Universidad de San Carlos de Guatemala
Facultad de Ingeniería
Escuela de Ciencias y Sistemas
Estructuras de Datos
Sección A
Catedrático: Rene Ornelis
Auxiliar: Daniel Monterroso

PROYECTO

SISTEMA DE GESTIÓN DE AEROPUERTO

Objetivos

General

Aplicar los conocimientos del curso de Estructuras de Datos en la creación de soluciones de software.

Específicos

- Hacer un uso correcto de la memoria dinámica y apuntadores en el lenguaje de programación C++.
- Aplicar los conocimientos adquiridos sobre estructuras de datos no lineales.
- Utilizar la herramienta graphviz para generar reportes.

Descripción general

En este proyecto se le solicita al estudiante del curso de Estructuras de Datos, implementar nuevas funcionalidades al sistema de gestión de aeropuerto que se desarrolló como parte de la práctica. Se mantendrán las mismas funciones, como la gestión de vuelos, pasajeros, equipajes y algunas otras características más. Este sistema utilizará diversas estructuras de datos adicionales a las que ya se tenían. Las estructuras a considerar son: árbol binario de búsqueda equilibrado, árbol B, matriz dispersa, tabla hash y grafos.

Implementación

Luego de haber realizado la simulación de la llegada de aviones, se analizará una entidad más que serán los pilotos del aeropuerto. La entidad piloto estará involucrada en varias estructuras y también tendrá interacción con los aviones. Se analizará también las rutas que pueden recorrer los aviones.

A continuación se describen las entidades a utilizar y las estructuras de datos que se deben implementar:

Aviones

Al inicio de la simulación se establecerá el listado de aviones a partir de un archivo de entrada el cual tendrá toda la información necesaria de cada avión. Estos se estarán almacenando en un **árbol B y una lista circular doble**; el árbol B de orden 5, utilizando como llave el número de registro de cada avión, corresponderá a los aviones con estado “Disponible”, y el listado será para los aviones con estado “Mantenimiento”. Posteriormente, a través del menú (ver Imagen 10), los aviones pueden ser movidos de ambas estructuras a través de su número de registro cambiando también su estado.


```
[
    {
        "vuelo": "A100",
        "numero_de_registro": "N12345",
        "modelo": "Boeing 737",
        "capacidad": 180,
        "aerolinea": "AirlineX",
        "ciudad_destino": "Madrid",
        "estado": "Disponible"
    },
    {
        "vuelo": "A102",
        "numero_de_registro": "N54321",
        "modelo": "Airbus A320",
        "capacidad": 150,
        "aerolinea": "AirlineY",
        "ciudad_destino": "Zaragoza",
        "estado": "Disponible"
    }
]
```


Imagen 1: Entrada en formato json para los aviones

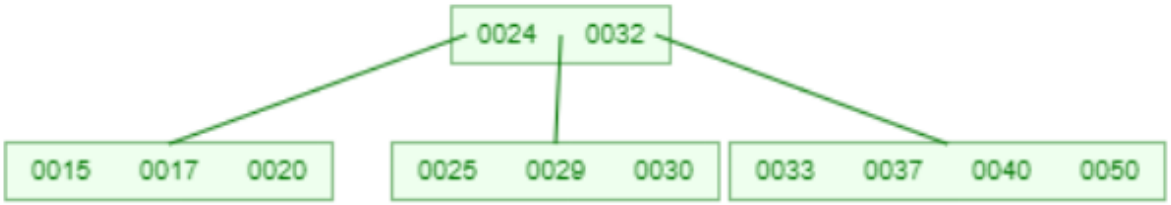

Imagen 2: árbol B

Pilotos

Los pilotos se registrarán a partir de un archivo de entrada el cual tendrá toda la información necesaria, este simulará el registro de dichos pilotos. Estos serán almacenados en un **árbol binario de búsqueda**, y serán ordenados por el número de horas de vuelo que tenga cada uno. Posteriormente, a través del menú (ver Imagen 9), se podrá recorrer y visualizar el árbol en sus diferentes formas: **preorden, inorden, postorden**. Esta última información deberá mostrarse en consola.


```
[
    {
        "nombre": "John Doe",
        "nacionalidad": "Estados Unidos",
        "numero_de_id": "P12345678",
        "vuelo": "A100",
        "horas_de_vuelo": 600,
        "tipo_de_licencia": "PPL"
    },
    {
        "nombre": "Jane Smith",
        "nacionalidad": "Reino Unido",
        "numero_de_id": "P98765432",
        "vuelo": "A200",
        "horas_de_vuelo": 99,
        "tipo_de_licencia": "CPL"
    }
]
```


Imagen 2: Entrada en formato json para los pilotos

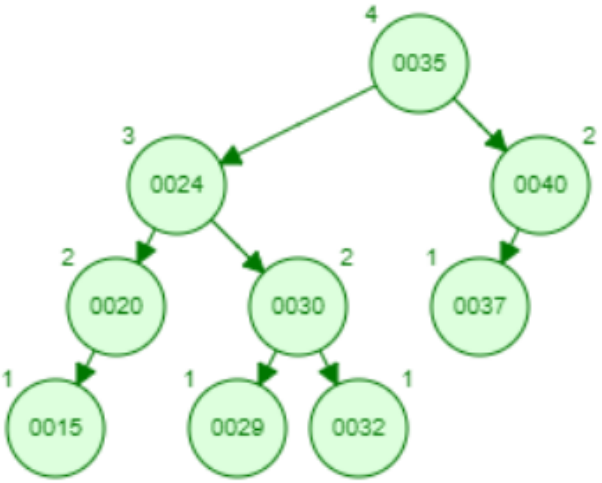

Imagen 3: árbol binario de búsqueda

En simultáneo, se estarán almacenando también a los pilotos en una **tabla hash**, utilizando como llave el número de ID, esto para tener un control de los mismos.

- El tamaño inicial de la tabla hash será 18 (M=18)
- La función de dispersión a utilizar será: h(llave) = llave mod M

Rutas

La aplicación deberá pedir la carga de un archivo el cual tendrá las rutas que pueden seguir los aviones, en el archivo se proporcionará:

- Origen
- Destino
- Distancia de ruta (km)

Por lo que deberá realizar un mapa general de todas las rutas, creando un **grafo dirigido** que será implementado a través de **listas de adyacencia**.

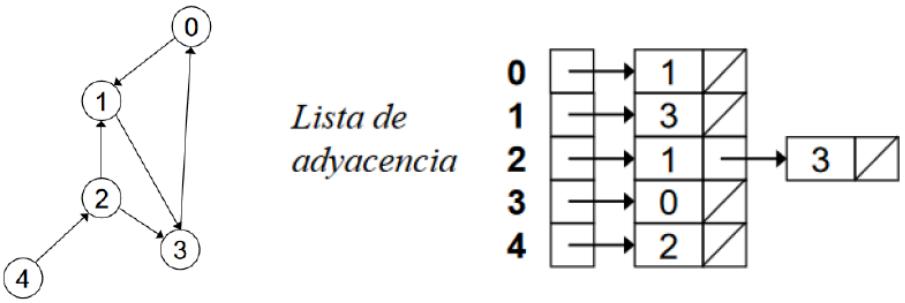

Imagen 4: Lista de adyacencia

Las rutas se cargarán por medio de un archivo de texto plano con el siguiente formato:

```
Cadiz/Sevilla/125;
Sevilla/Cadiz/125;
Sevilla/Granada/256;
Granada/Sevilla/256;
Sevilla/Jaen/242;
Jaen/Sevilla/242;
Granada/Jaen/99;
```

Imagen 5: Archivo de entrada para las rutas (Origen/Destino/Distancia;)

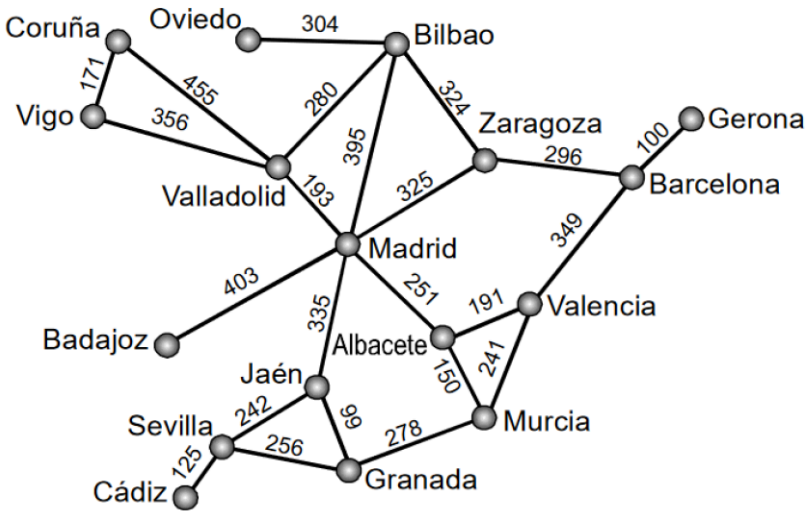

Imagen 6: Grafo dirigido de rutas

Recomendación de ruta

Una vez cargadas las rutas, la aplicación tendrá la opción de recomendar la ruta más corta entre 2 ciudades para futuros viajes, esto mediante un algoritmo implementado por el estudiante. En consola se solicitará la ciudad origen y la ciudad destino para realizar el cálculo de la ruta más corta (ver imagen 11), para luego mostrar dicha ruta en la misma consola.

Finalizada la carga de los aviones, rutas y pilotos, se necesita saber el piloto que estará a cargo de determinado vuelo y de la ciudad destino a la que debe llegar. Esto se hará a través de una **matriz dispersa** que conectará el vuelo y la ciudad destino. Conforme se vaya asignando un vuelo y una ciudad destino, así se deberá ir construyendo la matriz; si un **piloto se da de baja**, deberá quitar su información de la matriz, así como sus encabezados en caso que se queden sin ninguna conexión.

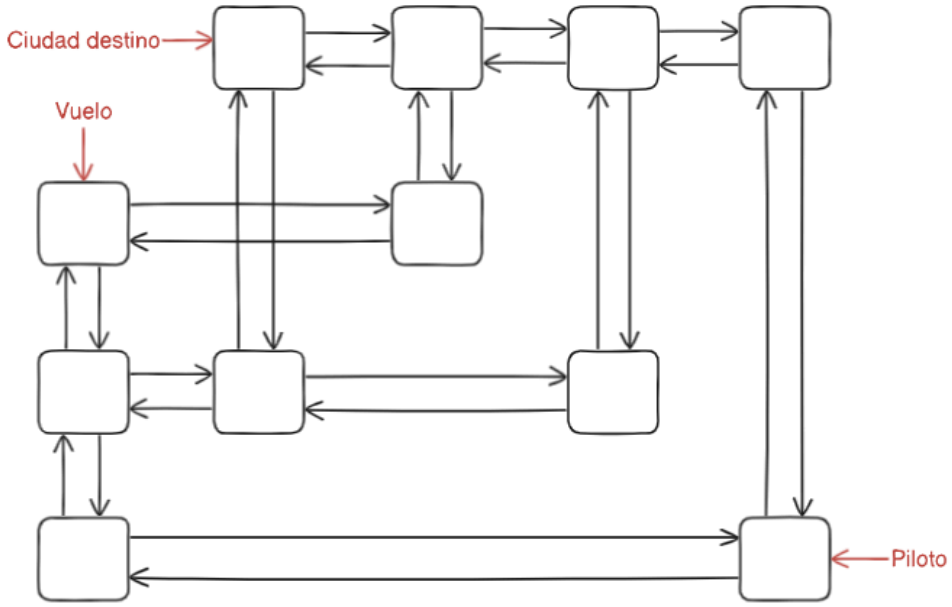

Imagen 7: Matriz dispersa

Nota: Los pilotos pueden darse de baja en el aeropuerto, por lo cual debe eliminarse la información en las estructuras involucradas a dicho piloto (árbol binario de búsqueda, matriz dispersa y tabla hash).

Flujo de la aplicación

La aplicación se ejecutará por medio de una consola, la cual pedirá la información necesaria para su correcto funcionamiento. La aplicación tendrá un menú donde el usuario podrá navegar y realizar el ingreso de los datos, lectura de archivos, generación de reportes y las diferentes operaciones en las estructuras de datos.

Ejemplo para la interfaz de la consola con las opciones mínimas que debe incluir:

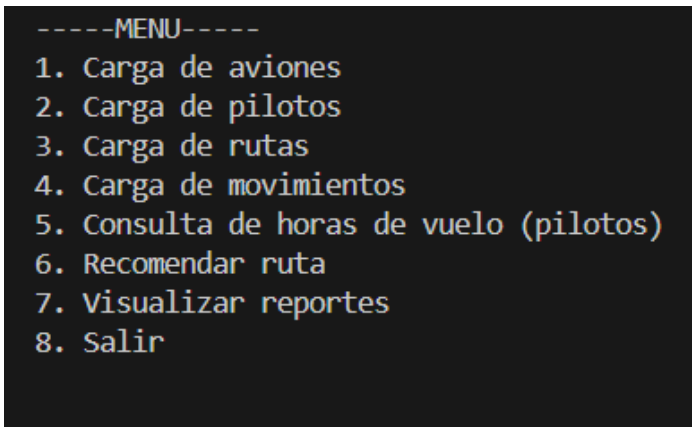

Imagen 8: Menú principal

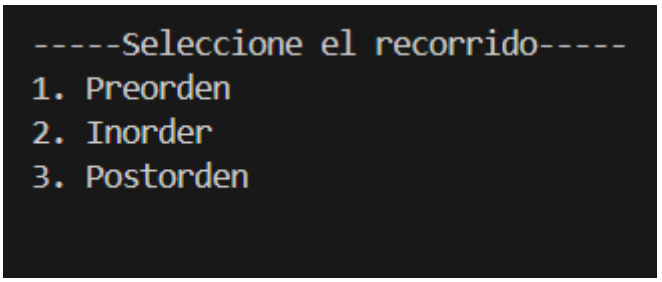

Imagen 9: Recorrido del árbol binario

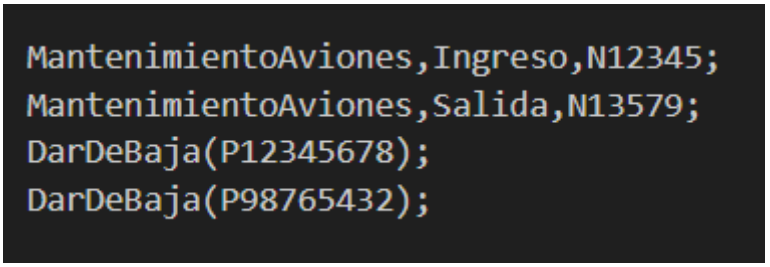

Imagen 10: Carga de movimientos

Nota: El comando *MantenimientoAviones*, seguido de su estado y número de registro, será para la lista circular doble de mantenimiento; el comando *DarDeBaja(#id)* servirá para la modificación de las estructuras involucradas con el piloto.

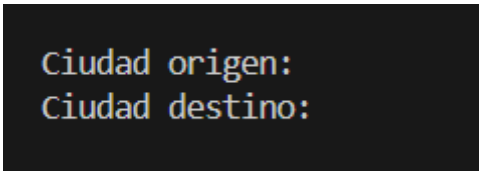

Imagen 11: Ingreso de datos para recomendar la ruta más corta

Reportes

Por medio del menú en la consola el usuario podrá generar y visualizar los siguientes reportes:

- Árbol B con aviones disponibles
- Lista de aviones en mantenimiento
- Árbol binario de búsqueda con las horas de vuelo de cada piloto
- Tabla hash de pilotos
- Grafo dirigido con las rutas
- Matriz dispersa de vuelos y ciudades

Manual técnico

Este manual tendrá los requerimientos del sistema operativo para llevar con éxito la ejecución del programa, así como también las dependencias instaladas y utilizadas para su desarrollo.

Restricciones

- El proyecto se realizará individualmente.
- El lenguaje de programación será C++.
- El IDE de desarrollo para la aplicación queda a discreción del estudiante.
- Los reportes deben ser implementados con la herramienta Graphviz.
- Los reportes deben abrirse desde la aplicación. No se permitirá buscar en directorios.
- Las estructuras de datos deben ser implementadas por el estudiante.
- **Las copias serán merecedoras de una nota de 0 puntos y los responsables serán reportados con el catedrático de la sección para su respectiva sanción**.

Entrega de la proyecto

- **La fecha de entrega será el domingo 30 de junio del 2024 antes de las 23:59 horas.**
- La entrega en UEDI será el link del repositorio que tendrá todos los archivos del proyecto.
- El archivo del manual técnico deberá estar en formato pdf y ser subido al repositorio.
- **El repositorio deberá tener como nombre el número de carnet de estudiante, ejemplo: 200000000_EDD_Proyecto.**
- **Mandar invitación de colaboración al siguiente usuario: EDD-Junio2024**