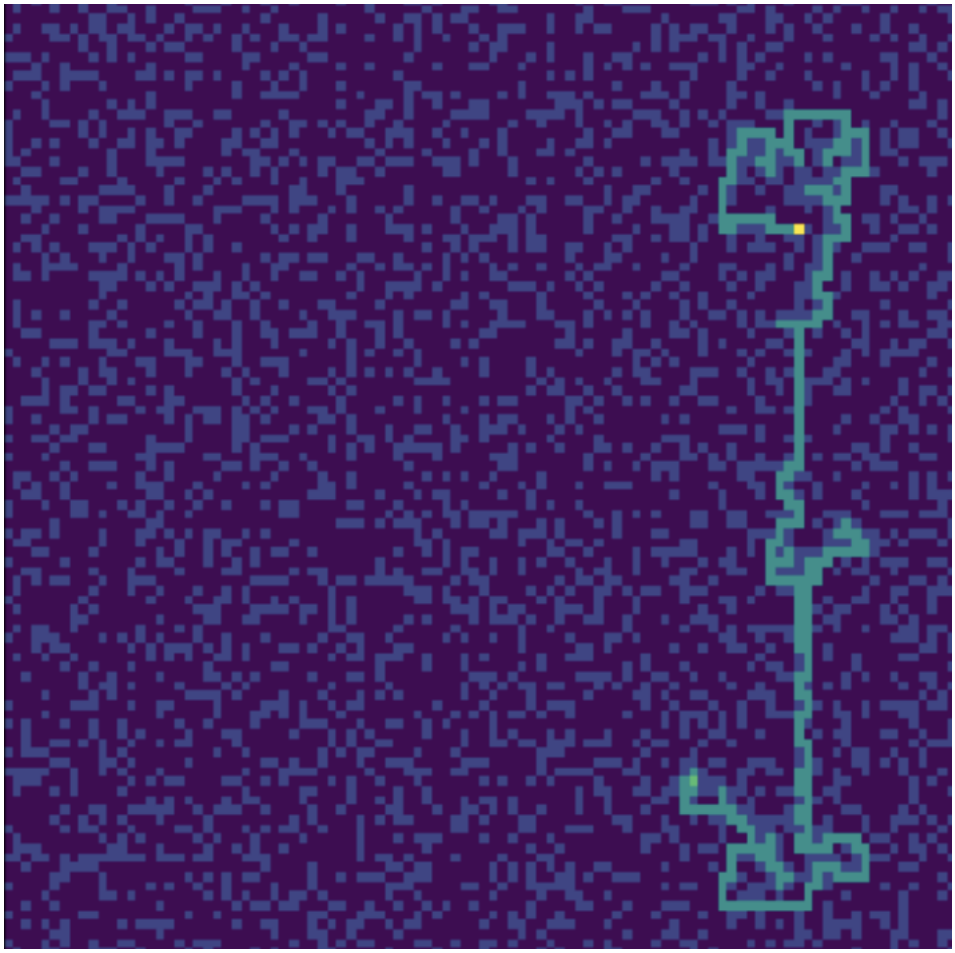


Figure 6: Adaptive A*

Question 3

After running backward A* 50 times we got an average nodes expanded of 38166.68. Forward A had 555.7 expanded nodes. This can be explained by the limited information the program has when searching backwards. The only blocked cells visible to the agent are the ones that it physically goes near. When you search backwards, it is assumed that all cells near the goal are unblocked. This can only be confirmed when the agent goes near it, but when the path is blocked, a lot of rerouting may be needed. These factors make it so that generally, the paths found by backwards are less optimal.

Question 4

A heuristic is consistent if it satisfies for every node n and a node after it n':
(i) $h(n) \leq c(n,n')+h(n')$
(ii) $h(Goal) = 0$

For Manhattan distances, (i) is true because the heuristic is always the most direct path to the goal (acting as though there are no blocked tiles). Therefore, when you take any path it costs either equal or greater than taking the direct path given by the Manhattan distance. For example, if the goal is 2 north and 3 east of the robot's current location: If you move north or east the cost is 1 and the new heuristic is 1 lower ($5 \leq 1+4$). If you move south or west the cost is 1 and the new heuristic is 1 greater due to the robot moving further away ($5 \leq 1+6$). (ii) is inherently true because the Manhattan distance from anything to itself is always 0.

Adaptive A* updates a heuristic based on: New $h(s) = g(goal) - g(s)$ for a node s. C(n,n') is the cost to go from n to n'. This could be reinterpreted as $g(n') - g(n)$. So for it to be consistent it must satisfy:
$g(goal) - g(n) \leq h(n') + g(n')$ -g (n) -> $g(goal) \leq h(n') + g(n')$
This must be true because the heuristic for n' is consistent meaning that it must follow the triangle inequality. (ii) is inherently true because $g(goal) - g(goal) = 0$.

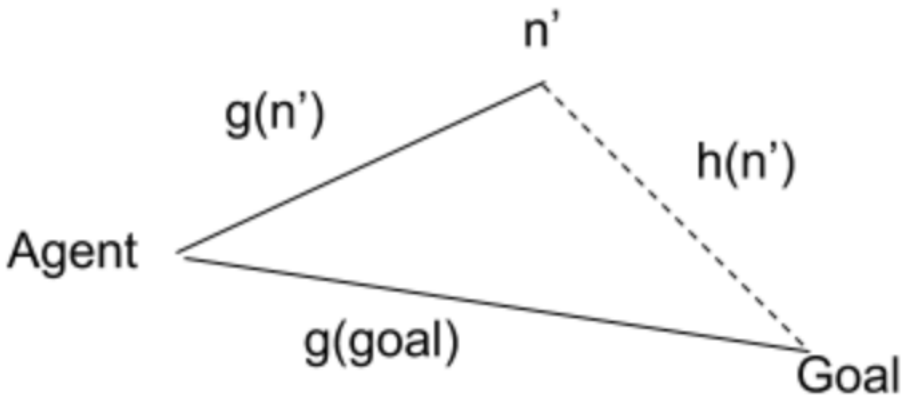


Figure 7: Due to the triangle inequality, g(goal) $\leq$ h(n') + g(n')

Question 5

Adaptive had an average of 526.66 nodes expanded after 50 runs. Forward A had 555.7 nodes expanded. AdaptiveA has the benefit of improving its heuristic to be more accurate, which can help the agent make better informed choices. The benefit may not be very significant because it is offset by the time needed to recompute the heuristic values, which forwardA does not have to do.

Question 6

If the gridworld has blocked cells, this must be stored in each cell adding 1 bit to the cost per cell. To find the parent cell we can assign up, down, left, right to 00, 01, 10, 11 which would point to the direction of the parent cell. This would add another 2 bits per cell. (Hence, the tree-pointer implementation) H, F, can be calculated on the spot, and G can be found by recursively tracing the treePointers. Only searchValue had to be stored in our implementation, which is also upper limited by the amount of moves the agent can do at most, which is stated to be the number of blocks squared (1001 * 1001) and can be represented in 20 bits. In total, this implementation on a 1001 by 1001 gridworld would take (1+2+20) * 1001 * 1001 = 23,046,023 bits. 4 MBytes (32 million bits) be able to handle a maximum of a 1179 by 1179 cell gridworld.