Wednesday, April 16, 2025 8:47 PM

Floyd Warshall Algorithm :-

Floyd Warshall

Difficulty: **Medium** Accuracy: **32.89%** Submissions: **182K+** Points: **4** Average Time: **15m**

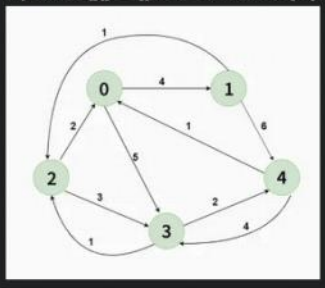
You are given an weighted **directed** graph, represented by an adjacency matrix, **dist[][]** of size **n x n**, where **dist[i][j]** represents the weight of the edge from **node i to node j**. If there is no direct edge, **dist[i][j]** is set to a large value (i.e., **10^8**) to represent infinity.
The graph may contain **negative edge weights**, but it does not contain any **negative weight cycles**.

Your task is to find the **shortest distance** between every pair of nodes **i** and **j** in the graph.

Note: Modify the distances for every pair **in place**.

Examples :

Input: dist[][] = [[0, 4, 10^8, 5, 10^8], [10^8, 0, 1, 10^8, 6], [2, 10^8, 0, 3, 10^8], [10^8, 10^8, 1, 0, 2], [1, 10^8, 10^8, 4, 0]]



Output: [[0, 4, 5, 5, 7], [3, 0, 1, 4, 6], [2, 6, 0, 3, 5], [3, 7, 1, 0, 2], [1, 5, 5, 4, 0]]

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

Explanation: Each cell dist[i][j] in the output shows the shortest distance from node i to node j, computed by considering all possible intermediate nodes.

Input: dist[][] = [[0, -1, 2], [1, 0, 10^8], [3, 1, 0]]

	0	1	2
0	0	-1	2
1	1	0	10^8
2	3	1	0

Output: [[0, -1, 2], [1, 0, 3], [2, 1, 0]]

	0	1	2
0	0	-1	2
1	1	0	3
2	2	1	0

Explanation: Each cell dist[i][j] in the output shows the shortest distance from node i to node j, computed by considering all possible intermediate nodes.
From 2 to 0 shortest distance should be 2 by following path 2 -> 1 -> 0
From 1 to 2 shortest distance should be 3 by following path 1 -> 0 -> 2

Constraints:
$1 \le$ dist.size() ≤ 100
$-1000 \le$ dist[i][j] ≤ 1000

Try more examples

```
class Solution{
  public:
  void floydWarshall(vector<vector<int>> &dist){
      int n=dist.size();

      for(int k=0;k<n;k++){
          for(int i=0;i<n;i++){
              for(int j=0;j<n;j++){
if(dist[i][k]!=1e8 && dist[k][j]!=1e8 &&
dist[i][k]+dist[k][j]<dist[i][j]){
```

dist[i][j] = dist(i)[k] + dist(k)[j];