PEL2 ~ Curso 2024/25

Práctica de Árboles

PRIMER CUATRIMESTRE

Estructura de Datos - María José Domínguez 27/10/2024

Luis Miguel Herrá Alpuente - luis.herra@edu.uah.es - 06612001F

Jairo Moraga Monterroso - jairo.moraga@edu.uah.es - 49226007S

Índice

Parte 3 TADs creados:

A través del enunciado y la información compartida para la realización de la práctica, hemosdecidido la creación de los siguientes tipos abstractos de datos.

NodoABB:

Representa un nodo dentro de un Árbol Binario de Búsqueda (ABB), donde cada nodo contiene una lista de pasajeros y referencias a sus nodos hijos izquierdo y derecho.
Atributos

ListaPas (listapas): Una lista de pasajeros asociada a este nodo (ListaPas).

NodoABB hi:* Puntero al nodo hijo izquierdo (NodoABB*), que puede ser otro nodo con una lista de pasajeros o un nodo vacío.

NodoABB hd:* Puntero al nodo hijo derecho (NodoABB*), que también puede ser otro nodo con una lista de pasajeros o un nodo vacío

ABB:

Representa la común estructura de datos Árbol Binaria. Su implementación viene dada primordialmente por la compartida por el profesorado en el aula virtual Blackboard.

Atributos

Raíz: Nodo raíz del árbol, puede tener nodos hijos.

ListaPas y NodoListaPas: Funcionamiento análogo a la clase Lista, simplemente, el tipo de dato que contiene difiere. Aquí almacena pasajeros en lugar de boxes.

Pasajero:

Se ha modificado la clase Pasajero, ahora incluye dos nuevos atributos. Recordemos que la clase Pasajero representa a aquellos viajeros que acuden al aeropuerto.

Atributos

País: cadena de caracteres que muestra el país destino del Pasajero, previamente irrelevante.

Atención: entero que representa el tiempo hasta completar todos los procesos.

Parte 3 Operaciones en los TADs:

Descripción de los métodos empleados en los nuevos TADs implementados o modificados para el correcto funcionamiento de la práctica.

NodoABB.cpp

En este archivo se gestionan los nodos del Árbol Binario de Búsqueda (ABB) que contienen listas de pasajeros. Cada nodo tiene una lista asociada, y los nodos pueden tener hijos izquierdo y derecho. Los métodos implementados permiten inicializar un nodo, ver la información de un pasajero y gestionar su memoria.

Para comenzar, el constructor de la clase NodoABB inicializa los miembros de la clase: listapas, hi (hijo izquierdo) y hd (hijo derecho), utilizando los valores proporcionados como parámetros.

NodoABB(ListaPas pas): El constructor recibe una lista de pasajeros y crea un nodo con ella, inicializando los punteros a nullptr para los hijos izquierdo y derecho. Si la lista de pasajeros no está vacía, se muestra el país del primer pasajero.

NodoABB(ListaPas pas, NodoABB izq, NodoABB* der):* Este constructor recibe una lista de pasajeros y dos nodos hijos (izquierdo y derecho). Inicializa el nodo con estos valores, y si la lista no está vacía, muestra el país del primer pasajero.

~NodoABB(): El destructor muestra un mensaje indicando que el nodo ha sido destruido. Este método se ejecuta cuando el nodo es eliminado o liberado de memoria.

verPais(): Muestra el país del primer pasajero de la lista asociada al nodo. Si la lista está vacía, indica que no hay pasajeros.

¿Por qué dos constructores?

Es necesaria la presencia de dos constructores, uno de ellos para el caso inicial y posteriormente para los hijos relativos. En otras palabras, si estás construyendo el árbol desde cero, puedes usar el constructor con solo la lista de pasajeros, y luego ir agregando los nodos hijos. Si ya estás trabajando con nodos existentes (por ejemplo, durante la inserción en un árbol binario), puedes utilizar el constructor que permite pasar los nodos hijos directamente. Esta explicación es análoga en otras clases del proyecto.

Parte 3 Operaciones en los TADs II:

NodoABB.h

Para empezar, se incluye mediante #include "ListaPas.h" la definición de la clase ListaPas, que es necesaria para declarar un miembro listapas dentro de la clase NodoABB.

A continuación, se declaran los atributos privados de la clase: listapas (la lista de pasajeros), hi (hijo izquierdo) y hd (hijo derecho).

Cuenta con los siguientes métodos públicos:

- **Constructor de la clase** que inicializa los atributos listapas, hi y hd.
- **Método verPais()** que muestra el país del primer pasajero en la lista.

Se añade el método verPais() debido a la frecuencia de esta operación en el proyecto, permitiendo un rápido y sencillo visionado de la información.

Cambio de Paradigma respecto a PL1

En la primera práctica, la mayoría de funciones se implementaban en la clase utilidades, sin embargo, dada la corrección de la PL1 y la estructura del árbol binario, consideramos más óptimo y legible incluir los métodos en ABB.cpp.

ABB.cpp

En este archivo se gestionan las operaciones del Árbol Binario de Búsqueda (ABB), que organiza nodos de tipo NodoABB, cada uno con una lista de pasajeros. Se implementan métodos para insertar nodos, mostrar la información de los pasajeros y realizar búsquedas.

Para comenzar, el constructor de la clase ABB inicializa la raíz del árbol (raiz) de las siguientes maneras:

ABB(): Crea un árbol vacío con la raíz como nullptr.

***ABB(NodoABB* r):** Crea un árbol con un nodo raíz específico.

***ABB(ListaPas listapas, NodoABB hIz, NodoABB* hDer):** Crea un árbol con una lista de pasajeros y dos nodos hijos (izquierdo y derecho).

Destructor ~ABB(): Destruye el árbol y libera recursos.

Parte 3 Operaciones en los TADs III:

El árbol tiene varios métodos para mostrar la información de los pasajeros en diferentes órdenes:

verInOrden(): Muestra los pasajeros en orden ascendente.

verPostOrden(): Muestra los pasajeros en postorden.

verInOrdenCompleto(): Muestra todos los pasajeros del árbol.

mostrarPorPais(): Muestra los pasajeros de un país específico.

Métodos de búsqueda:

esta(): Verifica si hay pasajeros de un país en el árbol.

paisMenosVisitado(): Muestra el país con menos pasajeros.

paisMasVisitado(): Muestra el país con más pasajeros.

Métodos para insertar:

insertar(ListaPas pas): Inserta una lista de pasajeros en el árbol.

***insertar2(ListaPas pas, NodoABB* nodo)**: Método auxiliar para insertar recursivamente en el árbol.

Métodos adicionales:

mediaPorPais(): Calcula la media de tiempo de los pasajeros de un país.

muestraMedias(): Muestra la media de tiempo de todos los pasajeros en el árbol en preorden.

insertarPasajeroEnLista(): Inserta un pasajero en la lista del nodo correspondiente.

ABB.h

El archivo .h contiene la definición de los métodos que posteriormente serán implementados en ABB.cpp. Por tanto, contiene la definición de los métodos constructores, destructores, **verInOrden**(), **verPostOrden**(), **verInOrdenCompleto**(), **mostrarPorPais**(), **esta**(), **paisMenosVisitado**(), **paisMasVisitado**(), **insertar**(), **insertar2**(), **mediaPorPais**(), **muestraMedias**(), **insertarPasajeroEnLista**(). Muchas de estas funciones han sido creadas a merced del menú solicitado en el enunciado.

Parte 3 Operaciones en los TADs IV:

ListaPas.cpp

En este archivo, se gestiona una lista dinámica de pasajeros, permitiendo insertar, eliminar, acceder a elementos y calcular estadísticas sobre los pasajeros en la lista. La implementación utiliza nodos enlazados para mantener la estructura de la lista.

Constructores: ListaPas()**:** Constructor por defecto que inicializa una lista vacía. ListaPas(const ListaPas& otra)**:** Constructor de copia que clona otra lista ListaPas utilizando el método copiarLista.
Destructor: El destructor se encarga de liberar toda la memoria utilizada por la lista: Recorre los nodos, eliminándolos uno a uno. Muestra un mensaje indicando que la lista ha sido destruida.

Métodos principales:

copiarLista(const ListaPas& otra): Copia todos los elementos de otra lista ListaPas, asegurando que la nueva instancia sea una copia independiente.

insertar(Pasajero& p): Añade un pasajero al final de la lista: Si la lista está vacía, el pasajero se convierte en el primer nodo. De lo contrario, recorre hasta el final para añadir el nuevo nodo. Actualiza la longitud de la lista.

eliminarPorPrioridad(int prioridad_minima): Elimina todos los pasajeros cuya prioridad sea menor que el valor especificado. Reajusta los enlaces entre nodos para mantener la lista válida tras cada eliminación.

estaVacia(): Devuelve true si la lista está vacía, y false en caso contrario.

mostrarListaPas(): Imprime en consola los datos de todos los pasajeros en la lista. Verifica que la lista no esté vacía antes de recorrerla.

getLongitud(): Devuelve el número total de elementos en la lista.

getPrimero() const: Devuelve un puntero al primer nodo de la lista.

obtenerPasajeroEnPosicion(int indice): Retorna un puntero al pasajero en la posición especificada. Lanza una excepción si el índice está fuera de rango.

mediaLista(): Calcula y devuelve la media de los tiempos de atención de los pasajeros en la lista. Devuelve 0.0 si la lista está vacía.

ListaPas& operator=(const ListaPas& otra): Permite asignar el contenido de una lista ListaPas a otra. Libera los nodos actuales de la lista destino antes de copiar los de la lista fuente.

Parte 3 Operaciones en los TADs V:

ListaPas.h

Define la clase Lista con los atributos privados primero y longitud, y declara los métodos públicos que se implementan en Lista.cpp, incluyendo el constructor, destructor y los métodosdescritos anteriormente.

Pasajeros.cpp

En el constructor, se inicializa por defecto a cero el valor de atención, este se actualizará al finalizar los procesos con su respectivo setter. De esta forma, únicamente en el árbol se tiene el tiempo en el aeropuerto por pasajero. Es necesario definir los setters y getters de ambos atributos.
setPais(): establece el valor del atributo país del pasajero.
setAtencion(): establece el valor del atributo atención del pasajero.
getPais(): obtiene el país del pasajero.
getAtencion(): obtiene el tiempo de atención del pasajero.

Utilidades.cpp

Se ha añadido la función **insertarMejorado**(), permite insertar un pasajero en un Árbol Binario de Búsqueda (ABB) de manera eficiente, manejando la creación de nodos por país y añadiendo pasajeros a listas asociadas con dichos nodos. En caso de existir el país destino del pasajero, insertará dicho pasajero en la lista del nodo correspondiente, de manera contraria, en caso de no existir aún dicho país destino, se creará una lista vacía para la posterior inserción del pasajero en dicho nodo.

¿Por qué hacemos la función insertarMejorado()?

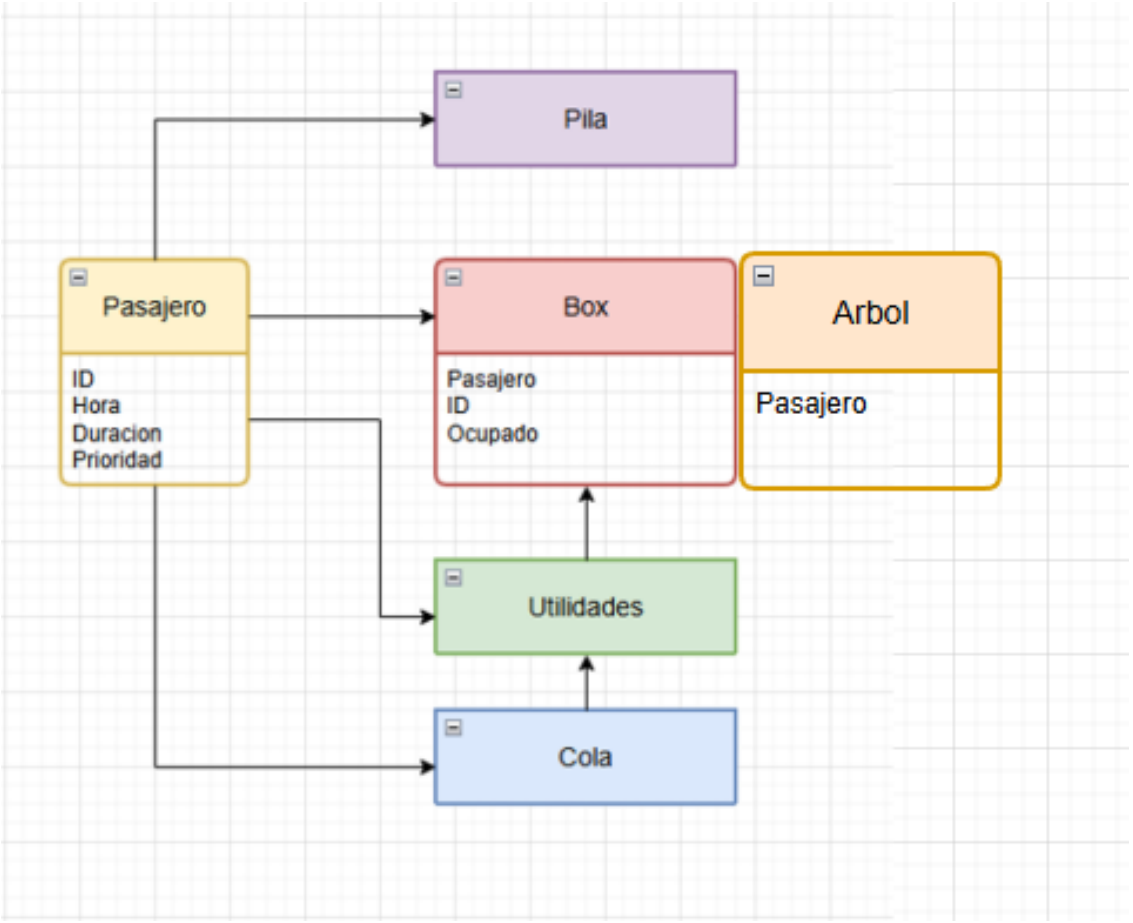
Simplificación de la función simular_min, cumpliendo lo solicitado en la corrección previa, optimización del código y mayor control ante posibles errores en inserción. Cumple lo descrito en el enunciado de la práctica a la perfección.

Universidad de Alcalá

Dificultades encontradas:

El desarrollo del trabajo ha sido marcado por un mayor tiempo de duración del proceso de análisis; esto ha favorecido un mejor entendimiento de los requisitos del sistema. Sin embargo, hemos experimentado problema en la implementación dada del árbol binario. Problemas solucionados gracias al contenido mostrado en la bibliografía.

Modelo Relacional de los TADs:



Este diseño busca centralizar las operaciones en la clase árbol y utilidades. De esta forma favorecemos menor acoplamiento, un código más legible y cómodo para el desarrollo la centralización de funciones en ambas clases. Este enfoque difiere relativamente de la PL1, donde se buscaba la centralización en la clase Utilidades.

Explicación del comportamiento del programa P3:

El comportamiento del programa sigue los criterios establecidos en la práctica con la implementación de un menú con las siguientes funcionalidades.

1. **Crear Pila de Pasajeros**: Se generan y apilan pasajeros con atributos como ID, prioridad, tiempo de atención y nivel.
2. **Mostrar Pila de Pasajeros**: Muestra todos los pasajeros apilados.
3. **Borrar Pila de Pasajeros**: Desapila todos los pasajeros hasta vaciar la pila.
4. **Mostrar Cola de Espera**: Permite seleccionar un box específico y muestra la cola deespera asociada.
5. **Mostrar Estado de Boxes**: Imprime el estado (ocupado/no ocupado) de todos losboxes disponibles.
6. **Simular N Minutos**: Simula el paso de un número especificado de minutos, ejecutandola función de simulación correspondiente.
7. **Simular Hasta que Todos Sean Atendidos**: Realiza una simulación completa hastaatender a todos los pasajeros.
8. **Mostrar Box Menos Ocupado**: Identifica y muestra el box con menos ocupación.
9. **Contar Boxes Operativos**: Cuenta y muestra cuántos boxes están operativos.
10. **Insertar un Pasajero en el Árbol Directamente**: Permite al usuario ingresar los datos de un pasajero (ID, prioridad, tiempo de atención, nivel, y país) y lo agrega al Árbol Binario de Búsqueda (ABB).
11. **Mostrar Países y Pasajeros del Árbol**: Muestra todos los países con sus respectivos pasajeros en orden ascendente, basándose en el orden inorden del ABB.
12. **Mostrar Pasajeros con un Destino Seleccionado**: Permite al usuario ingresar un país y muestra los pasajeros asociados a ese destino si existen en el ABB.
13. **Mostrar Todos los Países del Árbol**: Muestra únicamente los nombres de los países almacenados en el ABB, recorriéndolo en orden inorden.
14. **Mostrar País con Mayor y Menor Número de Pasajeros**: Identifica y muestra el país con la mayor cantidad de pasajeros y el país con la menor cantidad de pasajeros en el ABB.
15. **Calcular Media de Tiempo por País**: Dado un país ingresado por el usuario, calcula y muestra el tiempo promedio de atención de los pasajeros asociados a ese país.
16. **Mostrar Medias de Todos los Países (Preorden)**: Recorre el ABB en preorden y muestra la media de tiempo de atención por cada país.
17. **Mostrar Países en Postorden**: Recorre el ABB en postorden y muestra los nombres de los países almacenados.
18. **Salir**: Termina la ejecución del programa, cerrando el menú de opciones.

El bucle principal se repite hasta que el usuario elige la opción de salir, permitiendo unagestión dinámica de los pasajeros y los recursos del aeropuerto.

Evolución ~ Problemas acontecidos:

A la hora de insertar listas o elementos en el árbol no se comprobaba que el puntero del nodo del árbol donde se insertase no fuera nulo, ya que si no daba error de punteros. Para ello en todas las funciones donde están involucrados los nodos del árbol se comprueba antes de inicializarla que el nodo no sea igual a nullptr, en cuyo caso se terminaría la acción.

Las listas que se crean con los pasajeros cuyo país es el mismo se destruyen para liberar memoria innecesaria. Ahora bien, se insertaba en el árbol el puntero a la lista de un país determinado cuando la lista ya había sido destruida, lo que transmitía un error de punteros por pantalla. Para solucionarlo, se crea una lista copia la cual si se puede manejar para introducirla en el árbol, y al finalizar el programa destruirla.

Otro problema fue asegurar que las clases NodoListaPas y ListaPas manejasen la memoria correctamente revisando todas las funciones donde se trabajase con punteros.

Evolución ~ Mejoras propuestas:

Hemos creado una versión más concisa de la función simular_min, tal como se solicitó en la última corrección. Sin embargo, como desarrolladores, estamos más acostumbrados a la versión anterior, que nos resulta más legible. Por esta razón, la versión antigua se mantiene por defecto. Si deseas trabajar con la nueva versión, puedes copiar los archivos Utilidades.h y Utilidades.cpp en sus respectivas carpetas, sobrescribiendo los archivos existentes. Aquí un resumen de las nuevas funciones:

La función simular_min es esencial para gestionar un ciclo de la simulación en un solo minuto. Su tarea principal es mover los pasajeros de la pila de espera a la cola del box con menos carga, atender a los pasajeros que ya están siendo atendidos y eliminar los boxes vacíos si es necesario. Además, actualiza el tiempo de simulación y determina si debe continuar o finalizar, dependiendo de la disponibilidad de los boxes y los pasajeros.

La función esSimulacionFinalizada verifica si la simulación debe finalizar, asegurándose de que la pila de pasajeros esté vacía y que todos los boxes estén vacíos, lo que indica que no hay más pasajeros para atender ni boxes ocupados. Mientras tanto, la función moverPasajerosAPilas gestiona la llegada de nuevos pasajeros, asignándolos al box con menor carga cuando su hora de llegada coincide con el tiempo de simulación. Si es necesario, también puede crear nuevos boxes para equilibrar la carga.

Por último, la función procesarBoxes se encarga de gestionar a los pasajeros ya atendidos, moviéndolos al árbol de búsqueda y liberando espacio en los boxes. Si un box está vacío pero tiene una cola de espera, se le asigna un nuevo pasajero. La función procesarPasajeroEnBox reduce el tiempo de atención del pasajero y lo mueve al árbol binario de búsqueda cuando termina. Finalmente, asignarNuevoPasajeroABox asigna nuevos pasajeros a boxes vacíos para continuar con su atención.

Bibliografía:

Tutoriales útiles seguidos

https://www.youtube.com/watch?v=y3K3jb3wv2I

https://www.youtube.com/watch?v=_pcfFMFs9-g

https://www.youtube.com/watch?v=bgfH_HB341M

https://www.youtube.com/watch?v=slzcWKWCMBg

Páginas de aprendizaje online

https://www.w3schools.com/cpp/cpp_pointers.asp

Documentos PDF subidos

Especificación de Árbol Binario