Laboratory practice No. 3: LinkedList and ArrayList

Juan S. Cárdenas Rodríguez
Universidad EAFIT
Medellín, Colombia
jscardenar@eafit.edu.co

David Plazas Escudero
Universidad EAFIT
Medellín, Colombia
dplazas@eafit.edu.co

September 24, 2017

1) CODE FOR DELIVERING ON GITHUB

The source code can be found in `Code.py` inside the `codigo` folder.

2) ONLINE EXERCISES

The source code can be found in `Code.py` inside the `codigo` folder.

3) SIMULATION OF PROYECT PRESENTATION QUESTIONS

3.a. Complexity of algorithms using ArrayList and LinkedList

	ArrayList	**LinkedList**
Exercise 1.1	$O(n)$	$O(n)$
Exercise 1.2	$O(n)$	$O(n)$
Exercise 1.3	$O(n^2)$	$O(n^3)$
Exercise 1.4	$O(n^2m)$	$O(nm)$

Table 1: Complexity for algorithms for ArrayList and LinkedList

3.b. How does exercise 2.1 work?

Exercise 2.1 is simple. First, every time that it finds the character '[', it assigns 0 to the variable `index`; representing that it will insert in the beginnig from that point. On the other hand, if it finds a ']', `index` is assigned with the end of the array.

In any other case, when it finds a character different from '[' and ']', it just adds this characters at the `index` position; therefore, it adds 1 to the `index` to keep it at the end or at the start.

3.c. What's the complexity of exercise 2.1?

```
def manage_string(string):                  # c0
    ll = LinkedList()                       # c1
    array = string.split("\n")              # c2
    resp = ""                               # c3
    for line in array:                      # c4*m
        ll.clear()                          # c5*m
        index = 0                           # c6*m
        for char in line:                   # c7*m*n
            if char == "[":                 # c8*m*n
                index = 0                   # c9*m*n
            elif char == "]":               # c10*m*n
                index = ll.size()           # c11*m*n
            else:                           # c12*m*n
                ll.insert(char, index)      # c13*m*n
                index += 1                  # c14*m*n
        for item in ll:                     # c15*n
            resp += item                    # c16*n
        resp += "\n"                        # c17
    return resp                             # c18
```

Therefore, `manage_string` is $O(c_{k1} + c_{k2}m + c_{k3}n + c_{k4}mn)$. When the product and sum properties are applied, `manage_string` is $O(mn)$, where n is the length of the string and m is the number of lines.

4) TEST SIMULATION

i. c)

ii. c)

iii.
- q.size()
- ¡=
- q.remove()
- q.remove()

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