**INFORME**

**1. Complejidad de inserciones y eliminaciones en un montículo binario**

En un montículo binario, el orden de complejidad O en:

* *Inserciones:* dado que la función Insertar agrega el elemento al final de la lista y que luego es desplazado hacia arriba con la función InfiltrarArriba, el orden de complejidad, en el peor caso, es , que representa la altura en el montículo (siendo n la cantidad de elementos).
* *Eliminaciones:* realizando el mismo análisis que con las inserciones, la función eliminarMin también tiene una complejidad . Depende de la cantidad de elementos y la cantidad de comparaciones que realiza entre esos elementos la función InfiltrarAbajo, la cual es utilizada por eliminarMin.

**2. Elección del montículo binario y la cola de prioridad**

La estructura seleccionada para este problema fue un montículo binario, debido a su eficiencia en cuanto al orden de los datos. Acceder al primer dato, es decir al paciente con mayor prioridad, tiene un orden de complejidad . Una vez el primer dato es eliminado del montículo (cuando el paciente con mayor prioridad es atendido), los datos se reorganizan (en ). Esta reorganización del montículo asegura la eficiencia de la estructura, dado que los datos siempre mantienen el orden entre ellos, garantizando que el primer criterio se cumpla.

A su vez, se implementó una cola de prioridad, la cual usa al montículo binario. La elección de la cola de prioridad radica en su política FIFO, la cual nos asegura que el primer dato en ingresar será el primero en salir. Esta política de la cola de prioridad acompaña al segundo criterio implementado para la cola de espera en la sala de emergencias.

En el contexto del problema, el montículo binario asegura el orden de los pacientes según el nivel de riesgo de su salud, mientras que la cola de prioridad asegura el orden de los pacientes según su orden de llegada a la sala de emergencias. De esta manera, se garantiza que el paciente más delicado sea el primero en atenderse y, en el caso de que el nivel de riesgo sea el mismo, será el primero que haya llegado a la sala de emergencias.

**3. Breve explicación de la solución**

Para resolver este problema, se tuvo en cuenta que la consigna solicita aplicar una estructura de datos adecuada para almacenar los pacientes conforme ingresan al centro de salud, utilizando dos criterios: el primero y más importante radica en un proceso que permite una gestión de riesgo clínico para manejar adecuadamente los flujos de pacientes, categorizándolos en tres niveles de riesgo, siendo el primero el más crítico; el segundo criterio que implementamos fue el orden de llegada de cada paciente, el cual es un número asignado en la siguiente línea:

self.\_\_llegada = Paciente.contador\_llegada

Se guarda la información de cada paciente que ingresa a la sala de emergencias: nombre, apellido, nivel de riesgo y orden de llegada.

En las Salas de Emergencias (una en la carpeta app, la cual fue brindada por la cátedra y se le hicieron ciertas modificaciones; la otra en la carpeta test, la cual fue hecha para corroborar rápidamente si los pacientes están siendo atendidos en el orden adecuado, según los criterios adaptados), se determina que las variables cola\_de\_espera y cola son las colas de prioridad. También, en cada sala, se denomina una variable “paciente” y se inserta a la cola de prioridad para ser acomodada entre los demás datos, tomando en cuenta su nivel de riesgo y si coincide con algún otro dato, se acomoda en base al orden de llegada. De esta manera, obtenemos una cola de espera de pacientes, los cuales serán atendidos en función del nivel de riesgo y del orden de llegada a la sala de emergencias.