
# UNIVERSIDAD NACIONAL DE ENTRE RÍOS

# FACULTAD DE INGENIERÍA

**CARRERA: Tecnicatura Universitaria en Procesamiento y Explotación de Datos**

**MATERIA: Algoritmos y Estructuras de Datos**

**ACTIVIDAD: Trabajo Práctico N°2**

**Profesores:** Javier Eduardo Diaz Zamboni

Jordán F. Infrán

Diana Vertiz del Valle

Belén Ferster

**Alumnos:** Ruiz Diaz, Enzo

Venturini, Angelo

**Fecha de Entrega: 02/11/2022**

# Trabajo Práctico N°2

## Objetivos

- Aplicar conceptos teóricos sobre estructuras jerárquicas, grafos y sus algoritmos asociados.

## Soluciones y Resultados

### Ejercicio 1

En el primer ejercicio se optó por utilizar una cola de prioridad, ya que esta utiliza internamente un montículo de mínimos cuya propiedad es que, al momento de eliminar, siempre devuelve el mínimo de todo el montículo, en este caso el paciente cuyo número de riesgo es menor (1-crítico, 2-moderado, 3-bajo) .El criterio de comparación primario para este ejercicio es el riesgo del paciente, donde si el riesgo es igual en dos pacientes se utiliza el orden de llegada como criterio de prioridad secundario.
La estructura implementada cuenta con todos los métodos necesarios para ser operada, de estos analizaremos dos:

| Análisis del orden de complejidad Big-O | |
|---|---|
| método insertar | Teniendo en cuenta el metod infiltrar arriba,tiene complejidad promedio O(log(N)), en su mejor caso O(1) y peor caso O(log(N)) |
| método avanzar | este metodo eliminar el minimo posee complejidad promedio de O(log(N)), su mejor caso O(1) y en su peor caso O(log(N)) |

# Ejercicio 2

Para este ejercicio se nos pedio implementar una base de datos de temperaturas que utilize un árbol AVL. Utilizamos como referencia la implementación disponible en la bibliografia, agregamos a la implementación los metodos rotar derecha y actualizar equilibrio rem junto con la modificación de los métodos recursivos remover y agregar. También implementamos una clase llamada temperaturas_db donde agregamos los metodos necesarios para su uso cuyo análisis encontramos en la tabla 1, dicha clase utiliza el árbol avl anteriormente mencionado como TAD. Por último, implementamos un test utilizando, el modulo unitest, el cual prueba todos los metodos de la clase temperaturas_db

**Tabla1**
*Análisis Big-O de los metedos de la clase temperaturas_db*

| Análisis del orden de complejidad Big-O | | |
|---|---|---|
| Método | Orden de complejidad (Promedio) | Explicación |
| guarda_temperatura(temperatura,fecha) | O(log(N)) | Al estar el árbol balanceado la búsqueda de la posición donde se quiere insertar es O(log(N)). Luego el reequilibrio, de ser necesario, es O(1) por lo que en el peor caso todo el método es O(log(N)) |
| devolver_temperatura(fecha) | O(log(N)) | Como consecuencia de que el árbol se encuentra balanceado, el peor caso en la búsqueda es de O(log(N)) |
| max_temp_rango(fecha1,fecha2) | O(N log(N)) | La busqueda del inicio del rango es O(log(N)), luego se busca el sucesor de cada nodo del rango , por lo que esta operación se repite D veces donde D es la cantidad de elementos en el rango, finalmente concluimos que es O(N log(N)) |
| min_temp_rango(fecha1,fecha2) | O(N log(N)) | La busqueda del inicio del rango es O(log(N)), luego se busca el sucesor de cada |

| | | |
|---|---|---|
| | | nodo del rango , por lo que esta operación se repite D veces donde D es la cantidad de elementos en el rango, finalmente concluimos que es O(N log(N)) |
| temp_extremos_rango(fecha1,fecha2) | O(N log(N)) | Similar a los métodos de minimo y máximo, cambiando que en este metodo se llama a ambos lo que nos da 2N log(N), concluimos O(N log(N) |
| borra_temperatura(fecha) | O(log(N)) | Como el árbol se encuentra balanceado la búsqueda del nodo a eliminar es O(log(N)). Luego el reequilibrio, de ser necesario, es O(1) por lo que en el peor caso todo el método es O(log(N)) |
| mostrar_temperaturas(fecha1,fecha2) | O(N log(N)) | Recorre todo el rango donde la busqueda del primer elemento del rango es log(N), por lo que esta operación se repite D veces donde D es la cantidad de elementos en el rango, finalmente concluimos que es O(N log(N)) |
| mostrar_cantidad_muestras() | O(1) | El tamaño se lleva en una variable por lo que no recorre el árbol nuevamente |

*Nota:* N es la cantidad de elementos en el árbol

# Ejercicio 3

En este ejercicio se nos plantea un tipo de problema conocido como “el problema del camino más amplio o del máximo cuello de botella” donde a partir de un archivo con rutas se nos pide hallar la ruta con el peso máximo y que a su vez tenga el precio mínimo. Es decir, primeramente debemos hallar el cuello de botella el cual será la máxima capacidad que se podrá transportar en esa ruta por lo que buscamos obtener el máximo valor entre los mínimos. A partir de esto,armamos un grafo con los datos proporcionados donde la

ponderación de las aristas será el peso. Luego implementamos un algoritmo de dijkstra modificado en el cual al vértice de inicio, su distancia se la inicializa en infinito y al resto de vértices en cero, obteniendo así el cuello de botella máximo. Una vez obtenido, procedemos a armar un nuevo grafo con todos los vértices pero solo se añaden las aristas que cumplan con la condición de que el peso de esa arista sea mayor o igual al cuello de botella pero se considera ahora como ponderación el costo. Ah este grafo aplicamos un algoritmo de dijkstra y obtenemos el costo mínimo.

Una vez obtenido  estos datos, generamos uno de los caminos óptimos, a partir del vértice destino, recorriendo por el antecesor del mismo hasta el origen.