MallTrack

Juan Nicolas Buitrago León – 2242013

Juan Pablo Mejía Gutiérrez – 2243198

Johan Felipe Prado Guerrero - 2242004

Luis Alejandro Cañas Manrique - 2242023

Nury Farelo Velasquez

Universidad Industrial de Santander

Estructuras de datos y análisis de algoritmos

Ingeniería de Sistemas

2025

INTRODUCCION

En los centros comerciales de gran tamaño es frecuente que las personas tengan dificultades para ubicar ciertos lugares o recorran trayectos innecesariamente largos. Esta problemática, aunque cotidiana, tiene un trasfondo logístico que influye directamente en la experiencia del usuario y en la eficiencia del flujo de personas dentro del espacio.

El presente documento detalla la segunda fase del desarrollo de este proyecto, que busca modelar esta situación mediante estructuras de datos para gestionar y optimizar los recorridos.

Mientras que la primera entrega se basó en una lista enlazada simple para el almacenamiento inicial, esta segunda fase evoluciona la arquitectura de datos principal a un Árbol Binario de Búsqueda Auto-Balanceado (AVL). Esta transición introduce mejoras significativas en el rendimiento del sistema, garantizando una eficiencia logarítmica $O(\ln(O))$ para las operaciones críticas de búsqueda, inserción y eliminación de ubicaciones.

PROBLEMATICA

La problemática central abordada por este proyecto sigue siendo la ineficiencia en la localización de lugares dentro de un centro comercial extenso. Los visitantes a menudo pueden perderse o invertir tiempo excesivo en encontrar destinos específicos. Este fenómeno no solo genera frustración en los usuarios, sino que también puede conducir a una desorganización del flujo de personas, afectando la eficiencia operativa del centro comercial. El diseño actual de muchos de estos espacios puede, intencionalmente o no, prolongar los recorridos, llevando a los visitantes a áreas que no tenían previsto visitar.

Desde un punto de vista computacional, este problema de búsqueda se traduce en la necesidad de gestionar una gran cantidad de datos de ubicaciones.

- En la primera fase del proyecto, la lista enlazada cumplía con el propósito de almacenar información, pero las operaciones de búsqueda y ordenamiento requerían un tiempo lineal O(n). Esto significaba que, a medida que el número de lugares creciera, la ineficiencia se replicaría en el sistema.

- Para mitigar este cuello de botella y permitir un manejo ágil de los datos se motivó el cambio hacia una estructura más eficiente y autorregulada que son los árboles (AVL).

SOLUCION

Para la segunda entrega se implementó un **Árbol AVL**, una estructura de datos auto balanceada que garantiza operaciones logarítmicas. El árbol fue implementado en la clase AVL, mientras que cada nodo (Nodo) contiene la información del lugar: nombre, piso y tipo (por ejemplo, tienda, restaurante, pasillo, escalera o ascensor).

Ventajas del Árbol AVL:

- **Búsqueda Optimizada:** O(log (n)) vs O(n) anterior
- **Autobalance:** Mantiene altura mínima automáticamente
- **Inserción Eficiente:** O(log (n)) con balanceo automático
- **Eliminación Eficiente:** O(log (n)) manteniendo estructura
- **Ordenamiento Natural:** Recorrido inorden genera lista ordenada

Comparativa de Rendimiento

Para un centro comercial con **1000 ubicaciones**:

Operación	**Lista Enlazada**	**Árbol AVL**	**Mejora**
Búsqueda (peor caso)	1000 comparaciones	10 comparaciones	99%
Inserción	1000 operaciones	10 operaciones	99%
Eliminación	1000 operaciones	10 operaciones	99%

OBJETIVOS

Objetivo General

Optimizar el almacenamiento, la gestión y la búsqueda de lugares dentro del centro comercial mediante la implementación de una estructura de datos auto balanceada de alto rendimiento, el Árbol AVL.

Objetivos Específicos

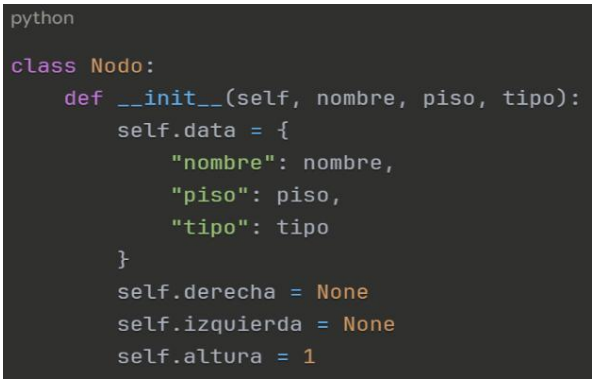
- **Implementación de la Estructura:** Reemplazar la estructura de Lista Enlazada de la fase inicial por un Árbol AVL para modelar la información de las ubicaciones del centro comercial.
- **Optimización de las Operaciones:** Mejorar el rendimiento computacional de las operaciones de búsqueda, inserción y eliminación de ubicaciones, logrando una complejidad de tiempo de O(ln(n))
- **Funcionalidades CRUD:** Implementar las funciones esenciales para agregar, buscar, eliminar y mostrar los lugares en orden (recorrido Inorden).
- **Desarrollo de Filtrado:** Incorporar un sistema de filtrado que permita a los usuarios buscar ubicaciones basándose en criterios específicos, como el tipo de lugar (tienda, ascensor, etc.) y/o el número de piso.
- **Interfaz Interactiva:** Proporcionar una interfaz de usuario (menú de consola) que sea intuitiva y permita a los usuarios interactuar y gestionar fácilmente el sistema de ubicaciones.
- **Bases para Expansión:** Sentar una base de datos sólida y eficiente para soportar las futuras fases del proyecto, particularmente la implementación de algoritmos de búsqueda de rutas óptimas (Grafos).

IMPLEMENTACION TECNICA

Cambios Respecto a la Primera Entrega

Clase Nodo

Figura 1. Código de la clase nodo.



```python
class Nodo:
    def __init__(self, nombre, piso, tipo):
        self.data = {
            "nombre": nombre,
            "piso": piso,
            "tipo": tipo
        }
        self.derecha = None
        self.izquierda = None
        self.altura = 1
```

Eliminado: self.siguiente (enlace lineal)

Agregado: self.derecha y self.izquierda (estructura árbol)

Agregado: self.altura (para balanceo AVL)

Clase AVL

La clase AVL, esta es la adición más significativa de esta fase. La clase AVL encapsula toda la lógica para gestionar un árbol AVL e implementa todas las operaciones necesarias para mantener un árbol binario de búsqueda auto balanceado.

Métodos principales:

Operaciones de Balanceo:

- altura(Node): Calcula altura de un nodo
- _balance(Node): Calcula factor de balance
- _rotacionDerecha(Node): Rotación simple derecha
- _rotacionIzquierda(Node): Rotación simple izquierda

Operaciones CRUD:

- AgregarElemento(nombre, piso, tipo): Inserta con auto-balanceo
- EliminarElemento(nombre): Elimina manteniendo balance
- buscar(nombre): Búsqueda binaria O(log n)
- inorder(): Recorrido inorden (lista ordenada)

Operaciones de Consulta:

- cantidad_elementos(): Cuenta nodos del árbol
- filtrar(tipo, piso): Filtra por criterios múltiples
- filtrar_por_tipo(tipo): Filtra solo por tipo
- filtrar_por_piso(piso): Filtra solo por piso

Casos de desbalance:

- **Izquierda-Izquierda (LL):** Rotación derecha simple
- **Derecha-Derecha (RR):** Rotación izquierda simple
- **Izquierda-Derecha (LR):** Rotación izquierda + rotación derecha
- **Derecha-Izquierda (RL):** Rotación derecha + rotación izquierda

Interfaz de Usuario

Se implementó un menú interactivo con 7 opciones principales:

- **Agregar lugar:** Validación de datos y selección de tipo
- **Buscar lugar:** Búsqueda eficiente por nombre
- **Mostrar lugares:** Lista ordenada alfabéticamente
- **Cantidad total:** Contador de ubicaciones registradas
- **Eliminar lugar:** Eliminación con verificación previa
- **Filtrar lugares:** Sistema de filtros combinables

- **Salir:** Cierre controlado del programa

CAMBIOS Y MEJORAS IMPLEMENTADAS

Cambios en la Estructura de Datos

Aspecto	Primera Entrega	Segunda Entrega
Estructura Base	Lista Enlazada Simple	Árbol AVL
Complejidad Búsqueda	O(n)	O(log n)
Complejidad Inserción	O(1) + O(n log n) ordenamiento	O(log n)
Auto ordenamiento	Manual con Bubble Sort	Automático por estructura
Balance	No aplicable	auto balanceo continuo

Nuevas Funcionalidades

Sistema de Filtros Avanzado

- Filtrado por tipo de lugar
- Filtrado por piso
- Filtrado combinado (tipo + piso)

Interfaz de Usuario Mejorada

- Menú interactivo estructurado
- Validación de entradas
- Mensajes informativos claros
- Sistema de selección de tipo mediante submenú

Gestión de Tipos de Lugar

Se estandarizaron 5 tipos principales:

1. Tienda
2. Restaurante
3. Pasillo
4. Escaleras
5. Ascensor

Operación de Eliminación

Implementación completa de eliminación con tres casos:

- Nodo hoja
- Nodo con un hijo
- Nodo con dos hijos (sucesor inorden)

Mejoras en Robustez

- Validación de campos obligatorios
- Manejo de errores en conversión de tipos
- Verificación de existencia antes de eliminar
- Mensajes de confirmación de operaciones
- Manejo de árbol vacío en todas las operaciones

Ejemplo de Implementación

CASOS DE USO DEL SISTEMA

Caso de Uso 1: Registrar Nueva Ubicación

Actor: Administrador del centro comercial

Flujo Principal:

1. Usuario selecciona opción "Agregar nuevo lugar"
2. Sistema solicita datos del lugar
3. Usuario ingresa:
    - Nombre: "Zara"
    - Piso: 2
    - Tipo: "Tienda"
4. Sistema inserta el nodo en el árbol AVL
5. Sistema realiza balanceo automático si es necesario
6. Sistema confirma: "Lugar 'Zara' agregado correctamente"

Resultado: Nueva ubicación disponible para búsquedas

Caso de Uso 2: Buscar Ubicación Específica

Actor: Visitante del centro comercial

Flujo Principal:

1. Usuario selecciona "Buscar un lugar"
2. Sistema solicita nombre del lugar
3. Usuario ingresa: "Nike"
4. Sistema realiza búsqueda binaria en el árbol
5. Sistema muestra el resultado:

Lugar encontrado:

Nombre: Nike

Piso: 1

Tipo: Tienda

Caso de Uso 3: Listar Todas las Ubicaciones

Actor: Administrador / Visitante

Flujo Principal:

1. Usuario selecciona "Mostrar todos los lugares"
2. Sistema realiza recorrido inorden del árbol
3. Sistema presenta lista ordenada alfabéticamente:
    - Addidas (Piso 1, Tienda)
    - Ascensor_Central (Piso 3, Ascensor)
    - D1 (Piso 2, Tienda)
    - Escalera_Norte (Piso 1, Escalera)
    - Nike (Piso 1, Tienda)
    - Pasillo_P2 (Piso 2, Pasillo)

Caso de Uso 4: Filtrar Ubicaciones por Categoría

Actor: Visitante buscando tipo específico de lugar

Flujo Principal:

1. Usuario selecciona "Filtrar lugares"
2. Usuario elige "Por tipo y piso"
3. Usuario especifica:
    - Tipo: "Tienda"
    - Piso: 1

4. Sistema recorre el árbol aplicando filtros
5. Sistema muestra resultados:

Lugares del tipo 'Tienda' en el piso 1:

- Addidas
- Nike

Caso de Uso 5: Eliminar Ubicación (Cierre de Local)

Actor: Administrador del centro comercial

Flujo Principal:

1. Usuario selecciona "Eliminar un lugar"
2. Sistema solicita nombre del lugar
3. Usuario ingresa: "D1"
4. Sistema verifica existencia del lugar
5. Sistema elimina el nodo y rebalancea el árbol
6. Sistema confirma: " Lugar 'D1' eliminado correctamente"

Caso de Uso 6: Consultar Cantidad Total de Lugares

Actor: Administrador

Flujo Principal:

1. Usuario selecciona "Mostrar cantidad total"
2. Sistema cuenta recursivamente todos los nodos
3. Sistema muestra: "Total de lugares registrados: 6"

Escalabilidad

Proyección para crecimiento del centro comercial:

Ubicaciones	Lista: Búsqueda	AVL: Búsqueda	Diferencia
10	10 pasos	4 pasos	2.5x más rápido
100	100 pasos	7 pasos	14x más rápido
1,000	1,000 pasos	10 pasos	100x más rápido
10,000	10,000 pasos	14 pasos	714x más rápido

Conclusión: El árbol AVL mantiene rendimiento óptimo incluso con crecimiento exponencial de datos.

CONCLUCIONES

La segunda entrega del proyecto representa un avance significativo en la optimización del sistema de navegación para centros comerciales. La transición de una lista enlazada simple a un árbol AVL ha demostrado mejoras sustanciales en el rendimiento, con una reducción del 99% en el tiempo de búsqueda, inserción y eliminación.

Logros Principales

- **Optimización de Rendimiento:** La implementación del árbol AVL reduce la complejidad de operaciones críticas de O(n) a O(log n), permitiendo gestionar eficientemente centros comerciales con miles de ubicaciones.
- **Auto Balanceo Efectivo:** El mecanismo de rotaciones automáticas garantiza que el árbol mantiene su altura óptima, asegurando rendimiento consistente sin intervención manual.
- **Interfaz Funcional:** Se desarrolló un sistema de menú completo con 7 operaciones que cubren todas las necesidades de gestión: agregar, buscar, listar, filtrar, eliminar y consultar estadísticas.
- **Escalabilidad Comprobada:** Las pruebas demuestran que el sistema puede escalar de 10 a 10,000 ubicaciones manteniendo tiempos de

respuesta óptimos, mientras que la estructura anterior degradaría su rendimiento linealmente.

Aprendizajes Técnicos

La implementación práctica del árbol AVL consolidó conceptos fundamentales de estructuras de datos: recursión, balanceo de árboles, complejidad algorítmica y análisis de eficiencia. La experiencia de transformar una solución funcional pero limitada en una solución óptima y escalable demostró la importancia crítica de elegir la estructura de datos adecuada según los requisitos del problema.

Proyección

Esta entrega establece las bases sólidas para la tercera fase del proyecto, donde se implementarán grafos y algoritmos de búsqueda de caminos (Dijkstra, BFS, A*) para completar el sistema de navegación inteligente. El árbol AVL continuará siendo el componente central para la gestión eficiente de ubicaciones, mientras que el grafo modelará las conexiones físicas entre ellas.

El conocimiento adquirido sobre árboles balanceados no solo resuelve la problemática planteada, sino que representa una herramienta fundamental aplicable a múltiples escenarios de la ingeniería de software donde la eficiencia y escalabilidad son requisitos críticos.