Laboratory practice No. 3: LinkedLists and ArrayLists

Mauricio Jaramillo Uparela
Universidad Eafit
Medellín, Colombia
mjaramillu@eafit.edu.co

Juan Pablo Ossa Zapata
Universidad Eafit
Medellín, Colombia
jpossaz@eafit.edu.co

3) Practice for final project defense presentation

3.1

Exercise	ArrayList	LinkedList
1.1	**O(nm)** Where n is the student count and m is the course count.	**O(nm)** Where n is the student count and m is the course count.
1.2	**O(n)**	**O(n)**
1.3	**O(nm)** Where n is the number of requests and m is the number of fridges in a request.	**O(nm)** Where n is the number of requests and m is the number of fridges in a request.
1.4	**O(n)**	**O(1)** for add. **O(n)** for remove and insert.

For this set of problems, it is essentially equivalent in terms of complexity to use either ArrayLists or LinkedLists. This is due to the fact that all of them involve linear lookup of data, negating the positive effect of ArrayLists; and most of them don't modify the data at all, thus negating the positive effect of LinkedLists.

Something to note is the fact that problem 1.1 and 1.3 can be solved in O(log n), using array lists, by sorting the list before operating on it, and then utilizing binary lookup instead of linear lookup.

3.2 Explanation of exercise implementation:

- **2.1:** Our solution to this problem involves the use of a parser that holds a “selected node” variable and a “flag” variable. This variable is a reference to a node in a linked list. The parser iterates over the input string. When it finds a ‘[‘ character, it makes it so that the selected node is set to the head of the linked list, and it raises the flag. Similarly, when it finds a ‘]’ character, it makes it so that the selected node is set to the tail of the linked list (without raising the flag). When it finds any other character, it simply adds it before or after the selected node, depending on the state of the flag. It also raises the flag after a character is added before the selected node.

3.3 Complexity

- **2.1:** O(n)

3.3 Complexity

- **2.1:** n is the length of the string.

4) Practice for midterms

4.1 *a*
4.2 *b*
4.3
    a. q.size() > 1
    b. <=
    c. q.remove()
    d. q.get(0)
4.4 opcional
4.5.1 auxiliar1.size() > 0 , auxiliar2.size() > 0
4.5.2 personas.offer(edad)
4.6 a
4.7 c
4.8
    4.8.1 a
    4.8.2 b
    4.8.3 c
4.9
    4.9.1 d
    4.9.2 a
    4.9.3 b
4.10
    4.10.1 b
    4.10.2 b
4.11
    4.11.1 !s1.isEmpty()
    4.11.2 s1.pop()
    4.11.3 s2.pop()
4.12
    4.12.1 ii

- *4.12.2 i.O(1)*

4.13

- *4.13.1 iii*
- *4.13.2 iv*

4.14 iii