Laboratorio Nro. 4
Algoritmos voraces o codiciosos
(Greedy algorithms)

Objetivos

Diseñar algoritmos usando la técnica de algoritmos voraces o codiciosos *(Greedy algorithms)*

Consideraciones iniciales

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 	En caso de tener **algún comentario** sobre la nota recibida en este u otro laboratorio, pueden **enviarlo** a través de **http://bit.ly/2g4TTKf**, el cual será atendido en la menor brevedad posible.

DOCENTE MAURICIO TORO BERMÚDEZ
Teléfono: (+57) (4) 261 95 00 Ext. 9473. Oficina: 19 - 627
Correo: mtorobe@eafit.edu.co

Traducción de Ejercicios

En el GitHub del docente, encontrarán la traducción al español de los enunciados de los Ejercicios en Línea.

Visualización de Calificaciones

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

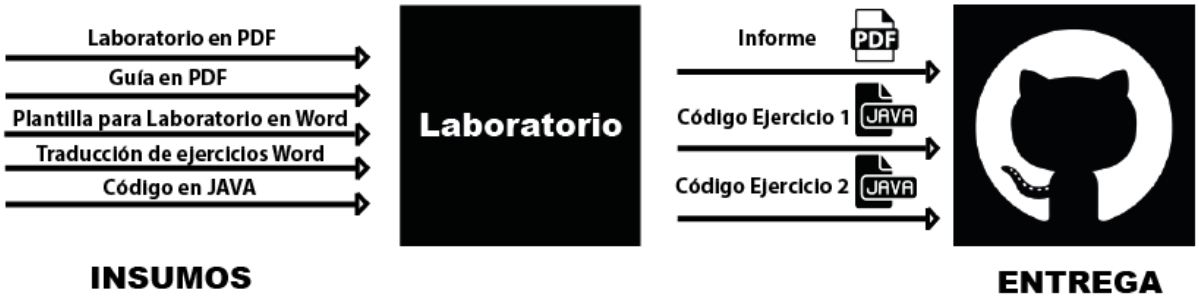
1. Crear un repositorio en su cuenta de GitHub con el nombre `st0247-suCodigoAqui`. **2.** Crear una carpeta dentro de ese repositorio con el nombre `laboratorios`. **3.** Dentro de la carpeta laboratorio, crear una carpeta con nombre `lab04`. **4.** Dentro de la carpeta `lab04`, crear tres carpetas: `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-suCodigoAqui
  laboratorios
      lab01
         informe
         codigo
         ejercicioEnLinea
      lab02
         ...
```

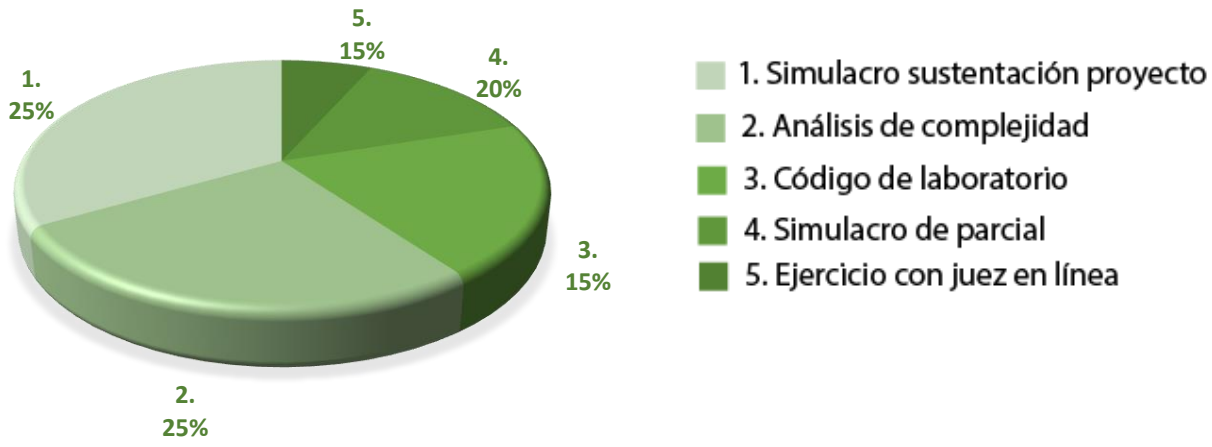
DOCENTE MAURICIO TORO BERMÚDEZ
Teléfono: (+57) (4) 261 95 00 Ext. 9473. Oficina: 19 - 627
Correo: mtorobe@eafit.edu.co

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



DOCENTE MAURICIO TORO BERMÚDEZ
Teléfono: (+57) (4) 261 95 00 Ext. 9473. Oficina: 19 - 627
Correo: mtorobe@eafit.edu.co

Resolver Ejercicios

1. Códigos para entregar en GitHub:

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

Véase Guía ***en Sección 3, numeral 3.4***

*Código de laboratorio en **GitHub.*** Véase Guía en **Sección 4, numeral 4.24**

*Entregar documentación en **JavaDoc** o equivalente. El uso de JavaDoc es opcional.*

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

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

1.1 Teniendo en cuenta lo anterior, realicen una implementación de la solución al problema del agente viajero usando un algoritmo voraz.

.

En la vida real, una exigencia del mercado es que los ingenieros de sistemas conozcan la metodología de Desarrollo de Software dirigido por *testing* (en Inglés *TDD*), en la que primero se hacen los *tests* unitarios antes de empezar la programación

1.2 Prueben su programa usando el ejemplo descrito en el *"Taller en clase sobre algoritmos voraces"*.

En las respuestas de ese taller se encuentran ambas soluciones para ese grafo: la óptima y la voraz. Igualmente están los costos. Realicen una prueba unitaria usando JUnit o su equivalente en C++ o Python.

2) Ejercicios en línea sin documentación HTML en GitHub:

	Véase Guía en ***Sección 3, numeral 3.3***		**No entregar** documentación **HTML**
	*Entregar un archivo en **.JAVA***		***No se reciben** archivos en **.PDF***
	***Resolver** los problemas de **CodingBat** usando **Recursión***		*Código del ejercicio en línea en **GitHub.*** Véase Guía en **Sección 4, numeral 4.24**

DOCENTE MAURICIO TORO BERMÚDEZ
Teléfono: (+57) (4) 261 95 00 Ext. 9473. Oficina: 19 - 627
Correo: mtorobe@eafit.edu.co

2.1 Resuelvan el siguiente problema usando algoritmos voraces:

En una ciudad hay *n* conductores de bus. También hay *n* rutas de bus en la mañana y *n* rutas de bus en la tarde con varias duraciones. Cada conductor es asignado una ruta en la mañana y una ruta en la noche.

Para cada conductor, si la duración total de su ruta por un día excede *d*, él tiene que ser pagado por el tiempo extra por cada hora después de su básico a una tarifa de *r* pesos por hora.

Su tarea es asignar una ruta en la mañana y una ruta en la tarde a cada conductor de tal forma que el tiempo total extra que haya que pagar por parte de la empresa sea el mínimo.

Entrada

La línea de cada caso de prueba tiene 3 enteros, *n, d y r,* como descritores anteriormente. En la segunda línea, hay n enteros separados con espacios, que son las duraciones de la rutas de la mañana dadas en horas.

De igual manera, en la tercera línea hay *n* enteros separados por espacio que son las duraciones, en hroas, de las rutas de la tarde. Las duraciones son enteros positivos menores o iguales a 10.000. El fin de la entrada está dado por un caso con tres ceros.

Salida

Para cada caso de prueba, imprima el mínimo posible valor de horas extras que la empresa de transporte debe pagar.

Restricciones

- $1 < n < 100$
- $1 < d < 10000$
- $1 < r < 5$

Ejemplo de una entrada

```
2 20 5
10 15
10 15
2 20 5
10 10
10 10
0 0 0
```

Ejemplo de una entrada

```
50
0
```

3) Simulacro de preguntas de sustentación de Proyectos

Véase Guía en ***Sección 3, Numeral 3.4***

Entregar informe de laboratorio en **PDF**

*Usen la **plantilla** para responder laboratorios*

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

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

NOTA 1: Recuerden que debe explicar su implementación en el informe PDF

3.2 Para resolver el problema del agente viajero, usando un algoritmo voraz, aun cuando no arroje la solución óptima, ¿Qué debe cumplir el grafo para que el algoritmo, al menos, arroje una solución, así no sea óptima? ¿Por qué?

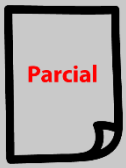
3.3 Expliquen con sus propias palabras la estructura de datos que utiliza para resolver el problema del numeral 2.1 y cómo funciona el algoritmo

NOTA 1: Recuerden que debe explicar su implementación en el informe PDF

***3.4* Calculen la complejidad del ejercicio trabajado en el numeral 2.1 y agréguenla al informe PDF**

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

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***.

1. El problema de **selección de actividades** consiste en encontrar el máximo número de actividades que una persona o una máquina puede resolver, asumiendo que la persona o máquina sólo puede hacer una actividad al mismo tiempo.

Este problema es de interés en Ingeniería de Producción. Cada actividad tiene un tiempo de inicio y un tiempo de fin. Como un ejemplo, consideren las siguientes actividades:

Act.	a1	a2	a3	a4	a5	a6	a7	a8
inicio	1	0	1	4	2	5	3	4
fin	3	4	2	6	9	8	5	5

Nuestra tarea es encontrar el máximo número de actividades que se puedan ejecutar y que no estén en conflicto (es decir, que no ocurran dos al mismo tiempo).

Un algoritmo voraz para resolver el problema funciona de la siguiente forma:

a) Ordenar las actividades de menor a mayor, según el tiempo de fin de la actividad

b) Seleccionar la primera actividad

c) Repetir hasta que no se puedan seleccionar más actividades:

- ☑ Seleccionar una nueva actividad cuyo tiempo de inicio sea mayor igual al tiempo de fin de la actividad previamente seleccionado.

Para el ejemplo anterior, el algoritmo debe entregar la siguiente respuesta:

Actividad	Inicio	Fin
a3	1	2
a7	3	5
a6	5	8

A continuación, observamos una implementación del algoritmo en Java:

```
class Actividad {
  public Actividad(String a,int b,int c) {
      id = a; inicio = b; fin = c;}
  String id;
  int inicio;
  int fin;
}
void seleccion(Actividad actividades[]) {
  int i, j;
  int n = actividades.length;
  Actividad temp;
  //paso 1
  for(i = 1; i < n; i++) {
    for(j = 0; j < n - 1; j++){
      if(actividades[j].fin > actividades[j+1].fin) {
        temp = actividades[j];
        actividades[j] = actividades[j+1];
        actividades[j+1] = temp;
      }
    }
  }
  //paso 2
  System.out.println("Actividad Inicio Fin");
  System.out.println(actividades[0].id + " " +
                     actividades[0].inicio + " " +
                     actividades[0].fin);
  //paso 3
  i = 0;
  for(j = 1; j < n; j++) {
    if(actividades[j].inicio >= actividades[i].fin) {
      System.out.println(actividades[j].id + " " +
                         actividades[j].inicio + " " +
                         actividades[j].fin);

      _________
    }
  }
```

d) Completen el espacio faltante en la última línea

_____

DOCENTE MAURICIO TORO BERMÚDEZ
Teléfono: (+57) (4) 261 95 00 Ext. 9473. Oficina: 19 - 627
Correo: mtorobe@eafit.edu.co

2. El problema del **agente viajero** consiste en responder la siguiente pregunta: Dada una lista de ciudades y las distancias entre cada par de ciudades, ¿cuál es la ruta posible más corta que visita a cada ciudad exactamente una vez y regresa a la ciudad de origen?

Un algoritmo voraz para solucionar este problema es el **algoritmo del vecino más cercano**. El algoritmo empieza en la primera ciudad y selecciona en cada iteración una nueva ciudad, no visitada, que sea la más cercana a la inmediatamente anterior.

A continuación, una implementación del algoritmo del vecino más cercano que recibe como parámetro un grafo representado como matriz de adyacencia.

La persona que hizo este código es un ingeniero matemático. En Matlab los índices empiezan en 1. Por esta razón, el camino que encuentra el algoritmo empieza con la ciudad numerada con 1 y trabaja con ciclos `while` en lugar de `for`.

Adicionalmente, el programador inicia los índices en 1, por ejemplo `i = 1`. Por si fuera poco, la variable `min` no era necesaria inicializarla fuera del bloque `while` y `minFlag` hubiera sido mejor declararla dentro del bloque `while`.

Finalmente, el arreglo de visitados sería más eficiente haberlo hecho con tipo `boolean`. No obstante, el programa funciona en Java y tiene una complejidad de , que es la esperada para este algoritmo.

```
01 public void tsp(int adjacencyMatrix[][]) {
02  Stack<Integer> stack = new Stack<Integer>();
03  int numberOfNodes = adjacencyMatrix[1].length - 1;
04  int[] visited = new int[numberOfNodes + 1];
05  visited[1] = 1;
06  stack.push(1);
07  int element, dst = 0, i;
08  int min = Integer.MAX_VALUE;
09  boolean minFlag = false;
10  System.out.print(1 + "\t");
11  while (!stack.isEmpty()) {
12   element = stack.peek();
13   i = 1;
14   min = Integer.MAX_VALUE;
15   while (i <= numberOfNodes) {
16    if (adjacencyMatrix[element][i] > 0
17        && visited[i] == 0) {
18         if (_______> ______) {
```

DOCENTE MAURICIO TORO BERMÚDEZ
Teléfono: (+57) (4) 261 95 00 Ext. 9473. Oficina: 19 - 627
Correo: mtorobe@eafit.edu.co

```
19          min = adjacencyMatrix[element][i];
20          dst = i;
21          minFlag = true;
22         }
23     }
24     i++;
25    }
26    if (minFlag) {
27     visited[dst] = 1;
28     stack.push(dst);
29     System.out.print(dst + "\t");
30     minFlag = false;
31     continue;
32    }
33    stack.pop();
34   }
}
```

`Continue` es una palabra reservada en Java que permite terminar una iteración de un ciclo abruptamente y pasar a la siguiente iteración del ciclo.

a) (10%) Completen el espacio en la línea 18

_____ > _____

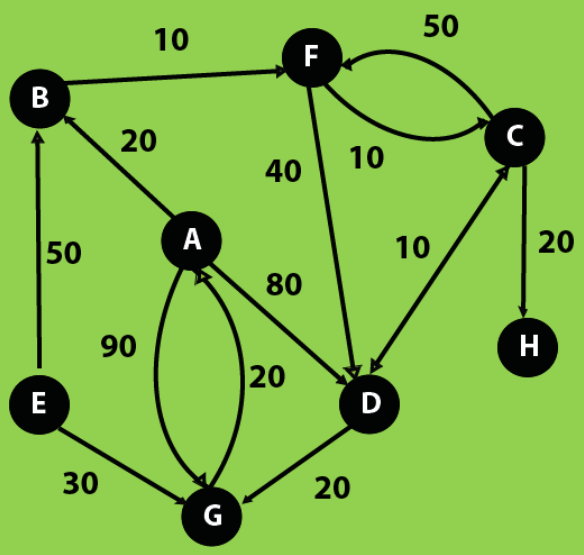
3. El algoritmo de Dijkstra sirve para encontrar el camino más corto de un vértice a todos los demás de un grafo. A continuación, una implementación en Java.

```
int minVertex (int [] dist, boolean [] v) {
 int x = Integer.MAX_VALUE; //Infinity
 int y = -1;
 for (int i=0; i<dist.length; i++)
  if (!v[i] && dist[i]<x)
     y=i; x=dist[i];
 return y;
}

int [] dijsktra(Graph dg, int source) {
  int [] dist = new int [dg.size()];
  int [] pred = new int [dg.size()];
  boolean [] visited = new boolean [dg.size()];
  for (int i=0; i<dist.length; i++)
```

```
    dist[i] = Integer.MAX_VALUE;
  dist[source] = 0;
  for (int i=0; i<dist.length; i++) {
    int next = minVertex (dist, visited);
    visited[next] = true;
    ArrayList<Integer> n =
        dg.getSuccessors (next);
    for (int j=0; j<n.size(); j++) {
      int v = n.get(j);
      int d = dist[next] +
          dg.getWeight(next,v);
      if (dist[v] > d) {
        dist[v] = d;
        pred[v] = next;
      }}}
  return pred;
}
```

Consideren el siguiente grafo de Dijkstra:



a) (20 %) Completen, por favor, la siguiente tabla, usando el algoritmo de Dijkstra para encontrar el camino más corto del punto A a todos los demás. En la tabla, la palabra "a" significa "hasta".

Paso	a	B	C	D	E	F	G	H
1	A	20,A	∞	80, A	∞	∞	90, A	∞
2	B	20,A	∞	80, A	∞	30,B	90,A	∞
3								
4								
5								
6								
7								
8								

b) (10 %) ¿Cuál es el camino más corto de A a G?

_____

5. [Ejercicio Opcional] Lectura recomendada

"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

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 ***"R.C.T Lee et al., Introducción al análisis y diseño de Algoritmos. Capítulo 3. Páginas 71 - 115.",*** realicen las siguientes actividades que les permitirán sumar puntos adicionales:

a) Escriban un resumen de la lectura que tenga una longitud de 100 a 150 palabras

b) Hagan un mapa conceptual que destaque los principales elementos teóricos.

NOTA 1: Si desean una lectura adicional en español, consideren la siguiente: ***"John Hopcroft et al., Estructuras de Datos y Algoritmos, Sección 10.3. 1983"***, que encuentran en biblioteca.

NOTA 2: Estas respuestas también deben incluirlas en el informe PDF

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

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

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

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

b) 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

c) 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

7. [Ejercicio Opcional] Laboratorio en inglés

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

Resumen de ejercicios a resolver

1.1 Realicen una implementación de la solución al problema del agente viajero usando un algoritmo voraz

1.2 Prueben su programa usando el ejemplo descrito en el *"Taller en clase sobre algoritmos voraces"*

2.1 Resuelvan el problema usando algoritmos voraces

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

3.2 Para resolver el problema del agente viajero, usando un algoritmo voraz, aun cuando no arroje la solución óptima, **¿Qué debe cumplir el grafo para que el algoritmo, al menos, arroje una solución, así no sea óptima? ¿Por qué?**

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

3.4 Calculen la complejidad del ejercicio trabajado en el numeral 2.1 y agréguenla al informe PDF

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

4. Simulacro de Parcial

5. Lectura recomendada [Ejercicio Opcional]

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

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

DOCENTE MAURICIO TORO BERMÚDEZ
Teléfono: (+57) (4) 261 95 00 Ext. 9473. Oficina: 19 - 627
Correo: mtorobe@eafit.edu.co

Ayudas para resolver los ejercicios

Ayudas para el Ejercicio 1

PISTA 1: Si deciden hacer la documentación, consulten la *Guía en* ***Sección 4, numeral 4.1*** *"Cómo escribir la documentación HTML de un código usando JavaDoc"*

Ayudas para el Ejercicio 2.1

PISTA 1: Construyan un algoritmo voraz para resolver el problema

Ayudas para el Ejercicio 3.2

PISTA 1: Léase **http://bit.ly/1ROGWOX**

Ayudas para el Ejercicio 3.4

PISTA 1: Véase ***Guía en Sección 4, numeral 4.11*** *"Cómo escribir la complejidad de un ejercicio en línea"*

Ayudas para el Ejercicio 4

PISTA 1: *Véase* ***Guía en Sección 4, Numeral 4.18*** "Respuestas del Quiz"

PISTA 2: Lean las diapositivas tituladas *"****Data Structures II: Greedy algortihms",*** encontrarán la mayoría de las respuestas

Ayudas para el Ejercicio 5a

PISTA 1: En el siguiente enlace, unos consejos de cómo hacer un buen resumen **http://bit.ly/2knU3Pv**

PISTA 2: **Aquí** les explican cómo contar el número de palabras en Microsoft Word

Ayudas para el Ejercicio 5b

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 6a

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

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

PISTA 1: ***Véase*** *Guía en Sección 4, Numeral 4.22* "***Cómo ver el historial de revisión de un archivo en Google Docs*"**