**Detect Cycle in a Directed Graph**

Given a directed graph, check whether the graph contains a cycle or not. Your function should return true if the given graph contains at least one cycle, else return false.

To detect a back edge, keep track of vertices currently in the recursion stack of function for DFS traversal. If a vertex is reached that is already in the recursion stack, then there is a cycle in the tree.

* **Algorithm:**
  1. Create the graph using the given number of edges and vertices.
  2. Create a recursive function that initializes the current index or vertex, visited, and recursion stack.
  3. Mark the current node as visited and also mark the index in recursion stack.
  4. Find all the vertices which are not visited and are adjacent to the current node. Recursively call the function for those vertices, If the recursive function returns true, return true.
  5. If the adjacent vertices are already marked in the recursion stack then return true.
  6. Create a wrapper class, that calls the recursive function for all the vertices and if any function returns true return true. Else if for all vertices the function returns false return false.

<https://github.com/hareramcse/Datastructure/blob/master/Graph/src/com/hs/cycle/DetectCycleInDirectedGraph.java>

**Detect cycle in an undirected graph**

Given an undirected graph, how to check if there is a cycle in the graph?   
  
**Algorithm:**

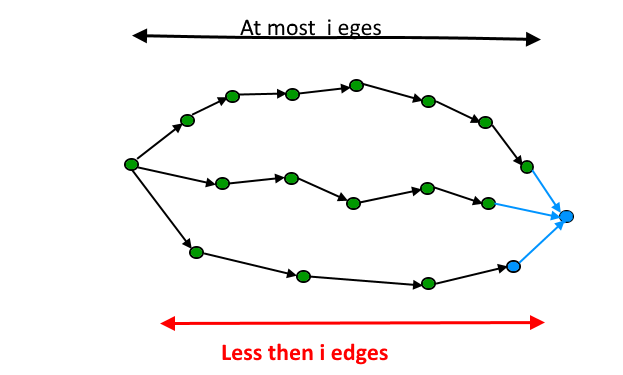
1. Create the graph using the given number of edges and vertices.
2. Create a recursive function that have current index or vertex, visited array and parent node.
3. Mark the current node as visited.
4. Find all the vertices which are not visited and are adjacent to the current node. Recursively call the function for those vertices, If the recursive function returns true return true.
5. If the adjacent node is not parent and already visited then return true.
6. Create a wrapper class, that calls the recursive function for all the vertices and if any function returns true, return true.
7. Else if for all vertices the function returns false return false.

<https://github.com/hareramcse/Datastructure/blob/master/Graph/src/com/hs/cycle/DetectCycleInUndirectedGraph.java>

**Detect a negative cycle in a Graph | (Bellman Ford)**

We are given a directed graph. We need to compute whether the graph has a negative cycle or not. A negative cycle is one in which the overall sum of the cycle becomes negative.

**How does it work?**   
As discussed, the [Bellman-Ford algorithm](https://www.geeksforgeeks.org/dynamic-programming-set-23-bellman-ford-algorithm/), for a given source, first calculates the shortest distances which have at most one edge in the path. Then, it calculates the shortest paths with at-most 2 edges, and so on. After the i-th iteration of the outer loop, the shortest paths with at most i edges are calculated. There can be a maximum |V| – 1 edge on any simple path, that is why the outer loop runs |v| – 1 time. If there is a negative weight cycle, then one more iteration would give a short route.



**How to handle a disconnected graph (If the cycle is not reachable from the source)?**   
The above algorithm and program might not work if the given graph is disconnected. It works when all vertices are reachable from source vertex 0.  
To handle disconnected graphs, we can repeat the process for vertices for which distance is infinite.

<https://github.com/hareramcse/Datastructure/blob/master/Graph/src/com/hs/cycle/DetectNegativeCycleBelmanFord.java>

**Detecting negative cycle using Floyd Warshall**

We are given a directed graph. We need compute whether the graph has negative cycle or not. A negative cycle is one in which the overall sum of the cycle comes negative

Distance of any node from itself is always zero. But in some cases, as in this example, when we traverse further from 4 to 1, the distance comes out to be -2, i.e. distance of 1 from 1 will become -2. This is our catch, we just have to check the nodes distance from itself and if it comes out to be negative, we will detect the required negative cycle.

<https://github.com/hareramcse/Datastructure/blob/master/Graph/src/com/hs/cycle/DetectNegativeCycleFloyedWarshall.java>