Vivekanand Education Society's
Institute of Technology
Approved by AICTE & Affiliated to University of Mumbai

**Artificial Intelligence and Data Science Department.**

AOA / Even Sem 2021-22 / Assignment 2 - Question 1.

YASH SARANG.
47 / D6AD.

ASSIGNMENT 2 - Question 1.

---

Comment on Dynamic Programming Approach. Explain in detail

- Single source shortest path: Bellman-Ford Algorithm.
- All pair shortest path: Floyd Warshall Algorithm.

---

***Single source shortest path: Bellman-Ford Algorithm.***

The Bellman-Ford algorithm is a very popular algorithm used to find the shortest path from one node to all the other nodes in a weighted graph. The Bellman-Ford algorithm is a single-source shortest path algorithm. This means that, given a weighted graph, this algorithm will output the shortest distance from a selected node to all other nodes.

It is very similar to the Dijkstra Algorithm. However, unlike the Dijkstra Algorithm, the Bellman-Ford algorithm can work on graphs with negative-weighted edges. This capability makes the Bellman-Ford algorithm a popular choice.

This algorithm takes as input a directed weighted graph and a starting vertex. It produces all the shortest paths from the starting vertex to all other vertices.

Now let's describe the notation that we used in the pseudocode.
The first step is to initialize the vertices.
The algorithm initially set the distance from starting vertex to all other vertices to infinity.
The distance between starting vertex to itself is 0.
The variable D[] denotes the distances in this algorithm.

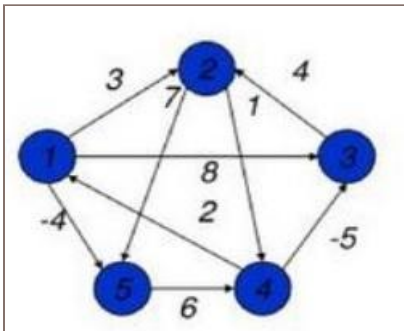
After the initialization step, the algorithm started calculating the shortest distance from the starting vertex to all other vertices.
This step runs $(|V| - 1)$ times.
Within this step, the algorithm tries to explore different paths to reach other vertices and calculates the distances.
If the algorithm finds any distance of a vertex that is smaller than the previously stored value then it relaxes the edge and stores the new value.
Finally, when the algorithm iterates $(|V| - 1)$ times and relaxes all the required edges, the algorithm gives a last check to find out if there is any negative cycle in the graph.

If there exists a negative cycle then the distances will keep decreasing. In such a case, the algorithm terminates and gives an output that the graph contains a negative cycle hence the algorithm can't compute the shortest path. If there is no negative cycle found, the algorithm returns the shortest distances.

The Bellman-Ford algorithm is an example of Dynamic Programming. It starts with a starting vertex and calculates the distances of other vertices which can be reached by one edge. It then continues to find a path with two edges and so on. The Bellman-Ford algorithm follows the bottom-up approach.

***All pair shortest path: Floyd Warshall Algorithm.***

The all-pair shortest path algorithm is also known as the Floyd-Warshall algorithm is used to find all pair shortest path problems from a given weighted graph. As a result of this algorithm, it will generate a matrix, which will represent the minimum distance from any node to all other nodes in the graph.




$$\begin{bmatrix} 0 & 1 & -3 & 2 & -4 \\ 3 & 0 & -4 & 1 & -1 \\ 7 & 4 & 0 & 5 & 3 \\ 2 & -1 & -5 & 0 & -2 \\ 8 & 5 & 1 & 6 & 0 \end{bmatrix}$$

At first, the output matrix is the same as the given cost matrix of the graph. After that, the output matrix will be updated with all vertices k as the intermediate vertex.

The time complexity of this algorithm is O(V3), where V is the number of vertices in the graph.

Input −

The cost matrix of the graph.

0 3 6 ∞ ∞ ∞ ∞

3 0 2 1 ∞ ∞ ∞

6 2 0 1 4 2 ∞

∞ 1 1 0 2 ∞ 4

∞ ∞ 4 2 0 2 1

∞ ∞ 2 ∞ 2 0 1

∞ ∞ ∞ 4 1 1 0

Output −

Matrix of all pair shortest path.

0 3 4 5 6 7 7
3 0 2 1 3 4 4
4 2 0 1 3 2 3
5 1 1 0 2 3 3
6 3 3 2 0 2 1
7 4 2 3 2 0 1
7 4 3 3 1 1 0

We initialize the solution matrix the same as the input graph matrix as a first step. Then we update the solution matrix by considering all vertices as intermediate vertex. The idea is to one by one pick all vertices and update all shortest paths which include the picked vertex as an intermediate vertex in the shortest path. When we pick vertex number k as an intermediate vertex, we already have considered vertices {0, 1, 2, .. k-1} as intermediate vertices. For every pair (i, j) of the source and destination vertices respectively, there are two possible causes.
1) k is not an intermediate vertex in shortest path from i to j. We keep the value of dist[i][j] as it is.
2) k is an intermediate vertex in shortest path from i to j. We update the value of dist[i][j] as dist[i][k] + dist[k][j] if dist[i][j] > dist[i][k] + dist[k][j]
The following figure shows the above optimal substructure property in the all-pairs shortest path problem.

**Time Complexity:** O(V^3)

---