Dijkstra's Algorithm

The Dijkstra algorithm is used to find the shortest path from one node to all others in a graph with non-negative weights.

Initially, we set the distance from the starting node to itself to 0. Then, we set the distance of all other nodes from the starting node to a very large number (theoretically infinite). We choose the node with the smallest distance that has NOT YET BEEN PROCESSED and update the distances of its neighboring nodes if smaller distances are found through this node. We repeat this until all nodes have been processed or until we find the path to the desired node.

Complexity $\Rightarrow O(|V|^2)$ using queue with an array
$\Rightarrow O(|E| \cdot \log|V|)$ using a queue with a binary heap
$\Rightarrow O(|E| + |V| \cdot \log|V|)$ using a queue with a Fibonacci heap

Bellman-Ford Algorithm

The Bellman-Ford algorithm is used to find the shortest path from one node to all others in a graph, even if it contains negative weights. Additionally, it checks for the presence of negative cycles and determines whether they affect the existence of a solution.

Initially, we set the distance from the starting node to itself to 0. Then, we set the distance of all other nodes from the starting node to a very large number (theoretically infinite). For every edge in the graph, we check if we can reduce the distance to the destination node via the source node. We repeat this process $|V| - 1$ times (where $|V|$ is the number of nodes). If we can still reduce the distance of any edge after this repetition, then a negative cycle exists, making it meaningless to search for the shortest path.

Complexity $\Rightarrow O(V \cdot E)$

Prim Algorithm

The Prim algorithm is used to find the minimum spanning tree (MST) of a connected, weighted graph. The MST is a subgraph that connects all nodes of the graph with the smallest possible total cost of edges.

Initially, we set the distance of all nodes from the starting node to infinity (theoretically), except for the distance from the starting node to itself, which is set to 0. Next, we add the node with the smallest cost to the MST and update the distances of its neighboring nodes. We repeat this until all nodes of the graph are included in the MST.

Complexity $\Rightarrow O(E \cdot \log(V))$ using a priority queue (where E is the number of edges in the graph)

Kruskal Algorithm

The Kruskal algorithm is used to find the minimum spanning tree (MST) of a connected graph. It works by selecting edges with the smallest weight, adding them one by one to the MST, provided that they do not form cycles.

Initially, we sort all edges in ascending order based on their weight. Then, for each edge, we check if the nodes u and v belong to the same component. If not, we add the edge to the MST and merge the components. We continue this process until the MST contains |V| - 1 edges.

Complexity $\Rightarrow O(E \cdot \log(E))$