

Laboratorio Nro. 3

Listas Enlazadas y Vectores Dinámicos

Resumen de ejercicios a resolver

1.1 Implementen un algoritmo que permita crear una estructura de datos con el mapa de una ciudad. Usar el archivo *medellin_colombia-grande.txt* que es el mapa de Medellín.

Se encuentra en github.

2.1 Resuelvan el problema del teclado roto usando listas enlazadas

Se encuentra en github.

3.1 Calculen la complejidad de cada ejercicio 1.x con cada implementación de listas.

#T(n,m) = C1 + T(n) + T(m)

#O(n+m)

3.3 Calculen la complejidad del numeral 2.1.

O(n)

3.4 Expliquen con sus palabras las variables del cálculo de complejidad del numeral 3.3

Porque el algoritmo se repite n veces.

4. Simulacro de Parcial

Resuelto más adelante

ESTRUCTURA DE DATOS 1 Código ST0245

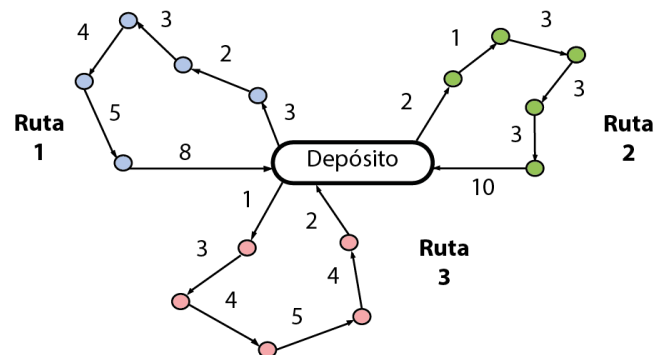
1.1 Los vehículos eléctricos son una de las tecnologías más promisoras para reducir la dependencia del petróleo y las emisiones de gases invernadero. El uso de vehículos eléctricos para carga y para el transporte de pasajeros tiene una limitación:

El rango de conducción es limitado y el tiempo de carga de la batería es relativamente alto. Por esta razón, es necesario considerar que los vehículos se desvíen de la ruta para ir a estaciones donde puedan recargar su batería

Texto tomado de <https://goo.gl/AvWs6B>



Un problema que requiere una urgente solución es cómo encontrar las rutas óptimas para que un conjunto de vehículos eléctricos reparta mercancía a un conjunto de clientes. Dado una lista de clientes ubicados en un mapa vial bidimensional, un depósito de inicio y fin, y restricciones como la autonomía de la batería, la duración máxima de una ruta.



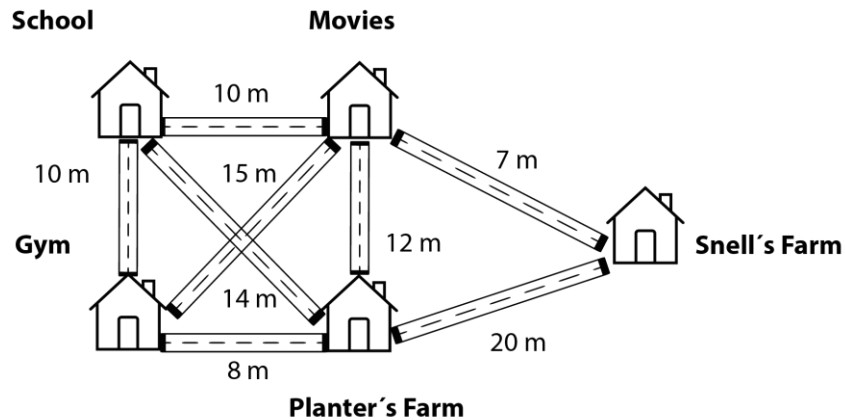
La solución a este problema debe responder la pregunta ¿cuáles son las rutas para una flota de vehículos eléctricos, para visitar todos los clientes de una empresa, minimizando el tiempo total? El tiempo total es la suma del tiempo del recorrido, el tiempo de visitar a los clientes y el tiempo que toman las recargas de batería. **El primer paso solucionar este problema es definir qué estructura de datos se utilizará para representar el mapa de una ciudad.**

► **Implementen un algoritmo que permita crear una estructura de datos con el mapa de una ciudad.**

ESTRUCTURA DE DATOS 1 Código ST0245



Para realizar una prueba, en la carpeta *datasets* encontrarán el archivo *medellin_colombia-grande.txt* que contiene un mapa de la ciudad de Medellín **Ejemplo 1**, para el siguiente mapa, el archivo de entrada es el siguiente:



Vertices. Formato: ID, coordenada x, coordenada y, nombre

```

10000 2.00000 0.00000 School
1 4.00000 1.00000 Movies
2 5.00000 2.00000 Snell
3 2.00000 5.00000 Planters
4 0.00000 2.00000 Gym
  
```

Arcos. Formato: ID, ID, distancia, nombre

```

10000 1 10.0 Calle 1
10000 3 14.0 desconocido
10000 4 10.0 desconocido
1 10000 10.0 Calle 2a
1 2 7.0 desconocido
1 3 12.0 desconocido
1 4 15.0 desconocido
2 1 7.0 desconocido
2 3 20.0 desconocido
  
```

PhD. Mauricio Toro Bermúdez

Docente | Escuela de Ingeniería | Informática y Sistemas

Correo: mtorobe@eafit.edu.co | Oficina: Bloque 19 – 627 Tel:

(+57) (4) 261 95 00 Ext. 9473

**ESTRUCTURA DE DATOS 1 Código
ST0245**

3 10000 14.0 desconocido
3 1 12.0 desconocido
3 2 20.0 desconocido
3 4 8.0 desconocido
4 10000 10.0 desconocido
4 1 15.0 desconocido
4 3 8.0 desconocido

PhD. Mauricio Toro Bermúdez

Docente | Escuela de Ingeniería | Informática y Sistemas
Correo: mtorobe@eafit.edu.co | Oficina: Bloque 19 – 627 Tel:
(+57) (4) 261 95 00 Ext. 9473



2.1

Resuelvan el siguiente ejercicio usando listas enlazadas:

Estás escribiendo un texto largo con un teclado roto. Bueno, no está tan roto. El único problema es que algunas veces la tecla “Inicio” o la tecla “Fin” se presionan solas (internamente).

Usted no es consciente de este problema, ya que está concentrado en el texto y ni siquiera mira el monitor. Luego de que usted termina de escribir puede ver un texto en la pantalla (si decide mirarlo).



Entrada

Habrán varios casos de prueba. Cada caso de prueba es una sola línea conteniendo por lo menos uno y como máximo 100 000 caracteres: letras, guiones bajos, y dos caracteres especiales '[' y ']'. '[' significa que la tecla “Inicio” fue presionada internamente, y ']' significa que la tecla 'Fin' fue presionada internamente. El input se finaliza con un fin-delínea (EOF).



Salida

Por cada caso de prueba imprima la forma en que quedaría el texto teniendo en cuenta que cuando se presiona la tecla 'Inicio' lo manda al principio de la línea, y 'Fin' al final de esta. Asuma que todo el texto está en una sola línea.



Ejemplo de entrada

This_is_a_[Beiju]_text

[[[]]]Happy_Birthday_to_Tsinghua_University
asd[fgh[jkl asd[fgh[jkl[

[[a[[d[f[[g[g[h[h[dgd[fgsfa[f


asd[gfh[[dfh]hgh]fdfhd[dfg[d]g[d]dg




Ejemplo de salida


BeijuThis_is_a__text
Happy_Birthday_to_Tsinghua_University
jklfghasd jklfghasd ffgsfadgdhggfda
dddfgdfhghfhasdhghfdfhgdgdg

- 3.1** Calculen la complejidad de cada ejercicio 1.x 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* (o vectores) y en la otra columna la complejidad de ese ejercicio usando *LinkedList*.


 ¿En qué ejercicios es mejor usar una estructura o la otra? ¿Es alguna de las dos eficiente para este problema o se necesitará otra estructura de datos aún más eficiente?

	<i>ArrayList o Vectores</i>	<i>LinkedList</i>
Ejercicio 1.1		
Ejercicio 1.2 (opt)		
Ejercicio 1.3 (opt)		
... (opt)		

 Si representamos el mapa de Medellín, del **problema 1.1**, con matrices, ¿cuánta memoria consumiría? Tengan en cuenta que hay alrededor de 300,000 vértices

 **En el problema 1.1**, ¿Cómo solucionaron el problema de que los identificadores de los puntos del mapa no empiezan en cero?

Sobre el simulacro de maratón de programación

- 3.3**  Calculen la complejidad del ejercicio realizado en el numeral 2.1

3.4



Expliquen con sus palabras las variables (qué es 'n', qué es 'm', etc.) del cálculo de complejidad del numeral 3.3



Ejemplos de su respuesta:

“n es el número de elementos del arreglo”,
“V es el número de vértices del grafo”, “n es el número de filas de la matriz y m el número de columnas”.

4.1 Cuando enviamos información a través de internet, por ejemplo, del videojuego *Valorant* al servidor de *Riot Games*, lo normal es enviar la información en forma de cadenas de caracteres. Una vez llega al servidor, tenemos que convertir esa información en números. Convertir una cadena de caracteres en un número es un problema –frecuente– en entrevistas de *Snapchat*, *Oracle* y *Uber*, según el portal *Leet Code*. Para este ejercicio, implementa un algoritmo que permita convertir un vector dinámico de caracteres (que contiene un número binario) en un entero decimal. Supongamos que no hay espacios en blanco, el número es positivo, el número es entero y que el vector sólo contiene los caracteres '0' y '1'. Como un ejemplo, para el vector [1,0,1,1] su equivalente en decimal es $1 \times 2^0 + 1 \times 2^1 + 0 \times 2^2 + 1 \times 2^3 = 11$.

Si trabajas en Python, considera este código:

**ESTRUCTURA DE DATOS 1 Código
ST0245**

```

1  def convertir(vector):
2      res = 0
3      for i in range(len(vector)):
4          ....
5      return res

```

En Python, la función `int(c)` convierte el carácter `c` en su valor entero; por ejemplo, convierte '2' en 2. Y el operador `a[i]` retorna el elemento en la posición `i` del vector dinámico `a`.

1. Completa la línea 4 `res = res + vector[i] * math.pow(2, len(vector)-i-1)` Teniendo en cuenta que se importa la librería `math`.

Y ¿Cuál es la complejidad asintótica, en el peor de los casos, del algoritmo anterior? `O(n)`

4.2 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 (o el anterior) de una lista enlazada en tiempo constante, es decir, en `O(1)`.

```

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

```

- a) $O(n^2)$
- b) `O(1)`
- c) $O(n)$
- d) $O(\log n)$

- 4.3** El método `push(i)` ingresa el elemento `i` al tope de la pila. El método `pop()` retira el elemento en el tope de la pila y retorna su valor. Considere el siguiente método:

```

1      void método(Stack<Integer> s){
2      for(int i = 10; i >= 0; i--){
3          if(i % 2 == 0){
4              s.push(i);
5          }
6      }
7      System.out.print(s.pop());
8      while(s.size() > 0){
9          System.out.print(" " + s.pop());
10         }
11     }

```



Eliján la respuesta acertada para cada una de las preguntas:

4.3.1 ¿Cuál es la salida del método anterior?

- i) 10, 8, 6, 5, 4, 2, 0
- ii) 2, 4, 6, 8, 10
- iii) 1, 2, 3, 4, 5, 6, 7, 8, 9, 10
- iv) 0, 2, 4, 6, 8, 10**

4.3.2 ¿Cuál es la complejidad asintótica, en el peor de los casos, de añadir un nuevo elemento al tope una pila que contiene n elementos?

- i) $O(1)$**
- ii) $O(n)$
- iii) $O(n * \log(n))$

iv) $O(n^2)$

4.4 Dijkstra escribió un algoritmo para convertir una *expresión matemática infija* en *notación polaca inversa*, pero se le borraron algunas líneas y no había hecho *commit* en git. La notación polaca inversa evita el uso de paréntesis; para lograrlo primero se colocan los operandos y luego los operadores. Considera los siguientes ejemplos:

- postfix("(5 + 7) * 2") retorna "5 7 + 2 *"
- postfix("5 + 7 / 2") retorna "5 7 2 / +"

```
String postfix(String infix) {
    String ops = "+-*/";
    StringBuilder output = new StringBuilder();
    Deque<String> stack = new LinkedList<>(); for
    (String token : infix.split(" ")) { if
    (ops.contains(token)){//es un operador while (
    ! stack.isEmpty() && isHigherPrec(token,
    stack.peek()))
        output.append(stack.pop()).append(' ');
    stack.push(token);
    } else if (token.equals("(")) { // es un (
    stack.push(token);
    } else if (token.equals(")")) { // es un )
    while ( ! stack.peek().equals("("))
    output.append(stack.pop()).append(' ');
    stack.pop();
    } else { // es un digito
        output.append(.....).append(' ');
    }
    }
    while ( ! stack.isEmpty())
        output.append(stack.pop()).append('
    '); return output.toString(); }
```

La clase `StringBuilder` es una implementación de cadenas de caracteres con listas enlazadas. La clase `Deque` es una implementación de una pila con listas enlazadas. El

**ESTRUCTURA DE DATOS 1 Código
ST0245**

método `isHigherPrec(a,b)` retorna verdadero si *a* tiene mayor precedencia que *b*; de lo contrario, falso. El método `S.append(B)` agrega los caracteres de un `String B` al final del `StringBuilder S`. El método `peek()` permite ver el elemento en el tope de una pila sin retirarlo. El método `pop()` retira un elemento de una pila y lo retorna. El método `push(e)` ingresa el elemento *e* a una pila. El método `S.split(" ")` genera un arreglo de cadenas separando la cadena *S* por espacios. El ciclo `for(String token : infix.split(" "))` se lee como “para cada `String token` en el arreglo `infix.split(" ")`, de izquierda a derecha, haga...” El método `A.contains(b)` retorna verdadero si *b* está en *A*; de lo contrario, falso.

► ¿Cuál es la complejidad asintótica, en el peor de los casos, de la operación `pop()` de una pila?

- a) $O(n^2)$
- b) $O(n)$
- c) $O(1)$
- d) $O(n^2 \times \log n)$

4.5 ¿Cuál es la salida de `mystery()`?

```
void
mystery(){
    Integer[] a = {7,8,3,1,2,1,3,8,9};
    ArrayList<Integer> s = new
ArrayList<>(Arrays.asList(a));
    System.out.println(mystery1(s, a));
}
String mystery1(ArrayList<Integer> s,
                Integer[] a){
    ArrayList<Integer> sol = new ArrayList<>();
    mystery2(sol, s, a); return
sol.toString();
}
void mystery2(ArrayList<Integer> sol,
ArrayList<Integer> s, Integer[] a){
```

```

for(int ele: s){
    if(!sol.contains(ele)){
        sol.add(ele);
    }
}

```



Seleccione la respuesta correcta

- a) [7,8,3,1,2,9]
- b) [7,8,3,1,2,1,3,8,9]
- c) [7,8,3,1,2,1,3]
- d) [3,1,2,1,3]

4.6

2.3

El **add(n)** añade el elemento n al final de la lista. El **get(i)** retorna el elemento en la posición i . El **size()** retorna el tamaño de la lista.

¿Cuál es la complejidad asintótica, en el peor de los casos, de la siguiente función?

```

public void imprimir(int n) {
    if (n == 1) println(1);
    else { println(n);
        imprimir(n-1); }}

void funcion1(LinkedList<Integer>
lista){    for(int i = 0; i < n; i++){
for(int j = 0; j < n; ++j){
    lista.add(i * j);
        }
    }
    for(int i = 0; i < n; i++){
for(int j = 0; j < n; j++){
    print(lista.get(j));
        }
    }
}

```

ESTRUCTURA DE DATOS 1 Código
ST0245



Elija la respuesta que considere acertada:

- a) $O(n^3)$
- b) $O(n^2)$
- c) $O(\log n)$
- d) $O(n)$

**ESTRUCTURA DE DATOS 1 Código
ST0245**

**ESTRUCTURA DE DATOS 1 Código
ST0245**

