UNIVERSIDAD DEL VALLE DE GUATEMALA

INTELIGENCIA ARTIFICIAL -

SECCIÓN - 30

Laberinto Proyecto

Manuel Rodas, 21509

GUATEMALA, 04 de marzo de 2023

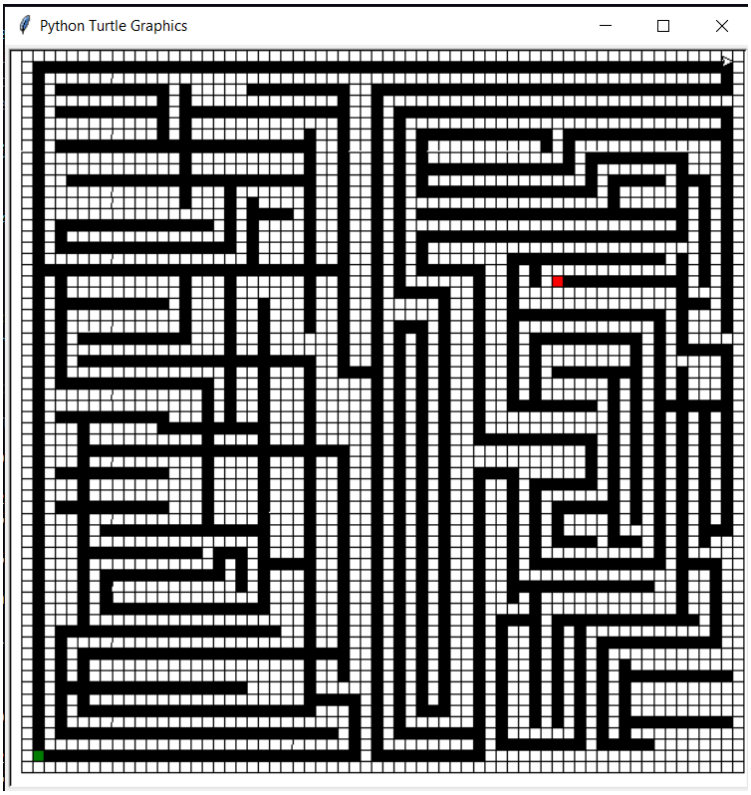
Desafío de Programación:

Principalmente tuve problemas en cómo adecuar el código pasado, a que recibiera un mapa, principalmente tuve que pasar el mapa a un grafo, para así poder resolver el mapa y buscar el camino que llegue del inicio al final.

También, peleé mucho con las coordenadas incorrectas, ya que literalmente las coordenadas de inicio y al final, me las daba mal, y no permitía poder ejecutar los algoritmos de búsqueda.

Mi otro gran problema fue saber si el camino era el más óptimo, ya que si cumplía en resolver el problema pero no sabía si el camino era el más eficiente, entonces perdí mucho tiempo intentando hacer el código lo más eficiente que podía y confiar en el proceso :)

Resultados:



Podemos observar como con ayuda de la libreria turtle, podemos ver el laberinto completo, graficado de color negro donde hay numeros 1, de color blanco donde hay numero 0 y verde y rojo el start y el fin.

Breadth-first search (BFS):

Mejor utilizado cuando se necesita encontrar el camino más corto desde el inicio hasta el objetivo.
Es ideal para grafos donde todos los bordes tienen el mismo costo.
No es eficiente en grafos grandes o cuando hay un gran número de nodos a explorar antes de llegar al objetivo.

Depth-first search (DFS):

Funciona bien en árboles con una profundidad máxima, pero puede quedar atrapado en ciclos infinitos si no se usa correctamente.
No garantiza encontrar la solución óptima, pero es útil en problemas de satisfacción de restricciones.

Depth-delimited search:

Útil cuando se sabe que la solución está dentro de un rango de profundidad específico en el árbol de búsqueda.
Evita problemas de DFS como quedar atrapado en ciclos infinitos o consumir recursos ilimitados.

Greedy best-first search:

Selecciona el siguiente nodo basado en la heurística, ignorando el costo real del camino hasta ese nodo.
A menudo es rápido pero puede no encontrar la solución óptima.

A*:

Combina la eficiencia de BFS con la capacidad de tomar decisiones basadas en una heurística, como la distancia Euclidiana o Manhattan.
Garantiza encontrar la solución óptima si se cumplen ciertas condiciones, como una heurística admisible y un costo de borde no negativo.