Laboratorio No. 3: Listas Enlazadas y Listas con Arreglos

Nicolás Restrepo López
Universidad Eafit
Medellín, Colombia
nrestrepol@eafit.edu.co

Mateo Restrepo Sierra
Universidad Eafit
Medellín, Colombia
mrestrepos@eafit.edu.co

3) Simulacro de preguntas de sustentación de Proyectos.

1.1. Desarrollen un método que reciba una lista y devuelva la multiplicación de todos los números introducidos en una lista.

Véase Anexo: Laboratorio3.

1.2. Implementen un método que reciba una lista e inserte un dato de modo similar a la función insertar al final de la lista. La diferencia es que ahora no se debe permitir insertar datos repetidos. Si un dato ya está almacenado, entonces la lista no varía, pero tampoco es un error. Llámenlo `SmartInsert(Lista l, int data)`.

Véase Anexo: Laboratorio3

1.3. Escriban un método que reciba una lista y calcule cuál es la posición óptima de la lista para colocar un pivote.

Véase Anexo: Laboratorio3.

1.4. Elaboren un programa que reciba las neveras en la forma en que están ordenadas en el almacén y las solicitudes, según el orden en que llegaron. Debe imprimir qué neveras del almacén quedan asignadas a cada tienda.

Véase Anexo: Laboratorio3.

2.1. Resuelvan el ejercicio planteado usando listas enlazadas.

Véase Anexo: TextoErroneo.

3.1. Completen la siguiente tabla con la complejidad de cada ejercicio para cada tipo de lista.

	ArrayList	LinkedList
Ejercicio 1.1	O(1)	O(n)
Ejercicio 1.2	O(n)	O(1)
Ejercicio 1.3	$O(n^2)$	O(n)
Ejercicio 1.4	No Aplica	$O(m^2 + mn)$

Complejidad de métodos de ArrayList y LinkedList tomado de http://bigocheatsheet.com/. Visitado el 21 de septiembre de 2017.

3.2. Expliquen con sus propias palabras cómo funciona la implementación del ejercicio 2.1.

```
public static void textoErroneo(LinkedList<String> lista) {
   LinkedList<String> lista1 = new LinkedList<String>();
   Scanner scan = new Scanner(System.in);
   System.out.println("Ingrese la linea que se desea corregir");
   String line = scan.nextLine();
   String line1 = line.replaceAll("_", " ");
   String aux = "";
   for (int i = line1.length() - 1; i >= 0; i--){
      aux = line1.charAt(i) + aux;
      if (line.charAt(i) == '[') {
         aux = aux.substring(1, aux.length());
         lista1.add(0, aux);
         aux = "";
      } else if (line.charAt(i) == ']') {
         aux = aux.substring(1, aux.length());
         lista.add(aux);
         aux = "";
      } else if (i == 0)
         lista.add(aux);
   }
   lista.addAll(lista1);
}
//Método que imprime una lista
public static void imprimirLista(LinkedList<String> lista) {
   String salida = "";
   for (int i = 0; i < lista.size(); i++) {
      salida = lista.get(i) + salida;
   }
   System.out.println(salida);
}
```

DOCENTE MAURICIO TORO BERMÚDEZ
Teléfono: (+57) (4) 261 95 00 Ext. 9473. Oficina: 19 - 627
Correo: mtorobe@eafit.edu.co

El ejercicio 2.1 básicamente corrige un texto que está mal escrito por un problema del teclado, ya que se presionan por sí mismas las teclas *Inicio* y *Fin*. Esto hace que el texto quede desorganizado. El algoritmo debe recibir el texto separado por guiones bajos en lugar de espacios. Se sabrá que se presionó la tecla *Inicio* cuando aparezca un corchete de apertura (“[“), y la tecla *Fin* cuando aparezca un corchete de cierre (“]”).

El método recibe como parámetro de entrada una Lista Enlazada que será la lista en la cual se añadirán las cadenas del texto resultante. Lo primero que se hace es crear una segunda Lista Enlazada que servirá como un auxiliar más adelante (línea 2). Luego, mediante la clase *Scanner* creamos la línea correspondiente al texto que será ingresado (líneas 3 a 5). Para mayor facilidad, creamos una nueva línea que elimine los guiones bajos y los reemplace por espacios (línea 6). Luego creamos una cadena auxiliar (línea 7). Posteriormente se crea un ciclo que recorrerá cada carácter de la línea ingresada con espacios, pero desde el final hasta el inicio. La cadena auxiliar servirá para concatenar cada carácter de la línea mientras no haya caracteres ‘[‘ o ‘]’ que indiquen un error del teclado. Luego, existen tres posibilidades:

- El carácter actual es ‘[‘. Para este caso lo primero es retirar el carácter que se insertó previamente en la cadena auxiliar. Luego añadiremos esa cadena al inicio de la Lista Enlazada auxiliar, y se procede a establecer la cadena auxiliar como vacía (líneas 10 a 13).

- El carácter actual es ‘]’. Para este caso procedemos de la misma manera que en el caso anterior, solo que esta vez añadiremos la cadena auxiliar al final de la Lista Enlazada de entrada (líneas 14 a 17).

- El carácter actual es el primero de la línea. Este es el caso antes de salir del ciclo, así que lo que hacemos es añadir la cadena auxiliar al final de la Lista Enlazada de entrada (líneas 18 y 19).

Finalmente, lo único que queda es combinar la lista auxiliar con la lista de entrada para obtener una Lista Enlazada que contenga el texto de salida.

Adicionalmente se tiene el método `imprimirLista` que, como su nombre lo indica, imprimirá los elementos de una Lista Enlazada que recibirá como parámetro. Esto se hará almacenando cada nodo de dicha lista parámetro en una cadena *salida* a través de un ciclo que la recorra (líneas 24 a 30).

Lo único que resta es implementar en el método *main* los dos métodos anteriormente mencionados con una lista que se creará allí mismo.

3.3. Calculen la complejidad del ejercicio realizado en el numeral 2.1 y agréguenla al informe PDF.

La complejidad del algoritmo como tal es O(n). No obstante, debido a que el *main* implementa un segundo método para imprimir los elementos de la lista, su complejidad se vuelve $O(m^2 + n)$, como se muestra a continuación.

```
public static void textoErroneo(LinkedList<String> lista1) {
   LinkedList<String> lista = new LinkedList<String>();          //C1
   Scanner scan = new Scanner(System.in);                        //C2
   System.out.println("Ingrese la linea que se desea corregir"); //C3
   String line = scan.nextLine();                                //C4
   String line1 = line.replaceAll("_", " ");                     //C5
   String aux = "";                                              //C6
   for (int i = line1.length() - 1; i >= 0; i--){                //C7+C8*n
      aux = line1.charAt(i) + aux;                               //C9*n
      if (line.charAt(i) == '[') {                               //C10*n
         aux = aux.substring(1, aux.length());                   //C11*n
         lista.add(0, aux);                                      //C12*n
         aux = "";                                               //C13*n
      } else if (line.charAt(i) == ']') {                        //C14*n
         aux = aux.substring(1, aux.length());                   //C15*n
         lista1.add(aux);                                        //C16*n
         aux = "";                                               //C17*n
      } else if (i == 0)                                         //C18*n
         lista1.add(aux);                                        //C19
   }
   lista1.addAll(lista);                                         //C20
}
//T(n) = C' + C*n
  T(n) = O(n)
public static void imprimirLista(LinkedList<String> lista) {
   String salida = "";                                           //C1
   for (int i = 0; i < lista.size(); i++) {                      //C2+C3*m
      salida = lista.get(i) + salida;                            //C4+m^2
   }
   System.out.println(salida);                                   //C5
}
//T(m) = C' + C*m^2
  T(m) = O(m^2)
public static void main (String [] args) {
   LinkedList<String> lista = new LinkedList<String>();          //C1
   textoErroneo(lista);                                          //O(n)
   imprimirLista(lista);                                         //O(m^2)
}
//La complejidad total será O(n) + O(m^2), es decir O(m^2 + n)
```

DOCENTE MAURICIO TORO BERMÚDEZ
Teléfono: (+57) (4) 261 95 00 Ext. 9473. Oficina: 19 - 627
Correo: mtorobe@eafit.edu.co

3.4. Expliquen con sus palabras qué son las variables m y n del cálculo de complejidad del numeral 3.3.

Como ya se ha observado en los laboratorios realizados anteriormente, las variables del cálculo de las complejidades representan las variables de entrada de los algoritmos. Hacemos referencia al ejercicio anterior, es decir el del texto erróneo, el cual tiene una complejidad de $O(n + m^2)$, la cual se ve claramente en el método *main*, ya que hacer el llamado del método `textoErroneo` es O(n) mientras que `imprimirLista` es $O(m^2)$. Ahora, la variable n refiere el tamaño de la cadena ingresada, es decir el tamaño de la línea que se quiere corregir. Por otro lado, m equivale al tamaño de la lista en la cual está la oración a corregir; vale la pena aclarar que es al cuadrado (m^2), ya que el método `get(i)` es O(m) y está adentro de un ciclo que se hace m veces.

4) Simulacro de Parcial.

1. ¿Cuál operación tiene una mayor complejidad asintótica, 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

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 obtener el siguiente elemento (o el anterior) de una lista enlazada en tiempo constante, es decir, en O(1).

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

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

DOCENTE MAURICIO TORO BERMÚDEZ
Teléfono: (+57) (4) 261 95 00 Ext. 9473. Oficina: 19 - 627
Correo: mtorobe@eafit.edu.co

3. En el juego *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 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.

g.size() > 1

b) Complete el espacio de la línea 03.

<=

c) Complete el espacio de la línea 04.

g.remove()

d) Complete el espacio de la línea 06.

g.remove()

DOCENTE MAURICIO TORO BERMÚDEZ
Teléfono: (+57) (4) 261 95 00 Ext. 9473. Oficina: 19 - 627
Correo: mtorobe@eafit.edu.co