Memoria del proyecto Daniel Alé López

Introducción

Este proyecto tiene como objetivo crear un programa interactivo que use el algoritmo A* para encontrar la ruta más corta en un mapa de una ciudad simulada. El sistema tiene que ser capaz de calcular rutas desde un punto de inicio hasta un destino, teniendo en cuenta zonas transitables y bloqueadas. Quiero ver cómo funciona A* y compararlo con otros algoritmos para ver cuál sería mejor para un sistema de navegación autónomo, como los que usan los coches sin conductor.

Fase de Investigación

Al principio, me pasé un buen rato investigando sobre el algoritmo A*. A* es un algoritmo súper útil para encontrar la ruta más corta, porque combina dos cosas: busca en todas las direcciones como el algoritmo de búsqueda en anchura, pero también tiene una estimación para ir directo al objetivo, lo que lo hace mucho más rápido. Lo bueno de A* es que puede usarse para planificar rutas en ciudades grandes y complejas, que es justo lo que quería hacer.

También investigué otros algoritmos como el de Dijkstra, que es como A* pero sin esa estimación (por eso es más lento). Además, vi cómo otros algoritmos como BFS y DFS no eran adecuados para este tipo de proyectos porque no optimizan bien las rutas.

Con todo esto, empecé a pensar en cómo representar la ciudad como un mapa de celdas, donde algunas serían transitables y otras bloqueadas. La idea era hacer que el programa pudiera manejar esas zonas bloqueadas de manera flexible.

Fase de Producción

La parte de la programación fue donde realmente me encontré con varios problemas, pero también aprendí un montón. Para empezar, creé una cuadrícula usando listas en Python, donde cada celda representa un área de la ciudad. Las celdas transitables tenían el valor de 0 y las bloqueadas -1. Esto me permitió representar el mapa de manera sencilla.

Lo más complicado fue hacer que el programa interactuara con el usuario. Quise que el usuario pudiera elegir el punto de inicio y el destino, así como marcar las zonas bloqueadas. Sin embargo, me di cuenta de que no era tan fácil como pensaba. Aunque logré que el usuario pudiera indicar las zonas bloqueadas, no conseguía hacerlo de manera totalmente flexible. Por ejemplo, no se podía decidir exactamente qué áreas bloquear en tiempo real, lo cual fue un punto negativo que tuve que aceptar.

Tras recibir críticas del profesor sobre mejorar la interactividad del programa, decidí añadir una nueva función: ahora el código pregunta si se quiere añadir una calle cortada antes de ejecutar la búsqueda. Esto permitió una mayor personalización del mapa y soluciona parte de los problemas iniciales con las zonas bloqueadas.

Después, implementé el algoritmo A*. La idea era que el automóvil (el símbolo que representa el coche en el mapa) pudiera moverse de la celda de inicio a la de destino siguiendo la ruta más corta. Para esto, utilicé una heurística basada en la distancia euclidiana entre las celdas, lo cual funcionó bastante bien. Sin embargo, uno de los desafíos fue conseguir que el coche pudiera moverse de manera diagonal y no solo en línea recta, lo cual quería para que el movimiento fuera más realista. Después de varios intentos, lo conseguí, pero no fue fácil.

Finalmente, logré que el programa mostrará la ruta en el mapa. Usé la biblioteca matplotlib para representar la cuadrícula y la ruta que había calculado A*. Aunque la ruta en zig-zag era visible, no se veía tan clara como esperaba, especialmente en mapas más grandes o cuando la ruta se alejaba mucho de la línea recta. Ahí fue cuando me di cuenta de que tal vez necesitaba mejorar la visualización.

Cada modificación tuvo su grado de dificultad. Algunas fueron más sencillas, como agregar la opción de marcar calles cortadas, pero otras, como hacer que el coche se moviera en diagonal, requieren muchos ajustes hasta que funcionaron correctamente. A pesar de todo, al final conseguí implementar todos los cambios de manera funcional.

Resultados

A pesar de las dificultades, el algoritmo A* funcionó bastante bien. Fue capaz de encontrar rutas rápidas en la mayoría de los casos. Con las mejoras hechas, ahora el usuario tiene más control sobre las calles bloqueadas, lo cual hace que el programa sea más realista y útil. La visualización sigue teniendo margen de mejora, pero cumple su función.

Comparativa con Otros Algoritmos

Comparado con otros algoritmos, A* fue mucho más eficiente. El algoritmo de Dijkstra, que también sirve para encontrar rutas, es mucho más lento porque no usa esa estimación de la distancia hasta el destino. A* fue más rápido y encontró rutas más cortas en mapas grandes. En cuanto a otros algoritmos como BFS, la diferencia es enorme: BFS no optimiza nada y se dedica a explorar todo el mapa sin ninguna idea de cuál es la ruta más corta, lo que lo hace mucho más lento.

Ejemplo

```
Ingrese el tamaño del mapa (ej. 10 para un mapa 10x10): 10
Ingrese la proporción de áreas bloqueadas (ej. 0.2 para un 20% bloqueado): 0.2
Seleccione el método de heurística (manhattan/euclidean): euclidean

Mapa generado:
[[0 0 0 0 1 0 0 0 0 0]
 [0 1 1 0 1 0 0 0 0 0]
 [0 0 1 0 0 0 1 0 0 0]
 [0 0 0 0 0 0 0 0 1 0]
 [1 0 0 0 1 1 0 0 0 1]
 [0 1 0 0 0 0 0 0 1 0]
 [0 0 1 1 0 0 1 0 1 0]
 [0 0 1 0 0 0 0 0 0 0]
 [0 0 1 0 0 0 0 0 0 0]
 [0 0 0 1 0 0 0 0 1 1]]
¿Desea añadir una calle cortada (s/n)? n
Ingrese las coordenadas de inicio (fila columna, ej. 0 0): 0 0
Ingrese las coordenadas de destino (fila columna, ej. 9 9): 8 9

Ruta encontrada: [(0, 0), (1, 0), (2, 1), (3, 2), (4, 3), (5, 4), (6, 5), (7, 6), (7, 7), (7, 8), (8, 9)]

Distancia Total Recorrida: 12.90
```