Project 1

Algorithmic Aspects of Telecommunication Networks (CS 6385)

An Application to Network Design

Rajan Jhaveri
rjj160330

Table of Content

PROBLEM STATEMENT

- CREATING A SOFTWARE WITH THE GIVEN REQUIREMENTS

- As input, it receives the number of nodes (N), the traffic demand values (bij) between pairs of nodes, and the unit cost values for the potential links (aij).
- As output, the program generates a network topology, with capacities assigned to the links, according to the studied model, using the shortest path based fast solution method (see at the end of the referred lecture note). The program also computes the total cost of the designed network.
- Let the number of nodes be N = 20 in each example.
- bij values, take your 10-digit student ID, and repeat it 2 times, to obtain a 20-digit number. For example, if the ID is 0123456789, then after repetition it becomes 01234567890123456789. Let d1, d2, . . . , d20 denote the digits in this 20-digit number. Then the value of bij is computed by the formula

  $$b_{ij} = |d_i - d_j|.$$

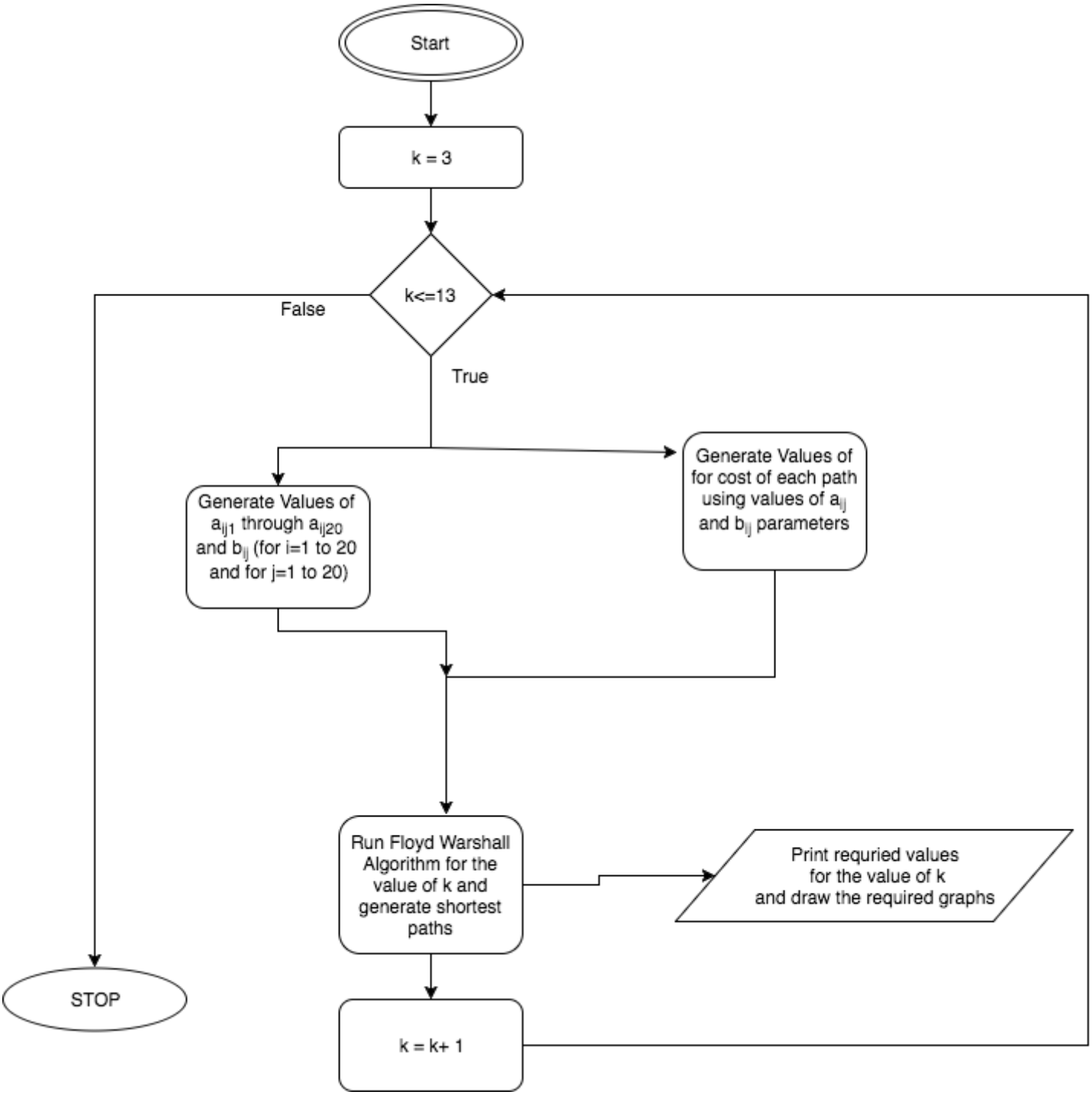
- For generating the aij values, do the following. For any given i, pick k random indices j1, j2, . . . , jk, all different from each other and from i. Then set $a_{ij1} = a_{ij2} = \ldots = a_{ijK} = 1$, and set $a_{ij} = 200$, whenever $j_6 \neq j_1, \ldots, j_k$. Carry out this independently for every i.

<u>OUTPUTS EXPECTED</u> :

- How does the total cost of the network depends on k?
- How does the density of the obtained network depends on k? Here the density is defined as the number of directed edges that are assigned nonzero capacity, divided by the total possible number of directed edges, which is $N(N - 1)$.
- Show some of the obtained network topologies graphically. Specifically, draw three of them: one with k = 3, one with k = 8, and one with k = 13.

FLOW CHART

- An overview of how the program works explained in the form of a flowchart .



FLOYD WARSHALL ALGORITHM

- Floyd-Warshall algorithm is a procedure, which is used to find the shorthest (longest) paths among all pairs of nodes in a graph, which does not contain any cycles of negative lenght. The main advantage of Floyd-Warshall algorithm is its simplicity.
- The Floyd–Warshall algorithm compares all possible paths through the graph between each pair of vertices. It is able to do this with $\Theta(|V|^3)$ comparisons in a graph. This is remarkable considering that there may be up to $\Omega(|V|^2)$ edges in the graph, and every combination of edges is tested. It does so by incrementally improving an estimate on the shortest path between two vertices, until the estimate is optimal.
- Consider a graph G with vertices V numbered 1 through N. Further consider a function shortestPath(i, j, k) that returns the shortest possible path from i to j using vertices only from the set {1,2,...,k} as intermediate points along the way. Now, given this function, our goal is to find the shortest path from each i to each j using only vertices 1 to $k + 1$.
- For each of these pairs of vertices, the true shortest path could be either
- (1) a path that only uses vertices in the set {1,2,….,k}.

  or

  (2) a path from i to k+1 and then from k+1 to j.

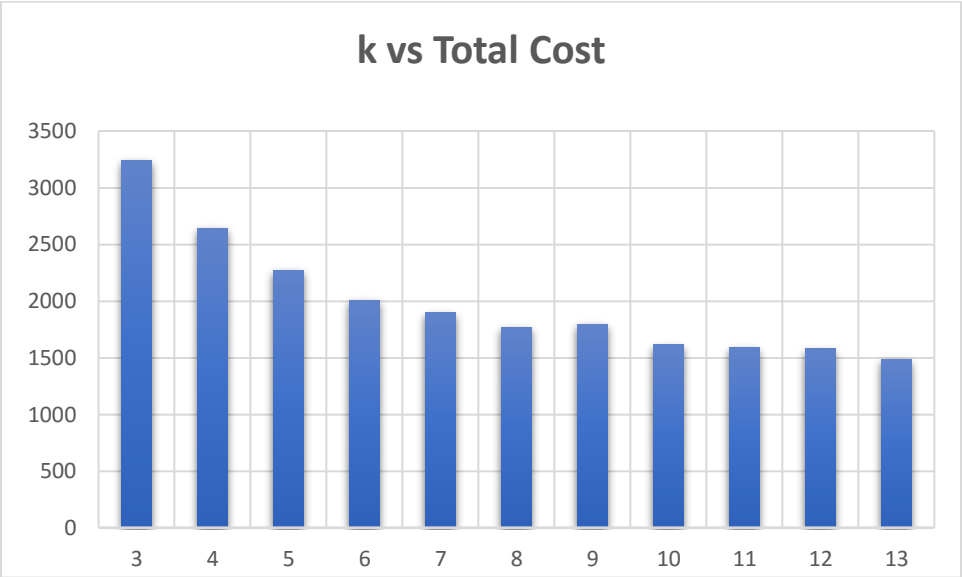
PSEUDOCODE

```
1 let dst be a |V| × |V| array of minimum distances initialized to ∞ (infinity)
2 for each vertex v
3    dst[v][v] ← 0
4 for each edge (u, v)
5    dst[u][v] ← w(u, v)  // the weight of the edge (u, v)
6 for k from 1 to |V|
7   for i from 1 to |V|
8     for j from 1 to |V|
9       if dst[i][j] > dst[i][k] + dst[k][j]
10          dst[i][j] ← dst[i][k] + dst[k][j]
11      end if
```

TOTAL COST OF NETWORK

- Values of Cost for values of K.

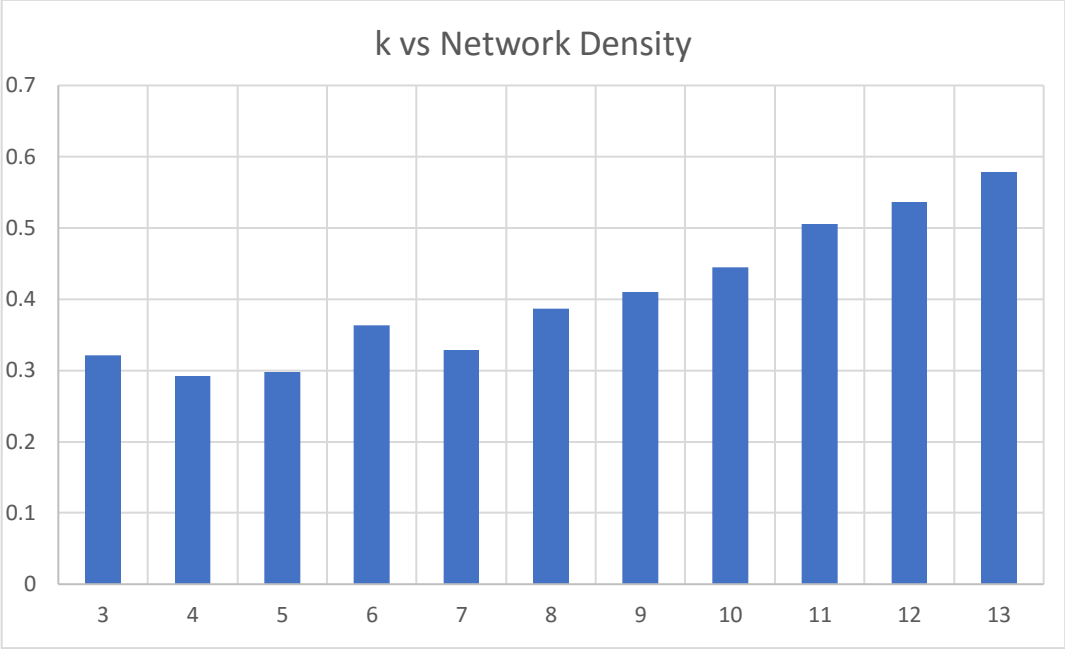
K	COST
3	3242
4	2644
5	2272
6	2008
7	1899
8	1769
9	1797
10	1614
11	1588
12	1578
13	1486



- Here, as seen in the graph, initially with low value of k the cost is high but as value of k increases the cost of the network decreases, the reason being increase in number of links with cost value 1. But after some value of k the cost tends to become constant, because of same reason.

DENSITY OF THE NETWORK

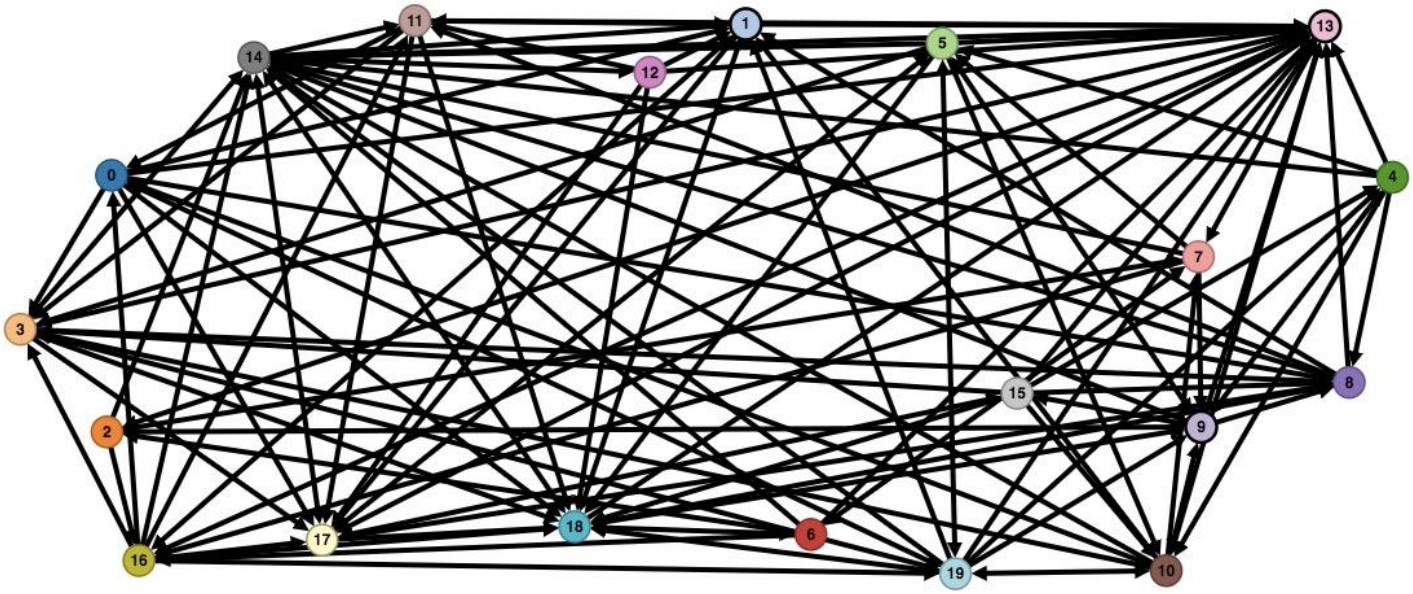
- Values of Network Density for values of K.



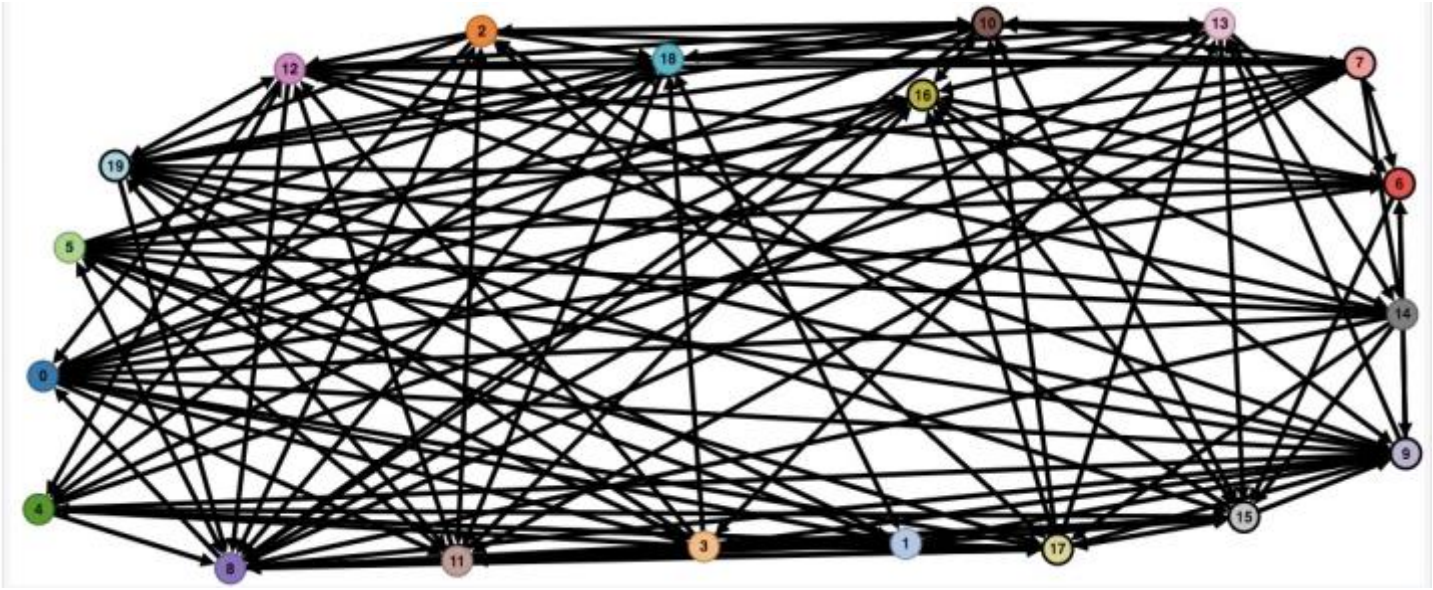
- Here, as seen in the graph, with increase in value of k the network density keeps on increasing.

NETWORK TOPOLOGIES(GRAPHS)

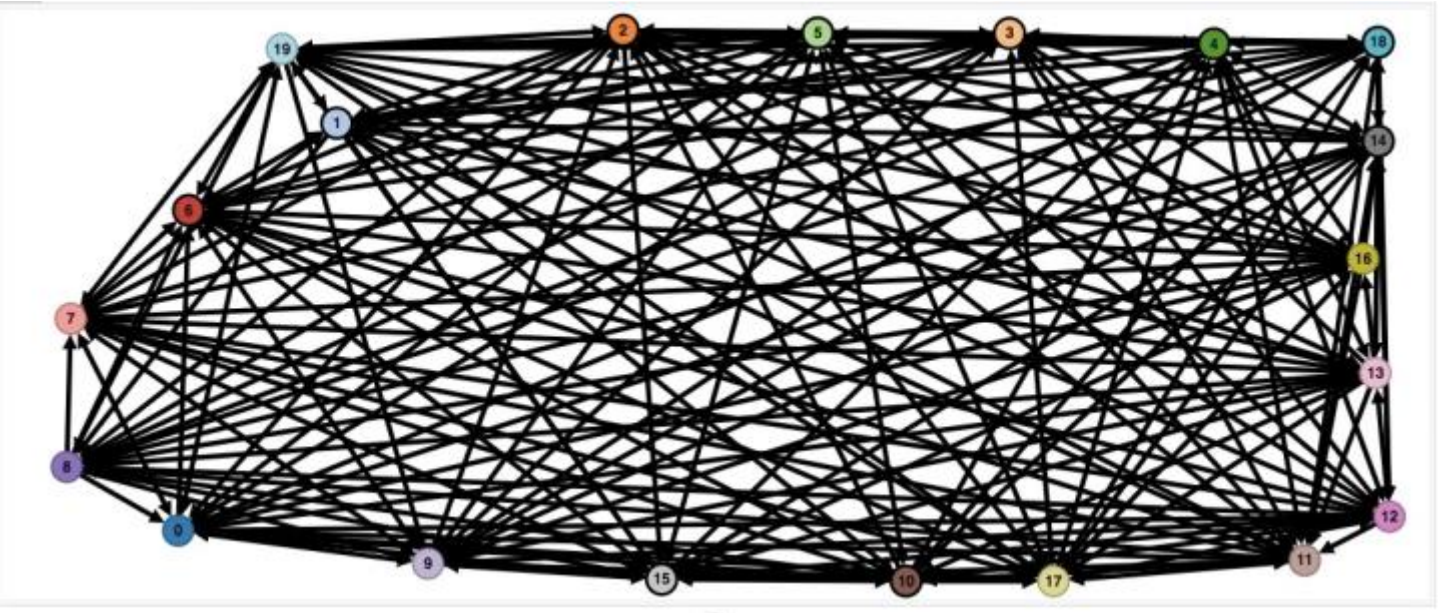
- **For value k=3**



- **For value k=8**



- **For value k = 13**



APPENDIX

```
import java.util.ArrayList;
import java.util.Arrays;
import java.util.Collections;


public class ShortestPath {

            public static void FW(int a[][],int b[][],int c[][],int k)
  {
    int sp[][]=new int[20][20];
    int adj_mat[][]=new int[20][20];
    int total_cost=0;

    String output="";
    for(int i=0;i<20;i++)
    {
      for(int j=0;j<20;j++)
      {
        sp[i][j]=-1;
        adj_mat[i][j]=0;
      }
    }

    for(int m=0;m<20;m++)
    {
      for(int i=0;i<20;i++)
      {
        for(int j=0;j<20;j++)
        {
          if(a[i][m]+a[m][j]<a[i][j])
          {
            a[i][j] =a[i][m] + a[m][j];
            sp[i][j]=m;

          }
        }
      }
    }
    for(int i=0;i<20;i++)
    {
      for(int j=0;j<20;j++)
      {
        int node=sp[i][j];
        int src=i;
        while(node !=-1)
        {
          c[src][node]=c[src][node]+b[src][node];
          adj_mat[src][node]=1;
          src=node;
          node=sp[src][j];
        }
        c[src][j]+=b[i][j];
        adj_mat[src][j]=1;
      }
    }

    for(int i=0;i<20;i++)
    {
      for(int j=0;j<20;j++)
      {
        total_cost=total_cost+a[i][j]*c[i][j];

      }
```

```
    }
    System.out.println("Total cost : "+total_cost);

    int nonZeroEdges=0;
    for(int i=0;i<20;i++)
    {
       for(int j=0;j<20;j++)
       {
          if(c[i][j]!=0)
          {
             nonZeroEdges++;
          }

       }
    }

    int total_edges=380;
    float density=(float)nonZeroEdges/total_edges;
    System.out.println("Density : "+density);

    for(int i=0;i<20;i++)
    {
       output+=""+i+":[";
       int flag =0;
       for(int j=0;j<20;j++)
       {
          if(adj_mat[i][j]!=0)
          {
             if(flag==0){
                output+= j;
                flag = 1;
             }
             else{
             output+="," + j;
             }
          }

       }
       output+= "],";
    }
    System.out.println(output);

}

                                    public static boolean contains(final int[] array, final int v) {

                                boolean result = false;

                                for(int i : array){
                                   if(i == v){
                                      result = true;
                                      break;
                                   }
                                }

                                return result;
                             }

            public static void main(String[] args) {
                                // TODO Auto-generated method stub
                                //System.out.println("Hello World");
    int n=20;
    int a[][] = new int[n][n];
    int b[][] = new int[n][n];
    int d[][] = new int[n][n];

    int[] c = {2,0,2,1,3,2,7,9,4,4,2,0,2,1,3,2,7,9,4,4};
```

```
    for(int k=3; k<14;k++){
    int[] val = new int[k];

    ArrayList<Integer> list = new ArrayList<Integer>();
    for (int i=0; i<20; i++) {
       list.add(new Integer(i));
    }


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

            Collections.shuffle(list);
       for (int m=0; m<3; m++) {
          val[m] =list.get(m);
       }


       for(int j=0;j<n;j++){
          if(contains(val, j)){
            a[i][j] = 1;
          }
          else{
            a[i][j] = 200;
          }
       }
    }

    for(int i=0;i<n;i++){
       for(int j=0;j<n;j++){
          b[i][j]= Math.abs(c[i]-c[j]);
       }
    }

    for(int i=0;i<n;i++){
       for(int j=0;j<n;j++){
            d[i][j]=0;
       }
    }

    System.out.println("\n Value of k = "+k);
                        ShortestPath t = new ShortestPath();
                        t.FW(a,b,d,k);
    }
            }
}
```

REFERENCES, WEBSITES AND SOFTWARE USED

Reference:

1) **Lecture Notes**

2) **Wikipedia - https://en.wikipedia.org/wiki/Floyd%E2%80%93Warshall_algorithm**

Websites :

1) **https://www.draw.io/ – Tool for drawing Flowchart**

2) **http://yiboyang.com/graphrel/ - Tool for drawing Graph using the edges**

Software :

1) **Eclipse (For java Development)**