Universidad Nacional de Entre Ríos

➢*Facultad:* Facultad de Ingeniería.

➢*Carrera:* Licenciatura en Bioinformática y Bioingeniería.

➢*Cátedra:* Algoritmos y estructuras de datos

➢*Docentes:* Javier E. Diaz Zamboni, Jordán F. Insfrán, Juan F. Rizzato

➢*Título del trabajo:* "Aplicaciones de estructuras jerárquicas y grafos -"*Problema 1 ""*.

➢*Alumnos:*

- Almirón Spahn, María Paz
- Leiva, Giuliana
- Saravia, Lucía Milagros

➢*Fecha de entrega:* 31 de Octubre del 2025

Problema 1:

En este proyecto se implementa una ***cola de prioridad*** para modelar el triage en una sala de emergencias. Se simula la llegada de pacientes con tres niveles de riesgo (1=crítico, 2=moderado y 3=bajo) y se asegura que al atender siempre salga primero el paciente de mayor prioridad (nivel de riesgo menor). Si dos pacientes comparten el mismo nivel de riesgo, se utiliza como segundo criterio el orden de llegada (el que primero llega, se atiende primero).

***Objetivo*:**

El principal propósito de este proyecto era diseñar, programar y aplicar una estructura de datos adecuada para almacenar pacientes a medida que ingresan, de modo que la extracción del siguiente paciente a atender cumpla la política de prioridad: 1) menor nivel de riesgo, 2) a igual riesgo, menor orden (el que haya llegado antes).

***Estructuras*:**

Utilizamos un **montículo mínimo** como estructura subyacente a la clase "ColaPrioridad", debido a diversos factores, como lo son:

- Un montículo mínimo permite extraer el elemento con prioridad mínima (en este caso sería menor riesgo) en ***O (log n)*** y realizar inserciones en ***O (log n)***, lo que es eficiente cuando la cantidad de pacientes puede crecer.
- El montículo no exige mantener una lista ordenada completa, por lo que las inserciones son asintóticamente más rápidas que en una lista ordenada (inserción O(n))
- Usando el método "__lt__" en la clase "Paciente" se define el criterio de comparación (riesgo, luego orden). De esta forma el montículo puede seguir funcionando con cualquier tipo de objeto que implemente comparación, manteniendo la generalidad.

***Complejidad*:**

- Insertar/agregar paciente: ***O (log n)***
- Extracción del siguiente paciente (eliminar/atender paciente): ***O (log n)***
- Iteración completa sobre los elementos almacenados: ***O (n)***
- Memoria (almacenar pacientes): ***O (n)***

***Conclusión*:**

Para finalizar, en nuestra opinión, la combinación de "Paciente" (que define el criterio de comparación) y un "MonticuloMin" es una solución adecuada y genérica para este problema de triage: garantiza que el paciente más crítico sea atendido primero y, a igualdad de criticidad, respeta el orden de llegada. Su rendimiento es eficiente para colas con crecimiento dinámico y permite extender criterios con facilidad.