

PONTIFICIA UNIVERSIDAD CATÓLICA DEL PERÚ
FACULTAD DE CIENCIAS E INGENIERÍA

ALGORITMIA Y ESTRUCTURA DE DATOS

Segundo Examen
(Primer Semestre 2025)

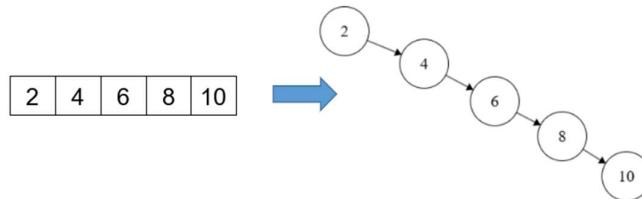
Duración: 2h 50 min.

- **No puede utilizar apuntes, solo hojas sueltas en blanco.**
- En cada función el alumno deberá incluir, a modo de comentario, la forma de solución que utiliza para resolver el problema. De no incluirse dicho comentario, el alumno perderá el derecho a reclamo en esa pregunta.
- No puede emplear plantillas o funciones no vistas en los cursos de programación de la especialidad.
- Los programas deben ser desarrollados en el lenguaje C++. Si la implementación es diferente a la estrategia indicada o no la incluye, la pregunta no será corregida.
- Un programa que no muestre resultados coherentes y/o útiles será corregido sobre el 50% del puntaje asignado a dicha pregunta.
- Debe utilizar comentarios para explicar la lógica seguida en el programa elaborado. El orden será parte de la evaluación.
- **Solo está permitido acceder a la plataforma de PAIDEIA, cualquier tipo de navegación, búsqueda y uso de herramientas de comunicación se considera plagio por tal motivo se anulará la evaluación y se procederá con las medidas disciplinarias dispuestas por la FCI.**
- Para esta evaluación solo se permite el uso de las librerías **iostream, iomanip, climits cstring, cmath o fstream**
- Su trabajo deberá ser subido a PAIDEIA.
- **Es obligatorio usar como compilador NetBeans.**
- Los archivos deben llevar como nombre su código de la siguiente forma **codigo_EX2_P#&** (donde # representa el número de la pregunta a resolver y & representa la alternativa a o b)

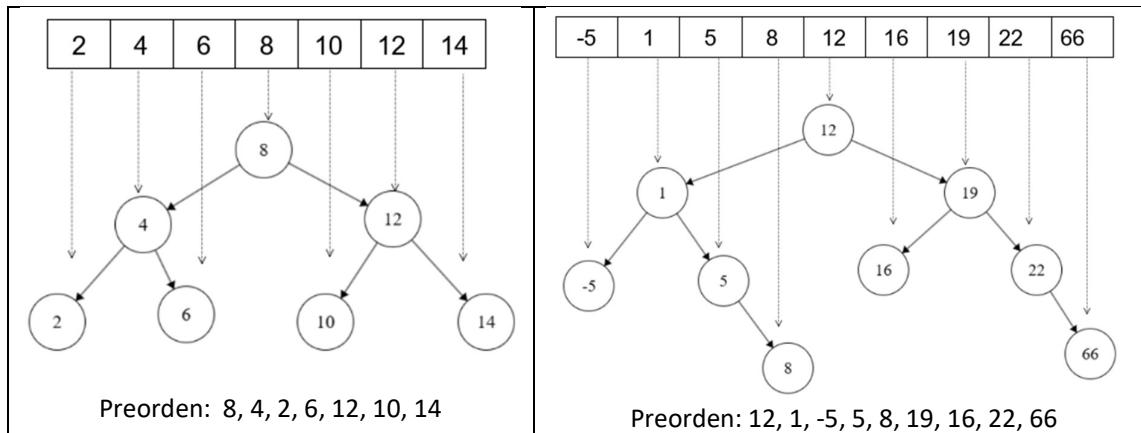
Elija 4 preguntas de las 5 que se muestran a continuación:

Pregunta 1 (5.0 puntos)

Dado un arreglo ordenado de números enteros, construir un Árbol Binario de Búsqueda (ABB) insertando los elementos en el orden en que aparecen genera un **árbol desbalanceado** (**ver imagen siguiente**). Este tipo de árbol presenta una altura $O(n)$, lo que degrada significativamente la eficiencia de las operaciones de búsqueda, inserción y eliminación.



Para evitar este problema, se requiere construir un ABB balanceado, es decir, un árbol en el que, para cada nodo, las alturas de sus subárboles izquierdo y derecho difieran como máximo en uno. Un ABB balanceado mantiene **una altura de orden $O(\log n)$** , lo que garantiza eficiencia en las operaciones básicas.



Requisitos de implementación:

- Implementar una función que construya el árbol binario de búsqueda balanceado a partir del arreglo ordenado.
 - Asegúrese de que la **complejidad total del algoritmo sea $O(n)$** .
 - Sugerencia: el árbol debe construirse dividiendo el arreglo en mitades de forma recursiva.
- Imprimir el árbol con recorrido preorden para verificar su correcta construcción.

Pregunta 2 (5.0 puntos)

Durante el monitoreo de eventos registrados por un sistema informático, se almacena un arreglo de 13 códigos de alerta ordenados de menor a mayor. Sin embargo, debido a una falla, el arreglo fue rotado: es decir, una parte del arreglo se movió al final, cambiando el orden original.

Por ejemplo, si originalmente era

[100, 145, 190, 235, 280, 300, 350, 400, 450, 500, 550, 600, 650]

tras la rotación puede quedar como

[300, 350, 400, 450, 500, 550, 600, 650, 100, 145, 190, 235, 280]

Se le pide implementar un programa en C++ utilizando la estrategia Divide y Vencerás, donde una función que reciba el arreglo rotado y un código de alerta a buscar, y retorne el índice donde se encuentra ese valor (o -1 si no está). La función debe tener complejidad $O(\log n)$.

```
X
Ingrese el codigo de alerta a buscar: 100
Codigo de alerta 100 encontrado en la posicion 9
```

```
X
Ingrese el codigo de alerta a buscar: 150
Codigo de alerta 150 no encontrado.
```

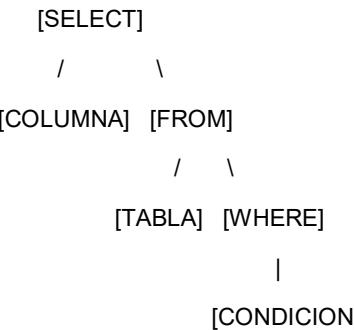
```
X
Ingrese el codigo de alerta a buscar: 300
Codigo de alerta 300 encontrado en la posicion 1
```

Pregunta 3 (5 puntos)

Implementar un Árbol Binario de Análisis Sintáctico (AST) que valide sentencias SQL básicas. Específicamente la sentencia **SELECT**.

- (4 puntos) **crearArbolSQL**. Construya el Árbol Binario (AST) para la sentencia dada (SELECT). El nodo debe manejar un elemento que soporte la estructura del árbol. Considere que van a existir datos fijos y otros variables. Se recomienda primero leer los datos. Luego plantarlos en el árbol. El árbol debe quedar listo para ser escalable, es decir admitir luego GROUP BY, ORDER BY, etc.
- (1 punto) **verificaArbolSQL**. Valide si cumple las reglas sintácticas para los casos de uso dados. Las validaciones consideran sólo cuatro casos, que no exista columna, tabla o condición, y que si existan las tres. (0.25 cada caso).

Árbol Esperado:



Ejemplo de impresión de lo que guarda el árbol con la consulta: **SELECT nombre FROM usuarios WHERE activo = 1**

```
SELECT
COLUMNA
nombre
FROM
TABLA
usuarios
WHERE
CONDICION
activo=1
```

CONSULTA VÁLIDA:

Caso de uso: `SELECT nombre FROM usuarios WHERE activo = 1`

Consulta válida: Estructura SELECT correcta.

ERRORES COMUNES:

Caso de uso: `SELECT FROM usuarios WHERE activo = 1`

Error: Falta la columna
Consulta inválida.

Caso de uso: `SELECT nombre WHERE activo = 1`

Error: Falta cláusula FROM.
Consulta inválida.

Caso de uso: `SELECT * FROM usuarios WHERE`

Error: WHERE debe tener una condición.
Consulta inválida.

Consideraciones:

- No está permitido el uso de variables globales.
- Se espera el uso correcto de las funciones de AB.
- Validar solo los cuatro casos de uso propuestos.
- Está permitido el uso de `string`.

Pregunta 4 (5.0 puntos)

Una empresa de manufactura tiene un registro diario de la **producción acumulada** de una línea de ensamblaje durante “n” días, representado por un arreglo $P[1...n]$, donde cada elemento indica la **cantidad total de unidades producidas** hasta ese día.

La gerencia desea identificar **los períodos anómalos**, definidos como pares de días (i,j) con $1 \leq i < j \leq n$, tales que:

- $P[i] > P[j]$, es decir, la **producción acumulada disminuyó** entre dos días consecutivos, lo cual no debería suceder en condiciones normales.

Por ejemplo:

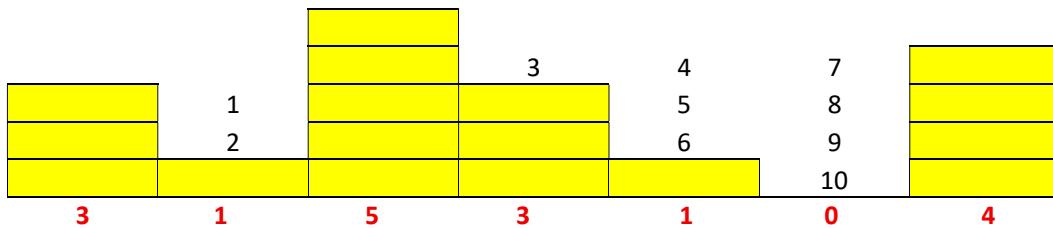
- Entrada:
 - Para $n = 6$ días.
 - Producción: $P = [10, 12, 14, 13, 11, 15]$
- Salida:
 - Existen 3 pares anómalos.

Solo para corroborar, los pares anómalos son $(3, 4), (3, 5), (4, 5)$, estos pares no se deben imprimir en la respuesta.

Implementa una función que cuente la cantidad total de estos pares anómalos. Para ello debe usar una estrategia de **Divide y Vencerás** para obtener una complejidad $O(n\log n)$.

Pregunta 5 (5 puntos)

Una empresa dedicada a la crianza de aves de corral desea evaluar los comederos más adecuados para alimentar a sus animales, buscando que puedan contener la mayor cantidad de granos. Los comedores están sido diseñados de tal forma que puedan contener el alimento deseado para las aves, por ejemplo, un diseñado dado por el arreglo = {3, 1, 5, 3, 1, 0, 4} tendrá el siguiente diagrama:



Por tal motivo podrá contener **10** unidades de semillas como máximo. Algunos detalles que tiene este comedero es que arriba del casillero 1, no se puede colocar más semillas ya que al no tener un respaldo al lado izquierdo, las semillas se caerían del recipiente. Por tal motivo para poder contener bien las semillas deben contar con un borde a la derecha y la izquierda a la misma altura.

Desarrolle un programa que, utilizando una pila o una cola auxiliar, **obtenga la cantidad de semillas** que puede contener el comedero que esta dado por un arreglo de entrada (no se debe modificar los valores del arreglo). **No puede emplear variables globales, no puede usar recursión, tampoco está permitido el uso de arreglos o tads adicionales a lo indicado. La respuesta debe tener una complejidad $O(n)$.**

Al finalizar el examen, comprima la carpeta de su proyecto empleando el programa Zip que viene por defecto en el Windows, **no se aceptarán los trabajos compactados con otros programas como RAR, WinRAR, 7zip o similares**. Luego súbalo a la tarea programa en Paideia para este examen.

Profesores del curso:

Ana Roncal
Fernando Huamán
David Allasi
Heider Sánchez
Rony Cueva

San Miguel, 12 de junio del 2025