Dijkstra’s Algorithm

What is Dijkstra’s Algorithm ?

Dijkstra's algorithm is a popular algorithm for finding the shortest path between a starting node and all other nodes in a weighted graph. It is named after the Dutch computer scientist Edsger W. Dijkstra, who proposed it in 1956.

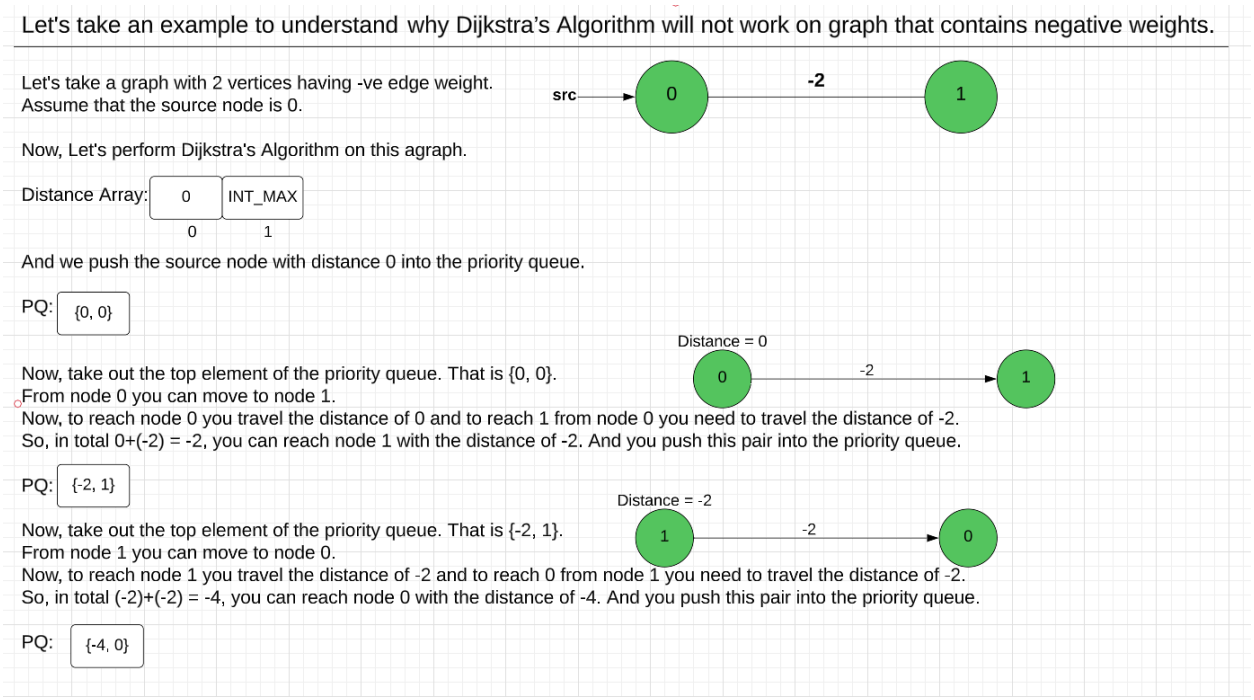
Dijkstra's algorithm can be implemented using various data structures, including:

1. Set.
2. Queue.
3. Priority Queue (Min-Heap).

> 💡 *Dijkstra's Algorithm will not work on the graphs that contains **negative edge weights**. Dijkstra's Algorithm is specifically designed for graphs that do not contain negative edge weights.*

> 💡 *If there are negative edge weights in the graph, then we can use other algorithms such as the Bellman-Ford Algorithm or the Floyd-Warshall Algorithm. These algorithms are able to handle negative edge weights and find the shortest path in such graphs.*

Reason Behind why Dijkstra’s Algorithm will not work on graph that contains negative weights:



So, because of negative edge weights, the distance reduces every time and it will stuck on an infinite loop. That's why Dijkstra's Algorithm will not work on graph that contains negative weights.

What is the significance of using a priority queue in Dijkstra's algorithm? What difference does it make if we use a normal queue?

A priority queue (MIN-HEAP) is a data structure that allows us to efficiently extract the smallest element. In Dijkstra's algorithm, we can use a priority queue to store the nodes we have processed so far, ordered by their distance from the source node. This allows us to quickly extract the node with the smallest distance and process it, without having to search through the entire list of nodes each time.

By always selecting the node with the shortest distance, we can be sure that we have found the shortest path to that node.

On the other hand, if we use a normal queue instead of a priority queue, we cannot guarantee that we are always selecting the node with the shortest distance.