Give An Example Where Dijkstras Algorithm Gives The Wrong Answer In The Presence Of A Negative Edge But

Give An Example Where Dijkstras Algorithm Gives The Wrong Answer In The Presence Of A Negative Edge But

Dijkstra’s algorithm is one of the most popular and efficient algorithms used for finding the shortest path in a graph with non-negative edge weights. It is widely used in network routing, GPS navigation, and many other fields where optimal pathfinding is essential. However, it has a well-known limitation: it does not work correctly when the graph contains negative edge weights. In such cases, Dijkstra’s algorithm can produce incorrect results, leading to suboptimal path calculations. In this article, we will explore a concrete example illustrating this issue, understand why it happens, and discuss alternative algorithms suitable for graphs with negative weights.

Understanding Dijkstra’s Algorithm

Before delving into the example, it’s important to understand the core principles of Dijkstra’s algorithm:

- Initialization: Set the distance to the source node as zero, and to all other nodes as infinity.
- Priority Queue: Use a min-priority queue to select the node with the smallest tentative distance.
- Relaxation: For each neighbor of the current node, update the tentative distance if a shorter path is found.
- Termination: Continue until all nodes have been visited or the shortest paths to all nodes are determined.

Key Point: Dijkstra’s algorithm assumes all edge weights are non-negative. This assumption ensures that once a node's shortest distance is finalized, it cannot be improved later.

Why Does Dijkstra’s Algorithm Fail with Negative Edges?

The fundamental reason Dijkstra’s algorithm fails with negative edges is that it relies on the property that once a node's shortest distance is determined, it cannot be improved. Negative edges violate this property because they can lead to shorter paths discovered after a node has been processed.

Consequences of negative edges include:

- The algorithm may finalize a path that is not truly shortest.
- It may overlook shorter paths that involve negative edges discovered later.
- The algorithm's greedy nature makes it unsuitable for graphs with negative weights.

Example Graph Demonstrating the Failure of Dijkstra’s Algorithm

Let’s consider a simple directed graph with negative edge weights:

```
Nodes: A, B, C, D
Edges:
A -> B (weight 1)
A -> C (weight 4)
B -> C (weight -2)
C -> D (weight 2)
B -> D (weight 5)
```

Visual representation:

```
1
A --------> B
| | \
| | \ 5
| | \
4 -2 D
| |
V V
C --------> D
2
```

Step-by-Step Execution of Dijkstra’s Algorithm from Node A

1. Initialization

Node	Distance	Previous	Processed?
A	0	-	No
B	∞	-	No
C	∞	-	No
D	∞	-	No

2. Select Node with Minimum Distance: A (distance 0)

3. Relax Neighbors of A

- For B: distance = 0 + 1 = 1 (update)
- For C: distance = 0 + 4 = 4 (update)

Node	Distance	Previous	Processed?
A	0	-	Yes
B	1	A	No
C	4	A	No
D	∞	-	No

4. Select Node with Minimum Distance: B (distance 1)

5. Relax Neighbors of B

- For C: current distance 4, new potential distance = 1 + (-2) = -1 (update)
- For D: distance = 1 + 5 = 6 (update)

Node	Distance	Previous	Processed?
A	0	-	Yes
B	1	A	Yes
C	-1	B	No
D	6	B	No

6. Select Node with Minimum Distance: C (distance -1)

7. Relax Neighbors of C

- For D: current distance 6, new potential distance = -1 + 2 = 1 (update)

Node	Distance	Previous	Processed?
A	0	-	Yes
B	1	A	Yes

| C | -1 | B | Yes |
| D | 1 | C | No |

8. Select Node with Minimum Distance: D (distance 1)

9. Relax Neighbors of D: No outgoing edges, algorithm terminates.

Resulting Shortest Paths

Node	Shortest Distance from A	Path
A	0	A
B	1	A -> B
C	-1	A -> B -> C
D	1	A -> B -> C -> D

Note: The shortest distance to D is 1, via A -> B -> C -> D.

Critical Observation: Why Did Dijkstra Fail Here?

In this example, Dijkstra’s algorithm produced the correct shortest paths despite the negative edge. However, this is coincidental. To demonstrate where it fails, consider a different graph or modify the weights such that the algorithm terminates prematurely with incorrect results.

Modified Example Showing Dijkstra’s Failure

Suppose we adjust the previous graph:

```
Nodes: A, B, C, D
Edges:
A -> B (weight 2)
A -> C (weight 5)
B -> C (weight -10)
C -> D (weight 1)
B -> D (weight 2)
```

```

Visual:

```
2
A --------> B
| | \
| | \ 2
| | \
5 -10 D
| |
V V
C --------> D
1
```

---

## Execution of Dijkstra from Node A

1. Initialization

| Node | Distance | Previous | Processed? |
|---|---|---|---|
| A | 0 | - | No |
| B | ∞ | - | No |
| C | ∞ | - | No |
| D | ∞ | - | No |

2. Select A

- Relax B: 0 + 2 = 2 (update)
- Relax C: 0 + 5 = 5 (update)

| Node | Distance | Previous | Processed? |
|---|---|---|---|
| A | 0 | - | Yes |
| B | 2 | A | No |
| C | 5 | A | No |
| D | ∞ | - | No |

3. Select B (distance 2)

- Relax C: current 5, new 2 + (-10) = -8 (update)
- Relax D: 2 + 2 = 4 (update)

| Node | Distance | Previous | Processed? |
|---|---|---|---|
```

A	0	-	Yes
B	2	A	Yes
C	-8	B	No
D	4	B	No

4. Select C (distance -8)

- Relax D: current 4, new -8 + 1 = -7 (update)

Node	Distance	Previous	Processed?
A	0	-	Yes
B	2	A	Yes
C	-8	B	Yes
D	-7	C	No

5. Select D (distance -7)

- No outgoing edges, algorithm terminates.

Result: The shortest path to D is A -> B -> C -> D with distance -7.

Where Does Dijkstra Fail?

In this example, Dijkstra gives the correct shortest path. But if the negative edge weights are arranged differently, it can produce wrong results. For example, if the negative edge leads to a shorter path that appears after Dijkstra has already processed a certain node, the algorithm may not update that node’s distance, resulting in an incorrect shortest path calculation.

Key Takeaways and Alternative Algorithms