Laboratory practice No. 3: Linked Lists and Dynamic Vectors

Martin Ospina Uribe
Universidad Eafit
Medellín, Colombia
mospinau1@eafit.edu.co

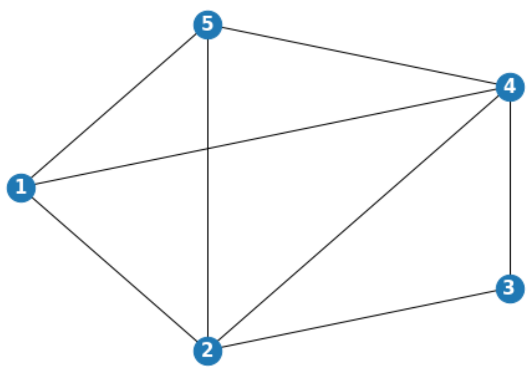
María José Bernal Vélez
Universidad Eafit
Medellín, Colombia
mjbernalv@eafit.edu.co

3) Practice for final project defense presentation

3.1 Complexity of project simulation exercises

1.1 Map

In this exercise we propose 2 alternatives for the solution. The first one reads two csv files, one with places and another one with streets and roads and creates a dictionary with each one. The second one reads the same files and creates a graph of the city.



The figure above shows an example of a map created using graphs.

- **Complexity:**

$$O(n, m)$$

- n = number of places (vertices)
- m = number of streets and roads (arcs)

1.4. Refrigerators

In this problem we create an algorithm that receives one stack of refrigerators and a queue of requests and distributes the refrigerators to the different stores so that the first order that was received is completed first, and the refrigerators are stored in a room with one door only.

- **Complexity:**

$$O(m * n)$$

- m = amount of requests
- n = number of refrigerators available

1.5 Doubly linked list

This algorithm implements a doubly linked list, with the methods get, add, remove, size, contains, contains position and to string.

- **Complexity:**

Get: $O(n)$
Add: $O(n)$
Remove: $O(n)$
Size: $O(1)$
Contains: $O(n)$
Contains position: $O(n)$
To string: $O(n)$

- n = size of the doubly linked list

1.6 Bank

This algorithm simulates how queues in a bank work. To achieve this, it has 4 lines that are represented by using a queue and 2 ATMs. It returns the order in which the two ATMs will receive clients, as ATM 1 receives one person and then ATM two receives another one, and people are attended in ascending order of lines (line 1, then line 2, and so on).

- **Complexity:**

$$O(m)$$

- m = amount of people in the bank (people in line 1 + line 2 + line 3 + line 4).

3.2 How does exercise 2.1 (broken keyboard) work?

The broken keyboard algorithm uses a linked list to "fix" the input that was typed into the keyboard. The broken keys are "start" and "end", that are represented with "[" y "]" respectively.

First an empty linked list is created, and it will store the fixed output. A temporal string is also created to store part of the text, and a boolean as an indicator to see if the temporal string is going to be added at the beginning (true) or at the end (false) of the linked list. To do this, a cycle is used to iterate through the whole text. If the current character is not a broken key, it is added to the temporal string; else if the current character is a "[", it adds the temporal string to the linked list by checking the boolean variable and sets it to True; else if it is a "]" it attaches the temporal string to the list, and sets the boolean to False; and finally, if the character corresponds to the final position of the string, it also adds it. To finish, it returns the fixed text as a string.

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

3.3/3.4 Explanation and complexity of project simulation exercises and meaning of variables

2.1. Broken keyboard
*Explanation on point 3.2.

- **Complexity:**

$$O(n)$$

- n = length of the text that is going to be fixed.

2.4 Balanced Brackets

This algorithm checks whether a given set of brackets (which could be (,), [,], {, or }) is a matched pair or not. A matched pair is when they are correctly organized so that every bracket that is opened is closed in the correct order. If it does, it returns YES, else, it returns NO.

- **Complexity:**

$$O(n)$$

- n = length of the text (set of brackets) sent.

4) Practice for midterms

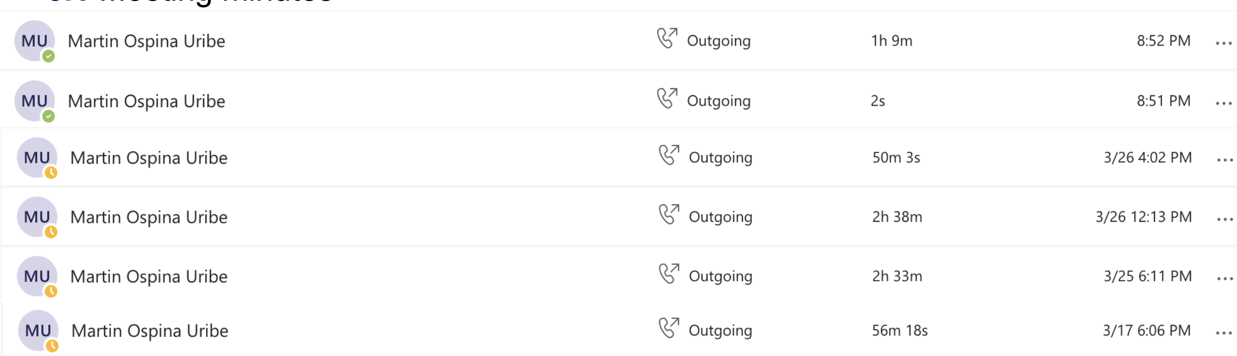
4.1 1. $res = res * 2 + int(vector[i])$
2. $O(n)$
4.2 c
4.3 4.3.1. iv
4.3.2. i
4.4 1. token
2. c
4.5 a
4.6 a
4.7 iii
4.8 d
4.9 4.9.1. a
4.9.2. c
4.9.3 c
4.10 4.10.1 d
4.10.2 a
4.10.3 b
4.11 4.11.1 c
4.11.2 b

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

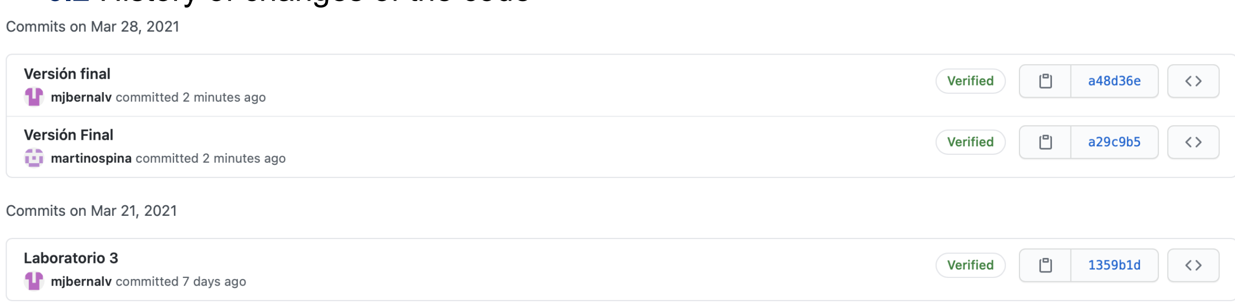
6) Teamwork and gradual progress (optional)

Member	Date	Done	Doing	To do
Martín and Maria José	17/03/2021	Practice for midterms	Project simulation exercises	Practice for final project defense presentation
Martín and Maria José	25/03/2021	Exercises 2.1 and 2.4	Finding exercise complexities	Practice for final project defense presentation
Martín and Maria José	26/03/2021	Exercises 1.1, 1.4	Exercises 1.5 and 1.6	Work on the report
Martín and Maria José	28/03/2021	Work and finish the report. Upload laboratory on GitHub.		

6.1 Meeting minutes

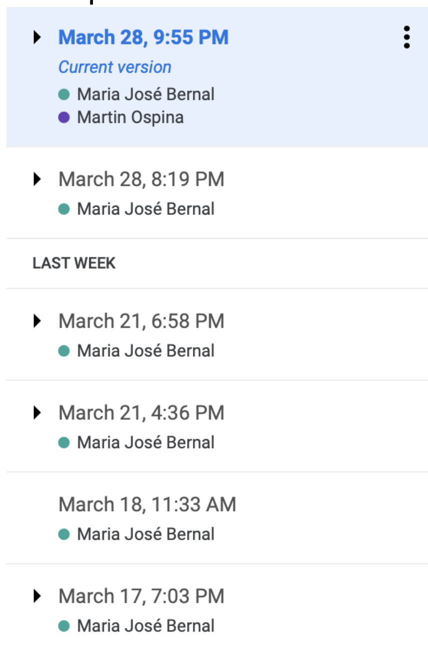


6.2 History of changes of the code



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

6.3 History of changes of the report



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