**I did everything for minimum and core and answer question about AP and MST also I used the iterative version of the articulation points algorithm and used Kruskal's algorithm with the disjoint set data structure.**

Iterative AP Algorithm pesucode:

**IAP**(){

Initialise depth(node) = Integer max value

HashSet APs = New HashSet

Randomly select a node as the root node,

set depth(root) = 0;

numSubTrees = 0;

for (each neighbour of root) {

if (depth(neighbour) = Integer max value) {

iterArtPts(neighbour, 1, root);

numSubTrees ++;

}

if (umSubTrees > 1) {

add root into APs;

}

**iterArtPts**(Node firstNoode, int depth, Node root){

Initialise stack ;

stack.add(new Articulation(firstNode, depth, root));

while (stack is not empty){

Node nodeArt = stack.peek().firstNode;

int depthArt = stack.peek().Depth;

Node parentArt = stack.peek().root;

if (nodeArt.depth == Integer.MAX\_VALUE) {

depth(nodeArt) =depth\*,

reachBack(depthArt)=depth\*;

nodeArt.reachBack=depthArt;

children(nodeArt) = all the neighbours of n\* except parent\*;

}

else if (nodeArt.Children.size() != 0){

Node child = nodeArt.Children.remove(0);

if(child.depth < Integer.MAX\_VALUE){

nodeArt.reachBack=Math.min(child.depth,nodeArt.reachBack);

}

else{

stack.push(new Articulation(child, depthArt+1, nodeArt));

}

}

else{

if(nodeArt.nodeID is not equal firstNode.nodeID ){

parentArt.reachBack = Min(nodeArt.reachBack, parentArt.reachBack);

if(nodeArt.reachBack>= parentArt.depth){

AP.add(parentArt);

}

}

stack.pop();

}

}

}

Kruskal’s Algorithm Pesucode:

**KA**(){

Set forest as N node sets

Set fringe as a priority queue of all the edges 〈n1, n2, length〉;

Set tree as an empty set of edges;

for( graph’s Nodes n) {

makeSet(n);

}

while(forest contains more than one tree and edges is not empty ){

f = fringe.poll();

if (*findRoot*(f.start)!= *findRoot*(f.end)){

*Union*(f.start,f.end);

tree.add(f);

}

}

return tree

}

**void makeSet**(Node node){

node.Parent = node;

node.Depth=0;

*Forest*.add(node);

}

**Node Find**(Node node){

if (node.Parent == node){return node;}

else{

root =Find(node.Parent);

return root;

}

}

**Void Union**(Node x,Nodey){

Node xRoot = *findRoot*(x);

Node yRoot = *findRoot*(y);

if (xRoot == yRoot ){

return ;

}

else {

if(xRoot.Depth<yRoot.Depth){

xRoot.Parent = yRoot;

*Forest*.remove(xRoot);

}

else{

yRoot.Parent = xRoot;

*Forest*.remove(yRoot);

if (xRoot.Depth == yRoot.Depth){

xRoot.Depth++;

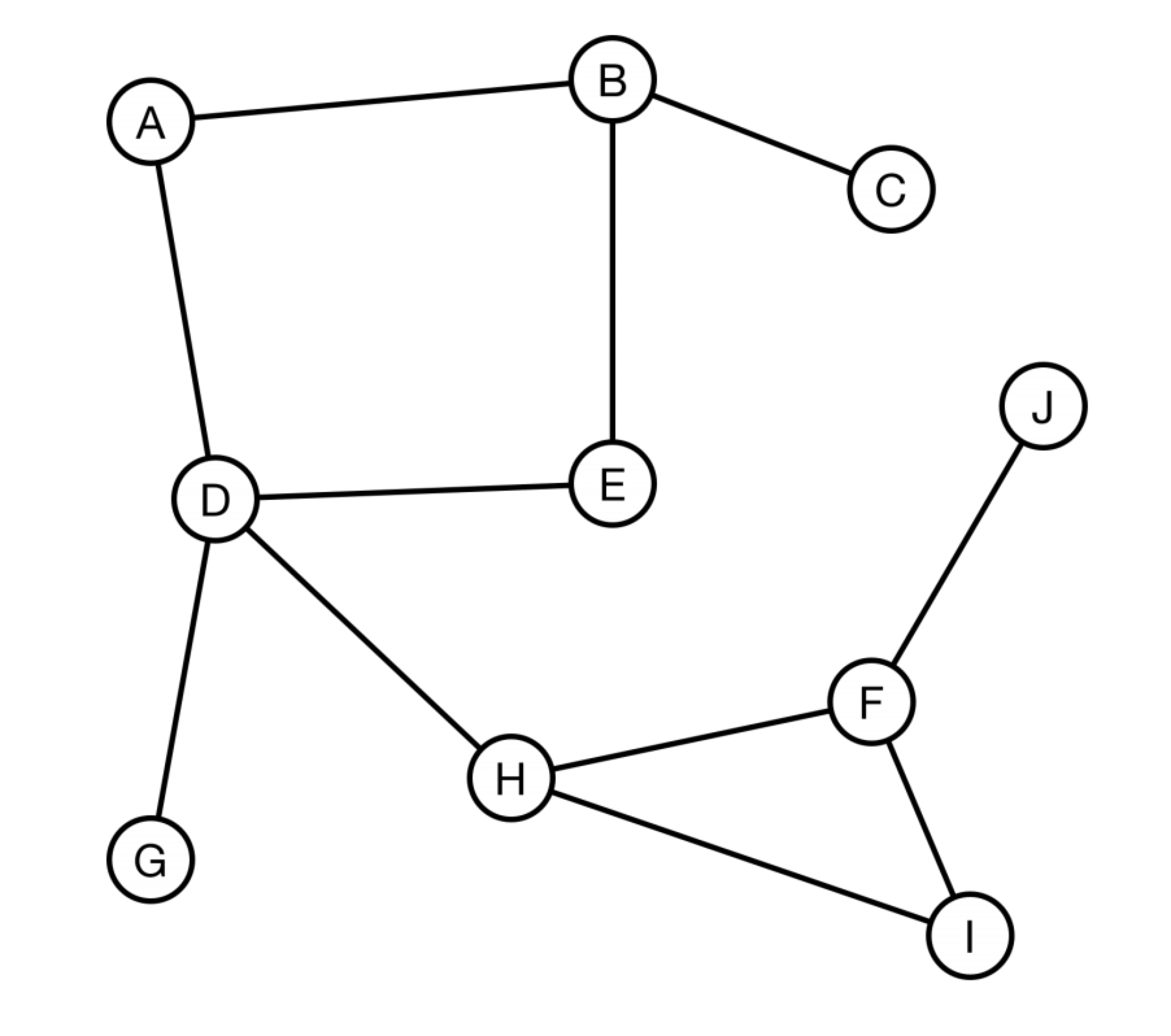
}

}

}

}

Question 1 :

(1)

A: Depth:0

B: Depth:1 reachBack:1,0(1 changed to 0 because E has Parent B,0<1)

C:Depth:2 reachBack:2(2>1, put B into APs)

E:Depth:2 reachBack:2,0(2 changed to 0 because D has Parent E,0<2)

D:Depth:3 reachBack:3,0(3 changed to 0 because D has child A,0<3)

G:Depth:4 reachBack:4(4>3,put D into APs)

H:Depth:4 reachBack:4

F:Depth:5 reachBack:5,4(4=4,put H into APs)

(5 changed to 4 because I has Parent F,5<6)

I:Depth:6 reachBack:6,4 (6 changed to 4 because I has child H,4<6)

J:Depth:6 reachBack:6(6>5,add F into APs)

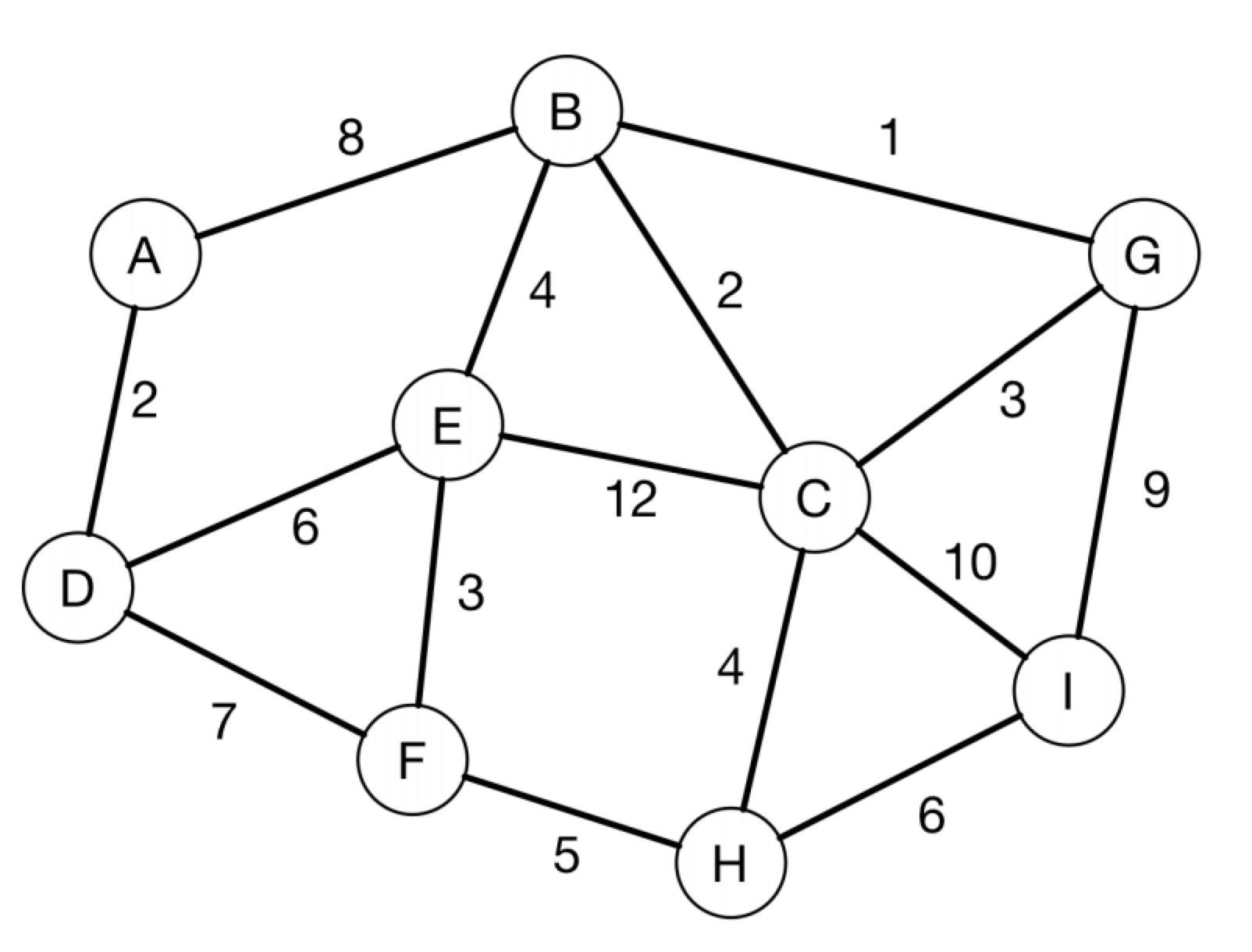
(2)

AP: B((2>1 (reachBack(C) > depth(B)),D(4>3,reachBack(G) > depth(D)),H(4=4,reachBack(F) = depth(H)),F (6>5,reachBack(J) > depth(F))

Question 2:

(1)

fringe: {D AD2} {B AB8} {E DE6} {F DF 7} {B EB 4} {C EC12} {F EF 3} {H FH5}{C HC4} {B CB2}{G CG3} {I CI10} {G BG1} {I GI9}

**AD2**

**DE6**

**EF3**

**FH5**

**CH4**

**BC2**

**BG1**

**GI9**

(2)

fringe : BG1, AD2, BC2, CG3,EF3,CH4,BE4,FH5,DE6,HI6,DF7,AB8,GI9,CI10,CE12

