```
class graph:
    size,
    root

#this is the node object
class node:
    element,
    parent,
    children, #iterative
    size, # size of tree from below node
    discovery, # discovery time
    finish, # finish time
    downUp, # down&up value
    cutVertex, # true/false if cut vertex
    breakableNode, # true/false if a cut vertex will immediately make this a seperate forest
    visited,
    disconnectingPower

#DFS runs in O(m) since it is a connected graph

def DFS(G):
    for vertex u in G:
        u.visted = False
        u.parent = None
    time = 0
    for vertex u in G:
        if u.visted == False:
            u.size = DFS_VISIT(u)
    return root

def DFS_VISIT(u):
    time = time + 1
    u.discovery() = time
    u.visited() = True
    for children v in u:
        if v.visted() == False:
            v.parent() = u
            DFS_VISIT(v)
    time = time + 1
    u.finish = time
    u.size = the number of nodes below the current node



#Cut Verticies Algorithm: runs O(n^2)

def getCutVertex(G):
    root = DFS(G) # run DFS on G to compute T and d[u for eac u in V
    if root.children.count > 1:
        root.cutVertex = True
        for child in root.children:
            child.breakableNode = True
    for v internal node of T:
        for each child v of u in T:
            if v.downUp = u.discovery:
                v.cutVertex = True
                v.breakableNode = True
    return G

# this function will calculate the disconnecting power
def DisconnectingPower():
    G = getCutVertex()
    for vertex in G:
        if vertex.cutVertex == False:
            vertex.disconnectingPower = 0
        else:
            tempCount = G.size - 1 # take into account the removed node
            tempArray = []
            for child in vertex.children:
                if child.breakableNode:
                    tempArray.append(child.size) # append smaller forest size into array
                    G.size = G.size - child.size # remove all nodes below the child (inclusive) to get the larger forest
                vertex.disconnectingPower = sum of the multiplication of each array element in tempArray x tempCount
    return G
```