Name: Anuj Mutha Class: TE4 Batch: M4

Assignment Number: 03

Title: Implement Greedy search algorithm for Dijkstra's Minimal Spanning Tree.

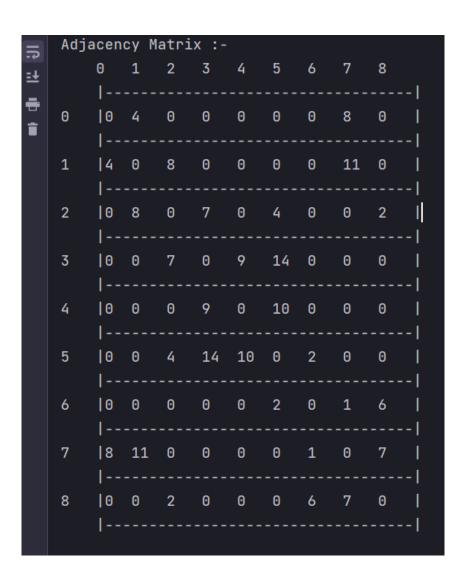
Code:

```
class Dijkstra{
   static final int V = 9;
    int minDistance(int dist[], Boolean sptSet[])
       int min = Integer.MAX \ VALUE, min index = -1;
            if (sptSet[v] == false && dist[v] <= min) {</pre>
               min = dist[v];
               min index = v;
       return min index;
    void printShortestPaths(int dist[], int src)
        System.out.println("" +
            System.out.println("| "+src+"->"+i + " \t| \t\t\t " +
dist[i]+"\t\t|");
       System.out.println("|-----|----|------|;);
    void computeShortestPath(int graph[][], int src)
       int dist[] = new int[V]; // The output array. Here, <math>dist[i] will hold the
       Boolean shortestPathTreeSet[] = new Boolean[V];
```

```
dist[i] = Integer.MAX VALUE;
           shortestPathTreeSet[i] = false;
       dist[src] = 0;
           int u = minDistance(dist, shortestPathTreeSet);
           shortestPathTreeSet[u] = true;
               if (!shortestPathTreeSet[v] && graph[u][v] != 0 && dist[u] !=
Integer.MAX VALUE && dist[u] + graph[u][v] < dist[v])</pre>
                   dist[v] = dist[u] + graph[u][v];
       printShortestPaths(dist,src);
   public static void main(String[] args)
        int graph[][] = new int[][]{
        System.out.println("\nAdjacency Matrix :- ");
        System.out.print("\t");
        for(int k=0;k<graph.length;k++){</pre>
           System.out.print(k+"\t");
       System.out.println("\n\t|-----|");
        for(int x=0;x<graph.length;x++) {</pre>
            System.out.print(x+"\t|");
            for(int y=0;y<graph[x].length;y++) {</pre>
                System.out.print(graph[x][y]+"\t");
            }System.out.print("|");System.out.println("\n\t|-----
```

```
System.out.println("\n\nShortest Paths from Source to ith vertex :- ");
Dijkstra t = new Dijkstra();
t.computeShortestPath(graph, 0);
}
```

Output:



		Source to i	
		tance from S	ource
	 	0	I I
0->1	1	4	1
0->2	1	12	1
0->3	1	19	1
0->4	1	21	1
0->5	1	11	1
0->6	1	9	1
0->7	1	8	1
0->8	1	14	1
			1

Graph (Figure):

