Bellman Ford Algorithm:

1. Used to find shortest distances from a source to all the other vertices.
2. V : No of vertices
3. E : No of edges
4. Time complexity O(VE)
5. |V|-1 iterations required
6. All edges checked in each of the iteration.
7. ith iteration uses i edges between (u,v) to minimise the distances.
8. If at |V| iteration any distance reduces then we have a negative cycle.
9. Note the following code works for only one source.
10. To detect all the negative cycles in a graph add a dummy node.
    CSES Cycle Finding

```
const int N = 2505;
const ll INF = LLONG_MAX;
ll dis[N];

struct edge
{
    ll u, v, w;
};
vector<edge> edges;

int main()
{

    kira;
    ll n, m;
    cin >> n >> m;
    ll eu, ev, ew;

    forz(i, m)
    {
        cin >> eu >> ev >> ew;
        edges.pb({eu, ev,ew});
    }

    for (int i = 1; i <= n; i++)
    {
        dis[i] = INF;
    }
    dis[1] = 0;

    for (int i = 1; i <= n - 1; i++)
    {
        for (auto x : edges)
        {
            ll u = x.u;
            ll v = x.v;
```

```
            ll w = x.w;
            if (dis[u] != INF && dis[u] + w < dis[v])
            {
                dis[v] = dis[u] + w;
            }
        }
    }

    for (auto x : edges)
    {
        ll u = x.u;
        ll v = x.v;
        ll w = x.w;
        if (dis[u] != INF && dis[u] + w < dis[v])
        {
            cout << -1;
            return 0;
        }
    }

    cout << dis[n];
    run_time();
    return 0;
}
```