ALGORITMOS Y ESTRUCTURAS DE DATOS 2023/1

PROFESOR: Manuel Alejandro Moscoso Domínguez
manuel.moscoso.d@gmail.com

Laboratorio Semana 12

En esta oportunidad realizaremos actividades de programación en C + + y el trabajo con grafos.

Objetivos

- Resolver ejercicios que involucren la implementación de funciones o clases en C + +.
- Desarrollar algoritmos de recorridos asociados a grafos en C++.
- Desarrollar algoritmos que permitan entregar una solución a los problemas entregados.

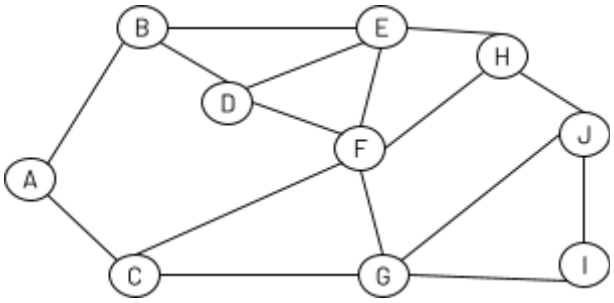
Ejercicios

Nomenclatura para nombre de archivos fuentes

El nombre del archivo en el cual se almacena el código fuente debe considerar el siguiente formato: labsemanaX.EXT donde; X corresponde a la semana y EXT a la extensión del lenguaje de programación utilizado.

Ir a la ayuda memoría

Considerando el siguiente grafo:



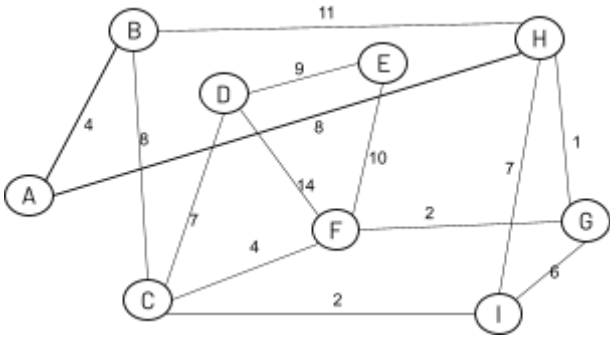
Ejercicio número 1

Crear una estructura que permita relacionar el vértice con el índice dentro de una matriz de adyacencia.

Ejercicio número 2

Crear una función que permita determinar si existe un camino entre dos vértices. La salida de la ejecución debe ser el orden de los vértices en el camino.

Considerando el siguiente grafo:



Ejercicio número 3

Crear una función que implemente el algoritmo de Dijkstra que permita determinar el camino más corto desde un origen a todos los vértices de un grafo. La salida debe ser la distancia del origen a todos los vértices.

Ejercicio número 4

Crear una función que implemente el algoritmo de Floyd-Warshall que permita determinar el camino más corto desde todos los par de vértices de un grafo. La salida debe ser la matriz de distancias.

Ayuda memoria

```
#include <iostream>
#include <fstream>
#include <sstream>
#include <vector>
#include <queue>
#include <limits>
#include <unordered_map>
#include <string>

using namespace std;

const int INF = numeric_limits<int>::max();

int main() {

    int V = 0;

    unordered_map<string, int> myMapStringInt;
    myMapStringInt["string1"] = 1;
    myMapStringInt["string2"] = 2;
    myMapStringInt["string3"] = 3;

    // Range-based for loop
    for (const auto& pair : myMapStringInt) {
        const string& key = pair.first;
        int value = pair.second;
        cout << "Key: " << key << ", Value: " << value << endl;
    }

    // Iterator-based loop
    for (auto it = myMapStringInt.begin(); it != myMapStringInt.end(); ++it) {
        const string& key = it->first;
        int value = it->second;
        cout << "Key: " << key << ", Value: " << value << endl;
    }

    unordered_map<int, string> myMapIntString;
    myMapIntString[1] = "String 1";
```

```
    myMapIntString[2] = "String 2";
    myMapIntString[3] = "String 3";
        // Range-based for loop
    for (const auto& pair : myMapIntString) {
        int key = pair.first;
        const string& value = pair.second;
        cout << "Key: " << key << ", Value: " << value << endl;
    }

    vector<vector<int>> graph = {
        {0, 4, 0, 0, 0, 0, 0},
        {4, 0, 8, 0, 0, 0, 0},
        {0, 8, 0, 7, 0, 4, 0},
        {0, 0, 7, 0, 9, 14, 0},
        {0, 0, 0, 9, 0, 10, 0},
        {0, 0, 4, 14, 10, 0, 2},
        {0, 0, 0, 0, 0, 2, 0}
    };

    V = graph.size();
    for (int i = 0; i < V; ++i) {
        cout << "Vertice " << i << ":";
        for (int j = 0; j < V; ++j) {
            cout << graph[i][j] << " ";
        }
        cout << endl;
    }

    //Directed Graph
    cout << endl << "Directed Graph" << endl;
    vector<vector<int>> graphDirected  = {
        {0, 2, INF, 1, INF},
        {INF, 0, 4, INF, 5},
        {INF, INF, 0, INF, INF},
        {INF, INF, 2, 0, INF},
        {INF, INF, INF, 3, 0}
    };

    V = graphDirected.size();
    for (int i = 0; i < V; ++i) {
        cout << "Vertice " << i << ":";
        for (int j = 0; j < V; ++j) {
            cout << graph[i][j] << " ";
        }
        cout << endl;
    }

    string filename = "data.csv";
    ifstream file(filename);
    if (file.is_open()) {
        string line;
        getline(file, line); // Skip the header line

        while (getline(file, line)) {
            stringstream ss(line);
            string data1, data2;
```

```
            getline(ss, data1, ',');
            getline(ss, data2, ',');

            cout << "Data1: " << data1 << " / Data2: " << data2 << endl;
        }

        file.close();
    }
    else {
        cout << "Failed to open file: " << filename << endl;
    }

    return 0;

}
```