Question#3:

1. Explain Bellman Ford algorithm.

Ans) Bellman Ford algorithm helps us find the shortest path from a vertex to all other vertices of a weighted graph. It is similar to Dijkstra's algorithm but it can work with graphs in which edges can have negative weights

Bellman Ford algorithm works by overestimating the length of the path from the starting vertex to all other vertices. Then it iteratively relaxes those estimates by finding new paths that are shorter than the previously overestimated paths. By doing this repeatedly for all vertices, we can guarantee that the result is optimized.

Steps of bellman ford algorithm:

1) Start with a weighted graph.

2) Choose Starting vertex and assign infinity path values to all other vertex.

3) Visit each edge and relax the path distance if they are inaccurate or don’t follow the condition:

For the edge from the vertex u to the vertex v, if d[u]+w(u,v)<d[v] is satisfied, update d[v] to d[u]+w(u,v)

The vertices *u* and *v* stand the neighbors in the graph and *d*[*u*] and *d*[*v*] stand the reaching cost to the vertices *u* and *v* respectively. Also, *w*(*u*,*v*) stands the weight of the edge from the vertex *u* to the vertex *v*

4) We need to do this V times, because in the worst case ,A vertex path length needs to be adjusted V times.

Bellman Ford Pseudocode

We need to maintain the path distance of every vertex. We can store that in an array of size v, where v is the number of vertices.

We also want to be able to get the shortest path, not only know the length of the shortest path. For this, we map each vertex to the vertex that last updated its path length.

Once the algorithm is over, we can backtrack from the destination vertex to the source vertex to find the path.

1. Differentiate between Bellman Ford and Dijikstra’s Algorithm.

Ans) Bellman Ford's algorithm and Dijkstra's algorithm are very similar in structure. While Dijkstra looks only to the immediate neighbors of a vertex, Bellman goes through each edge in every iteration.

Bellman-Ford algorithm is a single-source shortest path algorithm, so when you have negative edge weight then it can detect negative cycles in a graph.

The only difference between the two is that Bellman-Ford is also capable of handling negative weights whereas Dijkstra Algorithm can only handle positives

Dijkstra's algorithm greedily selects the minimum-weight node that has not yet been processed, and performs this relaxation process on all of its outgoing edges; in contrast, the Bellman–Ford algorithm simply relaxes all the edges, and does this |V | − 1 times, where |V | is the number of vertices in the graph. In each of these repetitions, the number of vertices with correctly calculated distances grows, from which it follows that eventually all vertices will have their correct distances. This method allows the Bellman–Ford algorithm to be applied to a wider class of inputs than Dijkstra

Dijkstra is however generally considered better in the absence of negative weight edges, as a typical binary heap priority queue implementation has O((|E|+|V|)log|V|) time complexity [A Fibonacci heap priority queue gives O(|V|log|V| + |E|)], while the Bellman-Ford algorithm has O(|V||E|) complexity

1. Solve the graph step by step taking A as source using Bellman Ford algorithm.
2. Explain the time complexity of Bellman Ford algorithm

Ans

||

||

V





