**Laboratorio Nro. 2  
Fuerza Bruta (*Brute force* o *Exhaustive search*)**

## 

**Objetivo**

**1.** Diseñar algoritmos usando la técnica de diseño de fuerza bruta

**Consideraciones iniciales**

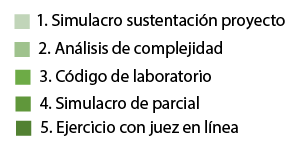
|  |  |
| --- | --- |
| guia-01**Leer la Guía** | Antes de comenzar a resolver el presente laboratorio, leer la ***“Guía Metodológica para la realización y entrega de laboratorios de Estructura de Datos y Algoritmos”*** que les orientará sobre los requisitos de entrega para este y todos los laboratorios, las rúbricas de calificación, el desarrollo de procedimientos, entre otros aspectos importantes. |
| **Registrar Reclamos**  boton-de-preguntas-frecuentes | En caso de tener **algún comentario** sobre la nota recibida en este u otro laboratorio, pueden **enviarlo** a través de [**http://bit.ly/2g4TTKf**](http://bit.ly/2g4TTKf), el cual será atendido en la menor brevedad posible. |
| **Traducción de Ejercicios**  Traducción-01 | En el GitHub del docente, encontrarán la traducción al español de los enunciados de los Ejercicios en Línea. |
| **Visualización de Calificaciones**  **examen-01** | A través de ***Eafit Interactiva*** encontrarán **un enlace** que les permitirá **ver un registro de las calificaciones** que **emite el docente** para cada taller de laboratorio y según las rubricas expuestas. ***Véase sección 3, numeral 3.7.*** |
| **GitHub** | **1.** Crear un repositorio en su cuenta de GitHub con el nombre st0247. **2.** Crear una carpeta dentro de ese repositorio con el nombre laboratorios. **3.** Dentro de la carpeta laboratorio, crear una carpeta con nombre lab02. **4.** Dentro de la carpeta lab02, crear tres carpetas: informe, codigo y ejercicioEnLinea. **5.**Subir el informe pdf a la carpeta infome, el código del ejercicio 1 a la carpeta codigo y el código del ejercicio en línea a la carpeta ejercicioEnLinea. Así:  st0247  laboratorios  lab01  informe  codigo  ejercicioEnLinea  lab02  ... |

**Intercambio de archivos**

Los archivos que **ustedes deben entregar** al docente son: **un archivo PDF** con el informe de laboratorio usando la plantilla definida, y **dos códigos**, uno con la solución al numeral 1 y otro al numeral 2 del presente. Todo lo anterior se entrega en **GitHub**.



**Porcentajes y criterios de evaluación para el laboratorio**



**Resolver Ejercicios**

1. **Códigos para entregar en GitHub:**

|  |  |
| --- | --- |
| ***Vida  Real-01*** | **En la vida real, la documentación del software hace parte de muchos estándares de calidad como CMMI e ISO/IEC 9126** |

|  |  |  |  |
| --- | --- | --- | --- |
| **guia-01** | Véase Guía ***en Sección 3, numeral 3.4*** |  | *Código de laboratorio en* ***GitHub.*** Véase Guía en **Sección 4, numeral 4.24** |
| **No HTML** | Es opcional entregar documentación. Si lo hace, utilice **Javadoc** o equivalente. No suba el HTML a GitHub. | **no rar** | ***No******se reciben*** *archivos en* ***.RAR*** *ni en* ***.ZIP*** |

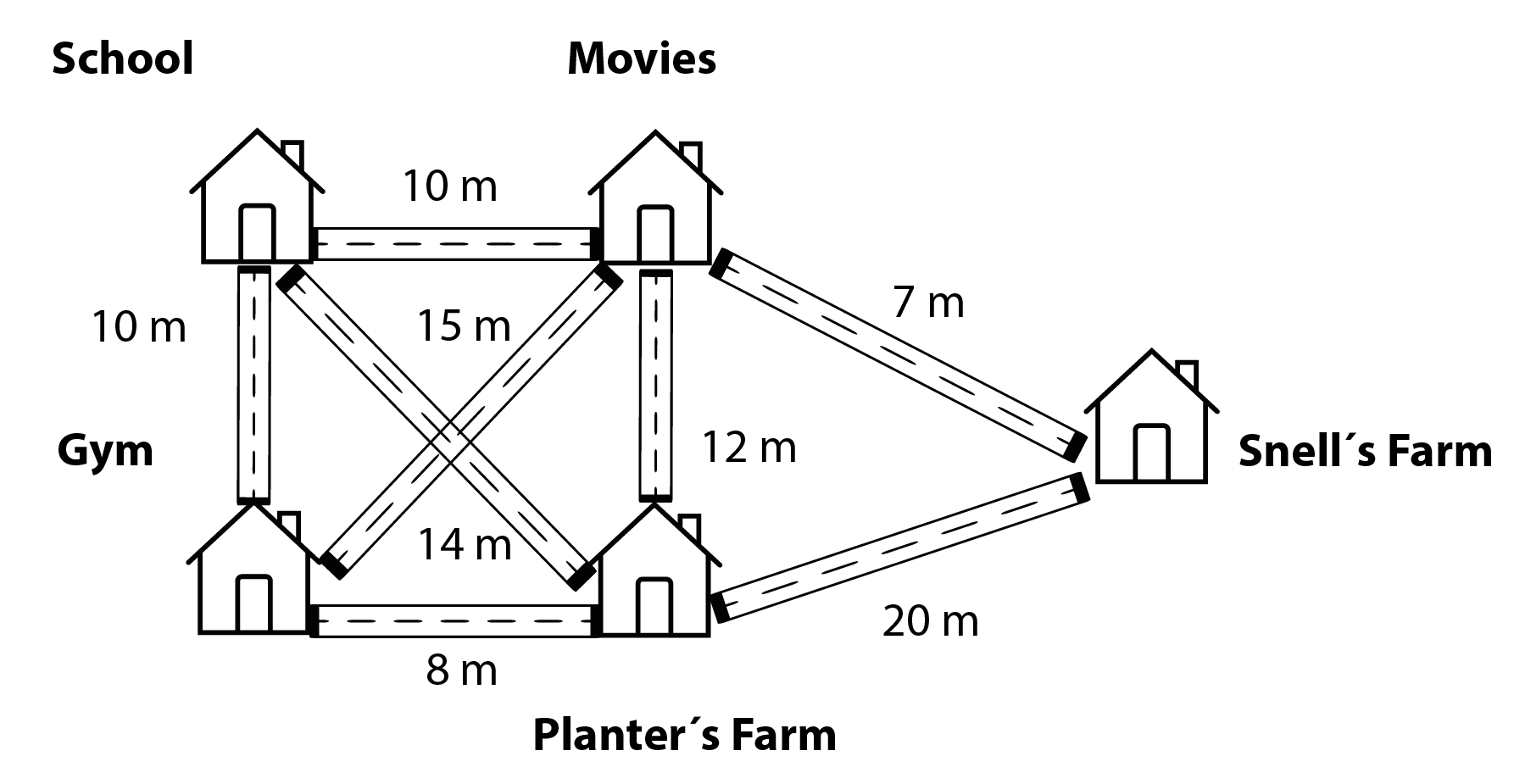
***1. Una de las funciones sustanciales del grupo EMI es brindar el servicio médico en casa. Un problema que enfrenta*** ***actualmente EMI es encontrar el orden óptimo para visitar a sus clientes de tal forma que se minimice la distancia total del recorrido que tiene que hacer cada uno de sus médicos.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

C:\Users\Luisa Alzate\AppData\Local\Microsoft\Windows\INetCache\Content.Word\triangulo_ED2-01.png***Dado un grafo dirigido completo, 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.***

**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.**



**Como un 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

**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*

**2) Ejercicios en línea sin documentación HTML en GitHub**

|  |  |  |  |
| --- | --- | --- | --- |
| **guia-01** | Véase Guía en ***Sección 3, numeral 3.3*** | **No HTML** | **No entregar** documentación **HTML** |
| **C:\Users\Luisa Alzate\AppData\Local\Microsoft\Windows\INetCache\Content.Word\java.png** | *Entregar un archivo en* ***.JAVA*** | **no pdf** | ***No******se reciben*** *archivos en* ***.PDF*** |
|  |  |  | *Código del ejercicio en línea en* ***GitHub.*** Véase Guía en **Sección 4, numeral 4.24** |

|  |  |
| --- | --- |
| alerta-simbolo-dibujado-a-mano | **NOTA 1:**Recuerden que, **si toman la respuesta de alguna fuente**, deben **referenciar** según el **tipo de cita** correspondiente***.*** Véase *Guía en* ***Sección 4, numerales 4.16 y 4.17*** |

**2.1 Resuelvan el siguiente problema:**

|  |
| --- |
| 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 *nxn* 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.  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 < n < 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  **Entrada de ejemplo**  8  ........  ........  ........  ........  ........  ........  ........  ........  4  .\*..  ....  ....  ....  0  **Entrada de ejemplo**  Case 1: 92  Case 2: 1 |

**2.2 [Ejercicio Opcional] Resuelvan el siguiente problema** [**http://bit.ly/2gTLZ53**](http://bit.ly/2gTLZ53)

**2.3 [Ejercicio opcional]** **Resuelvan el siguiente ejercicio** [**http://bit.ly/2hGqJPB**](http://bit.ly/2hGqJPB)

**2.4 [Ejercicio Opcional]** **Resuelvan el siguiente ejercicio** [**http://bit.ly/2hrrCfS**](http://bit.ly/2hrrCfS)

**2.5 [Ejercicio Opcional] Resuelvan el siguiente ejercicio** [**http://bit.ly/2k8CGSG**](http://bit.ly/2k8CGSG)

**3 Simulacro de preguntas de sustentación de Proyectos**

|  |  |  |  |
| --- | --- | --- | --- |
| **guia-01** | Véase Guía en ***Sección 3, Numeral 3.5*** | **C:\Users\Luisa Alzate\AppData\Local\Microsoft\Windows\INetCache\Content.Word\pdf-simbolo-de-formato-de-archivo.png** | *Entregar informe de laboratorio*en **PDF** |
| **C:\Luisa Alzate\2. Laboral\Informal\Mauricio Toro\Iconos y Logos\papel.png** | *Usen la* ***plantilla*** *para responder laboratorios* | ***ICONTECr*** | ***No*** *apliquen* ***Normas Icontec*** *para esto* |

|  |  |
| --- | --- |
| ***Vida  Real-01*** | **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.***

**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**.

|  |  |
| --- | --- |
| alerta-simbolo-dibujado-a-mano | **NOTA 1:** Recuerden que debe explicar su implementación en el informe PDF |

**3.5 Calculen la complejidad del ejercicio 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 3.5. 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 el informe PDF**

|  |  |  |  |
| --- | --- | --- | --- |
| **hoja-de-papel-01** | Para este simulacro, agreguen ***sus respuestas* en el informe PDF.**  Resuelva, como mínimo, los ejercicios marcados con color rojo. | **ordenador-peral-01** | *El* ***día del Parcial no tendrán computador, JAVA o acceso a internet.*** |

|  |
| --- |
| **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.  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, resuelva lo siguiente:   1. (10%) Completen el espacio en línea 07   \_\_\_\_\_\_\_\_\_\_\_\_   1. (10%) ¿Cuál es la complejidad asintótica, para el peor de los casos, del algoritmo? O(\_\_\_\_)   **2.** 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 estaOrdenado 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.    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:   1. (10%) Completen los espacios vacíos en el llamado recursivo del método ordenar   \_\_\_\_\_\_\_\_\_\_\_\_, \_\_\_\_\_\_\_\_\_\_\_\_   1. (10 %) Completen la complejidad, en el peor de los casos, del método ordenar   O(\_\_\_\_)  **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.  **Como un 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 }  **3.1** Completen la línea 12  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  **3.2** Completen la línea 13  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  **3.3** ¿Cuál es la complejidad asintótica, para el peor de los casos, del algoritmo? (En términos de *n* y *m*)  O(\_\_\_\_)  **PISTA:** NO es *O*(*n*!).  **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≤*a*,*b*,*c*,*d*,*e*≤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 es el conjunto de todos los dígitos de algún numero *N*≤*x*≤*M* y *T*={*a*,*b*,*c*,*d*,*e*}, siempre se cumple que *S*∩*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 &&  13 set[c] > 0 && set[d] > 0 &&  14 set[e] > 0){  15 total = total + 1;  16 }  17 }  18 return total;  19 }  **4.1** Complete la línea 8 ..............  **4.2** ¿Cuál es la complejidad del algoritmo?  (a) *O*(|*N*−*M*|)  (b)  (c) *O*(|*N*−*M*|×*N*)  (d) *O*(|*N*−*M*|×*M*)  **5.** Considere el siguiente problema. Tenemos un arreglo de *n* números no negativos . Queremos determinar si es posible encontrar un *i*,0≤*i*<*n* de tal manera que . Ayúdanos a resolver este 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;  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;  }  **5.1.** Complete la línea 7 ...........  **5.2.** Complete la línea 10 .......... |

**5) [Ejercicio Opcional] Lectura recomendada**

|  |  |
| --- | --- |
| ***Vida  Real-01*** | **"Quienes se preparan para el ejercicio de una profesión requieren la adquisición de competencias que necesariamente 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**](http://bit.ly/2gJKzJD) |

|  |  |
| --- | --- |
| guia-01 | *Véase Guía en* ***Sección 3, numeral 3.5 y 4.20*** *de la Guía Metodológica, “Lectura recomendada” y “Ejemplo para realización de actividades de las Lecturas Recomendadas*”, respectivamente |

Posterior a la lectura del texto ***Anany Levitin, Introduction to the  Design & Analysis of Algorihtms Chapter 3: Brute Force and Exhaustive Search. Páginas 97 – 120***.”***,*** realicen las siguientes actividades que les permitirán sumar puntos adicionales:

1. Escriban un resumen de la lectura que tenga una longitud de 100 a 150 palabras
2. Hagan un mapa conceptual que destaque los principales elementos teóricos.

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

|  |  |
| --- | --- |
| ***Vida  Real-01*** | **El trabajo en equipo es una exigencia actual del mercado. "Mientras algunos medios retratan la programación como un trabajo solitario, la realidad es que requiere de mucha comunicación y trabajo con otros. Si trabajas para una compañía, serás parte de un equipo de desarrollo y esperarán que te comuniques y trabajes bien con otras personas”**  **Tomado de** [**http://bit.ly/1B6hUDp**](http://bit.ly/1B6hUDp) |

|  |  |
| --- | --- |
| guia-01 | *Véase Guía en* ***Sección 3, numeral 3.6*** *y* ***Sección 4, numerales 4.21, 4.22*** *y* ***4.23*** *de la Guía Metodológica* |

1. Entreguen copia de todas las actas de reunión usando el tablero Kanban, con fecha, hora e integrantes que participaron
2. Entreguen el reporte de *git*, *svn* o *mercuria*l con los cambios en el código y quién hizo cada cambio, con fecha, hora e integrantes que participaron
3. Entreguen el reporte de cambios del informe de laboratorio que se genera *Google docs* o herramientas similares

**7. [Ejercicio Opcional] Laboratorio en inglés:**

|  |  |
| --- | --- |
| ***Vida  Real-01*** | **El inglés es un idioma muy 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 y es sólo un resumen de la información original.**  **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 traducido al inglés. Utilicen la plantilla dispuesta en este idioma para el laboratorio**

**Ayudas para resolver los ejercicios**

|  |  |
| --- | --- |
| **Ayudas para el Ejercicio 1.2…………….** | [**Pág. 18**](#Ejer12) |
| **Ayudas para el Ejercicio 2.1…………….** | [**Pág. 18**](#Ejer21) |
| **Ayudas para el Ejercicio 3.1…………….** | [**Pág. 18**](#Ejer31) |
| **Ayudas para el Ejercicio 3.2…………….** | [**Pág. 18**](#Ejer32) |
| **Ayudas para el Ejercicio 3.5…………….** | [**Pág. 19**](#Ejer35) |
| **Ayudas para el Ejercicio 4.0…………….** | [**Pág. 19**](#Ejer40) |
| **Ayudas para el Ejercicio 5A…………….** | [**Pág. 19**](#Ejer5A) |
| **Ayudas para el Ejercicio 5B…………….** | [**Pág. 19**](#Ejer5B) |
| **Ayudas para el Ejercicio 6A…………….** | [**Pág. 20**](#Ejer6A) |
| **Ayudas para el Ejercicio 6B…………….** | [**Pág. 20**](#Ejer6B) |
| **Ayudas para el Ejercicio 6C…………….** | [**Pág. 20**](#Ejer6C) |

**Ayudas para el** **Ejercicio 1**

|  |  |
| --- | --- |
| clave | **PISTA 1:** *Genere todas las permutaciones de vértices* |
| clave | **PISTA 2:** Evalúe de todas las permutaciones de vértices cuál es el ciclo hamiltoniano más corto |

**Ayudas para el** **Ejercicio 2.1**

|  |  |
| --- | --- |
| clave | **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 |

|  |  |
| --- | --- |
| clave | **PISTA 2:** Primero definan una estructura de datos para representar los huecos |
| clave | **PISTA 3:** Véase Guía en **Sección 4, numeral 4.13** *“Cómo usar Scanner o BufferedReader”* |

**Ayudas para el Ejercicio 3.5**

|  |  |
| --- | --- |
| clave | **PISTA:** Véase***Guía en Sección 4, numeral 4.11*** *“Cómo escribir la complejidad de un ejercicio en línea”* |

**Ayudas para el Ejercicio 4.0**

|  |  |
| --- | --- |
| clave | **PISTA 1:** *Véase* ***Guía en Sección 4, Numeral 4.18*** “Respuestas del Quiz” |
| clave | **PISTA 2:** Lean las diapositivas tituladas “***Data Structures II: Brute Force”***, encontrarán la mayoría de las respuestas |

**Ayudas para el** **Ejercicio 5A**

|  |  |
| --- | --- |
| clave | **PISTA 1:** En el siguiente enlace, unos consejos de cómo hacer un buen resumen[**http://bit.ly/2knU3Pv**](http://bit.ly/2knU3Pv) |
| clave | **PISTA 2:** [**Aquí**](http://bit.ly/28JXqIG)le explican cómo contar el número de palabras en Microsoft Word |

**Ayudas para el** **Ejercicio 5B**

|  |  |
| --- | --- |
| clave | **PISTA 1:** Para que hagan el mapa conceptual se recomiendan herramientas como las que encuentran en [**https://cacoo.com/**](https://cacoo.com/)o **[https://www.mindmup.com/#m:new-a-1437527273469](https://www.mindmup.com/" \l "m:new-a-1437527273469)** |

|  |  |
| --- | --- |
| alerta-simbolo-dibujado-a-mano | **NOTA 1:**Si desean una lectura adicional en inglés, consideren la siguiente:**“Skiena, The algorithm design manual (2nd edition), Section 14.4. 2010***”,* que pueden encontrarla en biblioteca. |
| clave | **NOTA 2:** Estas respuestas también deben incluirlas en el informe PDF |

**Ayudas para el** **Ejercicio 6A**

|  |  |
| --- | --- |
| clave | **PISTA 1:** *Véase* ***Guía en Sección 4, Numeral 4.21*** “*Ejemplo de cómo hacer actas de trabajo en equipo usando Tablero Kanban”* |

**Ayudas para el** **Ejercicio 6B**

|  |  |
| --- | --- |
| clave | **PISTA 1: *Véase*** *Guía en Sección 4, Numeral 4.23*“**Cómo generar el historial de cambios en el código de un repositorio que está en svn”** |

**Ayudas para el** **Ejercicio 6C**

|  |  |
| --- | --- |
| clave | **PISTA 1: *Véase*** *Guía en Sección 4, Numeral 4.22* “***Cómo ver el historial de revisión de un archivo en Google Docs*”** |
| alerta-simbolo-dibujado-a-mano | **NOTA:** Estas respuestas también deben incluirlas en el informe PDF |