Astar Algorithm Pseudocode

// A*
Function A*(start, goal)

// Initialize the closed list, the set of nodes already evaluated
Closedset = the empty set

// Initialize the open list, the set of tentative nodes to be evaluated
// initially containing the start node
Start g score = 0
Start f score = g score plus h “Manhattan distance from start to goal”
Openset = {start}
came_from = empty // The map of the navigated nodes

while the open list is not empty
Find the node with the least f on the open list, call it "q"
Pop q off the open list
Generate q's 4 neighbors
For each neighbor
    If neighbor is the goal, stop the search
    neighbor.g = q.g + distance between neighbor and q
    // h distance is the “Manhattan” distance on a square grid
    neighbor.h = distance from goal to neighbor
    neighbor.f = neighbor.g + neighbor.h
    If a node with the same position as neighbor is in the OPEN list \ which has a lower f than neighbor, skip adding this neighbor
    if a node with the same position as neighbor is in the CLOSED list \ which has a lower f than neighbor, skip adding this neighbor
    otherwise, add the node to the open list
    and came_from[neighbor] = q // set neighbor's parent equal to q
end
push q on the closed list
end

// After Astar has run call reconstruct_path passing it the final goal point to combine the total path (but in reverse order from finish to start.

function reconstruct_path(came_from, current) // first current is goal
while current in came_from:
    total_path.append(current)
    current = came_from[current] // assign current to the parent
return total_path