Laboratorio Nro. 3
Listas Enlazadas (Linked List) y Listas Hechas con Arreglos (Array List)

Jamer José Rebolledo Quiroz
Universidad EAFIT
Medellín, Colombia
jjrebolleq@eafit.edu.co

Andrés Felipe Tamayo Arango
Universidad EAFIT
Medellín, Colombia
aftamayoa@eafit.edu.co

April 9, 2018

3) Simulacro de preguntas de sustentación de Proyectos

3.1 Teniendo en cuenta lo anterior, calculen la complejidad de cada ejercicio con cada implementación de listas. Es decir, hagan una tabla, en cada fila coloquen el número del ejercicio, en una columna la complejidad de ese ejercicio usando ArrayList y en la otra columna la complejidad de ese ejercicio usando LinkedList.

	ArrayList	LinkedList
Ejercicio 1.1	$O(n)$	$O(n)$
Ejercicio 1.2	$O(n)$	$O(n)$
Ejercicio 1.3	$O(n^2)$	$O(n^2)$
Ejercicio 1.4	$O(M * n)$	$O(M * n)$

i.

```
static int multiply(List<Integer> l){      // ArrayList     LinkedList
  int n = 1;                               // C1            C1
  while(!l.isEmpty())n *= l.remove(0);     // n * 1         n * 1
  return n;                                // C2            C2
}
```

ii.

```
static void SmartInsert(List<Integer> l, int data){     // AL LL
  if(l.indexOf(data) == -1)l.add(data);                 // n
}
```

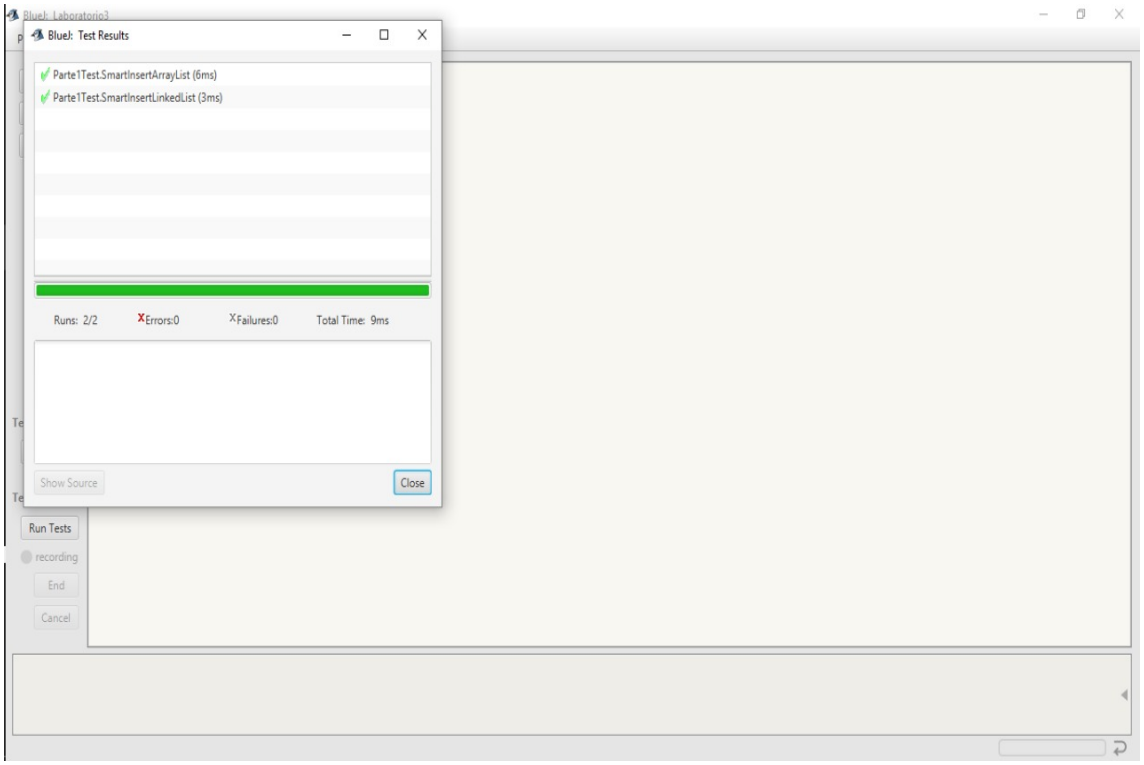
iii.

```
static void punto3(List<Integer> l){                    // AL      LL
  int min = Integer.MAX_VALUE, optime = 0;              // C1 + C2
  for(int i = 0; i < l.size(); i++){                    // n *
      int left = 0, rigth = 0;                          // C3 + C4
      for(int j = 0; j < l.size(); j++){                // n *
          if(j < i)left += l.get(j);                    // C5
          if(j > i)rigth += l.get(j);                   // C6
      }
      if(Math.abs(rigth - left) < min){min = rigth - left; optime = i;}// C7
  }
  System.out.println(optime);                           // C8
}
```

iv.

```
static void punto4(List<Nevera> inventario, List<Solicitud> s){   // AL   LL
  String entregas = "[";              // C1
  boolean primera = false;            // C2
  while(true){                                  // M *
      Solicitud f = s.remove(s.size() - 1);     // n
      int n = f.number;                         // C3
      Entrega e = new Entrega(f);               // C4
      do{                                       // n *
          if(!inventario.isEmpty())
          e.addNevera(inventario.remove(inventario.size() - 1));     // n
          else break;
      }while(e.number < n);
      if(!primera){entregas += e.toString(); primera = true;}
      else entregas += "\n" + e.toString();
      if(s.isEmpty() || inventario.isEmpty())break;
  }
  entregas += "]";
  System.out.println(entregas);
}
```

3.2 Teniendo en cuenta lo anterior, verifiquen, utilizando JUnit, que todos los tests escritos en el numeral 1.2 pasan. Muestren, en su informe de PDF, que los tests se pasan correctamente; por ejemplo, incluyendo una imagen de los resultados de los tests.



3.3 Expliquen con sus propias palabras cómo funciona la implementación del ejercicio 2.1 y [opcionalmente] el 2.2

En este ejercicio con un Scanner leemos un String que nos pasan por consola y lo almacenamos en una variable, luego iteramos a través de ese String separándolo y almacenando en una lista los Strings antes y despues de los símbolos " [" y "] ", luego iteramos a través de esa lista de la siguiente manera: si se encontraba un " [", se mandaba el siguiente elemento al inicio de la lista y se remueve el corchete, y si se encontraba un símbolo "] ", el siguiente elemento se mandaba al final de la lista y se removía el corchete, por último se imprime la cadena resultante de unir todas las cadenas que quedaron la lista.

3.4 Calculen la complejidad del ejercicio realizado en el numeral 2.1 y [opcionalmente] 2.2, y agregarla al informe PDF

```
private static String depure(String s){
        if(s.length() < 2)return s;
        if(s.startsWith("[[") || s.startsWith("[]") || s.startsWith("][")
        || s.startsWith("]]"))return depure(s.substring(1)); // T(n - 1) + C1
        return s.charAt(0) + depure(s.substring(1));
}
```

```
static void punto2(){
        Scanner sc = new Scanner(System.in); // C1
        String s = sc.next(); // C2
        s = depure(s); // T(n - 1)
        while(true){if(s.endsWith("]") || s.endsWith("["))
        s = s.substring(0,s.length() - 1); else break;} // n
        int keys = 0; // C3
        for(int i = 0; i < s.length(); i++)
        if(s.charAt(i) == '[' || s.charAt(i) == ']')keys++; // n
        LinkedList<String> partes = new LinkedList<>(); // C4
        for(int i = 0; i < s.length(); i++){ // n *
            String parte = "";
            String key = "";
            for(int j = i; j < s.length(); j++){ // n *
                if(s.charAt(j) != '[' && s.charAt(j) != ']')
                {parte += s.charAt(j); i++;}
                else{key = (s.substring(j,j+1)); break;} // C
            }
            partes.add(parte);
            if(key.length() != 0)partes.add(key);
        }
        for(int i = 0; i < keys; i++){ // m *
            for(int j = 0; j < partes.size(); j++){ // m *
                if(partes.get(j).equals("[")){ // m
                    partes.add(0, partes.get(j + 1));
                    partes.remove(j + 1);
                    partes.remove(j + 1);
                    keys--;
                }
                if(partes.get(j).equals("]")){
                    partes.add(partes.get(j+1));
                    partes.remove(j + 1);
                    partes.remove(j);
                    keys--;
                }
            }
        }
        String sucess = "";
        for(int i = 0; i < partes.size(); i++)sucess += partes.get(i); // m
        System.out.println(sucess);
    }
```

Complejidad $= O(m^3 + n^2)$

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

4) Simulacro de parcial en el informe PDF

1. ¿Cuál operación tiene mayor complejidad asintotica, para el peor de los casos, en una lista simplemente enlazada?

a) Buscar un dato cualquiera en la lista
b) Insertar un elemento cualquiera en la lista
c) Las dos tienen la misma complejidad asintótica

Respuesta. c) Las dos tienen la misma complejidad asintótica y es $O(n)$

2. Pepito quiere conocer el nombre de todos los productos de una tienda. Pepito diseñó un programa que imprime los elementos de la una lista enlazada. La variable n representa el tamaño de la lista. En el peor de los casos, ¿cuál es la complejidad asintótica para el algoritmo?

Importante Recuerde que el ciclo for each implementa iteradores que permiten btener el siguiente (o el anterior) de una lista enlazada en tiempo constante, es decir, en $O(1)$

```
01 public void listas(LinkedList<String> lista) {
02    for(String nombre: lista)
03       print(nombre); }
```

a) O(n2)
b) O(1)
c) O(n)
d) O(log n)

Respuesta. c. Complejidad $O(n)$

3. En el juego de hot potato (conocido en Colombia como Tingo, Tingo, Tango), los niños hacen un círculo y pasan al vecino de la derecha, un elemento, tan rápido como puedan. En un cierto punto del juego, se detiene el paso del elemento. El niño que queda con el elemento, sale del círculo. El juego continúa hasta que sólo quede un niño.

Este problema se puede simular usando una lista. El algoritmo tiene dos entradas:

Una lista q con los nombres de los niños y una constante entera *num*. El algoritmo retorna el nombre de la última persona que queda en el juego, después de pasar la pelota, en cada ronda, *num* veces.

Como un ejemplo, para el círculo de niños [Bill, David, Susan, Jane, Kent, Brad], donde último es el primer niño, Brad el primer niño, y *num* es igual a 7, la respuesta es Susan.
En Java, el método add agrega un elemento al comienzo de una lista, y el método remove retira

un elemento del final de una lista y retorna el elemento.

```
01 String hotPotato(LinkedList<String> q, int num)
02 while (____________)
03 for (int i = 1; i ____ num; i++)
04 q.add(_________);
05 q.remove();
06 return ___________;
```

A continuación, complete los espacios restantes del código anterior:

a) Complete el espacio de la línea 02
q.size() > 1

b) Complete el espacio de la línea 03
i ≤ num

c) Complete el espacio de la línea 04
q.remove()

d) Complete el espacio de la línea 06
q.remove()

4. El siguiente es un algoritmo invierte una lista usando como estructura auxiliar una pila:

```
01 public static LinkedList <String> invertir (LinkedList <String> lista){
02   Stack <String> auxiliar = new Stack <String>();
03   while(..... > 0){
04     auxiliar.push(lista.removeFirst());
05   }
06   while(auxiliar.size() > 0){
07     ..............
08   }
09   return lista;
10 }
```

De acuerdo a lo siguiente respondan dos preguntas:

a) ¿Qué condición colocaría en el ciclo while de la línea 3? (10%)

q.size() > 0

b) Complete la línea 7 de forma que el algoritmo tenga sentido (10%)

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

lista.add(auxiliar.pop());

5. En un banco, se desea dar prioridad a las personas de edad avanzada. El ingeniero ha propuesto un algoritmo para hacer que la persona de mayor edad en la fila quede en primer lugar.

Para implementar su algoritmo, el ingeniero utiliza colas. Las colas en Java se representan mediante la interfaz *Queue*. Una implementación de *Queue* es la clase *LinkedList*.

El método *offer* inserta en una cola y el método *poll* retira une elemento de una cola.

```
01 public Queue<Integer> organizar( Queue<Integer> personas){
02   int mayorEdad = 0;
03   int edad;
04   Queue<Integer> auxiliar1 = new LinkedList<Integer>();
05   Queue<Integer> auxiliar2 = new LinkedList<Integer>();
06   while(personas.size() > 0){
07     edad = personas.poll();
08     if(edad > mayorEdad) mayorEdad = edad;
09     auxiliar1.offer(edad);
10     auxiliar2.offer(edad);
11   }
12   while(.........){
13     edad = auxiliar1.poll();
14     if(edad == mayorEdad) personas.offer(edad);
15   }
16   while(.........){
17     edad = auxiliar2.poll();
18     if(edad != mayorEdad) ........;
19   }
20   return personas;
21 }
```

a) ¿Que condiciones colocaría en los 2 ciclos while de líneas 12 y 16, respectivamente?

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

b) Complete la línea 18 con el objeto y el llamado a un método, de forma que el algoritmo tenga sentido

personas.offer(edad);

6. ¿Cuál es la complejidad asintótica, para el peor de los casos, de la función procesarCola(q, n)?

```
public void procesarCola(Queue q, int n)
  for (int i = 0; i < n; i++)
    for(int j = 0; j < n; j++)
      q.add(j);
      //Hacer algo en O(1)
```

a) O(n)
b) O($|q|$), donde $|q|$ es el número de elementos de q
c) O(n^2)
d) O(2^n)

Respuesta. c. $O(n^2)$

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