INSTITUTO POLITÉCNICO NACIONAL

ESCUELA SUPERIOR DE COMPUTO

“Ejercicio Práctico 2: Búsqueda”

Nombre: Lizandro Hernandez Ramirez

Grupo: 6CV2

Materia: Inteligencia artificial

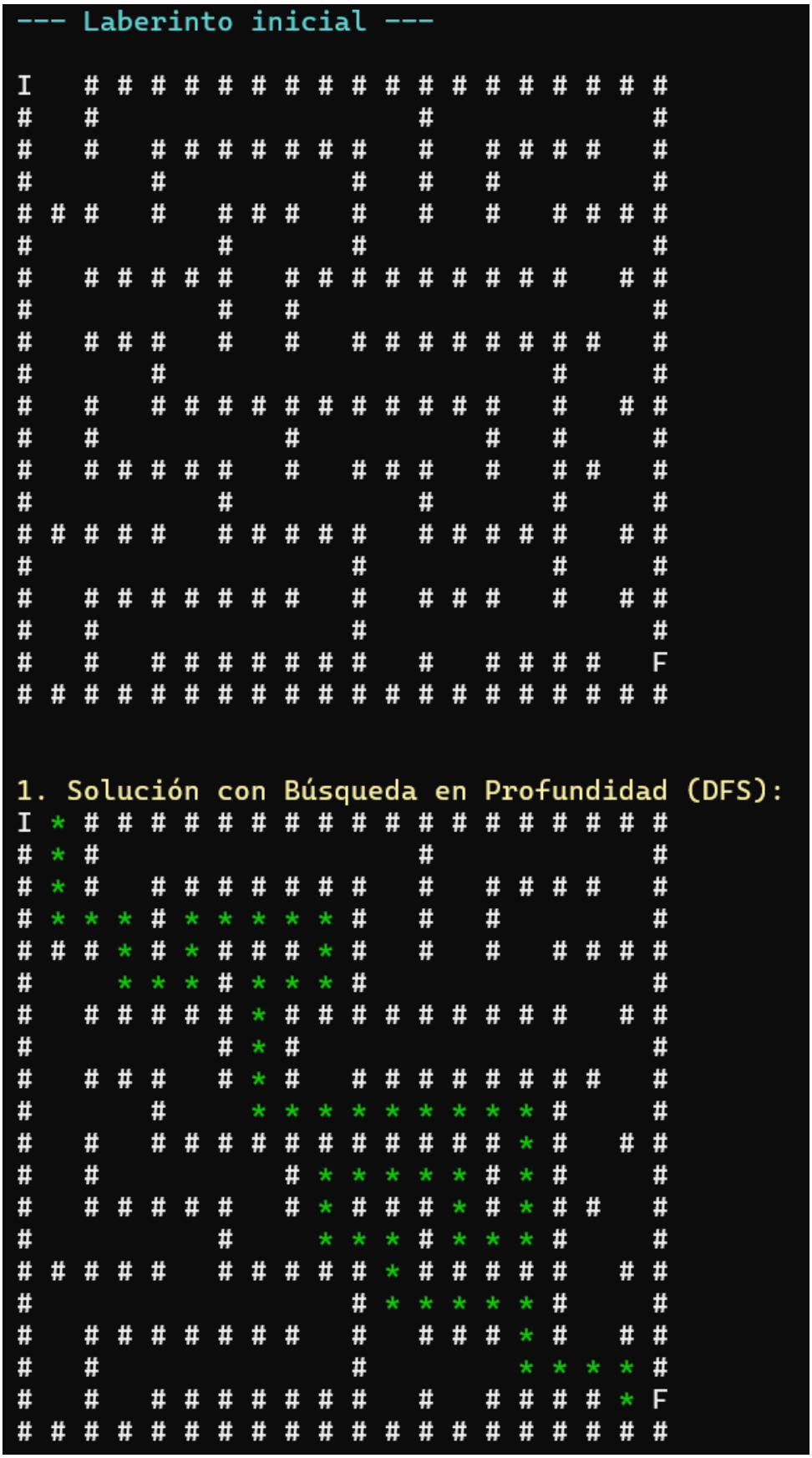
Profesor: Andrés García Floriano

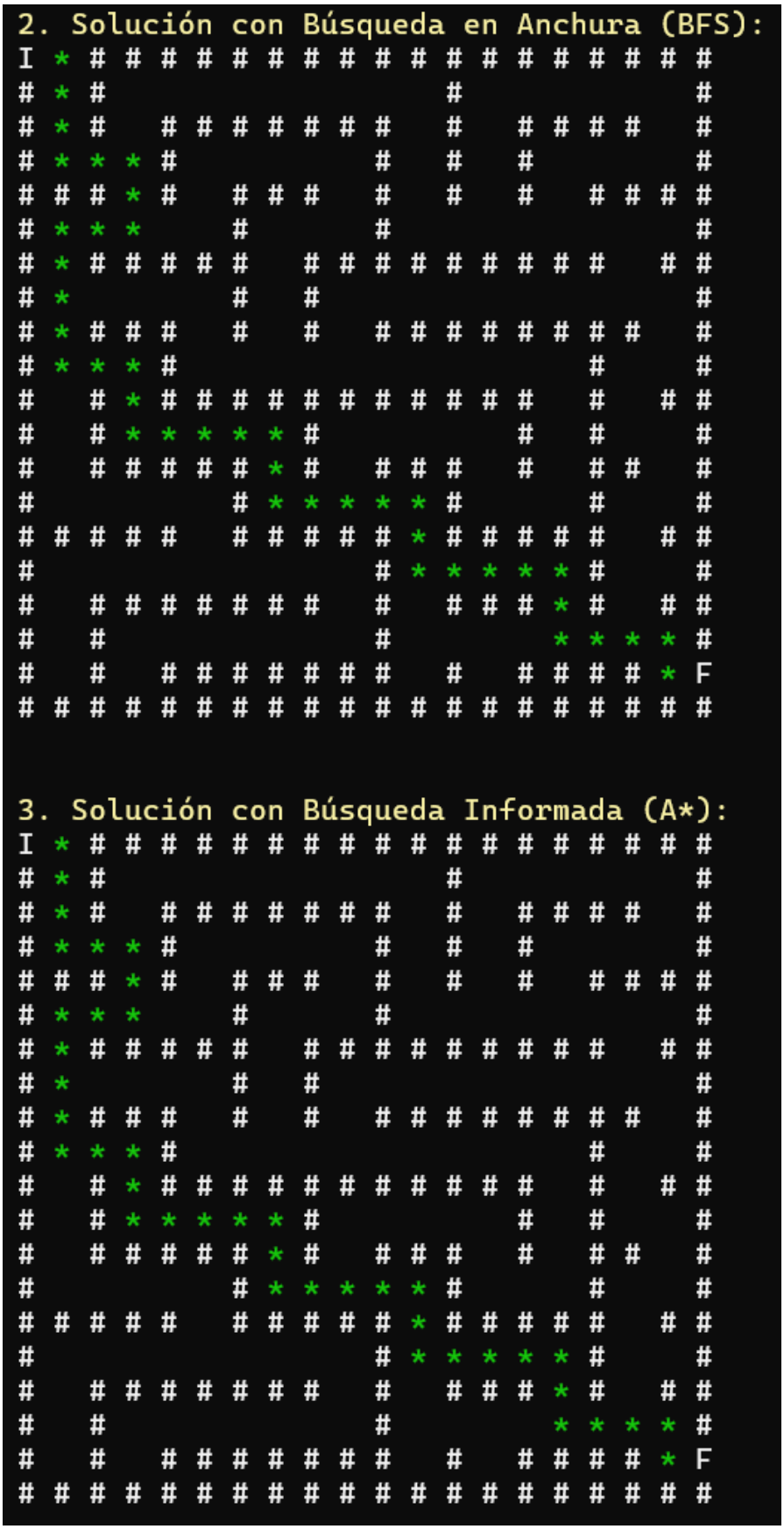
En el campo de la inteligencia artificial, la capacidad de explorar eficientemente un conjunto de posibles caminos para encontrar una solución es muy importante. En esta práctica se implementaron distintas estrategias de búsqueda, divididas en dos categorías principales: las búsquedas a ciegas, representadas por los algoritmos de Búsqueda en Profundidad (DFS) y Búsqueda en Anchura (BFS), y la búsqueda informada, ejemplificada por el algoritmo A*. A través de esto, se busca obtener una comparación de las ventajas, desventajas y la eficiencia de cada método según la naturaleza y complejidad del problema.

Solucionador de Laberintos

Este problema consistió en crear un programa capaz de determinar la mejor ruta válida desde un punto de inicio hasta una salida en un laberinto de 20x20. Para modelar el laberinto, se eligió una representación mediante una matriz bidimensional. En esta matriz, cada celda significó algo diferente dependiendo del carácter que contenía: '#' para los muros, 'I' para el inicio, 'F' para el final, y un espacio en blanco para los caminos transitables. Esta representación ayudó a la visualización posterior de la solución, la cual se marcó con el carácter '*'.

Para la resolución del laberinto se implementaron tres algoritmos distintos. Primero, se abordó la Búsqueda en Profundidad (DFS) mediante recursividad, el cual explora una ruta hasta el final antes de retroceder, utilizando backtracking. Para evitar ciclos infinitos, se utilizó una lista de celdas visitadas. Después, se implementó la Búsqueda en Anchura (BFS), que a diferencia de DFS, explora las posibles soluciones por niveles. Para ello, se utilizó una cola para los caminos a explorar, garantizando así que la primera solución encontrada también sea la que menos pasos tiene que dar para llegar al final. Finalmente, se desarrolló el algoritmo A*, una técnica de búsqueda informada, este método no explora el laberinto a ciegas, sino que se guía por una función heurística, en este caso la **distancia de Manhattan**, ya que nos da una estimación muy viable del costo restante para llegar al final, permitiendo que A* priorice los caminos más cortos a través de una cola de prioridad.

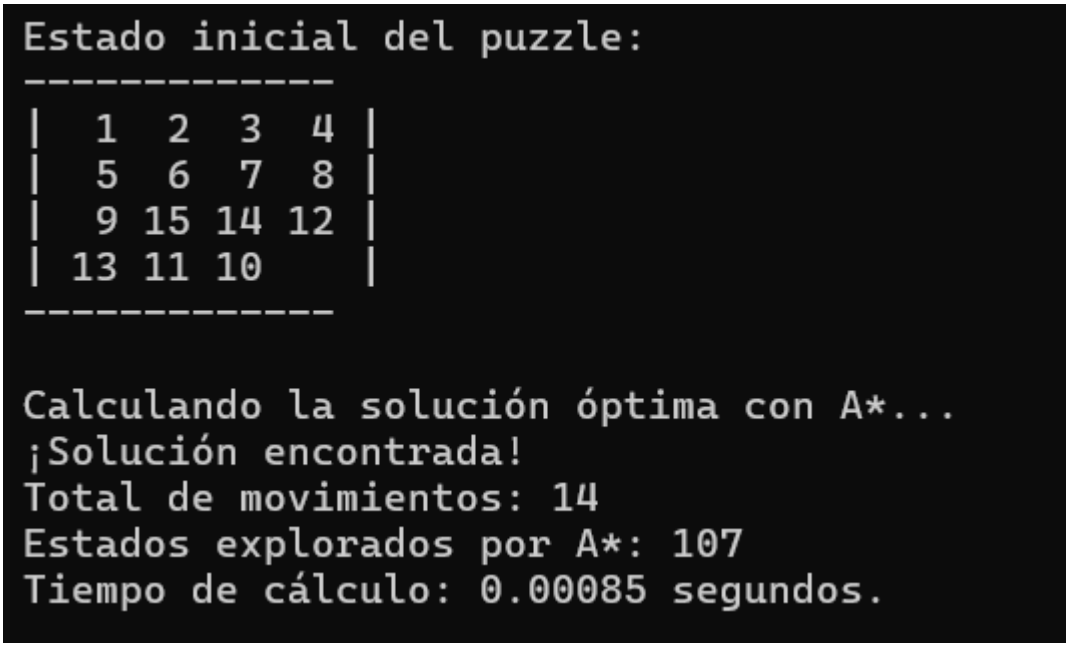

--- Laberinto inicial ---
1. Solución con Búsqueda en Profundidad (DFS):

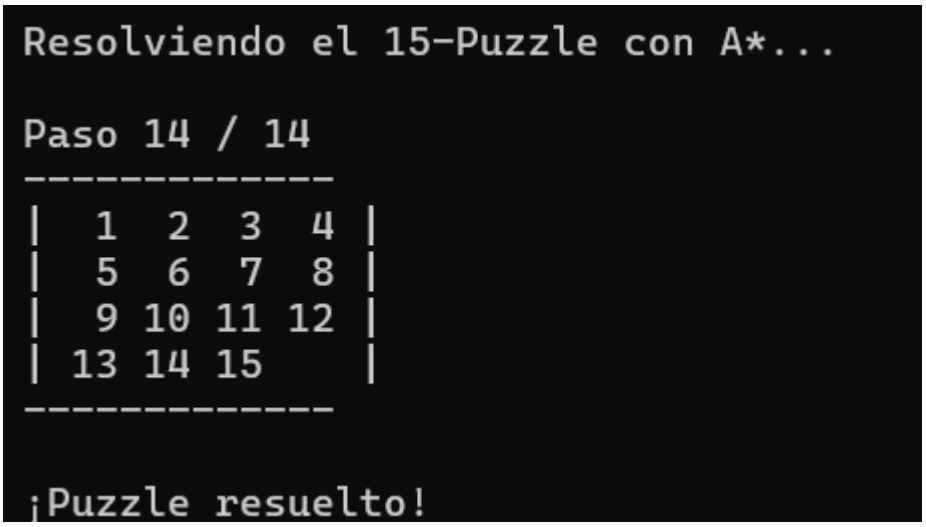

2. Solución con Búsqueda en Anchura (BFS):
3. Solución con Búsqueda Informada (A*):

Solucionador del 15-Puzzle

El segundo problema que se eligió fue el 15-Puzzle, tuvo una mayor complejidad debido a la cantidad de estados que puede tener. La representación del tablero era algo muy importante y se eligió una tupla de tuplas para asegurar la persistencia de cada estado, algo indispensable para un mejor rendimiento, ya que permite que los estados se almacenen en conjuntos y se utilicen como claves en diccionarios para poder rastrear los nodos visitados y sus costos.

Un aspecto importante del 15-Puzzle es que no todas sus configuraciones iniciales tienen solución, por eso se implementó una función para verificar que el estado inicial podía resolverse. Esta función analiza el número de inversiones en el tablero y la posición de la casilla vacía para determinar si existe una solución, evitando que el programa entre en una búsqueda infinita sin encontrar soluciones. Para la búsqueda, se implementó el algoritmo A*, ya que un método a ciegas como BFS sería bastante tardado para encontrar soluciones. Para no utilizar recursos innecesariamente, se evitó almacenar caminos completos en la memoria, en vez de eso, se utilizó un diccionario para registrar el costo mínimo para llegar a cada estado y otro para reconstruir el camino más corto una vez llegando a la solución. La heurística empleada fue la suma de las distancias de Manhattan de cada ficha a su posición final.





Resultados

La ejecución de los programas dio los resultados esperados teóricamente de cada algoritmo. En el problema del laberinto, DFS encontró una solución, pero su camino fue largo y por "fuerza bruta". Por otro lado, BFS y A* encontraron la misma ruta, la cual era la más corta. Este resultado confirmó que, para grafos no ponderados, BFS y A* nos dan una solución óptima en longitud, aunque A* lo hace explorando menos nodos.

En el caso del 15-Puzzle, las diferencias fueron más notorias. El algoritmo A* fue capaz de resolver configuraciones que requerían más movimientos en muy poco tiempo. La heurística de Manhattan demostró ser efectiva para este caso, guiando la búsqueda de manera que solo una pequeña parte de estados necesitó ser explorada. La búsqueda a ciegas, de haberse implementado, habría sido no viable en cuestión de tiempo y memoria. Finalmente, la resolución animada en la terminal para visualizar el puzzle ayuda a observar de manera gráfica los movimientos que el algoritmo determina como la mejor solución.

Como conclusión general, esta práctica nos ayudó confirmar la superioridad de la búsqueda informada sobre la búsqueda a ciegas en problemas de alta complejidad. Mientras que DFS y BFS se pueden usar en espacios de búsqueda pequeños, A* resulta mejor cuando lo que se busca es la eficiencia del programa.