NAME: SADIQ ALI
FACULTY NO. : 18COB204
ENROLLMENT NO. : GI0206

QUESTION:
Part1: Write a program in Java to find Minimum Spanning Tree (MST)of a given weighted
graph using

(i) PRIM's Algorithm and
(ii)KRUSKAL"s algorithm.

Both of these algorithms can be implemented in many ways using different data structures. Go
through the related text and implement each of these algorithms using the most efficient data
structure. Show the results of different steps of these algorithms for the given graph and Write
your observations
Part2: Analyse the complexity of each of these two MST algorithms for the selected
implementation. (For example, consider an undirected graph with 10 vertices and 20 edges with
non-zero weights on it. You may randomly generate these 20 edges with non-zero weights, and
compute the time taken by the selected implementation. Repeat this process ten times and
compute the time taken. Compute average time taken by each of the two MST algorithms).

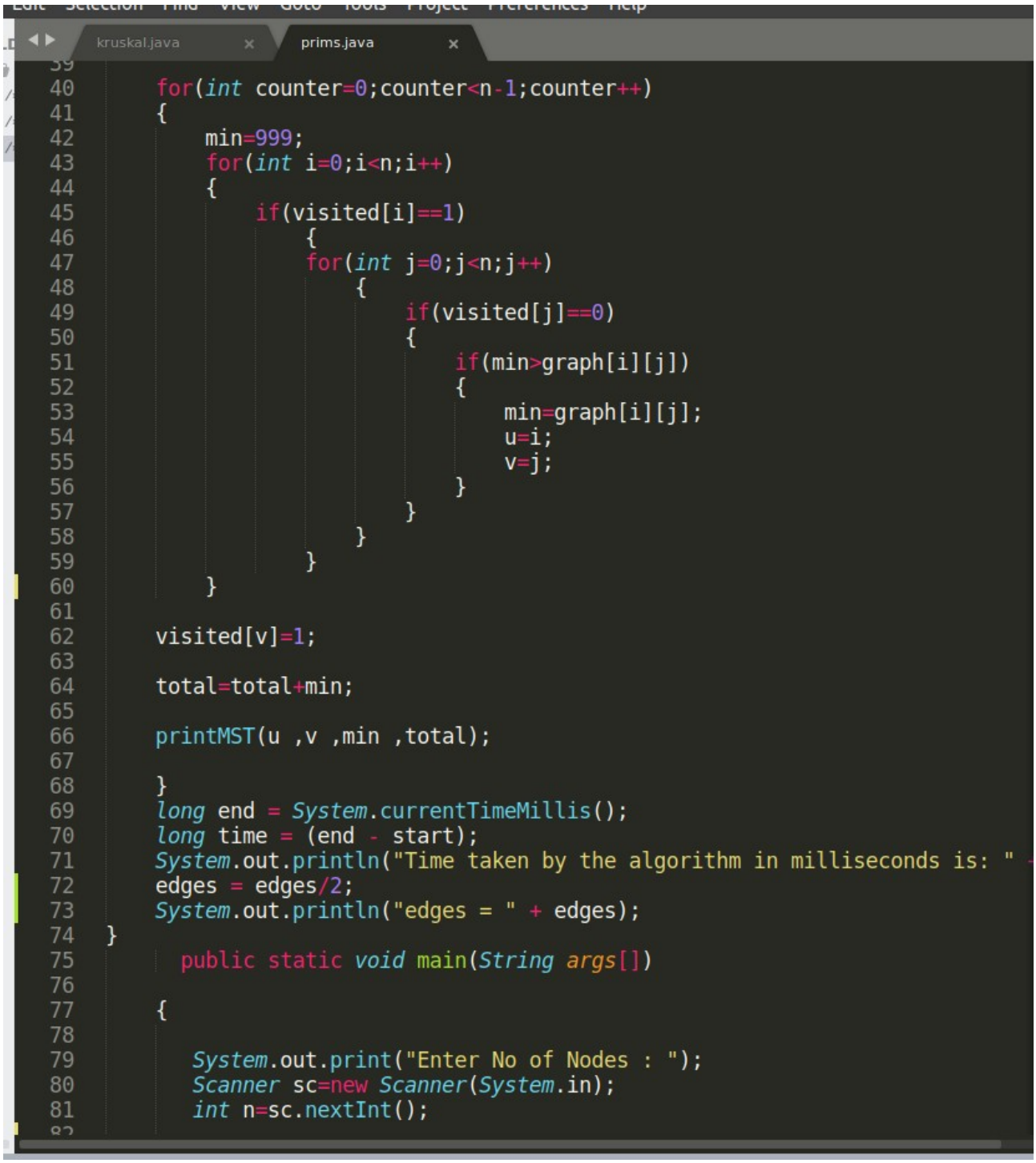
SOLUTION:
PART 1:
A) PRIM'S ALGORITHM

For time calculation i used, System.currentTimeMillis(), by adding this line at the starting and ending of our algorithm and storing the values at both instances (start and end) we can calculate the time taken by the algorithm in milli seconds.

JAVA IMPLEMANTATION FOR PRIM'S ALGORITHM!

kruskal.java | prims.java

```
1  import java.util.*;
2  import java.lang.*;
3  import java.io.*;
4
5  class prims
6
7  {
8
9          void printMST(int u ,int v ,int min ,int total){
10
11            System.out.println("source "+u+"-->"+v+": "+min);
12         System.out.println("the total weigh of the graph: "+total);
13
14         }
15
16       long start = System.currentTimeMillis();
17       void primMST(int graph[][], int n){
18          int visited[]=new int[n];
19
20          int min=999;int u=0;int v=0;int total=0;
21
22          int edges = 0;
23
24          for(int i=0;i<n;i++)
25          {
26              visited[i]=0;
27              for(int j=0;j<n;j++)
28              {
29                  if(graph[i][j]==0)
30                  {
31                      graph[i][j]=999;
32                  }
33                  else {
34                      edges++ ;
35                  }
36              }
37          }
38      visited[0]=1;
39
40      for(int counter=0;counter<n-1;counter++)
41      {
42          min=999;
43          for(int i=0;i<n;i++)
```


kruskal.java
prims.java
for(int counter=0;counter<n-1;counter++)
{
min=999;
for(int i=0;i<n;i++)
{
if(visited[i]==1)
{
for(int j=0;j<n;j++)
{
if(visited[j]==0)
{
if(min>graph[i][j])
{
min=graph[i][j];
u=i;
v=j;
}
}
}
}
}
visited[v]=1;
total=total+min;
printMST(u ,v ,min ,total);
}
long end = System.currentTimeMillis();
long time = (end - start);
System.out.println("Time taken by the algorithm in milliseconds is: "
edges = edges/2;
System.out.println("edges = " + edges);
}
public static void main(String args[])
{
System.out.print("Enter No of Nodes : ");
Scanner sc=new Scanner(System.in);
int n=sc.nextInt();

kruskal.java | prims.java

```
        System.out.print("Enter No of Nodes : ");
        Scanner sc=new Scanner(System.in);
        int n=sc.nextInt();


        int graph[][] = new int[n][n];


         for(int i=0; i<n;i++)
          {
              for(int j=0; j<i;j++)
              {
                  int value = (int)(Math.random()*10);
                  if(i == j){
                    graph[i][j] = 0;
                  } else {
                    graph[i][j]= value;
                    graph[j][i] = value;
                  }
              }
          }


          System.out.print("\nData you entered : \n");
          for(int []x:graph){
              for(int y:x){
              System.out.print(y+"         ");
              }
              System.out.println();
          }
           prims t =new prims();

          // Print the solution

              //0=a,1=b,2=c,3=d,4=e,5=f,6=z

        t.primMST(graph,n);

          }
}
```

INPUT AND OUTPUT FOR PRIM'S ALGORITHM

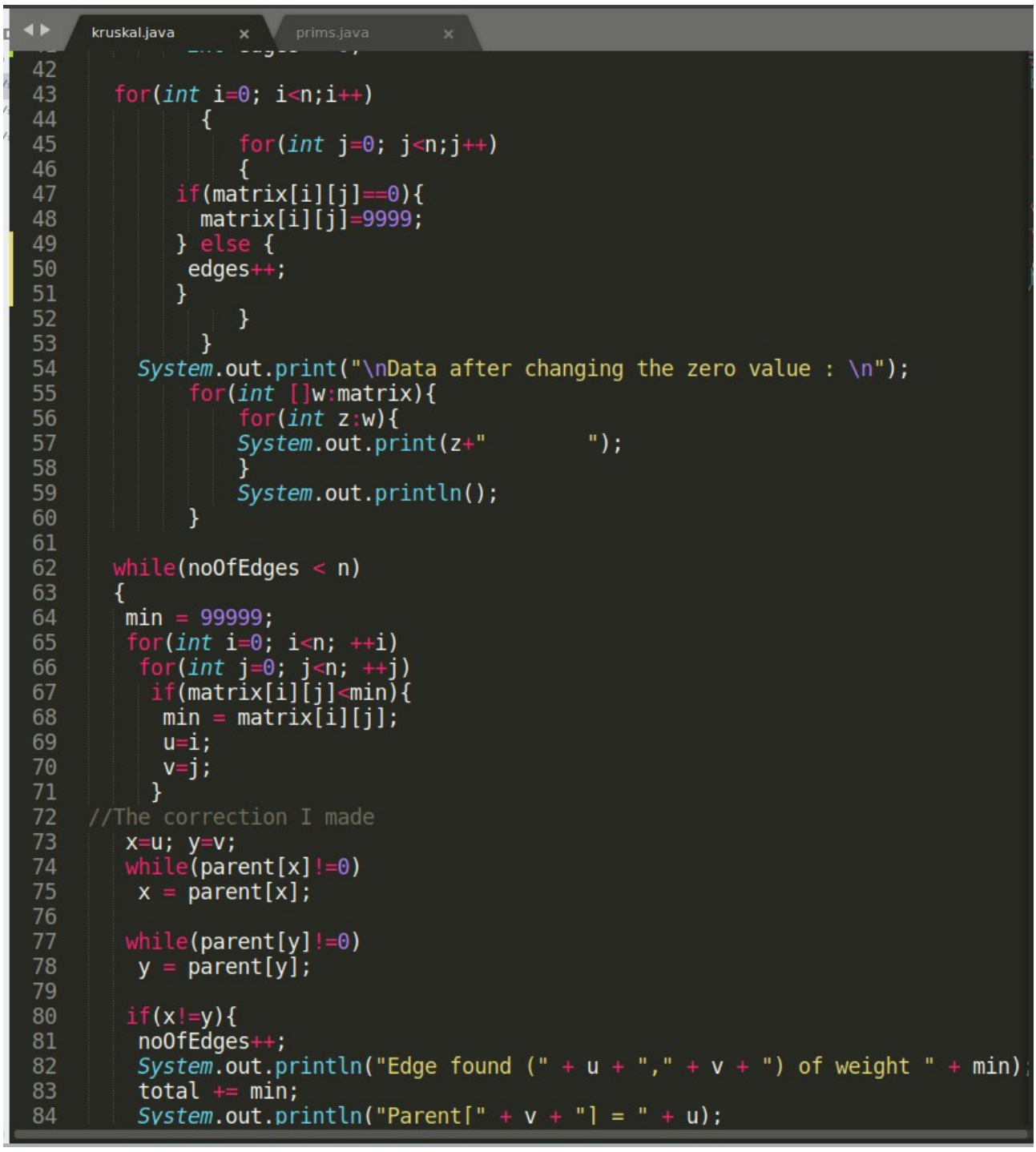
PART 1:

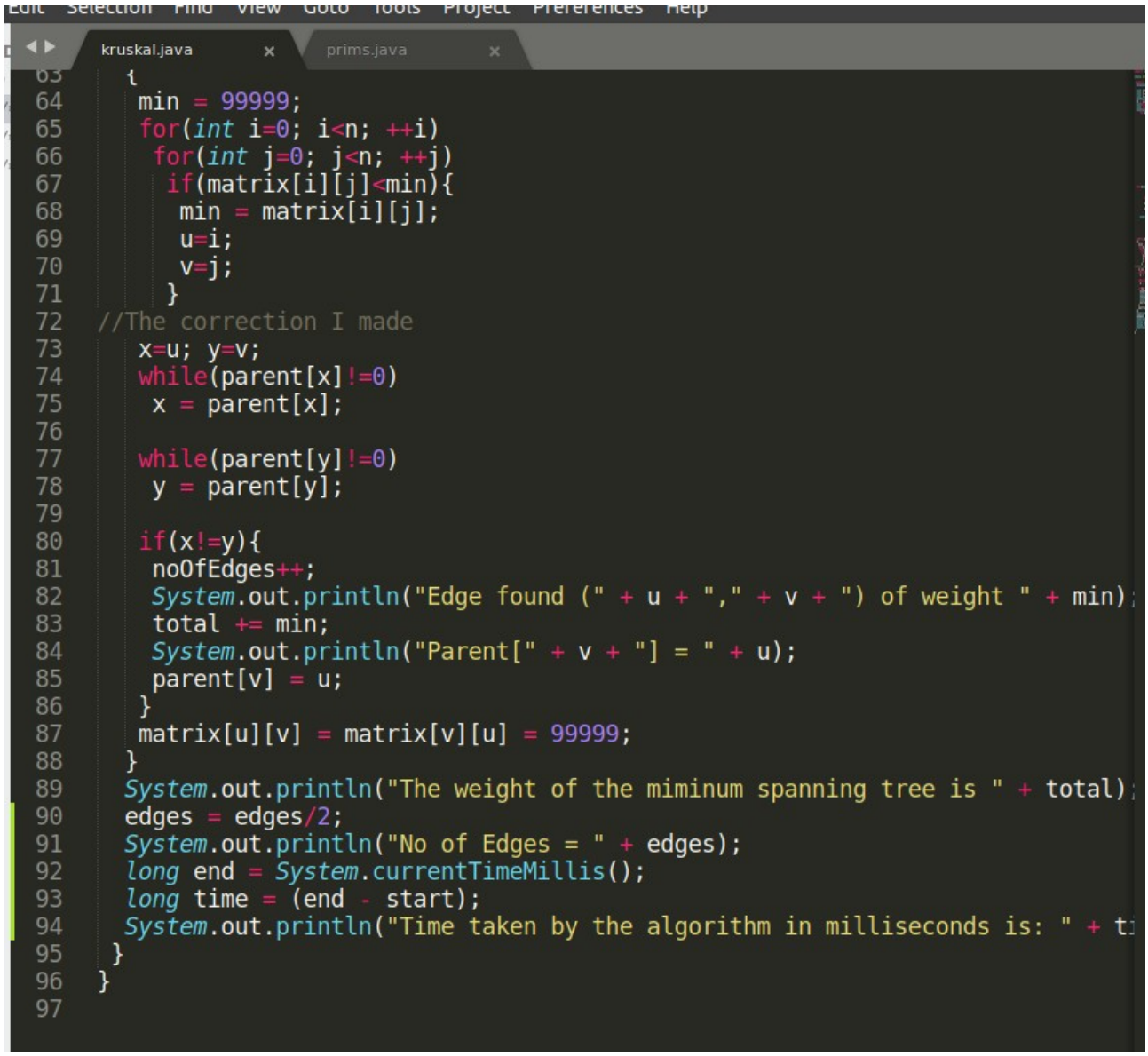
B) KRUSKAL'S ALGORITHM

JAVA IMPLEMANTATION FOR KRUSKAL'S ALGORITHM!

kruskal.java | prims.java

```java
import java.util.Scanner;

class kruskal {

 public static void main(String[] args) {
  Scanner sc = new Scanner(System.in);

  int noOfEdges=1;
  int n, u=0, v=0, min, total=0;
  int x,y;
  System.out.print("Enter the total no. nodes : ");
  n = sc.nextInt();
  int[][] matrix = new int [n][n];
  int [] parent = new int [n];
  System.out.println("Enter the adjacency matrix :");
    for(int i=0; i<n;i++)
          {
             for(int j=0; j<i;j++)
             {
                 int value = (int)(Math.random()*10);
                 if(i == j){
                   matrix[i][j] = 0;
                 } else {
                   matrix[i][j]= value;
                   matrix[j][i] = value;
                 }
             }
          }


        System.out.print("\nData you entered : \n");
        for(int []w:matrix){
             for(int z:w){
             System.out.print(z+"    ");
             }
             System.out.println();
        }

        long start = System.currentTimeMillis();

        int edges = 0;

  for(int i=0; i<n;i++)
```

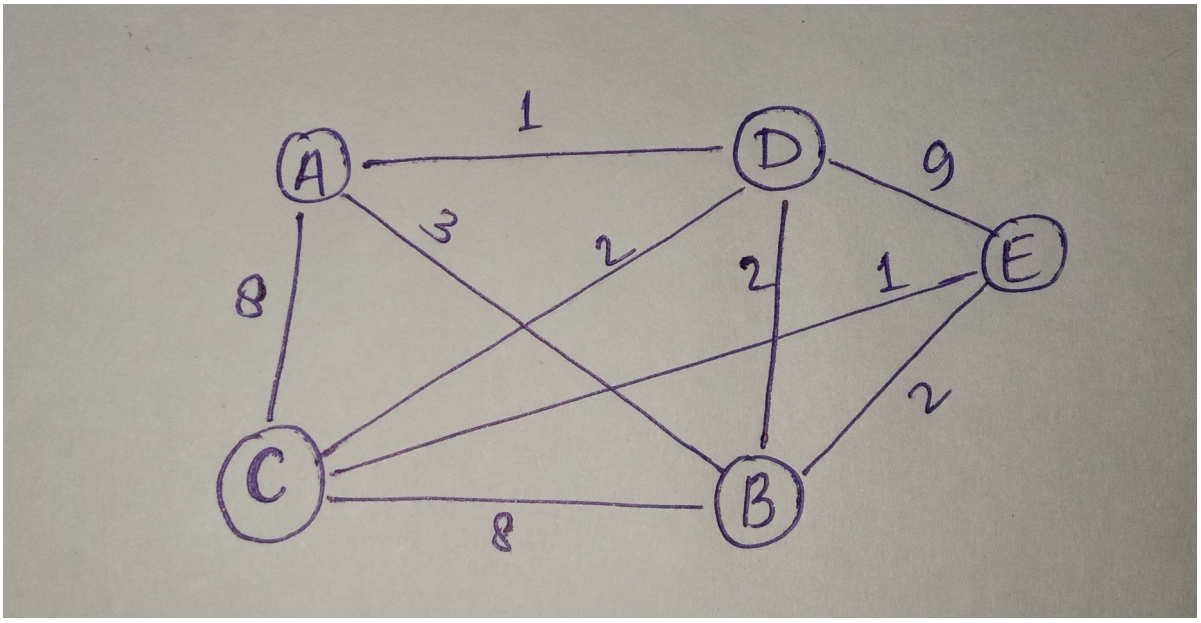

kruskal.java
prims.java
for(int i=0; i<n;i++)
{
for(int j=0; j<n;j++)
{
if(matrix[i][j]==0){
matrix[i][j]=9999;
} else {
edges++;
}
}
}
System.out.print("\nData after changing the zero value : \n");
for(int []w:matrix){
for(int z:w){
System.out.print(z+" ");
}
System.out.println();
}
while(noOfEdges < n)
{
min = 99999;
for(int i=0; i<n; ++i)
for(int j=0; j<n; ++j)
if(matrix[i][j]<min){
min = matrix[i][j];
u=i;
v=j;
}
//The correction I made
x=u; y=v;
while(parent[x]!=0)
x = parent[x];
while(parent[y]!=0)
y = parent[y];
if(x!=y){
noOfEdges++;
System.out.println("Edge found (" + u + "," + v + ") of weight " + min);
total += min;
System.out.println("Parent[" + v + "] = " + u);


Edit Selection Find View Goto Tools Project Preferences Help
kruskal.java
prims.java
{
min = 99999;
for(int i=0; i<n; ++i)
for(int j=0; j<n; ++j)
if(matrix[i][j]<min){
min = matrix[i][j];
u=i;
v=j;
}
//The correction I made
x=u; y=v;
while(parent[x]!=0)
x = parent[x];
while(parent[y]!=0)
y = parent[y];
if(x!=y){
noOfEdges++;
System.out.println("Edge found (" + u + "," + v + ") of weight " + min);
total += min;
System.out.println("Parent[" + v + "] = " + u);
parent[v] = u;
}
matrix[u][v] = matrix[v][u] = 99999;
}
System.out.println("The weight of the miminum spanning tree is " + total);
edges = edges/2;
System.out.println("No of Edges = " + edges);
long end = System.currentTimeMillis();
long time = (end - start);
System.out.println("Time taken by the algorithm in milliseconds is: " + t
}
}

INPUT AND OUTPUT FOR KRUSKAL'S ALGORITHM

```
sadiq@sadiqali: ~/MyJavaDirectory
sadiq@sadiqali:~/MyJavaDirectory$ javac kruskal.java
sadiq@sadiqali:~/MyJavaDirectory$ java kruskal

Data you entered :
0    3    8    1    0
3    0    8    2    2
8    8    0    2    1
1    2    2    0    9
0    2    1    9    0

Data after changing the zero value :
9999         3         8         1         9999
3         9999         8         2         2
8         8         9999         2         1
1         2         2         9999         9
9999         2         1         9         9999
Edge found (0,3) of weight 1
Parent[3] = 0
Edge found (2,4) of weight 1
Parent[4] = 2
Edge found (1,3) of weight 2
Parent[3] = 1
Edge found (1,4) of weight 2
Parent[4] = 1
The weight of the miminum spanning tree is 6
No of Edges = 9
```

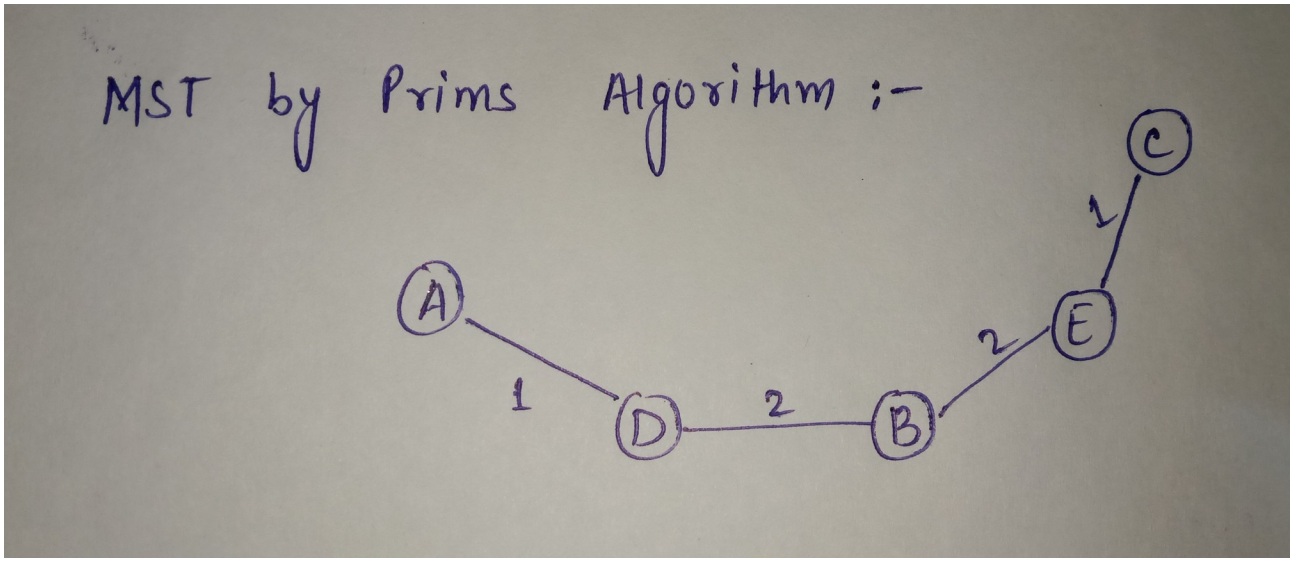
- **EXAMPLE GRAPH**



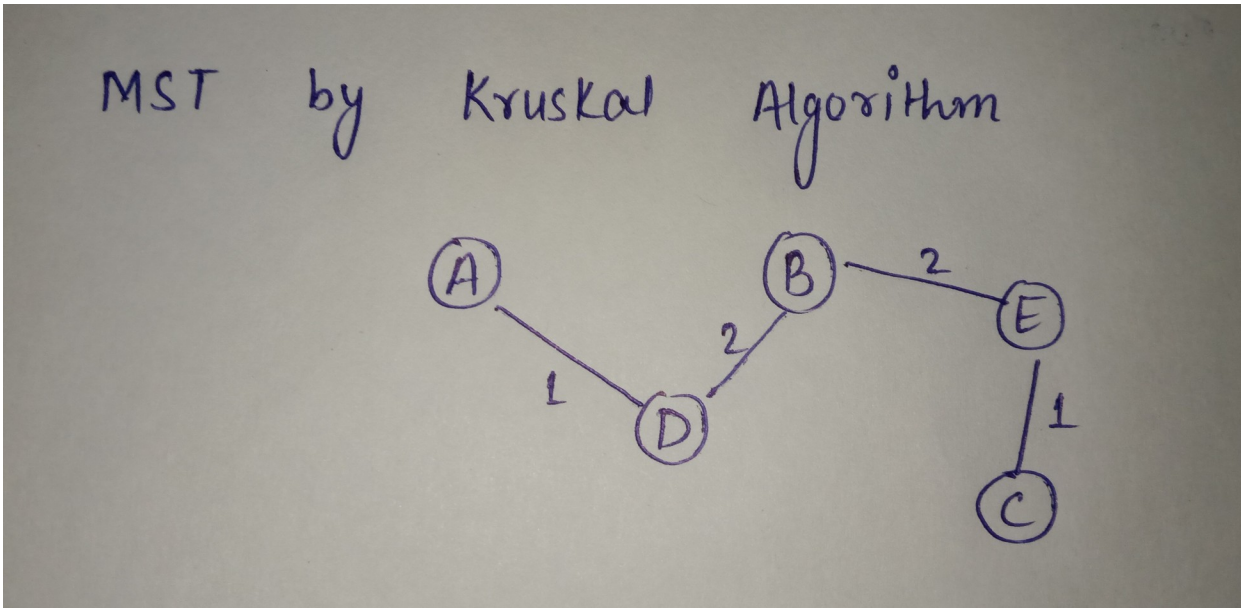
- Adjacency Matrix for the above graph will be

	A	B	C	D	E
A	0	3	8	1	0
B	3	0	8	2	2
C	8	8	0	2	1
D	1	2	2	0	9
E	0	2	1	9	0

- **Minimum spanning tree using prim's output (attached above)**



- **Minimum spanning tree using kruskal's output (attached above)**



Observations

- As we observe that the minimum cost, we get using any of the above algorithm are equal that is 6 in the above example.
- The connection between the nodes are also same in the above example but sometimes they migth be different but the minimum cost given by any of the above example will be same.

Conclusion

- Steps can be same or different to get the minimum cost of the graph but the minimum cost is same.
- Both the algorithm follows different – different procedures to get the minimum cost.
- In Prim's algorithm: The idea behind Prim's algorithm is, a spanning tree means all vertices must be connected. So, the two disjoint subsets of vertices must be connected to

make a Spanning Tree. And they must be connected with the minimum weight edge to make it a Minimum Spanning Tree.

- In Kruskal algorithm: If the graph is connected, it finds a minimum spanning tree. It is a greedy algorithm in graph theory as in each step it adds the next lowest-weight edge that will not form a cycle to the minimum spanning forest.

- **PART 2**
- **FOR PRIM'S ALGORITHM**
- **TIME TAKEN IS IN MILLISECONDS**

Number of Nodes	Number of edges	Time taken(milli-sec)
5	8	405
10	14	3045
15	18	10200
20	25	24100
25	28	47000
30	36	81430

- **PART 2**
- **FOR K'S KRUSKAL'S ALGORITHM**
- **TIME TAKEN IS IN MILLISECONDS**

Number of Nodes	Number of edges	Time taken(milli-sec)
5	8	280
10	14	2030
15	18	6800
20	25	16080
25	28	31340
30	36	54100