### Laboratorio Nro. 1 Recursión



**Objetivos:** 1. Identificar el caso base y el general de un problema definido recursivamente. 2. Usar ecuaciones de recurrencia para determinar la complejidad en tiempo y espacio de algoritmos definidos de forma recursiva. 3. Usar la notación O para encontrar formalmente la complejidad asintótica en espacio y memoria de algoritmos



Consideraciones: Lean y verifiquen las consideraciones de entrega,



Leer la Guía



Trabajo en Parejas



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



Ver
calificaciones
en Eafit
Interactiva



En el GitHub docente, encontrarán la traducción de los Ejercicios en Línea

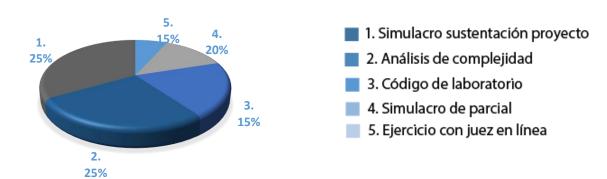


Hoy, plazo de entrega



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

### Porcentajes y criterios de evaluación



PhD. Mauricio Toro Bermúdez





# 1. Simulacro de Proyecto

Códigos para entregar en GitHub en la carpeta codigo

# Simulacro de Proyecto Códigos para entregar en GitHub en la carpeta codigo



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



Vean Guía numeral 3.4



Código del ejercicio en línea 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** 







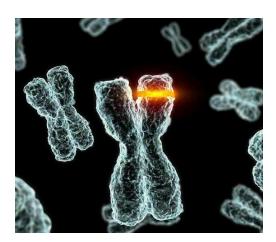




Utilicen Java, C++ o Python

El genoma humano fue decodificado en su totalidad en el 2003. Desde ese entonces, una de las grandes líneas de investigación en Bioinformática es encontrar, en el genoma humano, las secuencias de ADN responsables de la aparición de diferentes tipos de cáncer.

Las cadenas de ADN son, simplemente, cadenas de caracteres. El problema de encontrar un patrón cancerígeno en el ADN de una persona, en algunos casos, puede verse como el problema de la subsecuencia común más larga.



#### PhD. Mauricio Toro Bermúdez







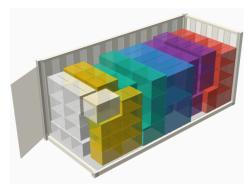
El problema de la subsecuencia común más larga es el siguiente: Dadas dos secuencias, encontrar la longitud de la secuencia más larga presente en ambas. Una subsecuencia es una secuencia que aparece en el mismo orden relativo, pero no necesariamente de forma contigua.



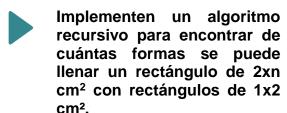
Implementen un algoritmo que calcule la longitud de la subsecuencia común más larga a dos cadenas de caracteres.

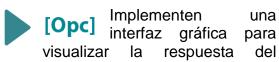
Puerto Antioquia en Urabá es un proyecto en curso que traerá mucho desarrollo a la región. Su construcción está prevista para terminar a finales de 2020.

Por sus características y ubicación geográfica, Puerto Antioquia podría convertirse en el centro logístico de mayor importancia del país. Un problema que tendrán, una vez inicie la operación, es encontrar la forma óptima de empacar la mercancía en los contenedores.



Una versión simplificada de este problema es una versión en dos dimensiones: Hay un tablero de  $2 \times n$  cuadrados y usted necesita saber de cuantas maneras se puede llenar el tablero usando rectángulos de  $1 \times 2$ .







algoritmo anterior utilizando la librería JavaFX, Graphics, print o equivalente.

Foto extraída de <a href="https://bit.ly/2DycZEb">https://bit.ly/2DycZEb</a>

#### PhD. Mauricio Toro Bermúdez





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

en la carpeta ejercicioEnLinea

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



Vean Guía numeral 3.3



No se requiere documentación para los ejercicios en línea



Utilicen Java, C++ o Python



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



Resolver los problemas de CodingBat usando Recursión



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

- NOTA: Si toman la respuesta de alguna fuente, deben referenciar según el tipo de cita. Vean *Guía en numerales 4.16 y 4.17*
- Resolver al menos 5 ejercicios del nivel *Recursion 1* de CodingBat: http://codingbat.com/java/Recursion-1
- Resolver al menos 5 ejercicios del nivel *Recursion 2* de la página CodingBat: http://codingbat.com/java/Recursion-2
- NOTA: No está permitido el ejercicio GroupSum

PhD. Mauricio Toro Bermúdez





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

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



Vean **Guía** numeral 3.4



Exporten y entreguen informe de laboratorio en PDF, en español o Inglés



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



No apliquen **Normas Icontec** para esto

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

#### Sobre el Ejercicio 1



Calculen la complejidad asintótica para el peor de los casos del Ejercicio 1.2



Tome tiempos para 20 tamaños del problema diferentes, genere una gráfica y analice los resultados. Estime, ¿Cuánto tiempo se demorará este algoritmo en calcular las formas que existen de llenar un rectángulo de 50x2 cm² con rectángulos de 1x2 cm²?



¿La complejidad de este algoritmo hace que sea viable utilizarlo en Puerto Antioquia, en 2020, con contenedores que miden miles de centímetros?

#### PhD. Mauricio Toro Bermúdez



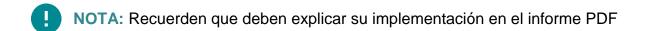


#### Sobre el Ejercicio 2





[Opcional] Expliquen con sus propias palabras cómo funciona el ejercicio *GroupSum5* 





Calculen la complejidad de los Ejercicios en Línea de los numerales 2.1 y 2.2 y agréguenla al informe PDF





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



#### Ejemplos de su respuesta:

"n es el número del 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











# 4. Simulacro de parcial en informe PDF

# 4 Simulacro de Parcial en el informe PDF



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



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



Pista 1: Vean Guía en numeral 4.18

En la vida real, los palíndromes se utilizan para desarrollar algoritmos de compresión de cadenas de ADN. Para este parcial, considera un algoritmo capaz de decir si una cadena de caracteres es un palíndrome o no. Un palíndrome es una cadena que se lee igual de izquierda a derecha que de derecha a izquierda. A continuación, algunos ejemplos:

- Para "amor a roma", la respuesta es true
- · Para "mamita", la respuesta es false
- Para "cocoococ", la respuesta es true

El algoritmo isPal soluciona el problema, pero le faltan unas líneas. Complétalas, por favor.

```
01 static boolean isPal(String s) {
02  if(s.length() == 0 || s.length() == 1)
03   return .....;
04  if(.....)
05   return isPal(s.substring(1, s.length()-1));
06  //else
07  return false;
08 }
```

#### PhD. Mauricio Toro Bermúdez







En Java, el método s.charAt(i) permite saber qué caracter hay en la posición *i* de la cadena s y s.substring(a,b) retorna una subcadena de s entre los índices a y b-1.



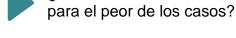
Completen, por favor, las líneas faltantes:

Línea 3: _		
Línea 4:		

4.2 Pepito escribió el siguiente código usando recursión:

```
private int b(int[] a, int x, int low, int high) {
  if (low > high) return -1;
  int mid = (low + high)/2;
  if (a[mid] == x) return mid;
  else if (a[mid] < x)
      return b(a, x, mid+1, high);
  else
    return b(a, x, low, mid-1);
}</pre>
```

¿Cuál ecuación de recurrencia describe el comportamiento del algoritmo anterior



- a) T(n)=T(n/2)+C
- **b)** T(n)=2.T(n/2)+C
- c) T(n)=2.T(n/2)+Cn
- **d)** T(n)=T(n-1)+C

#### PhD. Mauricio Toro Bermúdez









Dayla y Kefo están aquí de nuevo. En esta vez han traído un juego muy interesante, en el cual Kefo, en primer lugar, escoge un numero  $n (1 \le n \le 20)$  y, en segundo lugar, escoge tres números a,b y  $c (1 \le a \le 9, 1 \le b \le 9, 1 \le c \le 9)$ .

Después, Kefo le entrega estos números a Dayla y Dayla le tiene que decir a Kefo la cantidad máxima de números tomados de a, b y c (se puede tomar un número más de una vez) que al sumarlos dan el valor n.

**Como un ejemplo**, si Kefo escoge n=14 y a=3,b=2,c=7. ¿Qué posibilidades hay de obtener 14 con a,b y c?

7+7=14	cantidad es 2
7+3+2+2=14	cantidad es 4
3+3+3+3+2=14	cantidad es 5
 2+2+2+2+2+2+2=14	cantidad es 7

La cantidad máxima de números es 7. Esta sería la respuesta que da Dayla a Kefo.

Como Dayla es muy astuta, ha diseñado un algoritmo para determinar la cantidad máxima de números y quiere que le ayudes a terminar su código.

Asuman que hay al menos una forma de sumar n usando los números a, b y c en diferentes cantidades, incluso si algunos de los números se suman 0 veces como sucede en el ejemplo anterior.



4.3.1 Completen el espacio de la línea 04

\_\_\_\_\_



**4.3.2** Completen los espacios de la línea 05

#### PhD. Mauricio Toro Bermúdez







4.3.3 Completen los espacios de la línea 06

¿Qué calcula el algoritmo desconocido y cuál es la complejidad asintótica en el peor de los casos del algoritmo desconocido?

```
01 public int desconocido(int[] a) {
02    return aux(a, a.length-1); }
03
04 public int aux(int[] a, int n) {
05    if(n < 1) return a[n];
06    else return a[n] + aux(a, n-1); }</pre>
```



#### Elija la respuesta que considere acertada:

- a) La suma de los elementos del arreglo a y es
- **b)** Ordena el arreglo a y es O(n.log n)
- c) La suma de los elementos del arreglo a y es O(1)
- d) El máximo valor de un arreglo a y es O(n)
- e) La suma de los elementos del arreglo a y es O(n)

En la vida real, la teoría de juegos ha demostrado ser muy útil en la inteligencia artificial moderna; por ejemplo, para hacer transferencia de estilo que permite generar obras de arte con el estilo de un pintor determinado. Para este parcial, considera un juego en el que un jugador puede ganar 3, 5 o 7 puntos en un solo turno. ¿De cuántas maneras puede el jugador obtener un total de *T* puntos? A continuación, algunos ejemplos:

#### PhD. Mauricio Toro Bermúdez







- Para *T*=10. Respuesta: 3 que son 5+5, 7+3, 3+7.
- Para *T*=2. Respuesta: 0.
- Para *T*=15. Respuesta: 8.

El algoritmo ways soluciona el problema, pero le faltan unas líneas. Complétalas, por favor.

```
01 int ways(int T) {
02    //Caso(s) base(s).
03    .........
04    ........
05    int f1 = ways(T - 3);
06    int f2 = ways(T - 5);
07    int f3 = ways(T - 7);
08    return    .....;
09 }
```

4. 5.1 Completen las líneas faltantes:

Línea 3:	
Línea 4:	
Línea 8:	

4.5.2 ¿Cuántas instrucciones ejecuta el algoritmo en el peor de los casos?:

- **a.** T(n)=T(n-1)+C
- **b.** T(n)=T(n-1)+T(n-2)+C
- **c.** T(n)=T(n/2)+C
- **d.** T(n)=T(n+1)+C

Alek y Krish están jugando *Número*. Número es un juego en el que un jugador 1, entrega un numero n (1<=n<=10100) a un jugador 2 y el jugador 2 debe determinar la suma de todos los dígitos de n, exceptuando el caso en el que hay dos dígitos adyacentes (es decir, contiguos, seguidos) que son iguales.

#### PhD. Mauricio Toro Bermúdez



Si hay dos dígitos adyacentes, no se suma ninguno de los dos números adyacentes. Entre Alek y Krish escribieron un código para hacer esto más rápido, pero se ha borrado una parte. ¿Podrían ayudarles a reconstruir el código a Alek y Krish?

```
1 public int suma(String n) {
2
     return sumaAux(n, 0);
3 }
4
5 private int sumaAux(String n, int i){
     if (i \ge n.length()) {
7
       return 0;
8
     }
9
     if(i + 1 < n.length() \&\&
        n.charAt(i) == n.charAt(i + 1)){
10
        return ___;
11
12
     return (n.charAt(i) - '0') + ;
13
     }
```

La operación n.charAt(i) - '0' convierte un caracter en su equivalente en entero, por ejemplo, el caracter'1' lo transforma en el número 1.

# 4.6.1 Completen la línea 10:

# 4. 6.2 Completen la línea 12:

Dado un conjunto S de *n* elementos se quiere determinar si existe un subconjunto *R* de S tal que la suma de los elementos de *R* es igual a *t*, con la condición que, si se toma un elemento par, el siguiente elemento no puede estar en *R*.

#### PhD. Mauricio Toro Bermúdez

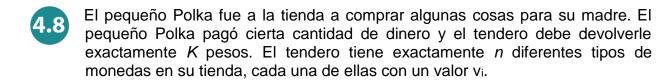




Al anterior código le faltan algunas líneas las cuales deben completar:



# **4.7.2** Línea 10:



Como el tendero no es muy astuto en asuntos de matemáticas, te ha pedido el favor de decirle de cuantas maneras puede él devolverle tal cantidad a Polka. Por favor ayúdanos a completar el siguiente código.

**Ejemplo:** Sea n = |v| = 3,  $v = \{3,4,1\}$ , K=7 Las formas posibles de devolver 7 dólares serían 5:

```
1. 1 + 1 + 1 + 1 + 1 + 1 + 1
2. 3 + 1 + 1 + 1
```

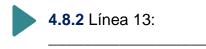
#### PhD. Mauricio Toro Bermúdez





```
3.3 + 3 + 1
     4.3 + 4
     5.4+1+1+1
01 int cuantas(int K, int[] v, int n){
    if(K == 0) {
03
      return 1;
04 }
05 boolean imposible;
06 imposible = n \le 0 \&\& K >= 1;
07 imposible = imposible || K < 0;
08 if(imposible){
09
10 }
11 int ni = cuantas(K, v, n - 1);
12 int nj = cuantas(K - v[n-1], v, n);
13 int suma =
14 return suma;
15 }
```

### **4.8.1** Línea 9:



Considere el siguiente programa. ¿Cuál es la salida generada por fun(11,5)? Como un ejemplo: fun(10,3)=20.

```
int fun(int n, int m) {
  if (n % m == 2) return n;
  return fun(n + m, n - m);
}
```



#### PhD. Mauricio Toro Bermúdez







- **a.** 11
- **b.** 5
- **c.** 22
- **c.** 2
- Considere el siguiente programa. ¿Cuál es la salida para *fun*(1,4)? Como un ejemplo: *fun*(1,2)=4.

```
int fun(int m,int n){
  if(m==0) {
    return (n+1);
  }
  if(m>0 && n==0) {
    return fun(m-1,1);
  }
  int a=fun(m,n-1);
  return fun(m-1,a);
}
```

#### Elija la respuesta que considere acertada:

- **a.** 4
- **b.** 6
- **c.** 5
- **d**. 12

Lika y Kefo están estudiando para el examen de matemáticas en su colegio. Hoy, ellos encontraron muchas secuencias de números interesantes, una de ellas los números Fibonacci. Para Lika –que ya conocía estos números– se le hace aburrido escribirlos todos; sin embargo, ellos encontraron otra secuencia de números llamados los números de Lucas. En el libro donde ellos encontraron esta secuencia de números, sólo hay 7 números pertenecientes a la secuencia y al final de la hoja dice: "¿Podrás definir una relación de recurrencia que nos permita deducir el n-ésimo número de Lucas?" "Por supuesto" –dijeron ambos. Ahora, ellos quieren escribir un algoritmo que nos permita retornar el n-ésimo número de Lucas. ¿Puedes ayudarlos a escribir el algoritmo? La secuencia encontrada fue la siguiente: 2, 1, 3, 4, 7, 11, 18, .... Se sabe

#### PhD. Mauricio Toro Bermúdez







que el número 2 y el número 1 son el primer y segundo término, respectivamente, de la secuencia de los números de Lucas.

```
1 int lucas(int n) {
2   if(n == 0) return 2;
3   if(n == 1) return 1;
4   return lucas(_____) + ____;
5 }
```

- **4.11.1** Completa la línea 4: \_\_\_\_\_, \_\_\_\_
- **4.11.2** La complejidad asintótica del algoritmo anterior, para el peor de los casos, en términos de *n*, es:
  - a. T(n)=T(n-1)+c, que es O(n)
  - **b.** T(n)=4T(n/2)+c, que es  $O(n^2)$
  - c. T(n)=T(n-1)+T(n-2)+c, que es  $O(2^n)$
  - **d.** T(n)=c, que es O(1)

Un conejo está ubicado en la celda (0,0) de un tablero A de  $n \times m$ . Si el conejo está en la casilla (i,j), el conejo puede avanzar –únicamente– horizontalmente a la casilla (i+1,j) o verticalmente a la casilla (i,j+1). En algunas celdas (ik,jk), hay manzanas que le darán un grado de satisfacción d al conejo y, en otras celdas (ik,jh), hay zanahorias que le darán satisfacción k al conejo. Donde hay manzanas estará el carácter '\*' y donde hay zanahorias estará el carácter '#'. Donde no hay ni zanahorias ni manzanas, está el carácter '.' ¿Cuál es la mayor satisfacción que puede tener el conejo, si el conejo recorre el tablero de la manera anteriormente mencionada y tiene que llegar a la posición (n-1,m-1) del tablero? Hemos escrito el siguiente algoritmo para ayudar al conejo, pero faltan algunas líneas, por favor complétalas. Puedes usar Math.max(a, b), de la librería de Java.

#### PhD. Mauricio Toro Bermúdez





```
7
          sat = d;
8
9
       if(A[i][j] == \#'){
10
          sat = k;
11
12
       if(i == n - 1 \&\& j == m - 1) {
          return ____;
13
14
15
       int fi = conejo(A, n, m, i + 1, j, d, k);
16
       int fj = conejo(A, n, m, i, j + 1, d, k);
17
       sat += ____;
18
       return ;
19
   }
```

- **4.12.1** Completa la línea 13: \_\_\_\_\_
- **4.12.2** Completa la línea 17: \_\_\_\_\_
- **4.12.3** Completa la línea 18:









# 5. [Opcional] Lecturas Recomendadas

# [Opc] Lecturas recomendadas



Vean Guía er 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



No apliquen **Normas Icontec** para esto

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



"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 <a href="http://bit.ly/2gJKzJD">http://bit.ly/2gJKzJD</a>



Vean Guía en numerales 3.5 y 4.20



Lean a ""Narasimha Karumanchi, Data Structures and Algorithms made easy in Java, (2<sup>nd</sup> edition), 2011. Chapter 3: Recursion and Backtracking" y sumen puntos adicionales, así:



Hagan un mapa conceptual con los principales elementos teóricos.

#### PhD. Mauricio Toro Bermúdez





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

# [Opc] Trabajo en Equipo y Progreso Gradual



Vean Guía numeral 3.5 y 4.20



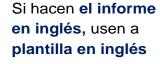
Exportar y entregar informe de laboratorio en PDF, en español o Inglés



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



No apliquen **Normas Icontec** para esto



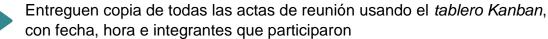


El trabajo en equipo es una exigenciadel mercado. "Mientras algunos medios retratan la programación como un trabajo solitario, pero requiere mucha comunicación y trabajo grupal. Si trabajas para una compañía, serás parte de un equipo de desarrollo y esperarán que te comuniques bien con otras personas" Tomado de http://bit.ly/1B6hUDp

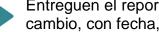


Vean Guía en numerales 3.6, 4.21,4.22,4.23









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





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







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





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



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* 

Entreguen el código y el informe en inglés.

PhD. Mauricio Toro Bermúdez







#### Resumen de ejercicios a resolver

- **1.1** Implementen un algoritmo que calcule la longitud de la subsecuencia común más larga a dos cadenas de caracteres.
- **1.2** Implementen un algoritmo recursivo para encontrar de cuántas formas se puede llenar un rectángulo de 2xn cm<sup>2</sup> con rectángulos de 1x2 cm<sup>2</sup>.

**[Ejercicio opcional]** Implementen una interfaz gráfica para visualizar la respuesta del algoritmo anterior utilizando la librería *JavaFX*, *Graphics*, *print* o equivalente.

- **2.1** Resuelvan al menos 5 ejercicios del nivel *Recursion* 1 de CodingBat: http://codingbat.com/java/Recursion-1
- **2.2** Resuelvan al menos 5 ejercicios del nivel *Recursion* 2 de la página CodingBat: http://codingbat.com/java/Recursion-2
- 3.1. Calculen la complejidad asintótica, para el peor de los casos, del ejercicio 1.2
- **3.2.** Tome tiempos para 20 tamaños del problema diferentes, genere una gráfica y analice los resultados. Estime, ¿cuánto tiempo se demorará este algoritmo en calcular las formas que existen de llenar un rectángulo de 50x2 cm² con rectángulos de 1x2 cm²?
- **3.3.** ¿La complejidad de este algoritmo hace que sea viable utilizarlo en Puerto Antioquia, en 2020, con contenedores que miden miles de centímetros?
- **3.4.** [Ejercicio opcional] Expliquen con sus propias palabras cómo funciona el ejercicio *GroupSum5*
- **3.5.** Calculen la complejidad de los Ejercicios en Línea de los numerales 2.1 y 2.2 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 ejercicio 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. Utilicen la plantilla dispuesta en este idioma para el laboratorio.

#### PhD. Mauricio Toro Bermúdez





# Ayudas para resolver los Ejercicios

Ayudas para todos los ejercicios	<u>Pág. 28</u>
Ayudas para el Ejercicio 1	<u>Pág. 29</u>
Ayudas para el Ejercicio 2.1	Pág. 29
Ayudas para el Ejercicio 2.2	Pág. 29
Ayudas para el Ejercicio 3.4	<u>Pág. 30</u>
Ayudas para el Ejercicio 3.5	Pág. 31
Ayudas para el Ejercicio 3.6	Pág. 31
Ayudas para el Ejercicio 5	<u>Pág. 32</u>
Ayudas para el Ejercicio 6.1	<u>Pág. 32</u>
Ayudas para el Ejercicio 6.2	<u>Pág. 32</u>
Ayudas para el Ejercicio 6.3	Pág. 32



### Ayudas para todos los ejercicios



Pista 1: Procedimiento sugerido:

- 1 Implementen el algoritmo
- 2 Identifiquen qué representa el tamaño del problema para cada algoritmo
- 3 Identifiquen valores apropiados para tamaños del problema
- 4 Tomen los tiempos para los anteriores algoritmos con 20 tamaños del problema diferentes.
- 5 Hagan una gráfica en Excel para cada algoritmo y pasarla a Word luego
- 6 Entreguen el archivo de Word pasado a PDF.



Pista 2: Vean la Guía en el numeral 4.2



Pista 3: Vean la Guía en el numeral 4.3 y 4.4



Pista 4: Vean la Guía en el numeral 4.7



Pista 5: Vean la Guía en el numeral 4.6

PhD. Mauricio Toro Bermúdez









### Ayudas para el Ejercicio 1.1

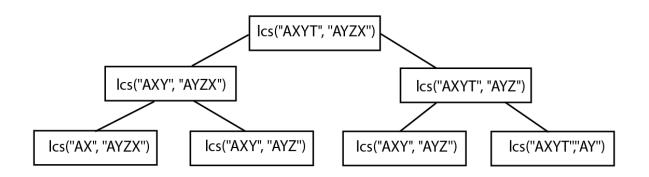


**Ejemplo 1:** "abc", "abg", "bdf", "aeg" y "acefg" son subsecuencias de "abcdefg". Entonces, para una cadena de longitud n existen 2<sup>n</sup> posibles subsecuencias.



**Ejemplo 2:** 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. Una forma de resolver este problema es usando *recursión*, como un ejemplo, para las cadenas "AXYT" y "AYZX", dada una función recursiva los que resuelve el problema, se obtendría el siguiente árbol (parcial) de recursión:





Pista 2: Completen el siguiente método:

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

#### PhD. Mauricio Toro Bermúdez







### Ayudas para el Ejercicio 1.2



**Pista 1:** Consideren llenar un tablero de  $2\times n$ . Si le quitamos la primera baldosa tenemos un tablero de  $2\times (n-1)$  (usando recursión). Si le quitamos 2 baldosas queda un tablero de  $2\times (n-2)$ . Hagan el dibujo. ¿Se le pueden quitar 3 baldosas, o con lo anterior ya puedo formar el de  $2\times (n-3)$ ?



### Ayudas para el Ejercicio 2.1



**Error Común** 





## Ayudas para el Ejercicio 2.2

#### PhD. Mauricio Toro Bermúdez







- Pista 1: El algoritmo *GroupSum* falla porque al llamarse recursivamente con el parámetro start se queda en una recursión infinita
- **Pista 2:** El algoritmo *GroupSum* falla porque al llamarse recursivament con el parámetro start-1 se sale del arreglo cuando start = 0.
- Pista 3: El algoritmo *GroupSum* falla porque al llamarse recursivamente con el parámetro start se sale del arreglo cuando start = length-1
- A continuación, encontrarán algunos errores comunes que pueden ser aplicados con los ejercicios



### Ayudas para el Ejercicio 3.4



Vean la *Guía en* el numeral 4.11



Vean la Guía en el numeral 4.19

#### PhD. Mauricio Toro Bermúdez

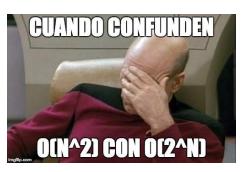














## Ayudas para el Ejercicio 3.5



Pista 1: <a href="http://bit.ly/2ix7rjl">http://bit.ly/2ix7rjl</a>



**Pista 2:** Si todos los tiempos de un algoritmo dan más de 5 minutos, realice otra tabla, para ese algoritmo, tomando tiempos para tamaños del problema más pequeños.



### Ayudas para el Ejercicio 3.6



Pista 1: http://bit.ly/2ix7rjl

#### PhD. Mauricio Toro Bermúdez







## Ayudas para el Ejercicio 5



Pista 1: Para que hagan el mapa conceptual se recomiendan herramientas como las que encuentran en <a href="https://cacoo.com/">https://cacoo.com/</a> o <a href="https://cacoo.com/">https://cacoo.com/</a> o <a href="https://www.mindmup.com/#m:new-a-1437527273469">https://www.mindmup.com/#m:new-a-1437527273469</a>



### Ayudas para el Ejercicio 6.1



Pista 1: Vean Guía en numeral 4.21



### Ayudas para el Ejercicio 6.2



Pista 1: Vean Guía en numeral 4.23



## Ayudas para el Ejercicio 6.3



Pista 1: Vean Guía en numeral 4.22

PhD. Mauricio Toro Bermúdez







# ¿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