

# Why Dijkstra's Algorithm Cannot Handle Negative Edge Weights

Manav Mantry

## Problem Statement

Explain why Dijkstra's algorithm cannot be applied to graphs with negative edge weights.

## Overview of Dijkstra's Algorithm

Dijkstra's algorithm is a greedy algorithm used to find the shortest paths from a single source vertex to all other vertices in a weighted graph.

The key assumption of the algorithm is that:

*Once a vertex is selected with the minimum tentative distance, its shortest path is finalized and will not change.*

This assumption holds only when all edge weights are non-negative.

## Effect of Negative Edge Weights

If a graph contains a negative edge weight, a shorter path to a vertex may be found *after* the vertex has already been processed.

Consider a vertex  $v$  whose shortest distance is assumed to be finalized. A negative-weight edge from another vertex may later reduce the distance to  $v$ , violating the greedy choice made earlier.

Thus, the core invariant of Dijkstra's algorithm is broken.

## Why the Greedy Strategy Fails

Dijkstra's algorithm relies on the fact that adding non-negative edge weights can only increase path length. With negative weights:

- A longer path may later become shorter
- Previously finalized distances can become incorrect

As a result, the algorithm may produce incorrect shortest paths.

## Illustrative Explanation

Suppose a node  $u$  is chosen as the closest unvisited node. If there exists a path from the source to another node  $v$ , and then a negative-weight edge from  $v$  to  $u$ , the total distance to  $u$  may become smaller than the one already fixed.

Since Dijkstra's algorithm does not revisit finalized nodes, it fails to account for this improvement.

## Conclusion

Dijkstra's algorithm cannot be applied to graphs with negative edge weights because its greedy assumption fails in such cases.

For graphs with negative edge weights, algorithms such as the **Bellman–Ford algorithm** should be used instead, as they correctly handle such cases.