ALGORITMOS Y ESTRUCTURAS DE DATOS 2023/1

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

Laboratorio Semana 11

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 asociados a grafos en C++.
- Desarrollar algoritmos que permitan entregar una solución a los problemas entregados.

Ejercicios

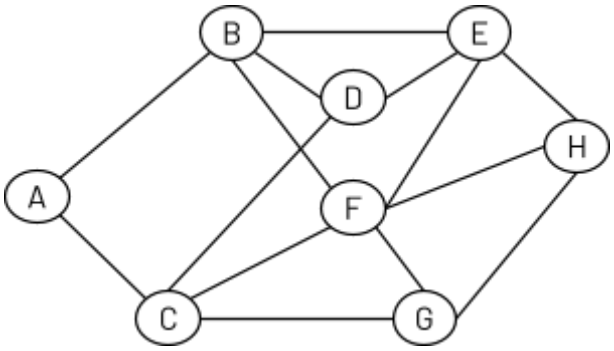
Nomenclatura para nombre de archivos fuentes

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

Grafos

Crear una estructura que permita trabajar con este tipo de grafos a través de una matriz de adyacencia.



Ejercicio número 1

Crear una estructura o clase que permita trabajar con este tipo de grafo y crear las instrucciones que permitan incorporar la información al grafo en el orden señalado en la imagen de referencia.

Ejercicio número 2

Crear una función que permita conocer la matriz del grafo.

Ejercicio número 3

Crear una función que permita conocer si dos vértices son adyacentes. Recordar que dos vértices son adyacentes cuando se encuentran conectados por una arista.

Ejercicio número 4

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

Ejercicio número 5

Crear las instrucciones que permitan incorporar la información al **grafo transformándolo en un grafo dirigido** que:

- Que contenga un vértice que no puede ser alcanzado desde otro vértice.

Ejercicio número 6

Crear las instrucciones que permitan incorporar la información al **grafo transformándolo en un grafo dirigido** que:

- Que contenga un ciclo.

Ejercicio número 7

Crear las instrucciones que permitan incorporar la información al **grafo transformándolo en un grafo dirigido** que:

- Mediante la eliminación de uno o más vértices se transforme en dos componentes (sub grafoes) con un mismo número de vértices.

Ejercicios adicionales:

1. Cree una estructura para trabajar con el grafo a través de listas de adyacencia.
2. Cree una función para sí los vértices son adyacentes.
3. Cree una función para insertar en el grafo.
4. Cree una función para conocer si existe camino entre dos vértices.

Ayuda memoria

```cpp
#include <iostream>
#include <vector>
using namespace std;

// Function to add an edge between vertices u and v
void addEdge(vector<vector<int>>& graph, int u, int v) {
    graph[u][v] = 1;
    graph[v][u] = 1;
}
```

```cpp
// Function to print the adjacency matrix
void printGraph(const vector<vector<int>>& graph) {
    cout << "Adjacency Matrix:\n";
    for (const auto& row : graph) {
        for (int val : row) {
            cout << val << " ";
        }
        cout << '\n';
    }
}

int main() {
    int V; // Number of vertices
    cout << "Enter the number of vertices: ";
    cin >> V;

    // Create an empty adjacency matrix
    vector<vector<int>> graph(V, vector<int>(V, 0));

    graph[0][1] = 1;
    addEdge(graph,2,3);

    // Print the adjacency matrix
    printGraph(graph);

    return 0;
}
```