**Minimum cut (Nagamochi-lbaraki):**

This Algorithm use maximum adjacency (MA) ordering of vertices to compute the edge connectivity.

Let us Consider graph G and we will merge nodes x1 and x2 recursively until tow nodes are left but need to be careful to watch loop and exclude them

Two nodes should be found which has minimum edge connectivity and need to remove it by merging two nodes and we can do recursively.

**Implementation**:

Maximum adjacency:

We need to select random node and store it in temporary node store

Then we go through all the nodes and see the number of edges they are connected to all the vertices and the maximum number of edges are stored at next position. Then again recursively find all the nodes with the maximum number of edges and store them in temp store. It looks like nodes with number of edges are stored in sorted order in decreasing order. The last two nodes will have minimum edge connectivity and we need to merge them.

Merge Graph:

We will merge the two nodes we got from maximum adjacency and will look to avoid for self loops.

EdgeConnectivity:

We need to recursively find maximum adjacency and merge the graph, until there are only two nodes left and return the total which is total min cut.

**PseudoCode:**

Input:

E: Number of edges

N: Number of nodes

Output:

Minimum cut

G: graph

edgeConnectivity() {

//two nodes left

if(N == 2){

return G[0][1]

} else if(G is connected){

//if the graph is connected we will excute the code

We will get maximum adjacenecy for nodes first and second

totalAB =. Maximumadjacency()

//Merge the graph with N-1 edges

mergeGraph()

return min(totalAB, edgeConnectivity())

}else {

//if the graph is disconnected

Return 0;

}

Maximumadjacency() {

Total = 0

Temp\_store[]

Temp\_store[0] = random\_node

for(int i=0;i<N;i++) {

if node has maximum edges

temp\_store[i] = node

}

First = temp\_store[N-1]

Second = temp\_store[N-2]

For(int j=0;j<N;j++) {

Total += edge weights of First to merge

}

Return total

}

MergeNode() {

Newgraph nG has N-1 nodes

For(int i=0;i<N;i++) {

For(int j=0;j<N;j++) {

Merge graph with no loops

}

}

Return new Graph

}