Métodos de Búsqueda

Sistemas de Inteligencia Artificial

TP1

- Banfi, Malena
- Fleischer, Lucas
- Perez Rivera, Mateo
- Szejer, Ian

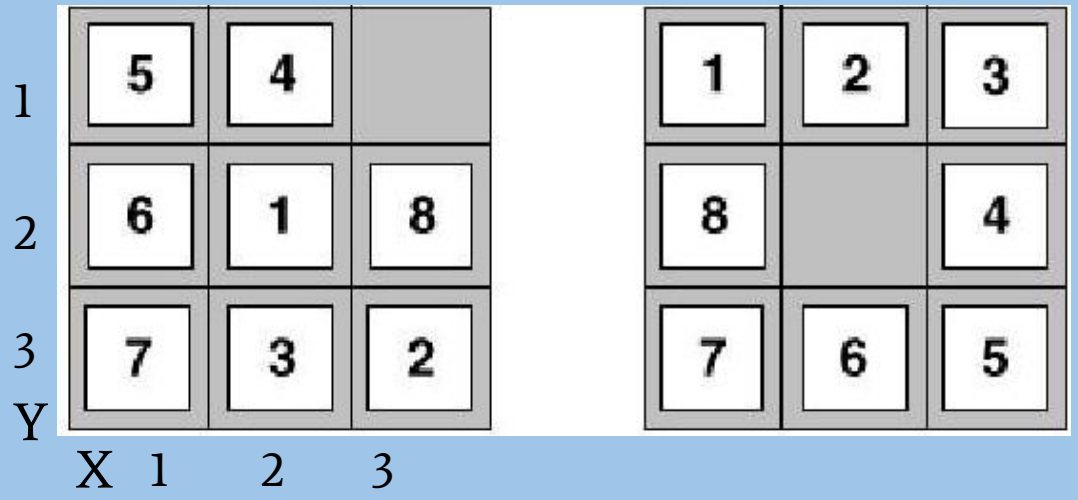
Grupo 3

Ejercicio 1

8-puzzle

Estructura de estado:

- Para el problema del 8-puzzle, se puede utilizar una matriz de 3x3, donde cada posición de la matriz representa la ubicación de una ficha numerada del [1-8] en el tablero.
- En el lugar restante se encuentra un espacio en blanco.

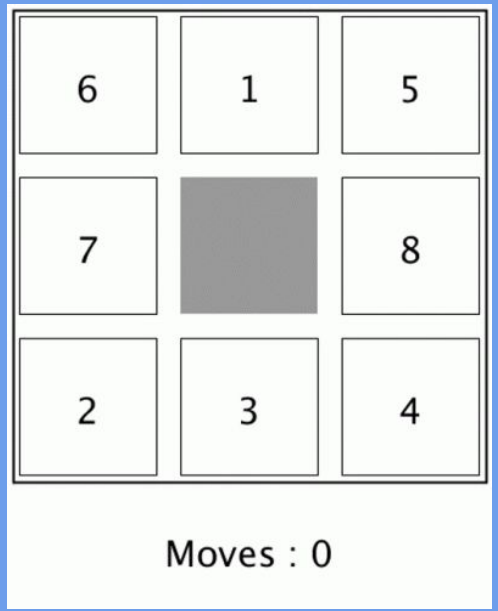


Acciones:

- Mover números adyacentes al espacio vacío.
- Se puede desplazar hacia arriba, abajo, izquierda o derecha. Esto depende de la ubicación del espacio vacío.

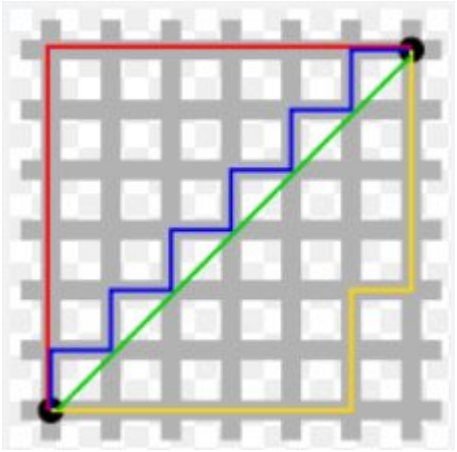
Objetivo:

- El objetivo del juego es mover las piezas para que estén en orden numérico, es decir, con el 1 en la esquina superior izquierda y los números aumentando de izquierda a derecha y de arriba a abajo.



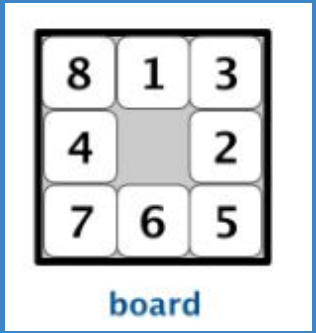
Heurísticas no triviales, admisibles

- Distancia Manhattan
- Número de piezas mal colocadas



Número de piezas mal colocadas

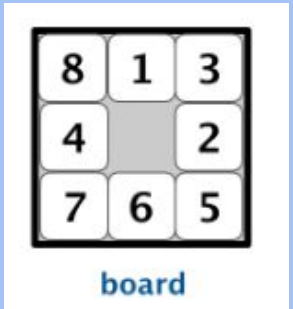
- La heurística cuenta el número de piezas que están en una posición incorrecta y las suma para obtener una estimación de la cantidad de movimientos necesarios para resolver el rompecabezas.
- Es admisible ya que nunca sobrestima el costo real



- Piezas desacomodadas: 1, 2, 4, 8 => total de 4
- Piezas acomodadas: 5, 6, 7, 3 => total de 4
- Valor de la heurística: 8-4 = 4

Distancia Manhattan

- Es la sumatoria de las distancias verticales y horizontales de cada cuadrado desde su posición actual hasta su posición objetivo.
- Es admisible ya que nunca sobreestima el costo real.



- 8 => vertical 1, horizontal 0 = 1
- 1 => vertical 0, horizontal 1 = 1
- 2 => vertical 1, horizontal 1 = 2
- 4 => vertical 0, horizontal 2 = 2

Valor de la heurística = 6

$$\text{Manhattan Distance} = d(x,y) = \left(\sum_{i=1}^{m} |x_i - y_i|\right)$$

¿Métodos de búsqueda desinformados o informados?

Es posible utilizar métodos de búsqueda desinformados pero estos métodos no utilizan ninguna información específica sobre el problema, como las heurísticas, para guiar la búsqueda y pueden ser muy ineficientes en términos de tiempo y espacio.

Por lo tanto, decidimos usar algoritmos de búsqueda informados, ya que estos utilizaran las heurísticas para guiar la búsqueda y reducir el espacio y tiempo de búsqueda.

Dicho esto, BFS y DFS quedan descartados.

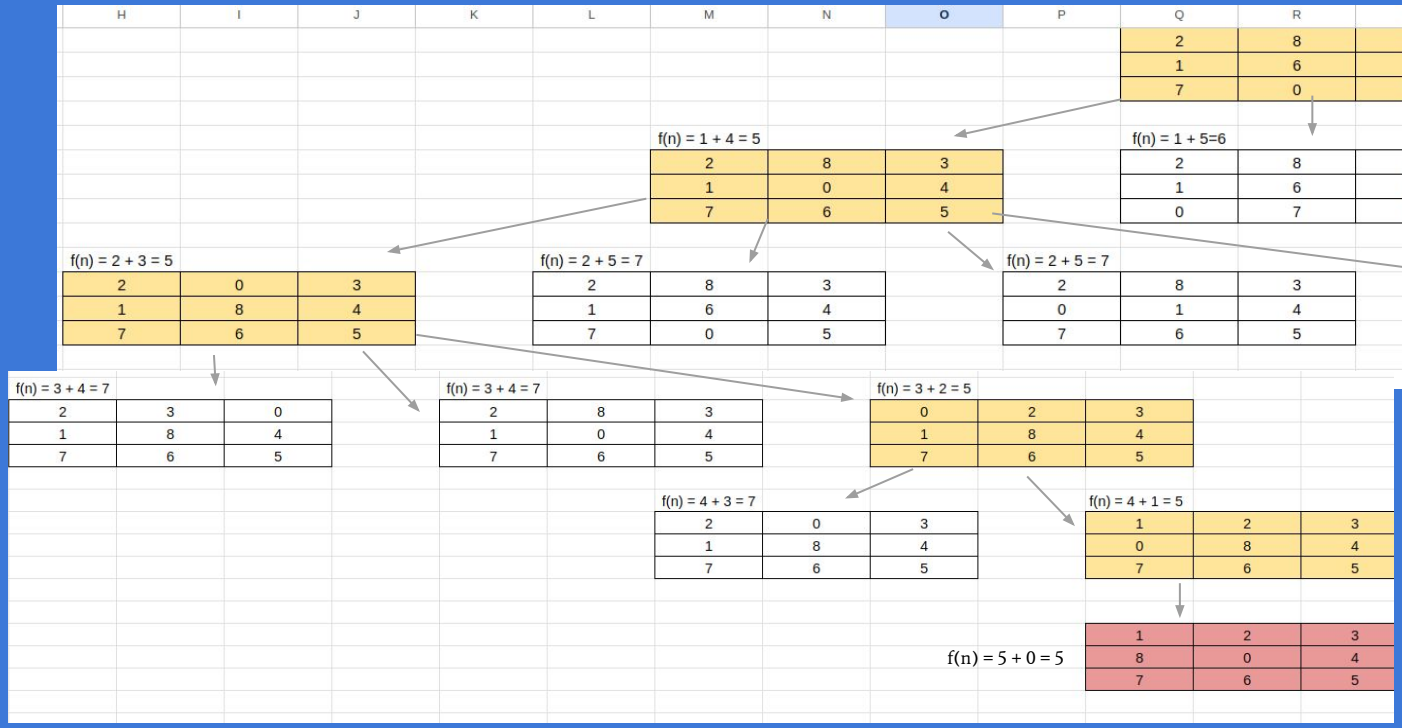
Greedy Search

- Solo considera el costo heurístico estimado desde el nodo actual hasta la meta, y no tiene en cuenta el costo real del nodo. Esto significa que Greedy Search puede ser más rápido que A*, pero no garantiza encontrar la solución óptima.

A* Search

- Este método es muy efectivo ya que garantiza encontrar la solución óptima, siempre y cuando la heurística utilizada sea admisible y consistente.
- A* sería la mejor opción. Usa una cola de prioridades con la suma del costo a llegar a un nodo + el costo establecido por la heurística. Gracias a esto considera el camino más corto y óptimo a la solución.

A* Search con heurística de Distancia Manhattan



$$f(n) = g(n) + h(n)$$

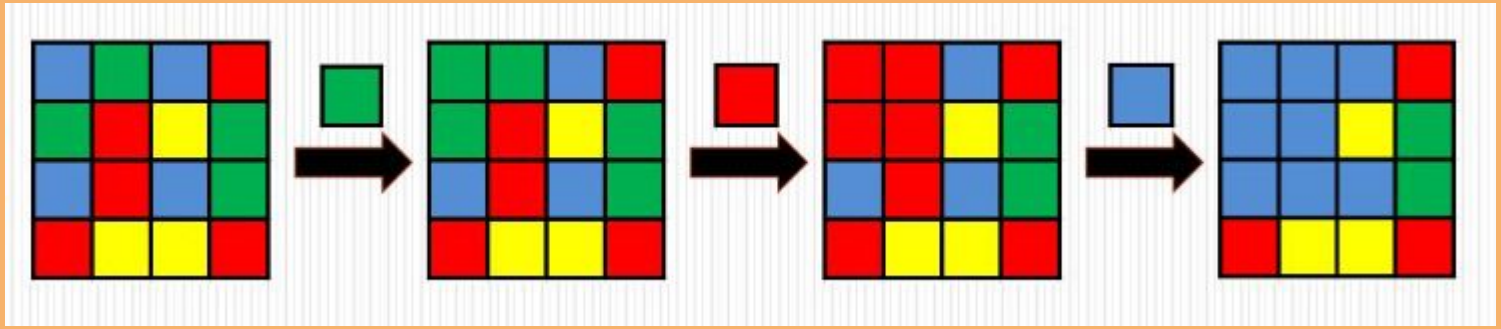
La función se encuentra compuesta por:

g(n) : es el costo de las movidas realizadas.
h(n) : es la función heurística. Utilizaremos la heurística de distancia Manhattan.

(http://www.mygamesworld.com/game/7682/Fill_Zone.html)

Estructura de estado

- La estructura de estado del juego Fill Zone se puede representar mediante una matriz bidimensional de N x N celdas, donde cada celda puede contener un número que representa el color o estado actual de esa celda.
- El jugador puede modificar la estructura de estado al elegir un nuevo color para llenar las celdas adyacentes del mismo color.
- La estructura de estado se actualiza en cada turno hasta que se completa el juego (llegando a un mismo color).



Heurísticas encontradas

- Heurística 1: Se estima cuántos pasos van a faltar para llegar a la celda más lejana del tablero (utiliza Dijkstra para obtener ese valor).
  - Es admisible, ya que para finalizar el juego, se tiene que llegar a la celda más lejana y la cantidad de pasos no podrá ser inferior.

- Heurística 2: Se calculan cuántos colores diferentes quedan en el tablero y en base a eso, estima cuántos pasos quedan para finalizar el juego.
  - Es admisible, ya que se debe cambiar al menos una vez cada color para completar el juego

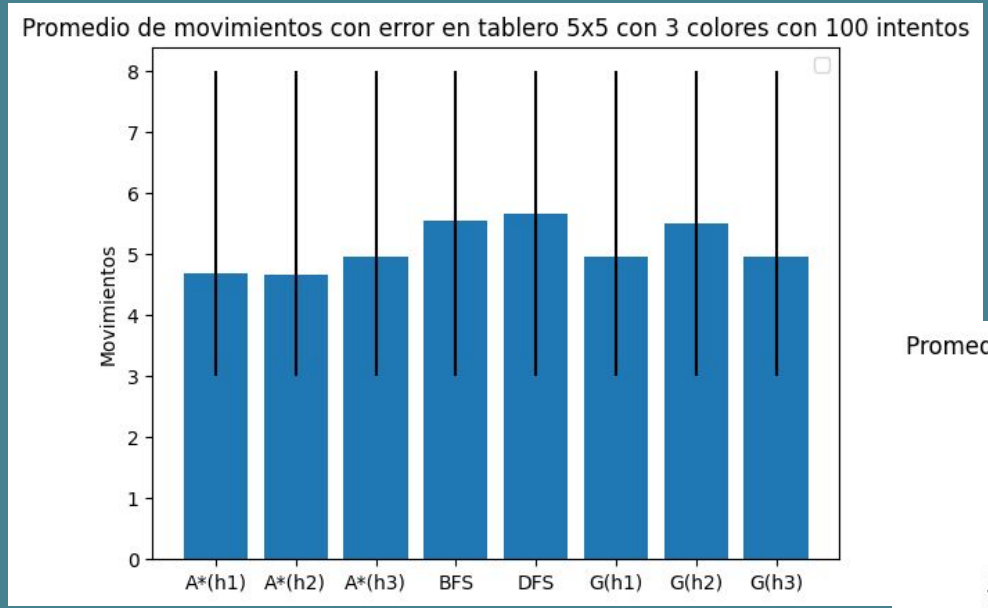
- Heurística 3: Desde el color actual, cuantas posiciones van a cambiar con el cambio de color.
  - No es admisible

Resultados

A continuación se van a presentar gráficos con los resultados obtenidos luego de hacer 100 corridas con distintos tamaños de tablero y utilizando distintos colores:

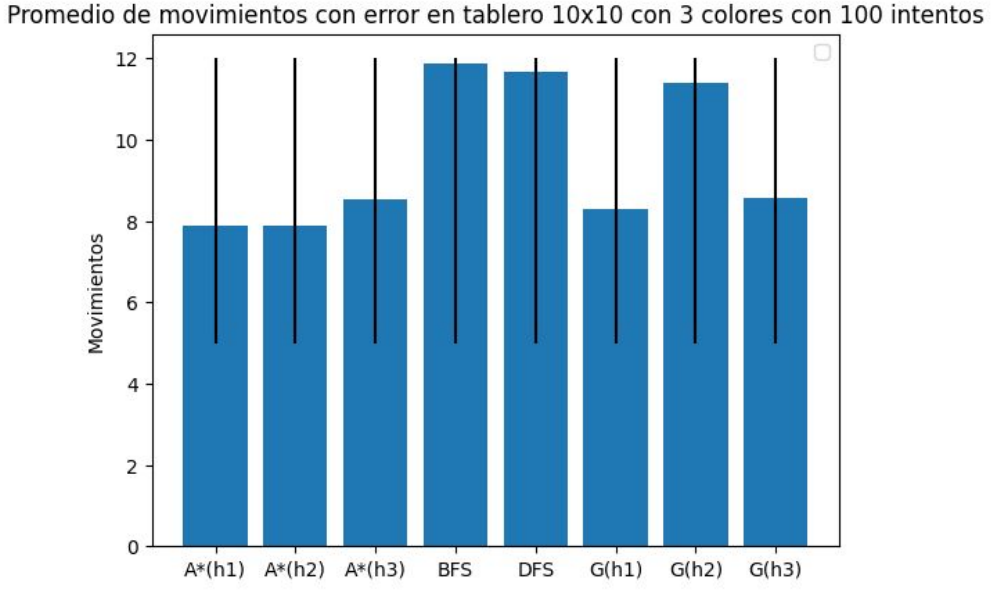
- Tablero de 5x5 con 3 colores
- Tablero de 10x10 con 3 colores

Movimientos

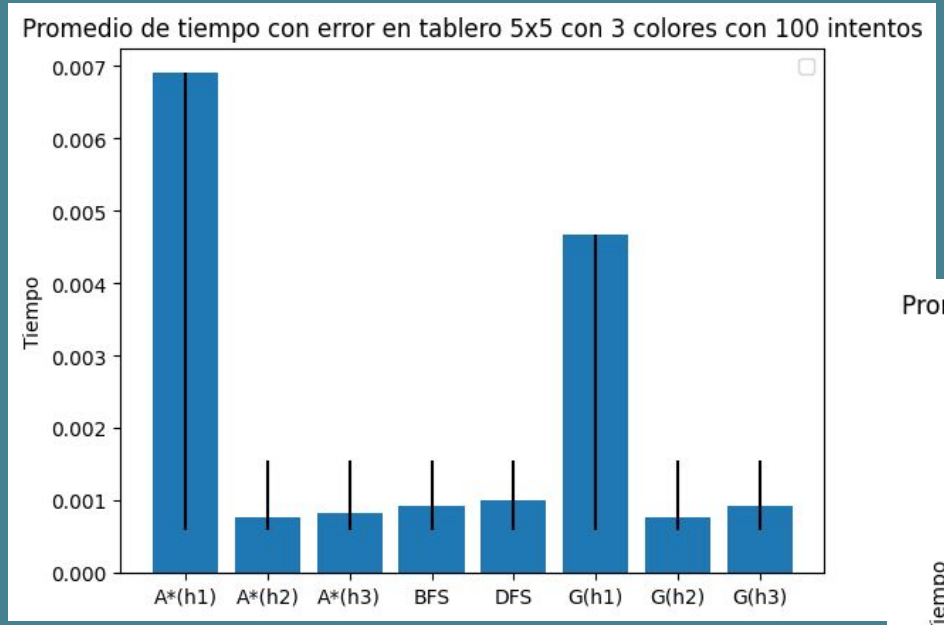


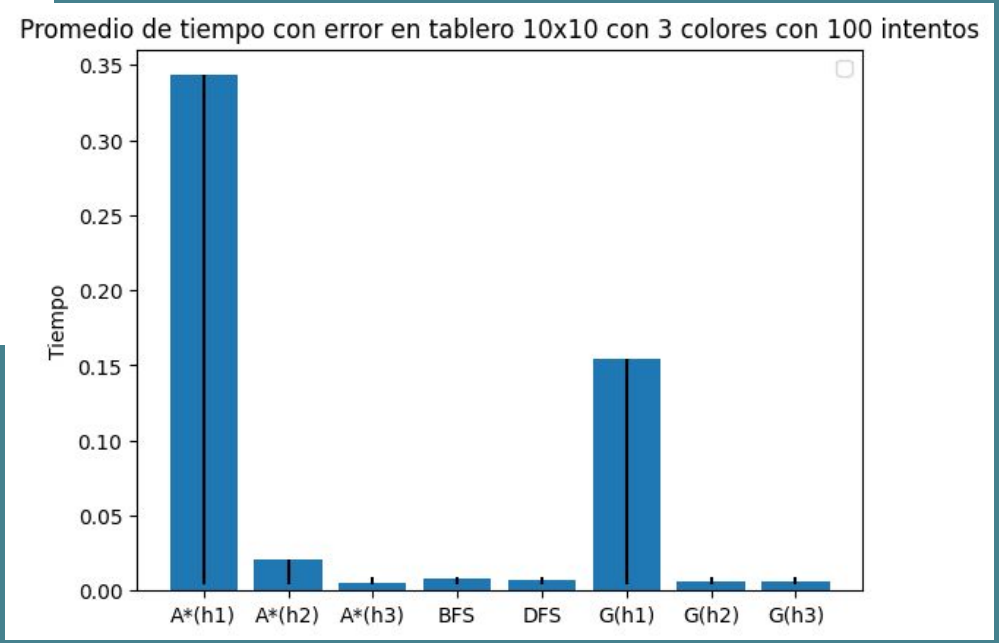
Podemos observar cómo aumentan las diferencias entre cantidad de movimientos a medida que se expande el rango de celdas (5x5 a 10x10).

Al aumentar la cantidad de celdas en el tablero vemos como BFS, Greedy con la heurística 2 y DFS aumentan la cantidad de pasos.



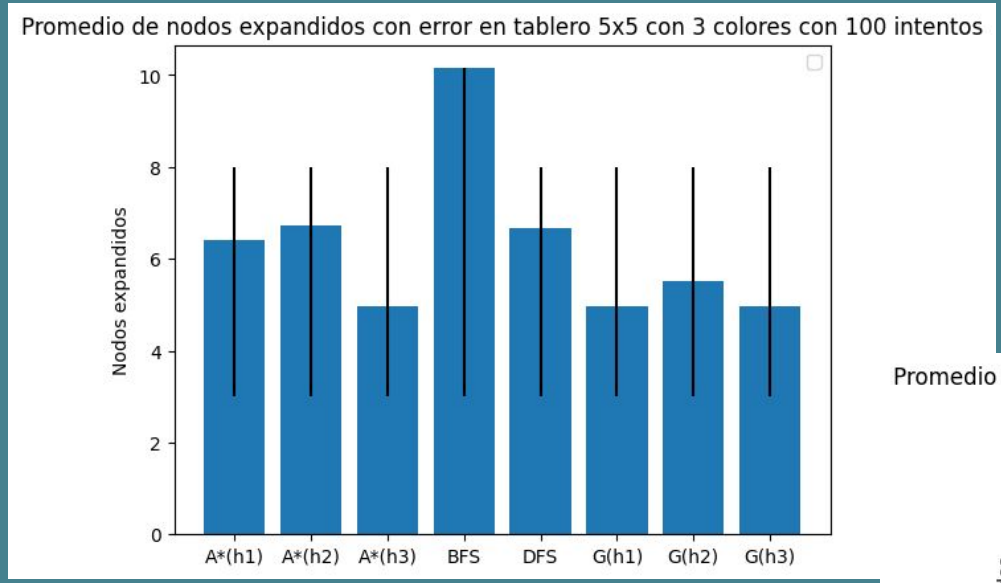
Tiempo



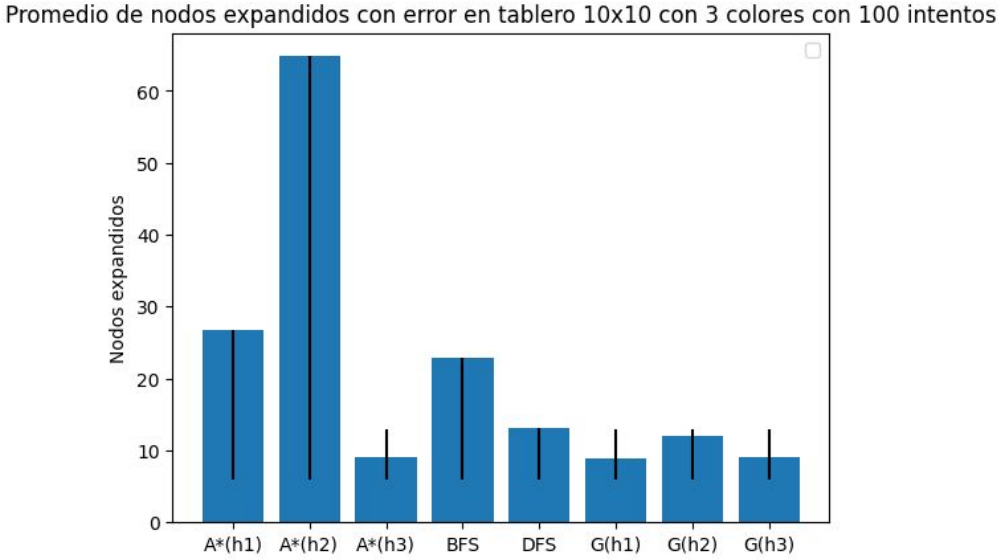


Como se puede observar, podemos concluir en que el algoritmo con mayor tiempo de ejecución es el A* con la heurística 1

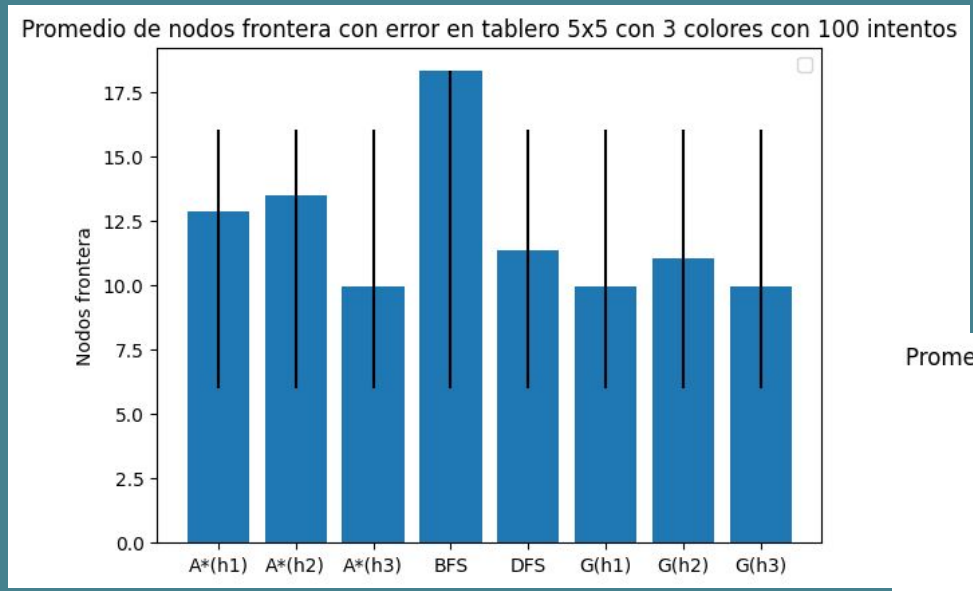
Nodos Expandidos



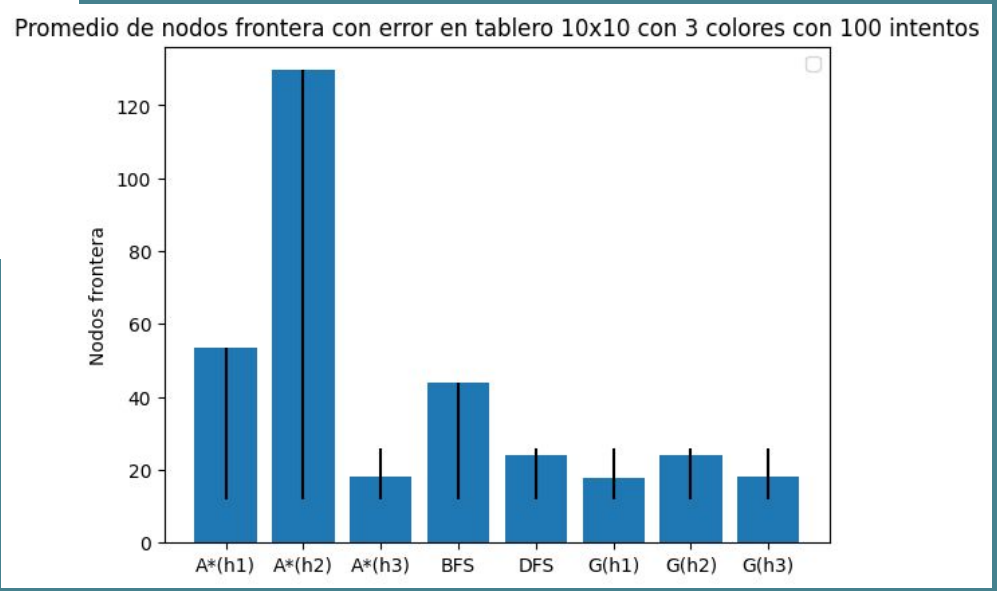
A medida que se va agrandando el tablero podemos observar que el algoritmo A* con la heurística 2 (colores restantes) expande muchos más nodos que el resto.



Nodos Frontera



En grillas más pequeñas el BFS al no tener una heurística a la cual seguir, debe expandir todos los nodos posibles.



Se observa que al aumentar las dimensiones de la grilla, los algoritmos informados con heurística 2 acumulan la mayor cantidad de nodos, debido a la cantidad de ramas diferente que evalúan

Fin.