

Laboratorio Nro. 1 Recursión



Objetivos: 1. Identificar el caso base y el recurrente de un problema definido recursivamente. 2. Usar ecuaciones de recurrencia para determinar la complejidad en tiempo de algoritmos definidos de forma recursiva.



Consideraciones: Lean y verifiquen las consideraciones de entrega,



Leer la Guía



Trabajo en
Parejas



Si tienen reclamos,
regístrenlos en
<http://bit.ly/2q4TTKf>



Ver
calificaciones
en Eafit
Interactiva



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

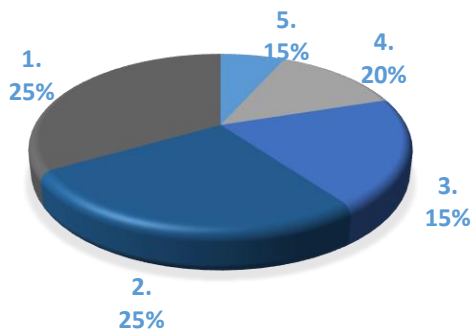


Hoy, plazo de
entrega



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



- 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

Resumen de los ejercicios a resolver

1.1 Implementen un algoritmo que calcule la longitud de la subsecuencia común (en el mismo orden relativo), más larga, entre dos cadenas de caracteres. Y probarla con los conjuntos de datos (en inglés, *datasets*) incluidos en github.

1.2 Implementen un algoritmo recursivo para encontrar de cuántas formas se puede llenar un rectángulo de $2 \times n$ cm² con rectángulos de 1×2 cm².

[Ejercicio opcional] Implementen una interfaz gráfica para visualizar la respuesta del algoritmo anterior utilizando la librería *JavaFX*, *Graphics*, *PyGame*, *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, para el peor de los casos, del ejercicio 1.1

3.2. Para el ejercicio 1.1, tomen tiempos para 20 tamaños del problema diferentes, generen una gráfica y analicen los resultados. Estimen cuánto tiempo se demorará este algoritmo en la subsecuencia común más larga entre dos ADNs mitocondriales (que tienen alrededor de 300.000 caracteres cada uno).

3.3. ¿La complejidad del algoritmo del ejercicio 1.1 es apropiada para encontrar la subsecuencia común más larga entre ADNs mitocondriales como los de los *datasets*?

3.4. [Ejercicio opcional] Expliquen con sus propias palabras cómo funciona *GroupSum5*

3.5. Calculen la complejidad de los **Ejercicios en Línea** de los numerales 2.1 y 2.2

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 en inglés. Utilicen la plantilla dispuesta en este idioma para el laboratorio.

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`



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**, **pydoc** o equivalente. No suban el HTML a GitHub.



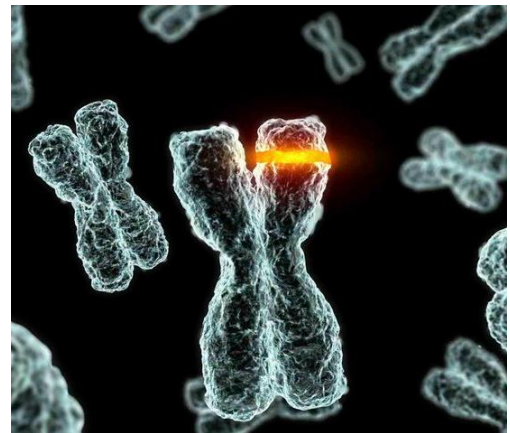
No se reciben archivos en **.RAR** ni en **.ZIP**



Utilicen Java, C++ o Python

1.1 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 (ordenada) más larga.



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

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 (en el mismo orden relativo) más larga a dos cadenas de caracteres.**

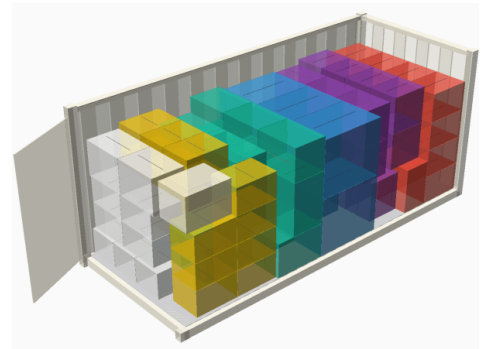


Utilicen los conjuntos de datos de ADNs que se encuentra en la carpeta *datasets*, en Github, para probar su algoritmo.

1.2

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 2021.

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 empaquetar 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 cuántas maneras se puede llenar el tablero usando rectángulos de 1×2 .

► **Implementen un algoritmo recursivo para encontrar de cuántas formas se puede llenar un rectángulo de $2 \times n$ cm² con rectángulos de 1×2 cm².**

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



Foto extraída de <https://bit.ly/2DycZEB>

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, en GitHub, dentro de 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**

2.1 Resolver --al menos-- 5 ejercicios del nivel *Recursion 1* de CodingBat:
<http://codingbat.com/java/Recursion-1>

2.2 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 ni Factorial**

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



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 1.1



3.1 Calculen la complejidad para el peor de los casos del Ejercicio 1.1.



3.2 **Tomen tiempos para 20 tamaños del problema diferentes, generen una gráfica y analicen los resultados.** Estimen cuánto tiempo se demorará este algoritmo en la subsecuencia común más larga entre dos ADNs mitocondriales (que tienen alrededor de 300.000 caracteres cada uno).



3.3 ¿La complejidad del algoritmo del ejercicio 1.1 es apropiada para encontrar la subsecuencia común más larga entre ADNs mitocondriales como los de los datasets?

Sobre el Ejercicio 2




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


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

 **NOTA:** Recuerden que deben explicar su implementación en el informe PDF

 **3.5** Calculen la complejidad de los Ejercicios en Línea de los numerales 2.1 y 2.2 y agréguela 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.

 **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”.

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*

4.1

Una de las tecnologías cruciales para que la Cuarta Revolución Industrial tenga éxito es la ciberseguridad. Una de las áreas más importantes de la ciberseguridad es la criptografía. Criptografía es el arte y técnica de escribir claves secretas de tal forma que lo escrito solamente sea inteligible para quien sepa descifrarlo. Consideremos una aplicación de criptografía. Algunas palabras del alfabeto español se pueden escribir como la unión consecutiva de los símbolos de la tabla periódica universal. Por ejemplo, la palabra “Población” se puede escribir como la unión de los símbolos {Po, B, La, C, I, O, N}, PoBLaCION. Te entregan una cadena de caracteres S y un conjunto de símbolos T. El objetivo es determinar si S se puede escribir como la unión consecutiva de cero o más símbolos de T. Por ejemplo, sea $T = \{Ti, B, I, O, C\}$ y $S = \text{“Biótico”}$, la respuesta es verdadero; para $S = \text{“”}$, la respuesta es verdadero; y para $S = \text{“Titan”}$, la respuesta es falso. El siguiente código resuelve el problema, pero le faltan algunas líneas; por favor, complétalas. ¡Gracias!

Puedes asumir que T contiene todas las posibles combinaciones de acentos, mayúsculas y minúsculas de cada símbolo de la tabla periódica. En Java, el método `s.substring(a,b)` retorna la subcadena de s entre los índices a y b-1, incluidos. El método `t.contains(s)` retorna verdadero si la cadena s se encuentra dentro de t; de lo contrario, falso.

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  boolean solve(String s, List<String> t){
2      return solve(t, s, s.length());
3  }
4  boolean solve(List<String> t, String s,
5              int n){
6      for(int i = 1; i <= n; ++i){
7          String pfx = .....;
8          if(t.contains(pfx)){
9              if(i == n){
10                 return .....;
11             }
12             return .....;
13         }
14     }
15     return false;

```

1. Completa la línea 6
 - a. s.substring(0, i)
 - b. s.substring(0, n)
 - c. s.substring(i, n)
2. Completa la línea 9
 - a. false
 - b. s.substring(0, i)
 - c. true
3. Completa la línea 11
 - a. solve(t, s.substring(i), n - i)
 - b. solve(pfx, t), n - i)
 - c. solve(t, s.substring(n), I - n)

4.2 Rellenar por difusión se utiliza para implementar el tarrito de *Microsoft Paint*. Además, según el portal *Geeks for Geeks*, es un problema –muy frecuente– en entrevistas de Amazon, Microsoft y Adobe. El problema es el siguiente. Dado una matriz de dimensión $N \times M$ (screen), la ubicación de un pixel en la matriz (x,y) y un color

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

($newC \in N$), nuestra tarea es reemplazar el color ($prevC \in N$) del pixel (x,y) –y el de todos sus vecinos que tengan el color $prevC$ – con el color $newC$. **¡Incluyamos los vecinos diagonales!** Como un ejemplo, si tenemos $x=4,y=4,prevC=2,newC=3$ y la matriz (`screen`) en el lado izquierdo, después de aplicar el algoritmo, debe lucir como la del lado derecho. Faltan unas líneas, complétalas, por favor.

```
{1, 1, 1, 1, 1, 1, 1, 1},    {1, 1, 1, 1, 1, 1, 1, 1},
{1, 1, 1, 1, 1, 1, 0, 0},    {1, 1, 1, 1, 1, 1, 0, 0},
{1, 0, 0, 1, 1, 0, 1, 1},    {1, 0, 0, 1, 1, 0, 1, 1},
{1, 2, 2, 2, 2, 0, 1, 0},    {1, 3, 3, 3, 3, 0, 1, 0},
{1, 1, 1, 2, 2, 0, 1, 0},    {1, 1, 1, 3, 3, 0, 1, 0},
{1, 1, 1, 2, 2, 2, 2, 0},    {1, 1, 1, 3, 3, 3, 3, 0},
{1, 1, 1, 1, 1, 2, 1, 1},    {1, 1, 1, 1, 1, 3, 1, 1},
{1, 1, 1, 1, 1, 2, 2, 1},};  {1, 1, 1, 1, 1, 3, 3, 1},};
```

Si trabajas en Java, revisa este código:

ESTRUCTURA DE DATOS 1
Código ST0245

```

1  void floodFillUtil(int screen[][], int x, int
    y, int prevC, int newC, int N, int M) {
2  if (x < 0 || x >= M || y < 0 || y >= N)
3      return;
4  if (screen[x][y] != prevC)
5      return;
6
7  screen[x][y] = newC;
8
9  ....
10 ....
11 ....
12 ....
13 floodFillUtil(screen, x+1, y, prevC, newC, N,
    M);
14 floodFillUtil(screen, x-1, y, prevC, newC, N,
    M);
15 floodFillUtil(screen, x, y+1, prevC, newC, N,
    M);
16 floodFillUtil(screen, x, y-1, prevC, newC, N,
    M); }
17
18 void floodFill(int screen[][], int x, int y,
    int newC, int N, int M) {
19 int prevC = screen[x][y];
20 floodFillUtil(screen, x, y, prevC, newC, N, M
    ); }

```

Si trabajas en Python, revisa este código:

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

```

1  def floodFillUtil(screen, x, y, prevC, newC, N
    , M):
2  if x < 0 or x >= M or y < 0 or y >= N or
    screen[x][y] != prevC:
3      return
4  if screen[x][y] == newC:
5      return
6
7  screen[x][y] = newC
8
9  ....
10 ....
11 ....
12 ....
13 floodFillUtil(screen, x + 1, y, prevC, newC,
    N, M)
14 floodFillUtil(screen, x - 1, y, prevC, newC,
    N, M)
15 floodFillUtil(screen, x, y + 1, prevC, newC,
    N, M)
16 floodFillUtil(screen, x, y - 1, prevC, newC,
    N, M)
17
18 def floodFill(screen, x, y, newC, N, M):
19     prevC = screen[x][y]
20     floodFillUtil(screen, x, y, prevC, newC, N, M
        )

```

1. Completa la línea 9
Y Completa la línea 10
2. Completa la línea 11
Y Completa la línea 12
3. ¿Cuál es la ecuación de recurrencia que representa la complejidad, en el tiempo, para el peor de los casos, en términos de $p=n+m$?
 $T(p) = \dots\dots\dots$

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 En la vida real, las grandes empresas de tecnología, solicitan calcular la complejidad de los algoritmos que preguntan en sus entrevistas técnicas; por ejemplo, Google, Facebook y Riot Games. Estas entrevistas se realizan para acceder a una práctica laboral como lo hizo Santiago Zubieta, en 2015, y Juan M. Ciro, en 2019, para hacer sus prácticas en Google y Facebook, respectivamente. ¡Ambos eran estudiantes de la Universidad Eafit! Para prepararnos para esas entrevistas, para cada uno de los siguientes códigos, determina la ecuación que representa su complejidad para el peor caso.

Para el siguiente código, determina la ecuación que representa su complejidad para el peor caso. Considera que C es una constante.

```

1  void mystery(int n, int m){
2      int res = 0;
3      for(int i = 0; i < n; ++i){
4          for(int j = 1; j < m; ++j){
5              for(int k = 1; k < m; ++k){
6                  res = res + 1;
7              } } } }

```

La complejidad de la función *mystery* es

- $T(n,m) = C \times n \times m$
- $T(n,m) = C \times n \times m^2$
- $T(n,m) = C \times n \times m \cdot \log m$
- $T(n,m) = C \times n \times m^3$

4.4 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 que

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

el número 2 y el número 1 son el primer y segundo término, respectivamente, de la secuencia de los números de Lucas. Al código le faltan algunas líneas, para completar el ejercicio deberás completarlas.

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



4.4.1 La complejidad del algoritmo anterior, para el peor de los casos, en términos de n y donde c es una constante, 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 $O(1)$

4.5

En la vida real, los palíndromos 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 palíndromo o no. Un palíndromo 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

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

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:

1. Completa la línea 3

- a. true
- b. false
- c. s

2. Completa la línea 4

- a. `s.substring(0,s.length()-1).equals(s.substring(s.length()-1, 0))`
- b. `s.charAt(0) == (s.charAt(s.length()-1))`
- c. true

4.6 [Opc] 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);
}
```



¿Cuál ecuación de recurrencia describe el comportamiento del algoritmo anterior para el peor de los casos?

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

4.7 [Opc] Alek y Krish están jugando *Número*. *Número* es un juego en el que un jugador 1, entrega un número n ($1 \leq n \leq 10100$) a un jugador 2 y el jugador 2 debe determinar la suma de todos los dígitos de n , exceptuando el caso

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

en el que hay dos dígitos adyacentes (es decir, contiguos, seguidos) que son iguales.

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){
6     if (i >= 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.



4.7.1 Completen la línea 10:



4.7.2 Completen la línea 12:

4.8

[Opc]

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

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

- 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, para poder resolver el ejercicio deberás completarlas.

```
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 }
```

▶ **4.8.1** ¿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. $T(n)=T(n+1)+C$

4.9 [Opc] 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

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

- c. 5
- d. 12

4.10 [Opc] 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 que el número 2 y el número 1 son el primer y segundo término, respectivamente, de la secuencia de los números de Lucas. Al código le faltan algunas líneas, para completar el ejercicio deberás completarlas.

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

▶ **4.10.1** La complejidad 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 $O(1)$

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**



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 <http://bit.ly/2gJKzJD>*



Vean *Guía* en **numerales 3.5 y 4.20**



Lean a **“Narasimha Karumanchi, Data Structures and Algorithms made easy in Java, (2nd 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

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 3.5 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 trabajo en equipo es una exigencia del 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**

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 3.5 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 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

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



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

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 1.1

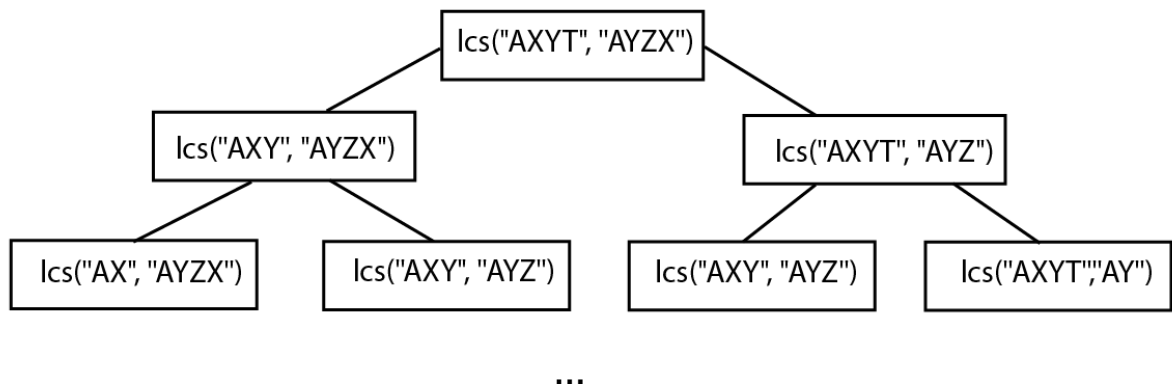


Ejemplo 1: “abc”, “abg”, “bdf”, “aeg” y “acefg” son subsecuencias de “abcdefg”. Entonces, para una cadena de longitud n existen 2^n posibles subsecuencias.



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

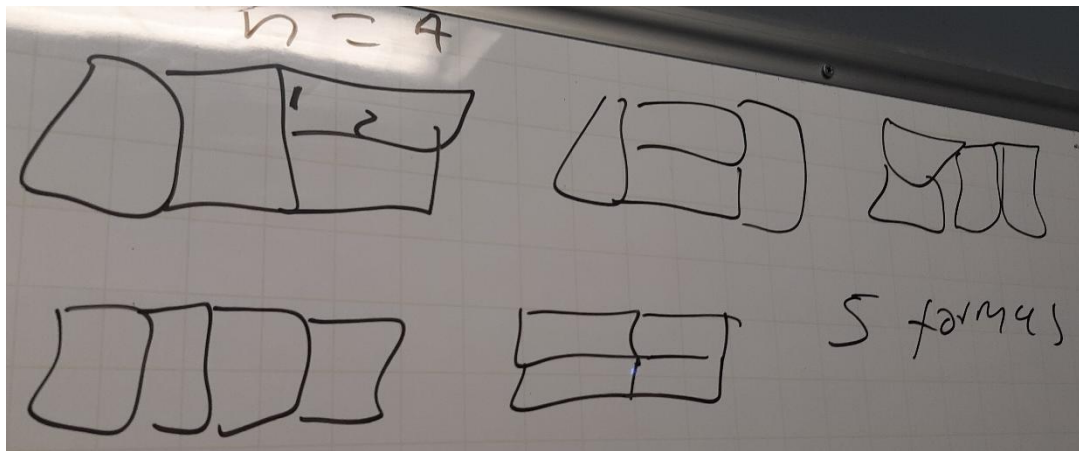
Docente | Escuela de Ingeniería | Informática y Sistemas
Correo: mtorobe@eafit.edu.co | Oficina: Bloque 19 – 627
Tel: (+57) (4) 261 95 00 Ext. 9473



Ayudas para el Ejercicio 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 $2 \times (n-3)$? Como un ejemplo, estas son las 5 formas para $n = 4$:



Ayudas para el Ejercicio 2.1



Error Común



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 2.2



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 recursivamente 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

```
public boolean groupSum(int start, int[] nums, int target) {
    if (start >= nums.length) return target == 0;
    return groupSum(start+1, nums, target - nums[start])
        || groupSum(start, nums, target );
}
```

```
public boolean groupSum(int start, int[] nums, int target) {
    if (start >= nums.length) return target == 0;
    return groupSum(start+1, nums, target - nums[start])
        || groupSum(start-1, nums, target );
}
```

```
public boolean groupSum(int start, int[] nums, int target) {
    if (start >= nums.length) return target == 0;
    return groupSum(start+1, nums, target - nums[start])
        || groupSum(start+1, nums, target - nums[start - 1] );
}
```



Ayudas para el Ejercicio 3.4

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



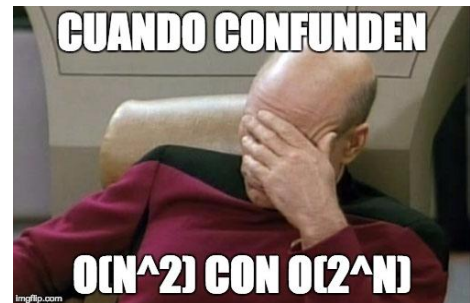
Vean la *Guía* en el numeral 4.11



Vean la *Guía* en el numeral 4.19



Errores Comunes



Ayudas para el Ejercicio 3.5



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



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

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 1: <http://bit.ly/2ix7rjl>



Ayudas para el Ejercicio 5



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



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

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>