# The A\* Pathfinding Algorithm

## Introduction

The A star algorithm was created as part of the Shakey project back in 1966, which a solution for a mobile robot that could walk by itself. There are two famous pathfinding algorithms before the A\*, they are Dijkstra’s Algorithm and Best-First Search. Both of the algorithm has its own advantage and disadvantage, the A star algorithm is combining both of them and try to take advantage from these two well-known algorithms.

For Dijkstra’s Algorithm, it starts with the starting point, then repeatedly examines the closest vertex and keep adding them to its visited vertices. This process looks like the graph when it applies on a tiled map:

Chart

Description automatically generated

On the other hand, the Best-First Search algorithm works with an estimation which called a heuristic, allows the algorithm to find the next vertex bases on the ending point of the map, this approach also known as Greedy Algorithm. The process looks like as below when it applies on a tiled map:

Chart, bar chart

Description automatically generated

The A star algorithm is like Dijkstra’s Algorithm in that it can be used to find a shortest path and always get a result even there an obstacle between the starting point and the ending point. It also like Greedy Best-First Search in that it can use a heuristic to guide itself if there is nothing in between and it will give a straight line very fast. Based on that information, the A star algorithm will examine the cost by using following equation:

Where, the g(n) represents vertices far from the starting point, and h(n) represents vertices far from the goal.

## The function of heuristic

The A star algorithm is suitable for many scenarios due to its flexibility, and heuristic function make this happen, changing the heuristic can be used to control the pathfinding algorithm’s behavior.

* If h(n) is zero, then the g(n) in cost estimation function will dominate, the A star turns into Dijkstra’s Algorithm.
* If h(n) is lower than the cost of moving from n to the goal, then A star will find a shortest path but work less efficiency.
* If h(n) is exactly equal to the cost of moving from n to the goal, this is the ideal case and will bring the algorithm the perfect performance.
* If h(n) is greater than the cost of moving from n to the goal, then A star will not give the shortest path every time but with better speed.
* On the other extreme, if h(n) dominates, the A star will work as a Greedy Best-First Search.

## Complexity

The time complexity of A star depends on the heuristic, and it’s polynomial when the search space is a tree which explains that A star algorithm is often used for the common pathfinding problem in video games.

The space complexity of A star is roughly the same as that of all other graph search algorithms since it will store all visited nodes in the memory, and this is the drawback of this algorithm because it offers the algorithm works only with memory-bounded heuristic searches.

## Variants of A\*

* Beam search
* Iterative deepening
* Weighted A\*
* Bandwidth search
* Bidirectional search
* Dynamic A\* and Lifelong Planning A\*
* Jump Point Search
* Theta\*

## Conclusion

There are plenty of pathfinding algorithms out there, however which one should we pick when we are looking for a better solution? The summary is as below:

* One source and one destination
  + A star algorithm (for unweighted as well as weighted graphs)
* One source, all destination
  + BFS (for unweighted graphs)
  + Dijkstra (for weighted graphs with negative weights)
  + Bellman Ford (for weighted graphs with negative weights)
* Between every paired of nodes
  + Floyd-Warshall
  + Johnson’s Algorithm