Week 6

Dijkstra

#include <stdio.h>

#include <limits.h>

#define V 100

int minDistance(int dist[], int sptSet[], int n) {

int min = INT\_MAX, min\_index;

for (int v = 0; v < n; v++)

if (sptSet[v] == 0 && dist[v] <= min)

min = dist[v], min\_index = v;

return min\_index;

}

void dijkstra(int graph[V][V], int src, int n) {

int dist[V];

int sptSet[V];

for (int i = 0; i < n; i++)

dist[i] = INT\_MAX, sptSet[i] = 0;

dist[src] = 0;

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

int u = minDistance(dist, sptSet, n);

sptSet[u] = 1;

for (int v = 0; v < n; v++)

if (!sptSet[v] && graph[u][v] && dist[u] != INT\_MAX

&& dist[u] + graph[u][v] < dist[v])

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

}

printf("Vertex \t Shortest Distance from Source %d\n", src);

for (int i = 0; i < n; i++)

printf("%d \t\t %d\n", i, dist[i]);

}

int main() {

int graph[V][V];

int n, src;

printf("Enter number of vertices: ");

scanf("%d", &n);

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

for (int i = 0; i < n; i++)

for (int j = 0; j < n; j++)

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

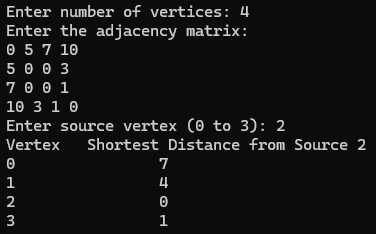
printf("Enter source vertex (0 to %d): ", n - 1);

scanf("%d", &src);

dijkstra(graph, src, n);

return 0;

}



Johnson Trotter

#include <stdio.h>

#include <stdlib.h>

#include <stdbool.h>

#define LEFT -1

#define RIGHT 1

int getMobile(int perm[], int dir[], int n) {

int mobile = 0;

for (int i = 0; i < n; i++) {

int nextIndex = i + dir[i];

if (nextIndex >= 0 && nextIndex < n) {

if (perm[i] > perm[nextIndex] && perm[i] > mobile) {

mobile = perm[i];

}

}

}

return mobile;

}

int getPos(int perm[], int n, int mobile) {

for (int i = 0; i < n; i++) {

if (perm[i] == mobile)

return i;

}

return -1;

}

void printPerm(int perm[], int n) {

for (int i = 0; i < n; i++) {

printf("%d ", perm[i]);

}

printf("\n");

}

void generatePermutations(int n) {

int\* perm = (int\*)malloc(n \* sizeof(int));

int\* dir = (int\*)malloc(n \* sizeof(int));

for (int i = 0; i < n; i++) {

perm[i] = i + 1;

dir[i] = LEFT;

}

printPerm(perm, n);

while (1) {

int mobile = getMobile(perm, dir, n);

if (mobile == 0)

break;

int pos = getPos(perm, n, mobile);

int nextPos = pos + dir[pos];

int temp = perm[pos];

perm[pos] = perm[nextPos];

perm[nextPos] = temp;

int tempDir = dir[pos];

dir[pos] = dir[nextPos];

dir[nextPos] = tempDir;

for(int i=0;i<n;i++){

if(perm[i]>mobile){

dir[i]=-dir[i];

}

}

printPerm(perm,n);

}

free(perm);

free(dir);

}

int main() {

int n;

printf("Enter the number of elements: ");

scanf("%d",&n);

generatePermutations(n);

return 0;

}

