Resolución de un laberinto por el método de profundidad iterativa

Neil Otniel Moreno Rivera
Universidad de Guanajuato, no.morenorivera@ugto.mx

***Resumen*— La resolución de un laberinto por el método de profundidad iterativa consiste en explorar todos los caminos posibles a partir del punto de partida, hasta encontrar el camino al objetivo. Este método utiliza una pila para almacenar los vértices que aún no han sido visitados, y se repite el proceso hasta que se encuentra el objetivo o no quedan más vértices por explorar.**

***Abstract*--- Solving a maze by the iterative depth method consists of exploring all possible paths from the starting point, until finding the path to the objective. This method uses a stack to store vertices that have not yet been visited, and the process is repeated until the target is found or there are no more vertices left to explore.**

I. Introducción

La profundidad iterativa es un algoritmo de búsqueda no informada que explora un espacio de estados en profundidad. Utiliza una pila para almacenar los vértices que aún no han sido visitados, lo que la hace más eficiente en términos de memoria que la búsqueda en profundidad recursiva. Sin embargo, puede ser ineficiente en términos de tiempo, ya que puede explorar caminos que no conducen a la solución.

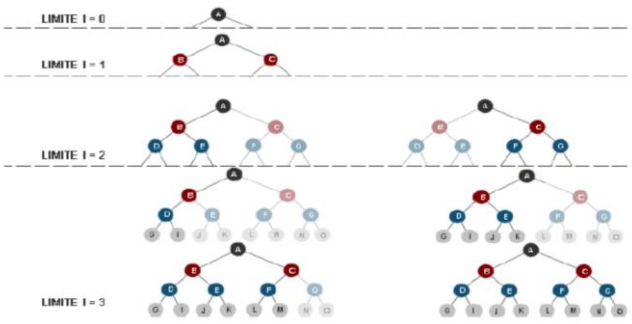

Fig. 1. Gráfico, de cómo funciona el algoritmo.

II. Teoria

Para el abordamiento del problema, primero se revisó el laberinto propuesto para el problema y se realizó un árbol para explicar el camino y las variantes de estados que pudiera tener al encontrar el camino.

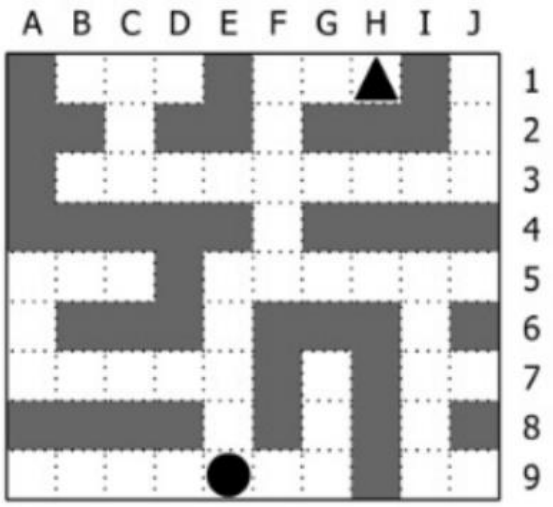

Fig. 2. Laberinto propuesto.

El árbol que se realizó fue el siguiente, propuesto a manera de código.

```
grafo = {
    "A5":["B5","A6"],
    "A6":["A5","A7"],
    "A7":["B7"],
    "A9":["B9"],
    "B1":["C1"],
    "B3":["C3"],
    "B5":["C5","A5"],
    "B7":["A7","C7"],
    "B9":["A9","C9"],
    "C1":["B1","D1"],
    "C2":["C1","C3"],
    "C3":["B3","D3"],
    "C5":["B5"],
    "C7":["B7","D7"],
    "C9":["B9","D9"],
    "D1":["C1"],
    "D3":["C3","E3"],
    "D7":["C7","E7"],
    "D9":["E9","C9"],
    "E3":["D3","F3"],
    "E5":["F5","E6"],
    "E6":["E5","E7"],
    "E7":["E6","E8","D7"],
    "E8":["E9"],
    "E9":["E8","D9","F9"],
    "F1":["G1","F2"],
    "F2":["F1","F3"],
    "F3":["E3","G3","F2","F4"],
    "F4":["F3","F5"],
    "F5":["F4","E5","G5"],
    "F9":["E9","G9"],
    "G1":["H1","F1"],
```

* Neil Otniel Moreno Rivera.

```
    "G3":["F3","H3"],
    "G5":["F5","H5"],
    "G7":["G8"],
    "G8":["G7","G9"],
    "G9":["F9","G8"],
    "H1":["G1"],
    "H3":["G3","I3"],
    "H5":["I5","G5"],
    "I3":["H3","J3"],
    "I5":["H5","J5"],
    "I6":["I5","I7"],
    "I7":["I6","J7","I8"],
    "I8":["I7","I9"],
    "I9":["I8","J9"],
    "J1":["J2"],
    "J2":["J3","J1"],
    "J3":["I3"],
    "J5":["I5"],
    "J7":["I7"],
    "J9":["I9"]
}
```

El algoritmo funciona de la siguiente manera:

1. Se inicia con una pila vacía y se introduce el vértice inicial en la pila.
2. Mientras la pila no esté vacía, se realiza lo siguiente:
    - Se extrae el vértice actual de la pila.
    - Si el vértice actual no ha sido visitado, se marca como visitado y se exploran sus vecinos.
    - Si el vértice actual tiene vecinos no visitados, se insertan en la pila en orden inverso.
    - Si el vértice actual es el vértice objetivo, se termina el algoritmo.

La diferencia clave entre la profundidad iterativa y la búsqueda en profundidad recursiva es que la profundidad iterativa utiliza una pila para almacenar los vértices que aún no han sido visitados. Esto hace que el algoritmo sea más eficiente en términos de memoria, ya que no es necesario almacenar todos los vértices en la pila al mismo tiempo.

La profundidad iterativa es un algoritmo simple y eficiente que puede utilizarse para resolver una variedad de problemas de búsqueda, como la resolución de laberintos, el juego de ajedrez o la planificación de rutas. Sin embargo, puede ser ineficiente en términos de tiempo, ya que puede explorar caminos que no conducen a la solución.

Algunos de los beneficios de la profundidad iterativa son:

- Es un algoritmo simple y fácil de implementar.
- Es eficiente en términos de memoria, ya que no es necesario almacenar todos los vértices en la pila al mismo tiempo.
- Es un algoritmo completo, lo que significa que siempre encontrará una solución si existe.

Algunos de los inconvenientes de la profundidad iterativa son:

- Puede ser ineficiente en términos de tiempo, ya que puede explorar caminos que no conducen a la solución.
- Puede no encontrar la solución más corta, ya que explora los caminos en orden inverso.

Para ampliar la explicación, podría añadir algunos ejemplos de cómo se puede utilizar la profundidad iterativa para resolver problemas concretos. Por ejemplo, podría explicar cómo se puede utilizar para resolver un laberinto o para encontrar una ruta entre dos ciudades.

III. Resultados

Los resultados obtenidos fueron los siguientes; los nodos visitados fueron los siguientes:
['H1', 'G1', 'F1', 'F2', 'F3', 'F4', 'F5', 'G5', 'H5', 'I5', 'J5', 'E5', 'E6', 'E7', 'D7', 'C7', 'B7', 'A7', 'E8', 'E9']

Como resultado se obtuvo la siguiente “gráfica”:

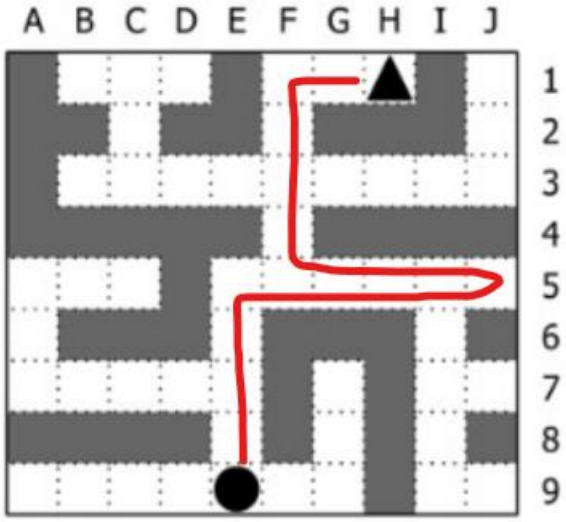

Fig. 3. Laberinto propuesto con el camino tomado, remarcado.

La figura 3, se muestra una solución al problema, pero esta solución no es óptima, puesto que toma un camino sin salida para posteriormente regresar y encontrar el camino correcto.

IV. Concluciones

Para concluir con esto, puedo decir que se encontró una solución, satisfactoria, aunque no optima, puesto que se desvía, para después encontrar la solución correcta, gracias a la recursividad.

Bibliografia

[1] Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein. Introduction to Algorithms. MIT Press, 2009.