**ESTRUCTURA DE DATOS 2**
**Código ST0247**

# Laboratory practice No. 1: Graphs Implementation

**Miguel Ángel Zapata Jiménez**
Universidad Eafit
Medellín, Colombia
mazapataj@eafit.edu.co

**Santiago Ochoa Castaño**
Universidad Eafit
Medellín, Colombia
sochoac1@eafit.edu.co

## 3) Practice for final project defense presentation

**3.1** The data structure to represent the city was to prioritize time over memory because it makes easier to have access to the data. For this reason, the algorithm implements an adjacency matrix. The type of data which save the matrix is an abstract data type, collecting the weight and name of the arc.

**3.2** The memory complexity of an adjacency matrix is O(n^2). For this specific case, the graph has 300.000 vertex the memory consumption is 300.000^2.

**3.3.** A conditional operator was used that evaluates if the newly read node was equal to 10,000, it is changed by a 0.

**3.4.** The algorithm works in a simple way. The main idea is to visit each node using the graph traversals method depth first search. In this way, for each node will be assign a color and them visit the adjacent nodes of it recursively. If a node had a color assigned previously, the algorithm will compare it with the previous node color to determinate if it has the same color. In the affirmative case, the method will return false.

**3.5**

```
public class Algorithm {

    public static boolean DFSColorFC(DigraphAM2 g){
        String[] visitados = new String[g.size]; // C_1
        int origen = g.getFirst(); //C_2
        return DFSColorFCAux(g, origen, visitados, "verde");
    }

    private static boolean DFSColorFCAux(DigraphAM2 g, int origen, String[] v, String color){
        if(v[origen] == null){ // C_3
            for(Integer s : g.getSuccessors(origen)){ // O(n)
                if(color.equals("verde")){ // C_4
                    v[origen] = "amarillo"; // C_5
                    if(!DFSColorFCAux(g, s, v, "amarillo")){
                        return false; // C_7
                    }
                }else{   // C_8
                    v[origen] = "verde"; // C_9
```

**PhD. Mauricio Toro Bermúdez**
Professor | School of Engineering | Informatics and Systems
Email: mtorobe@eafit.edu.co | Office: Building 19 – 627
Phone: (+57) (4) 261 95 00 Ext. 9473

```
                    if(!DFSColorFCAux(g, s, v, "verde")){
                        return false; // C_11
                    }
                }
            }
        }else{ // C_12
            if(v[origen].equals(color)){ // C_13
                return false; // C_14
            }else{ // C_15
                if(color.equals("verde")){ // C_16
                    return true; // C_17
                }else{ // C_18
                    return true; // C_19
                }
            }
        }
        return true; // C_20
    }
}
```

**Asymptotic complexity:**
**O(n)**

**3.6** The variable (n) represents the among of vertex that have the graph and can represent the number of successors that have each vertex.

## *4) Practice for midterms*

### *4.1*

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
|---|---|---|---|---|---|---|---|---|
| 0 | | | | 1 | 1 | | | |
| 1 | 1 | | 1 | | | 1 | | |
| 2 | | | | | 1 | | 1 | |
| 3 | | | | | | | | 1 |
| 4 | | | 1 | | | | | |
| 5 | | | | | | | | |
| 6 | | | 1 | | | | | |
| 7 | | | | | | | | |

### *4.2*

*0->[3,4]*
*1->[0,2,5]*
*2->[1,4,6]*
*3->[7]*
*4->[2]*
*5->[]*
*6->[2]*
*7->[]*

### *4.3*

**b)** O(n^2)

**PhD. Mauricio Toro Bermúdez**
Professor | School of Engineering | Informatics and Systems
Email: mtorobe@eafit.edu.co | Office: Building 19 – 627
Phone: (+57) (4) 261 95 00 Ext. 9473

**ESTRUCTURA DE DATOS 2**
**Código ST0247**

**PhD. Mauricio Toro Bermúdez**
Professor | School of Engineering | Informatics and Systems
Email: mtorobe@eafit.edu.co | Office: Building 19 – 627
Phone: (+57) (4) 261 95 00 Ext. 9473