Shortest Path Algorithm Djikstra's Method

Code Id 20

Date Updated 3/7/2010

Title Shortest path algorithm Djikstra s methd

Description

This is a program of shortest path between two node in graph using Djikstra algor

Codes Snippet

```
#include
#define MAX 10
#define TEMP 0
#define PERM 1
#define infinity 9999

struct node
{
        int predecessor;
        int dist; /*minimum distance of node from source*/
        int status;
};

int adj[MAX][MAX];
int n;
void main()
{
        int i,j;
        int source,dest;
        int path[MAX];
        int shortdist,count;

        create_graph();
        printf("The adjacency matrix is :n");
        display();

        while(1)
        {
                printf("Enter source node(0 to quit) : ");
                scanf("%d",&source);
                printf("Enter destination node(0 to quit) : ");
                scanf("%d",&dest);

                if(source==0 || dest==0)
                        exit(1);

                count = findpath(source,dest,path,&shortdist);
                if(shortdist!=0)
                {
                        printf("Shortest distance is : %dn", shortdist);
                        printf("Shortest Path is : ");
                        for(i=count;i>1;i--)
                                printf("%d->",path[i]);
                        printf("%d",path[i]);
                        printf("n");
                }
                else
                        printf("There is no path from source to destination noden
        }/*End of while*/
}/*End of main()*/

create_graph()
{
        int i,max_edges,origin,destin,wt;

        printf("Enter number of vertices : ");
        scanf("%d",&n);
        max_edges=n*(n-1);

        for(i=1;i<=max_edges;i++)
```


Recent Posts

Types of Cloud Computing
What is Cloud Computing ?
How to pass a multi-dimensional array to a function?
Memory Layout of a C Program
PHP and Its Advantages

```
                }
                else
                        adj[origin][destin]=wt;
        }/*End of for*/
}/*End of create_graph()*/

display()
{
        int i,j;
        for(i=1;i<=n;i++)
        {
                for(j=1;j<=n;j++)
                        printf("%3d",adj[i][j]);
                printf("n");
        }

}/*End of display()*/

int findpath(int s,int d,int path[MAX],int *sdist)
{
        struct node state[MAX];
        int i,min,count=0,current,newdist,u,v;
        *sdist=0;
        /* Make all nodes temporary */
        for(i=1;i<=n;i++)
        {
                state[i].predecessor=0;
                state[i].dist = infinity;
                state[i].status = TEMP;
        }

        /*Source node should be permanent*/
        state[s].predecessor=0;
        state[s].dist = 0;
        state[s].status = PERM;

        /*Starting from source node until destination is found*/
        current=s;
        while(current!=d)
        {
                for(i=1;i<=n;i++)
                {
                        /*Checks for adjacent temporary nodes */
                        if ( adj[current][i] > 0 && state[i].status == TEMP )
                        {
                                newdist=state[current].dist + adj[current][i];
                                /*Checks for Relabeling*/
                                if( newdist < state[i].dist )
                                {
                                        state[i].predecessor = current;
                                        state[i].dist = newdist;
                                }
                        }
                }/*End of for*/

                /*Search for temporary node with minimum distand make it current
                min=infinity;
                current=0;
                for(i=1;i<=n;i++)
                {
                        if(state[i].status == TEMP && state[i].dist < min)
                        {
                                min = state[i].dist;
                                current=i;
                        }
                }/*End of for*/

                if(current==0) /*If Source or Sink node is isolated*/
                        return 0;
                state[current].status=PERM;
        }/*End of while*/

        /* Getting full path in array from destination to source        */
        while( current!=0 )
        {
                count++;
                path[count]=current;
                current=state[current].predecessor;
        }
```