23-11-20

CN-Lab

GURU NANNA
1BM18CS081

Write a program for distance vector algorithm to find suitable path for transmission.

```
import java.io.*;
public class DVR {
    static int graph[][];
    static int via[][];
    static int rt[][];
    static int v;
    static int e;

    public static void main (String args[]) throws
                                    IO Exception
    {
        BufferedReader br = new BufferedReader
                (new InputStreamReader (System.in));
        System.out.println ("Please enter the no. of
                        vertices, and edges: ");
        V = Integer.parseInt (br.readLine());
        System.out
        e = Integer.parseInt (br.readLine());
```

```
graph = new int[v][v];
via = new int[v][v];
st = new int[v][v];

for (int i=0; i<v; i++)
    for (int j=0; j<v; j++) {
        if (i == j)
            graph[i][j] = 0;
        else
            graph[i][j] = 9999;
    }

for (int i=0; i<e; i++) {
    System.out.println("Please enter the source,
        destination and the cost between them: ");
    int s = Integer.parseInt(br.readLine());
    s--;
    int d = Integer.parseInt(br.readLine());
    d--;
    int c = Integer.parseInt(br.readLine());
    graph[s][d] = c;
    graph[d][s] = c;
```

~~System Code~~

```
init_tables();
update_tables();
System.out.println("The initial Routing Tables are: ");
print_tables();
```

(2)

```
int check = 1;
while(check != 0){
    System.out.println("Enter 1 to change cost of
        any edge, otherwise 0");
    if (check == 0)
        break;
    System.out.println("Enter new cost, and between
        which vertices :");
    int c = Integer.parseInt(br.readLine());
    int s = Integer.parseInt(br.readLine());
    s--;
    int d = Integer.parseInt(br.readline());
    d--;
    graph[s][d] = c;
    graph[d][s] = c;

    init_tables()
    updatetables();
    System.out.println("The new Routing Tables are:");
    print_table();
}
}
static void update_tables() {
    int k = 0;
    for (int i=0; i< 4 * V; i++) {
        update_table(k)
        k++;
        if (k == V)
            k = 0;
    }
}
```

```
static void update_table(int source) {
    for (int i=0; i<v; i++) {
        if (graph[source][i] != 9999) {
            int dist = graph[source][i];
            for (int j=0; j<v; j++) {
                int i_dist = st[i][j];
                if (via[i][j] == source)
                    i_dist = 9999
                if (dist + i_dist < st[source][j])
                {
                    st[source][j] = dist + i_dist;
                } via[source][j] = i;
            }
        }
    }
}

static void init_tables() {
    for (int i=0; i<v; i++) {
        for (int j=0; j<v; j++) {
            if (i==j) {
                st[i][j] = 0;
            } via[i][j] = i;
            else {
                st[i][j] = 9999;
                via[i][j] = 100;
            }
        }
    }
}
```

```
static void print_tables() {
        for (int i=0; i<V; i++) {
                System.out.println("Shortest distance from " + i);
                for (int j = 0; j < V; i++) {
                        System.out.print("to " + j + ": " + st[i][j]
                }
                System.out.println();
        }
    }
}
```