**Lab Practical #12:**

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

**Practical Assignment #12:**

1. **C/JAVA Program: Distance Vector Routing Algorithm using Bellman Ford's Algorithm.**
2. #include<stdio.h>
3. struct node
4. {
5. unsigned dist[20];
6. unsigned from[20];
7. }rt[10];
8. int main()
9. {
10. int costmat[20][20];
11. int nodes,i,j,k,count=0;
12. printf("\nEnter the number of nodes : ");
13. scanf("%d",&nodes);//Enter the nodes
14. printf("\nEnter the cost matrix :\n");
15. for(i=0;i<nodes;i++)
16. {
17. for(j=0;j<nodes;j++)
18. {
19. scanf("%d",&costmat[i][j]);
20. costmat[i][i]=0;
21. rt[i].dist[j]=costmat[i][j];//initialise the distance equal to cost matrix
22. rt[i].from[j]=j;
23. }
24. }
25. do
26. {
27. count=0;
28. for(i=0;i<nodes;i++)//We choose arbitary vertex k and we calculate the direct distance from the node i to k using the cost matrix
29. //and add the distance from k to node j
30. for(j=0;j<nodes;j++)
31. for(k=0;k<nodes;k++)
32. if(rt[i].dist[j]>costmat[i][k]+rt[k].dist[j])
33. {//We calculate the minimum distance
34. rt[i].dist[j]=rt[i].dist[k]+rt[k].dist[j];
35. rt[i].from[j]=k;
36. count++;
37. }
38. }while(count!=0);
39. for(i=0;i<nodes;i++)
40. {
41. printf("\n\n For router %d\n",i+1);
42. for(j=0;j<nodes;j++)
43. {
44. printf("\t\nnode %d via %d Distance %d ",j+1,rt[i].from[j]+1,rt[i].dist[j]);
45. }
46. }
47. printf("\n\n");
48. getch();
49. }

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

#include <stdio.h>

#include <string.h>

int main()

{

int count,src\_router,i,j,k,w,v,min;

int cost\_matrix[100][100],dist[100],last[100];

int flag[100];

 printf("\n Enter the no of routers");

scanf("%d",&count);

printf("\n Enter the cost matrix values:");

for(i=0;i<count;i++)

{

for(j=0;j<count;j++)

{

 printf("\n%d->%d:",i,j);

scanf("%d",&cost\_matrix[i][j]);

if(cost\_matrix[i][j]<0)cost\_matrix[i][j]=1000;

}

}

 printf("\n Enter the source router:");

scanf("%d",&src\_router);

for(v=0;v<count;v++)

{

flag[v]=0;

last[v]=src\_router;

dist[v]=cost\_matrix[src\_router][v];

}

flag[src\_router]=1;

for(i=0;i<count;i++)

{

min=1000;

for(w=0;w<count;w++)

{

if(!flag[w])

if(dist[w]<min)

{

v=w;

min=dist[w];

}

}

flag[v]=1;

for(w=0;w<count;w++)

{

if(!flag[w])

if(min+cost\_matrix[v][w]<dist[w])

{

dist[w]=min+cost\_matrix[v][w];

last[w]=v;

}

}

}

for(i=0;i<count;i++)

{

 printf("\n%d==>%d:Path taken:%d",src\_router,i,i);

w=i;

while(w!=src\_router)

{

 printf("\n<--%d",last[w]);w=last[w];

}

 printf("\n Shortest path cost:%d",dist[i]);

}

}