Universidad Nacional de Colombia
Introducción a los sistemas inteligentes 2020-I

8-puzzle

Jherson Adrian Medina Correa
Manuel David Medrano Monroy
Nicolas Eduardo Pardo Arias

Introducción

- Dado un tablero 3x3 con 8 piezas y un espacio vacío, donde el objetivo es llegar a la configuración final.
- Los únicos movimientos permitidos con mover cualquier pieza adyacente al espacio vacío a este.

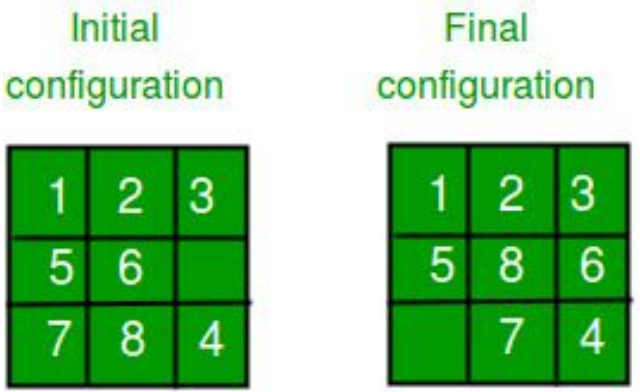

Figura 1: Tablero de 8-puzzle.

¿Cómo resolverlo?

Para resolver el 8-puzzle en la cantidad mínima de movimientos posibles, se plantean cuatro algoritmos de búsquedas, donde se compara el tiempo que le toma a cada uno llegar a la solución.

- **BFS**: Búsqueda en amplitud donde se recorre el árbol de posibilidades por niveles. Se garantiza que la solución encontrada es la que toma la menor cantidad de movimientos.
- **Limited DFS:** Búsqueda en profundidad limitada donde en caso de que una rama sea más profunda que un límite, esta se deja de explorar.
- **Iterative DFS:** Búsqueda en profundidad iterativa muy similar a la anterior donde se van probando distintas profundidades límites de manera iterativa.

¿Cómo resolverlo?

- **A* Search:** Algoritmo de búsqueda basado en heurísticas, muy similar a Dijkstra donde el orden de prioridad viene dado por una función *f* que toma valores dependiendo de la heurística aplicada.
  - **Distancia Manhattan total:** Por cada ficha se calcula la distancia Manhattan entre la posición actual y la posición en que debería estar, luego se calcula la suma total.

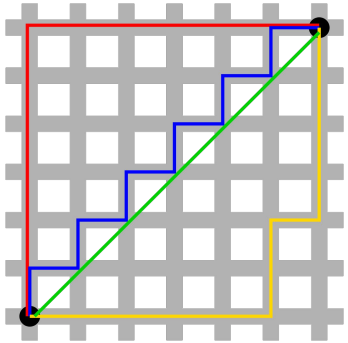

Figura 2: Distancia Manhattan

  - **Número total de piezas incorrectas:** Se calcula el número total de piezas que no están en la posición correcta.

Resultados

Después de treinta iteraciones, cada una con distintas configuraciones de juego, la figura 3 muestra los tiempos promedio, dado en microsegundos, de cada algoritmo.

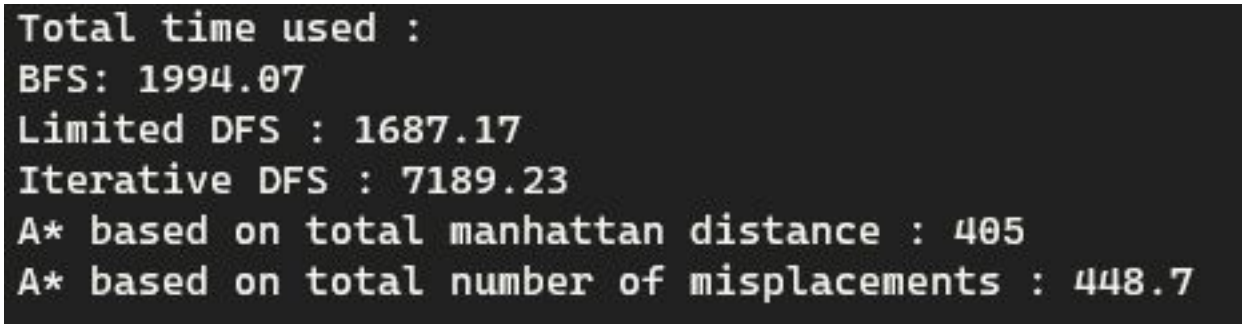

Figura 3: Tiempo promedio de cada búsqueda

Conclusiones

- A pesar de que la BFS encuentra una solucion mas rapido que la DFS iterativa y limitada, su propia naturaleza hace que sea unas cuatro veces más lenta que las heurísticas probadas en A*.
- La heurística de Distancia Manhattan total rusto ser más eficiente, por lo menos en tiempo, que el total de piezas incorrectas.

Referencias

- GeeksforGeeks. 2020. *8 Puzzle Problem Using Branch And Bound - Geeksforgeeks*. [online] Available at: <https://www.geeksforgeeks.org/8-puzzle-problem-using-branch-and-bound/> [Accessed 5 July 2020].
- GeeksforGeeks. 2020. *A* Search Algorithm - Geeksforgeeks*. [online] Available at: <https://www.geeksforgeeks.org/a-search-algorithm/> [Accessed 5 July 2020].
- Implementacion en C++ utilizada: https://github.com/CCCPUN/sistint/blob/master/8-puzzle/8puzzle.cpp