# Trabajo Práctico N°2

### Informe - Problema N°1

# Simulación de una Sala de Emergencias con Cola de Prioridad

En esta actividad se desarrolló una simulación de atención en una sala de emergencias hospitalaria, utilizando una cola de prioridad implementada mediante un montículo binario (heap).

El objetivo principal fue **modelar la atención de pacientes en función de su nivel de riesgo**, asegurando que los casos más críticos sean atendidos antes que los menos urgentes, independientemente del orden de llegada.

## Objetivo

Implementar un sistema de atención médica que gestione la llegada y atención de pacientes en una sala de emergencias, utilizando una estructura de datos eficiente para manejar prioridades.

Además, analizar el funcionamiento y desempeño de la **cola de prioridad** aplicada a un contexto de simulación realista.

## Descripción del sistema

El sistema se compone en los módulos principales:

#### 1. Módulo Paciente

Definimos la clase Paciente, cuyos atributos incluyen:

- Nombre y Apellido (generados aleatoriamente).
- Nivel de riesgo: 1 (crítico), 2 (moderado) o 3 (bajo).
- **Descripción del riesgo:** texto explicativo del nivel asignado.
- **Número de llegada:** indica el orden en que cada paciente ingresa al sistema.

Se implementa el método especial \_\_1t\_\_ para definir el **criterio de comparación** entre pacientes:

• Se priorizan a los de mayor nivel de riesgo (1er: Crítico, 2do: Moderado, 3ero: Bajo).

En caso de empate, se atiende al que llegó primero

### 2. Módulo Montículo Binario

Se trata de un **árbol binario completo** donde cada nodo padre tiene un valor menor que sus hijos. Esto garantiza que el elemento con **mayor prioridad** (el paciente mas critico) siempre se encuentre en la **raíz del montículo**.

Sus operaciones principales son:

- insertar(k): agrega un nuevo elemento manteniendo la propiedad del montículo (mediante filtrado hacia arriba).
- eliminar Min(): extrae el elemento de menor valor (mayor prioridad).
- Construir Montículo(lista): construye un montículo a partir de una lista existente.

Esta implementación asegura una eficiencia de **O(log n)** tanto para inserciones como para eliminaciones.

## 3. Módulo Cola de Prioridad

Encapsula el montículo binario y ofrece una interfaz más sencilla para la simulación. Sus métodos principales son:

- agregar(paciente): inserta un nuevo paciente en la cola.
- avanzar(): atiende (extrae) al paciente con mayor prioridad.
- tamano(): devuelve la cantidad actual de pacientes en espera.

Análisis de complejidad:

Función Nivel de complejidad

insertar(k)	O(log n)
eliminarMin()	O(log n)
ConstruirMonticulo(lista)	O(n)
tamano()	O(1)
estaVacia()	O(1)

agregar(paciente)	O(log n)
avanzar()	O(log n)

# Resultados y observaciones

Los resultados obtenidos muestran que:

- Los pacientes con **mayor nivel de criticidad** siempre son atendidos antes que los moderados o leves, **sin importar su orden de llegada**.
- El uso del montículo binario garantiza un comportamiento eficiente incluso cuando aumenta la cantidad de pacientes

## Conclusión

La implementación de una **cola de prioridad mediante montículos binarios** resulta ideal para modelar sistemas donde el orden de atención depende de la urgencia y no del tiempo de llegada.

Este ejercicio permitió comprender la utilidad de las **estructuras de datos eficientes** para resolver problemas del mundo real, mejorando tanto el rendimiento como la organización de procesos en entornos simulados.

En resumen, la simulación confirmó que el uso de algoritmos basados en **prioridades y montículos binarios** es una herramienta poderosa para la gestión dinámica de eventos en tiempo real, como la atención en una sala de emergencias.