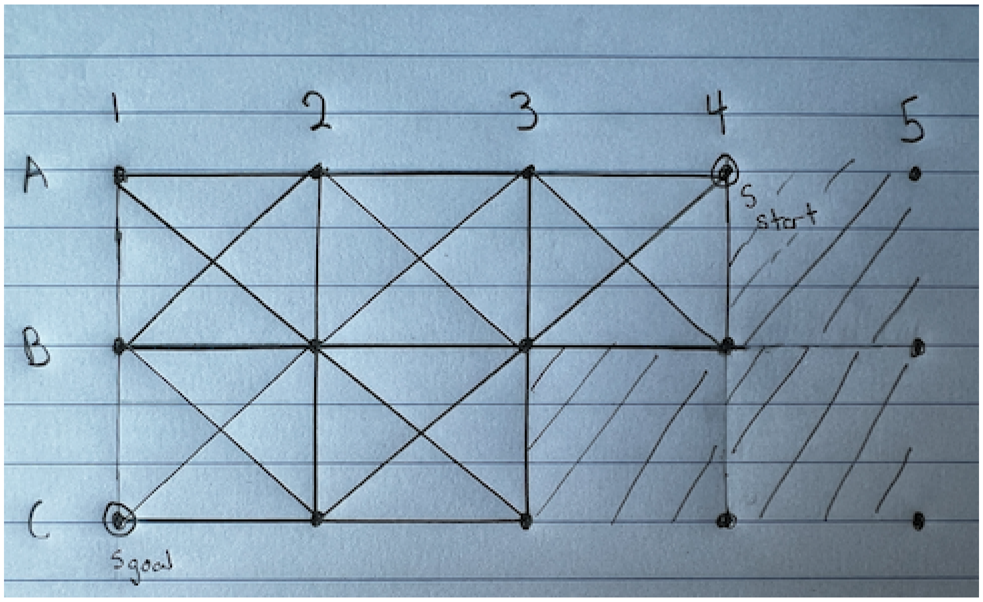
HW 1 - Intro to AI

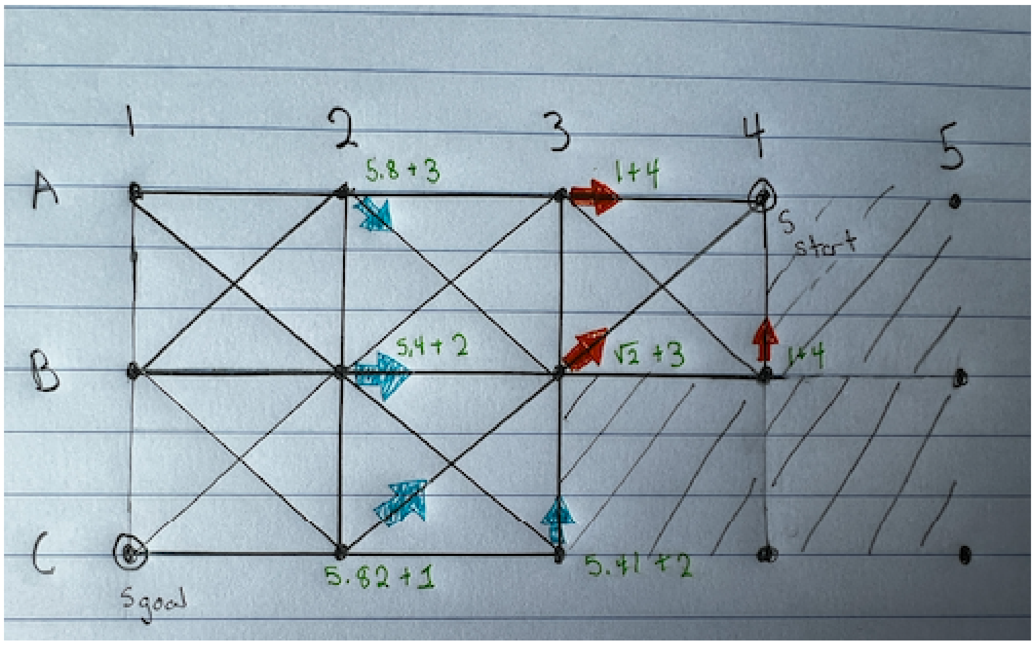
(vk374, ejs235, hc867)

September 2022

1 Part B

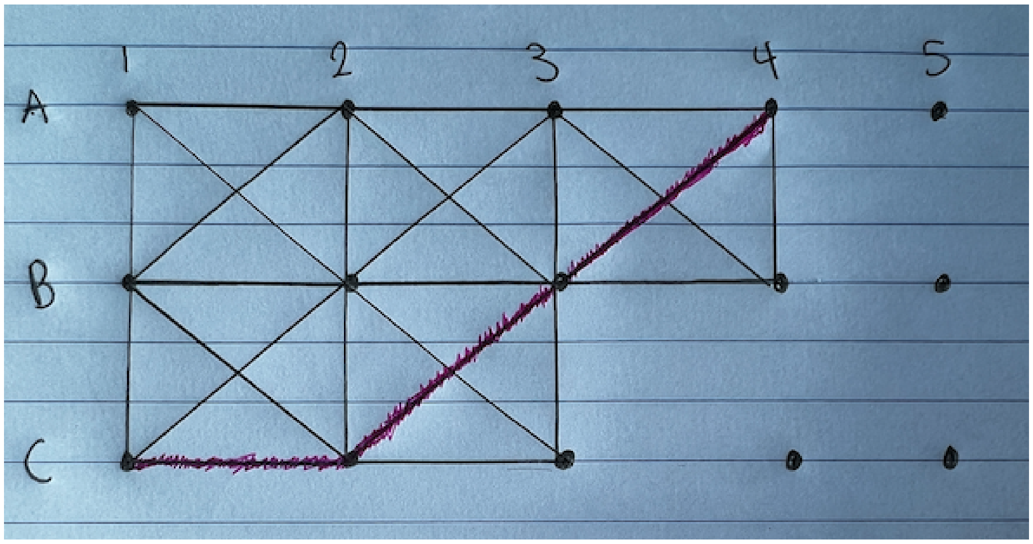


Start and Goal Node



A* First Two Iterations

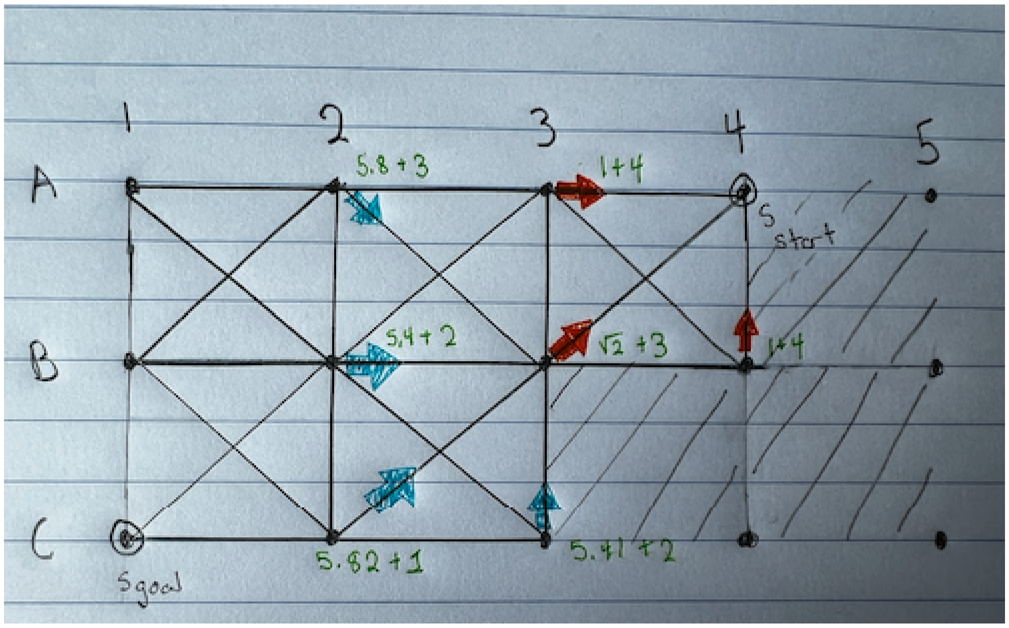
The second iteration of A* discovers that C2 has the smallest cost and the next iteration continues from that point. After C2 the cost at C1 is the smallest and A* is complete.



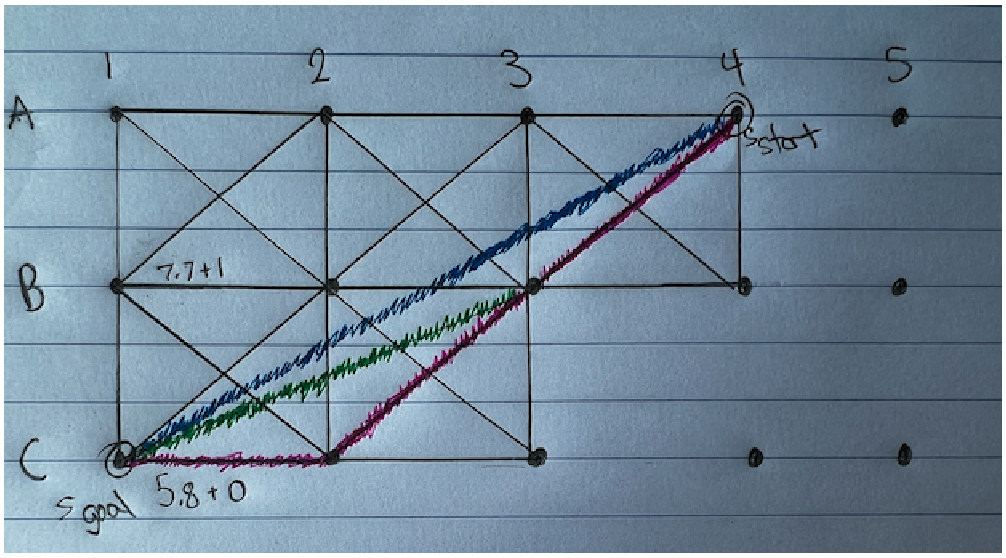
A* Final Trace Path

2 Part B – Any-Angle Path

For Theta* the heuristic doesn't change. Theta* expands B3 and finds C2 to be it's un-expanded successor. There is still a straight path between the start and goal node but since the path overall remains consistent there is nothing significantly different than before.



Theta (First two iterations)



Theta (Final iteration)

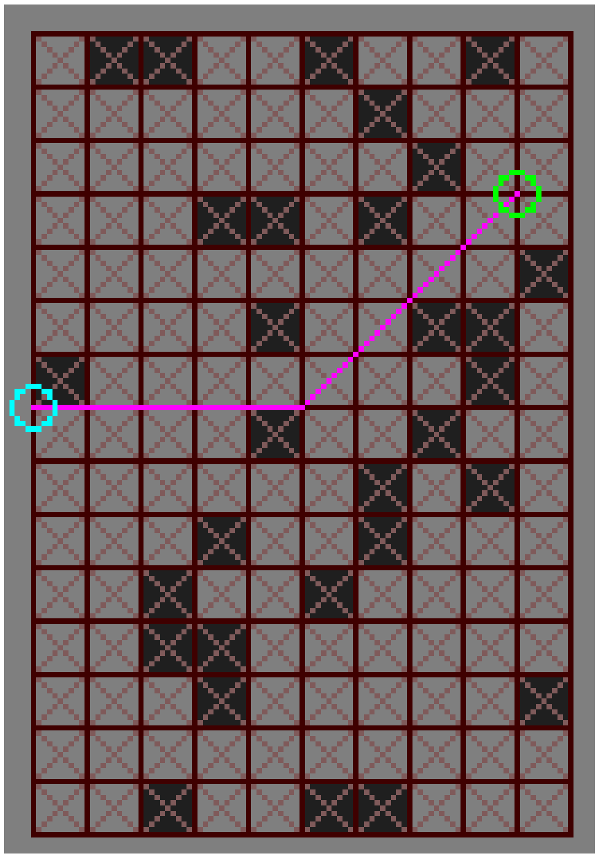
3 Part C

To implement the search Algorithim A* we first had to define our start and end nodes. Calling our start node s, we then marked s as "open" and calculated f(s).

The next step is to calculate which open node n had the smallest f cost. To calculate the f(n) we used it's definition f(n) = g(n) + h(n) where g(n) is the cost from the start node to node n and h(n) is the estimated cost of the lowest cost path from node n to the target node(goal).

If we find the goal node we return that the path has been found, otherwise we continue to open and expand the nodes with the lowest cost to the path according to the above function and close those we have visited until the goal is reached.

We check if the g value of a new path to state s' is better than the path of a node that already exists in the fringe.



A* Example

4 Part D

Implementing Theta* was simple in that it was almost identical to A* with a few key differences. One is that when it updates the g-value and parent of a previously unexpanded but visible neighbor node s' of node s such as is defined in the UpdateVertex pseudo code it looks at two paths instead of only one like A*. We made sure to look at this possibility and also made sure our program found the true shortest path.

5 Part E

In order to prove why A* is guaranteed to find the shortest grid paths we first need to show that h(n) is consistent so that we can state the values of f(n) will not decrease along any possible path. One example from the textbook states - Suppose n' is a successor of n; then g(n') = g(n) + c(n, a, n') for some action a, and we have f(n') = g(n') + h(n') = g(n) + c(n,a,n') + h(n') ¿= g(n) + h(n) = f(n). The next thing to do is show that when A* selects a node N to expand that the optimal and best path to that node has been found already. This must be true because if it wasn't there would have to be a different node n' on the way to the goal. Also since f doesn't decrease along a path n' would be chosen before n due to having a lower cost.

We can conclude from the two main statements above we can say that the nodes that are expanded by A* are in a non-decreasing sequence in terms of their f(n). We can conclude that the beginning node we choose for expansion has to be the best solution. We know this because f must be the real cost for goal nodes and that future nodes will be at minimum the same cost as the first.

We can say that A* expands the nodes where f(n) is less than the cost of the best path and that A* might expand some nodes where f(n) is equal to the cost of the best path first prior to choosing a target node. However A* doesn't expand anything that has a cost greater than the cost of the best path.

Finally we can say that for a given constant heuristic that A* is optimally efficient and won't expand more nodes than necessary.