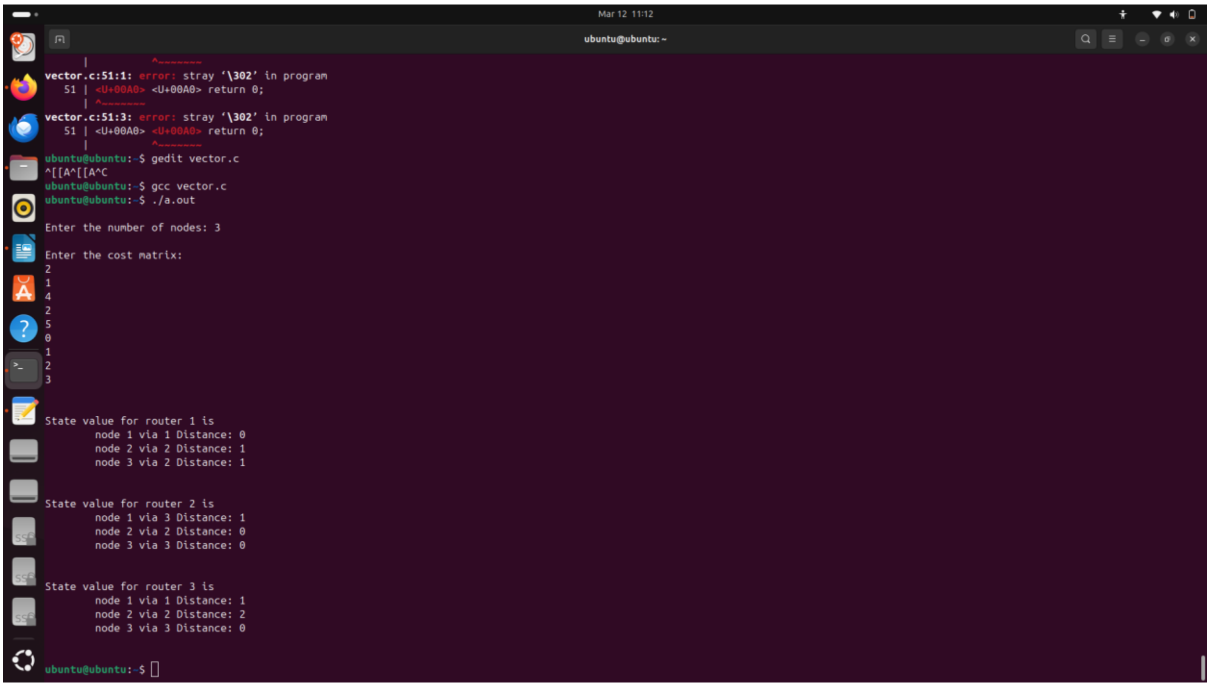
Arun M

76 R6B

Routing protocols Distance vector and Link state routing

```
#include <stdio.h>

struct node {
   unsigned dist[20];
   unsigned from[20];
} rt[10];

int main() {
   int dmat[20][20];
   int n, i, j, k, count = 0;

   printf("\nEnter the number of nodes: ");
   scanf("%d", &n);

   printf("\nEnter the cost matrix:\n");
   for(i = 0; i < n; i++) {
      for(j = 0; j < n; j++) {
         scanf("%d", &dmat[i][j]);
         if (i == j) {
            dmat[i][j] = 0;  // Ensure diagonal elements are zero
         }
         rt[i].dist[j] = dmat[i][j];
         rt[i].from[j] = j;
      }
   }
```

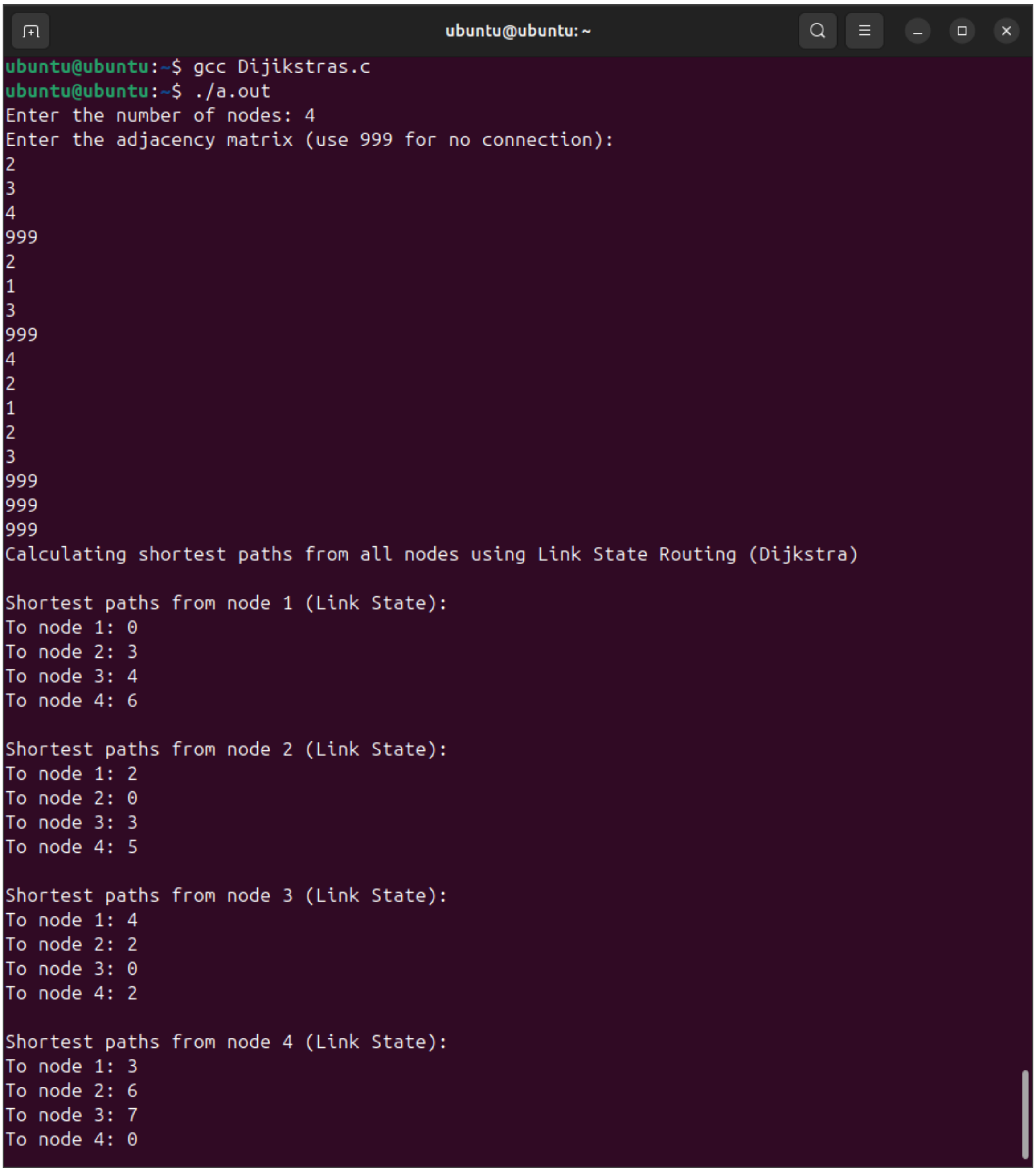
```
    do {
        count = 0;
        for(i = 0; i < n; i++) {
            for(j = 0; j < n; j++) {
                for(k = 0; k < n; k++) {
                    if(rt[i].dist[j] > dmat[i][k] + rt[k].dist[j]) {
                        rt[i].dist[j] = dmat[i][k] + rt[k].dist[j];
                        rt[i].from[j] = k;
                        count++;
                    }
                }
            }
        }
    } while(count != 0);


    for(i = 0; i < n; i++) {
        printf("\n\nState value for router %d is\n", i + 1);
        for(j = 0; j < n; j++) {
            printf("\tnode %d via %d Distance: %d\n", j + 1, rt[i].from[j] + 1, rt[i].dist[j]);
        }
    }
    printf("\n\n");
    return 0;
}
```



```
       |                 ^~~~~~~~
vector.c:51:1: error: stray '\302' in program
   51 | <U+00A0> <U+00A0> return 0;
      | ^~~~~~~~
vector.c:51:3: error: stray '\302' in program
   51 | <U+00A0> <U+00A0> return 0;
      |          ^~~~~~~~
ubuntu@ubuntu:~$ gedit vector.c
^[[A^[[A^C
ubuntu@ubuntu:~$ gcc vector.c
ubuntu@ubuntu:~$ ./a.out

Enter the number of nodes: 3

Enter the cost matrix:
2
1
4
2
5
0
1
2
3



State value for router 1 is
        node 1 via 1 Distance: 0
        node 2 via 2 Distance: 1
        node 3 via 2 Distance: 1


State value for router 2 is
        node 1 via 3 Distance: 1
        node 2 via 2 Distance: 0
        node 3 via 3 Distance: 0


State value for router 3 is
        node 1 via 1 Distance: 1
        node 2 via 2 Distance: 2
        node 3 via 3 Distance: 0


ubuntu@ubuntu:~$ 
```

```
#include <stdio.h>
#include <limits.h>

#define MAX_NODES 10
#define INF INT_MAX

void dijkstra(int graph[MAX_NODES][MAX_NODES], int n, int startNode) {
    int distance[MAX_NODES], visited[MAX_NODES] = {0}, count = 0, minDistance, nextNode;

    for (int i = 0; i < n; i++) {
        distance[i] = graph[startNode][i];
    }
    distance[startNode] = 0;
    visited[startNode] = 1;
    count = 1;

    while (count < n) {
        minDistance = INF;
        for (int i = 0; i < n; i++) {
            if (distance[i] < minDistance && !visited[i]) {
                minDistance = distance[i];
                nextNode = i;
            }
```

```
        }
        visited[nextNode] = 1;
        for (int i = 0; i < n; i++) {
            if (!visited[i] && (minDistance + graph[nextNode][i] < distance[i]) && graph[nextNode][i] !=
INF) {
                distance[i] = minDistance + graph[nextNode][i];
            }
        }
        count++;
    }

    printf("Shortest paths from node %d (Link State):\n", startNode + 1);
    for (int i = 0; i < n; i++) {
        printf("To node %d: %d\n", i + 1, distance[i]);
    }
}

int main() {
    int graph[MAX_NODES][MAX_NODES], n;

    printf("Enter the number of nodes: ");
    scanf("%d", &n);

    printf("Enter the adjacency matrix (use 999 for no connection):\n");
    for (int i = 0; i < n; i++) {
        for (int j = 0; j < n; j++) {
            scanf("%d", &graph[i][j]);
            if (graph[i][j] == 999) graph[i][j] = INF;
        }
    }

    printf("Calculating shortest paths from all nodes using Link State Routing (Dijkstra)\n\n");
    for (int i = 0; i < n; i++) {
        dijkstra(graph, n, i);
        printf("\n");
    }

    return 0;
}
```



```
ubuntu@ubuntu:~$ gcc Dijikstras.c
ubuntu@ubuntu:~$ ./a.out
Enter the number of nodes: 4
Enter the adjacency matrix (use 999 for no connection):
2
3
4
999
2
1
3
999
4
2
1
2
3
999
999
999
Calculating shortest paths from all nodes using Link State Routing (Dijkstra)

Shortest paths from node 1 (Link State):
To node 1: 0
To node 2: 3
To node 3: 4
To node 4: 6

Shortest paths from node 2 (Link State):
To node 1: 2
To node 2: 0
To node 3: 3
To node 4: 5

Shortest paths from node 3 (Link State):
To node 1: 4
To node 2: 2
To node 3: 0
To node 4: 2

Shortest paths from node 4 (Link State):
To node 1: 3
To node 2: 6
To node 3: 7
To node 4: 0
```