Dijkstra's Algorithm on Example Graph

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

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

Steps:

1. Initialize $d[A] = 0$, $d[B] = d[C] = d[D] = d[E] = d[F] = \infty$. Priority queue: $Q = \{A(0)\}$.
2. Dequeue A, relax B ($d[B] = 1$) and C ($d[C] = 4$). Update $Q = \{B(1), C(4)\}$.
3. Dequeue B, relax D ($d[D] = 3$). Update $Q = \{C(4), D(3)\}$.
4. Dequeue D, relax F ($d[F] = 9$). Update $Q = \{C(4), F(9)\}$.
5. Dequeue C, relax E ($d[E] = 9$). Update $Q = \{F(9), E(9)\}$.
6. Dequeue F, and finish.

Shortest Paths and Costs:

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

4. A* Algorithm on Example Graph

Heuristic Values:

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

Steps:

1. Initialize $d[A] = 0$, $Q = \{A(10)\}$.
2. Dequeue A, relax B ($d[B] = 1,$ priority 9) and C ($d[C] = 4,$ priority 10). Update $Q = \{B(9), C(10)\}$.
3. Dequeue B, relax D ($d[D] = 3,$ priority 7). Update $Q = \{D(7), C(10)\}$.
4. Dequeue D, relax F ($d[F] = 9,$ priority 9). Update $Q = \{F(9), C(10)\}$.
5. Dequeue F, goal reached.

Shortest Path:

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

$$\text{Cost: } 9$$

5. Graphs Where A* Isn't Helpful

Graph 1: Sparse Graph with Poor Heuristics If the heuristic $h(u)$ significantly overestimates or underestimates distances, A* degenerates into Dijkstra's algorithm, exploring many unnecessary nodes.

Graph 2: Dense Graph with Uniform Weights If the graph is dense and the heuristic adds no useful information, A* may perform as poorly as Dijkstra.

Fixes:

- Use an admissible and consistent heuristic.
- Implement bidirectional search to reduce the search space.