Prims Algorithm:

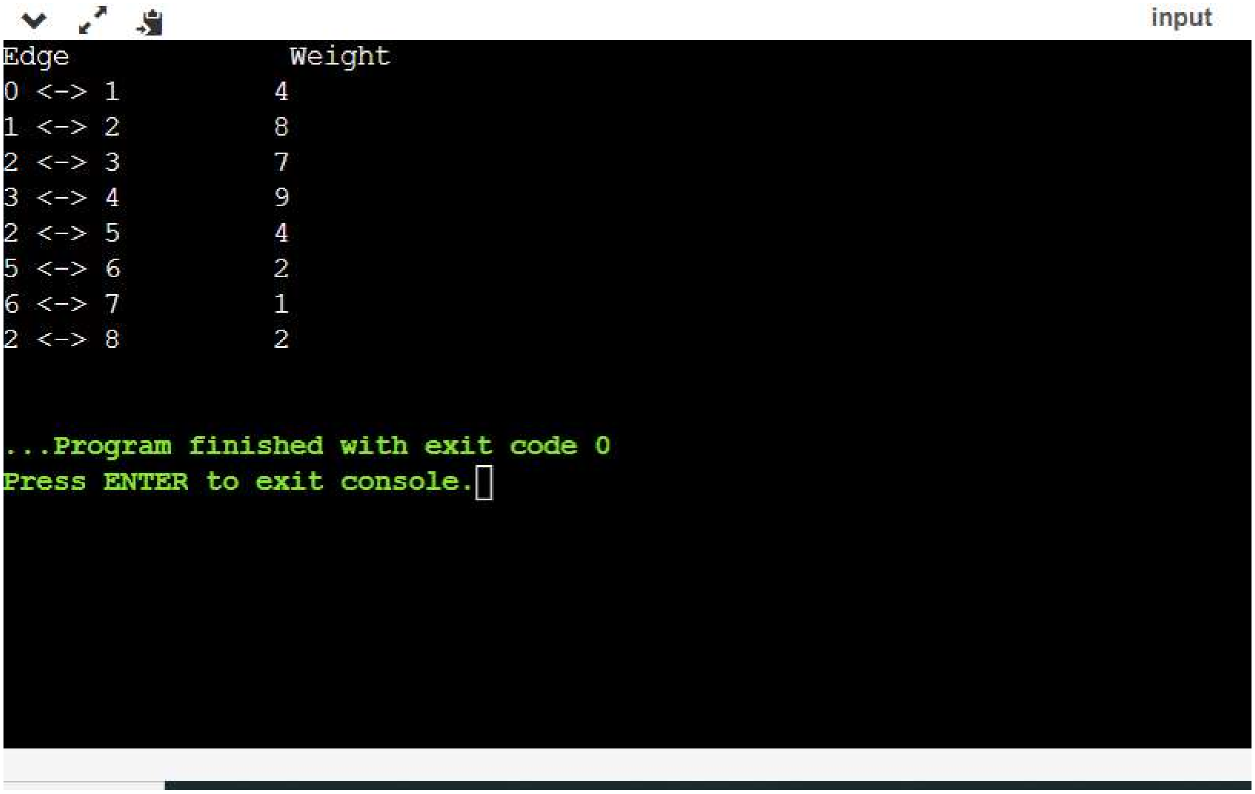
Name: E.JAYANTH REDDY

Reg.No.: 19BCE7548

Code:

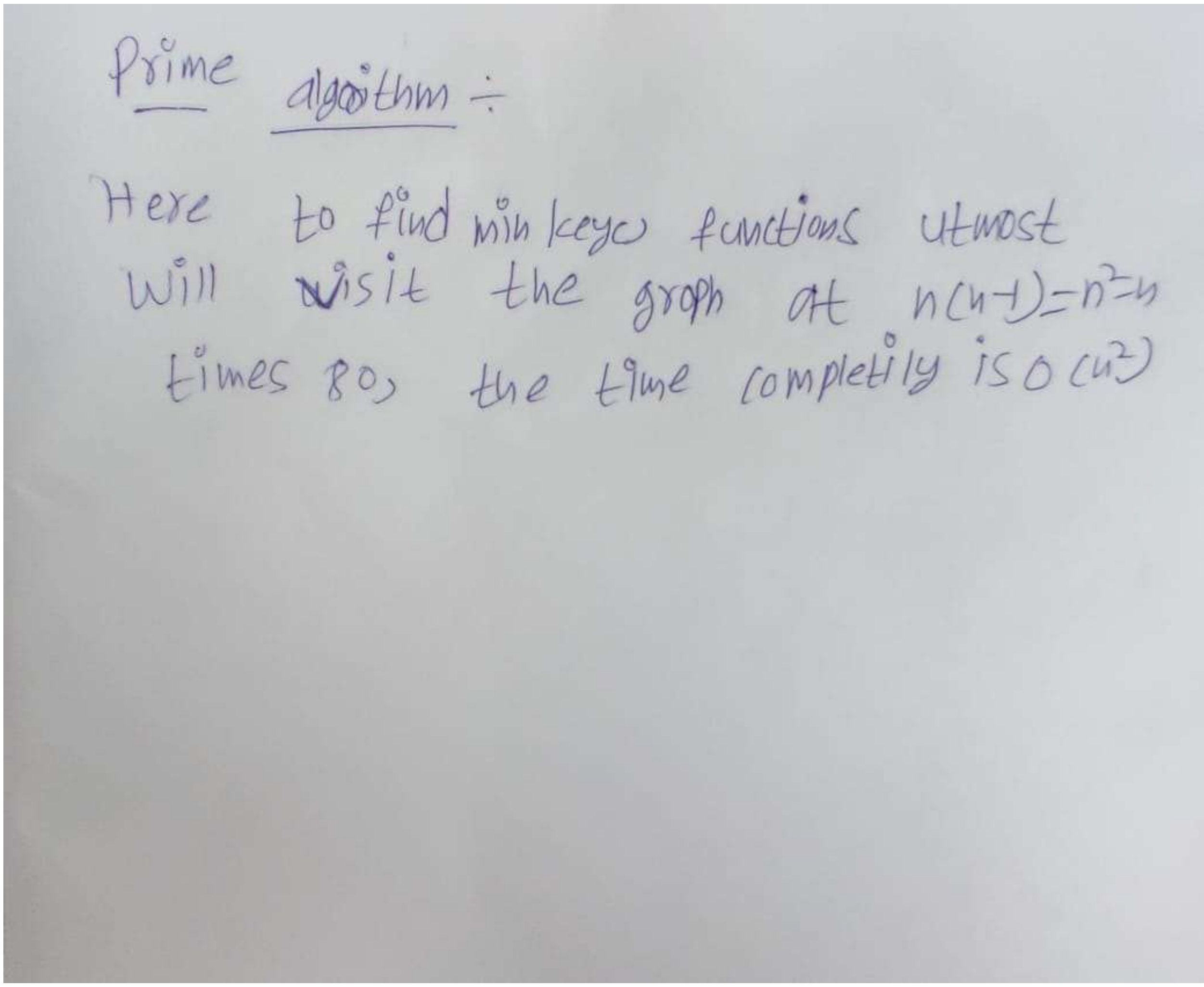
```
import java.lang.*;

import java.util.*;

import java.io.*;

class Main {

  private static final int countOfVertices = 9;


  int findMinKeyVertex(int keys[], Boolean setOfMST[])

  {

    int minimum_index = -1;

    int minimum_value = Integer.MAX_VALUE;


    for (int vertex = 0; vertex < countOfVertices; vertex++)

      if (setOfMST[vertex] == false && keys[vertex] < minimum_value) {

        minimum_value = keys[vertex];

        minimum_index = vertex;

      }


    return minimum_index;

  }


  void showMinimumSpanningTree(int mstArray[], int graphArray[][])

  {

    System.out.println("Edge \t\t Weight");
    for (int j = 1; j < countOfVertices; j++)
        System.out.println(mstArray[j] + " <-> " + j + "\t \t" + graphArray[j][mstArray[j]]);
}



void designMST(int graphArray[][])
{
    int mstArray[] = new int[countOfVertices];


    int keys[] = new int[countOfVertices];



    Boolean setOfMST[] = new Boolean[countOfVertices];


    for (int j = 0; j < countOfVertices; j++) {
        keys[j] = Integer.MAX_VALUE;
        setOfMST[j] = false;
    }
    keys[0] = 0; // it select as first vertex
    mstArray[0] = -1; // set first value of mstArray to -1 to make it root of MST


    for (int i = 0; i < countOfVertices - 1; i++) {
        int edge = findMinKeyVertex(keys, setOfMST);


        setOfMST[edge] = true;


        for (int vertex = 0; vertex < countOfVertices; vertex++)
        if (graphArray[edge][vertex] != 0 && setOfMST[vertex] == false && graphArray[edge][vertex] < keys[vertex]) {
          mstArray[vertex] = edge;
          keys[vertex] = graphArray[edge][vertex];
        }
    }

    showMinimumSpanningTree(mstArray, graphArray);
  }
  public static void main(String[] args)
  {

    Main mst = new Main();
    int graphArray[][] = new int[][]{{ 0, 4, 0, 0, 0, 0, 0, 8, 0 },
          { 4, 0, 8, 0, 0, 0, 0, 11, 0 },
          { 0, 8, 0, 7, 0, 4, 0, 0, 2 },
          { 0, 0, 7, 0, 9, 14, 0, 0, 0 },
          { 0, 0, 0, 9, 0, 10, 0, 0, 0 },
          { 0, 0, 4, 14, 10, 0, 2, 0, 0 },
          { 0, 0, 0, 0, 0, 2, 0, 1, 6 },
          { 8, 11, 0, 0, 0, 0, 1, 0, 7 },
          { 0, 0, 2, 0, 0, 0, 6, 7, 0 }};

    mst.designMST(graphArray);
  }
}
```

Output:



```
Edge             Weight
0 <-> 1         4
1 <-> 2         8
2 <-> 3         7
3 <-> 4         9
2 <-> 5         4
5 <-> 6         2
6 <-> 7         1
2 <-> 8         2


...Program finished with exit code 0
Press ENTER to exit console.
```

Analysis:



Prime algorithm :-

Here to find min key() functions utmost will visit the graph at $n(n-1)=n^2-n$ times so, the time completily is $O(n^2)$