**Lab Practical #13:**

To develop network using distance vector routing protocol and link state routing protocol.

**Practical Assignment #13:**

1. **C/Java Program: Distance Vector Routing Algorithm using Bellman Ford's Algorithm.**

**import java.util.Scanner;**

**public class DistanceVectorRouting {**

**private static final int INF = 9999;**

**public static void main(String[] args) {**

**Scanner scanner = new Scanner(System.in);**

**System.out.print("Enter the number of routers: ");**

**int numRouters = scanner.nextInt();**

**int[][] costMatrix = new int[numRouters][numRouters];**

**System.out.println("Enter the cost matrix (use " + INF + " for infinity):");**

**for (int i = 0; i < numRouters; i++) {**

**for (int j = 0; j < numRouters; j++) {**

**costMatrix[i][j] = scanner.nextInt();**

**}**

**}**

**int[][] distanceVector = new int[numRouters][numRouters];**

**int[][] nextHop = new int[numRouters][numRouters];**

**for (int i = 0; i < numRouters; i++) {**

**for (int j = 0; j < numRouters; j++) {**

**distanceVector[i][j] = costMatrix[i][j];**

**nextHop[i][j] = (costMatrix[i][j] != INF && i != j) ? j : -1;**

**}**

**}**

**boolean updated;**

**do {**

**updated = false;**

**for (int i = 0; i < numRouters; i++) {**

**for (int j = 0; j < numRouters; j++) {**

**for (int k = 0; k < numRouters; k++) {**

**if (distanceVector[i][k] + distanceVector[k][j] < distanceVector[i][j]) {**

**distanceVector[i][j] = distanceVector[i][k] + distanceVector[k][j];**

**nextHop[i][j] = nextHop[i][k];**

**updated = true;**

**}**

**}**

**}**

**}**

**} while (updated);**

**System.out.println("\nFinal Distance Vector Table:");**

**for (int i = 0; i < numRouters; i++) {**

**System.out.println("Router " + (i + 1) + ":");**

**for (int j = 0; j < numRouters; j++) {**

**if (distanceVector[i][j] == INF) {**

**System.out.print("INF ");**

**} else {**

**System.out.print((distanceVector[i][j] + 1) + " ");**

**}**

**}**

**System.out.println();**

**}**

**scanner.close();**

**}**

**}**

**2. C/Java Program: Link state routing algorithm.**

**import java.util.\*;**

**public class Dijkstra {**

**static final int INF = Integer.MAX\_VALUE;**

**static int findKey(boolean[] visited, int[] distance, int V) {**

**int min = INF;**

**int key = -1;**

**for (int i = 0; i < V; i++) {**

**if (!visited[i] && distance[i] < min) {**

**min = distance[i];**

**key = i;**

**}**

**}**

**return key;**

**}**

**static void dijkstra(int[][] graph, int src) {**

**int V = graph.length;**

**boolean[] visited = new boolean[V];**

**int[] distance = new int[V];**

**Arrays.fill(distance, INF);**

**distance[src] = 0;**

**for (int i = 0; i < V - 1; i++) {**

**int u = findKey(visited, distance, V);**

**if (u == -1) break;**

**visited[u] = true;**

**for (int v = 0; v < V; v++) {**

**if (graph[u][v] != 0 && !visited[v] && distance[u] != INF**

**&& distance[v] > distance[u] + graph[u][v]) {**

**distance[v] = distance[u] + graph[u][v];**

**}**

**}**

**}**

**System.out.println("\nShortest distances from node " + src + ":");**

**for (int i = 0; i < V; i++) {**

**if (distance[i] == INF)**

**System.out.println("Node " + i + ": INF");**

**else**

**System.out.println("Node " + i + ": " + distance[i]);**

**}**

**}**

**public static void main(String[] args) {**

**Scanner sc = new Scanner(System.in);**

**System.out.print("Enter number of vertices: ");**

**int V = sc.nextInt();**

**int[][] graph = new int[V][V];**

**System.out.println("Enter adjacency matrix (0 if no edge):");**

**for (int i = 0; i < V; i++) {**

**for (int j = 0; j < V; j++) {**

**graph[i][j] = sc.nextInt();**

**}**

**}**

**System.out.print("Enter source node (0 to " + (V - 1) + "): ");**

**int src = sc.nextInt();**

**System.out.println("\nGraph:");**

**for (int i = 0; i < V; i++) {**

**for (int j = 0; j < V; j++)**

**System.out.print(graph[i][j] + "\t");**

**System.out.println();**

**}**

**dijkstra(graph, src);**

**}**

**}**