ESTRUCTURA DE DATOS 2
Código ST0247

Laboratorio Nro. 5
Dividir para Conquistar y Programación Dinámica

Objetivo: 1. Usar programación dinámica para resolver un problema apropiado. **2.** Para cada una de las estrategias de diseño de algoritmos, identificar un ejemplo práctico en el que se usaría. **3.** Usar un algoritmo de dividir y conquistar para resolver un problema adecuado

Consideraciones: Lean y verifiquen las consideraciones de entrega,

Leer la Guía

Trabajar en Parejas

Si tienen reclamos, regístrenlos en http://bit.ly/2g4TTKf

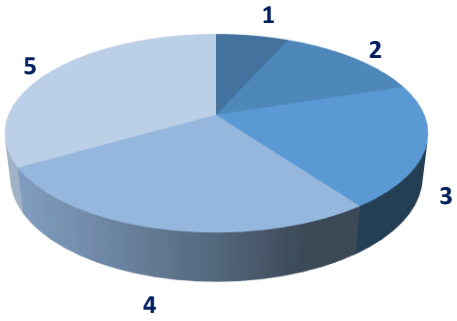
Ver calificaciones en Eafit Interactiva

Subir el informe pdf en la carpeta `informe`, el código del ejercicio 1 en la carpeta `codigo` y el código del 2 en la carpeta `ejercicioEnLinea`

Hoy, plazo de entrega

Si toman la respuesta de **alguna fuente**, deben referenciar según el **tipo de cita**.
Vean *Guía* ***numerales 4.16 y 4.17***

Porcentajes y criterios de evaluación



1. Simulacro sustentación proyecto
2. Análisis de complejidad
3. Código de laboratorio
4. Simulacro de parcial
5. Ejercicio con juez en línea

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

1. Simulacro de Proyecto

Códigos para entregar en GitHub en la carpeta `codigo`

1 Simulacro de Proyecto

Códigos para entregar en GitHub en la carpeta `codigo`:

Vean Guía numeral 3.4

*Código de laboratorio en **GitHub.*** Vean Guía en **numeral 4.24**

Documentación opcional. Si lo hacen, utilicen **Javadoc** o equivalente. No suban el HTML a GitHub.

No se reciben** archivos en **.RAR** ni en **.ZIP

Utilicen Java, C++ o Python

En la vida real, la documentación del software hace parte de muchos estándares de calidad como CMMI e ISO/IEC 9126

En la vida real, el problema del agente viajero se aplica para la construcción de circuitos electrónicos, recolectar monedas de teléfonos de monedas, entre otras. Lean más en http://bit.ly/2i9JdIV

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

1.1 Los vehículos eléctricos son una de las tecnologías más promisorias para reducir la dependencia del petróleo y las emisiones de gases invernadero.

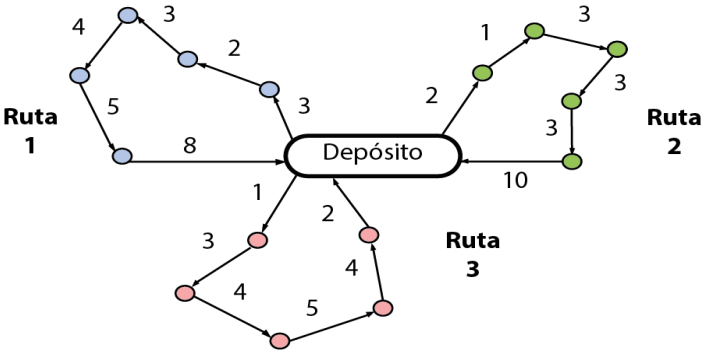
Un ejemplo de carro eléctrico es Kratos, desarrollado por EAFIT. **Foto de https://bit.ly/2SaNotS**

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 el algoritmo de *Held-Karp*, también conocido como algoritmo de programación dinámica para el agente viajero.

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

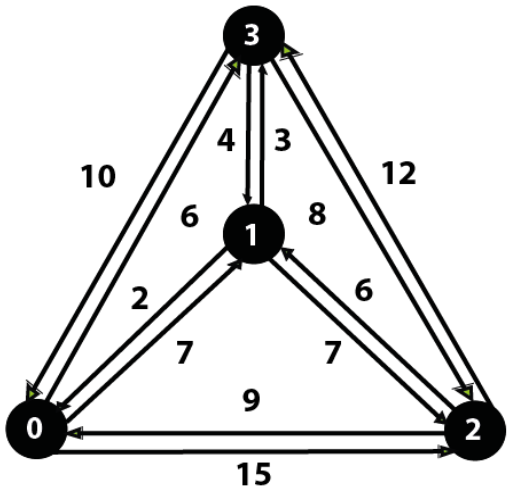
Es el mismo problema que hemos trabajado antes (tanto con bactracking como con la técnica *greedy* del vecino más cercano), solo que esta vez lo resolveremos de manera exacta y mediante la forma más eficiente conocida.

Importante: Para que la implementación sea lo más eficiente posible deben representar los conjuntos utilizando máscaras de bits.

En caso de que tengan problemas con las máscaras de bits puede utilizar la clase *BitmaskSet* que hemos creado para usted, la cual pretende abstraer las máscaras de bits en forma de *Set* de Java (aunque con muchas limitaciones, pero ideal para la implementación de este algoritmo).

Si deciden usarla recuerden leer la documentación.

Por ejemplo, en el siguiente grafo el costo total del recorrido de costo mínimo que pasa por los cuatro vértices (iniciando y volviendo a 0) de 22:



Iniciando en 0, esta es la tabla de ejecución del algoritmo:

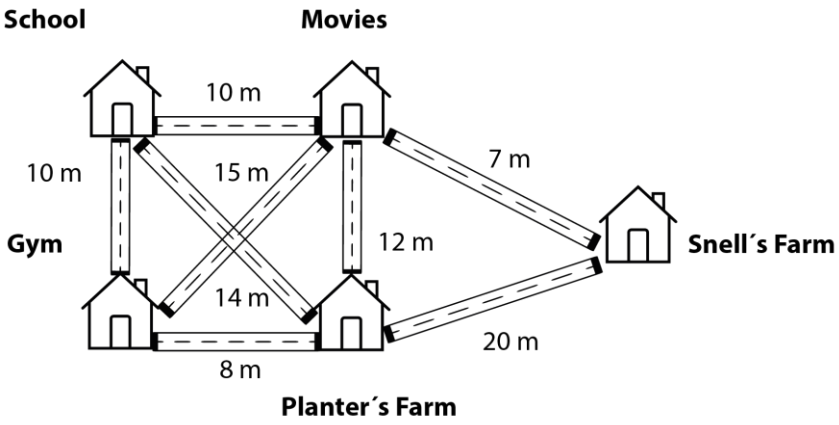
	Costo	Padre
[1, {}]	7	0
[2, {}]	15	0
[3, {}]	6	0
[2, {1}]	14	1
[3, {1}]	10	1

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

[1, {2}]	21	2
[3, {2}]	27	2
[1, {3}]	10	3
[2, {3}]	14	3
[3, {1, 2}]	Min (3 + [1, {2}], 12 + [2, {1}]) = Min (24, 26) = 24	1
[1, {2, 3}]	Min (6 + [2, {3}], 4 + [3, {2}]) = Min (20, 31) = 20	2
[2, {1, 3}]	Min (7 + [1, {3}], 8 + [3, {1}]) = Min (17, 18) = 17	1
[0, {1, 2, 3}]	Min (2 + [1, {2, 3}], 9 + [2, {1, 3}], 10 + [3, {1, 2}]) = 22	1

Reconstruyendo tenemos 0 → 3 → 2 → 1 → 0.

Otro Ejemplo, 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

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

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
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

En la vida real, la distancia de Levenshtein se utiliza para algoritmos de reconocimiento óptico de caracteres, es decir, pasar de imagen a texto. También se utiliza en correctores de ortografía, como el que trae Microsoft Word y en especial el de los teclados de los celulares, al igual que en procesamiento del lenguaje natural como Siri de Apple.

[Opc] Dadas dos cadenas de caracteres *a* y *b*, determinen la distancia Levenshtein que hay entre ellas, es decir, la cantidad mínima de operaciones (insertar, remover o cambiar una letra) que se necesitan para transformar una en la otra **utilizando programación dinámica.**

Por ejemplo, si tenemos las palabras “carro” y “casa” la distancia Levenshtein que hay entre ellas es 3:

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

1. Remover una letra: “carr”
2. Cambiar una letra: “casr”
3. Cambiar una letra: “casa”

Noten que las operaciones y su orden pueden ser diferentes, pero lo importante es que el número mínimo de operaciones para transformar “carro” en “casa” son 3.

Asuman que las cadenas dadas están ambas completamente en minúscula o mayúscula.

En la vida real, la solución de programación dinámica del problema de la subsecuencia común más larga es utilizado en la implementación del comando *diff*, para comparación de archivos, disponible en sistemas Unix. También tiene muchas aplicaciones en bioinformática.

[Opc] El problema de la subsecuencia común más larga es el siguiente: Dadas dos secuencias, encontrar la longitud de la secuencia más larga presente en ambas. Una subsecuencia es una secuencia que aparece en el mismo orden relativo, pero no necesariamente de forma contigua.

Como un ejemplo, “abc”, “abg”, “bdf”, “aeg” y “acefg” son subsecuencias de “abcdefg”. Entonces, para una cadena de longitud n existen 2^n posibles subsecuencias.

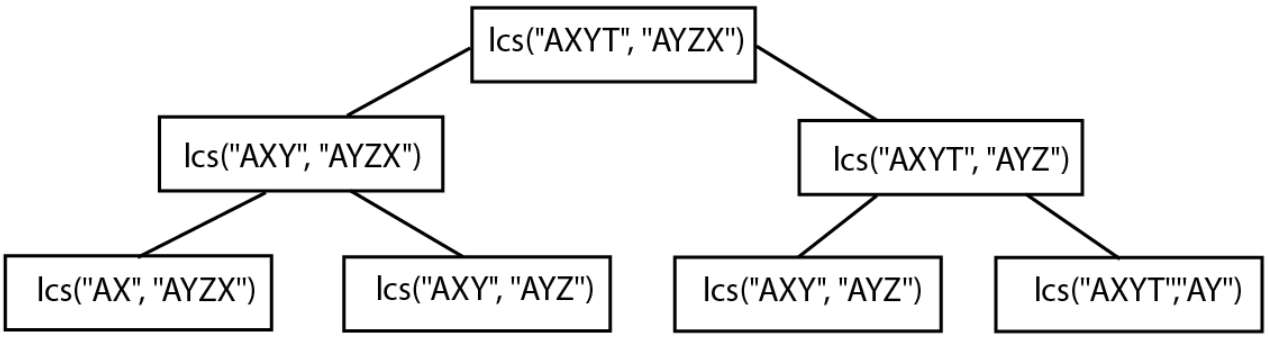
[Opc] Modifiquen el método del **ejercicio 1.3** para retornar, no la longitud de la subsecuencia común más larga, sino la subsecuencia común más larga (es decir, retornar *String* en lugar de *int*)

Como un ejemplo, consideren los siguientes ejemplos para el problema:

Para “ABCDGH” y “AEDFHR” es “ADH” y su longitud es 3. Para “AGGTAB” y “GXTXAYB” es “GTAB” y su longitud es 4.

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

Una forma de resolver este problema es usando *backtracking*, como un ejemplo, para las cadenas “AXYT” y “AYZX”, dada una función recursiva `lcs` que resuelve el problema, se obtendría el siguiente árbol (parcial) de recursión:



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. Simulacro de Maratón de Programación sin documentación HTML, en GitHub,

en la carpeta `ejercicioEnLinea`

2 Simulacro de Maratón de Programación sin documentación HTML, en GitHub, en la carpeta `ejercicioEnLinea`

Vean Guía numeral 3.3

No entregar documentación **HTML**

Utilicen Java, C++ o Python

No se reciben *archivos en* ***.PDF***

No se reciben *archivos en* ***.RAR*** *ni en* ***.ZIP***

Código del ejercicio en línea en ***GitHub.*** Vean Guía en **numeral 4.24**

2.1 Resuelvan el siguiente problema usando programación dinámica:

Karolina es un robot que vive en un sistema rectangular de coordenadas donde cada lugar está designado por un conjunto de coordenadas enteras (*x* y *y*).

Su trabajo es diseñar un programa que ayudará a Karolina a escoger un número de desechos radioactivos que están ubicados en su mundo. Para hacerlo, ustedes deben dirigir a Karolina a la posición en que se encuentra cada uno de los desechos radioactivos.

Encuentre la longitud del camino más corto que llevará a Karolina desde su posición inicial, a cada uno de los desechos radioactivos, y retornar a la posición inicial.

Karolina se puede mover en los ojos *x* y *y*, **nunca en diagonal**. Karolina se puede mover de una posición (i,j) a una posición adyacente (i,j+1), (i+1,j), (i-1,j) o (i,j-1) con costo de 1.

Ustedes pueden asumir que el mundo de Karolina nunca es más grande de 20 x 20 cuadrados y que nunca habrá más de 10 desechos radioactivos para recoger.

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

Cada coordenada es un par (x,y) donde cada valor está en el rango de 1 al rango máximo de la coordenada respectiva.

Entrada

Primero, habrá una línea que tiene el número de escenarios que le piden que ayuda a Karolina a resolver. Para cada escenario habrá una línea con el tamaño del mundo, dado como dos enteros que representan, respectivamente, el *x* y el *y*.

Después, habrá una línea con dos números que es la posición inicial de Karolina. En la siguiente línea estará el número de desechos radioactivos. Para cada desecho radioactivo, habrá una línea con dos números que representan las coordenadas de cada desecho radioactivo.

Salida

La salida será una sola línea por cada escenario, entregando la distancia mínima que Karolina tiene que moverse para recoger todos los desechos, desde su posición inicial, y regresar, de nuevo, a la posición inicial.

Ejemplos de las entradas

```
1
10 10
1 1
4
2 3
5 5
9 4
6 5
```

Ejemplos de la salida

```
The shortest path has length 24
```

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

[Opc] Resuelvan los siguientes ejercicios

2.2 https://goo.gl/bk3zuu

2.3 https://goo.gl/N2LnRY

2.4 https://goo.gl/9jzdx2

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

3. Simulacro de preguntas de sustentación de Proyecto en la carpeta informe

3 Simulacro de preguntas de sustentación de Proyecto en la carpeta informe

Vean **Guía** numeral 3.4

Exportar y entregar informe de laboratorio en **PDF, en español o Inglés**

Si hacen el ***informe en español***, usen la **plantilla en español**

Si hacen **el informe en inglés,** usen a **plantilla en inglés**

No apliquen **Normas Icontec** para esto

Sobre el Simulacro del Proyecto

3.1 

Expliquen con sus propias palabras la estructura de datos que utilizan para resolver el problema del numeral 1.1 y cómo funciona el algoritmo.

Nota: Recuerden que deben explicar su implementación en el informe PDF

3.2 

Para resolver el problema del agente viajero, la solución óptima más eficiente que se conoce es la de programación dinámica, desafortunadamente, no es aplicable para un número grande de vértices.

¿Cuántas operaciones habría que realizar, aproximadamente, para resolver este problema en un grafo con 50 vértices?

3.3 

Expliquen con sus propias palabras la estructura de datos que utiliza para resolver el problema del numeral 2.1 y cómo funciona el algoritmo

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

¿El objetivo sería pasar por todos los vértices del grafo o sólo por los puntos donde hay que entregar un domicilio? ¿Cómo calcular la distancia que hay entre los puntos donde hay que entregar un domicilio?

Sobre el simulacro de maratón de programación

3.4 Expliquen con sus propias palabras la estructura de datos que utiliza para resolver el problema del numeral 2.1 y cómo funciona el algoritmo.

3.5 Calculen la complejidad de los ejercicios en línea del numeral 2.1 y agréguenla al informe PDF

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

Ejemplo de esta 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'.

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

4. Simulacro de parcial en informe PDF

4 Simulacro de Parcial en el informe PDF

Resuelvan los ejercicios

Para este simulacro, agreguen ***sus respuestas*** **en el informe PDF.**

El ***día del Parcial no tendrán computador, JAVA o acceso a internet.***

Si hacen el ***informe en español***, usen la **plantilla en español**

Exportar y entregar informe de laboratorio en **PDF, en español o inglés**

Si hacen **el informe en inglés,** usen a **plantilla en inglés**

No apliquen **Normas Icontec** para esto

4.1 [Opc]

La distancia de Levenshtein es el número mínimo de operaciones requeridas para transformar una cadena de caracteres en otra. Se entiende por operación, una inserción, eliminación o la sustitución de un caracter.

Es útil en programas que determinan cuan similares son dos cadenas de caracteres, como es el caso de los correctores de ortografía.

Como un ejemplo, la distancia de Levenshtein entre \casa" y \calle" es de 3 porque se necesitan al menos tres ediciones elementales para cambiar uno en el otro:

- Casa a Cala (sustitución de 's' por 'l')
- Cala a Calla (inserción de 'l' entre 'l' y 'a')
- Calla a Calle (sustitución de 'a' por 'e')

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

A continuación, esta una implementación del algoritmo en Java:

```
int minimum(int a, int b, int c) {
   return Math.min(Math.min(a, b), c);
}


int Levenshtein(String lhs, String rhs) {
   int[][] distance = new int[lhs.length() + 1][rhs.length()
+1];
   for (int i = 0; i <= lhs.length(); i++)
       distance[i][0] = i;
   for (int j = 1; j <= rhs.length(); j++)
       distance[0][j] = j;
   for (int i = 1; i <= lhs.length(); i++)
       for (int j = 1; j <= rhs.length(); j++)
            distance[i][j] = minimum(distance[i - 1][j] + 1,
            distance[i][j - 1] + 1,distance[i - 1][j - 1] +
           ((lhs.charAt(i - 1) == rhs.charAt(j - 1)) ? 0 : 1));
   return distance[lhs.length()][rhs.length()];
}
```

4.1.1 Completen la siguiente tabla, siguiendo el algoritmo de programación dinámica de la distancia de Levenshtein:

		c	a	l	l	e
c						
a						
s						
a						

4.1.2 Completen la siguiente tabla siguiendo el algoritmo de programación dinámica de la distancia de Levenshtein para encontrar la distancia entre madre y mama:

		m	a	d	r	e
m						
a						
m						
a						

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

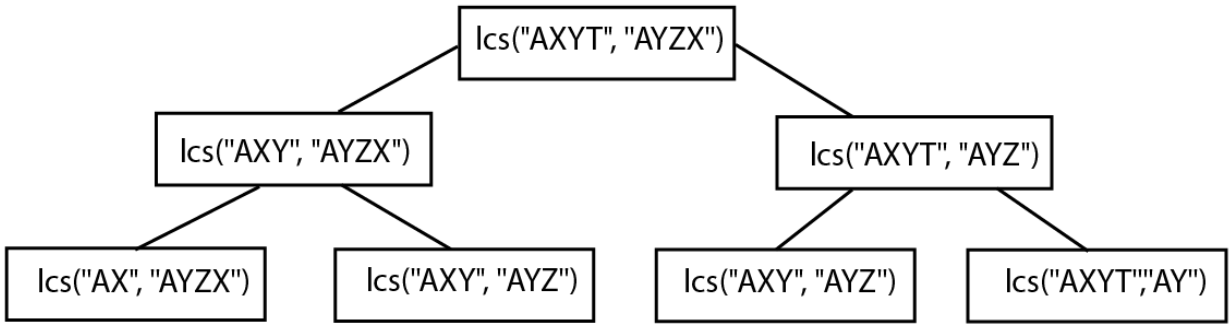
4.2 El problema de la subsecuencia común más larga es el siguiente: Dadas dos secuencias, encontrar la longitud de la secuencia más larga presente en ambas. Una subsecuencia es una secuencia que aparece en el mismo orden relativo, pero no necesariamente de forma contigua.

Como un ejemplo, “abc”, “abg”, “bdf”, “aeg” y “acefg” son subsecuencias de “abcdefg”. Entonces, para una cadena de longitud n existen 2n posibles subsecuencias.

Este problema es utilizado en la implementación del comando diff, para comparación de archivos, disponible en sistemas Unix. También tiene muchas aplicaciones en bioinformática

Considere los siguientes ejemplos para el problema: Para “ABCDGH” y “AEDFHR” es “ADH” y su longitud es 3. Para “AGGTAB” y “GXTXAYB” es “GTAB” y su longitud es 4.

Una forma de resolver este problema es usando backtracking, como un ejemplo, para las cadenas “AXYT” y “AYZX”, dada una función recursiva lcs que resuelve el problema, se obtendría el siguiente árbol (parcial) de recursión:



Usando backtracking para ese problema, el problema lcs(“AXY”, “AYZ”) se resuelve dos veces. Si dibujamos el árbol de recursión completo, veremos que aparecen más y más problemas repetidos, así como en el caso de serie de Fibonacci.

Este problema se puede solucionar guardando la soluciones, que ya se han calculado para los subproblemas, en una tabla; es decir, usando programación dinámica, como en el algoritmo siguiente:

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

```
01 // Precondición: Ambas cadenas x, y son no vacías
02 public static int lcsdyn(String x, String y) {
03    int i,j;
04    int lenx = x.length();
05    int leny = y.length();
06    int[][] table = new int[lenx+1][leny+1];
07
08    // Inicializa la tabla para guardar los prefijos
09    // Esta inicialización para las cadenas vacías
10    for (i=0; i<=lenx; i++)
11        table[i][0] = 0;
12     for (i=0; i<=leny; i++)
13        table[0][i] = 0;
14
15    // Llena cada valor de arriba a abajo
16    // y de izquierda a derecha
17    for (i = 1; i<=lenx; i++) {
18    for (j = 1; j<=leny; j++) {
19        // Si el último caracter es igual
20        if (x.charAt(i-1) == y.charAt(j-1))
21            table[i][j] = 1+table[i-1][j-1];
22       // De lo contrario, tome el máximo
23       else // de los adyacentes
24           table[i][j] = Math.max(table[i][j-1],table[i-
1][j]);
25       System.out.print(table[i][j]+" ");
26     }
27     System.out.println();
28  }
29
30 // Retornar la respuesta (tipo entero, ojo)
31 ____________________________________
32 }
```

4.2.1 ¿Cuál es la complejidad asintótica para el peor de los casos? **Ojo:** *n* no es una variable en este problema

_____

4.2.2 Completen la línea 31

_____

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

4.3 **[Opc]** La serie de Fibonacci esta dada por *1,1,2,3,5,8,13,...*

La complejidad de calcular el elemento n-esimo de la serie de Fibonacci, usando el siguiente algoritmo, es...

```
public int fibo(int n) {
    int[] fibos = new int[n+1];
    for (int i = 0; i <= n; i++) {
         if (i<=1)
             fibos[i] = i;
         else
             fibos[i] = fibos[i-1]+fibos[i-2];
     }
    return fibos[n];
}
```

4.3.1 Elijan la opción que consideren acertada:

a) O (n)
b) $O(n^2)$
c) $O(2^n)$
d) $O(3^n)$
e) O(n!)

4.3.2 Elijan la opción que consideren acertada:

a) T(n) = c1:n + c2
d) T(n) = c1:n2 + c2:n
c) T(n) = c12n + c2
d) T(n) = c13n + c2:n
e) T(n) = c13n + c2:n!

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

4.4 **[Opc]** Seleccionen la afirmación que consideren acertada:

```
public int fibo(int n) {
   if (n <= 1) return n;
   else return fibo(n-1)+fibo(n-2);
}
```

a) O(n) y se optimiza con programación estática
b) O(n^2) y se optimiza con programación dinámica
c) O(2^n) y se optimiza con programación dinámica
d) O(3^n) y se optimiza con programación dinámica
e) Ninguna de las anteriores

4.5 Consideren una secuencia de números *a*: Para esta secuencia siempre se cumple que, es decir, esta secuencia está ordenada de menor a mayor (siendo el menor elemento de *a* y el mayor elemento de *a*).

Ahora, consideren un entero *z*. Queremos encontrar, en la secuencia, el mayor número que es menor o igual a *z*. La solución a este problema simplemente es hacer **búsqueda binaria** en la secuencia, teniendo en cuenta, en la búsqueda, que el elemento actual sea menor o igual al elemento examinado.

Ayúdennos a terminar el siguiente código.

Ejemplo: Sea *a* = {3, 7, 11, 12, 23, 24, 27, 77, 178} y *z* = 45. La respuesta es 27.

```
01  int bus(int[]a,int iz,int de,int z){
02    if(iz>de){
03      return -1;       }
04    if(a[de]>=z){
05      return a[de];    }
06    int mitad=(iz+de)/2;
07    if(a[mitad]==z){
08    return .........; }
09    if(mitad>0){
10      if(a[mitad-1]<=z && z<a[mitad]){
11        return a[mitad - 1];      }    }
```

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

```
12    if(z<a[mitad]){
13      return bus(a,iz,mitad-1,z); }
14    //else
15    return bus(...., ...., ...., ....);
16  }
17  public int bus(int[] a, int z) {
18     return bus(a, 0, a.length - 1, z);
19  }
```

De acuerdo a lo anterior:

4.5.1 ¿Cuál es la complejidad asintótica, en el peor de los casos, del algoritmo anterior?

a) $T(n)=2.T(n/2)+C$ que es $O(n)$
b) $T(n)=2.T(n/2)+Cn$ que es $O(n.\log n)$
c) $T(n)=T(n/2)+C$ que es $O(\log n)$
d) $T(n)=4.T(n/2)+C$ que es

4.5.2 Completen la línea 8 _____

4.5.3 Completen la línea 15_____

4.6 Dado un conjunto de enteros, queremos encontrar su **subsecuencia creciente máxima**. La subsecuencia creciente máxima de una secuencia, es una subsecuencia de esta secuencia cuyos elementos están ordenados y esta subsecuencia es tan larga como sea posible.

Los elementos de esta subsecuencia no necesariamente tienen que ser contiguos en la secuencia original. Este problema tiene aplicaciones en bioinformática para análisis de nicho ecológico y para filogenética. **Nota:** Vean más adelante algunos ejemplos.

En este problema estamos interesados en encontrar el tamaño de la subsecuencia creciente máxima. La ecuación recursiva está definida de esta forma:

El siguiente código hace el trabajo, utilizando programación dinámica, pero faltan algunas líneas, complétalas por favor.

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

```
01 int scm(int arr[]) {
02   int n = arr.length;
03   int scm[] = new int[n];
04   int i,j,max = 0;
05   /* Inicializar la tabla scm */
06   for ( i = 0; i < n; i++ ) {
07       ...........; }
08   /* Calcular usando la tabla scm*/
09   for ( i = 1; i < n; i++ ) {
10       for ( j = 0; j < i; j++ ) {
11         if ( arr[i] > arr[j] && scm[i] < scm[j] + 1) {
12           ...........; } } }
13   /* El maximo en la tabla scm */
14   for ( i = 0; i < n; i++ ) {
15       if ( max < scm[i] ) {
16          ...........;  } }
17   return max;
```

Ejemplo 1: Sea *a* = {7, 4, 8, 1, 3, 9, 2, 6, 10}. La subsecuencia creciente máxima es **{1, 2, 6, 10}**.

Noten que **{1, 3, 6, 10}** es otra subsecuencia creciente máxima. En todo caso, la longitud de la secuencia creciente máxima es 4.

Ejemplo 2: Sea *a* = {4, 2, 1, 9, 3, 4, 10, 2, 1}. La subsecuencia creciente máxima es **{1, 3, 4, 10}**. La longitud de la secuencia creciente máxima es 4.

De acuerdo a lo anterior:

4.6.1 Completen la línea 7 _____

4.6.2 Completen la línea 12 _____

4.6.3 Completen la línea 16 _____

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

4.6.4 %) ¿Cuál es la complejidad asintótica, en el peor de los casos, del método anterior?

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

4.7 El **algoritmo de Floyd-Warshall** encuentra la distancia mínima entre cada par de vértices *i*,*j* de un grafo con pesos.Este algoritmo muchas veces es usado para determinar la clausura transitiva de un grafo, que es útil en Lenguajes Formales y Compiladores.

Dada la representación usando **matrices de adyacencia** *g* de un grafo, donde la posición *i*,*j* de la matriz es ∞ si el vértice *i* no está conectado con el vértice *j* o la posición *i*,*j* es un entero si existe una conexión entre el vértice *i* y *j*, queremos encontrar la distancia mínima entre el vértice *u* y el vértice *v*.

Ayúdennos a completar el siguiente código:

```
int calcular(int[][]g,int u,int v){
  int n = g.length;
  int[][] d = new int[n][n];
  for(int i = 0; i < n; ++i){
     for(int j = 0; j < n; ++j){
       d[i][j] = g[i][j];
     }
  }
  for(int k = 0; k < n; ++k){
     for(int i = 0; i < n; ++i){
       for(int j = 0; j < n; ++j){
         int ni = .......;
         int nj = .......;
         int nk = .......;
         int res = Math.min(ni, nj + nk);
         d[i][j] = res;
       }
     }
  }
```

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

```
  return d[u][v];
}
```

De acuerdo a lo anterior:

4.7.1 Completen la línea 12 _____

4.7.2 Completen la línea 13 _____

4.7.3 Completen la línea 14 _____

4.7.4 ¿Cuál es la complejidad asintótica, en el peor de los casos, del algoritmo anterior?

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

5. [Opcional]
Lecturas Recomendadas

5 [Opc] Lecturas recomendadas

Vean Guía en ***numeral 3.5 y 4.20***

Exportar y entregar informe de laboratorio en **PDF, en español o Inglés**

Si hacen el ***informe en español***, usen la **plantilla en español**

Si hacen **el informe en inglés,** usen a **plantilla en inglés**

No apliquen **Normas Icontec** para esto

> **"El ejercicio de una profesión requiere la adquisición de competencias que se sustentan en procesos comunicativos. Así cuando se entrevista a un ingeniero recién egresado para un empleo, una buena parte de sus posibilidades radica en su capacidad de comunicación; pero se ha observado que esta es una de sus principales debilidades…" *Tomado de* http://bit.ly/2gJKzJD**

Lean a del ***"R.C.T Lee et al., Introducción al análisis y diseño de Algoritmos. Capítulo 7: Programación Dinámica.",*** y sumen puntos adicionales, así:

Hagan un mapa conceptual con los principales elementos teóricos.

Si desean una lectura adicional en español, consideren la siguiente ***"John Hopcroft et al., Estructuras de Datos y Algoritmos, Sección 10.3. 1983"***, que encuentran en biblioteca

Nota: Estas respuestas también deben incluirlas en el informe PDF

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

6. [Opcional] Trabajo en Equipo y Progreso Gradual

6 [Opc] Trabajo en Equipo y Progreso Gradual

Vean Guía en ***numeral 3.6, 4.21, 4.22*** *y* ***4.23***

Exportar y entregar informe de laboratorio en **PDF, en español o Inglés**

Si hacen el ***informe en español***, usen la **plantilla en español**

Si hacen **el informe en inglés,** usen a **plantilla en inglés**

No apliquen **Normas Icontec** para esto

El trabajo en equipo es imprescindible. "Algunos medios retratan la programación como un trabajo solitario, la realidad es que requiere de mucha comunicación y trabajo con otros. En la vida laboral serás parte de un equipo de trabajo y deberás comunicarte con otras personas". *Tomado de* http://bit.ly/2qJKzJD

6.1 Entreguen copia de todas las actas de reunión usando el *tablero Kanban*, con fecha, hora e integrantes que participaron

6.2 Entreguen el reporte de *git* con los cambios en el código y quién hizo cada cambio, con fecha, hora e integrantes que participaron

6.3 Entreguen el reporte de cambios del informe de laboratorio que se genera *Google docs* o herramientas similares

NOTA: Estas respuestas también deben incluirlas en el informe PDF

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

7. [Opcional] Laboratorio en Inglés con plantilla en Inglés

7 [Opc] Laboratorio en inglés

*Vean Guía en **numeral 3.6, 4.21, 4.22** y **4.23***

Exportar y entregar informe de laboratorio en **PDF, en español o Inglés**

Si hacen el ***informe en español***, usen la **plantilla en español**

Si hacen **el informe en inglés,** usen a **plantilla en inglés**

No apliquen **Normas Icontec** para esto

El inglés es un idioma importante en la Ingeniería de Sistemas porque la mayoría de los avances en tecnología se publican en este idioma y la traducción, usualmente se demora un tiempo.

Adicionalmente, dominar el inglés permite conseguir trabajos en el exterior que son muy bien remunerados. *Tomado de [goo.gl/4s3LmZ](goo.gl/4s3LmZ)*

Entreguen el código y el informe en inglés.

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

Resumen de ejercicios a resolver

1.1 Implementen el algoritmo de *Held-Karp*, también conocido como algoritmo de programación dinámica para el agente viajero.

1.2 [Ejercicio Opcional] Dadas dos cadenas de caracteres *a* y *b*, determinen la distancia Levenshtein que hay entre ellas, es decir, la cantidad mínima de operaciones (insertar, remover o cambiar una letra) que se necesitan para transformar una en la otra **utilizando programación dinámica.**

1.3 [Ejercicio Opcional] El problema de la subsecuencia común más larga es el siguiente: Dadas dos secuencias, encontrar la longitud de la secuencia más larga presente en ambas. Una subsecuencia es una secuencia que aparece en el mismo orden relativo, pero no necesariamente de forma contigua.

1.4 [Ejercicio Opcional] Modifiquen el método del ejercicio 1.3 para retornar, no la longitud de la subsecuencia común más larga, sino la subsecuencia común más larga (es decir, retornar *String* en lugar de *int*)

2.1. Resuelvan el problema usando programación dinámica

2.2. [Ejercicio Opcional] Resuelvan el siguiente problema **https://goo.gl/bk3zuu**

2.3. [Ejercicio Opcional] Resuelvan el siguiente problema **https://goo.gl/N2LnRY**

2.4. [Ejercicio Opcional] Resuelvan el siguiente problema **https://goo.gl/9jzdx2**

3.1 Expliquen con sus propias palabras la estructura de datos que utilizan para resolver el problema del numeral 1.1 y cómo funciona el algoritmo.

3.2 Para resolver el problema del agente viajero, la solución óptima más eficiente que se conoce es la de programación dinámica, desafortunadamente, no es aplicable para un número grande de vértices. ¿Cuántas operaciones habría que realizar, aproximadamente, para resolver este problema en un grafo con 50 vértices?

3.3 Expliquen con sus propias palabras la estructura de datos que utiliza para resolver el problema del numeral 2.1 y cómo funciona el algoritmo

3.4 Expliquen con sus propias palabras la estructura de datos que utiliza para resolver el problema del numeral 2.1 y cómo funciona el algoritmo.

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

3.5 Calculen la complejidad de los ejercicios en línea del numeral 2.1 y agréguenla al informe PDF
3.6 Expliquen con sus palabras las variables (*qué es 'n', qué es 'm', etc.*) del cálculo de complejidad del numeral anterior

4. Simulacro de Parcial

5. [Ejercicio Opcional] Lectura recomendada

6. [Ejercicio Opcional]Trabajo en Equipo y Progreso Gradual

7. [Ejercicio Opcional] Entreguen el código y el informe traducido al inglés.

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

Ayudas para resolver los Ejercicios

Ayudas para el Ejercicio 1

Pista: Vean Guía en numeral 4.1

Ayudas para el Ejercicio 1.1

Pista 1:

```
public static int heldKarp(Digraph g) {
     // complete...
}
```

Pista 2: Si tienen problemas para la implementación del algoritmo le recomendamos el siguiente video: **http://bit.ly/2kqN9dz** . Recuerden activar los subtítulos (en inglés) en caso de que los necesite.

Ayudas para el Ejercicio 1.2

Pista 1:

```
public static int levenshtein(String a, String b) {
     // complete...
}
```

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

Pista 2: Como recomendación, solucionen el siguiente problema para tener una mayor seguridad de que su implementación es correcta: **http://www.spoj.com/problems/EDIST/**

Pista 3: Utilicen el siguiente simulador **http://www.let.rug.nl/kleiweg/lev/**

Ayudas para el Ejercicio 1.4

Pista 1: Usando backtracking para ese problema, el problema lcs("AXY", "AYZ") se resuelve dos veces. Si dibujamos el árbol de recursión completo, veremos que aparecen más y más problemas repetidos, así como en el caso de serie de Fibonacci.

Este problema se puede solucionar guardando la soluciones, que ya se han calculado para los subproblemas, en una tabla; es decir, usando programación dinámica.

Pista 2: Completen el siguiente método

```
// Precondición: Ambas cadenas x, y son no vacías
public static int lcsdyn(String x, String y) {
   …
}
```

Ayudas para el Ejercicio 3.2

Pista: Lean **http://bit.ly/2xjXZs1**

Ayudas para el Ejercicio 3.4

Pista: Vean Guía en numeral 4.11

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

Ayudas para el Ejercicio 4.0

Pista: Vean Guía en numeral 4.18

Ayudas para el Ejercicio 5

Pista: Para que hagan el mapa conceptual se recomiendan herramientas como las que encuentran en ***https://cacoo.com/*** o ***https://bit.ly/2QM9D3M***

Ayudas para el Ejercicio 6.1

Pista: *Vean Guía en numeral 4.21*

Ayudas para el Ejercicio 6.2

Pista: Vean *Guía en numeral 4.23*

Ayudas para el Ejercicio 6.3

Pista: Vean *Guía en numeral 4.22*

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

¿Alguna inquietud?

CONTACTO

Docente Mauricio Toro Bermúdez
Teléfono: (+57) (4) 261 95 00 Ext. 9473
Correo: mtorobe@eafit.edu.co
Oficina: 19- 627

Agenden una cita dando clic en la pestaña -*Semana*- de http://bit.ly/2gzVg10