

## Laboratorio Nro. 2

### Fuerza Bruta (*Brute force* o *Exhaustive search*)



**Objetivo:** Diseñar algoritmos usando la técnica de diseño de fuerza bruta



**Consideraciones:** Lean y verifiquen las consideraciones de entrega,



Leer la Guía



Trabajar en  
Parejas



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



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**

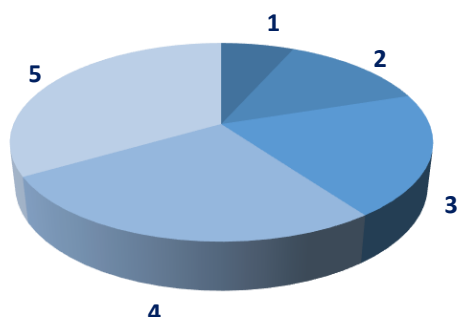


Hoy, plazo  
de entrega



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

Una de las funciones sustanciales del grupo EMI es brindar el servicio médico en casa. Un problema que enfrenta actualmente EMI es encontrar el orden óptimo para visitar a sus clientes de tal forma que se minimice la distancia total del recorrido que tiene que hacer cada uno de sus médicos. Este problema se conoce como el problema del 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

## ESTRUCTURA DE DATOS 2

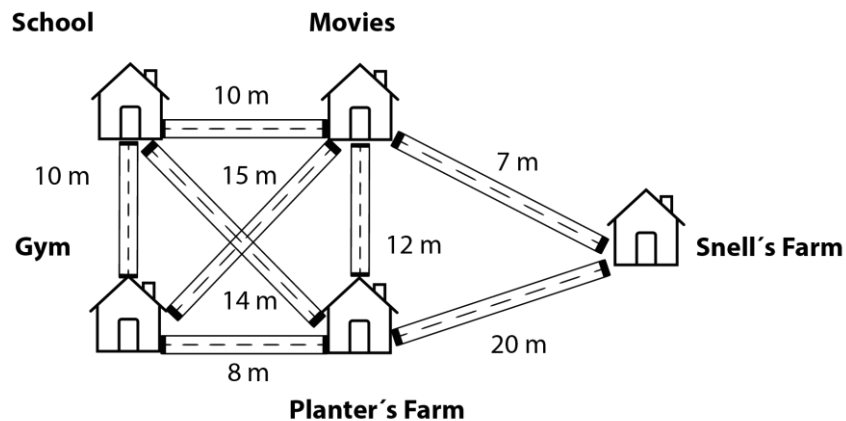
### Código ST0247

► Dado un grafo dirigido, hallen el costo mínimo del recorrido que pasa por todos los vértices exactamente una vez y vuelve al nodo inicial. Diseñe su algoritmo utilizando fuerza bruta.

! **Nota:** Para realizar una prueba, en Github encontrará el archivo *puentes\_colgantes.txt* que contiene el ejemplo a continuación. También encontrará un mapa de la ciudad de Medellín.



**Ejemplo 1**, para el siguiente mapa, el archivo de entrada es el siguiente:



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

```

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

**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 2**  
**Código ST0247**

**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

**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



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

### 2.1 Resuelvan el siguiente ejercicio

El problema de la  $n$  reinas es conocido por cualquier persona que haya estudiado Ingeniería de Sistemas.

En este problema ustedes deben contar el número de formas de ubicar  $n$ -reinas en un tablero de  $n \times n$  de tal forma que no haya ni un par de reinas que se ataquen. Para hacer el problema un poco más duro (¿o más fácil?), resulta que hay unos cuadros malos en los que no se pueden poner reinas.

Por favor, tengan en cuenta que los cuadros malos no se pueden usar para bloquear un ataque de una reina, es decir, no es un muro, es sólo un cuadro malo. También consideren que, incluso si dos soluciones se vuelven la misma después de hacer unas rotaciones al tablero, se deben contar como diferentes.

**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 2

### Código ST0247

Por consiguiente, existen 92 soluciones en el problema tradicional de las 8 reinas.



#### Entrada

La entrada consiste de al menos 10 casos de prueba. Cada caso contiene un entero  $n$  ( $3 \leq n \leq 15$ ) en la primera línea. Posteriormente, las siguientes líneas representan el tablero donde los cuadros en blanco son representados por puntos (.) y los cuadros malos son representados por asteriscos (\*). El último caso es seguido de un número 0, que no debe ser procesado.



#### Salida

Para cada caso de prueba, impriman el número del caso y el número de soluciones. **El tiempo máximo de ejecución es de 5 segundos.** Tengan en cuenta que  $3 \leq n \leq 15$ .



#### Ejemplos de las entradas

8

.....  
 .....  
 .....  
 .....  
 .....  
 .....  
 .....  
 .....

4

\*.  
 ....  
 ....  
 ....

0

**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





### Ejemplos de la salida

Case 1: 92

Case 2: 1



**[Opc]** Para los numerales 2.2 al 2.5, resuelvan los siguientes problemas

2.2

<http://bit.ly/2gTLZ53>

2.3

<http://bit.ly/2hGqJPB>

2.4

<http://bit.ly/2hrrCfS>

2.5

<http://bit.ly/2k8CGSG>

**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



Veán **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**



No apliquen **Normas**  
**Icontec** para esto

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



En la vida real, las técnicas usadas para resolver muchos  
problemas en Ingeniería de Sistemas son las mismas que las  
existentes para las N reinas

### Sobre el Ejercicio 1

3.1



Escriban una explicación entre 3 y 6 líneas de texto **del código del numeral 1.**





**Nota:** La explicación del código debe ir dentro del informe y no dentro  
del 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 2

### Código ST0247

- 3.2**  ¿Cuál es la complejidad asintótica, en el peor de los casos, para el ejercicio 1 ? Exprese su complejidad en término de dos variables:  $V$  (el número de vértices) y  $E$  (el número de aristas) del grafo.
- 3.3**  La solución del punto 1, es aplicable al problema de médico en casa de EMI para 50 clientes? Estime cuánto tiempo tomaría ejecutar su algoritmo para un grafo con 50 clientes.

### Sobre el Ejercicio 2

- 3.4**  Expliquen con sus propias palabras la estructura de datos que utilizan para resolver el problema del numeral 2.1 y cómo funciona el algoritmo.
- 3.5**  Calculen la complejidad del ejercicio 2.1 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 numeral 3.5. Ver ejemplo a continuación:



#### 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** El problema de encontrar el **subarreglo máximo** consiste en encontrar un subarreglo contiguo dentro de un arreglo de números cuya suma sea la máxima.

Como un ejemplo, para la secuencia de números  $-2, 1, -3, 4, -1, 2, 1, -5, 4$ ; el subarreglo contiguo con suma máxima es  $4, -1, 2, 1$  y su suma es 6.

El siguiente algoritmo es una solución con fuerza bruta del problema. El algoritmo consiste en buscar cada posible subarreglo contiguo y luego encontrar la suma de cada uno de ellos.

**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 2

### Código ST0247

```

01 int subarregloMax(int[] a) {
02     int maximo = 0;
03     for (int i = 0; i < a.length; i++) {
04         int actual = 0;
05         for (int j = i; j < a.length; j++) {
06             actual = actual + a[j];
07             if (_____)
08                 maximo = actual;
09         }
10     }
11     return maximo;
12 }

```



**De acuerdo a lo anterior, resuelvan lo siguiente:**

**4.1.1** Completen el espacio vacío en la línea 7

\_\_\_\_\_

**4.1.2** ¿Cuál es la complejidad, para el peor de los casos, del algoritmo?

O (\_\_\_\_)

**4.2 [Opc]** La función ordenar es un algoritmo de ordenamiento, de menor a mayor, por fuerza bruta. Dicho algoritmo calcula todas las permutaciones posibles de un arreglo *arr* hasta encontrar una permutación donde los elementos están ordenados (es decir, cuando esta Ordenado retorna verdadero).

Tengan en cuenta que ordenar imprime en la pantalla el arreglo *arr* ordenado, pero no necesariamente deja el arreglo a ordenado por la forma en que está diseñado el algoritmo. Aunque esto último no es deseable, ese no es el problema de este parcial.

**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 2**  
**Código ST0247**

```
static boolean estaOrdenado(int[] a) {
    for (int i = 0; i < a.length - 1; i++)
        if (a[i] > a[i + 1])
            return false;
    return true;
}
static void cambiar(int[] arr, int i, int k){
    int t = arr[i];
    arr[i] = arr[k];
    arr[k] = t;
}
static void ordenar(int[] arr, int k){
    for(int i = k; i < arr.length; i++){
        cambiar(arr, i, k);
        if (estaOrdenado(arr))
            System.out.println(
                Arrays.toString(arr));
        ordenar(_____, _____);
        cambiar(arr, k, i);
    }
}
```



**De acuerdo a lo anterior, resuelvan lo siguiente:**

**4.2.1** Completen los espacios vacíos en el llamado recursivo del método ordenar

\_\_\_\_\_, \_\_\_\_\_

**4.2.2** ¿Cuál es la complejidad, para el peor de los casos, del método ordenar?

O (\_\_\_\_)

**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** Un problema frecuente es buscar si una una cadena llamada **patrón** (`pat`) se encuentra dentro de otra cadena llamado **texto** (`txt`). Esto es lo que sucede en muchos programas cuando le damos *Edición, Buscar*. El objetivo es encontrar la posición en la que aparece por primera vez `pat` en `txt`



**Ejemplo:** el patrón “*atr*” aparece dentro del texto “patrón” en la posición 1.



**Como otro ejemplo:** el patrón “*mat*” no aparece en el texto “patrón”. Cuando no aparece, el algoritmo retorna la longitud del texto, es decir, en este caso, 5

Una forma de resolverlo es por fuerza bruta, probando todas las posibles posiciones en las que `pat` puede aparecer dentro de `txt` como se muestra a continuación:

```
01 int indexOf(String pat, String txt) {
02     int m = pat.length();
03     int n = txt.length();
04     int i, j;
05     for (i = 0, j = 0; i < n && j < m; i++) {
06         if (txt.charAt(i) == pat.charAt(j)) j++;
07         else {
08             i = i - j;
09             j = 0;
10         }
11     }
12     if (j == m) return _____ // encontrado
13     else      return _____; // no encontrado
14 }
```



De acuerdo a lo anterior, resuelvan lo 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

## 4.3.1 Completen la línea 12

---

## 4.3.2 Completen la línea 13

---

**4.3.3** ¿Cuál es la complejidad asintótica, para el peor de los casos, del algoritmo? (En términos de  $n$  y  $m$ )

$O(\rule{1cm}{0.4pt})$



**Pista:** NO es  $O(n!)$ .

**4.4**

Tenemos dos enteros,  $M$  y  $N$ . Queremos contar en el rango  $[M, N]$  todos los enteros que contienen los dígitos  $a, b, c, d$  y  $e$ ,  $1 \leq a, b, c, d, e \leq 9$ . Al siguiente algoritmo le faltan algunas líneas. Ayúdanos a completarlas



**Ejemplo:** Considere  $a=1, b=1, c=4, d=1, e=4$  y el rango  $[200, 300]$ . Los enteros que cumplen la condición son: 214, 241. Por lo tanto, la solución debería ser 2



**Nota:** Observe que  $a, b, c, d, e$  no tienen que ser necesariamente distintos; además sólo importa que  $a, b, c, d, e$  estén todos en el número, es decir, si  $S$  es el conjunto de todos los dígitos de algún número  $N \leq x \leq M$  y  $T = \{a, b, c, d, e\}$ , siempre se cumple que  $S \cap T = T$ .

```
01 public int contar(int N, int M){
02     int total = 0;
03     int[] set;
04     for(int i = N; i <= M; ++i){
05         set = new int[10];
06         int temp = i;
07         while(temp > 0){
08             int rem = _____;
09             set[rem] = set[rem] + 1;
10             temp = temp / 10;
11         }
12         if(set[a] > 0 && set[b] > 0 &&
```

**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 2

### Código ST0247

```

13         set[c] > 0 && set[d] > 0 &&
14         set[e] > 0){
15             total = total + 1;
16         }
17     }
18     return total;
19 }

```



De acuerdo a lo anterior, resuelvan lo siguiente:

#### 4.4.1 Completen la línea 8

---

#### 4.4.2 ¿Cuál es la complejidad del algoritmo?

- a.  $O(|N-M|)$
- b.  $O(|N-M|) \cdot \log_{10} M$
- c.  $O(|N-M| \times N)$
- d.  $O(|N-M| \times M)$

4.5

Tenemos un arreglo de  $n$  números no negativos  $\{x_1, x_2, x_3, \dots, x_n\}$ . Queremos determinar si es posible encontrar un  $i$ ,  $0 \leq i < n$  de tal manera que la sumatoria de los elementos del arreglo hasta la posición  $i$  (incluida) sea igual a la sumatoria de los elementos del arreglo desde la posición  $i+1$  (incluida) hasta el final del arreglo. Ayúdanos a resolver el problema.



**Como un ejemplo**, para  $x=\{3,4,7,1,8,1,1,2,2,1\}$ , la respuesta es verdadero y el  $i=3$ .

```

boolean sol(int[] x){
    boolean can = false;
    int left, right;
    right=left=0;

```

**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 2**  
**Código ST0247**

```
int n=x.length;
for(int i=0; i<n; i++){
    left = left+x[i];
    for(int j=.....; j<n; j++){
        right = right+x[j];
    }
    can = can || (.....);
    right = 0;
}
return can;
}
```



**De acuerdo a lo anterior, resuelvan lo siguiente:**

**4.5.1** Completen la línea 7

---

**4.5.2** Completen la línea 10

---

**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>



Lean a "**Anany Levitin, Introduction to the Design & Analysis of Algorithms Chapter 3: Brute Force and Exhaustive Search. Páginas 97 – 120**" 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 3.6, 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**



No apliquen **Normas Icontec** para esto

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



**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**



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](https://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 el Ejercicio 1.....	<b><u>Pág. 24</u></b>
Ayudas para el Ejercicio 2.1.....	<b><u>Pág. 24</u></b>
Ayudas para el Ejercicio 2.2.....	<b><u>Pág. 25</u></b>
Ayudas para el Ejercicio 2.3.....	<b><u>Pág. 25</u></b>
Ayudas para el Ejercicio 2.4.....	<b><u>Pág. 25</u></b>
Ayudas para el Ejercicio 3.4.....	<b><u>Pág. 25</u></b>
Ayudas para el Ejercicio 3.5.....	<b><u>Pág. 26</u></b>
Ayudas para el Ejercicio 5.....	<b><u>Pág. 26</u></b>
Ayudas para el Ejercicio 6.1.....	<b><u>Pág. 27</u></b>
Ayudas para el Ejercicio 6.2.....	<b><u>Pág. 27</u></b>
Ayudas para el Ejercicio 6.3.....	<b><u>Pág. 27</u></b>



## Ayudas para el Ejercicio 1



**Pista 1:** Genere todas las permutaciones de vértices



**Pista 2:** Evalúe de todas las permutaciones de vértices cuál es el ciclo hamiltoniano más corto



## Ayudas para el Ejercicio 2.1



**Pista 1:** Este algoritmo será muy lento si se hace con fuerza bruta. Usen una técnica de diseños de algoritmos que sea más eficiente para resolver este problema



**Pista 2:** Primero definan una estructura de datos para representar los huecos



**Pista 3:** Vean Guía en numeral 4.13 para “*Cómo usar Scanner o BufferedReader*”



## Ayudas para el 3.5



**Pista 1:** Vean Guía en numeral 4.11 para “*Cómo escribir la complejidad de un ejercicio 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



## Ayudas para el Ejercicio 4.0



**Pista 1:** Vean **Guía en numeral 4.18** para respuestas del Quiz



## Ayudas para el Ejercicio 2.4



**Pista 1:** Algoritmos para hallar componentes fuertemente conexos. Ordenamiento topológico. DFS. Léase en <http://bit.ly/2gTeJKh>



## 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://bit.ly/2QM9D3M>

## Ayudas para el Ejercicio 6.1



**Pista 1:** Vean **Guía en numeral 4.21** “Ejemplo de cómo hacer actas de trabajo en equipo usando Tablero Kanban”

## Ayudas para el Ejercicio 6.2



**Pista 1:** Vean **Guía en numeral 4.23** “Cómo generar el historial de cambios en el código de un repositorio que está en svn”

**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 6.3



**Pista 1:** Vean *Guía en numeral 4.22* “**Cómo ver el historial de revisión de un archivo en Google Docs**”

# ¿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>