Activity 11.1

Ramzy Izza Wardhana - 21/472698/PA/20322

Run2Dijkstra.java

```java
import java.util.Scanner;

public class Run2Dijkstra {
    static double[][] map;
    static int src;
    static int dst;

    public static void main (String[] args){
        doInput();
        Dijkstra dj = new Dijkstra(map);
        dj.solve(src, dst);
        System.out.println(dj.getDistance(dst));
    }

    public static void doInput() {
        Scanner sc = new Scanner(System.in);
        int nTown = sc.nextInt(); //Number of Node
        int nRoute = sc.nextInt(); //Number of Edge
        map = new double[nTown][nTown]; //Adjacency Matrix

        for (int i = 0; i < nRoute; i++){
            int from = sc.nextInt();
            int to  = sc.nextInt();
            double len = sc.nextDouble();
            map[from][to] = map[to][from] = len;
        }

        //Source and destination
        src = sc.nextInt();
        dst = sc.nextInt();
    }
}
```

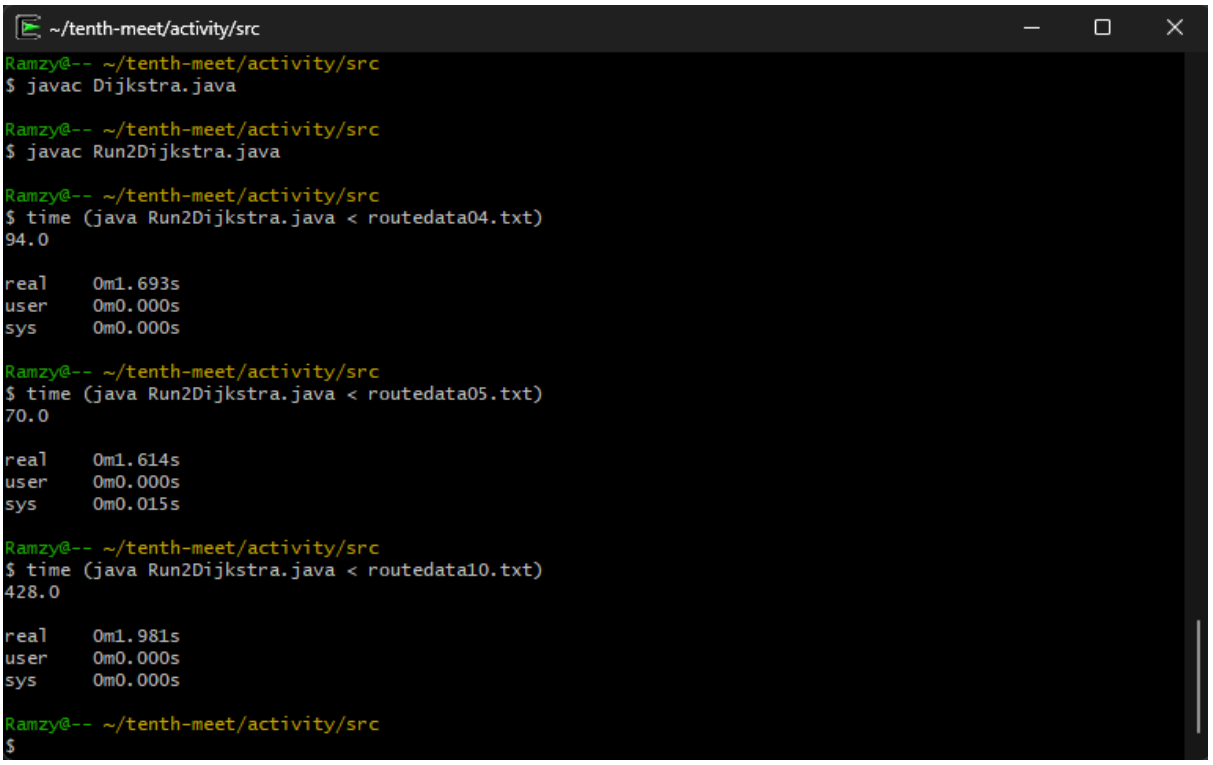
Dijkstra.java

```java
public class Dijkstra {
    int nTown;
    double[][] map;
    double[] distance;
    int src;
```

```
    public Dijkstra(double[][] map) {
        this.map = map;
        nTown = map.length; // number of towns
    }

    public void solve(int src, int dst) {
        this.src = src; //Set Starting Node
        boolean[] selected = new boolean[nTown]; //To verify whether node already
visisted or not
        distance = new double[nTown]; //To store distance from source to each node

        for(int i = 0; i < nTown; i++){
            distance[i] = Double.MAX_VALUE; //Set all distance to infinity
            selected[i] = false; //Set all node to unvisited
        }
        distance[src] = 0; //Set distance from source to source to 0

        while (true) {
            int marked = minIndex(distance, selected);

            if (marked < 0) return; //If all node already visited, stop
            if (distance[marked] == Double.MAX_VALUE) return; //If all node is
unreachable, stop
            selected[marked] = true; //Mark node as visited
            if (marked == dst) {
                return; // If destination node is reached, stop
            }


            for (int j = 0; j < nTown; j++){ //For every connected node
                if(map[marked][j] > 0 && !selected[j]){
                    double newDistance = distance[marked] + map[marked][j];
//Calculate new distance
                    if(newDistance < distance[j]) distance[j] = newDistance; //If
new distance is shorter, update distance
                }
            }
        }
    }

    public int minIndex(double[] distance, boolean[] selected){
        double dist  = Double.MAX_VALUE; //Set distance to infinity
        int index = -1; //Set index to -1
```

```
        for (int i = 0; i < nTown; i++) {
            if (!selected[i] && distance[i] < dist) { // If node is unvisited and
distance is shorter than current distance
                dist = distance[i]; // Update distance
                index = i; // Update index
            }
        }
        return index;
    }

    public double getDistance(int dst) {
        return (int)distance[dst];
    }
}
```

Output



Routedata10.txt (Modified the distance from node 0 -> node 8099)