**Dijkstra's Shortest Path Routing Algorithm**

#include<stdio.h>

#include<string.h>

#include<math.h>

#define IN 99

#define N 6

int dijkstra(int cost[][N], int source, int target);

char \*strrev(char\*);

int main()

{

int cost[N][N], i, j, w, ch, co;

int x, y, source, target;

int s, t;

for(i = 1; i < N; i++)

for(j = 1; j < N; j++)

cost[i][j] = IN;

for(x = 1; x < N; x++)

{

for(y = x+1; y < N; y++)

{

printf("Enter weight between %d and %d: ", x, y);

scanf("%d", &w);

if(w==0) w = IN;

cost[x][y] = cost [y][x] = w;

}

printf("\n");

}

printf("\nEnter the source:");

scanf("%d", &source);

printf("Enter the target:");

scanf("%d", &target);

co = dijkstra(cost, source, target);

printf("\nDISTANCE: %d\n", co);

}

int dijkstra(int cost[][N], int source, int target)

{

int dist[N], prev[N], selected[N] = {0}, i, m, min, start, d, j;

char path[N];

for(i = 1; i < N; i++)

{

dist[i] = IN;

prev[i] = -1;

}

start = source;

selected[start] = 1;

dist[start] = 0;

while(selected[target]==0)

{

min = IN;

m = 0;

for(i = 1; i < N; i++)

{

d = dist[start] + cost[start][i];

if(d< dist[i]&&selected[i]==0)

{

dist[i] = d;

prev[i] = start;

}

if(min>dist[i] && selected[i]==0)

{

min = dist[i];

m = i;

}

}

start = m;

selected[start] = 1;

}

start = target;

j = 0;

while(start != -1)

{

path[j++] = start+64;

start = prev[start];

}

path[j]='\0';

strrev(path);

printf("\nSHORTEST PATH: %s", path);

return dist[target];

}

char \*strrev(char \*str)

{

char \*p1, \*p2;

if (! str || ! \*str)

return str;

for (p1 = str, p2 = str + strlen(str) - 1; p2 > p1; ++p1, --p2)

{

\*p1 ^= \*p2;

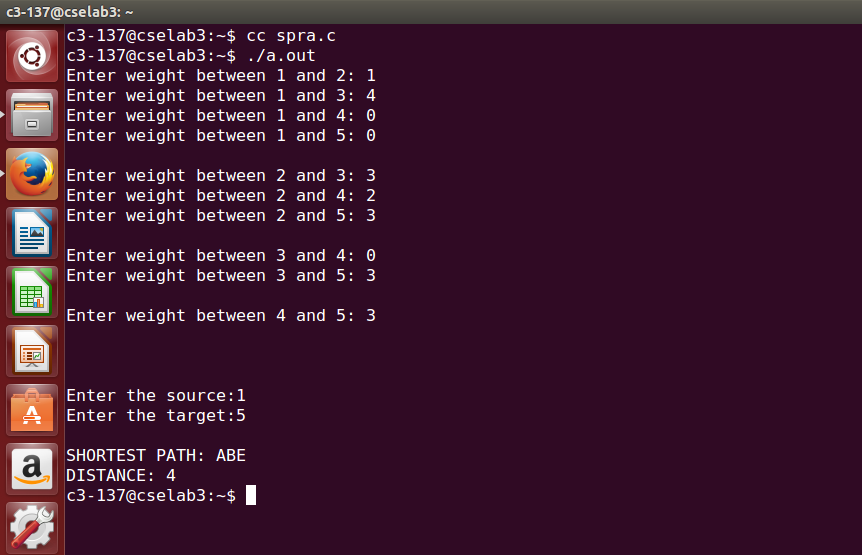
\*p2 ^= \*p1;

\*p1 ^= \*p2;

}

return str;

}



**Bellman-Ford Distance Vector Routing Algorithm**

#include<stdio.h>

struct node

{

unsigned dist[20];

unsigned from[20];

}rt[10];

int main()

{

int costmat[20][20];

int nodes,i,j,k,count=0;

printf("\nEnter the number of nodes : ");

scanf("%d",&nodes);

printf("\nEnter the cost matrix :\n");

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

{

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

{

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

costmat[i][i]=0;

rt[i].dist[j]=costmat[i][j];

rt[i].from[j]=j;

}

}

do

{

count=0;

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

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

for(k=0;k<nodes;k++)

if(rt[i].dist[j]>costmat[i][k]+rt[k].dist[j])

{

rt[i].dist[j]=rt[i].dist[k]+rt[k].dist[j];

rt[i].from[j]=k;

count++;

}

}while(count!=0);

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

{

printf("\n\n ROUTER %d\n",i+1);

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

printf("\n| To | Via | Distance |");

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

{

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

printf("\t\n| %d | %d | %d\t |",j+1,rt[i].from[j]+1,rt[i].dist[j]);

}

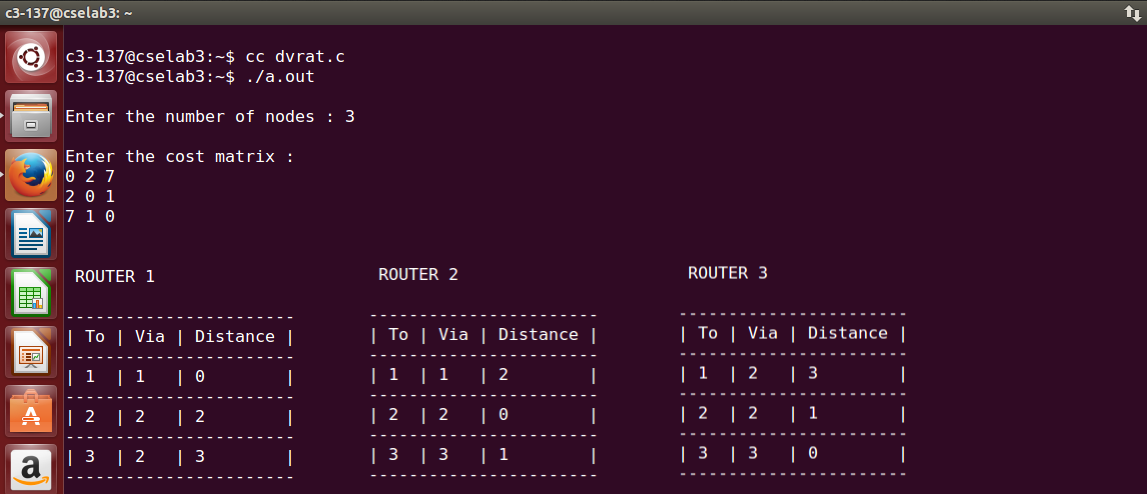
printf("\n-----------------------");

}

printf("\n\n");

return 0;

}

****