Universidad Nacional de Entre Ríos

Facultad de Ingeniería

Algoritmos y Estructuras de Datos

Informe General del Trabajo Práctico N°2

Aplicaciones de conceptos teóricos sobre estructuras jerárquicas, grafos y sus algoritmos asociados

Rodriguez Esteban

Trevisán Sergio

Romero Sheizza

2025

Consigna 1. Sala de Emergencias

Para la resolución de esta consigna se desarrollaron tres módulos: uno correspondiente a la estructura de datos, un montículo binario de mínimos, otro con la clase Paciente, y otro que implementa la lógica de la sala de emergencias (sistema de triaje).

La estructura seleccionada es un montículo de mínimos, adecuado para implementar una cola de prioridad. En el contexto de triaje, se requiere que los pacientes con mayor urgencia (menor nivel numérico de riesgo) sean atendidos primero. Esta estructura garantiza que el paciente de mayor prioridad se ubique siempre en la raíz del montículo, permitiendo su acceso en tiempo constante O(1), y sus operaciones de inserción y eliminación mantienen una eficiencia de O(log n), como se detalla en la **Tabla 1.**

Además, se define un criterio secundario de desempate en caso de pacientes con igual nivel de riesgo: se prioriza a quien haya llegado antes. Esto se logra implementando el método **__lt__** en la clase Paciente, considerando primero el nivel de riesgo y luego el orden de llegada. Esta personalización permite que el montículo se mantenga genérico, sin depender de una implementación específica para pacientes.

Cabe destacar que el montículo binario implementado es una estructura genérica, capaz de almacenar cualquier tipo de datos que implemente el operador "`es menor que`" (a través del método __lt__), permitiendo así su reutilización en diferentes contextos más allá del sistema de triaje.

Operación	Método	Orden de Complejidad
Inserción	insertar	O(log n)
Eliminación	eliminarMinimo	O(log n)

Tabla 1.

Facultad de
UNER Ingeniería

Consigna 2. Temperaturas_DB

Para la resolución de esta consigna se desarrollaron dos módulos:
uno correspondiente a la implementación del árbol AVL (AVL.py) y otro que representa la base de datos de temperaturas (Temperaturas_DB.py).

La clase **Temperaturas_DB** permite almacenar temperaturas asociadas a fechas, utilizando como estructura subyacente un árbol binario AVL, una estructura balanceada de búsqueda binaria que mantiene ordenados los datos cronológicamente (por fecha) y garantiza complejidad logarítmica en las operaciones básicas.

Se implementaron los siguientes métodos:

- **guardar_temperatura(fecha, temperatura)**: inserta la temperatura correspondiente a una fecha dada.
- **devolver_temperatura(fecha)**: devuelve la temperatura almacenada para la fecha ingresada.
- **max_temp_rango(fecha1, fecha2)**: recorre todas las fechas del rango y obtiene la mayor temperatura.
- **min_temp_rango(fecha1, fecha2)**: recorre todas las fechas del rango y obtiene la menor temperatura.
- **temp_extremos_rango(fecha1, fecha2)**: devuelve una tupla con las temperaturas mínima y máxima encontradas en el rango especificado.
- **borrar_temperatura(fecha)**: elimina una entrada del árbol AVL correspondiente a la fecha indicada.
- **devolver_temperaturas(fecha1, fecha2)**: devuelve una lista con las temperaturas registradas dentro del rango de fechas dado.

La estructura del árbol AVL, contenida en el archivo **AVL.py**, asegura que todas las operaciones de inserción, búsqueda y eliminación se realicen en tiempo O(log n), manteniendo balanceado el árbol después de cada modificación.

Método	Descripción	Complejidad Temporal
guardar_temperatura	Inserta una temperatura asociada a una fecha en el árbol AVL	O(log n)
devolver_temperatura	Devuelve la temperatura correspondiente a una fecha dada	O(log n)
max_temp_rango	Recorre el rango de fechas y devuelve la temperatura máxima	O(k·log n)
min_temp_rango	Recorrerá el rango de fechas para obtener la mínima	O(k·log n)
borrar_temperatura	Elimina la temperatura correspondiente a una fecha	O(log n)
cantidad_muestras	Devuelve la cantidad de muestras de la base de datos	O(1)
devolver_temperaturas	Devuelve un listado ordenado de temperaturas entre dos fechas	O(k·log n)
temp_extremos_rango	Devuelve la temperatura mínima y máxima entre dos fechas	O(k·log n)

Tabla 2.

*Donde **n** es la cantidad total de temperaturas almacenadas y **k** es la cantidad de días en el rango especificado.*

Para probar el funcionamiento de los métodos de **Temperaturas_DB**, se implementó un archivo **main,py**, donde se cargan distintas temperaturas en fechas espaciadas y se realiza una consulta utilizando **max_temp_rango**. La salida del programa permite verificar que el valor retornado sea correcto.

En el módulo AVL.py hay una prueba para ver las rotaciones que se producen cuando se agregan nodos al árbol -si bien no es un ensayo exhaustivo de todas las posibilidades, nos permitió verificar su funcionamiento.

Consigna 3. Palomas mensajeras

Se utilizó una estructura de grafo no dirigido, donde:

- Cada vértice representa una aldea.
- Cada arista representa una conexión directa entre dos aldeas y su distancia (en leguas).

Los componentes que se han utilizado para la implementación son los siguientes:
Clase Vértice: representa cada aldea y las conexiones con sus vecinos. Almacena la distancia mínima para formar el árbol de expansión y la aldea desde la cual se recibe la noticia.
Clase Grafo: administra la colección de aldeas (vértices) y sus conexiones (aristas), permitiendo agregar nuevas aldeas y rutas de comunicación.
Algoritmo Prim: utilizamos este algoritmo para construir un árbol de expansión mínima, que permite encontrar la red de conexión más eficiente entre todas partiendo desde “peligros”.
main: controla el flujo del programa -tomando los datos del archivo provisto para armar el grafo- e imprime los resultados.

Como salida del programa se obtuvo, entonces, la lista en orden alfabético de las aldeas y, luego, desde que aldea parte la noticia para ser entregada a su vecino, informando también la cantidad de distancia que recorren las palomas.

Obteniéndose así que la mínima distancia que recorrerán las palomas para que el mensaje llegue a todas las aldeas será de 94 leguas.

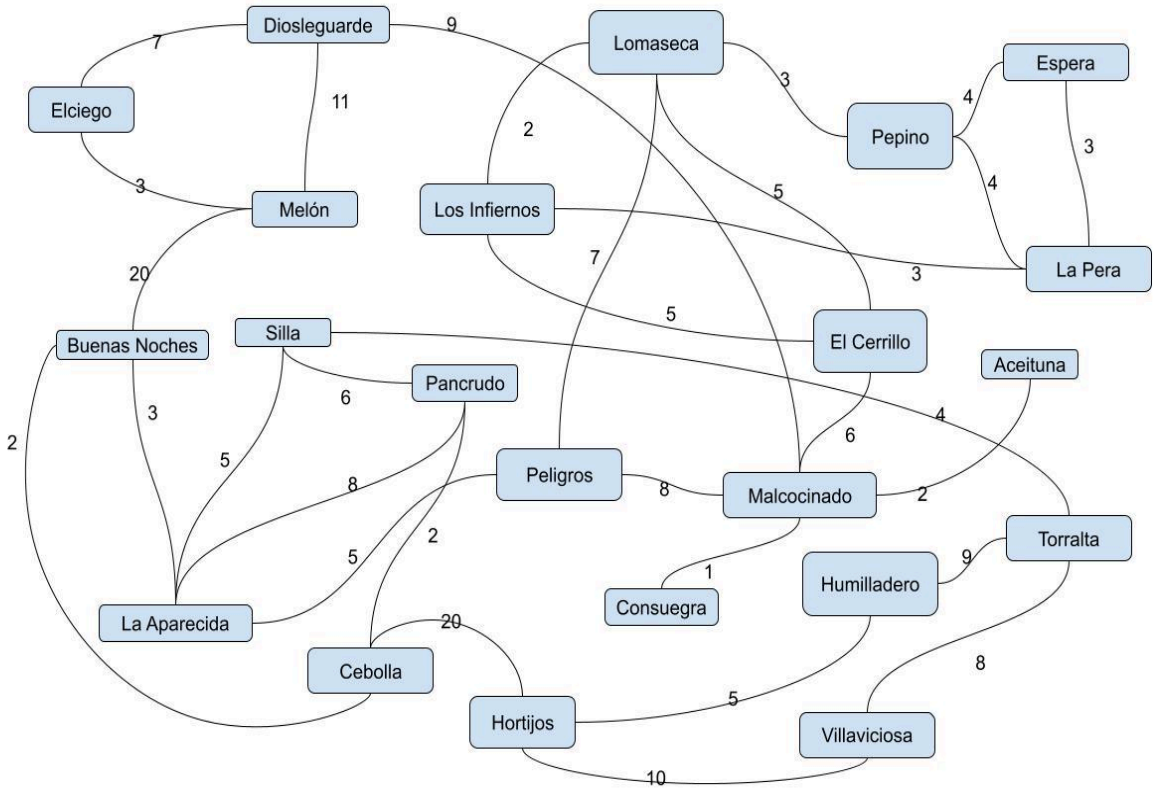

Gráfico 1: es un esquema que nos permitió hacer un chequeo visual del recorrido del mensaje para verificar la información que nos arroja el algoritmo de Prim (la longitud de los arcos no se corresponde con su ponderación).