Laboratory practice 5: Graphs

Santiago Isaza Cadavid
Universidad EAFIT
Medellín, Colombia
sisazac@eafit.edu.co

Hamilton Smith Gómez Osorio
Universidad EAFIT
Medellín, Colombia
hsgomezo@eafit.edu.co

October 22, 2018

1) Project questions Simulation

1.a. Write an explanation between 3 and 6 lines of the code of number 1.1. How it works, how the graph is implemented with matrices and the lists you made, highlighting the data structures and algorithms used

In the implemented method with matrices we established a value one (1) for those graphs that were connected and zero (0) for those that did not and all the information was taken from this matrix. For the case with lists we created an array list in which for each position of this (that indicated a vertex) it was assigned a linked list with the vertex-value pairs of each of the graphs attached to it.

1.b. To represent the map of the city of Medellín from the exercise of number 1.3, What is better to use, Adjacency Matrices or Adjacency Lists? Why?

For the exercise of graphs in the city it is advisable to use the adjacent lists since they only keep the information of those graphs that are connected and not all as it would be with the implementation of adjacent matrices. With this structure, memory would be saved although at the time of execution a very similar performance is had.

1.c. What is better to use, Adjacency Matrices or Adjacency Lists? Why?

It is better to use adjacent lists as it saves memory and allows adding or removing elements without problems, which can not be done efficiently with adjacent matrices.

1.d. What is better to use, Adjacency Matrices or Adjacency Lists? Why?

If you want to save memory it is better to use the adjacency lists but at the time of execution, find the route from the starting point to the end point. The procedures must be the same since in each case the connections between nodes must be analyzed, see the connections of these new nodes with others.

1.e. Exercise 2.1's complexity

i.

```
/* Title: esColoreable
 * Authors: Mesa, Daniel
 * Date: October 22, 2018
 * Taken from:https:https://github.com/damesaa201710054010/
 ST0247-032/blob/master/laboratorios/lab01/ejercicioEnLinea/Ejercicio2.java
 */
public boolean isColorable(int [][] graph)
{
    int [] Color = new int[graph.length]; //C1
    int i = 0; //C2
    while(i < Color.length) //n
    {
        Color[i] = -1; //C3*n
        i++; //C4*n
    }
    Color[0] = 1; //C5
    boolean is =  aux(graph, Color); //C6*($n^2$)
    return is? true : false; //C7
}
```

$$T(n) = C1 + C2 + n + C3 * n + C4 * n + C5 + C6 * n^2 + C7$$
$$T(n) = O(C1 + C2 + n + C3 * n + C4 * n + C5 + C6 * n^2 + C7)$$
$$T(n) = O(C * n^2)$$
$$T(n) = O(n^2)$$

The complexity of this algorithm is O(n^2)

i.

```
/* Title: aux
 * Authors: Mesa, Daniel
 * Date: October 22, 2018
 * Taken from:https:https://github.com/damesaa201710054010/
 ST0247-032/blob/master/laboratorios/lab01/ejercicioEnLinea/Ejercicio2.java
 */
private boolean aux(int [][] graph, int [] color)
{
    Stack<Integer> round =  new Stack<>(); //C1
    round.push(0); //C2
    while(round.size() != 0) //n
    {
        int actual = round.pop(); //C4*n
        if(graph[actual][actual] == 1) return false; //C5*n
        for(int i = 0; i < graph.length; i++) // n*n
```

```
            {
                if(graph[actual][i] == 1 && color[i] == -1) //C6*n*n
                {
                    color[i] = 1-color[actual]; //C7*n*n
                    round.push(i); //C8n*n
                }else if(graph[actual][i] == 1  &&
                color[actual] == color[i])  return false;//C9*n*n
            }
        }
        return true; //C10
    }
```

$$T(n) = C1 + C2 + n + C4 * n + C5 * n + n * n + C6 * n * n + C7 * n * n + C9 * n * n + C10$$

$$T(n) = O(C1 + C2 + n + C4 * n + C5 * n + n * n + C6 * n * n + C7 * n * n + C9 * n * n + C10)$$

$$T(n) = O(C * n * n)$$

$$T(n) = O(n^2)$$

The complexity of this algorithm is O(n^2)

i.

```
/* Title: esColoreable
 * Authors: Mesa, Daniel
 * Date: October 22, 2018
 * Taken from:https:https://github.com/damesaa201710054010/
 ST0247-032/blob/master/laboratorios/lab01/ejercicioEnLinea/Ejercicio2.java
 */
public static void main(String[] args)
{
    Exercise2 g = new Exercise2(); //C1
    Scanner console =  new Scanner(System.in); //C2
    int vertices= console.nextInt(); //C3
    int cont = 1; //C4
    int edges = 0; //C5
    int origin = 0; //C6
    int destiny = 0; //C7
    while(vertices != 0) //n
    {
        edges = console.nextInt(); //C8*n
        int [][] graph = new int[vertices][vertices]; //C9*n
        cont = 1; //C10*n
        while(cont <= edges) //n*n
        {
            origin = console.nextInt(); //C11*n*n
            destiny = console.nextInt(); //C12*n*n
```

```
                graph[origin][destiny] = 1; //C13*n*n
                cont++; //C14*n*n
            }
            String res = g.isColorable(graph)?
            "BICOLORABLE" : "NOT BICOLORABLE"; // C15*n*n
            System.out.println(res); //C16
            vertices = console.nextInt(); //C17
        }
    }
```

$$T(n) = C1+C2+C3+C4+C5+C6+C7+n+C8*n+C9*n+C10*n+n*n+C11*n*n+$$
$$C12*n*n+C13*n*n+C14*n*+C15*n*+C16+C17$$
$$T(n) = O(C1+C2+C3+C4+C5+C6+C7+n+C8*n+C9*n+C10*n+n*n+C11*n*n+$$
$$C12*n*n+C13*n*n+C14*n*+C15*n*+C16+C17)$$
$$T(n) = O(C*n^2)$$
$$T(n) = O(n^2)$$

The complexity of this algorithm is O(n^2)

1.f. Explain what the variables means in the previous exercises

The variables C, are constants, the n always represents the number of vertices that the graph has in the worst-case scenario, it can travel across all the vertices being n * n regardless of whether there is a connection or not, because it has to check it you have to go through the entire matrix, besides, n in the main method is the number of edges that in the worst case is the same as vertices, but may be smaller.

2) Midterm Simulation

2.a. Exercise 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								

2.b. Exercise 2

- 0 -> [3,4]

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

2.c. Exercise 3

2.3.1 How much memory (not time but memory) occupies a representation using adjacency lists for the worst graph directed with n vertices?

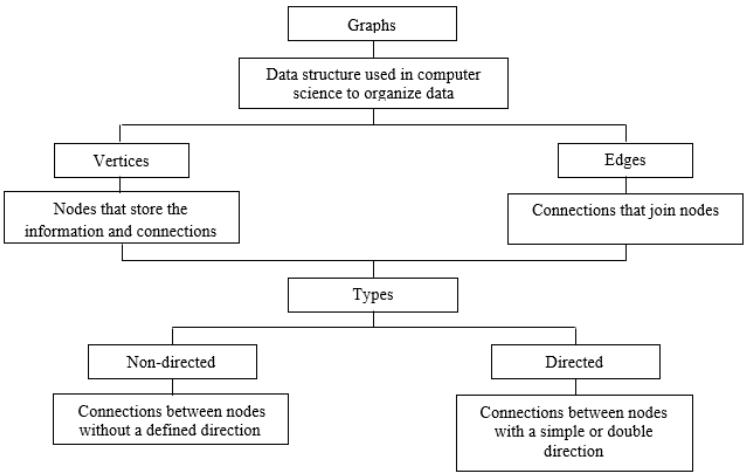
b) O(n^2)

3) Recommended reading

3.a. Summary

The graphs are a data structure very used in the organization of information and is also very used in computer science to analyze routes and connections. To be implemented, certain nodes (vertices) are used, which are connected to others due to edges which represent the interaction with other elements. There are two types of graphs, the directed graph and the non-directed graph, the first type is the graph whose edges have a defined direction, it shows a specific path of access form one graph to another, and on the other hand, the non -directed graphs are those whose nodes are connected but they do not have a defined path (order)

3.b. Chart



4) Team work and progress

4.a. Meetings

Team work (meetings)		
Date	time	Description
17.10.18	1 hour	work distribution
20.10.18	3 hours	construction of the report part 1
22.10.18		construction of the report part 2

4.b. History of changes in the code

History of changes in the code		
version	includes	status
1.0	Adjacency matrices and adjacency list Greatest successor problem	unfinished
2.0	Adjacency matrices and adjacency list Greatest successor problem Colorable or not colorable problem	finished

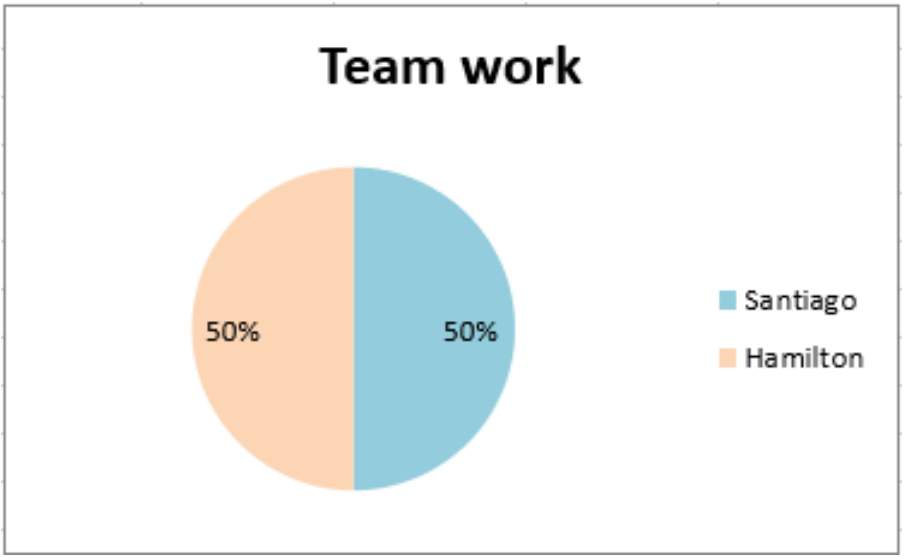
Professor Mauricio Toro Bermudez
Phone: (+57)(4)2619500 Ext. 9473. Office: 19 – 627
E-mail: mtorobe@eafit.edu.co

4.c. History of changes in the report

History of changes in the report		
version	includes	status
1.0	project questions simulation (3.1- 3.2 -3.3 -3.4)	unfinished
	midterm simulation	
	recomended reading	
2.0	project questions simulation finished	finished
	midterm simulation	
	recomended reading	
	Team work and gradual process	

4.d. Team Work

Member	part of the laboratory	Description
Hamilton	1	1.1 Adjacency matrices and adjacency list
		1.2 Greatest successor problem
	3	3.1 Bow 1.1 excercise work
		3.2 Medellin city map- best structure
		3.3 Best structure in general cases
		3.4 Routing table- best structure
	5	Recommended reading
	6	Team work and gradual process
Santiago	2	Colorable or not colorable problem
	3	3.5 2.1 Exercise completity
		3.6 Meaning of 'n' and 'm' in the complexity
	4	Midterm simulation
	6	Team work and gradual process
	7	Translate lab



Professor Mauricio Toro Bermudez
Phone: (+57)(4)2619500 Ext. 9473. Office: 19 – 627
E-mail: mtorobe@eafit.edu.co