# Laboratory practice No. 3: Linked List

**Jairo Miguel Marulanda Giraldo**
Universidad Eafit
Medellín, Colombia
jmmarulang@eafit.edu.co

**Sebastian Herrera Beleño**
Universidad Eafit
Medellín, Colombia
sherrerab1@eafit.edu.co

## 3) Practice for final project defense presentation

### 3.1

We implemented a data structure by constructing a simple hash map made from linked lists. Linked lists, compared to matrices, have the disadvantage of slower iteration of elements and inefficient random access. However, they present two decisive advantages. First, addition of elements, at the beginning of or the end, it's very efficient. Second, since they do not require memory cohesion, they have very efficient memory utilization.
Since we are working with such a large data set, this memory and addition speed advantages make the use of linked lists more desirable. Additionally, hash maps have faster access to elements, which may help compensate for the relative slowness of linked lists.

To understand the scope of space complexity required by a matrix, let us now assume we wish to solve this exercise by the way of them. We would then need a matrix n*n, where n is the number of vertices. This means the structure would have a space complexity of O(n*n).
Since need to store around 300,000 vertices, which are composed of two doubles, the algorithm has a space complexity of approximately 90.000.000.000.

To save all data in hash map the complexly or O(n*m) where m is the length of nodes and m is the length of edges.

To extract data from the finish HashMap the complex is o(n) where n is the size of the final HashMap.

Additionally, we note that our implementation had no need for the IDs to begin at 0.

### 3.2

**For the exercise 2.1:**

We get for the worst case:

T(n) = c1 + c2 + c3 + c4 + c5 + n*c6 + n*c7 + n*c8 + n*c9 + n*c10 + c 17 + n*c18 + n*c19

With n the length of the string text and ci constant.

Note that for the final part of the algorithm, where the sections are printed, we know that the number of sections is equal or smaller than the length of the string.

Then
T(n) = O(n)

**3.3**
**3.4**
**3.5**
**3.6**

## 4) Practice for midterms

***4.1*** Line 4: res = res + (2*int(vector[i]))**(len(vector) - 1 - i) and the complexity of the algorithm is O(n).
***4.2*** c: O(n).
***4.3*** The output is iv: 0, 2, 4, 6, 8, 10 and the complexity is I: O(1).
***4.4*** Line 21: output.append(token).append(' '); and the worst case for pop is c: O(1).
***4.5*** a: [7,8,3,1,2,9]

## *5) Recommended reading (optional)*

Mapa conceptual

## 6) Team work and gradual progress (optional)

***6.1*** Meeting minutes
***6.2*** History of changes of the code
***6.3*** History of changes of the report

## Bibliography

Dhiraj Rai (2019) Pros and Cons of Collection Types in Java. https://www.devglan.com/corejava/pros-and-cons-collection-java

Algorithmics: Data Structures and Data Types. Retrieved from: http://www0.cs.ucl.ac.uk/staff/Daniel.Alexander/D0b/alg07.pdf

**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

Common Data Structure Operations
https://www.bigocheatsheet.com

**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