**PRESENTACION DE ANÁLISIS DE PATHFINDING EN UN LABERINTO**

Este algoritmo crea un laberinto mediante un grafo utilizando el algoritmo Prim.

Luego, comprobamos las diferentes maneras para conseguir encontrar la solución a nuestro laberinto con los algoritmos: BFS, DFS, Dijkstra, y A\*. Que explicaremos en profundidad más tarde.

Estos son algoritmos de búsqueda y utilizando tracemalloc y timeit comprobamos como son de eficientes.

En la pantalla de inicio tienes diferentes opciones a elegir como resolver el laberinto y también si quieres saber cuanto tarda y la memoria utilizada.

BFS:

Breath- first search. Es un algoritmo de búsqueda de grafos que busca por niveles. Este algoritmo crea una cola y utilizando FIFO va sacando cada nodo hasta recorrer todo el grafo. Además, encontrando el camino más corto ya que recorre por niveles En nuestro algoritmo va buscando por niveles hasta encontrar la salida y luego crea el camino de vuelta al nodo inicio

DFS:

Depth-first search. También es un algoritmo de búsqueda que busca en profundidad. Este algoritmo crea un stack utilizando LIFO va haciendo pop a casa nodo de la lista hasta que buscando profundamente encuentra la salida.

Aquí también hace un back tracking para crear la solución

Dijkstra:

Dijkstra es un algoritmo voraz o greedy que en nuestro caso para implementarlo en un laberinto.

Utilizamos un mini heap para ir sacando el nodo con menos distancia. Donde cada nodo neighobuor le sumamos uno de la distancia anterior. Finalmente devolviéndote el camino mas corto localmente según el algoritmo

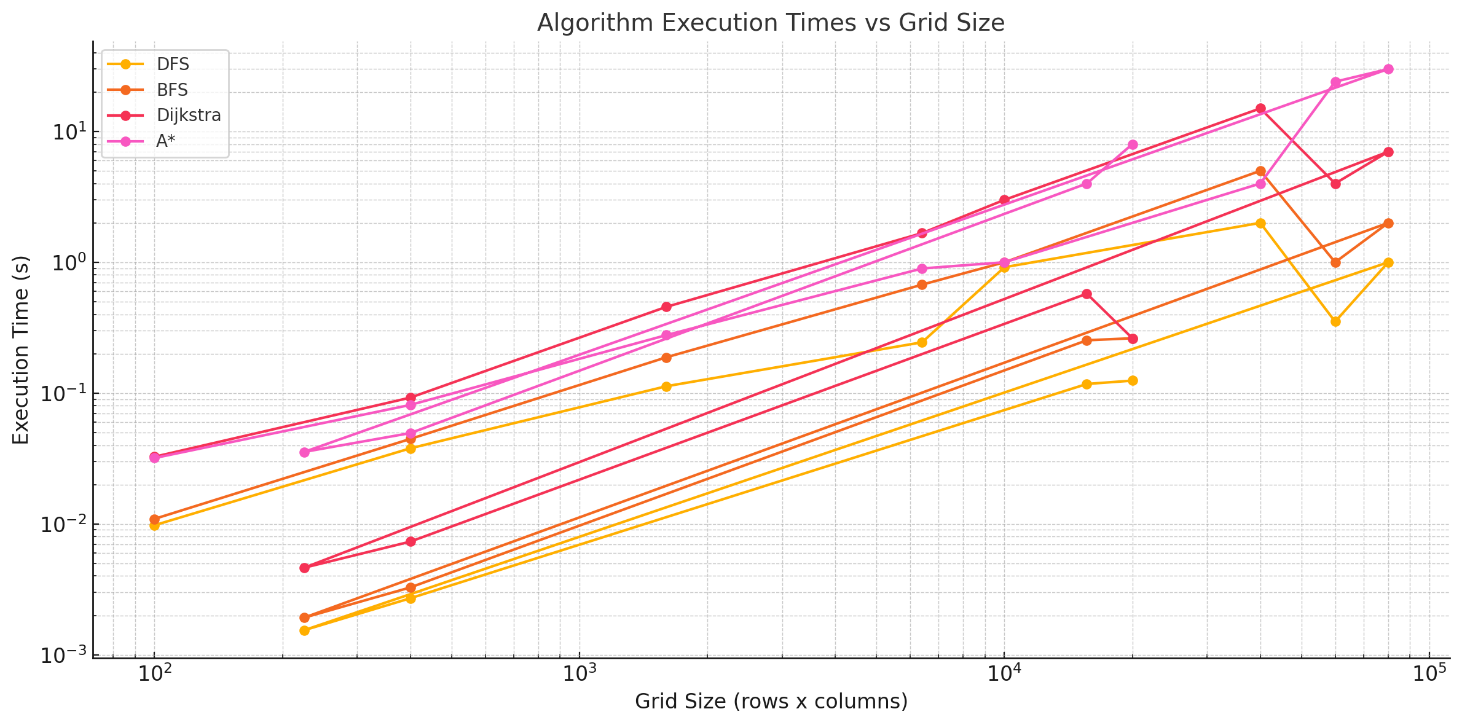
A\*:

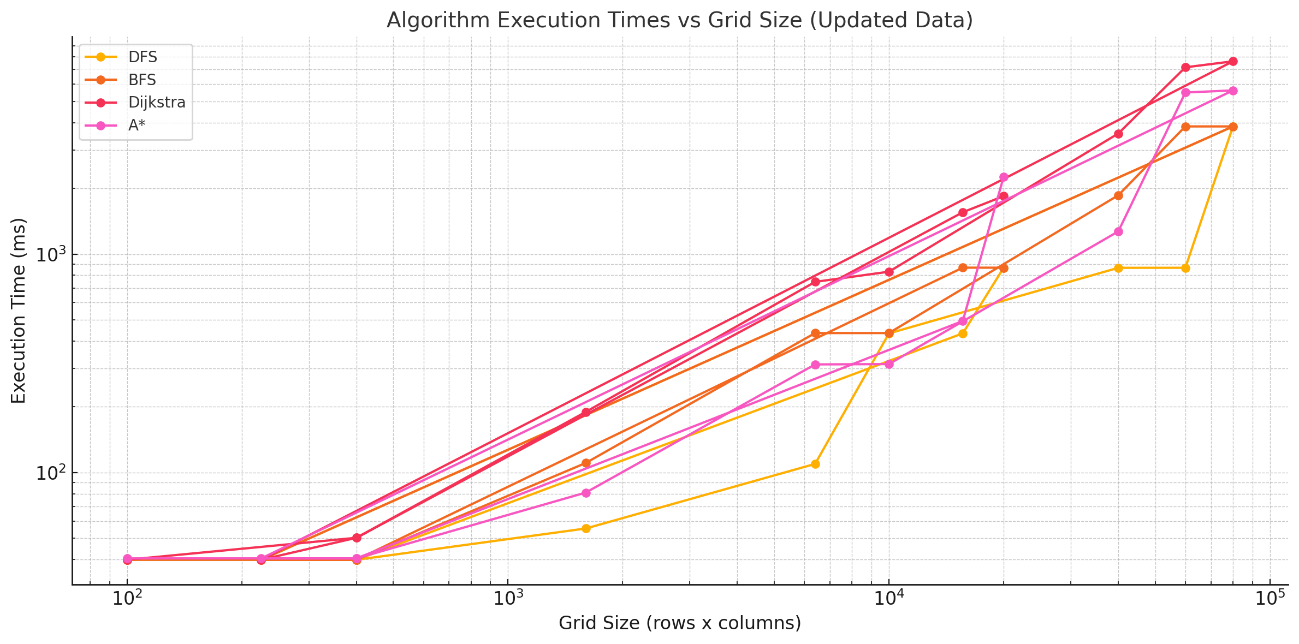
A\* es uno de los mas conocidos algoritmos porque implementa lo que nosotros llamamos Branch and bound. Funciona de la siguiente manera. Creamos un F\_score que es la suma de la distancia hamiltoniana a la salida desde el punto que estamos más la distancia al nodo vecino, Esto entonces crea un algoritmo de búsqueda optimo ya que va recorriendo poco a poco. No como DFS y BFS que normalmente casi recorren todo el grafo. También como BFS encuentra caminos más cortos

Sus utilidades son infinitas ya que es uno de los mas utilizados en muchos juegos y búsqueda del mejor camino como Google Maps.

Finalmente, tenemos nuestros resultados de correr los programas.

Con la data mínima muestra proporcionada ya que no podemos poner un laberinto grande de 400x400. Entonces, concluimos que DFS es el mejor algoritmo mayoritariamente. Pero en 100x100 podemos ver que A\* esta casi a la misma altura que DFS e BFS. En otras palabras, aunque puede que en este muestreo sea DFS el ganador seguramente si el laberinto es suficientemente grande entonces A\* sea más efectivo



En 

En coclusion, DFS es el algoritmo mas optimo para encontrar tu resultado en un laberinto.