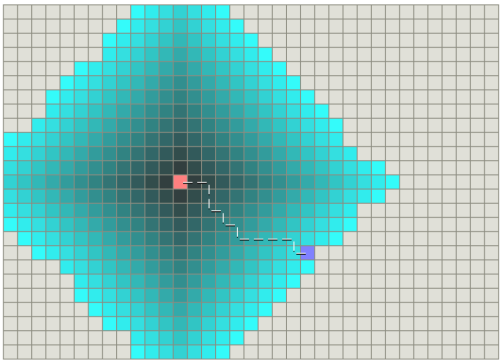
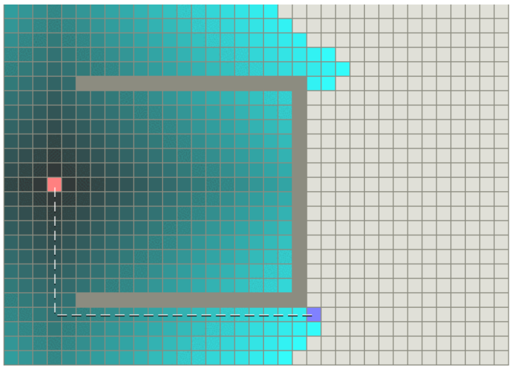
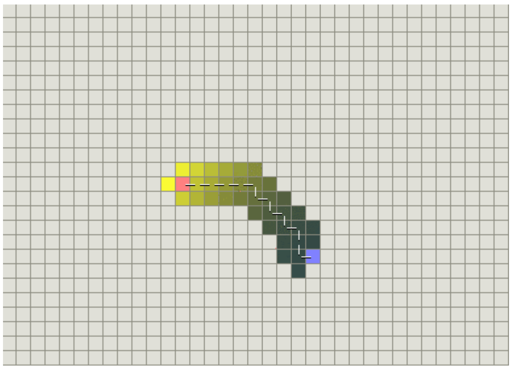
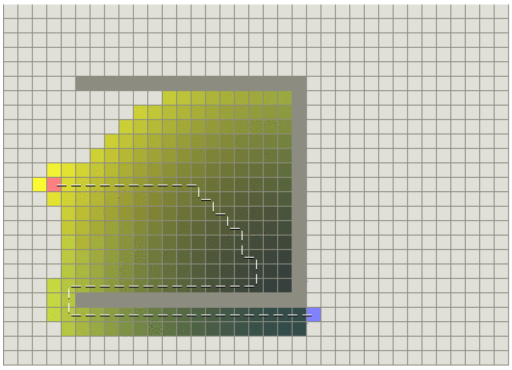
Dijkstra's Algorithm is guaranteed to find a shortest path from the starting point to the goal, as long as *none* of the edges have a negative cost:



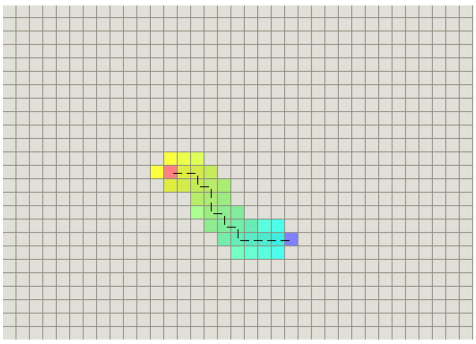


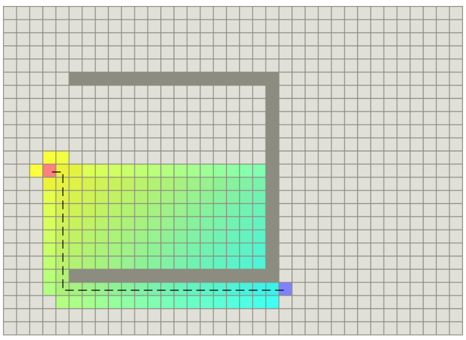
The Greedy Best-First-Search algorithm works in a similar way, except that it has some estimate (called a *heuristic*) of how far from the goal any vertex is. It is *not* guaranteed to find a shortest path but runs much quicker:





A* is like Dijkstra's Algorithm in that it can be used to find a shortest path, and is like Greedy Best-First-Search in that it can use a heuristic to guide itself. In the simple case, it is as fast as Greedy Best-First-Search, in the example with a concave obstacle, A* finds a path as good as what Dijkstra's Algorithm found.





In short, A* computes f(n) = g(n) + h(n), where g(n) is the value calculated from starting point to current point, and h(n) is the heuristic value calculated from current point to the destination point.

- If h(n) is 0, then only g(n) plays a role, and A* turns into Dijkstra's Algorithm.
- If h(n) is very high relative to g(n), then A* turns into Greedy Best-First-Search.
- If h(n) < the cost of moving from n to the goal, then A* is guaranteed to find a shortest path. The lower h(n) is, the more node A* expands, making it slower.
- If h(n) > the cost of moving from n to the goal, then A* is not guaranteed to find a shortest path, but it can run faster.
- If h(n) is exactly equal to the cost of moving from n to the goal, then A* will only follow the best path and never expand anything else, making it very fast.

Comparison

Algorithm	Type	Purpose	Time Complexity	Space Complexity	Suitable for
Bellman-Ford	Single-source	Finding shortest paths with negative weights	O(V * E) (with optimizations, O(V^3))	O(V)	Negative weight graphs
Floyd-Warshall	All-pairs	Finding shortest paths between all pairs	O(V^3)	O(V^2)	Dense graphs, with or without negative weights
Dijkstra	Single-source	Finding shortest paths in weighted graphs	O((V + E) * log(V))	O(V)	Non-negative weighted graphs, single-source
A*	Single-source	Finding shortest path to a target	O(b^d) (exponential, worst case)	O(b^d) (for visited nodes)	Graphs with a well-defined heuristic, single-source

"v": number of vertices in the graph.

"E": number of edges in the graph.

"b": average number of edges from each node.

"d": number of nodes on the resulting path.