CN Lab 8

Anirudh Sathish CS20B1125

1. Given the Routing Table with "n" nodes, Work on "Distance Vector Routing Algorithm", where

Inputs: No of nodes, edges, and weights (distance)

Output: the calculated value of distance (minimum).

```c
// @ Author : Anirudh Sathish

// @ Roll_No : CS20B1125



#include<stdio.h>

#include<stdlib.h>

#include<string.h>

#include<stdbool.h>



#define N 20

#define inf 10000



struct distanceVector

{

    int destination;

    int distance;

    int nexthop;
```

```
};


struct distanceVectorTable

{

    struct distanceVector TableEntry[N];

};


struct Neighbor

{

    bool areNeighbours[N];


};



int main()

{

    int routerNumber;

    printf("Enter no of router \n");

    scanf("%d",&routerNumber);

    int adjacencyMatrix[routerNumber][routerNumber];


    int i ,j ;

    printf("Enter the adjacency matrix values \n");

    printf("Enter 10000 to indicate no edge between the matrices\n");
```

```
    printf("Take required lengths only below 10000 \n");

    printf("10000 is used to represent infinity \n");

    printf("\n\n Enter adjacency Matrix values \n\n");

    for(i = 0; i < 4 ; i++)

    {

        for(j = 0 ; j < 4; j++)

        {

            scanf("%d",&adjacencyMatrix[i][j]);

        }

    }

    printf("The adjacency Matrix is \n");

    for(i = 0; i < 4 ; i++)

    {

        for(j = 0 ; j < 4; j++)

        {

            printf("%d\t",adjacencyMatrix[i][j]);

        }

        printf("\n");

    }



    struct Neighbor neighbour_ver[4];

    for(i = 0 ; i < 4 ; i++)

    {
```

```
        for(j = 0 ; j < 4 ; j++)
        {
            neighbour_ver[i].areNeighbours[j] = false;
        }
    }

    // Establishing distance vector routing for each of the routers
    struct distanceVectorTable table[4];

    for(i =0 ; i < 4 ; i++)
    {
        for(j = 0 ; j < 4 ; j++)
        {
            table[i].TableEntry[j].destination = j;
            table[i].TableEntry[j].distance = adjacencyMatrix[i][j];
            if((table[i].TableEntry[j].distance > 0) &&
(table[i].TableEntry[j].distance < inf-1))
            {
                neighbour_ver[i].areNeighbours[j] = true;
            }
            table[i].TableEntry[j].nexthop = j;
        }
    }
```

```
    printf("Printing the tables \n");

    printf("NOTE : %d stands for infinity \n",inf);

    for(i =0 ; i < 4 ; i++)

    {

        printf("\n\nTable for Router %d \n",i+1);

        printf(" Destination\tDistance\tNext Hop \n");

        for(j = 0 ; j < 4 ; j++)

        {

            printf(" Router %d\t%d\t\tRouter
%d\n",table[i].TableEntry[j].destination+1,table[i].TableEntry[j].distance,table[i]
.TableEntry[j].nexthop+1);

        }

    }



    // Calculate the value got after n-2 iterations

    int n  = 4;

    int remainingIterationCount = n-2;

    int current_cost , t , l , m  ;

    int n_value;



    while(remainingIterationCount != 0 )

    {
```

```
        printf("\n\n-----------------------------------------------\n\n");

        for(i =0 ; i < 4 ; i++)

        {

            for(j = 0 ; j < 4 ; j++)

            {

                // No need to change the case from router to itself

                if(table[i].TableEntry[j].distance != 0)

                {

                    current_cost = table[i].TableEntry[j].distance;

                    for(t = 0 ; t < 4 ; t++)

                    {

                        // For vertices without a particular neighbour , it is not
necceary to consider

                        if(neighbour_ver[i].areNeighbours[t] == true &&
neighbour_ver[t].areNeighbours[j] )

                        {

                            n_value = table[i].TableEntry[t].distance +
table[t].TableEntry[j].distance ;

                            if(current_cost > n_value)

                            {

                                table[i].TableEntry[j].distance = n_value;

                                table[i].TableEntry[j].nexthop = t;

                            }

                        }

                    }
```

```
                }

            }

        }

        remainingIterationCount--;

        printf("Iterations Left %d \n",remainingIterationCount);

        printf("Printing the tables \n");

        printf("NOTE : %d stands for infinity \n",inf);

        for(l =0 ; l < 4 ; l++)

        {

            printf("\n\nTable for Router %d \n",l+1);

            printf(" Destination\tDistance\tNext Hop \n");

            for(m = 0 ; m < 4 ; m++)

            {

                printf(" Router %d\t%d\t\tRouter
%d\n",table[l].TableEntry[m].destination+1,table[l].TableEntry[m].distance,table[l]
.TableEntry[m].nexthop+1);

            }

        }

    }




}
```

Output

```
Enter no of router
4
Enter the adjacency matrix values
Enter 10000 to indicate no edge between the matrices
Take required lengths only below 10000
10000 is used to represent infinity


 Enter adjacency Matrix values

0
2
10000
1
2
0
3
7
10000
3
0
11
1
7
11
0
```

```
The adjacency Matrix is
0       2       10000   1
2       0       3       7
10000   3       0       11
1       7       11      0
Printing the tables
NOTE : 10000 stands for infinity


Table for Router 1
 Destination    Distance        Next Hop
 Router 1       0               Router 1
 Router 2       2               Router 2
 Router 3       10000           Router 3
 Router 4       1               Router 4


Table for Router 2
 Destination    Distance        Next Hop
 Router 1       2               Router 1
 Router 2       0               Router 2
 Router 3       3               Router 3
 Router 4       7               Router 4
```

```
Table for Router 3
 Destination    Distance        Next Hop
 Router 1       10000           Router 1
 Router 2       3               Router 2
 Router 3       0               Router 3
 Router 4       11              Router 4


Table for Router 4
 Destination    Distance        Next Hop
 Router 1       1               Router 1
 Router 2       7               Router 2
 Router 3       11              Router 3
 Router 4       0               Router 4


 ------------------------------------------
```

```
Iterations Left 1
Printing the tables
NOTE : 10000 stands for infinity


Table for Router 1
 Destination    Distance        Next Hop
 Router 1       0               Router 1
 Router 2       2               Router 2
 Router 3       12              Router 4
 Router 4       1               Router 4


Table for Router 2
 Destination    Distance        Next Hop
 Router 1       2               Router 1
 Router 2       0               Router 2
 Router 3       3               Router 3
 Router 4       3               Router 1


Table for Router 3
 Destination    Distance        Next Hop
 Router 1       12              Router 4
 Router 2       3               Router 2
 Router 3       0               Router 3
 Router 4       6               Router 2
```

```
Table for Router 4
 Destination    Distance        Next Hop
 Router 1       1               Router 1
 Router 2       3               Router 1
 Router 3       6               Router 2
 Router 4       0               Router 4


-----------------------------------------------

Iterations Left 0
Printing the tables
NOTE : 10000 stands for infinity


Table for Router 1
 Destination    Distance        Next Hop
 Router 1       0               Router 1
 Router 2       2               Router 2
 Router 3       7               Router 4
 Router 4       1               Router 4
```

```
Table for Router 2
 Destination    Distance        Next Hop
 Router 1       2               Router 1
 Router 2       0               Router 2
 Router 3       3               Router 3
 Router 4       3               Router 1


Table for Router 3
 Destination    Distance        Next Hop
 Router 1       7               Router 4
 Router 2       3               Router 2
 Router 3       0               Router 3
 Router 4       6               Router 2


Table for Router 4
 Destination    Distance        Next Hop
 Router 1       1               Router 1
 Router 2       3               Router 1
 Router 3       6               Router 2
 Router 4       0               Router 4
```