UNcomedor

Natalia Andrea Quiroga Castillo, Sofia Quimbay Cadena, Laura Camila López Pardo, Andres Fernando Rojas Pedroza, Julián Esteban Villate Pinzón

Grupo de trabajo 1

I. INTRODUCCIÓN

En el presente documento se busca presentar UNcomedor, el cual es un programa que da solución al problema de las aglomeraciones en los comedores de la Universidad Nacional de Colombia sede Bogotá; presentando un programa que utiliza los conceptos de estructuras de datos para ayudar a la comunidad, dentro del documento se encuentra mas explicado tanto el problema como el funcionamiento del programa, a quienes va dirigido, una descripción de la interfaz de usuario, entornos de desarrollo y operación, el prototipo de software inicial y el prototipo inicial de la interfaz.

A su vez se presenta el diseño, implementación y aplicación de las estructuras de datos en el programa, las pruebas del prototipo y análisis comparativo,la información de acceso al video demostrativo del prototipo de software, los roles y actividades, las dificultades y lecciones aprendidas, y se termina con las referencias que fueron usadas en todo el documento. En esta entrega del proyecto se presenta el uso de las estructuras de datos vistas en clase, como BTS, AVL y cola de prioridades, además también se presenta una comparación entre ellas.

II. DESCRIPCIÓN DEL PROBLEMA A RESOLVER

Durante el periodo actual del 2022-1 se viene presentando un aforo máximo en los comedores, este es generado por múltiples factores que se presentan desde tiempo atrás, como lo son la alta demanda en los comedores más centrales y la concurrencia en horas particulares por parte de la comunidad, sin embargo se vuelve un problema de solución urgente dadas las nuevas medidas de seguridad en la salud que iniciaron desde la pandemia del 2019, las cuales intentan evitar estos aforos dando soluciones como el distanciamiento, uso constante de tapabocas y lavado de manos.

Sin embargo los aforos no son consecuencia de la pandemia, sino que son eventos que se presentan desde tiempo atrás, y que perjudican el ambiente, bien sea por lo extenuante que resulta hacer la fila larga, o lo incómodo de las aglomeraciones e incluso si se presenta junto con la necesidad de comer rapido por el horario que existe académico.

Uncomedor busca disminuir estos aforos y facilitar a la comunidad una espera adecuada en las filas de los comedores. El propósito es disminuir el tiempo de espera, regular los aforos y brindar al usuario una experiencia controlada de la situación en tiempo real de los comedores de la Universidad Nacional de Colombia sede Bogotá.

III. USUARIOS DEL PRODUCTO DE SOFTWARE

En el programa se esperan dos roles inicialmente, el de administrador y el de usuario, el perfil de administrador va a tener acceso a todos los datos dentro del programa, puede ver las colas en cada uno de los comedores, puede ver cuantas personas hay en cada comedor, que turno tiene cada persona y puede modificar los datos que se le presentan al usuario, como lo es por ejemplo la disponibilidad de variedad de almuerzo o si está habilitado o no el comedor.

Mientras que en el otro perfil identificado como el usuario se espera poder ver la lista de comedores, la disponibilidad en cada uno de los comedores, ver que almuerzo ofrecen y la disponibilidad de los menús; a su vez la función principal de solicitar un turno en el comedor elegido, y poder realizar el pago (esta función estará disponible en actualizaciones futuras).

Esta clasificación se realiza teniendo en cuenta las funciones que son útiles para cada perfil, a los usuarios en este caso los y las estudiantes necesitan tener acceso a la información de disponibilidad de comedores mientras que los administradores necesitan saber cuánta demanda tendrá cada comedor y tener la posibilidad de cambiar la información que se le brinda al usuario.

IV. REQUERIMIENTOS FUNCIONALES DEL SOFTWARE

- Autenticación de usuarios

  Descripción: A través de la herramienta de google de Firebase y con el IDE de Android Studio se realiza la autenticación del usuario con correo y contraseña, para llevar un control de los usuarios de la aplicación.

- *Acciones iniciadoras y comportamiento esperado*:

  El usuario deberá diligenciar datos básicos, como nombre, identificación, correo electrónico, contraseña y tipo de apoyo socioeconómico a través de la interfaz de la aplicación para que posteriormente se haga la autenticación de los datos, en dado caso que ya se encuentre registrado en el sistema podrá iniciar sesión con su usuario y contraseña.

Requerimientos funcionales: Autenticación de datos.

- Almacenamientos de datos

  Descripción: Nuevamente con la herramienta de Firebase se almacenan los datos de la base de datos no relacionales que esta herramienta brinda. Una rama para todos los usuarios y otra para los administradores o cada uno de los comedores.

- *Acciones iniciadoras y comportamiento esperado*: Con los datos de la autenticación se creará un nodo con los datos para cada usuario.

Requerimientos funcionales: Almacenamiento de datos.

- Actualización de datos

  Descripción: Se tiene la opción de actualizar los datos de cada usuario en la base de datos de Firebase.

Requerimientos funcionales: Actualización de datos.

- Pila de layouts

  Descripción: Con la ayuda de la estructura de datos de pilas se harán las operaciones pop y push con los diferentes layouts de cada pestaña gráfica.

- *Acciones iniciadoras y comportamiento esperado*:

  Cada vez que el usuario oprima la tecla para devolverse una pestaña la pila realizará un pop() y cuando se oprima un botón para dirigirse a otra pestaña se hará un push() con el respectivo layout y actividad lógica.

Requerimientos funcionales: Creación de pila.

- Inserción, eliminación y organización de la cola de prioridades

  Se creó una cola cuya prioridad es el número de identificación del usuario ordenado de mayor a menor, con esta cola se puede remover el usuario con mayor jerarquía, ya que este tendrá prioridad a la hora de ser atendido en el comedor. Esta estructura tiene como datos objetos de tipo usuario.

- *Acciones iniciadoras y comportamiento esperado*:

  Se podrá, una vez creada la cola, organizar la estructura tal que los IDs de usuario se encuentren de mayor a menor y que se pueda extraer un usuario en la primera posición o en cualquier otra.

Requerimientos funcionales: Creación de cola de prioridades, actualización, búsqueda y eliminación de datos.

- Mapa de distancias

  Descripción: Mediante una ayuda visual que contiene las distancias mínimas desde un punto entre los ya establecidos por la aplicación, el usuario podrá encontrar los comedores que tienen una menor distancia entre su ubicación, es decir los más cercanos al usuario.

- Acciones iniciadoras y comportamiento esperado:

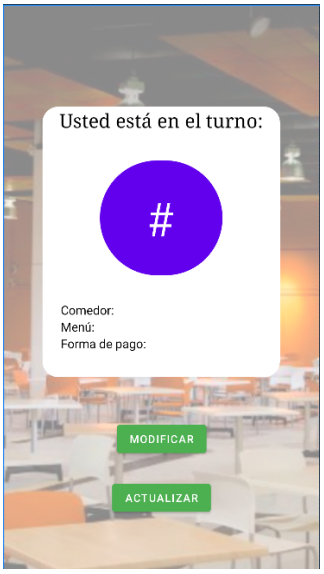
  Cada vez que el usuario ingresé a la aplicación y desee solicitar un cupo en un comedor con capacidad, se le retornará la distancia entre estos para que pueda identificar cual es el comedor más cerca con turnos disponibles, y así facilitar su desplazamiento por la institución.

Requerimientos funcionales: Creación de mapa visual de distancias.

V. AVANCE EN LA IMPLEMENTACIÓN DE LA INTERFAZ DE USUARIO

Al ingresar a la aplicación se encontrará un primer layout de inicio para ir a la interfaz para crear una cuenta de usuario o en el caso de ya tenerla poder iniciar sesión, una vez realizada la autenticación se ingresa a un menú donde se ven los datos del usuario y las opciones de cerrar sesión de usuario, actualizar los datos e ir a visualizar los comedores.

Posterior al menú principal se encuentra la sección de los diferentes comedores disponibles, seleccionando el comedor de preferencia aparece la sopciones para el mismo y la opción de hacer fila en este, al momento de seleccionar y aceptar hacer la fila se puede ver la confirmación del turno y la respectiva información del mismo, como el número y el menú.



VI. ENTORNOS DE DESARROLLO Y DE OPERACIÓN

En el entorno que se utilizó para desarrollar el proyecto fueron los IDE Visual Studio, Netbeans y Android Studio, para un desarrollo de aplicación con un SDK mínimo de Android 5.0 (lollipop), válido para el 98,8% de los dispositivos Android, a su vez se utilizó el lenguaje de programación Java y la herramienta de Google de Firebase. Para las reuniones de grupo se utilizó la plataforma de videollamadas meet.

VII. DESCRIPCIÓN DEL PROTOTIPO DE SOFTWARE

Repositorio: https://github.com/camilall/ProyectoED

En dicho repositorio se encuentra el README.md con la descripción del prototipo. En la carpeta de docs hallamos la documentación del proyecto, es decir el informe y las diapositivas con la explicación del prototipo. La carpeta Data contiene los datos de prueba utilizados y en lib las librerías utilizadas. En el branch de “entrega 2 AndroidStudio“ se encuentra el proyecto de Android Studio de UNcomedor se encuentra la carpeta src donde se encuentra almacenado el código fuente, con la parte lógica y gráfica de la aplicación

VIII. DISEÑO, IMPLEMENTACIÓN Y APLICACIÓN DE LAS ESTRUCTURAS DE DATOS

En este prototipo se implementaron las estructuras de datos tipo: Lista encadenada, pila, cola, cola de prioridad, árboles BTS y el AVL. Dado que para la primera entrega se especificaron la lista encadenada, pila y cola, en esta entrega se especificarán los faltantes que son: cola de prioridad, árboles BTS, el AVL y la pila implementada desde la perspectiva gráfica del proyecto.

Cola de prioridades

La cola de prioridades fue utilizada en el manejo de los usuarios, para esto inicialmente se le pide a la persona que ingrese los datos personales, que son nombre y apellido, número de identificación (id), que indique si cuenta con un apoyo socioeconómico y su usuario que en este caso es el correo electrónico de la universidad (@unal.edu.co), la cola de prioridades lo que hace es organizar todos los usuarios teniendo en cuenta su número de documento.

Este proceso se realiza a la hora de registrar el usuario, entonces para la implementación de la cola de prioridades se hizo creando la cola con objetos y no con números, pero a la hora de organizar los usuarios la comparación se realiza con el número de documento.
En cuanto a la cola de prioridades en sí, se tienen los métodos de sort(), en el cual se organizan los usuarios, se tiene removeMax(), LeftChild(), RightChild(), remove(), siftDown(), siftUp(), insert() y print (), son los demás métodos presentes para poder organizar la cola y poder agregar más usuarios a esta.

Árboles BST

La implementación del árbol BST es usada para desplegar los comedores y sus capacidades según la forma como se inserten. La implementación se hace a partir de una estructura de nodo que tiene cuatro variables internas: un nombre de comedor, una capacidad, y una referencia a un nodo izquierdo y derecho.
Los métodos de inserción, búsqueda y eliminación fueron implementados usando funciones recursivas para recorrer la estructura por cada uno de los nodos; cada una de estas acciones tiene una complejidad de tiempo de O(h), donde h es la altura del árbol.
El BST es ineficiente para fines del proyecto, pues no hay implementadas funciones de rebalanceo, por lo que la altura es difícilmente reducida aún después de numerosas eliminaciones y por consiguiente, el método de búsqueda (que se hace a partir de comparaciones entre las capacidades de cada nodo) tiene complejidades de tiempo bastante altas a las comparadas con estructuras dinámicas que se organicen a medida que se insertan o eliminan datos.

Arboles AVL

Para los árboles AVL la implementación se realiza mediante nodos con los atributos de nombre, ítem y peso para distinguir cada uno, a su vez cada uno maneja referencias al nodo izquierdo y al derecho, y un nodo root que va cambiando.

A diferencia del árbol BTS para el AVL se utilizan las funciones de height, rotación a la derecha y rotación a la izquierda, las cuales ayudan a la función getBalanceFactor, encargada de balancear el árbol para mantener las propiedades del AV, la propiedad de balanceo garantiza que la altura del árbol sea de O(log n).Luego en cada función se busca actualizar y balancear, esto a fin de mantener estas propiedades en el inserter, buscar y eliminar que corresponden a todas las estructuras de datos.

En el programa capacidadesComedoresAVL en el cual se ordenan los comedores por su capacidad se observan todos los atributos y funciones descritos anteriormente y los generadores de las distintas pruebas para esta estructura de datos, esto a fin de mantener el control de la capacidad y posteriormente comparar con la ocupación y obtener en orden los comedores más adecuados para solicitar los turnos. Estos se caracterizan por garantizar mayor facilidad y agilidad en las distintas operaciones de los árboles binarios.

Pilas

Para el desarrollo de la segunda entrega del proyecto UNcomedor se decidió usar la estructura de datos de Pilas genéricas usando referencias, con su respectivo constructor, métodos pop, push y empty para gestionar la navegación entre los distintos layouts de la parte gráfica del proyecto, con el método de push() al entrar a un layout y el de pop() en la opción de cerrar sesión e ir a un layout anterior presionando el boton de atras.

IX. PRUEBAS DEL PROTOTIPO DE SOFTWARE

Insertar (ms)					
	10.000	100.000	1.000.000	10.000.000	Notación Big O
Lista enlazada con cola	62	425	4.277	52.456	O(1)
Cola con referencia	86	658	5.332	61.549	O(1)
Pila con referencia	65	469	3.284	56.781	O(1)
Arreglos dinámicos	78	554	4.820	59.113	O(1)
Lista enlazada sin cola (Struct)	867	125.990	52.895.081	690.433.909	O(n)
BST	1510	44.970	3.243.920	77.380.125	O(n)
AVL	26	67	362	3.181	O(log(n))
Heap	113	585	5.034	60.082	O(log(n))

Tabla 1. Tabla comparativa de las pruebas realizadas insertar.

Buscar (ms)					
	10.000	100.000	1.000.000	10.000.000	Notación Big O
Lista enlazada	5	30	125	456	O(1)

con cola					
BST	21	240	3012	37370	O(n)
AVL	7,2896	21,639	129,5229	1660,7507	O(log(n))
Heap	4	19	903	12.932	O(log(n))

Tabla 2. Tabla comparativa de las pruebas realizadas buscar.

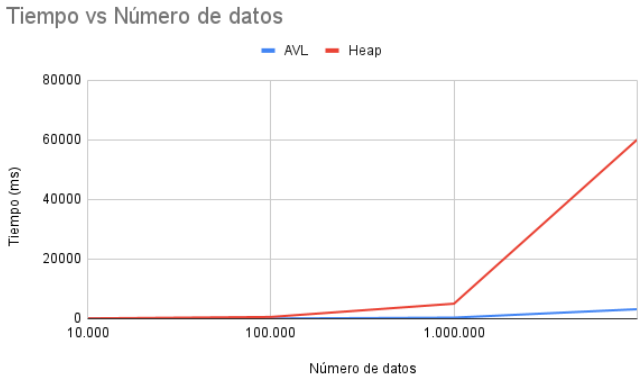
Eliminar (ms)					
	10.000	100.000	1.000.000	10.000.000	Notación Big O
Lista enlazada con cola	1	14	112	543	O(1)
BST	3	28	554	1921	O(h)
AVL	5.7447	32.6075	151.4323	1673.7189	O(log(n))
Heap	2	22	89	268	O(1)

Tabla 3. Tabla comparativa de las pruebas realizadas eliminar.

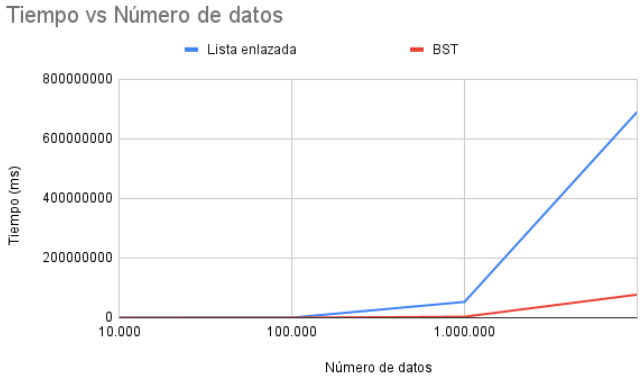
Gráficas

Insertar datos:

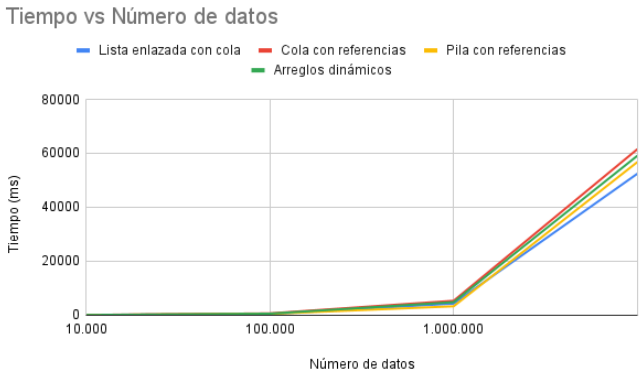
Primero se realizaron las gráficas de tiempo vs número de datos de las distintas estructuras para las operaciones de inserción, según la escala de valores correspondiente.



Gráfica 1. Tiempo vs Número de datos estructura AVL y heap

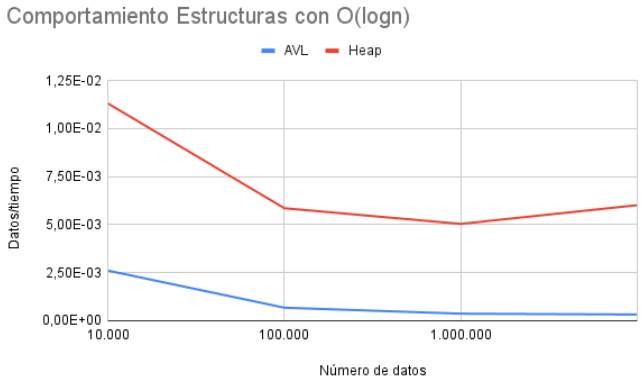


Gráfica 2. Tiempo vs Número de datos estructura Lista enlazada sin cola y BST.

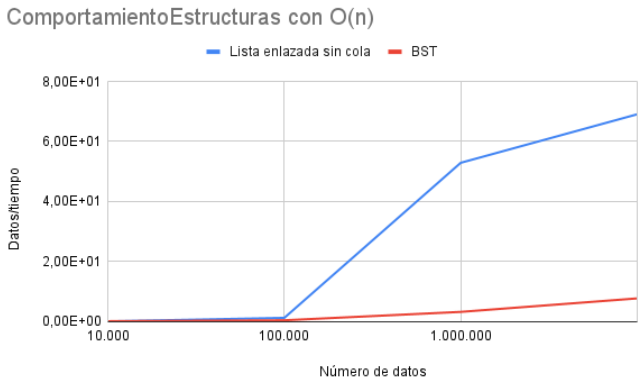


Gráfica 3. Tiempo vs Número de datos estructura Lista enlazada con cola, pila y cola con referencias y arreglos dinámicos.

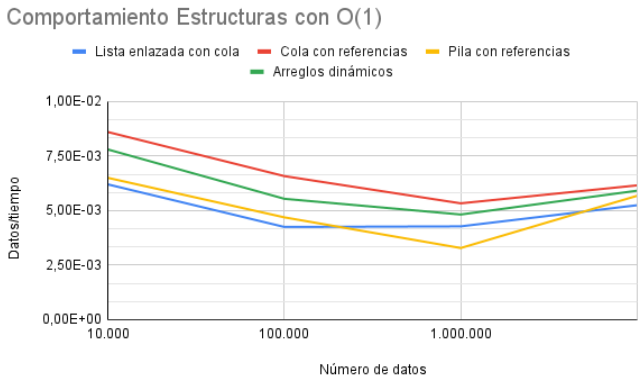
Posteriormente realizamos las gráficas en orden de complejidad de Big O, donde para visualizar correctamente la complejidad realizamos en cada medida la división entre el tiempo obtenido sobre la cantidad de datos analizado.



Gráfica 4. Comportamiento de las estructuras AVL y heap

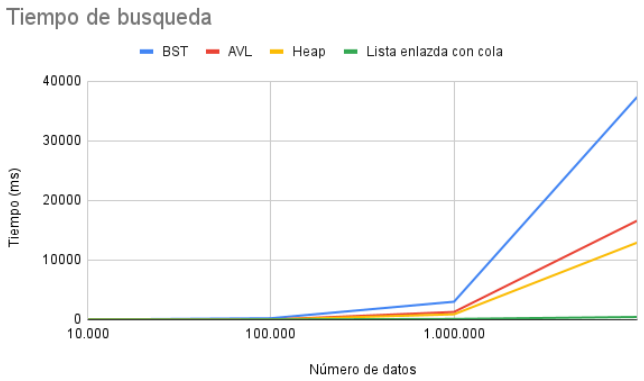


Gráfica 5. Comportamiento de las estructuras Lista enlazada sin cola y BST.



Gráfica 6. Comportamiento de las estructuras Lista enlazada con cola, pila y cola con referencias y arreglos dinámicos.

Búsqueda de datos:



Gráfica 7. Tiempo vs Número de datos estructura Lista enlazada con cola, AVL, BST, y heap en la operación de búsqueda de datos.

Eliminación de datos:

Gráfica 8. Tiempo vs Número de datos estructura Lista enlazada con cola, AVL, BST, y heap en la operación de eliminación de datos.

X. ACCESO AL REPOSITORIO DE SOFTWARE

Link del repositorio: https://github.com/camilall/ProyectoED

XI. ACCESO AL VIDEO DEMOSTRATIVO DE SOFTWARE

Video demostrativo: https://youtu.be/_OTORHU5rSg

XII. ROLES Y ACTIVIDADES

INTEGRANTE	ROL(ES)	ACTIVIDADES REALIZADAS (Listado)
Natalia Andrea Quiroga Castillo	Líder	-Organización de reuniones -Distribución de labores -Elaboración del documento, presentación y gráficas.
		-Programación interfaz gráfica, autenticación de usuarios, administración de bases de datos y programación de pila genérica para la interfaz.
Sofia Quimbay Cadena	Experto	- Programación del ingreso, manejo y organización de usuarios con cola de prioridades
	Secretaria	-Revisión y corrección del documento final, generación de pruebas de las diferentes estructuras de datos.
Laura Camila López Pardo	Coordinador	- Prestar atención a las fechas y cambios sugeridos para las próximas entregas.
	Observador	-Revisión del repositorio para confirmar posibles modificaciones
Andres Fernando Rojas Pedroza	Investigador	-Búsqueda de formas eficientes de implementación de árboles BST.
	Animador	-Mantener el buen ánimo en medio de las adversidades
Julián Esteban Villate Pinzón	Secretario	-Revisión del documento final,

		generación de pruebas de las diferentes estructuras de datos.
	Tecnico	-Programación interfaz gráfica, selección de comedores, implementación de la cola, administración de base de datos y funciones de administrador .

XIII. DIFICULTADES Y LECCIONES APRENDIDAS

Las principales dificultades en la elaboración del proyecto fueron la coordinación por parte de todo el equipo, el planeamiento del proyecto y el correcto manejo del repositorio, ya se nos dificulto el manejo de versiones y ramas de git y github.

Durante el proceso tuvimos valiosos aprendizajes, como el correcto planteamiento de los problemas a abordar desde un inicio, un correcto mejoramiento de los comandos de git y en el próximo ciclo de trabajo realizar reuniones con mayor frecuencia para ir viendo los avances de cada uno de los integrantes del grupo.

Dado que los integrantes del grupo presentan distintos horarios de estudio y trabajo, no fue tan fácil acomodar horarios para las actividades que requieren ser trabajadas grupalmente. Entre estas las reuniones de control de avance y dificultades en las distintas implementaciones.

Encontrar herramientas que ayuden a la hora de controlar los tiempos, dado que una de las dificultades que se presentó fue a la hora de cumplir con los plazos establecidos para cada parte de esta entrega, debido a la alta carga académica que cada uno tenía, tanto por la falta de claridad de estas mismas como por aplazar los tiempos para último momento.

Otra dificultad que se presentó fue que no todos los integrantes del grupo pudieron descargar la aplicación de Android Studio, ya que consume un gran volumen de recursos de almacenamiento del disco duro.

XIV. REFERENCIAS BIBLIOGRÁFICAS

[1] J. T. Streib y T. Soma, Guide to Data Structures. Cham: Springer International Publishing, 2017. Accedido el 13 de mayo de 2022. [En línea]. Disponible: https://doi.org/10.1007/978-3-319-70085-4

[2] X. F. Gutierrez, Estructuras de Datos. Alfaomega Grupo Editor, 2003.

[3] División de bienestar Sede Bogotá, Universidad Nacional de Colombia. http://www.bienestar.unal.edu.co/