Lab N. 1: Graph Implementation

Juan Felipe Londoño Gaviria
Universidad Eafit
Medellín, Colombia
jflondonog@eafit.edu.co

Juan Pablo Giraldo Restrepo
Universidad Eafit
Medellín, Colombia
jpgiraldor@eafit.edu.co

3) Project questions drill

3.1 Basically what the exercise does is that when done with matrixes, first it creates the matrix of [size][size] which in this case size is the number of values that the graph has. In other words n. It implements the method getWeight which returns the weight of each connection. After, addArc is the one in charge of adding arcs to the graph. Last, getSuccesors returns the succesors of each node. Now with graphs made with lists, it's a bit similar. First an arraylist of pairs is created, and all the methods used with matrixes are also implemented in order for it to have the same functionalities.

3.2 Both are convenient because they work as directed and non-directed. If there is a certain situation in which one or another Is required one may choose depending on what is needed, but both are convenient.

3.3 Adjacency matrixes because it can show all the possible routes.

3.4 Adjacency matrixes because it will ease the vision of the average of friendships that a Facebook user has.

3.5 Adjacency lists because it will only show the shortest path**.**

4)Exam Drill

1. The adjacency matrix is as follows:

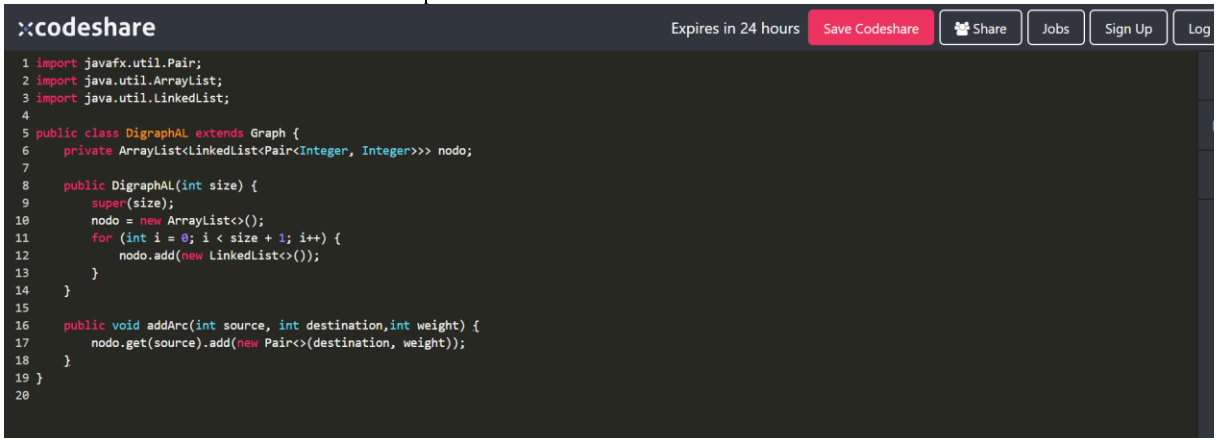
	0	1	2	3	4	5	6	7
0				1	1			
1	1		1			1		
2		1			1		1	
3								1
4			1					
5								
6			1					
7								

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

3. O(n²) Is the complexity in the worst case scenario.

5) Team Work

Platforms such as codeshare, Whatsapp and Google docs were used in order to facilitate communication between us. Some screenshots will be shown as proof.

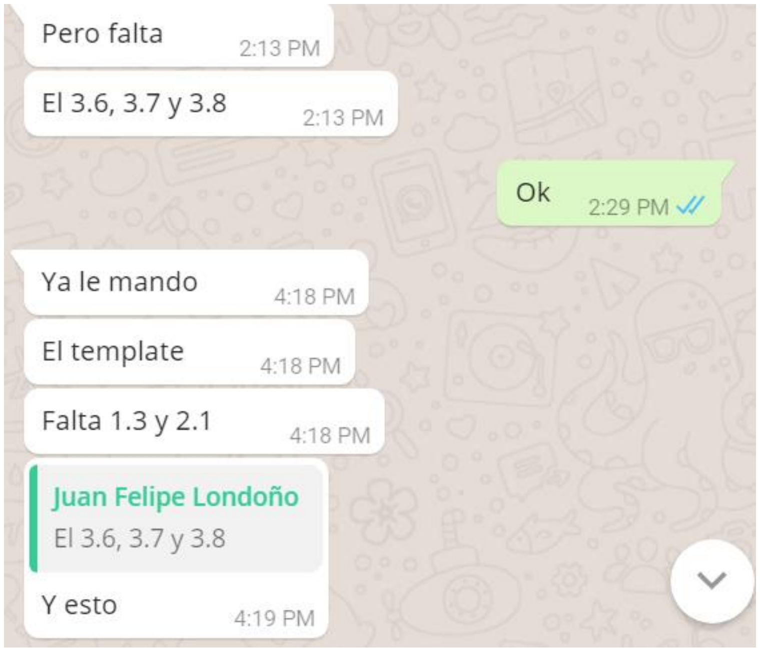


```java
import javafx.util.Pair;
import java.util.ArrayList;
import java.util.LinkedList;

public class DigraphAL extends Graph {
    private ArrayList<LinkedList<Pair<Integer, Integer>>> nodo;

    public DigraphAL(int size) {
        super(size);
        nodo = new ArrayList<>();
        for (int i = 0; i < size + 1; i++) {
            nodo.add(new LinkedList<>());
        }
    }

    public void addArc(int source, int destination,int weight) {
        nodo.get(source).add(new Pair<>(destination, weight));
    }
}
```



Pero falta 2:13 PM

El 3.6, 3.7 y 3.8 2:13 PM

Ok 2:29 PM

Ya le mando 4:18 PM

El template 4:18 PM

Falta 1.3 y 2.1 4:18 PM

Juan Felipe Londoño
El 3.6, 3.7 y 3.8

Y esto 4:19 PM