Dijkstra's Algorithm on Example Graph

Graph Description: Nodes: A, B, C, D, E, F Edges:

- $A \to B(1)$
- $A \to C(4)$
- $B \to D(2)$
- $B \to E(7)$
- $C \to D(5)$
- $D \to F(6)$
- $F \to E(3)$

Initialization:

$$\begin{aligned} d[A] &= 0 \\ d[B] &= \infty \\ d[C] &= \infty \\ d[D] &= \infty \\ d[E] &= \infty \\ d[F] &= \infty \\ Q &= \{A, B, C, D, E, F\} \end{aligned}$$

Iterations of the given psuedo code:

the algorithm given doesn't save the shortest path. After the some modification to the algorith, the shortest path would be
$A \longrightarrow B \longrightarrow D \longrightarrow F$. THe Q and d are updated as follows:

1. Q = { B,C,D,E,F} and d = {0,1,4,∞,∞,∞}
2. Q = { C,D,E,F} and d = {0,1,4,3,8,∞}
3. Q = { D,E,F} and d = {0,1,4,3,8,9}
4. Q = { E,F} and d = {0,1,4,3,8,9}
5. Q = { F} and d = {0,1,4,3,8,9}
6. Q = {} and d = {0,1,4,3,8,9} and the loop will break since the Q is empty. d is the shortest distances between the start to all other nodes.

Shortest Paths and Costs:

$$A \to B : 1$$
$$A \to C : 4$$
$$A \to D : 3$$
$$A \to E : 8$$
$$A \to F : 9$$

4. A* Algorithm on Example Graph

Heuristic Values:

$$h(A) = 10,\ h(B) = 8,\ h(C) = 6,\ h(D) = 4,\ h(E) = 2,\ h(F) = 0$$

Initialization:

$$d[A] = 0$$
$$d[B] = \infty$$
$$d[C] = \infty$$
$$d[D] = \infty$$
$$d[E] = \infty$$
$$d[F] = \infty$$
$$Q = \{A, B, C, D, E, F\}$$

Iterations of the given psuedo code for A^* Algorithm:

1. Q = { B,C,D,E,F} and d = {0,1,4,∞,∞,∞}
2. Q = { C,D,E,F} and d = {0,1,4,3,8,∞}
3. Q = { C,E,F} and d = {0,1,4,3,8,9}
4. Q = { C,E} and d = {0,1,4,3,8,9} and the loop will break since the target node has been reached. Evrery item we popped in the loop is the shortest path.

Shortest Path:

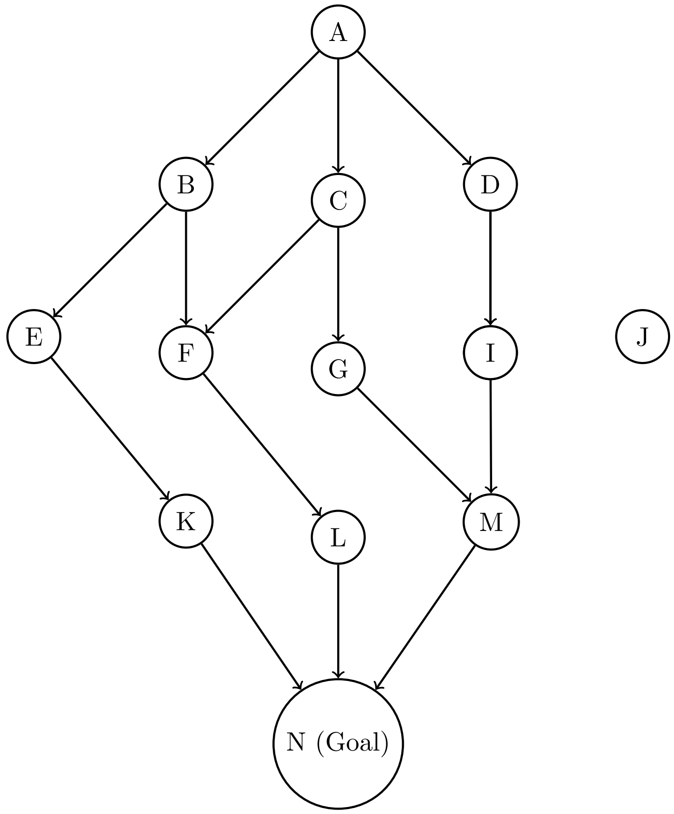
$$A \to B \to D \to F$$

Cost: 9

5. Graphs Where A* Isn't Helpful

Graph 1: Highly Connected Graphs.

A* may explore multiple alternative paths due to the abundance of connections, especially when edge weights are nearly equal. This increases the number of nodes explored unnecessarily.



Misleading Heuristic:

In highly connected graphs, even a reasonable heuristic can mislead A* into exploring unnecessary nodes. Below is a dense graph where A* fails to prioritize the shortest path efficiently.

A heuristic misaligned with the graph's geometry can guide A* in the wrong direction, causing it to explore irrelevant paths.

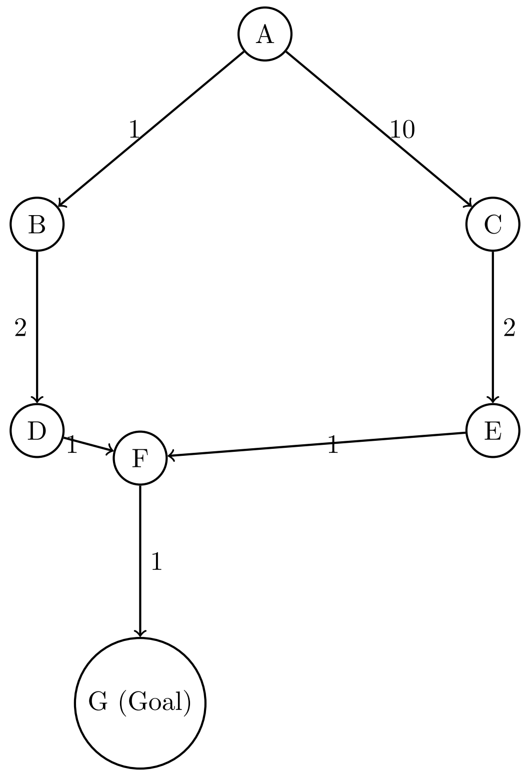
Heuristic Values

The heuristic for the nodes is:

$$h(A) = 10,\ h(B) = 8,\ h(C) = 1,\ h(D) = 8,\ h(E) = 1,\ h(F) = 5,\ h(G) = 0$$

Problem

- The heuristic overestimates costs in B and D, guiding A* toward the path $C \rightarrow E \rightarrow F \rightarrow G$, even though the shortest path is $A \rightarrow B \rightarrow D \rightarrow F \rightarrow G$. - The algorithm explores C and E, wasting time on an unnecessary detour.



To improve efficiency:

- Use bidirectional A* to reduce the number of explored nodes. Start one search from the start and another from the goal, meeting in the middle.
- Ensure the heuristic is admissible (never overestimates the cost) and consistent (satisfies the triangle inequality).In cases like grids, use Manhattan or Euclidean distances to reflect actual geometry.