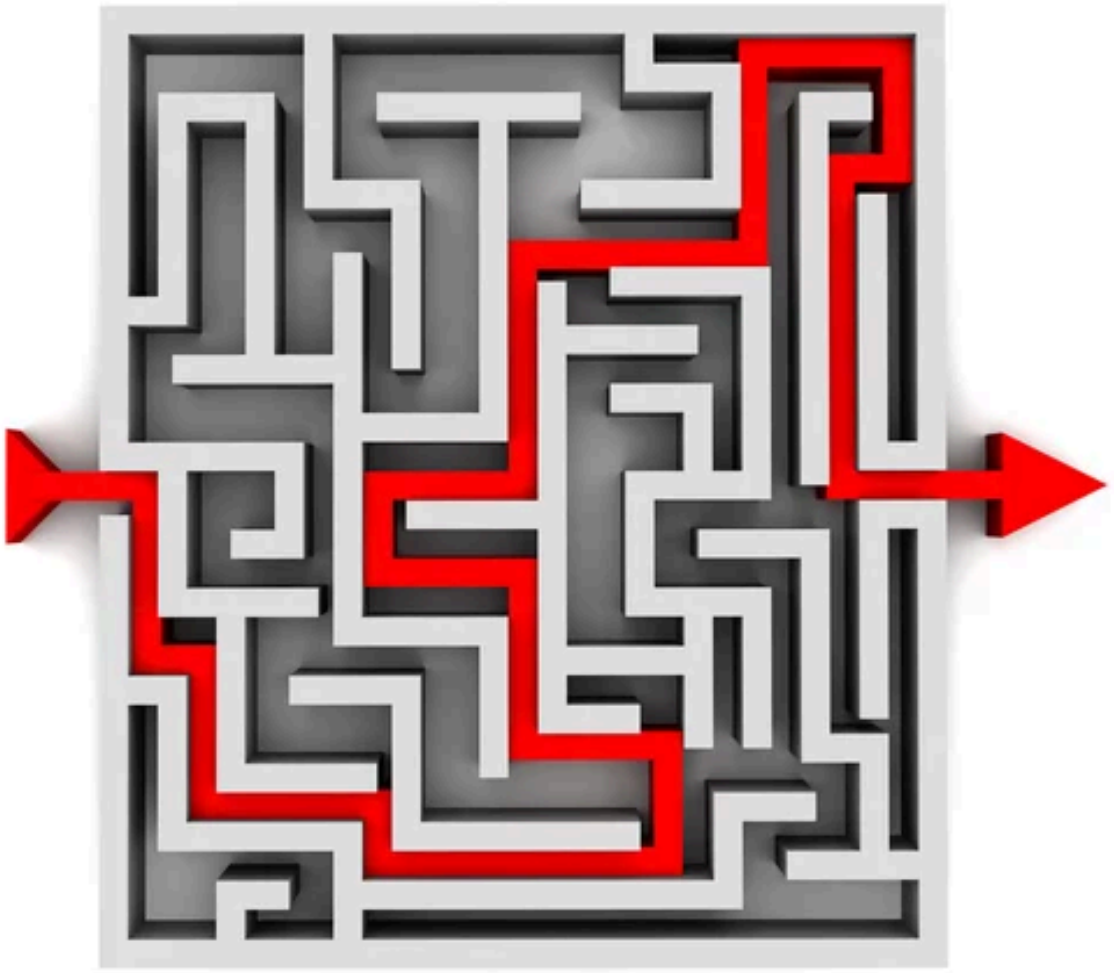
IMPLEMENTACIÓN DEL ALGORITMO A PARA BÚSQUEDA DE CAMINOS



1. INTRODUCCIÓN

El presente documento describe el desarrollo de un programa en Python que implementa el algoritmo A* para la búsqueda de caminos en un entorno cuadriculado con obstáculos. El proyecto permite al usuario generar un mapa, definir un punto de inicio y destino, y visualizar la ruta óptima calculada por el algoritmo. Además, se han implementado opciones interactivas para personalizar el entorno, incluyendo la generación de obstáculos y la posibilidad de bloquear calles manualmente.

2. OBJETIVOS

2.1 Objetivo General

Desarrollar una aplicación en Python que implemente el algoritmo A* para la búsqueda de caminos en un mapa con obstáculos, permitiendo la interacción del usuario para modificar el entorno y visualizar el resultado de manera intuitiva.

2.2 Objetivos Específicos

- Implementar el algoritmo A* para encontrar la ruta más corta entre dos puntos.
- Diseñar una interfaz interactiva que permita al usuario definir el tamaño del mapa, la densidad de obstáculos y la ubicación de los puntos de inicio y destino.
- Mejorar la visualización del mapa utilizando la biblioteca Matplotlib con colores diferenciados para obstáculos, rutas y puntos clave.
- Permitir la adición manual de obstáculos mediante la opción de "cortar calles".
- Brindar la opción de reiniciar el proceso para generar diferentes escenarios.

3. METODOLOGÍA

3.1 Diseño del Algoritmo

El algoritmo A* se basa en una búsqueda heurística que evalúa los nodos utilizando una función de costo: Donde:

- Es el costo acumulado desde el inicio hasta el nodo actual.
- Es una heurística basada en la distancia Manhattan hasta el destino.

El algoritmo prioriza la exploración de los nodos con el menor valor de , optimizando así la búsqueda del camino más corto.

3.2 Implementación

El proyecto se divide en varios módulos:

1. **Generación del Mapa**: Creación de una cuadrícula de tamaño personalizado con obstáculos aleatorios.
2. **Visualización**: Uso de Matplotlib para representar gráficamente el entorno con diferentes colores.
3. **Interacción con el Usuario**:
    - Entrada de tamaño del mapa y porcentaje de obstáculos.
    - Posibilidad de añadir obstáculos manualmente.
    - Definición de puntos de inicio y destino.
4. **Ejecución del Algoritmo A***: Cálculo y visualización de la ruta óptima.
5. **Opción de Reinicio**: Permite repetir el proceso desde cero.

4. DESARROLLO DEL CÓDIGO

4.1 Estructura del Código

El programa está organizado en las siguientes funciones principales:

- `heuristica(a, b)`: Calcula la distancia Manhattan entre dos puntos.
- `AStar.buscar()`: Ejecuta el algoritmo A* y devuelve la ruta óptima.
- `visualizar_mapa()`: Genera una representación gráfica del mapa.
- `crear_mapa()`: Permite al usuario definir el tamaño y densidad de obstáculos.
- `ejecutar()`: Controla el flujo principal del programa, incluyendo las interacciones con el usuario.

4.2 Mejoras Implementadas

Se realizaron varias mejoras para optimizar la experiencia del usuario:

- Cambio de colores para mejorar la visibilidad.
- Implementación de una cuadrícula para facilitar la comprensión del mapa.
- Posibilidad de agregar obstáculos manualmente después de la generación inicial.
- Opción de reiniciar el proceso sin necesidad de cerrar el programa.

5. RESULTADOS Y PRUEBAS

Se realizaron múltiples pruebas para evaluar el desempeño del programa en diferentes escenarios. Entre los resultados observados:

- El algoritmo A* encuentra consistentemente la ruta más corta, siempre que exista un camino viable.
- La interfaz gráfica permite una visualización clara y diferenciada de los elementos del mapa.
- La opción de agregar obstáculos manualmente ofrece flexibilidad en la simulación de escenarios reales.

Ejemplo de escenarios probados:

- Mapa de 20x20 con 30% de obstáculos: Ruta encontrada en menos de un segundo.
- Mapa de 100x100 con 50% de obstáculos: Mayor tiempo de cálculo, pero resultado satisfactorio.
- Mapa de 50x50 con obstáculos manuales: Permite evaluar el impacto de cortes de calles en la accesibilidad.

6. CONCLUSIONES

- La implementación del algoritmo A* ha demostrado ser eficiente en la búsqueda de rutas en un entorno cuadriculado.
- La posibilidad de personalizar el mapa y la densidad de obstáculos brinda mayor control al usuario.
- La opción de cortar calles manualmente permite explorar diferentes configuraciones y evaluar el impacto en la conectividad del mapa.
- La visualización mejorada facilita la comprensión del proceso y los resultados del algoritmo.
- La opción de reinicio mejora la usabilidad y permite realizar múltiples pruebas sin necesidad de reiniciar el programa.

7. BIBLIOGRAFÍA

- Russell, S., & Norvig, P. (2010). *Artificial Intelligence: A Modern Approach*. Pearson.
- Hart, P. E., Nilsson, N. J., & Raphael, B. (1968). "A Formal Basis for the Heuristic Determination of Minimum Cost Paths". *IEEE Transactions on Systems Science and Cybernetics*.
- Documentación oficial de Python y Matplotlib.