

02-613 Week 4

Algorithms and Advanced Data Structures

Aidan Jan

September 15, 2025

A-Star Algorithm

Suppose we have a GPS, and we want to get from Pittsburgh to Philly. Currently, we have discussed how to do that with Dijkstra's algorithm, however:

- This requires us to explore paths to nodes on the whole map!
- Slow, and most calculations are unnecessary.
- Is there a better way?

In our GPS case, we would want to tell the algorithm, somehow, you should search paths going east.

- At the same time though, depending on where you are, not every path should go east. Maybe a highway starts slightly to the west, and it will save time.
- How do we decide how far west is allowed?

What if we use the straight line distance as an estimate to which node (city) to visit next?

- In addition to the $d[] = \infty$ for the start of Dijkstra's, we also initialize every city with a $h[] =$ (straight line distance to destination).
- The $h[]$ is called a *heuristic*.

```
for u in V, d[u] = INTEGER_MAX, p[u] = 0, F = V
d[s] = 0
while F != Q:
    f[u] = d[u] + h[u]
    u = (vertex in F with min f[u])
    remove u from F
    for v in Neighbor(u):
        if d[u] + h[v] < d[v]:
            p[v] = u
            d[v] = d[u] + h[v]
```

- Note that F is a heap, which allows for quick removal of the minimum item.

Heuristics

What do we choose the $h[]$ function to be? We chose distance in this case, but what else may be good?

- **Theorem:** A^* is guaranteed to find the optimal route if $h(u)$ is *admissible*.
- In this case, the best heuristic we can come up with is the time to travel from one city to another, since that instantly finds a solution.

- The worst heuristic we can come up with is $h(u) = 0$, since that would cause us to explore every node just like Dijkstra's.
- In other words, the heuristic is always an *underestimate* of the correct solution.
 - If it wasn't an underestimate, it may cause us to miss the optimal path!
- We want to show that with a good enough heuristic, A* would find the optimal route, even without exploring every node.

Proof

Let P^* be the optimal path from point A to point B. This path includes nodes w_0, \dots, w_m , with w_0 also being the start, and w_m is the last node before the end, t .

- **Lemma:** When each node w_0 to w_{m-1} was last removed from the heap, $d(w_1) = d^*(w_1)$
- This is provable by induction.
 - Base: $d[s] - d[w_0] = 0$
 - Hypothesis: We assume that every next node is also on the optimal path. Therefore,

$$\begin{aligned} d(w_{k+1}) &= d(w_k) + \text{len}(w_k, w_{k+1}) \\ &= d^*(w_k) + \text{len} \\ &= d^*(w_{k+1}) \end{aligned}$$

Suppose for the purpose of contradiction, the A* algorithm found a non-optimal solution. Let w_m be a node that we decided to visit over t , the ending node.

$$\begin{aligned} f[t] &= d[t] + h[t] \\ &= d[t] \\ &\leq f[w_m] \\ &= d[w_m] + h[w_m] \\ &= d^*[w_m] + h[w_m] \\ &\leq d^*[w_m] + h^*[w_m] \\ &= \text{length}(p^*) \end{aligned}$$

p^* is the known optimal solution, and h^* is the optimal heuristic.

- Remember that $h[t]$ is just 0, since it is the ending node.
- Since we visited w_m instead of t , that means $f[w_m]$ must have been lower, hence step 3.
- $d^*[w_m]$ gives the same value as $d[w_m]$, since we assume the path we took to get there was optimal. This is based on the lemma.
- We know that $h[w_m] \leq h^*[w_m]$, since the heuristic is an underestimate. $h[w_m]$ was an estimate from w_m to t , $h^*[w_m]$ was the actual distance.
- Since the sum of the optimal distance plus the optimal heuristic is just the length of the optimal path, this path through w_m must be optimal. However, this is a contradiction since this path was defined as non-optimal.

Shortest Paths Summary

- BFS, $O(n + m)$, unweighted graphs
- Dijkstra, $O(m \log(n))$, positive edge weights, single source all paths
- A-Star, $O(m \log(n))$, needs heuristic
- Bellman-Ford, $O(mn)$, arbitrary large weights (incl. negatives)