Aim:

To Implement Distance Vector And Link State Routing Algorithm in Java

Algorithm:

1.***Distance Vector Routing*** :

1. Accept all nodes in the form of adjacency matrix
2. After accepting the nodes we run Bellman Ford Algorithm
3. Here we discover all nodes and update vector according to cost if less than existing
4. Mark unreachable nodes as inf
5. Traverse in lexicographic order
6. Repeat step 3
7. Change source nodes and run through same algorithm and print distance vector
8. Indices are the nodes

2.***Link State Routing Algorithm***:

1. Accept All Inputs as adjacency Matrix

2. After accepting the nodes we run Dijkstra Algorithm

3, Here also we discover the nodes and keep track of previous nodes in the form od ArrayList in Java [Node no,isDiscovered,cost,Node Previous]

4. We run the loop until all IsDiscovered Subarray is 1

5.We cut the array into 4 part chunks as Subarray in java for easier use and manipulation of list entries

6.Inside the loop we check for minimum cost nodes and jump to that node and discover others while we keep track of existing node cost and update the subarray of neighbouring nodes iff the cost + adj[i][j]<existing cost.

7.We write another function to traceback the path to 0 or source node by jumping to the subarray.get[3]->pre node and print the path + cost

Code:

Distance Vector Routing.java

```
package Exercise9;
import java.util.ArrayList;
import java.util.List;
class BellmanFord{
	int [][] adjmat;
	int source,len;
	int inf=45558;
	BellmanFord(int[][]adjmat,int arrlen,int source){
		this.adjmat=adjmat;
		this.source=source;
		len=arrlen;
	}
	List <Integer>vector=new ArrayList <Integer>(len);
	void initialize() {
		for(int i=0;i<len;i++) {

				vector.add(adjmat[source][i]);
			}
	}
	void iterate() {
		initialize();
		int cur;
		int j=(source+1)%len;

		int [][] arr1=new int[len-1][len];
		int row=0;
			while(j!=source) {
				cur=vector.get(j);
				for(int k=0;k<len;k++) {
					arr1[row][k]=adjmat[j][k];
```

```
                }
                row++;
                if(row==len-1) {break;}

                j++;
                j=j%4;
            }
            j=(source+1)%len;
            for(int i=0;i<len-1;i++) {
                cur=vector.get(j);
                for(int k=0;k<len;k++) {
                    if(Math.abs(cur+arr1[i][k])<vector.get(k)) {

                        vector.set(k, (cur+arr1[i][k]));
                    }
                }
                j++;
                j=j%len;
            }
    }
    void printVector() {
        for(int i=0;i<len;i++) {
            System.out.print(vector.get(i)+"\t");
        }
        System.out.println();
    }
    void PrintRouterVectorTable(int[][]mat,int arrlen,int src) {
        for(int i=0;i<len;i++) {
            System.out.println(i+" Routing Table: ");
            this.source=i;
            vector=new ArrayList <Integer>(len);
```

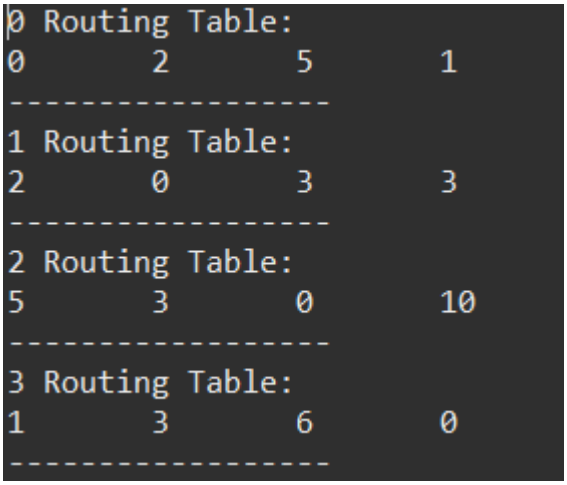
```
                iterate();
                printVector();
                System.out.println("------------------");
            }
        }

}
public class distanceVectorRouting {
        public static void main(String []args) {
            int inf=45556;
            int [][] adjacencyMatrix= {{0,2,inf,1},
                                       {2,0,3,7},
                                       {inf,3,0,11},
                                       {1,7,11,0}};
            BellmanFord f= new BellmanFord(adjacencyMatrix,4,0);
            //f.PrintRouterVectorTable(adjacencyMatrix,4,0);
            f.iterate();
            f.printVector();

        }

}
```

Output:



LinkStateRouting.java:

```
package Exercise9;
import java.util.*;
class dijkstra{
	int mat[][];
	int nodes,node_no=0,n_nodes;
		// Node no , isVisited, Distance , prev
	dijkstra(int n_nodes,int[][] mat){
		this.mat=mat;
		this.n_nodes=n_nodes;
	}
	ArrayList <Integer> sl=new ArrayList<Integer>(n_nodes*n_nodes);
	int k=0;
	void set() {
		sl.add(k++,node_no++ );
		sl.add(k++, 0);
		sl.add(k++, 0);
		sl.add(k++, null);
		for(int i=1;i<n_nodes;i++) {
			sl.add(k++,node_no++ );
			sl.add(k++, 0);
			sl.add(k++, Integer.MAX_VALUE);
			sl.add(k++, null);
		}

	}
	void print() {
		for(int i=0;i<sl.size();i+=4) {
			//System.out.println(i);
			System.out.println(sl.subList(i, i+4));
		}
	}
```

```
int findmin(ArrayList <Integer> k) {
        int min=k.subList(4, 8).get(2),idx=k.subList(4, 8).get(0);

        for(int i=4;i<mat.length*4;i+=4) {
                //System.out.println(k.subList(i,i+4).get(2)+"====");

                if(k.subList(i,i+4).get(2)<=min && (k.subList(i,i+4).get(1))==0) {

                        min=k.subList(i,i+4).get(2);
                        idx=k.subList(i, i+4).get(0);
                }
        }
        //System.out.println(min+"  "+idx);
        return idx;
}
void run() {
        int cur;
        int iter=0;

        int k=0,m,i,cost=0;
        for(int j=0;j<mat.length;j++) {
                i=j*4;
                if(mat[k][j]>0) {
                        if(sl.subList(i,i+4).get(2)==Integer.MAX_VALUE) {
                                sl.subList(i, i+4).set(2, mat[k][j]);
                                sl.subList(i, i+4).set(3, 0);
                        }
                }
        }
        sl.subList(0, 4).set(1, 1);
        int listidx,precost,pre,collst,tmp;
```

```
        while(!isVisited()) {
                i=findmin(sl);
                //print();
                listidx=i*4;
                precost=sl.subList(listidx,listidx+4).get(3);
                cost=sl.subList(precost*4,(precost*4)+4).get(2);
                //cost=sl.subList(listidx,listidx+4).get();
                for(int c=0;c<mat.length;c++) {
                        if(mat[i][c]>0) {
                                collst=c*4;
                                 if (cost+mat[i][c]<sl.subList(collst,collst+4).get(2)) {
                                         tmp=sl.subList(collst,collst+4).get(2);
                                        sl.subList(collst, collst+4).set(2, cost+mat[i][c]);
                                        sl.subList(collst,collst+4).set(3, i);
                                }
                        }
                }
                sl.subList(listidx, listidx+4).set(1, 1);
        }


}
boolean isVisited() {

        for(int i=0;i<mat.length;i+=4) {
                if(sl.subList(i, i+4).get(1)==0) {
                        return false;
                }
        }
        return true;
}
void RouteConfigPrint(int targetNode) {
```

```
                int listidx=targetNode;
                listidx*=4;
                int cost=0;
                System.out.print("0");
                while(sl.subList(listidx, listidx+4).get(0)!=0) {
                        cost+=sl.subList(listidx, listidx+4).get(2);
                        System.out.print("-->"+sl.subList(listidx, listidx+4).get(0));
                        listidx=sl.subList(listidx, listidx+4).get(3);
                        listidx*=4;
                }
                System.out.print("=" +cost);
                cost=0;
        }
}
public class linkStateRouting {
        public static void main(String []  args) {


                int [][]adjmat=   { { 0, 0, 1, 2, 0, 0, 0 },
                                  { 0, 0, 2, 0, 0, 3, 0 },
                                  { 1, 2, 0, 1, 3, 0, 0 },
                    { 2, 0, 1, 0, 0, 0, 1 },
                    { 0, 0, 3, 0, 0, 2, 0 },
                    { 0, 3, 0, 0, 2, 0, 1 },
                    { 0, 0, 0, 1, 0, 1, 0 } };


                dijkstra s=new dijkstra(7,adjmat);
                s.set();
                //s.findmin(s.sl);
                s.run();


                s.print();
```

```
            for(int i=1;i<7;i++) {
                System.out.println();
                s.RouteConfigPrint(i);
        }

        }
}
```

Output:

```
[0, 1, 0, null]
[1, 1, 2, 2]
[2, 1, 1, 0]
[3, 1, 1, 2]
[4, 0, 3, 2]
[5, 1, 2, 6]
[6, 1, 2, 3]

0-->1-->2=3
0-->2=1
0-->3-->2=2
0-->4-->2=4
0-->5-->6-->3-->2=6
0-->6-->3-->2=4
```

Result:

Thus Distance Vector And Link State Routing Algorithms were studied and implemented in Java