Laboratory practice 3: Linked List and Arraylist

Santiago Isaza Cadavid
Universidad EAFIT
Medellín, Colombia
sisazac@eafit.edu.co

Hamilton Smith Gómez Osorio
Universidad EAFIT
Medellín, Colombia
hsgomezo@eafit.edu.co

October 8, 2018

1) Project questions Simulation

1.a. Explain the implementation of Exercise 2.2

For the solution of the problem, we considered a LinkedList, the variables int "index" and boolean "start" helped us to solve the problem in a relatively easy way. A for cycle runs through the entered string, and in case of finding the character "[", the variable "start" will be true and false if we find the character "]", then an if will evaluate other condition, the character at index n, if the character at index is not "[" neither "]" then the status of the variable start is evaluated. If it is true, the character is inserted in the current index of the list, and if it is false, the character is added at the end of the list and in both cases we proceed to increase the variable index, we increase it by one. After running through the entire string, we go through the list of characters already ordered and a new string is created with the characters that are in the list, finally, we return this new string.

1.b. Exercise 2.2's complexity

i.
```
public static String pc(String str){
      ArrayList<String> list = new LinkedList<>(); //C1
      boolean start= true; // C2
      int index=0;  //C3
      String newString=""; //C4
      for(int i=0;i<str.length()-1;i++){ //C5*n
          if(str.substring(i,i+1).equals("[")){ //C6*(n-1)
              start=true;//C7*(n-1)
              index=0;//C8*(n-1)
          }
          else if(str.substring(i,i+1).equals("]")){ //C9*(n-1)
```

```
                start=false; //C10*(n-1)
            }else if(!str.substring(i,i+1).equals("[") //C11*(n-1)
            && !str.substring(i,i+1).equals("]")){ //C12*(n-1)
                if(start){ //C13*(n-1)
                    list.add(index,str.substring(i,i+1)); //C14*(n-1)*1
                    index++; //C15*(n-1)
                }else{//C16*(n-1)
                    list.add(str.substring(i,i+1));//C17*(n-1)
                }
            }
        }

        for(int i=0;i<list.size();i++){ //C18*n
            newString= newString+list.get(i);//C19*(n-1)
        }

        return newString; //C20
    }
```

$$T(n) = C1 + C2 + C3 + C4 + C5 * n + C6 * (n-1)$$
$$C7 * (n-1) + C8 * (n-1) + C9 * (n-1) + C10 * (n-1) + C11 * (n-1) +$$
$$C12 * (n-1) + C13 * (n-1) + C14 * (n-1) * 1 + C15 * (n-1) + C16 * (n-1) +$$
$$C17 * (n-1) + C18 * n + C19 * (n-1)$$

$$T(n) = O(C1 + C2 + C3 + C4 + C5 * n + C6 * (n-1)$$
$$C7 * (n-1) + C8 * (n-1) + C9 * (n-1) + C10 * (n-1) + C11 * (n-1) +$$
$$C12 * (n-1) + C13 * (n-1) + C14 * (n-1) * 1 + C15 * (n-1) + C16 * (n-1) +$$
$$C17 * (n-1) + C18 * n + C19 * (n-1))$$

$$T(n) = C1 + C2 + C3 + C4 + C5 * n + C6 * (n-1) + C7 * (n-1) +$$
$$C7 * (n-1) + C8 * (n-1) + C9 * (n-1) + C10 * (n-1) + C11 * (n-1) +$$
$$C12 * (n-1) + C13 * (n-1) + C14 * (n-1) * 1 + C15 * (n-1) + C16 * (n-1) +$$
$$C17 * (n-1) + C18 * n + C19 * (n-1)$$

$$T(n) = O(C1 + C2 + C3 + C4 + C5 * n + C6 * (n-1) + C7 * (n-1) +)$$
$$C7 * (n-1) + C8 * (n-1) + C9 * (n-1) + C10 * (n-1) + C11 * (n-1) +$$
$$C12 * (n-1) + C13 * (n-1) + C14 * (n-1) * 1 + C15 * (n-1) + C16 * (n-1) +$$
$$C17 * (n-1) + C18 * n + C19 * (n-1)$$
$$T(n) = O(14 * (n-1) + n)$$

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

$$T(n) = O(13n - 14)$$
$$T(n) = O(13n)$$
$$T(n) = O(n)$$

The complexity of this algorithm is O(n)

1.c. Explain what the variables means in the previous exercises

- **LinkedList list:** the list, data structure, where the strings are going to be stored, in order to the status of the variable start.
- **Boolean start:** determines if the substrings are going to be added at the end of the list or at the beginning.
- **Integer index:** does the function of a counter, determines the current position at the list.
- **String newString:** is the string in which the substrings stored in the list are going to be added, according to the their position in the list.

2) Midterm Simulation

2.a. Exercise 1

a) Looking for data in the list

2.b. Exercise 2

c) O(n)

2.c. Exercise 3

2.3.1 Complete line 02

q.size()>1

2.3.2 Complete line 03

<=

2.3.3 Complete line 04

q.remove()

2.3.4 Complete line 06

q.remove()

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

2.d. Exercise 4

2.4.1 What is the condition of the while cycle?

lista.size()

2.4.2 Complete line 07

lista.add(auxiliar.pop())

2.e. Exercise 5

2.5.1 What is the condition of the while cycle? Lines 12 and 16

auxiliar1.size()>0 , auxiliar2.size()>0

2.5.2 Complete line 18

personas.offer(edad)

2.f. Exercise 6

c) O(n^2)

2.g. Exercise 7

c) O(n^3)

2.h. Exercise 8

d) O(1)

2.i. Exercise 9

2.9.1 What is the asymptotic complexity in the worst case-scenario?

a) O(k)

2.9.2 What does the algorithm prints when k=21

b)9

2.9.3 What is the asymptotic complexity in the worst case-scenario, when you are adding data to a queue of n elements?

c) O(1)

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

2.j. Exercise 10

2.10.1 What is the asymptotic complexity in the worst case-scenario?

d) O(n)

2.10.2 What does the algorithm prints when x=8 and n=20

a)6

2.10.3 What is the asymptotic complexity in the worst case-scenario, when you are searching whether data is or not in a stack?

b) O(n)

3) Recommended reading

3.a. Summary

The linked lists are data structures in which is possible to store different types of elements without being properly organized in the memory of the computer, it means, it has the control over the consecutive order of the elements by links or connectors between elements but these can also be anywhere in the memory.

This structure is very useful because it allows to optimize the memory space, it has a shorter execution time when inserting or deleting an element and it is very used to build other structures such as stacks or queues. For its implementation nodes are usually used, which are the "closets" that are interconnected and store the desired information, which have positional attributes referring to the data next to it, and in the case of double-linked list, the previous element.