Name: Elgene Menon Leo Anthony

Username: leoanelge

Student Id: 300492604

The report should: –

1) **Describe what your code does and does not do (e.g., which stages did you do).**

   I have completed this assignment from minimum until challenge.

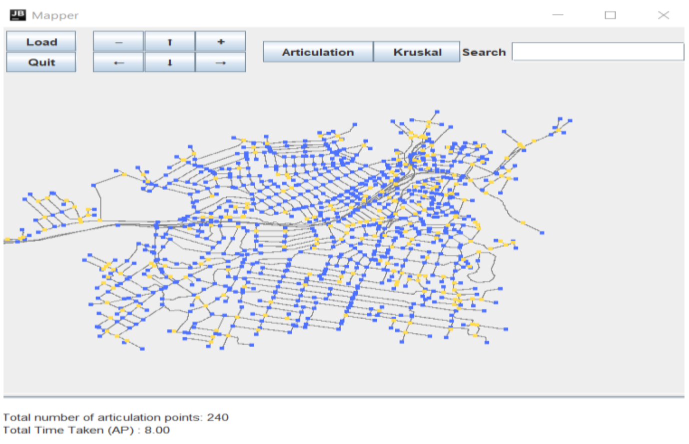
   For minimum part, my code will print out the total number of articulation points from both small and large data, 240 points and 10853 points, respectively. It has a button for articulation. Besides this, it will highlight all the articulations node, this is done by using the iterative method. It will also print out the time taken to run this algorithm. My program does not run articulation in recursive way.

   For core part, I used Kruskal's algorithm to find the minimum spanning tree. It has a Kruskal button. My code will find all the MSTs for all the connected components and highlight all the segments including the disconnected graph. Besides this, it will print out the time taken to run this algorithm and print out the total of Kruskal's points for both small and large data, 1079 and 35371, respectively. My code does not do the prim's algorithm for MST.

   For challenge, my code will perform disjoint set data structure for Kruskal's algorithm. It will have a makeForestSet (), findRoot () and union () methods

2) **Outline how you tested that your program worked.**

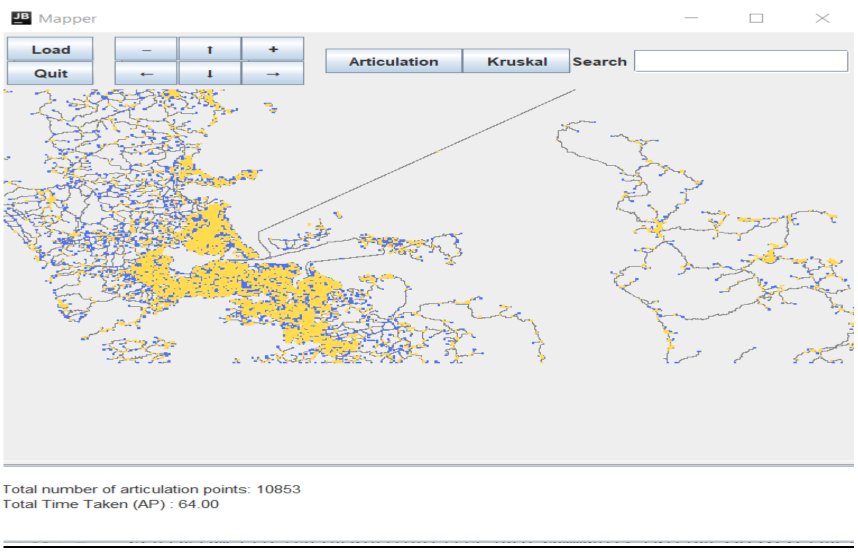
   <u>Testing for articulation points for small data.</u>



```
Total number of articulation points: 240
Total Time Taken (AP) : 8.00
```

This test shows that the number of articulations is 240 and it highlights the all the articulation nodes in small graph.

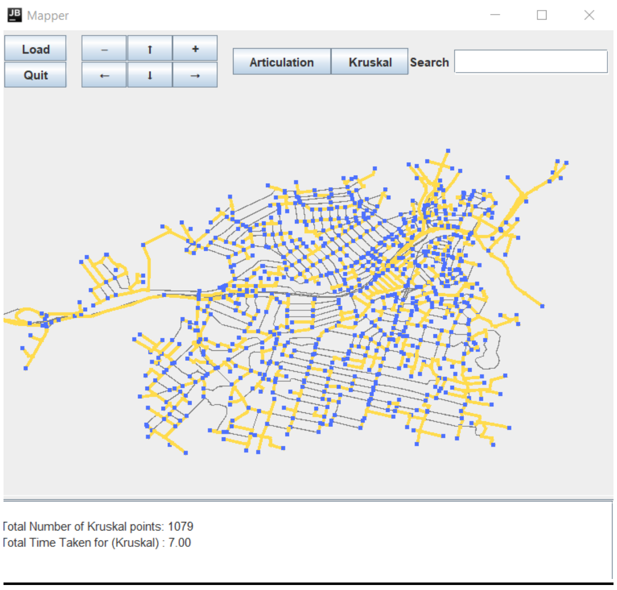
Testing for articulation points for large data.



```
Total number of articulation points: 10853
Total Time Taken (AP) : 64.00
```

The test above shows the number of articulations is 10853 and it highlights all the articulation nodes in large graph.

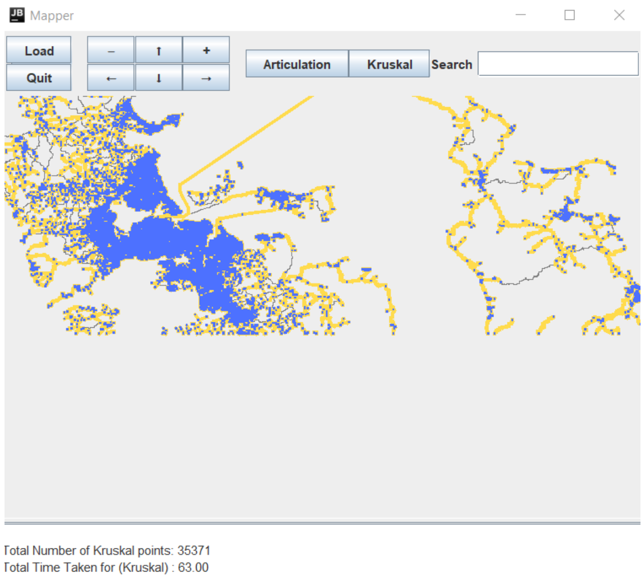
Testing for Kruskal's algorithm for MST for small data.



```
Total Number of Kruskal points: 1079
Total Time Taken for (Kruskal) : 7.00
```

The test above shows the total number of Kruskal point is 1079 and it highlights all the minimum spanning tree connection in small graph.

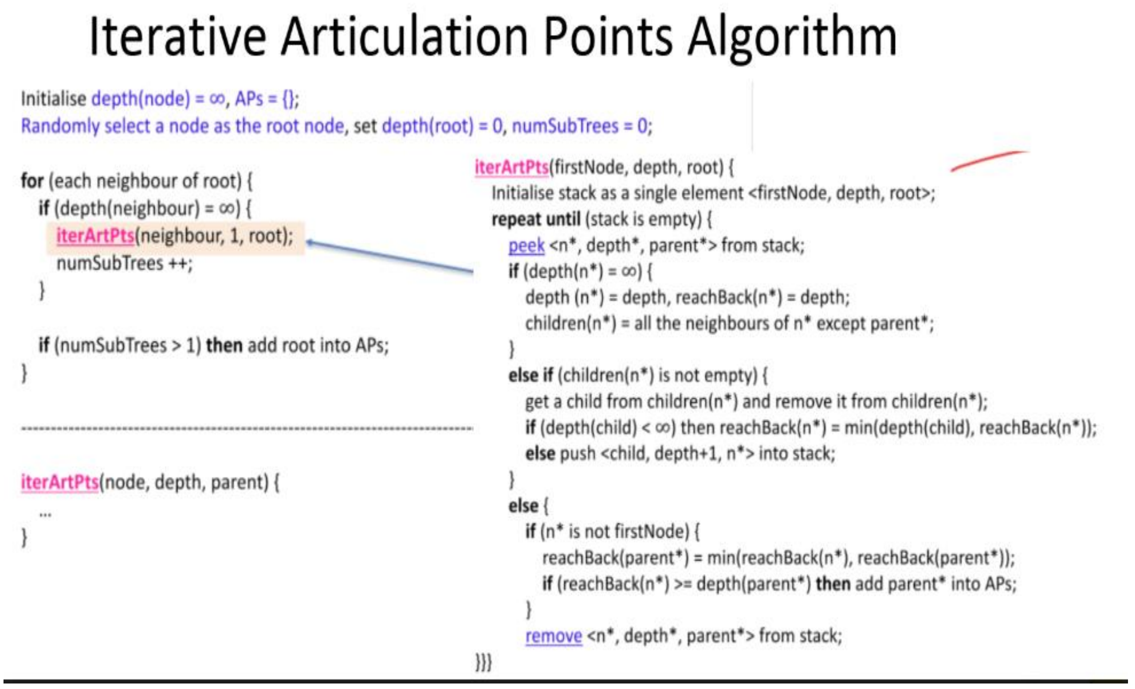
Testing for Kruskal's algorithm for MST for large data.



```
Total Number of Kruskal points: 35371
Total Time Taken for (Kruskal) : 63.00
```

The test above shows the total number of Kruskal point is 35371 and it highlights all the minimum spanning tree connection in large graph.

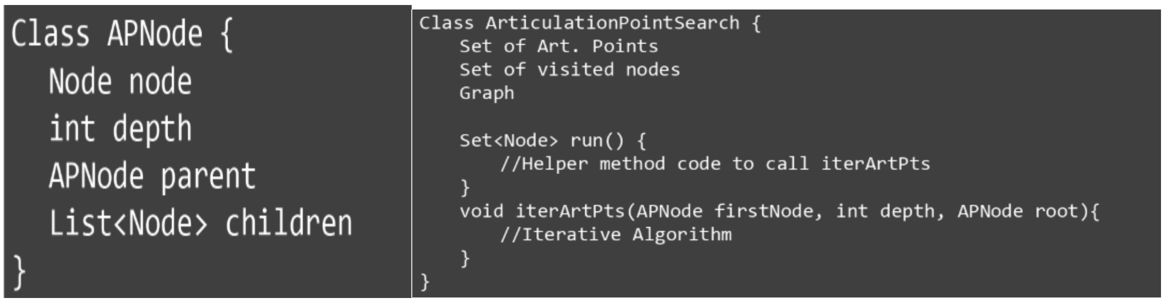
3) Give a detailed pseudocode for the algorithms (articulation points and minimum spanning tree).



Iterative Articulation Points Algorithm

```
Initialise depth(node) = ∞, APs = {};
Randomly select a node as the root node, set depth(root) = 0, numSubTrees = 0;

for (each neighbour of root) {
  if (depth(neighbour) = ∞) {
    iterArtPts(neighbour, 1, root);
    numSubTrees ++;
  }

  if (numSubTrees > 1) then add root into APs;
}

-----------------------------------------------------------

iterArtPts(node, depth, parent) {
  ...
}

iterArtPts(firstNode, depth, root) {
  Initialise stack as a single element <firstNode, depth, root>;
  repeat until (stack is empty) {
    peek <n*, depth*, parent*> from stack;
    if (depth(n*) = ∞) {
      depth (n*) = depth, reachBack(n*) = depth;
      children(n*) = all the neighbours of n* except parent*;
    }
    else if (children(n*) is not empty) {
      get a child from children(n*) and remove it from children(n*);
      if (depth(child) < ∞) then reachBack(n*) = min(depth(child), reachBack(n*));
      else push <child, depth+1, n*> into stack;
    }
    else {
      if (n* is not firstNode) {
        reachBack(parent*) = min(reachBack(n*), reachBack(parent*));
        if (reachBack(n*) >= depth(parent*) then add parent* into APs;
      }
      remove <n*, depth*, parent*> from stack;
}}}
```

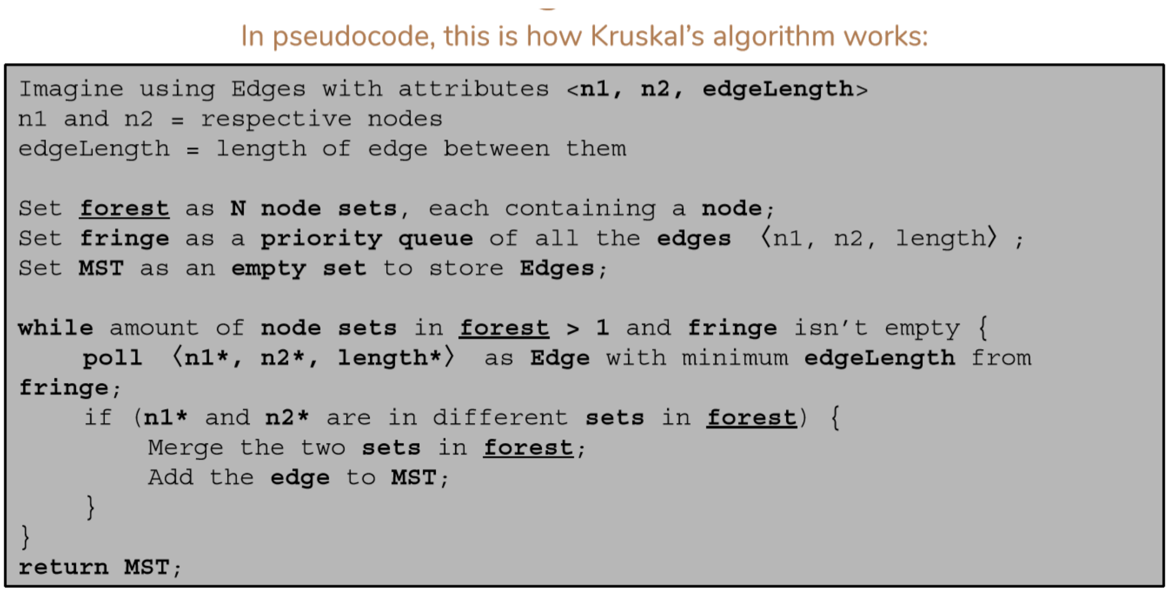
Figure 1: The initial attempt of Articulation using iterative(minimum)



```
Class APNode {
   Node node
   int depth
   APNode parent
   List<Node> children
}
```

```
Class ArticulationPointSearch {
    Set of Art. Points
    Set of visited nodes
    Graph

    Set<Node> run() {
        //Helper method code to call iterArtPts
    }
    void iterArtPts(APNode firstNode, int depth, APNode root){
        //Iterative Algorithm
    }
}
```

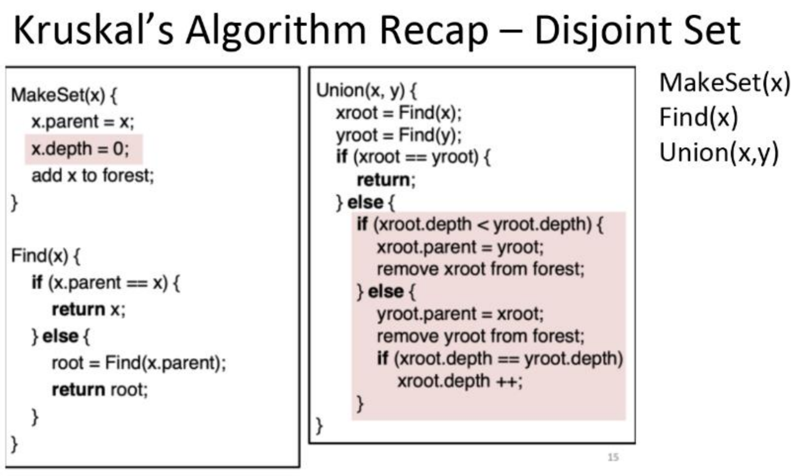
Figure 2: Improved version of Articulation algorithm using iterative method(minimum)



In pseudocode, this is how Kruskal's algorithm works:

```
Imagine using Edges with attributes <n1, n2, edgeLength>
n1 and n2 = respective nodes
edgeLength = length of edge between them

Set forest as N node sets, each containing a node;
Set fringe as a priority queue of all the edges ⟨n1, n2, length⟩ ;
Set MST as an empty set to store Edges;

while amount of node sets in forest > 1 and fringe isn't empty {
     poll ⟨n1*, n2*, length*⟩ as Edge with minimum edgeLength from
fringe;
     if (n1* and n2* are in different sets in forest) {
          Merge the two sets in forest;
          Add the edge to MST;
     }
}
return MST;
```

Figure3: Pseudocode for Kruskal's algorithm (core)



Kruskal's Algorithm Recap – Disjoint Set

```
MakeSet(x) {
   x.parent = x;
   x.depth = 0;
   add x to forest;
}

Find(x) {
   if (x.parent == x) {
      return x;
   } else {
      root = Find(x.parent);
      return root;
   }
}
```

```
Union(x, y) {
   xroot = Find(x);
   yroot = Find(y);
   if (xroot == yroot) {
      return;
   } else {
      if (xroot.depth < yroot.depth) {
         xroot.parent = yroot;
         remove xroot from forest;
      } else {
         yroot.parent = xroot;
         remove yroot from forest;
         if (xroot.depth == yroot.depth)
            xroot.depth ++;
      }
}
```

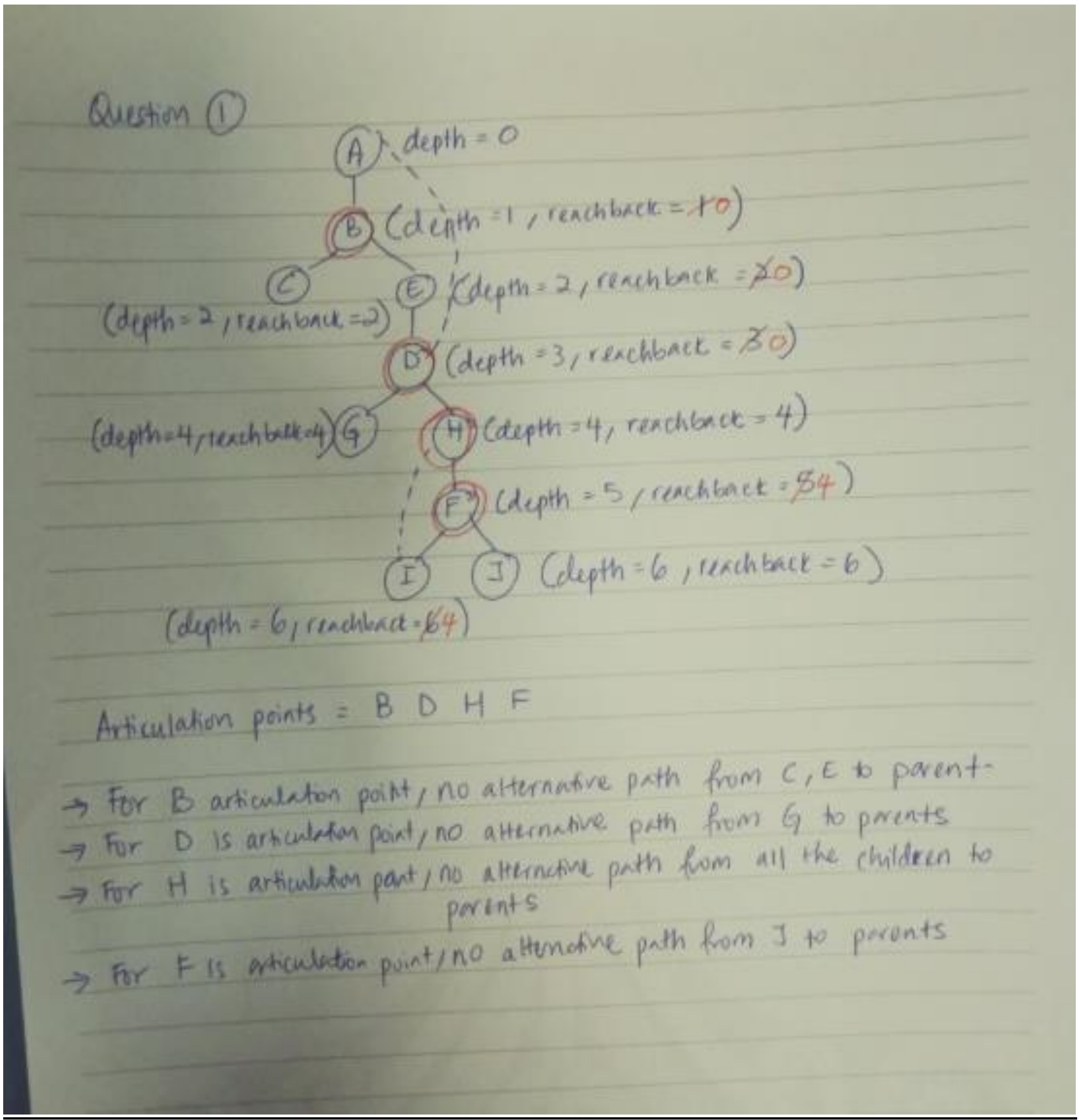
MakeSet(x)
Find(x)
Union(x,y)

15

Figure 4: Pseudocode for Kruskal's algorithm using disjoint set (Challenge)

4) Your answers to the questions from page 5.

<u>Question 1</u>


Question 1
A
depth = 0
B
(depth = 1, reachback = ~~1~~0)
C
(depth = 2, reachback = 2)
E
(depth = 2, reachback = ~~2~~0)
D
(depth = 3, reachback = ~~3~~0)
(depth = 4, reachback = 4)
G
H
(depth = 4, reachback = 4)
F
(depth = 5, reachback = ~~5~~4)
I
J
(depth = 6, reachback = 6)
(depth = 6, reachback = ~~6~~4)

Articulation points = B D H F

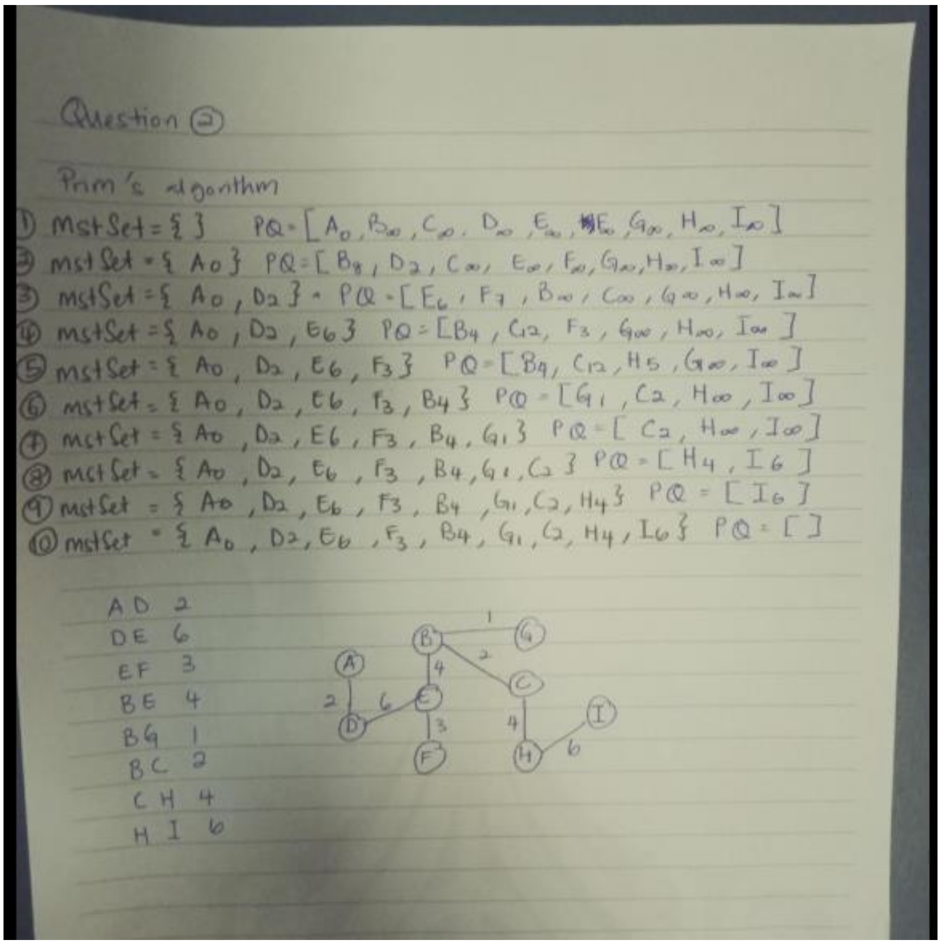
→ For B articulation point, no alternative path from C, E to parent.

→ For D is articulation point, no alternative path from G to parents

→ For H is articulation part, no alternative path from all the children to parents

→ For F is articulation point, no alternative path from J to parents

Question2

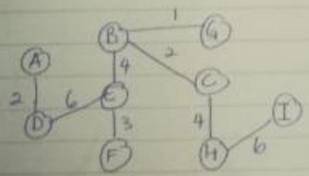


Question ②

Prim's algorithm

① mstSet = { } PQ = [A_0, B_∞, C_∞, D_∞, E_∞, ~~E~~ F_∞, G_∞, H_∞, I_∞]
② mstSet = { A_0 } PQ = [B_8, D_2, C_∞, E_∞, F_∞, G_∞, H_∞, I_∞]
③ mstSet = { A_0, D_2 } PQ = [E_6, F_7, B_∞, C_∞, G_∞, H_∞, I_∞]
④ mstSet = { A_0, D_2, E_6 } PQ = [B_4, C_{12}, F_3, G_∞, H_∞, I_∞]
⑤ mstSet = { A_0, D_2, E_6, F_3 } PQ = [B_4, C_{12}, H_5, G_∞, I_∞]
⑥ mstSet = { A_0, D_2, E_6, F_3, B_4 } PQ = [G_1, C_2, H_∞, I_∞]
⑦ mstSet = { A_0, D_2, E_6, F_3, B_4, G_1 } PQ = [C_2, H_∞, I_∞]
⑧ mstSet = { A_0, D_2, E_6, F_3, B_4, G_1, C_2 } PQ = [H_4, I_6]
⑨ mstSet = { A_0, D_2, E_6, F_3, B_4, G_1, C_2, H_4 } PQ = [I_6]
⑩ mstSet = { A_0, D_2, E_6, F_3, B_4, G_1, C_2, H_4, I_6 } PQ = [ ]

A D	2
D E	6
E F	3
B E	4
B G	1
B C	2
C H	4
H I	6

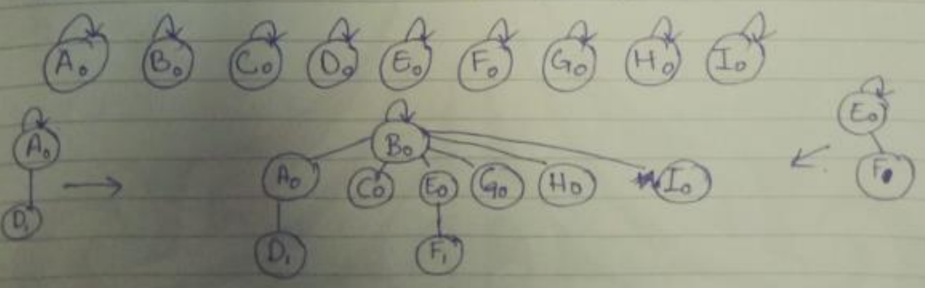


Question ②

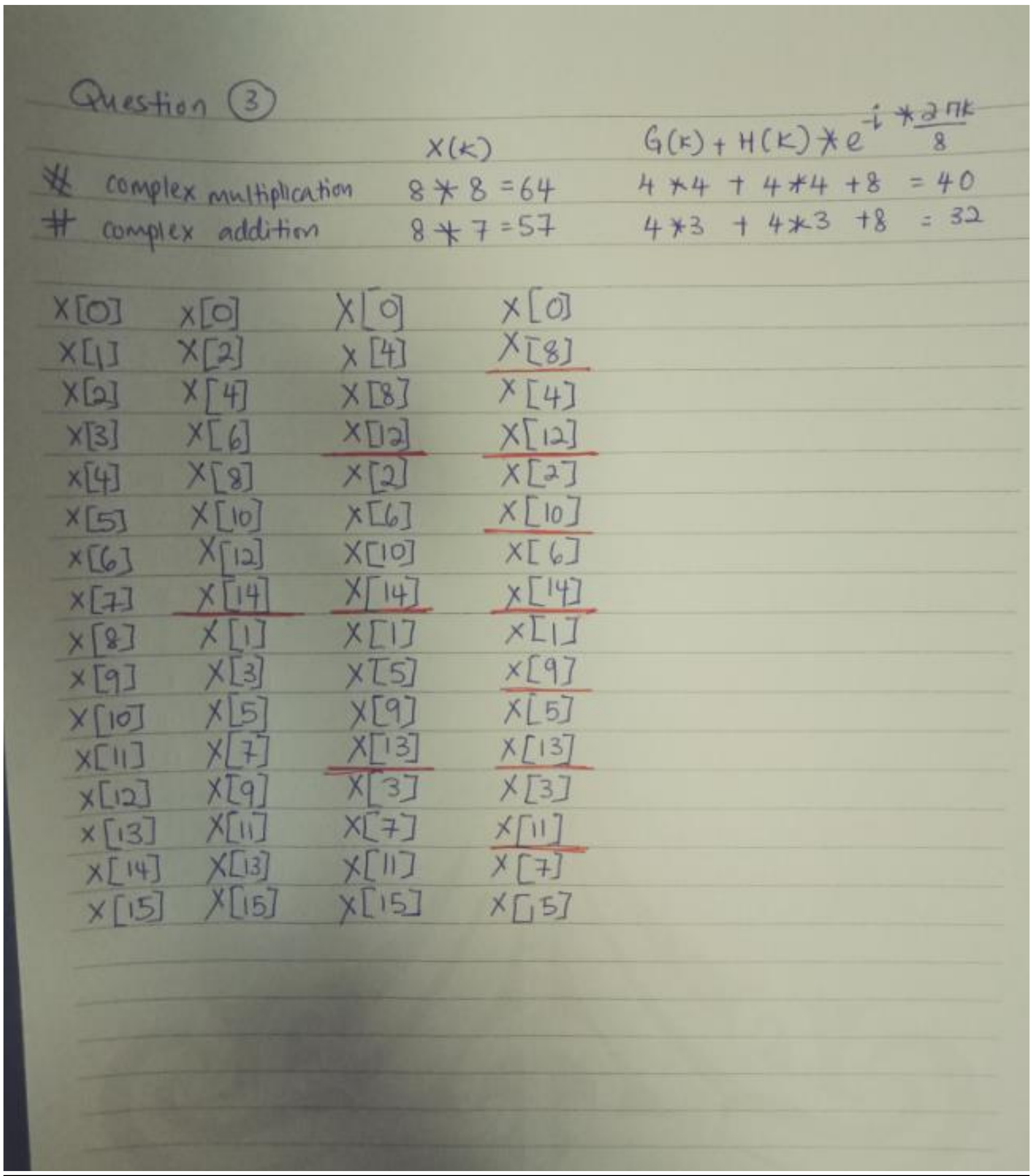
Kruskal's disjoint set

~~Union (B G)~~

Weight	Source	Dest	
1	B	G	union (B G)
2	B	C	union (B C)
2	A	D	union (A D)
3	E	F	union (E F)
3	C	G	Union (C G)
4	B	E	Union (B E)
4	C	H	Union (C H)
5	F	H	Union (F H)
6	D	E	Union (D E)
6	H	I	Union (H I)
7	D	F	Union (D F)
8	A	B	Union (A B)
9	G	I	Union (G I)
10	C	I	Unoin (C I)



Question3



Question ③

	$X(K)$	$G(K) + H(K) * e^{-i * \frac{2\pi k}{8}}$
# complex multiplication	8 * 8 = 64	4 * 4 + 4 * 4 + 8 = 40
# complex addition	8 * 7 = 57	4 * 3 + 4 * 3 + 8 = 32

X[0]	X[0]	X[0]	X[0]
X[1]	X[2]	X[4]	X[8]
X[2]	X[4]	X[8]	X[4]
X[3]	X[6]	X[12]	X[12]
X[4]	X[8]	X[2]	X[2]
X[5]	X[10]	X[6]	X[10]
X[6]	X[12]	X[10]	X[6]
X[7]	X[14]	X[14]	X[14]
X[8]	X[1]	X[1]	X[1]
X[9]	X[3]	X[5]	X[9]
X[10]	X[5]	X[9]	X[5]
X[11]	X[7]	X[13]	X[13]
X[12]	X[9]	X[3]	X[3]
X[13]	X[11]	X[7]	X[11]
X[14]	X[13]	X[11]	X[7]
X[15]	X[15]	X[15]	X[15]