A* Search Algorithm Visualization

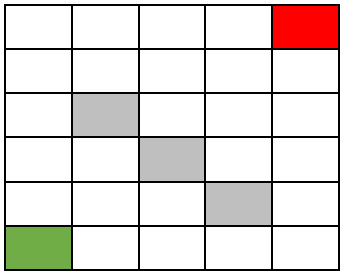
Oscar Jaimes
AUCSC 310

Introduction:

The A* Search Algorithm is a variant of Dijkstra's algorithm which is a *graph traversal and path search algorithm*. A* is a best-first search algorithm which means the algorithm explores a weighted graph by expanding the best valued node chosen according to a pre-determined function that outputs a certain cost for the node in order to reach the goal node in the graph.

Problem Definition:

Let's assume there exists a two-dimensional grid such that each unit square in the grid is a node that can be represented in a graph. Let's also assume that the green node is the starting node and the red node is the end node, while the grey nodes represent a form of a barrier such that a valid path cannot travel through:



The purpose of the A* search algorithm is to find the shortest open path between a start node and end node in a graph. In order to find the shortest path, the algorithm performs a series of steps. At each step, it picks the node according to a value from a function, *f(x)*, which is equal to the sum of two other functions, *g(x)* and *h(x)*. During each iteration, the next best node is the node which has the lowest *f(x)* output. In simple terms, respectively, *g(x)* and *h(x)* are functions that define the cost to move from the starting node to another given node in the graph and an estimated move cost to travel from a given node in the graph to the end node. The function *h(x)* is often defined as a heuristic function, which acts as a good estimate to calculate a movement cost from a given node to the end node. The algorithm will either find the shortest path from the start node to the end node, or, it will fail to find a path.

Solution:

I will be building a visualization of the A* path finding algorithm in JavaScript using the P5.js framework to graphically demonstrate the behavior of the algorithm. In order to do this, I will be utilizing priority queues, which were studied in class, to organize and classify the nodes to be traversed and expanded. Furthermore, I will be using a Euclidian heuristic function in order to compute *h(x)*.