# MASTERMIND

## Breve descripción de las estructuras de datos y algoritmos

**Projectes de Programació:** Quadrimestre Primavera 2022-23

**Grup 42.5:**
Elsa Boix Solina
Joel Macías Rojas
Ricard Medina Amado
Marcel Sánchez Roca


UNIVERSITAT POLITÈCNICA
DE CATALUNYA
BARCELONATECH

# Breve descripción de las estructuras de datos y algoritmos

## Explicación de las estructuras de datos utilizadas

1. **possibleCodes**: Es una lista de combinaciones representadas por enteros que representa las combinaciones que aún no han sido descartadas.
2. **totalcombinacionesPosibles**: Es una lista de todas las posibles combinaciones que pueden existir en la partida. Es generada a partir del nivel de dificultad del juego, representado por la clase NivelDificultad, que determina el número de columnas y repeticiones permitidas en la combinación.
3. **solucionesEnviadas**: Es una lista de listas de enteros que representa las combinaciones de colores que han sido enviadas por el algoritmo en turnos anteriores del juego. Se utiliza para llevar un registro de las combinaciones enviadas y devolverlas al finalizar el turno.
4. **enviosCandidatos**: Es una lista de combinaciones que representa las posibles combinaciones de colores que el algoritmo considera enviar en el próximo turno del juego. Se genera a partir de las combinaciones de colores en **totalcombinacionesPosibles** que tienen la puntuación más baja basándonos en las respuestas recibidas.
5. **envioActual**: Es una lista de enteros que representa la combinación de colores que el algoritmo enviará en el turno actual del juego.
6. **solucion**: Es una lista de enteros que representa la combinación de colores secreta que el algoritmo trata de adivinar.

## Explicación del algoritmo

El algoritmo utiliza las estructuras de datos mencionadas para generar nuevas combinaciones a enviar, evaluar las posibles soluciones basándonos en las respuestas recibidas y decidir qué combinación enviar en cada turno, con el objetivo de adivinar la combinación secreta en el menor número de intentos posible. A continuación se describe su funcionamiento paso por paso:

Se inicializa el número de ronda (ronda) en 1 y se guarda el nivel de dificultad del juego (representado por el objeto NivelDificultad) en la variable nivel.

Se implementan tres métodos privados: getMinScore, getMaxScore y generaNuevoEnvio, que son utilizados para calcular la puntuación mínima, máxima y generar la nueva combinación de código a ser enviada en el siguiente turno, respectivamente. Además se inicializa **totalcombinacionesPosibles** mediante backtracking obteniendo así todas las combinaciones posibles. **PossibleCodes** en este momento tiene el mismo contenido.

En primer lugar, se realiza un envío predeterminado para obtener el primer *feedback,* comparando el ***envioActual*** con la *solución*. Si este *feedback,* nos indica que ya hemos encontrado la solución se devuelve ***solucionesEnviadas*** conteniendo únicamente este envío preestablecido.

En el caso que no coincida con la solución, el algoritmo generará una nueva. El método ***generaNuevoEnvio****()* es el corazón del algoritmo.

Primero, se crea un mapa llamado **contadorPuntuaciones** para contar cuántas veces aparece cada resultado de fichas posibles (representado por un String). Para cada combinación de **totalcombinacionesPosibles** se comparará con todas las de **possibleCodes** y se mantendrá un recuento de los *feedbacks*.

Una vez iterado por todo **possibleCodes**, se obtiene la combinación con más apariciones, se inserta en **puntuaciones** y se vacía el mapa **contadorPuntuaciones** para dejarlo listo para la siguiente iteración. Se repite para todo **totalcombinacionesPosibles.**

Una vez obtenidas todas combinaciones con el número máximo de apariciones, nos quedaremos con el valor más pequeño y nos quedaremos con todas combinaciones con ese número mínimo de apariciones que insertamos en **enviosCandidatos**.

Únicamente queda elegir qué combinación enviar. Para ello se llama a **obtenSiguienteEnvio(),** que mirará primero si alguno de los **enviosCandidatos** aparecer en los **possiblesCodes**, si es así envía la primera combinación que

aparezca, sinó, la primera combinación que aparezca en **totalcombinacionesPosibles.**

En resumen, el algoritmo FiveGuess utiliza una estrategia de búsqueda exhaustiva para generar combinaciones candidatas a ser enviadas en cada turno, basándose en la puntuación obtenida por cada combinación en intentos anteriores, con el objetivo de encontrar la solución correcta con la menor cantidad de intentos posibles.