RAVI SAJJANAR

1BM19CS127

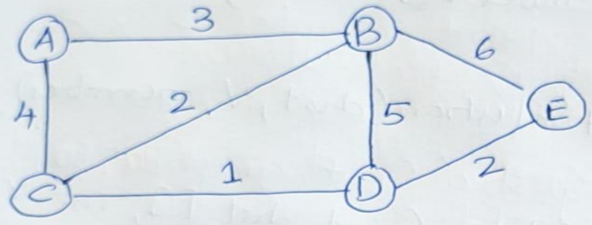
7) From a given vertex in a weighted connected graph, find shortest paths to other vertices using Dijkstra's algorithm.

Modification:
Print number of nodes along the shortest paths.

```
#include <stdio.h>
int minDistance(int dist[], int sptSet[],int V)
{
    int min = 999, min_index;
  int v;
    for ( v = 0; v < V; v++)
        if (sptSet[v] == 0 && dist[v] <= min)
            min = dist[v], min_index = v;

    return min_index;
}



int printSolution(int src,int dist[],int V )
{ int i;
    printf("\n\t Vertex \t\t Distance from Source\n\n");
    for (i = 0; i < V; i++)
        printf("\t%c ----> %c \t\t\t %d\n",src+65, i+65, dist[i]);
}



void dijkstra(int graph[10][10], int src,int V )
{
    int dist[V];
  int i,count,u,v;
    int sptSet[V];
    for ( i = 0; i < V; i++)
        dist[i] = 999, sptSet[i] = 0;

    dist[src] = 0;

    for ( count = 0; count < V - 1; count++) {

        u = minDistance(dist, sptSet,V);
        sptSet[u] = 1;

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

            if (!sptSet[v] && graph[u][v] && dist[u] != 999
                && dist[u] + graph[u][v] < dist[v])
                dist[v] = dist[u] + graph[u][v];
    }

    printSolution(src,dist,V);
}
```

```
int main()
{
    int i,j,V;
    int graph[10][10];
    printf("Enter number of vertices\n");
    scanf("%d",&V);
    printf("Enter adjacency matrix\n");
    for(i=0;i<V;i++)
    {
        for(j=0;j<V;j++)
    scanf("%d",&graph[i][j]);
    }
       dijkstra(graph, 0,V );

    return 0;
}

/*
999 3   4   999 999
3   999 2   5   6
4   2   999 1   999
999 5   1   999 2
999 6   999 2   999



*/
```

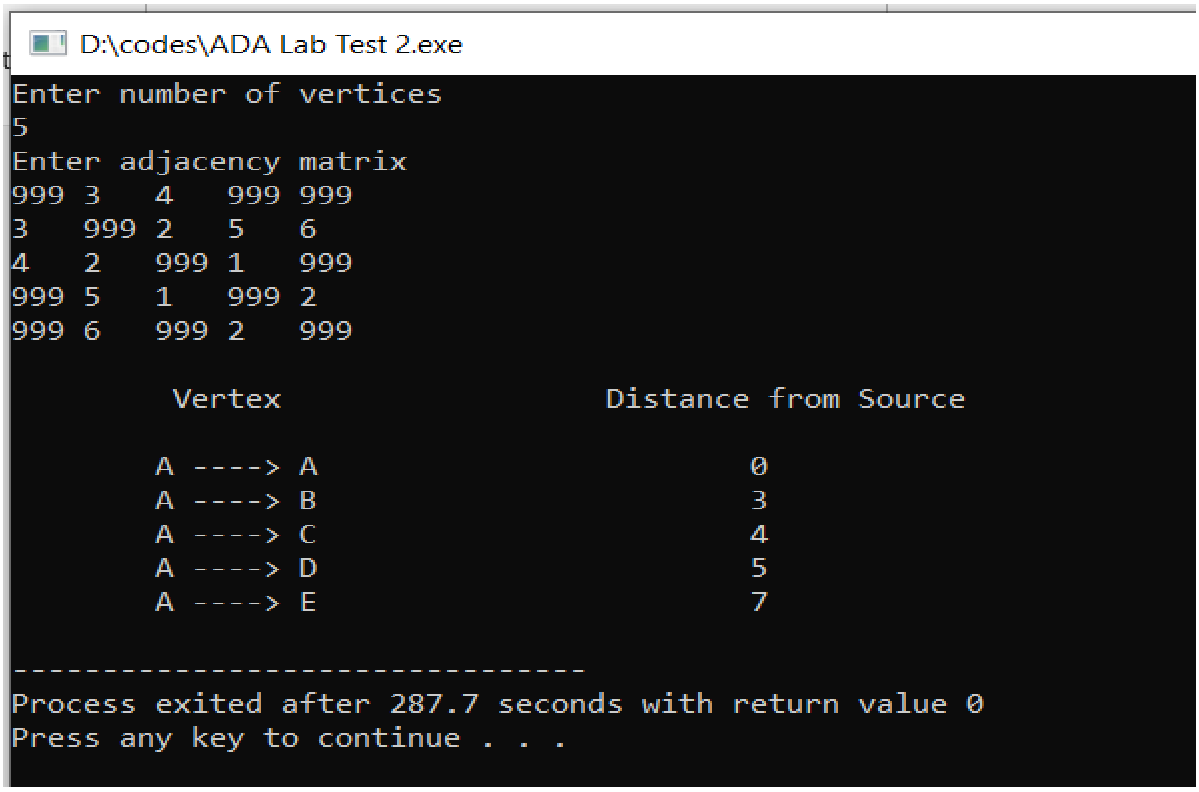
Graph:



⇓

Adjacency Matrix

$$\begin{array}{cccccc} & A & B & C & D & E \\ A & \infty & 3 & 4 & \infty & \infty \\ B & 3 & \infty & 2 & 5 & 6 \\ C & 4 & 2 & \infty & 1 & \infty \\ D & \infty & 5 & 1 & \infty & 2 \\ E & \infty & 6 & \infty & 2 & \infty \end{array}$$



```
Enter number of vertices
5
Enter adjacency matrix
999 3   4   999 999
3   999 2   5   6
4   2   999 1   999
999 5   1   999 2
999 6   999 2   999

          Vertex                       Distance from Source

         A ----> A                              0
         A ----> B                              3
         A ----> C                              4
         A ----> D                              5
         A ----> E                              7

--------------------------------
Process exited after 287.7 seconds with return value 0
Press any key to continue . . .
```

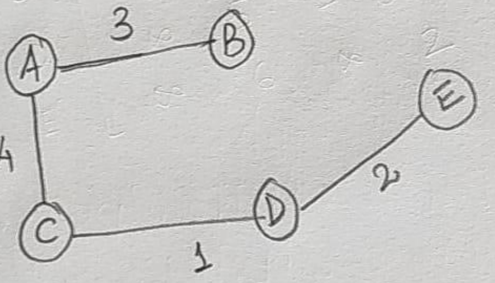
IBM19CS127

Dijkstra's Algorithm:

taking Source = A

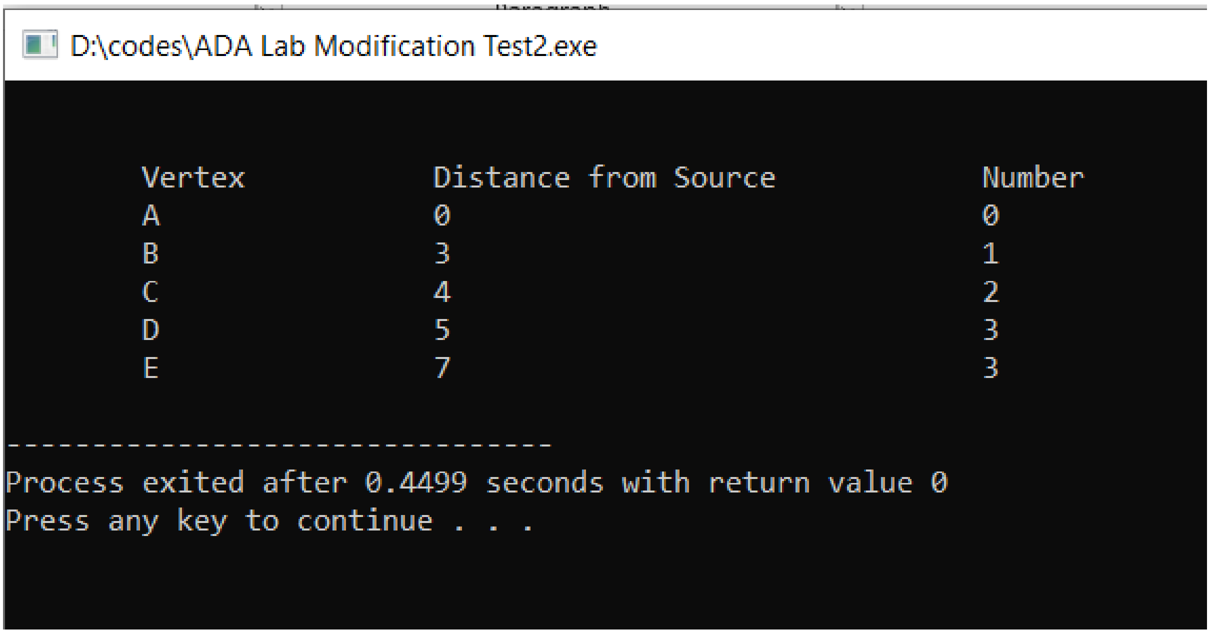
Source A	Destination B	 C	 D	 E
(0)	∞	∞	∞	∞
A	3	4	∞	∞
A, B	3	4	8	9
A, B, C	3	4	5	9
A, B, C, D	3	4	5	7
A, B, C, D, E	3	4	5	7

← optimal soln:



Modification:
Print number of nodes along the shortest paths.

```
#include <stdio.h>

int minDistance(int dist[], int sptSet[], int V)
{
    int min = 999, min_index;
    int v;
    for (v = 0; v < V; v++)
        if (sptSet[v] == 0 && dist[v] <= min)
            min = dist[v], min_index = v;

    return min_index;
}
int printSolution(int dist[], int V, int number[])
{
          printf("\n\n");
    int i;
    printf("\tVertex \t\t Distance from Source \t\t Number\n");
    for (i = 0; i < V; i++)
        printf("\t%c \t\t %d \t\t\t\t %d\n", i+65, dist[i],number[i]);
}
void dijkstra(int graph[5][5], int src, int V)
{
    int dist[V];
    int i, count, u, v;
    int sptSet[V];
    for (i = 0; i < V; i++)
        dist[i] = 999, sptSet[i] = 0;


    dist[src] = 0;
    int number[V];
    for (count = 0; count < V - 1; count++)
    {


        u = minDistance(dist, sptSet, V);
        number[u]=count;


        sptSet[u] = 1;


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


            if (!sptSet[v] && graph[u][v] && dist[u] != 999 && dist[u] + graph[u][v] < dist[v]){


                dist[v] = dist[u] + graph[u][v];
            }
    }


    for(i=0;i<V;i++){
    if(sptSet[i]==0){
        int max = 0;
        for(int j = 0; j<V;j++){
            if(number[j]>max && sptSet[j]!=0)
                max = number[j];
        }
        number[i] = max;
    }


    }
    printSolution(dist, V, number);
}

int main()
{


          int graph[5][5]{
          {999,3,  4,  999,999},
          {3,  999,2,  5,  6},
          {4,  2,  999,1,  999},
          {999,5,  1,  999,2},
          {999,6,  999,2,  999}
          };

          dijkstra(graph, 0, 5);

    return 0;
}
```



```
        Vertex           Distance from Source             Number
        A                0                                0
        B                3                                1
        C                4                                2
        D                5                                3
        E                7                                3

--------------------------------
Process exited after 0.4499 seconds with return value 0
Press any key to continue . . .
```