Playbook Notes: A* Algorithm

The A* (A-Star) algorithm is the most efficient path searching algorithm there is (at least at this current time).

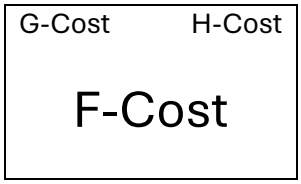
During an A* search, all adjacent nodes of the current node are evaluated with three calculations.

G-Cost: The distance from the starting position.

H-Cost (heuristic): The distance from the ending position. This cost is sometimes just an estimate.

F-Cost: (G-Cost) + (H-Cost)

These calculations are usually represented like this:



As nodes are evaluated, they are stored in priority queue where the lowest F-Cost has the highest priority.

A* Algorithm (This is a high-level design; many details are left out):

While (The goal node is not found)
{

1. Evaluate the three costs for all adjacent cells that are open.
2. Each cell is placed into a priority queue.
3. Pop the Priority queue and mark it as closed.
4. Go to step 1

}

Note: When coding A*, you will want to track the parent (the previous node) of each node. This step is often skipped when demonstrating the A* algorithm.

Note: If two nodes have the same F-Cost, the tie breaker would be lower H-Cost.

Very Simple Example:

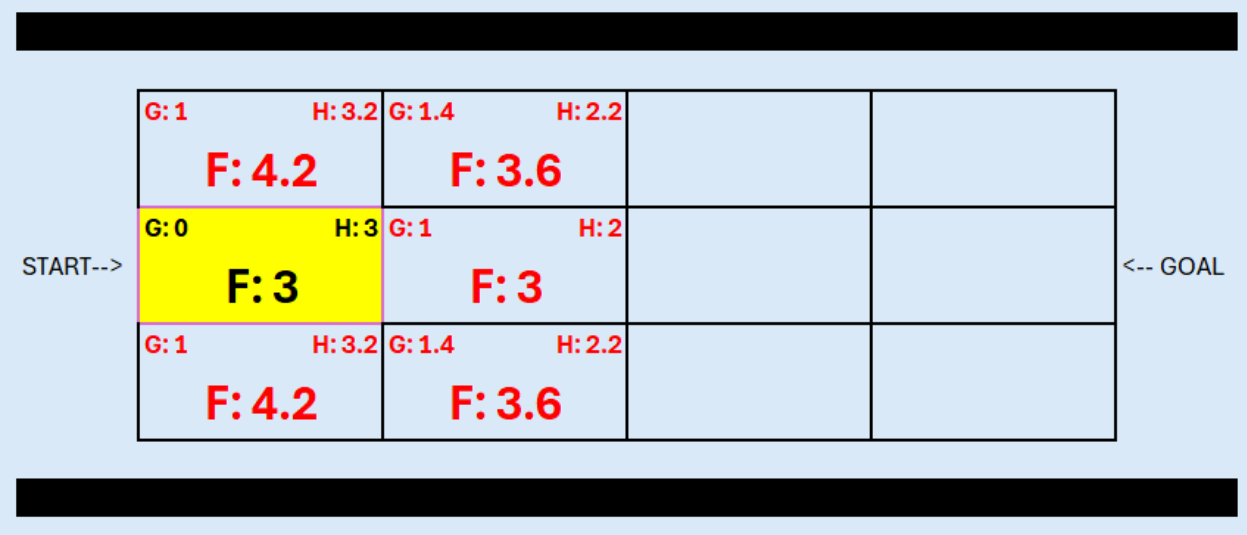
In the following graph, there is only one way to go and it's a straight line so all the F-Cost values are the same.

In the above example we start in the node at the left and "look" for a path to the cell on the right. (I use quotes because there is only one way to go.)

Now, let's look at something a little more complex by adding a row above and below.

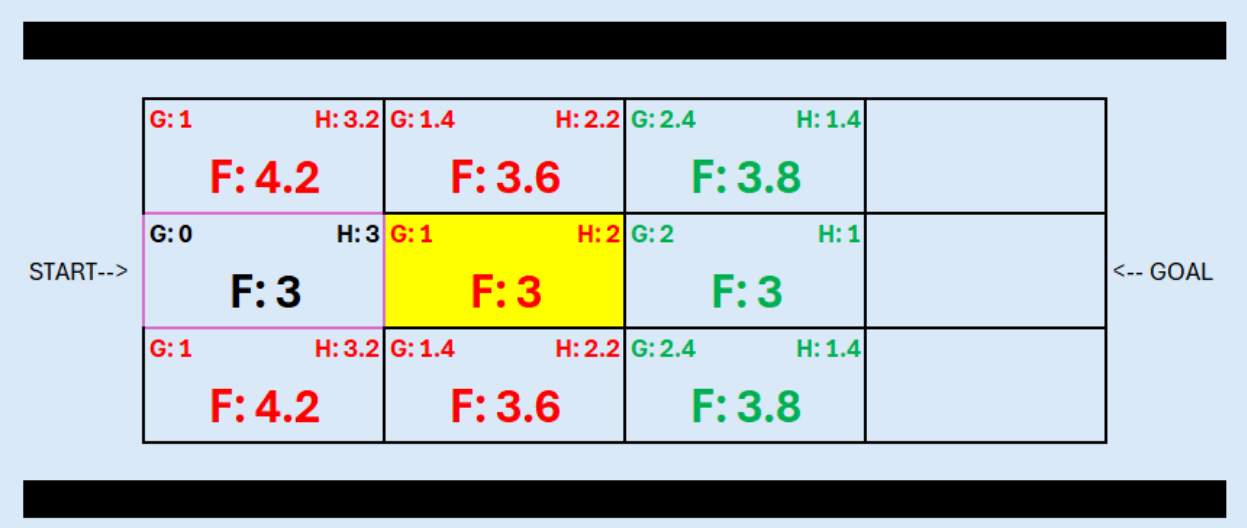
The start will be the middle row on the left and the end will be the middle row on the right.

From the starting position, we will calculate the cost of all the adjacent nodes.



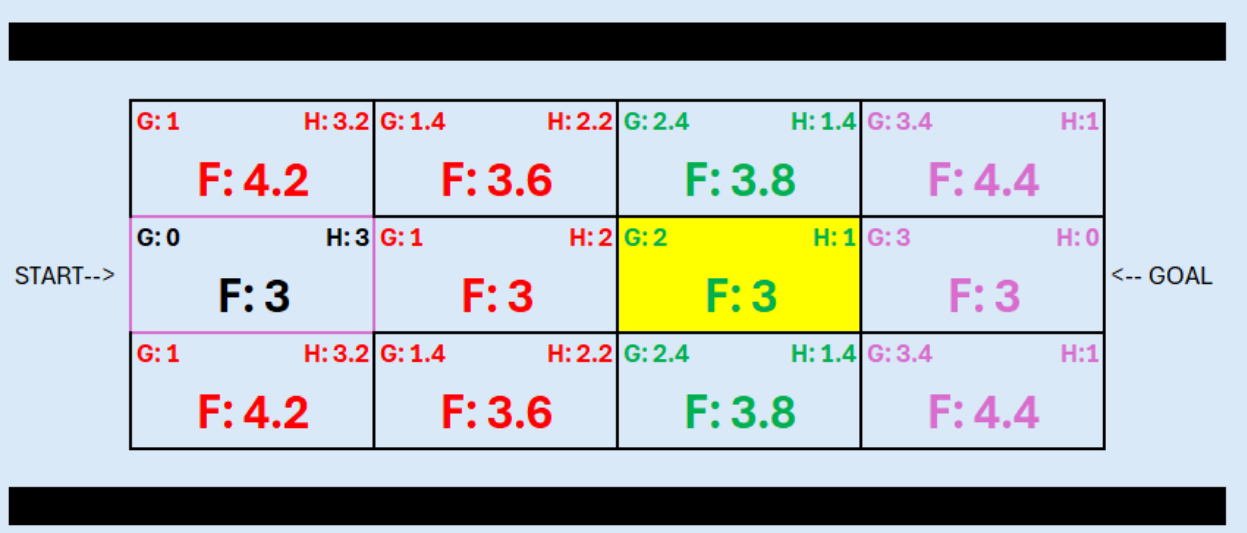
Note: Diagonal moves are calculated using the Pythagorean theorem.

After evaluating each node, the node is placed in a priority queue where the smallest F-Cost has the highest priority. Then we pop the queue (this will be the node to the right), move to the next cell, and evaluate all the adjacent cells.

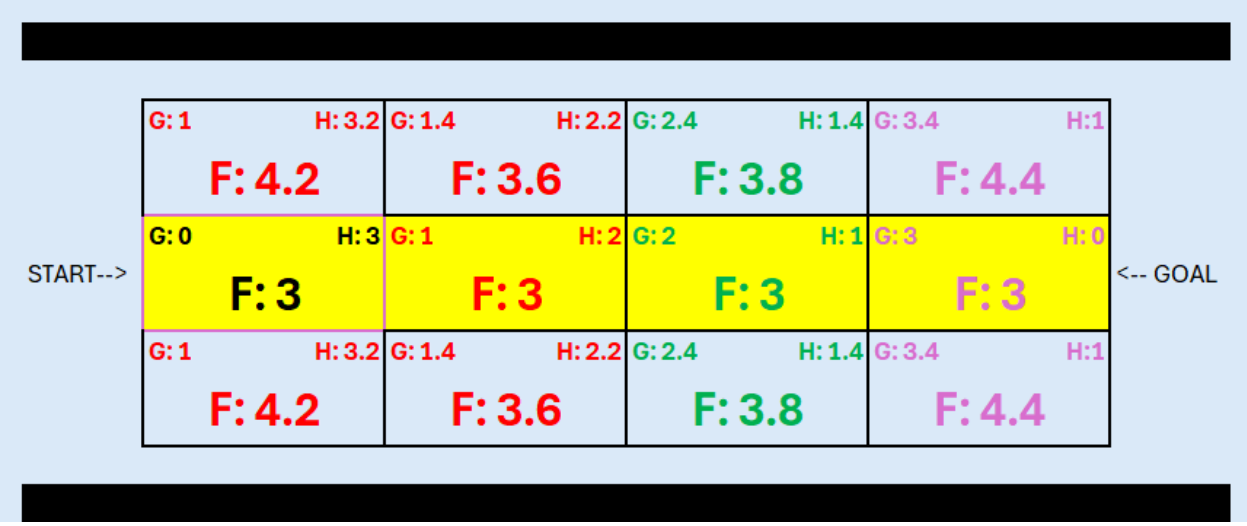


In this example, the node above and below would be reevaluated, but the because the G-Cost would increase to 2, the F-Cost would increase. Because the current data is better than the new data, the old data will remain.

After popping the queue, we move to right one spot and reevaluate all adjacent nodes.



After popping the queue and moving to next smallest F-Cost and with the smallest H-Cost, we have reached our goal! If we are keeping track of all the parent nodes, we can trace back the shortest path from the start node to the goal node.



Reference:

https://www.youtube.com/watch?v=-L-WgKMFuhE

Comparing Search Algorithms:

https://www.youtube.com/watch?v=GC-nBgi9r0U