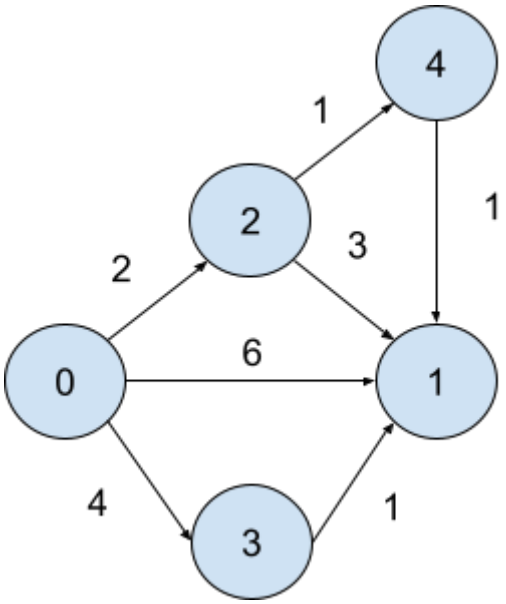
MEDIUM

Floyd Warshall Algorithm

Intuition

Its different from the Dijkstra and Bellman Ford as they are for the single source shortest path while Floyd warshall is for All pair/ Multi source shortest path algorithm. It can help you detect negative cycles as well.

Eg.



Eg.

0 to 1 :

Distance[0][1] : 0 - 1 : 6
0 - 2 - 1 : 5
0 - 3 - 1 : 5
0 - 4 - 1 : 4 // assume we have computer 0 - 4
Therefore we will use path from 4
Distance[0][1] = 4

0 - 4 - 1 : ([0][2] + [2][4])// pre-computation + [4][1]

We will be using Floyd Warshall to store this particular graph

0	2	inf	inf
1	0	3	inf
inf	inf	0	inf
3	5	4	0

// initially all the elements can go to themselves with a distance 0 and assign other with adjacent values and rest with infinity as we do in a adjacency matrix

0	2	inf	inf
1	0	3	inf
inf	inf	0	inf
3	5	4	0

// we will move like this [0][1] = [0][0]+[0][1] and [1][0] = [0][0]+[1][0]

According to this we can exactly copy the values in the 0th row and column and use them to reach some other positions

Move via 0 :

0	2	inf	inf
1	0	3	inf
inf	inf	0	inf
3	5	4	0

Move via 1 :

0	2	5	inf
1	0	3	inf
inf	inf	0	inf
3	5	4	0

….
….

Move via 4 : This will contain the shortest path from every node to other node

How to detect a negative cycle ?

If costing of any node to node itself is less than 0 then we can say that there is a negative cycle

Approach

- Calculate the number of nodes
- Traverse for all the elements in the matrix :
    - If weight given from node to other node is -1 ie. unreachable :
        - Convert the node to infinity ie. 1e9
    - If the node is going to node itself :
        - Mark it to go with 0 distance cost
- Traverse for number of nodes ie. via :
    - Traverse for row ie. i :
        - Traverse for cols ie. j :
            - Update the weight in the i,j with minimum of either via distance or the distance itself as matrix[u][v] = min(matrix[u][v],matrix[u][via],matrix[via][v])
- Traverse for all elements in matrix :
    - Convert the infinite elements to -1

Function Code

```
void shortest_distance(vector<vector<int>>&matrix){
          // Calculating the number of nodes
          int n = matrix.size();
          // going across entire matrix and converting the -1 to infinite
values
          for(int i=0;i<n;i++)
          {
              for(int j=0;j<n;j++)
              {
                  // if unreachable we will mark it as infinity
                  if(matrix[i][j]==-1)
                  {
                      matrix[i][j] = 1e9;
                  }
                  // checking if the node is going to itself we will mark
it to have a distance 0
```

```
                if(i==j)matrix[i][j]=0;

            }
        }
        // performing floyd warshall
        // loop for via
        for(int via = 0;via<n;via++)
        {
            // loop for row
            for(int i=0;i<n;i++)
            {
                // loop for column
                for(int j=0;j<n;j++)
                {
                    // updating the weight with minimum value either via
or original

matrix[i][j]=min(matrix[i][j],matrix[i][via]+matrix[via][j]);
                }
            }
        }

        // converting the unreachables to infinity
        for(int i=0;i<n;i++)
        {
            for(int j=0;j<n;j++)
            {
                if(matrix[i][j]==1e9)
                {
                    matrix[i][j]=-1;
                }
            }
        }
    }
```

Time Complexity

O(N^3)