Sebastián Almanza Galvis

Estructura de datos

Jhon Corredor

30/10/24

Lab Grafos

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

Explicación código:

**Función miniDist**

“int miniDist(int distance[], bool Tset[]) {

int minimum = INT\_MAX, ind;

for (int k = 0; k < 7; k++) {

if (Tset[k] == false && distance[k] <= minimum) {

minimum = distance[k];

ind = k;

}

}

return ind;

}”

Propósito: Encuentra el vértice con la distancia mínima que aún no ha sido procesado.

Parámetros:

distance[]: Un array que guarda la distancia mínima desde el nodo fuente a cada nodo.

Tset[]: Un array de booleanos que indica si un nodo ya ha sido incluido en el conjunto de nodos procesados.

Funcionamiento:

* Recorre todos los nodos. Si el nodo no ha sido procesado (Tset[k] == false) y su distancia es menor o igual que minimum, actualiza minimum y el índice ind.
* Retorna el índice del nodo con la menor distancia no procesada.

**Función DijkstraAlgo**

“void DijkstraAlgo(int graph[7][7], int src) {

int distance[7];

bool Tset[7];

for (int k = 0; k < 7; k++) {

distance[k] = INT\_MAX;

Tset[k] = false;

}

distance[src] = 0;

for (int k = 0; k < 7; k++) {

int m = miniDist(distance, Tset);

Tset[m] = true;

for (int k = 0; k < 7; k++) {

if (!Tset[k] && graph[m][k] && distance[m] != INT\_MAX && distance[m] + graph[m][k] < distance[k])

distance[k] = distance[m] + graph[m][k];

}

}

cout << "Vertice \t\t Distancia desde la fuente al Vertice" << endl;

for (int k = 0; k < 7; k++) {

char str = 65 + k;

cout << str << "\t\t\t" << distance[k] << endl;

}

}”

Propósito: Implementa el Algoritmo de Dijkstra para encontrar la distancia mínima desde el nodo fuente (src) a cada otro nodo.

Parámetros:

graph[7][7]: La matriz de adyacencia del grafo, donde graph[i][j] es el peso del arco entre los nodos i y j.

src: El nodo fuente desde el cual se calculan las distancias.

Funcionamiento:

1. Inicializa distance[] a INT\_MAX para indicar que todas las distancias iniciales son infinitas.

2. Inicializa Tset[] a false para indicar que ningún nodo ha sido procesado.

3. Establece la distancia de src a sí mismo como 0.

4. Para cada iteración:

Llama a miniDist para encontrar el nodo no procesado más cercano (m).

Marca m como procesado.

Para cada nodo adyacente k de m, si se encuentra una ruta más corta hacia k a través de m, actualiza distance[k].

**Imprime la distancia mínima desde el nodo fuente a cada nodo.**

Función main

“int main() {

int graph[7][7] = {

{0, 2, 4, 0, 0, 0, 0},

{2, 0, 0, 5, 0, 0, 0},

{4, 0, 0, 8, 0, 0, 0},

{0, 5, 8, 0, 10, 15, 0},

{0, 0, 0, 10, 0, 6, 2},

{0, 0, 0, 15, 6, 0, 6},

{0, 0, 0, 0, 2, 6, 0}

};

DijkstraAlgo(graph, 0);

return 0;

}”

**Descripción:**

* Define la matriz de adyacencia graph[7][7] con pesos en las conexiones. Un valor de 0 indica que no hay conexión directa entre esos nodos.
* Llama a DijksatraAlgo con el nodo fuente A calculando la distancia mínima de un nodo al otro.