PONTIFICIA UNIVERSIDAD CATÓLICA DEL PERÚ

FACULTAD DE CIENCIAS E INGENIERÍA

ALGORITMIA Y ESTRUCTURA DE DATOS

Segundo Examen
(Segundo Semestre 2024)

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 o 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 & la letra de la pregunta)

Pregunta 1 (10 puntos)

a. (5.0 puntos) Una empresa ha desarrollado un conjunto de lotes de productos. Cada lote tiene asociado un número y una cantidad. Ha colocado los lotes en un arreglo de forma ordenada por cantidad y número de lote, en ese orden.

Aquí un ejemplo:

N = 9 lotes

Índice	0	1	2	3	4	5	6	7	8
Lote:	15872	15865	15866	14357	14365	14368	14370	19258	19260
Cantidad:	3	4	4	6	6	6	6	8	8

Como se puede observar, el arreglo está ordenado por cantidad y en caso existan cantidades iguales, estás se encuentran ordenadas por el número de lote.

Se le pide elaborar un programa en C++ que permita realizar lo siguiente.

i. Implemente una función que encuentre el lote inicial para una cantidad determinada de productos. Esta función debe utilizar la técnica de divide y vencerás con una complejidad de ***O***(logn) (2.5 puntos).

ii. Implemente una función que encuentre el lote final para una cantidad determinada de productos. Esta función debe utilizar la técnica de divide y vencerás con una complejidad de ***O***(logn) (2.5 puntos).

Para verificar el uso correcto de las funciones implementadas en a y b debe desarrollar un programa principal que muestre ello.
Algunos ejemplos de ejecución serían los siguientes:
Para encontrar los lotes de 3 productos:
Lote Inicial: 15872
Lote Final: 15872
Para encontrar los lotes de 6 productos:
Lote Inicial: 14357
Lote Final: 14370
Para encontrar los lotes de 8 productos:
Lote Inicial: 19258
Lote Final: 19260

b. (5.0 puntos) Un restaurante utiliza un **sistema de reservas** que registra la cantidad de platos disponibles y las reservas realizadas para cada uno. Para gestionar mejor los recursos, es necesario identificar los platos con mayor porcentaje de reservas realizadas, **ordenándolos** de mayor a menor para priorizar su atención.

1. Implementar un algoritmo de **ordenamiento rápido** (ver líneas abajo) para ordenar los platos según el porcentaje de reservas realizadas.
2. El tiempo de ejecución del algoritmo debe ser ***O***(n log n) en promedio.
3. Mostrar los **3 platos con el mayor porcentaje de reservas**.
4. Se debe utilizar para la solución la estructura Plato dada a continuación.

```
struct Plato {
        string nombre;
        int cantidadDisponible;
        int cantidadReservada;
        double porcentajeReservas;
};
```

Datos de entrada (*nombre, cantidad disponible, cantidad reservada*). Estos pueden ser cargados al programa en forma fija o desde un archivo de texto.

- Lomo Saltado, 50, 30
- Ceviche, 40, 35
- Ají de Gallina, 30, 10
- Causa Limeña, 20, 20
- Arroz con Pollo, 60, 45

El método de ordenamiento rápido parte de datos desordenados y va a ir reduciendo el problema, dividiendo para vencer. El algoritmo escoge un elemento llamado pivote (el último elemento del arreglo actual).

				Pivote
60%	87.5%	33.33%	100%	75%

Y separa los elementos en dos grupos, los menores que el pivote y los mayores que el pivote.

Mayores		Menores		Pivote
87.5%	100%	60%	33.33%	75%

Formándose dos grupos aún desordenados de datos. Coloca el pivote en su posición correcta.

Mayores		Pivote	Menores	
87.5%	100%	75%	33.33%	60%

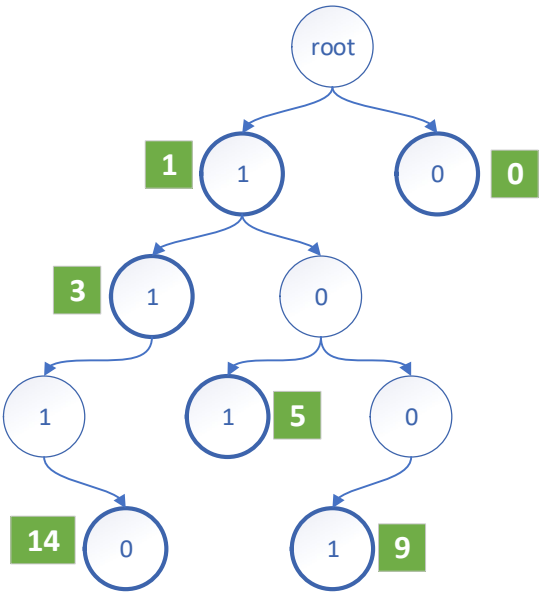
Volvemos a aplicar el mismo método al grupo de los mayores (desde el inicio hasta antes del pivote) y al grupo de los menores (después del pivote hasta el fin) en forma recursiva. El proceso se termina cuando no queden elementos por dividir.

Salida:

```
Top 3 platos con mayor porcentaje de reservas:
Causa Limeña - 100%
Ceviche - 87.5%
Arroz con Pollo - 75%
```

Pregunta 2 (10 puntos)

a. (5.0 puntos) Un árbol binario digital es una estructura eficiente para representar y Un árbol binario digital es una estructura eficiente para representar y comprimir números enteros, almacenando los prefijos comunes una sola vez. Por ejemplo, el siguiente árbol binario almacena los números {0, 1, 3, 5, 9, 14} de forma compacta al compartir caminos comunes en los dígitos binarios:



Se le pide**:**

i. Modificar el código del árbol binario proporcionado en clase para dar soporte estructural al problema. **No es necesario guardar el número en base decimal en el nodo**; los caminos en el árbol representarán los números en su formato binario. Construya manualmente el árbol en el main para representar un conjunto de números enteros insertando los nodos manualmente.

ii. (2 puntos) Implemente una función que reciba un número entero como parámetro y verifique si dicho número existe en el árbol. Pruebe la función con varios casos, como números presentes y ausentes en el árbol.

iii. (3 puntos) Implemente un método para recorrer el árbol en **preorden** e imprimir cada número almacenado junto a su representación binaria:

1 : 1

11 : 3

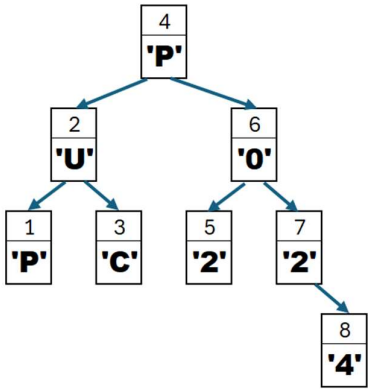
1110 : 14

101 : 5

1001 : 9

0 : 0

b. (5.0 puntos) Un sistema requiere verificar el acceso de un usuario basado en una contraseña predefinida la cual es almacenada en un ABB. El nodo del ABB contiene la posición del carácter en la contraseña y el carácter correspondiente. Como ejemplo, tome que la contraseña del sistema está compuesta por 8 caracteres: **PUCP2024**.



El usuario tiene un máximo de N intentos para ingresar la contraseña correctamente. En cada intento se valida que la longitud de la palabra ingresada coincida con la longitud de la contraseña. Luego se verifica cada carácter de la palabra ingresada, es decir que el carácter está en la contraseña y en la posición correcta. Si el carácter está en la contraseña, pero no en la posición correcta, o si el carácter no está en la contraseña, el intento se considera fallido.

Si el usuario ingresa la contraseña correcta dentro de los N intentos permitido, se muestra el mensaje "***Acceso concedido***". Si el usuario excede los intentos sin acertar la contraseña, se muestra el mensaje "***Se llegó al número de intentos fallidos permitidos***".

Desarrolle un programa en C++ que implemente el escenario descrito utilizando exclusivamente un ABB para gestionar y validar la contraseña. La contraseña del sistema debe ser creada como un ABB en el **main**().

Restricciones: La solución debe utilizar exclusivamente recursividad para todas las operaciones relacionadas con el ABB. No se permite el uso de arreglos, listas, ni otros TAD.

```
X
Ingrese el número máximo de intentos: 3
Intento 1/3. Ingrese la contraseña: PUKE24
Longitud incorrecta. Intento fallido.
Intento 2/3. Ingrese la contraseña: PUKE2024
Contraseña incorrecta. Intento fallido.
Intento 3/3. Ingrese la contraseña: PUCP2024
Acceso concedido.
```

```
                                                X
Ingrese el número máximo de intentos: 3
Intento 1/3. Ingrese la contraseña: PUKE23
Longitud incorrecta. Intento fallido.
Intento 2/3. Ingrese la contraseña: PUKE2024
```

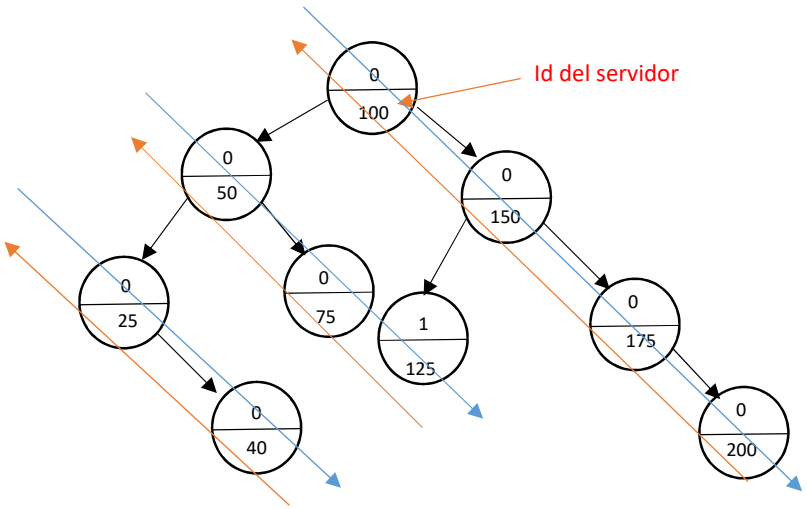
```
Contraseña incorrecta. Intento fallido.

Intento 3/3. Ingrese la contraseña: PUCP2025

Contraseña incorrecta. Intento fallido.

Se llegó al número de intentos fallidos permitidos.
```

Pregunta 3 (10 puntos)

Luego de los reiterados intentos fallidos del equipo de especialistas en algoritmia para detener el despliegue de SkyNerd, existe una última esperanza para poder eliminar a este sistema inteligente de la red. Es así como un reducido grupo de programadores que ha sobrevivido logra obtener la siguiente información:

- El sistema inteligente ha formado una red basada en un árbol binario de búsqueda, para su defensa
- Se sabe que cada nodo del árbol guarda un flag si es SkyNerd o no además de un número id del servidor el cual sirve para el ordenamiento del ABB
- El servidor inteligente es capaz de detectar los diferentes tipos de recorrido conocidos como, inorden, postorden, preorden, amplitud, amplitud inversa, etc.

Ante estas características el grupo de programadores decide realizar un nuevo recorrido poco usual, conocido como recorrido en diagonal. A continuación, un ejemplo:



La salida para este ejemplo será:

0-100 0-150 0-175 0-200 0-50 0-75 1-125 0-25 0-40

Con este recorrido los especialistas han detectado al servidor nocivo en el servidor con id 125

a. (5 puntos) Desarrolle una función **diagonal** que muestre el recorrido solicitado, en las líneas celestes.

A última hora llega información que SkyNerd, monitorea el servidor raíz de la red por tal motivo es necesario variar el recorrido de tal forma que se inicie desde el final de la siguiente forma:

0-40 0-25 1-125 0-75 0-50 0-200 0-175 0-150 0-100

b. (5 puntos) Desarrolle una función **diagonal_inv** que muestre el recorrido solicitado, en las líneas rojas.

Para el desarrollo de estas funciones la solución debe ser totalmente iterativa (por obvias razones cualquier función invocada también debe ser iterativa). Solo puede emplear como estructuras adicionales pilas y/o colas. **Recuerde que la pila o cola a emplearse solo puede recibir datos mas no nodos del árbol. Para la calificación de esta pregunta, los resultados deben ser correctos, no hay puntaje parcial para esta pregunta. Si desarrolla una de las funciones se considera el desarrollo de todo el bloque de 10 puntos, no es posible desarrollar solo una función y completarlo con parte de la pregunta 1 o 2, para completar los 10 puntos.**

El incumplimiento de las indicaciones invalida su respuesta.

Al finalizar el laboratorio, 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 laboratorio.

Profesores del curso:

Ana Roncal
Fernando Huamán
David Allasi
Rony Cueva
Heider Sanchez

San Miguel, 7 de diciembre del 2024