Bellman Ford

- It is a “single source shortest path” algorithm.
- Time Complexity: **O(V*E)**
- Dijkstra’s is much faster when we use a heap priority Queue.
  But can fail when the graph G(V,E) has negative edge weights!
- Bellman Ford can be used to: detect negative cycles.

Algorithm():

```
   input: V(num of vertices), E(num of edges), s the source node,
          D(array of size V) that tracks the best distance from s to any
           other node.

   > Set every entry in D to +oo // distance unknown
   > D[s] = 0 ; // we’re already there
   > // Relax every edge V-1 times
     for(i=0; i<V-1; i++){
        for(edge in graph.edges){
            // Relax edge(update D with a shorter path)
            if(D[edge.from] + edge.cost < D[edge.to]){
              D[edge.to] = D[edge.from] + edge.cost ;
            }
         }
      }
    > // Repeat to find nodes part or caught up in a negative cycles
      for(i=0; i<V-1; i++){
        for(edge in graph.edges){
            // Relax edge(update D with a shorter path)
            if(D[edge.from] + edge.cost < D[edge.to]){
              D[edge.to] = -oo ;
            }
         }
      }
```

Comment: You may be able to detect the cycles at the 1st loop !
              But worst case is V-1 loops!