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

Luis Carlos Rodríguez Zúñiga
Universidad Eafit
Medellín, Colombia
lcrodriguz@eafit.edu.co

Laura Sánchez Córdoba
Universidad Eafit
Medellín, Colombia
lsanchezc@eafit.edu.co

3) Simulacro de preguntas de sustentación de Proyectos

1. 1.1 (apto para ambos tipos de listas)

LinkedList:

```
public static int multiply(List<Integer> list) {
    int mult=1;    //c1
    for(int i=0; i<list.size() ;i++){ //c2*n
        mult=mult*list.get(i); //n
    }
    return mult; //c3
}
```

T(n)=c+c2*n*(n)
O(n)=c+c2*n*(n)
O(n)=c2*n*(n) R.S
O(n)=n*(n) R.P
O(n)=n^2

ArrayList:

```
public static int multiply(List<Integer> list) {
    int mult=1;    //c1
    for(int i=0; i<list.size() ;i++){ //c2*n
        mult=mult*list.get(i); //O(1)
    }
```

```
        return mult; //c3
    }
```

T(n)=c+c2*n
O(n)=c+c2*n
O(n)=c2*n R.S
O(n)=n R.P
O(n)=n

1.2 (apto para ambos tipos de listas)

linkedList

```
    public static void smartInsert(List<Integer> list, int data){
        for(int i=0; i<list.size() ;i++){  //c1*n
            if(list.get(i)==data){ //n
                System.out.println("este dato ya se encuentra almacenado"); //c2
                break;                  //c3
            }
            else{
                list.add(data); //n
                System.out.println("se agrego el elmento"); c4
                break; //c5
            }
        }
    }
```

T(n)=c1*n(c+2n)
O(n)=c1*n*(c+2n)
O(n)=n*(c+2n) R.P
O(n)=n*(2n) R.S
O(n)=n*(n) R.P
O(n)=n^2

ArrayList

```
    public static void smartInsert(List<Integer> list, int data){
        for(int i=0; i<list.size() ;i++){  //c1*n
            if(list.get(i)==data){ //O(1)
                System.out.println("este dato ya se encuentra almacenado"); //c2
                break;                  //c3
            }
            else{
```

```
            list.add(data); //n
            System.out.println("se agrego el elmento"); c4
            break; //c5
        }
    }
}
```

T(n)=c1*n(c+n)
O(n)=c1*n*(c+n)
O(n)=n*(c+n) R.P
O(n)=n*(n) R.S
O(n)=n^2

1.3 //Ejercicio 3
//Autor: Brian Morales
//Fecha: Septiembre 2017

(apto para ambos tipos de listas)

linkedList

```
public static int pivote(List<Integer> list){
    int[]sum = suma(list); //c1*n^2
    int ultimo = sum[sum.length - 1]; //c2
    int pivot = 0; //c3
    int minimo = ultimo + 1; //c4
    for(int i = 1; i < sum.length; i++){ //c5*n
        int val = Math.abs(sum[i-1] - (ultimo - sum[i])); //c6
        if(val < minimo){ //c7
            minimo = val; //c8
            pivot = i; //c9
        }
    }
    System.out.println(pivot -1); //c10
    return pivot -1;  //c11
}
```

T(n)= c+c1*n^2+c5*n
O(n)= c+c1*n^2+c5*n
O(n)= c1*n^2+c5*n R.S
O(n)= c1*n^2 R.S
O(n)= n^2 R.P
O(n)= n^2

```
    private static int[] suma(List<Integer> list){
        int sum = 0; // c1
        int[] suma = new int [list.size() + 1]; //c2
        for(int i= 0; i < list.size() ; i++){ // c3*n
            sum+= list.get(i); //n
            suma[i + 1]=sum; //c4
        }
        return suma; //c5
    }
```

T(n)= c+c3*n*(n)
O(n)= c+c3*n*(n)
O(n)= c3*n*(n) R.S
O(n)= n*(n) R.P
O(n)= n^2

ArrayList

```
    public static int pivote(List<Integer> list){
        int[]sum = suma(list); //c1*n
        int ultimo = sum[sum.length - 1]; //c2
        int pivot = 0; //c3
        int minimo = ultimo + 1; //c4
        for(int i = 1; i < sum.length; i++){ //c5*n
            int val = Math.abs(sum[i-1] - (ultimo - sum[i])); //c6
            if(val < minimo){ //c7
                minimo = val; //c8
                pivot = i; //c9
            }
        }
        System.out.println(pivot -1); //c10
        return pivot -1;  //c11
    }
```

T(n)= c+c1*n+c5*n
O(n)= c+c1*n+c5*n
O(n)= c1*n+c5*n R.S
O(n)= n*(c1+c5)
O(n)= n R.P
O(n)= n

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

```
    private static int[] suma(List<Integer> list){
        int sum = 0; // c1
        int[] suma = new int [list.size() + 1]; //c2
        for(int i= 0; i < list.size() ; i++){ // c3*n
            sum+= list.get(i); //O(1)
            suma[i + 1]=sum; //c4
        }
        return suma; //c5
    }
```

T(n)= c+c3*n
O(n)= c+c3*n
O(n)= c3*n* R.S
O(n)= n R.P
O(n)= n

Al no variar mucho del código base, su complejidad es la misma (O(n)) sin embargo, esto solo se tiene en cuenta para el caso de ArrayList, ya que el código base fue diseñado solamente para este tipo de lista y no para LinkedList.

2. El algoritmo funciona de la siguiente manera: el usuario le entra una cadena de caracteres desde la consola, la cual se almacena en la variable *cadena* la cual será recorrida hasta encontrar los caracteres especiales '[' o ']' que indican los cambios en la línea, una vez hallado alguno se procede a las decisiones**,** si el carácter es '[' (comienzo de línea), una variable entera como índice de cambio *cam* se pone en 0 y se va incrementando para que lo que esté después de éste se agregue al inicio de una lista enlazada en la que se almacenan los caracteres en el orden según los digitado por el usuario. Si el caracter es ']' (fin de línea), posicionamos a *cam* en el final de la lista y agregamos allí los caracteres que van después del caracter a medida que *cam* incrementa; si no se halla ninguno de los caracteres especiales, se añade el string igual a como se entró. Finalmente, cuando se recorra la cadena completa, convertimos la lista a un String y posteriormente se llama al método aString() para quitar de lo que se quiere imprimir, algunos caracteres como ' ', ',', '[', ']' ; devolviendo cómo queda realmente el texto en una cadena nueva *aux*.

3.
```
//basado en:
//Ejercicio21
//Autores: Daniel Arango y Kevin Parra
```

```
    //Fecha: Septiembre 2017

    public void cambiar(){
        LinkedList miLista = new LinkedList<String>();   //c1
        Scanner sc = new Scanner(System.in);            //c2
        String cadena;                                  /          /c3
        System.out.print("Introduzca texto: ");       //c4
        cadena = sc.nextLine();                         //c5
        int cam=0;                                      //c6
        for(int i=0; i<cadena.length();i++){ /          /c7*n
            if(cadena.charAt(i)=='['){          //c8
                cam=0;                  //c9
            }else if(cadena.charAt(i)==']'){      //c10
                cam=miLista.size();//c11
            } else {
                miLista.add(cam, cadena.charAt(i));//n
                cam++;                          //c12
            }
        }
        String correcto=miLista.toString(); //n
        System.out.println("Texto corregido: "+aString(correcto)); //n



    }

T(n)=c+n*(c'+c7*n)+2n
O(n)= c+n*(c'+c7*n)+2n
O(n)= n*(c'+c7*n) R.S
O(n)= n*(c7*n) R.S
O(n)= n*(n) R.P
O(n)= n^2



    public String aString(String nueva){
        String aux=""; //c1
        for(int i=0;i<nueva.length();i++){ //c2*n
            if(nueva.charAt(i)!=','&&nueva.charAt(i)!=32)//c3
            if((nueva.charAt(i)!='[')&&(nueva.charAt(i)!=']'))//c4
            aux=aux + nueva.charAt(i);//c5
```

```
        }
        return aux;//c6
    }
```

T(n)=c+c2*n
O(n)= c+c2*n
O(n)= c2*n R.S
O(n)= n R.P

4. Las variables n y m usadas en el cálculo de las complejidades (en el caso del punto previo, solo n), éstas representan el tamaño de la entrada que se da en la entrada del usuario al algoritmo y en función de la cual se mide la eficiencia de éste, pues permite ver la cantidad de tiempo que le tomará realizar cierta cantidad de pasos para emitir la respuesta correspondiente a la entrada.

4) Simulacro de Parcial

1. *c*
2. *c*
3. q.size()>1, <=, q.remove(), q.remove()

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