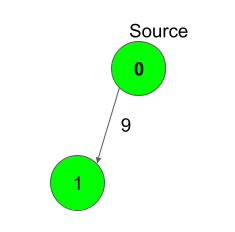
**Distance from the Source (Bellman-Ford Algorithm)**

**Medium**

Given a weighted, directed and connected graph of V vertices and E edges, Find the shortest distance of all the vertex's from the source vertex S.  
**Note:**If the Graph contains a negative cycle then return an array consisting of only -1.

**Example 1:**

**Input:**



**E** = [[0,1,9]]

**S** = 0

**Output:**

0 9

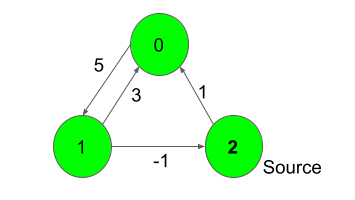
**Explanation**:

Shortest distance of all nodes from

source is printed.

**Example 2:**

**Input:**



**E** = [[0,1,5],[1,0,3],[1,2,-1],[2,0,1]]

**S** = 2

**Output:**

1 6 0

**Explanation**:

For nodes 2 to 0, we can follow the path-

2-0. This has a distance of 1.

For nodes 2 to 1, we cam follow the path-

2-0-1, which has a distance of 1+5 = 6,

**Expected Time Complexity:** O(V\*E).  
**Expected Auxiliary Space:** O(V).

**Constraints:**  
1 ≤ V ≤ 500  
1 ≤ E ≤ V\*(V-1)  
-1000 ≤ adj[i][j] ≤ 1000  
0 ≤ S < V

**Company Tags**

[**Amazon**](https://practice.geeksforgeeks.org/explore/?company%5b%5d=Amazon) [**Microsoft**](https://practice.geeksforgeeks.org/explore/?company%5b%5d=Microsoft)

//{ Driver Code Starts

import java.util.\*;

import java.io.\*;

import java.lang.\*;

class DriverClass {

public static void main(String args[]) throws IOException {

BufferedReader read =

new BufferedReader(new InputStreamReader(System.in));

int t = Integer.parseInt(read.readLine());

while (t-- > 0) {

String str[] = read.readLine().trim().split(" ");

int V = Integer.parseInt(str[0]);

int E = Integer.parseInt(str[1]);

ArrayList<ArrayList<Integer>> edges = new ArrayList<>();

int i = 0;

while (i++ < E) {

String S[] = read.readLine().trim().split(" ");

int u = Integer.parseInt(S[0]);

int v = Integer.parseInt(S[1]);

int w = Integer.parseInt(S[2]);

ArrayList<Integer> t1 = new ArrayList<>();

t1.add(u);

t1.add(v);

t1.add(w);

edges.add(t1);

}

int S = Integer.parseInt(read.readLine());

Solution ob = new Solution();

int[] ptr = ob.bellman\_ford(V, edges, S);

for (i = 0; i < ptr.length; i++) System.out.print(ptr[i] + " ");

System.out.println();

}

}

}

// } Driver Code Ends

// User function Template for Java

class Solution {

static int[] bellman\_ford(int V, ArrayList<ArrayList<Integer>> edges, int S) {

int[] fail=new int[1];

int[] dist=new int[V];

Arrays.fill(dist, 100000000);

dist[S]=0;

int k=0;

for(int i=0;i<V-1;i++){

for(int j=0;j<edges.size();j++){

int srs=edges.get(j).get(0);

int dest=edges.get(j).get(1);

int wt=edges.get(j).get(2);

if(dist[srs]!=100000000 && dist[srs]+wt<dist[dest]){

dist[dest]=dist[srs]+wt;

}

}

}

for(int j=0;j<edges.size();j++){

int srs=edges.get(j).get(0);

int dest=edges.get(j).get(1);

int wt=edges.get(j).get(2);

if(dist[srs]!=100000000 && dist[srs]+wt<dist[dest]){

Arrays.fill(fail, -1);

return fail;

}

}

return dist;

}

}