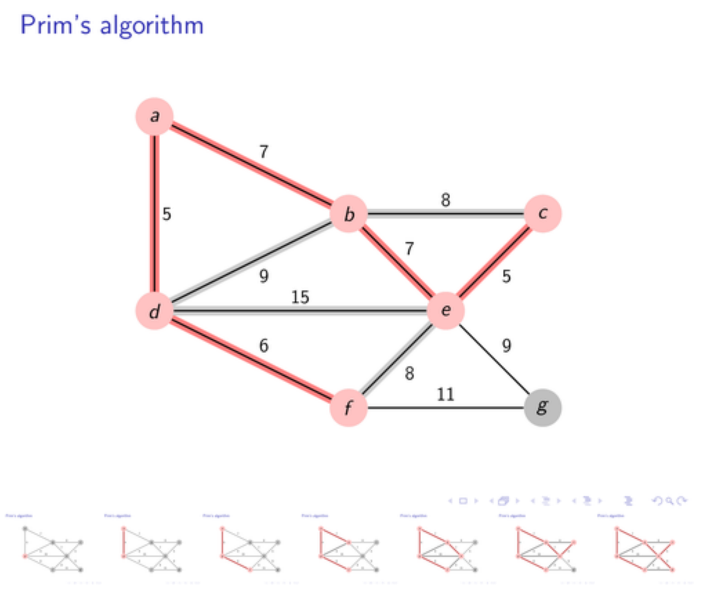
NETWORKS & COMMUNICATION (CSE 1004) PROJECT

ASHA S.

Osho Agyeya-15BCE1326

PRIM’s ALGORITHM



Prim's algorithm

USES OF PRIM’s ALGORITHM

MAIN PURPOSE: FINDING MINIMUM SPANNING TREE

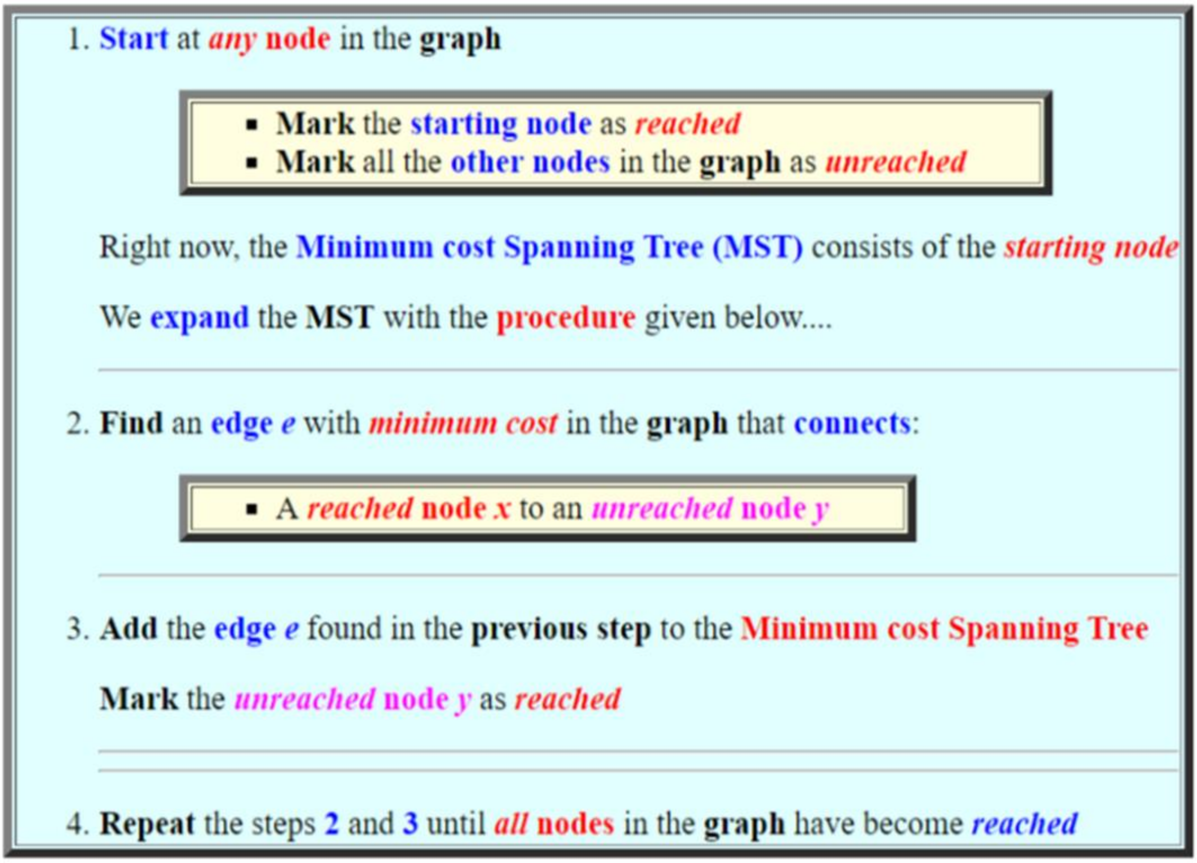
1.Distances between the cities for the minimum route calculation
for transportation.

2.For Establishing the network cables these play important role in finding
the minimum cables required to cover the whole region.

3. Prim Algorithm Approach to Improving Local Access Network in Rural Areas

4. AI (Artificial Intelligence)
5. Game Development
6. Cognitive Science

ALGORITHM



1. **Start** at ***any* node** in the **graph**

   - **Mark** the **starting node** as ***reached***
   - **Mark** all the **other nodes** in the **graph** as ***unreached***

   Right now, the **Minimum cost Spanning Tree (MST)** consists of the ***starting node***

   We **expand** the **MST** with the **procedure** given below....

2. **Find** an **edge *e*** with ***minimum cost*** in the **graph** that **connects**:

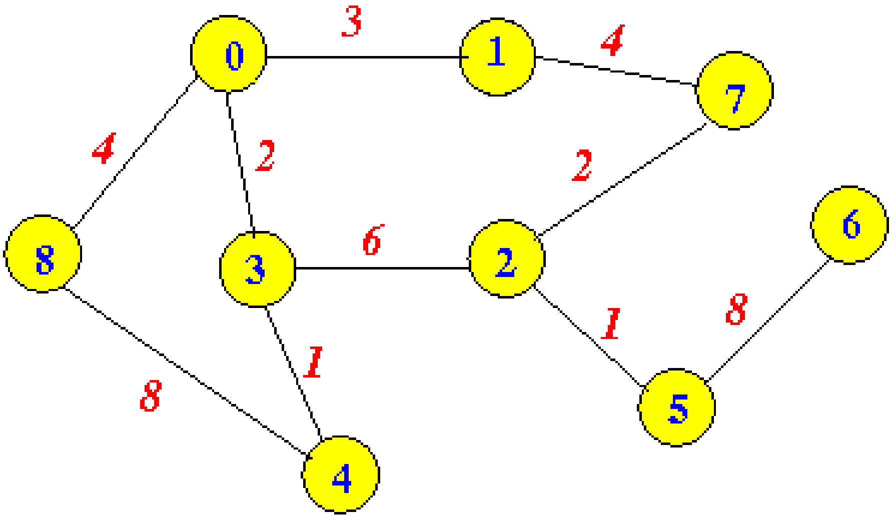
   - A ***reached* node *x*** to an ***unreached* node *y***

3. **Add** the **edge *e*** found in the **previous step** to the **Minimum cost Spanning Tree**

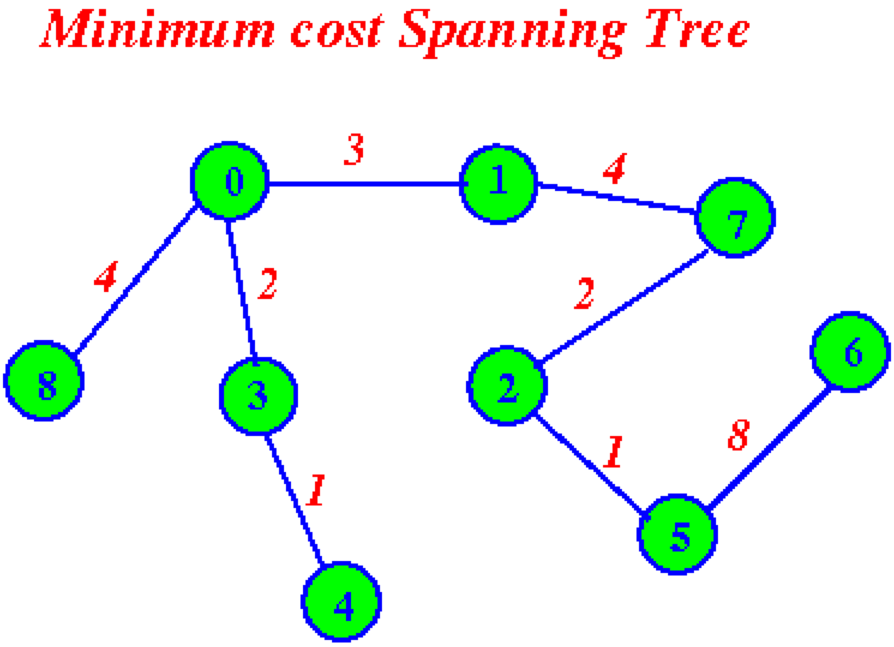
   Mark the ***unreached* node *y*** as ***reached***

4. **Repeat** the steps **2** and **3** until ***all* nodes** in the **graph** have become ***reached***

INPUT



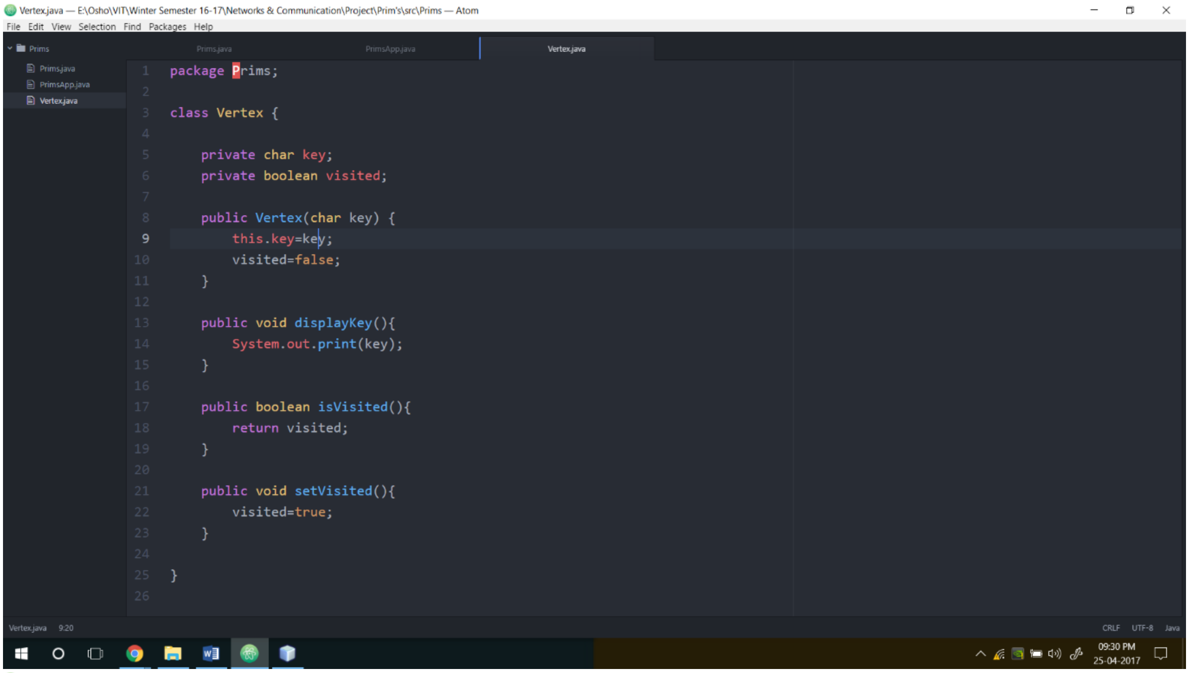
OUTPUT



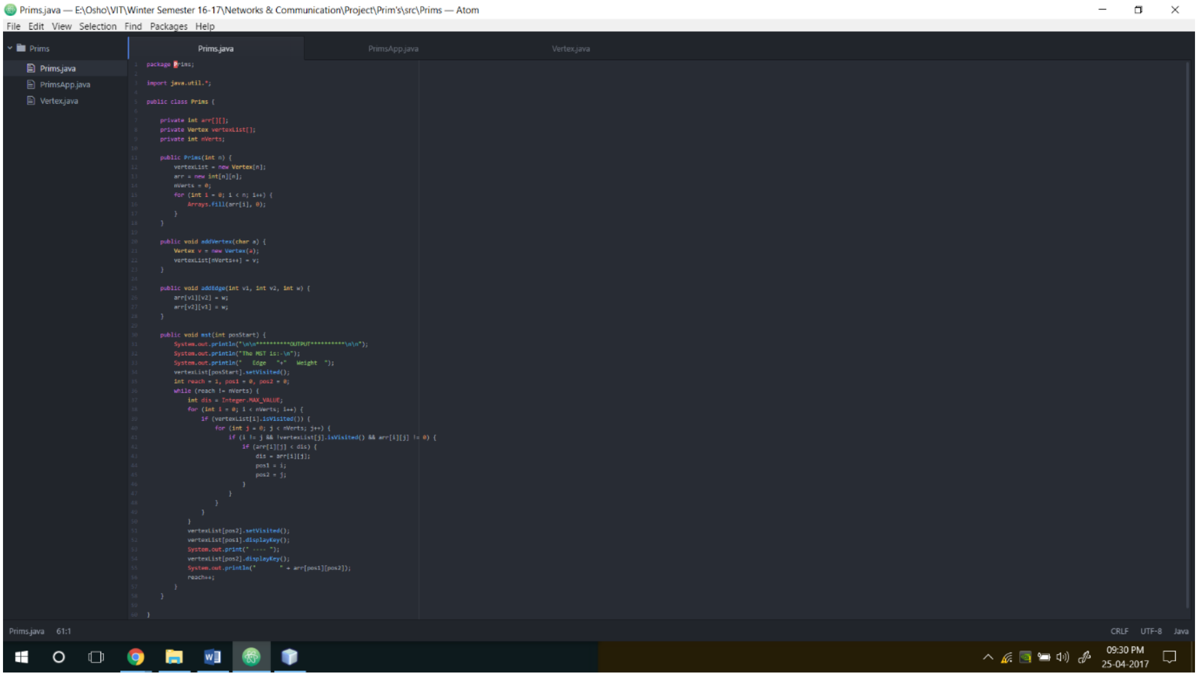
PSEUDO CODE

```
ReachSet = {0};
UnReachSet = {1, 2, ..., N-1};
   SpanningTree = {};

   while ( UnReachSet ≠ empty )
   {
      Find edge e = (x, y) such that:
       1. x ∈ ReachSet
      2. y ∈ UnReachSet
      3. e has smallest cost

      SpanningTree = SpanningTree ∪ {e};

      ReachSet   = ReachSet ∪ {y};
      UnReachSet = UnReachSet - {y};
   }
```

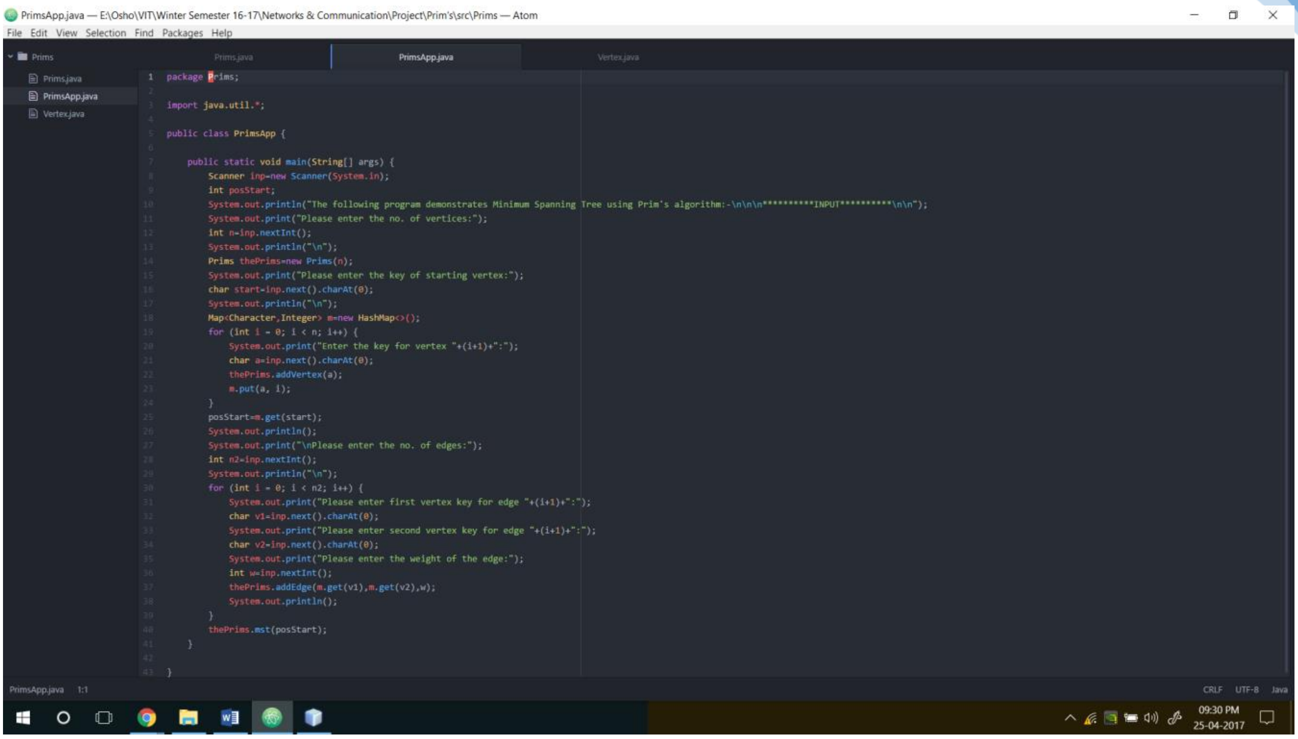
IMPLEMENTATION IN JAVA


Vertex.java — E:\Osho\VIT\Winter Semester 16-17\Networks & Communication\Project\Prim's\src\Prims — Atom
File Edit View Selection Find Packages Help
Prims
Prims.java
PrimsApp.java
Vertex.java
Vertex.java 9:20
CRLF UTF-8 Java
09:30 PM
25-04-2017

```java
package Prims;

class Vertex {

    private char key;
    private boolean visited;

    public Vertex(char key) {
        this.key=key;
        visited=false;
    }

    public void displayKey(){
        System.out.print(key);
    }

    public boolean isVisited(){
        return visited;
    }

    public void setVisited(){
        visited=true;
    }

}
```

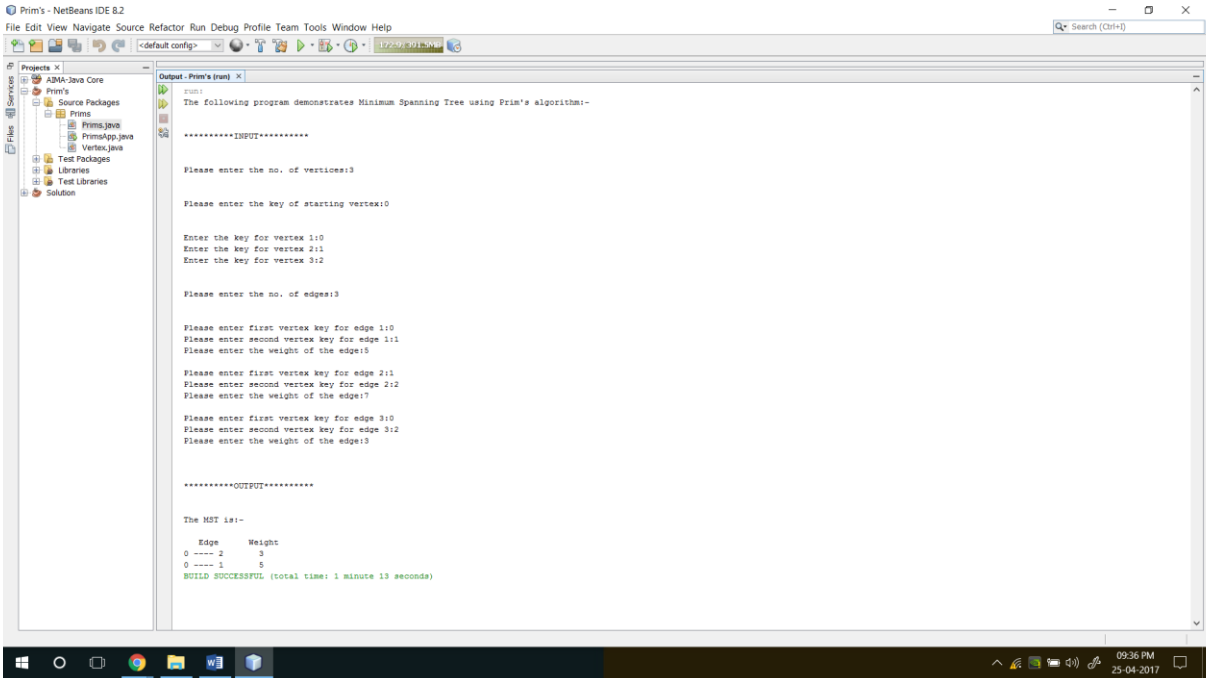

Prims.java — E:\Osho\VIT\Winter Semester 16-17\Networks & Communication\Project\Prim's\src\Prims — Atom
File Edit View Selection Find Packages Help
Prims
Prims.java
PrimsApp.java
Vertex.java
Prims.java 61:1
CRLF UTF-8 Java
09:30 PM
25-04-2017

```java
package Prims;

import java.util.*;

public class Prims {

    private int arr[][];
    private Vertex vertexList[];
    private int nVerts;

    public Prims(int n) {
        vertexList = new Vertex[n];
        arr = new int[n][n];
        nVerts = 0;
        for (int i = 0; i < n; i++) {
            Arrays.fill(arr[i], 0);
        }
    }

    public void addVertex(char a) {
        Vertex v = new Vertex(a);
        vertexList[nVerts++] = v;
    }

    public void addEdge(int v1, int v2, int w) {
        arr[v1][v2] = w;
        arr[v2][v1] = w;
    }

    public void mst(int posStart) {
        System.out.println("\n\n**********OUTPUT**********\n\n");
        System.out.println("The MST is:-\n");
        System.out.println("   Edge   "+"   Weight  ");
        vertexList[posStart].setVisited();
        int reach = 1, pos1 = 0, pos2 = 0;
        while (reach != nVerts) {
            int dis = Integer.MAX_VALUE;
            for (int i = 0; i < nVerts; i++) {
                if (vertexList[i].isVisited()) {
                    for (int j = 0; j < nVerts; j++) {
                        if (i != j && !vertexList[j].isVisited() && arr[i][j] != 0) {
                            if (arr[i][j] < dis) {
                                dis = arr[i][j];
                                pos1 = i;
                                pos2 = j;
                            }
                        }
                    }
                }
            }
            vertexList[pos2].setVisited();
            vertexList[pos1].displayKey();
            System.out.print(" ---- ");
            vertexList[pos2].displayKey();
            System.out.println("       " + arr[pos1][pos2]);
            reach++;
        }
    }

}
```


PrimsApp.java — E:\Osho\VIT\Winter Semester 16-17\Networks & Communication\Project\Prim's\src\Prims — Atom
File Edit View Selection Find Packages Help
Prims
Prims.java
PrimsApp.java
Vertex.java
PrimsApp.java 1:1
CRLF UTF-8 Java
09:30 PM
25-04-2017

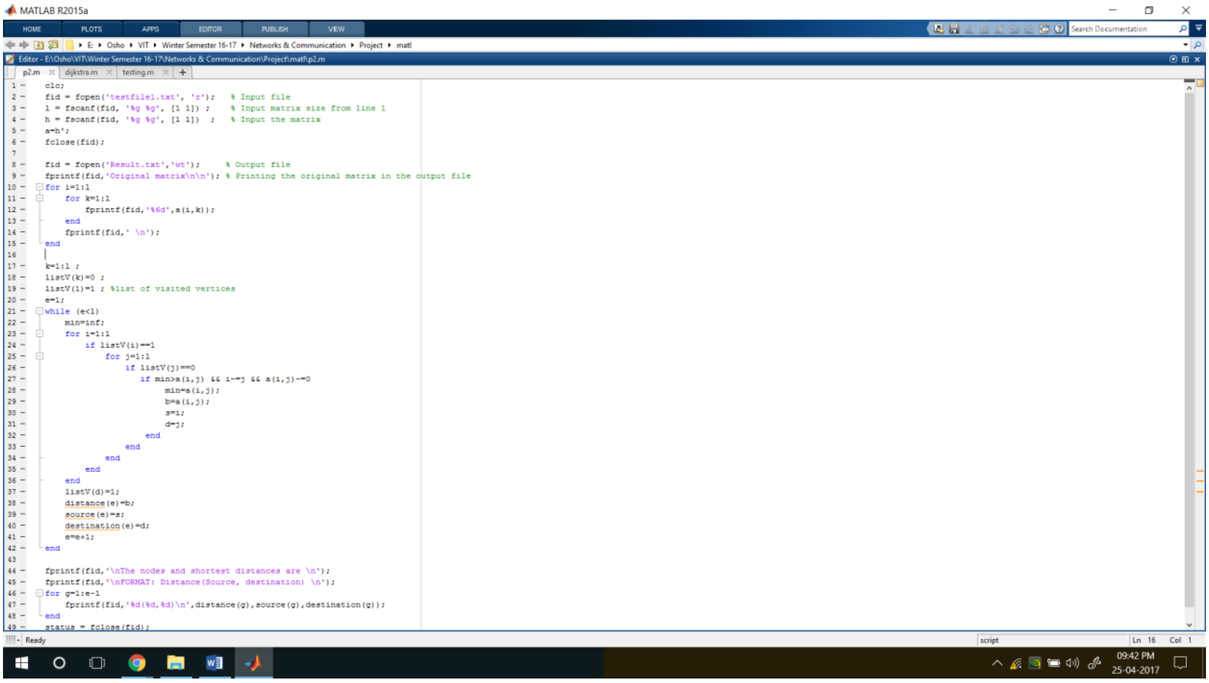
```java
package Prims;

import java.util.*;

public class PrimsApp {

    public static void main(String[] args) {
        Scanner inp=new Scanner(System.in);
        int posStart;
        System.out.println("The following program demonstrates Minimum Spanning Tree using Prim's algorithm:-\n\n\n**********INPUT**********\n\n");
        System.out.print("Please enter the no. of vertices:");
        int n=inp.nextInt();
        System.out.println("\n");
        Prims thePrims=new Prims(n);
        System.out.print("Please enter the key of starting vertex:");
        char start=inp.next().charAt(0);
        System.out.println("\n");
        Map<Character,Integer> m=new HashMap<>();
        for (int i = 0; i < n; i++) {
            System.out.print("Enter the key for vertex "+(i+1)+":");
            char a=inp.next().charAt(0);
            thePrims.addVertex(a);
            m.put(a, i);
        }
        posStart=m.get(start);
        System.out.println();
        System.out.print("\nPlease enter the no. of edges:");
        int n2=inp.nextInt();
        System.out.println("\n");
        for (int i = 0; i < n2; i++) {
            System.out.print("Please enter first vertex key for edge "+(i+1)+":");
            char v1=inp.next().charAt(0);
            System.out.print("Please enter second vertex key for edge "+(i+1)+":");
            char v2=inp.next().charAt(0);
            System.out.print("Please enter the weight of the edge:");
            int w=inp.nextInt();
            thePrims.addEdge(m.get(v1),m.get(v2),w);
            System.out.println();
        }
        thePrims.mst(posStart);
    }

}
```

OUTPUT



```
run:
The following program demonstrates Minimum Spanning Tree using Prim's algorithm:-


**********INPUT**********


Please enter the no. of vertices:3


Please enter the key of starting vertex:0


Enter the key for vertex 1:0
Enter the key for vertex 2:1
Enter the key for vertex 3:2


Please enter the no. of edges:3


Please enter first vertex key for edge 1:0
Please enter second vertex key for edge 1:1
Please enter the weight of the edge:5

Please enter first vertex key for edge 2:1
Please enter second vertex key for edge 2:2
Please enter the weight of the edge:7

Please enter first vertex key for edge 3:0
Please enter second vertex key for edge 3:2
Please enter the weight of the edge:3



**********OUTPUT**********


The MST is:-

   Edge      Weight
0 ---- 2       3
0 ---- 1       5
BUILD SUCCESSFUL (total time: 1 minute 13 seconds)
```

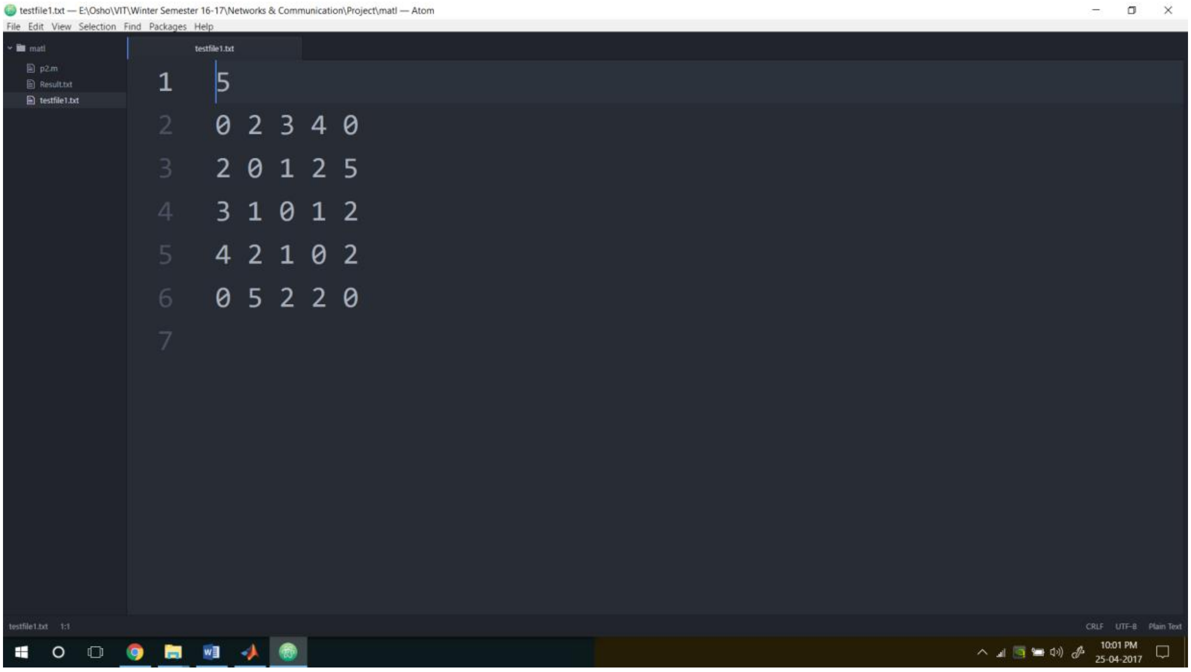
IMPLEMENTATION IN MATLAB



MATLAB R2015a
HOME
PLOTS
APPS
EDITOR
PUBLISH
VIEW
E: ▸ Osho ▸ VIT ▸ Winter Semester 16-17 ▸ Networks & Communication ▸ Project ▸ matl
Editor - E:\Osho\VIT\Winter Semester 16-17\Networks & Communication\Project\matl\p2.m
p2.m
dijkstra.m
testing.m
Ready
script
Ln 16 Col 1
09:42 PM
25-04-2017

```
clc;
fid = fopen('testfile1.txt', 'r');   % Input file
l = fscanf(fid, '%g %g', [1 1]) ;    % Input matrix size from line 1
h = fscanf(fid, '%g %g', [1 1])  ;   % Input the matrix
a=h';
fclose(fid);

fid = fopen('Result.txt','wt');     % Output file
fprintf(fid,'Original matrix\n\n'); % Printing the original matrix in the output file
for i=1:l
    for k=1:l
        fprintf(fid,'%6d',a(i,k));
    end
    fprintf(fid,' \n');
end

k=1:l ;
listV(k)=0 ;
listV(1)=1 ; %list of visited vertices
e=1;
while (e<l)
    min=inf;
    for i=1:l
        if listV(i)==1
            for j=1:l
                if listV(j)==0
                   if min>a(i,j) && i~=j && a(i,j)~=0
                        min=a(i,j);
                        b=a(i,j);
                        s=i;
                        d=j;
                    end
                end
            end
        end
    end
    listV(d)=1;
    distance(e)=b;
    source(e)=s;
    destination(e)=d;
    e=e+1;
end

fprintf(fid,'\nThe nodes and shortest distances are \n');
fprintf(fid,'\nFORMAT: Distance(Source, destination) \n');
for g=1:e-1
    fprintf(fid,'%d(%d,%d)\n',distance(g),source(g),destination(g));
end
status = fclose(fid);
```

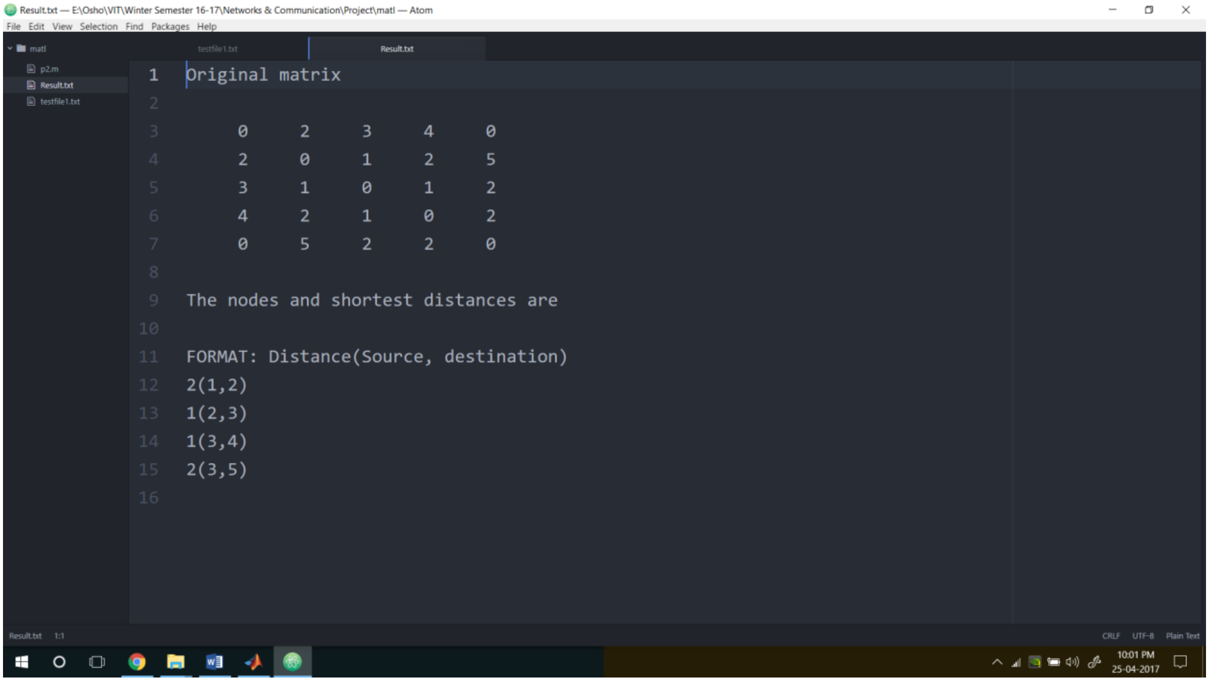
INPUT



testfile1.txt — E:\Osho\VIT\Winter Semester 16-17\Networks & Communication\Project\matl — Atom
File Edit View Selection Find Packages Help
matl
p2.m
Result.txt
testfile1.txt
testfile1.txt
testfile1.txt 1:1
CRLF UTF-8 Plain Text
10:01 PM
25-04-2017

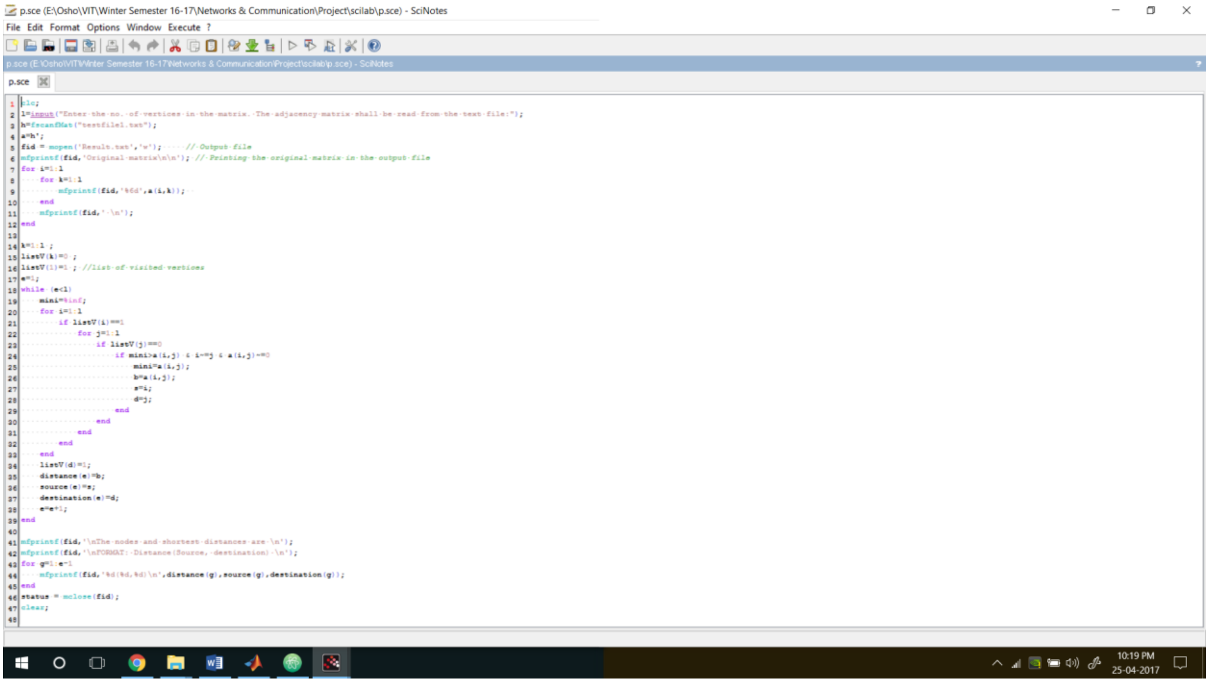
```
5
0 2 3 4 0
2 0 1 2 5
3 1 0 1 2
4 2 1 0 2
0 5 2 2 0

```

OUTPUT


Result.txt — E:\Osho\VIT\Winter Semester 16-17\Networks & Communication\Project\matl — Atom
File Edit View Selection Find Packages Help
matl
p2.m
Result.txt
testfile1.txt
testfile1.txt
Result.txt
Result.txt 1:1
CRLF UTF-8 Plain Text
10:01 PM
25-04-2017

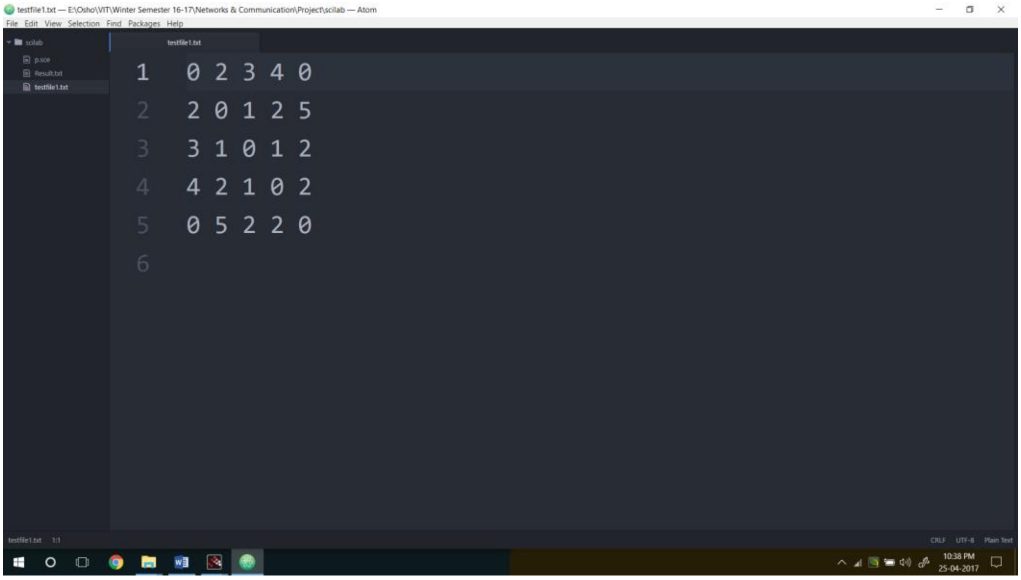
```
Original matrix

      0     2     3     4     0
      2     0     1     2     5
      3     1     0     1     2
      4     2     1     0     2
      0     5     2     2     0

The nodes and shortest distances are

FORMAT: Distance(Source, destination)
2(1,2)
1(2,3)
1(3,4)
2(3,5)

```

IMPLEMENTATION IN SCILAB


p.sce (E:\Osho\VIT\Winter Semester 16-17\Networks & Communication\Project\scilab\p.sce) - SciNotes
File Edit Format Options Window Execute ?
p.sce
10:19 PM
25-04-2017

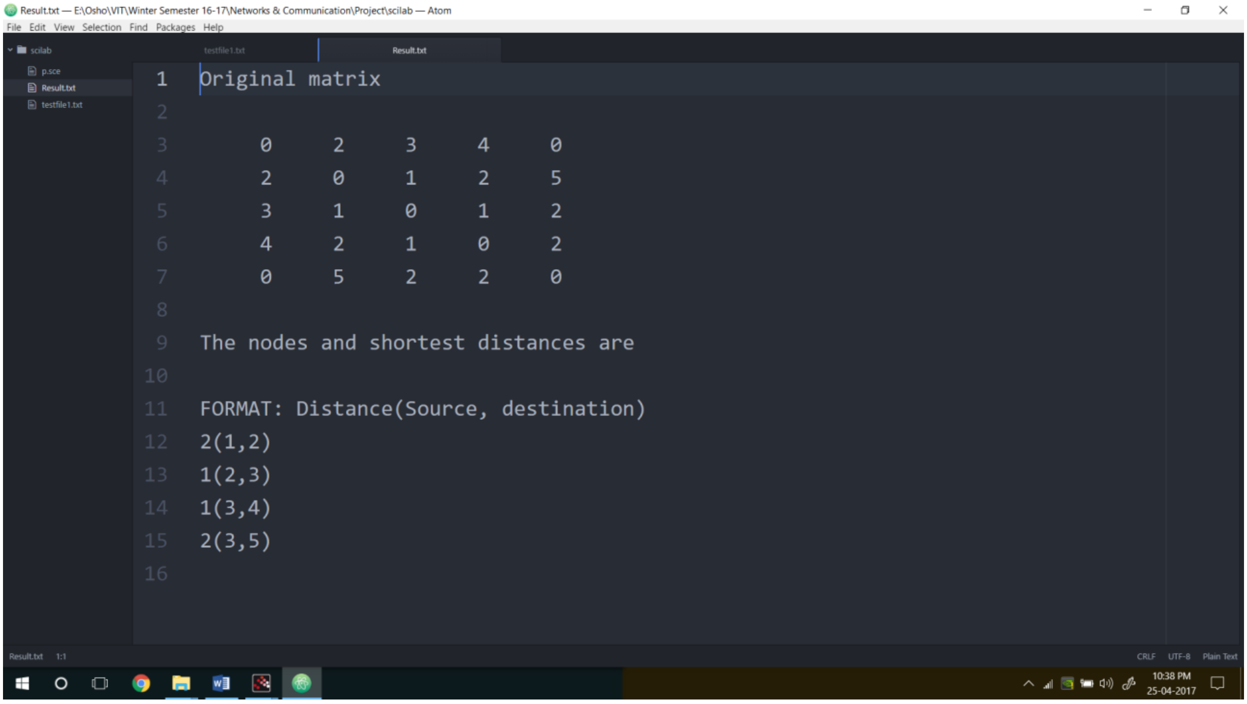
```
clc;
l=input("Enter the no. of vertices in the matrix. The adjacency matrix shall be read from the text file:");
h=fscanfMat("testfile1.txt");
a=h';
fid = mopen('Result.txt','w');     // Output file
mfprintf(fid,'Original matrix\n\n'); // Printing the original matrix in the output file
for i=1:l
    for k=1:l
        mfprintf(fid,'%6d',a(i,k));
    end
    mfprintf(fid,' \n');
end

k=1:l ;
listV(k)=0 ;
listV(1)=1 ; //list of visited vertices
e=1;
while (e<l)
    mini=%inf;
    for i=1:l
        if listV(i)==1
            for j=1:l
                if listV(j)==0
                    if mini>a(i,j) & i~=j & a(i,j)~=0
                        mini=a(i,j);
                        b=a(i,j);
                        s=i;
                        d=j;
                    end
                end
            end
        end
    end
    listV(d)=1;
    distance(e)=b;
    source(e)=s;
    destination(e)=d;
    e=e+1;
end

mfprintf(fid,'\nThe nodes and shortest distances are \n');
mfprintf(fid,'\nFORMAT: Distance(Source, destination) \n');
for g=1:e-1
    mfprintf(fid,'%d(%d,%d)\n',distance(g),source(g),destination(g));
end
status = mclose(fid);
clear;
```

INPUT


testfile1.txt — E:\Osho\VIT\Winter Semester 16-17\Networks & Communication\Project\scilab — Atom
File Edit View Selection Find Packages Help
scilab
p.sce
Result.txt
testfile1.txt
testfile1.txt 1:1
CRLF UTF-8 Plain Text
10:38 PM
25-04-2017

```
1  0 2 3 4 0
2  2 0 1 2 5
3  3 1 0 1 2
4  4 2 1 0 2
5  0 5 2 2 0
6
```

OUTPUT



Result.txt — E:\Osho\VIT\Winter Semester 16-17\Networks & Communication\Project\scilab — Atom
File Edit View Selection Find Packages Help
scilab
p.sce
Result.txt
testfile1.txt
testfile1.txt
Result.txt
Result.txt 1:1
CRLF UTF-8 Plain Text
10:38 PM
25-04-2017

```
Original matrix

     0     2     3     4     0
     2     0     1     2     5
     3     1     0     1     2
     4     2     1     0     2
     0     5     2     2     0

The nodes and shortest distances are

FORMAT: Distance(Source, destination)
2(1,2)
1(2,3)
1(3,4)
2(3,5)
```