

## Práctica 1 - Algoritmos de Búsqueda

Esta práctica consiste en la implementación de varios algoritmos de búsqueda en C#, aplicados al problema de las  $n$ -reinas. Es importante que el código esté correctamente comentado, y organizado en métodos y clases. **Cada fichero .cs debe incluir al principio un comentario con los nombres de ambos miembros del grupo y el grupo de prácticas (lunes o jueves).**

### 1. Problema

A una empresa se le propuso resolver el problema de las  $n$ -reinas, que trata de colocar  $n$  reinas en un tablero de  $n \times n$  de tal manera que no se ataquen entre ellas.

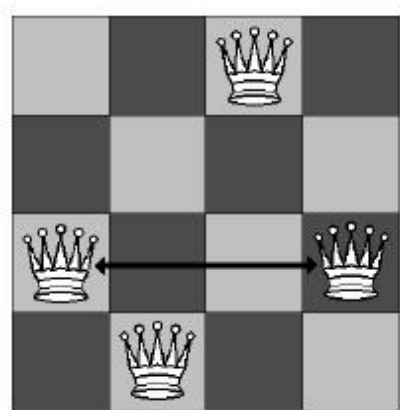


Figura 1: Ejemplo de  $n$ -reinas con  $n = 4$ .

La empresa puso a cargo de la tarea a un empleado, que resolvió el problema usando el método de búsqueda de  $A^*$ . Una vez visto el resultado, a la empresa le interesaba saber si se puede mejorar el resultado mejorando el código o usando otras técnicas de búsqueda. Lamentablemente, el empleado dejó la empresa, dejando un código totalmente sin comentar, e insertado en un único fichero llamado `n.reinas.py`. Además, lo hizo en un lenguaje de programación (Python) distinto al que utilizan el resto de módulos creados por la empresa (C#).

**Objetivo:** Retomar el trabajo dejado por el empleado que abandonó la empresa, creando una estructura más legible, y mejorando sus funcionalidades. En concreto, se han propuesto **3 objetivos semanales a realizar**, de manera incremental, hasta alcanzar el producto final.

## 2. Tareas

### 2.1. Semana 1: Especificación y traducción

La primera tarea será reordenar y dar un significado al código. Para ello se pide:

1. Transformar el código de Python en código de C#:
  - El funcionamiento debe ser análogo al del código de Python proporcionado
  - Se pueden modificar funciones para convertirlas en clases.
  - Las clases y métodos deben mantener el **mismo nombre**.
2. Estructurar el código, separando en distintos ficheros su contenido. En concreto, se pide:
  - El código ejecutable.
  - El algoritmo de búsqueda.
  - Las estructuras de almacenamiento de datos.
3. Documentar el código, explicando qué hace cada una de las funciones. En concreto, se pide tener especial atención al funcionamiento del algoritmo:
  - ¿Cuál es el estado inicial?
  - ¿Dado un estado, cómo se escogen los vecinos?
  - ¿Cuándo finaliza el algoritmo?
  - ¿Qué utilidad tienen las variables incluidas en la clase `ColaDePrioridad`?
4. Documenta el trabajo hecho en un fichero `semana_1.md`, que se debe entregar con el resto del proyecto.

## 2.2. Semana 2: Búsqueda no Informada

La segunda tarea será resolver el problema usando otros modelos de búsqueda no informada. Para ello se pide:

1. Resolver el problema usando la búsqueda en anchura:
  - Ejecuta el código, empezando en `reinas = 4`, y sube el valor hasta que el número de nodos evaluados sea mayor a 15 000. Indica en el fichero el número de nodos evaluados para cada valor de `reinas` probado.
2. Resolver el problema usando la búsqueda en profundidad:
  - Ejecuta el código, empezando en `reinas = 4`, y sube el valor hasta que el número de nodos evaluados sea mayor a 15 000. Indica en el fichero el número de nodos evaluados para cada valor de `reinas` probado.
3. Documenta el trabajo realizado, así como los resultados obtenidos, en un fichero llamado `semana_2.md`. Dicho fichero contendrá:
  - Una explicación de cómo se han implementado las soluciones para cada búsqueda.
  - Se indicará, para cada ejecución, el número de nodos evaluados para cada valor de `reinas` probado.
  - Explicar, de manera razonada, cuál de los dos modelos de búsqueda no informada es más adecuado para la resolución de esta tarea.

## 2.3. Semana 3: Búsqueda Informada

La tercera tarea será resolver el problema usando modelos de búsqueda informada. Para ello se pide:

1. Mejorar la resolución del problema usando  $A^*$ :
  - Modifica la función de coste para optimizar la búsqueda.
  - Modifica la función de heurística para optimizar la búsqueda.
  - Ejecuta el código, empezando en `reinas = 4`, y sube el valor hasta que el número de nodos evaluados sea mayor a 15 000. Indica en el fichero el número de nodos evaluados para cada valor de `reinas` probado.

2. Resolver el problema usando Búsqueda Avara:
  - Ejecuta el código, empezando en `reinas = 4`, y sube el valor hasta que el número de nodos evaluados sea mayor a 15 000. Indica en el fichero el número de nodos evaluados para cada valor de `reinas` probado.
3. Documenta el trabajo realizado, así como los resultados obtenidos, en un fichero llamado `semana_3.md`. Dicho fichero contendrá:
  - Una explicación de cómo se han implementado las soluciones para cada búsqueda.
  - Se indicará, para cada ejecución, el número de nodos evaluados para cada valor de `reinas` probado.
  - Explicar, de manera razonada, cuál de los dos modelos de búsqueda no informada es más adecuado para la resolución de esta tarea.

## Entrega

Se debe entregar en el campus virtual un proyecto de C# en formato `.zip` que incluya las soluciones a los problemas indicados y la documentación que se pide. El límite para la entrega será el lunes 02/03/26 a las 14:00 para el grupo del lunes y el jueves 05/03/26 a las 14:00 para el grupo del jueves.

## Defensa

Para que la práctica se considere entregada, deberá ser defendida. Se priorizará hacer la defensa en clase (lunes 02/03/26 o jueves 05/03/26), pero debido a la cantidad de estudiantes es posible que haya que realizar defensas fuera de clase. En cualquier caso, todas las defensas se realizarán **antes del viernes 06/03/26**.