**Laboratorio Nro. 3  
Listas Enlazadas(LinkedList) y Listas Hechas con arreglos(ArrayList)**

|  |  |
| --- | --- |
| **Susana Álvarez**  Universidad Eafit  Medellín, Colombia  salvarezz1@eafit.edu.co | **Mariajose Franco**  Universidad Eafit  Medellín, Colombia  mfrancoo@eafit.edu.co |

**3) Simulacro de preguntas de sustentación de Proyectos**

**3.1**

|  |  |  |
| --- | --- | --- |
| **Ejercicio** | **ArrayList** | **LinkedList** |
| 1. **Consulta 1**   **Consulta 2** | O(n)  O(n) | O(n2)  O(n) |
| 1. **Parte 2.1** | O(n) | O(n) |

* En el primer ejercicio, que es el de la estructura de datos para las notas de la universidad, se implementó con un ArrayList. La complejidad con el ArrayList es o(n) y si fuera implementada con LinkedList sería de O(n2). Es claro que usar ArrayList en este ejercicio es mucho mejor, pero no alcanza a ser lo suficientemente eficiente para este problema porque una universidad cuenta con muchísimos estudiantes, muchos cursos, y mucha información para cada semestre, entonces se necesitaría un algoritmo o(1).
* El ejercicio 2.1 se implemento con listas, al cual tienen una complejidad de O(n) en este algoritmo; si se hubiera implementado con ArrayList la complejidad sería la misma O(n). Se podría necesitar un algoritmo más rápido ya que si son textos demasiado largos el algoritmo tardara un poco en realizar las operaciones y lo ideal sería que fuera casi inmediato.

**3.2** El algoritmo recibe un String el cual va a ingresar el usuario, el String contiene los símbolos “[” y “]”, los cuales van a significar inicio y fin respectivamente; luego se crea una lista enlazada inicialmente vacía, la cual va a cumplir la función de “auxiliar” más adelante. Luego, se inicializa un valor boolean para determinar en que parte del String se encuentran los símbolos de inicio mencionados anteriormente; se recorre el arreglo con un ciclo. Finalmente, estos caracteres que se encuentren después del símbolo de inicio, antes del símbolo de fin o entre ambos, se van a almacenar en la lista enlazada para luego ser impresos.

**3.3** Complejidad ejercicio 2.1: O(n)

**3.4** En el caso de la complejidad del ejercicio 2.1, la n significa la cantidad de veces que se va a realizar el ciclo que se encuentra en el algoritmo. En este caso, el ciclo se repetirá la cantidad de veces como sea la longitud del arreglo, ya que este va desde 0, hasta el final del arreglo.

***4) Simulacro de Parcial***

**4.1) a.** Buscar un dato cualquiera en la lista

**4.2) c.** O(n)

**4.3) a.** q.size()!=1

**b.** <=

**c.** q.pop()

**d.** q.getLast()

**4.4) a.** lista.size()

**b.** lista.push(auxiliar.removeLast())

**4.5.1) 12.** auxiliar1.size() > 0 **16.** auxiliar2.size > 0

**4.5.2)** personas.offer(edad)

**4.6) c.** O(n2)

**4.7) a.** O(n3)

**4.8) c.** O(n)

**4.9.1) a.** *O(k)*

**4.9.2) c.**12

**4.9.3) c.** O(1)

**4.10.1) d.** O(n)

**4.10.2) b.** 8

**4.10.3) b.** O(n)

**4.11.1) b.** *O*(max(*list*)×*n*)

**4.11.2) b.** O(n)

**4.12.1)** s1.size() > 0

**4.12.2)** e

**4.12.3)** s2