

# Estructura de Datos

## EDD Learning

---

### Objetivo General

- Aplicar los conocimientos del curso de Estructura de Datos en la creación de soluciones de software.

### Objetivos Específicos

- Aplicar los conocimientos de estructura de datos no lineales
- Aplicar los conocimientos de estructura de datos en el lenguaje de programación de Java
- Utilizar la herramienta graphviz para la generación de reportes gráficos.
- Familiarizarse con el desarrollo de aplicaciones para uso didáctico

### Enunciado

Como estudiante de Ingeniería en Ciencias y Sistemas de la Universidad de San Carlos se le ha pedido que realice una aplicación que permita facilitar la enseñanza de los conceptos sobre el uso de las diferentes estructuras de datos. El objetivo de la aplicación es que pueda utilizarse como una herramienta de apoyo didáctico para que los estudiantes que lleven el curso de Estructura de Datos puedan comprender de una forma visual (utilizando animaciones y gráficos) el funcionamiento de los diferentes algoritmos que pueden ejecutarse sobre las estructuras impartidas en el curso. Esta herramienta tiene que poder ejecutarse en diferentes sistemas operativos por lo que se le indica que debe desarrollarse en el lenguaje de programación de Java, utilizando un entorno gráfico y amigable para el usuario.

## Descripción de la Aplicación

### Módulo de Seguridad

#### Usuarios


La aplicación debe de soportar el manejo de diferentes usuarios, cada uno de estos usuarios podrán loguearse a la aplicación para poder utilizarla. Como medida de seguridad se debe de tener un usuario administrador que pueda crear los usuarios permitidos para el uso de la aplicación. Los usuarios serán cargados por medio de un archivo JSON, **no existirá otra forma de cargar los usuarios a la aplicación, solamente utilizando el archivo JSON.** El usuario administrador es el único que puede cargar usuarios y realizar las operaciones de: edición (edición de atributos del usuario) y eliminación (eliminación de usuarios de la estructura) sobre los usuarios cargados. Un usuario tiene los siguientes atributos:

- Nombre
- Apellido
- Carnet
- Password (debe encriptar usando sha256)

Para realizar el proceso de autenticación para el inicio de sesión en la aplicación se utilizarán el carnet y el password, y al momento de entrar a la aplicación esta debe de mostrar los datos (nombre y apellido) del usuario que ha iniciado sesión. Al momento de realizar la carga de usuarios, la aplicación debe de validar que no se ingresen usuarios repetidos (número de carnet) y que al menos la contraseña tenga ocho caracteres, todos aquellos usuarios que no se logren cargar por no cumplir las especificaciones, deben de mostrarse en una tabla en la aplicación junto con la razón del porqué no se logró insertar el usuario.

La estructura que maneja los usuarios de la aplicación será una **tabla hash**, que seguirá las siguientes especificaciones:

- Tendrá un tamaño inicial de 37 (size)
- La función de dispersión a utilizar será la siguiente:  **$f(\text{carnet}) = \text{carnet} \bmod \text{size}$**
- El porcentaje máximo de utilización será del 55%
- Para la resolución de colisiones se utiliza una doble dispersión utilizando la siguiente función:  **$f(\text{carnet}, i) = (\text{carnet} \bmod 7 + 1) * i$**
- Al momento de la estructura llegar al 55% de utilización esta debe de crecer (size) hasta el siguiente número primo. Ejemplo: al inicio la estructura tiene un tamaño de 37, al llegar



esta al 55% de utilización esta debería de redimensionarse hasta el tamaño de 41. Esta redimensión debe darse cada vez que se llegue al 55% de utilización.

- Se debe de tener una opción de mostrar el reporte de la tabla mediante graphviz indicando la llave de cada uno de los elementos y los datos del usuario.

## Módulo de Aprendizaje

Este módulo de la aplicación es el más importante ya que será la funcionalidad de la aplicación que permitirá al usuario interactuar con las diferentes estructuras de datos y explorar su funcionalidad. La funcionalidad de las estructuras de datos deben de mostrarse de forma gráfica y dinámica para que el aprendizaje del estudiante sea efectivo.

### Ejecución de Algoritmos

Para que el aprendizaje sea efectivo en el estudiante, la aplicación debe de tener dos modalidades para ejecutar los algoritmos:

- Automática: Esta modalidad debe de ejecutar el algoritmo de forma animada sin que el usuario tenga que indicar cuándo se debe de ejecutar el siguiente paso, el tiempo entre cada uno de los pasos del algoritmo debe de ser configurable. La medida a utilizar es en segundos.
- Guiada: Esta modalidad permite al usuario indicar cuándo pasar al siguiente paso del algoritmo.

Para cada una de las modalidades debe de apoyarse de la herramienta de graphviz para realizar las animaciones de la ejecución del algoritmo. La imagen de la estructura debe de mostrarse en la aplicación, no fuera de ella y debe de actualizarse conforme se ejecuten los pasos del algoritmo en cualquiera de las dos modalidades. **De no utilizar graphviz o actualizar la imagen el reporte, la funcionalidad queda anulada con un puntaje de 0.**

## Learning Trees

Este módulo le permitirá al estudiante familiarizarse con los diferentes tipos de árboles y los algoritmos relacionados con ellos.

### AVL Tree

Al momento de que el usuario seleccione esta opción, la aplicación le permitirá al usuario crear una sesión de aprendizaje donde se podrá explorar los siguientes algoritmos en el árbol AVL, este árbol AVL debe de ser de números enteros.

#### Inserción

Para la inserción se deberá de indicar un archivo JSON que permitirá cargar una lista de números que se insertarán a la estructura, estos números deben de colocarse en espera a poder insertarse. La inserción se ejecutará al momento de iniciar la ejecución del algoritmo, dependiendo de la modalidad escogida para la ejecución se hará lo siguiente:

- Modalidad Automática: Esta modalidad irá insertando cada uno de los números de forma automática, mostrando cada paso de una inserción sin intervención del usuario (ejecutado de acuerdo al tiempo configurado).
- Modalidad Guiada: el usuario irá indicando el momento para continuar con la ejecución del algoritmo.


Los pasos que debe mostrar en cada inserción en el árbol AVL son:

- Búsqueda de la posición de inserción del nodo nuevo
- Cálculo de los factores de equilibrio y alturas por nodo
- Cada uno de los pasos detallados de las rotaciones, esto quiere decir que debe de mostrar los movimientos de punteros paso a paso.
- Unión de los punteros
- Árbol AVL después de la inserción.

Estos pasos deben de poder mostrarse tanto en modalidad guiada como automática.

#### Eliminación

Para la eliminación se deberá indicar un archivo JSON que permitirá cargar una lista de números que se deben de insertar para tener un árbol ya formado para proceder a realizar la eliminación de claves en el árbol. El árbol original al cual se le va a ejecutar el algoritmo de eliminación debe mostrarse antes de iniciar la ejecución (con alturas y factores de equilibrios calculados). Para



realizar la eliminación se le debe de indicar como parámetro la clave a eliminar, luego se ejecutará el algoritmo en cualquiera de las dos modalidades.

Los pasos que debe de mostrar en cada eliminación en el árbol AVL son:

- Búsqueda de la clave a eliminar
- Nuevo cálculo de los factores de equilibrio y alturas por nodo luego de la eliminación
- Cada uno de los pasos detallados de las rotaciones luego de la eliminación (si aplica), esto quiere decir que debe de mostrar los movimientos de punteros paso a paso.
- Unión de punteros luego de la eliminación.
- Árbol AVL después de la eliminación

## Recorridos

Esta sección debe de mostrar la ejecución de los recorridos de un AVL, recorridos: preorden, inorden y postorden. Antes de iniciar se debe de cargar un archivo JSON que permitirá cargar una lista de números que se deben de insertar para tener arbol ya formado para proceder a mostrar los recorridos. El árbol cargado debe de mostrarse justo antes de iniciar la ejecución de los recorridos. Se debe de escoger uno de los tres recorridos y luego se procede a ejecutar el algoritmo en cualquiera de las dos modalidades.

Los pasos que debe de mostrar en cada uno de los recorridos es:


- El nodo que actualmente se está visitando
- El orden por el que se está visitando
- Lista de nodos visitados ( salida del recorrido elegido, paso a paso)

## B Tree

Al momento de que el usuario seleccione esta opción, la aplicación le permitirá al usuario crear una sesión de aprendizaje donde se podrá explorar los siguientes algoritmos en el árbol B de orden 5, este árbol B debe de ser de números enteros.

## Inserción

Para la inserción se deberá de indicar un archivo JSON que permitirá cargar una lista de números que se insertarán a la estructura, estos números deben de colocarse en espera a poder insertarse. La inserción se ejecutará al momento de iniciar la ejecución del algoritmo, dependiendo de la modalidad escogida (automática o guiada).



Los pasos que se debe mostrar en la inserción son:

- Búsqueda de la posición de la clave a insertar
- División de la página
- creación de una nueva página
- Movimiento de las claves
- Árbol B después de la inserción

### Eliminación

Para la eliminación se deberá indicar un archivo JSON que permitirá cargar una lista de números para contar con un árbol ya formado para proceder a realizar la eliminación de claves en el árbol. El árbol original al cual se le va a ejecutar el algoritmo de eliminación, debe mostrarse antes de iniciar la ejecución. Para realizar la eliminación se le debe de indicar como parámetro la clave a eliminar, luego se ejecutará el algoritmo en cualquiera de las dos modalidades.

Los pasos que debe de mostrar en la eliminación son:

- Búsqueda de la clave a eliminar
- Cambios en las páginas del árbol
- Movimientos en las claves del árbol
- División de las páginas
- Árbol B después de la eliminación

## Learning Graphs

Este módulo permitirá al estudiante familiarizarse con el uso y representación de grafos, incluyendo algunos algoritmos para el recorrido de los grafos.

### Matriz de Adyacencia

Esta sección permitirá al estudiante poder ingresar mediante un archivo JSON un grafo no dirigido. Con el grafo no dirigido ingresado por medio del archivo debe de mostrar la matriz de adyacencia que lo representa. La forma de representar el grafo debe de explicarse de las dos modalidades (Automática y Guiada)

Los pasos que debe de mostrar al formar la matriz de adyacencia son:

- Creación de cada vértice en la matriz de adyacencia
- Adyacencia por cada nodo del grafo

### Recorrido por Anchura

Esta sección permitirá al estudiante poder entender cómo funciona el algoritmo de recorrido por anchura. Se deberá de ingresar un archivo JSON que representa un grafo no dirigido, al momento de cargarlo, la aplicación debe de mostrar el grafo sobre el cual se realizará el recorrido por anchura. Esta sección debe de mostrar el orden por el cuál se van visitando los nodos y el árbol generador por anchura. La aplicación debe ser capaz de mostrar todos los pasos que se ejecutan al realizar este algoritmo. La demostración del algoritmo puede realizarse ya sea de forma automática o guiada (elección del usuario).

Los pasos que deben de mostrarse del recorrido son:

- Lista de nodos visitados (de forma dinámica mediante el orden que se van visitando)
- Estructuras auxiliares para el recorrido
- Marcaje de nodos visitados
- Creación del árbol generador del grafo por anchura

## Recorrido por Profundidad

Esta sección permitirá al estudiante poder entender cómo funciona el algoritmo de recorrido por profundidad. Se deberá de ingresar un archivo JSON que representa un grafo no dirigido, al momento de cargarlo, la aplicación debe de mostrar el grafo sobre el cual se realizará el recorrido por profundidad. Esta sección debe de mostrar el orden por el cuál se van visitando los nodos y el árbol generador por profundidad. La aplicación debe ser capaz de mostrar todos los pasos que se ejecutan al realizar este algoritmo. La demostración del algoritmo puede realizarse ya sea de forma automática o guiada (elección del usuario).

Los pasos que deben de mostrarse del recorrido son:

- Lista de nodos visitados (de forma dinámica mediante el orden que se van visitando)
- Estructuras auxiliares para el recorrido
- Marcaje de nodos visitados
- Creación del árbol generador del grafo por profundidad

## Learning Sorting Algorithms

Esta sección de la aplicación permitirá al estudiante entender el funcionamiento de algunos algoritmos de ordenamiento. En esta sección se permitirá cargar un archivo JSON que representará un arreglo desordenado, para luego poder elegir cualquiera de los siguientes algoritmos de ordenamiento y ejecutarlos sobre el arreglo. Los algoritmos se pueden ejecutar en modalidad Automática o Guiada.

Algoritmos que se pueden seleccionar:

- Ordenamiento Burbuja
- Ordenamiento por Inserción
- Ordenamiento Quicksort

En cada uno de estos algoritmos (dependiendo de lo que el usuario elija) debe de poder representarse los siguientes pasos:

- Estado del arreglo en cada intercambio
- Movimiento de Datos
- Punteros auxiliares (si aplica)
- Estado del arreglo ordenado al final



**Todos los algoritmos deben de incluir la imagen de graphviz que representa el paso que se está ejecutando y una explicación de lo que está sucediendo en el paso ejecutado.** Ya que es una aplicación didáctica esta debe de ser intuitiva para el usuario y con un aspecto amigable para la facilidad de su uso. **La imagen de graphviz debe de incluir a las estructuras auxiliares en caso de que el algoritmo lo requiera. Todas las imágenes en graphviz deben de actualizarse en la aplicación, no se permitirá mostrar las imágenes en visor del sistema operativo ni buscar las imagenes en carpetas, en caso de que sea así el reporte queda anulado.**

## Archivos JSON

A continuación se explican los diferentes archivos de entrada que se podrán cargar en la aplicación para las diferentes estructuras.

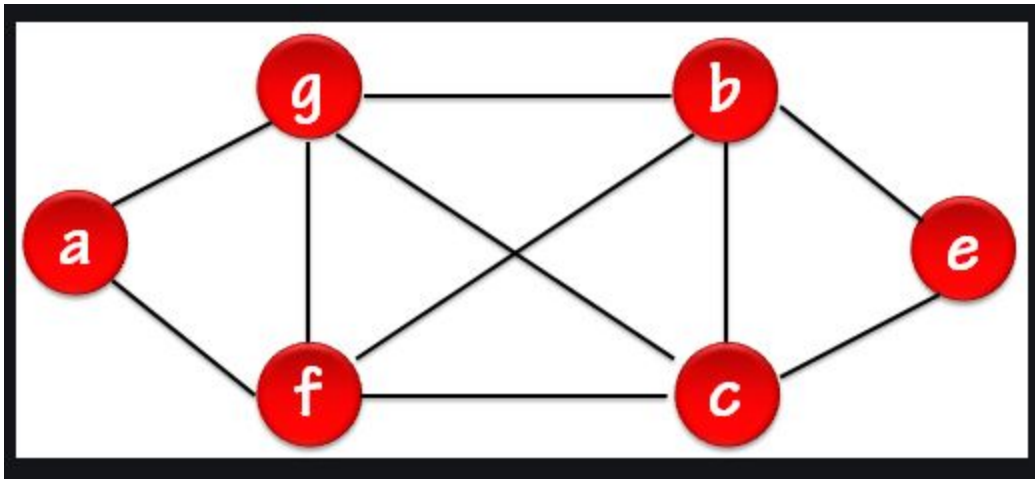
### AVL y BTree

La lista de números a ingresar al momento de mostrar los algoritmos de un ALV o Árbol B tendrán el siguiente formato, representa una lista de números enteros:

```
{
  "Input": [
    {
      "num": 45
    },
    {
      "num": 6
    },
    {
      "num": 8
    },
    {
      "num": 1
    },
    {
      "num": 100
    },
    {
      "num": 200
    },
    {
      "num": 300
    },
    {
      "num": 89
    }
  ]
}
```

## Grafos

El archivo JSON de entrada que se permitirá carga a la aplicación para la representación de los grafos tienen el siguiente formato, tomemos como ejemplo el siguiente grafo:




El archivo JSON que representaría el grafo tiene el siguiente formato:

```
{
  "Graph": [
    {
      "Node": "a",
      "Adjacency": [
        {
          "Node": "g"
        },
        {
          "Node": "f"
        }
      ]
    },
    {
      "Node": "g",
      "Adjacency": [
        {
          "Node": "b"
        },
        {
          "Node": "c"
        },
        {
          "Node": "f"
        },
        {
          "Node": "a"
        }
      ]
    }
  ]
}
```

## Ordenamientos

La lista de números a ingresar al momento de mostrar los algoritmos de ordenamiento será dada con el siguiente archivo JSON:

```
[{  
  "Array": [  
    {  
      "num": 20  
    },  
    {  
      "num": 100  
    },  
    {  
      "num": 50  
    },  
    {  
      "num": 30  
    },  
    {  
      "num": 100  
    },  
    {  
      "num": 200  
    },  
    {  
      "num": 300  
    },  
    {  
      "num": 80  
    }  
  ]  
}]
```



**IMPORTANTE:** para tener derecho a la calificación de las funcionalidades de la aplicación es indispensable contar con los reportes de las estructuras (Cada uno de los pasos de los algoritmos representados con graphviz), de no tener uno de los reportes se anula la funcionalidad asociada a la estructura que representa el reporte faltante. **TODA ESTRUCTURA UTILIZADA PARA EL DESARROLLO DE LOS ALGORITMOS EN LAS DIFERENTES SECCIONES DEBEN DE SER IMPLEMENTADAS POR EL ESTUDIANTE Y NO SE PERMITE EL USO DE LIBRERÍAS DE JAVA PREDEFINIDAS.**

## Restricciones

Las estructuras deben de ser desarrolladas por los estudiantes sin el uso de ninguna librería o estructura predefinida en el lenguaje a utilizar, incluyendo las estructuras auxiliares que se necesiten para ejecutar los algoritmos. **Los reportes son esenciales para verificar si se trabajaron correctamente las estructuras solicitadas, Si no se tiene el reporte de alguna estructura se anularán los puntos que tengan relación tanto al reporte como a la estructura en cuestión.**

## Penalizaciones

1. Falta de seguimiento de desarrollo continuo por medio Github tendrá una penalización del 10% (mínimo 1 o 2 commits por día).
2. Falta de seguimiento de instrucciones conforme al método de entrega (nombre del repositorio) tendrá una penalización del 5%.
3. Falta de puntualidad conforme a la entrega tendrá una penalización de la siguiente manera:
  - a. 0-6 horas – 20%.
  - b. 6-12 horas – 40%
  - c. 12-18 horas 60%
  - d. 18-24 horas – 80%.
  - e. Pasados 24 horas tendrá una nota de 0 y no se calificara.

**NOTA:** Las estructuras utilizadas deben ser implementadas por el estudiante y no se permite el uso librerías para realizar las estructuras, en caso de que fuese así el proyecto tendrá nota de 0.

## Observaciones

- Lenguaje a utilizar: Java
- IDE: Libre
- Sistema Operativo: Libre
- Librería para lectura de json: Libre
- Herramienta para desarrollo de reportes gráficos: Graphviz.
- Todas las imágenes de graphviz representan los pasos de cada uno de los algoritmos así que se deben de mostrar y actualizar dentro de la aplicación, de no ser así, el reporte queda anulado.
- La entrega se realizará por medio de: Github, cada estudiante deberá crear un repositorio con el nombre: EDD\_DIC\_2019\_PY2\_#carnet, y agregar a su auxiliar correspondiente como colaborador del mismo, para poder analizar su progreso y finalmente a partir del mismo repositorio realizar la calificación correspondiente. Usuario de Github: ricardcutzh
- Además de tener a su auxiliar como colaborador del repositorio para tener un control y orden de las personas que entreguen deberán de colocar el Link de su repositorio en la Tarea que cada auxiliar asignará en su classroom correspondiente.
- Fecha y hora de entrega: 4 de Enero, a las 7:00 A. M. horas (tentativo).
- Copias serán penalizadas con una nota de 0 y castigadas según lo indique el reglamento.
- En la entrega del classroom debe adjuntar su ejecutable (.jar) de donde se llevará a cabo la calificación (asegurarse que sea funcional)
- Aplicación con entorno gráfico, la herramienta para la interfaz GUI queda libre para escoger