Índice

Resumen

En este trabajo se realiza algunas implementaciones, comparaciones y mediciones del juego de n-puzzle como medir el tiempo que toma en terminar un algoritmo, medir la cantidad de estados expandidos, también incluye la elaboración de una tabla para comparar los resultados de los distintos algoritmos en cuanto a su tiempo, nodos expandidos y si encuentra o no la solución optima para distintos valores de N incluyendo valores relativamente grandes.

Introducción

Se busca resolver el puzzle para matrices de NxN mediante búsqueda en ancho y búsqueda A* probando con H(n): Distancia de Manhattan y Nro de piezas en lugar incorrecto, teniendo como parámetros el valor de N para el tamaño de la tabla y H(n) a utilizar por el A*. También se aplicará técnica para producir los tableros en desorden para testear el algoritmo y poder encontrar la solución que realice la menor cantidad de movimientos posibles

Descripción del problema

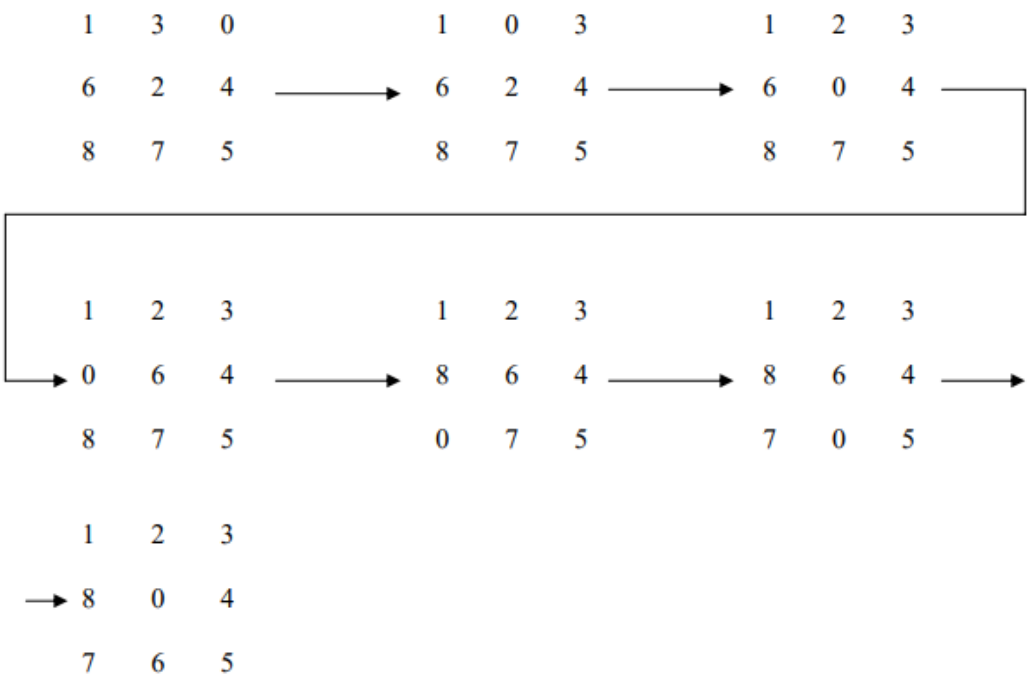
Dado algunos ejemplos de 8-puzzle aplicando la heurística de la distancia de Manhatan se ha demostrado que en problemas cortos es la que mejores resultados ha obtenido, Para llegar a estas conclusiones nos hemos basado en la resolución de 3 tableros diferentes de 8-puzzle con una complejidad de 6, 12 y 18 movimientos.

En todas ella se pretende llegar a la misma matriz solución:

1	2	3
8	0	4
7	6	5

- Puzzle con 6 movimiento (el 0 representa la casilla vacía):

  Para obtener su solución mas optima se debe resolver en 6 movimientos como se indica en la figura



H2 = 6 Nº de movimientos = 6 Nodos generados = 18

Todas las heurísticas utilizadas llegan a una solución óptima, pero la diferencia principal radica en le cantidad de recursos que necesitan para resolver el problema.

- Puzzle de 12 movimiento
  Para la prueba de 12 movimientos hemos utilizado la siguiente matriz:

6	1	4
8	3	2
0	7	5

H2 = 10 Nº de movimientos = 12 Nodos generados = 38

Una solución óptima encontrada para la resolución de esta matriz de forma óptima es: Arriba, Arriba, Derecha, Abajo, Izquierda, Abajo, Derecha, Arriba, Derecha, Arriba, Izquierda, Abajo.

- Puzzle de 18 movimiento

Para la prueba de 18 movimientos hemos utilizado la siguiente matriz:

6	3	1
8	0	4
7	5	2

H2 = 10 Nº de movimientos = 18 Nodos generados = 5421

Una solución óptima encontrada para la resolución de esta matriz de forma óptima es: Arriba, Izquierda, Abajo, Derecha, Arriba, Derecha, Abajo, Abajo, Izquierda, Arriba, Izquierda, Arriba, Derecha, Abajo, Derecha, Arriba, Izquierda, Abajo

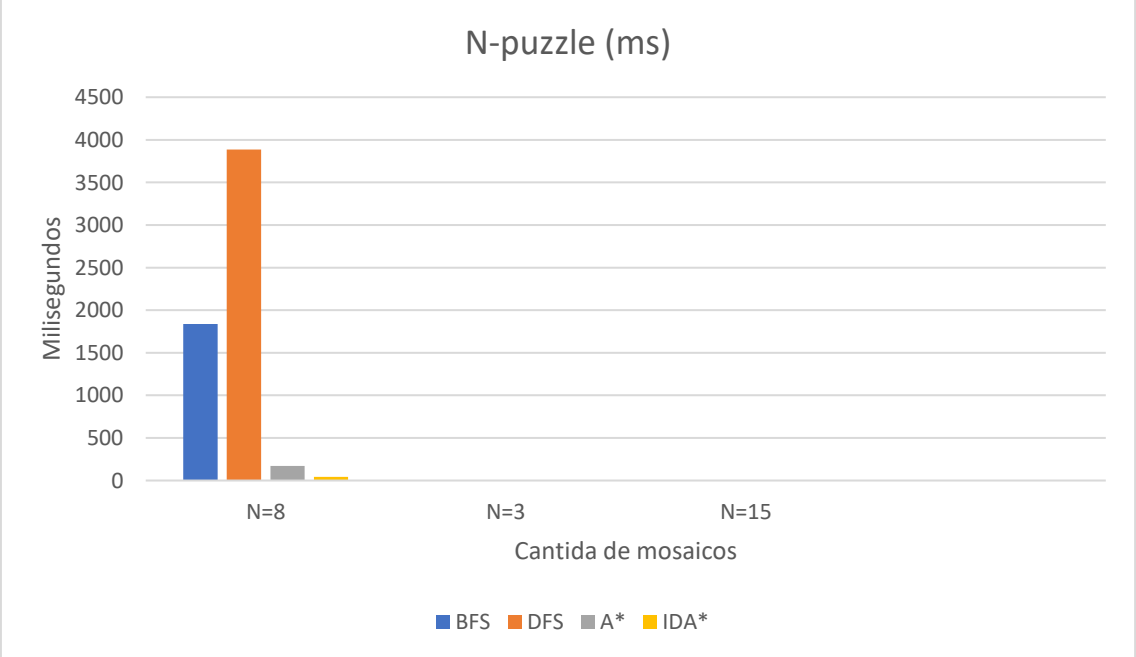
Algoritmos Implantados

- BFS (Búsqueda en anchura) : es un algoritmo de búsqueda para lo cual recorre los nodos de un grafo, comenzando en la raíz (eligiendo algún nodo como elemento raíz en el caso de un grafo), para luego explorar todos los vecinos de este nodo.
- DFS (Búsqueda en profundidad) : Su funcionamiento consiste en ir expandiendo cada uno de los nodos que va localizando, de forma recurrente (desde el nodo padre hacia el nodo hijo). Cuando ya no quedan más nodos que visitar en dicho camino, regresa al nodo predecesor, de modo que repite el mismo proceso con cada uno de los vecinos del nodo. Cabe resaltar que, si se encuentra el nodo antes de recorrer todos los nodos, concluye la búsqueda.
- A* : algoritmo de búsqueda inteligente o informada que busca el camino más corto desde un estado inicial al estado meta a través de un espacio de problema, usando una heurística óptima.
- IDA* : El algoritmo de Búsqueda en Profundidad Iterativa, es un algoritmo ciego que aplica reiteradamente la búsqueda en profundidad. En cada aplicación incrementa en una unidad la profundidad de la búsqueda.

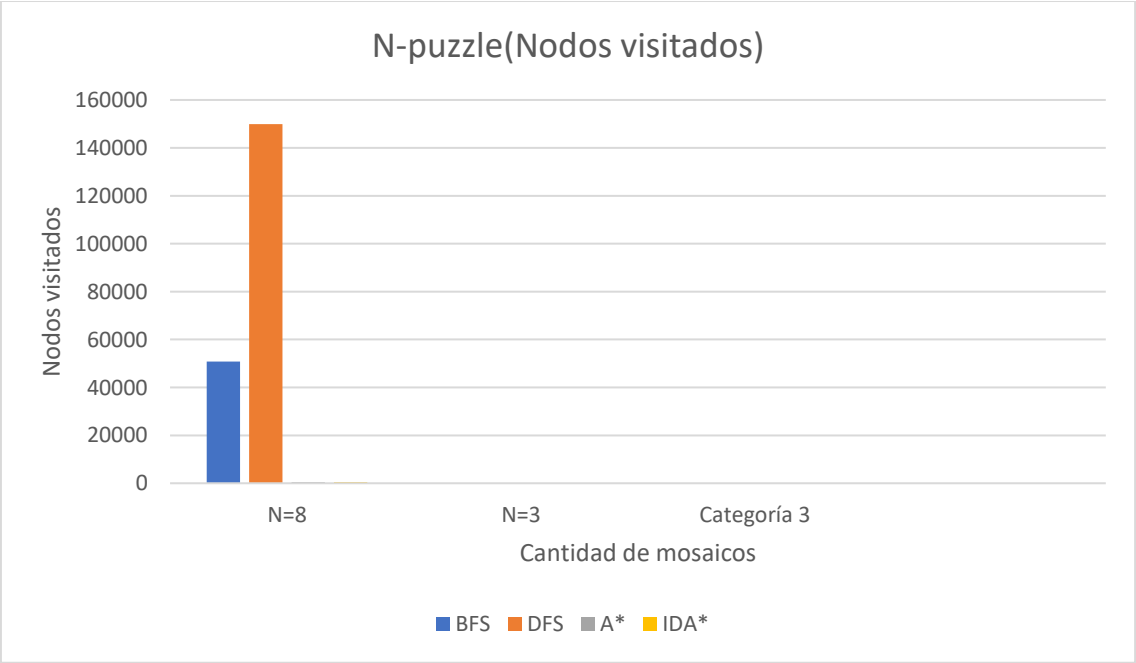
Resultados Experimentales

Para el primer experimento mediremos la eficiencia en cuanto a Tiempo de los distintos tipos de algoritmos

- Mediremos el tiempo que tarde en resolver la tabla con N=3 , N=8 , N=15



- El N=3 lanza valores muy pequeños (1ms) por lo que no es visible en el grafico
    - Con N=15 ya ocurre un desbordamiento de memoria (Killed)



- El N=3 lanza valores muy pequeños (12,8,12,12 nodos respectivamente) por lo que no es visible en el grafico
- Con N=15 ya ocurre un desbordamiento de memoria (Killed)

Conclusiones

En problemas de complejidad pequeña es decir al número reducido de movimientos para resolver el problema, la heurística más eficaz es la de Manhattan, ya que es capaz de resolver el problema con menor consumo de memoria ya que utiliza menos nodos.

A medida que la complejidad del problema va aumentando se va comprobando como la utilización de nodos por parte de las diferentes heurísticas va aumentando en grandes cantidades dejando de ser Manhattan la mejor opción para ser utilizada.

Referencias

- https://www.guru99.com/difference-between-bfs-and-dfs.html
- https://tristanpenman.com/demos/n-puzzle
- https://www.infor.uva.es/~calonso/IAI/TrabajoAlumnos/Algoritmo_A.pdf
- https://www.geeksforgeeks.org/a-search-algorithm/