Universidad Nacional
Sede Interuniversitaria de Alajuela
Escuela de Informática
EIF 4200 - Inteligencia Artificial

PRIMER PROYECTO PROGRAMADO

CURSO:	Inteligencia Artificial
PROFESOR:	Darío Ríos Navarro
GRUPOS:	**Máximo dos personas**.
FECHA DE ENTREGA:	**Domingo 10 de setiembre de 2023**

Para este primer proyecto van a desarrollar un ejercicio donde puedan comparar los diferentes algoritmos de búsqueda utilizados en inteligencia artificial, creando un agente que resuelva los laberintos. Teniendo en cuenta esto como objetivo, deben crear una aplicación que resuelva laberintos utilizando los siguientes algoritmos para resolver la ruta más corta.

- Dijkstra
- Bellman – Ford
- Depth First Search (DFS)
- Breadth-First Search (BFS)
- A* Search

El desarrollo se dividirá en varias etapas que se describen a continuación:

1. **Leer un archivo CSV con el laberinto**

Dado un laberinto de tamaño n x m, en un archivo CSV, cargue la información del laberinto de manera que pueda ser agregado en la aplicación como una matriz. En estos ejercicios, se recomienda como punto de inicio, estandarizar los valores para usarse en el laberinto mediante números. A continuación, se indica una posible opción:

0 – Espacios por los cuales puede moverse el agente
1 – Paredes o obstáculos donde no se permite al agente caminar por ahí
2 – El punto de inicio
3 – El punto final o meta
4 – El recorrido que hace cada camino, si lo quieren mostrar
5 – La ruta mínima obtenida por el algoritmo

Inicie con laberintos que tengan una meta y un punto de inicio. Agregue obstáculos hasta llegar a un punto donde el laberinto no tenga solución.

2. **Implementar una interfaz para escoger el algoritmo que se desea mostrar**

   El objetivo de esto es hacer un menú que permita escoger cual algoritmo usar. El menú puede ser creado en consola, o utilizar la interfaz gráfica. En esa interfaz deben aparecer un título del programa y los 5 algoritmos como opciones del menú del cualquier quiere mostrarse como mínimo.

3. **Implemente cada algoritmo**

   Todas las implementaciones tienes ciertos patrones en su implementación. Como mínimo usted debería tener una clase por cada uno de ellos, y un método que devuelva el path de resultado. Ese path de resultado, usted puede irlo reflejando en una matriz con la solución del laberinto. Como estaremos usando Python para el desarrollo de este proyecto, lo ideal, sería devolver una lista de tuplas, las cuales usted debe comparar con las posiciones de su matriz. Cada uno de los métodos en sus parámetros, debería tener tres de ellos de suma importancia, que son la matriz, la posición de inicio del laberinto y la meta.

4. **Dibujar la matriz resultante**

   Existen muchas maneras de mostrar el resultado. Puede imprimir la matriz varias veces mostrando cada movimiento, o usar la interfaz gráfica para pintarla. Nuevamente esto es a elección de cada equipo.

   Al final, debería mostrarse la matriz con la ruta mínima o más corta que llevo a ese resultado. Junto con las posibles rutas alternas.

5. **Tome estadísticas del tiempo y la memoria utilizada**

   La idea es presentar un informe de los resultados y el desempeño de cada algoritmo. Identifique cuando tarda cada algoritmo y cuanta memoria utiliza.

   La idea es sacar conclusiones como:

   - ¿Qué algoritmo es más veloz en obtener la mejor ruta?
   - ¿Qué algoritmo consume más memoria?

   La idea es encontrar la mejor relación tiempo – memoria para uno de los algoritmos. Como hemos visto, ninguno es perfecto y cada uno tiene sus desventajas, y la idea es analizar como se comportan. Considere además que con un laberinto pequeño y con pocos obstáculos, pero se probará con un laberinto aún más complejo. El algoritmo es capaz de resolverlo? Considere tomar laberintos de bastantes columnas y filas para su prueba. Ojala de 200 x 200 o más.

   Pueden utilizar librerías como psutil, tracemalloc, time y memory_profiler. Guarde la información de los resultados de memoria en un txt con el mismo nombre del laberinto pero con el algoritmo que se uso. Por ejemplo, si corrió el ejercicio con un a*, el archivo debería llamarse a_star_laberinto_1.txt.

6. **Guarde en otro archivo CSV, la matriz resultante.**

   Este ejercicio buscar incluir la mejor ruta óptima obtenida de la maximización realizada por el algoritmo. Para mantener la ruta, la idea es escribir la matriz en un archivo CSV nuevo con el resultado y las rutas alternas (Refiérase a los

números 4 y 5 que se especifican en el punto 1). No sobre escriba la matriz anterior o original. Siga el mismo patrón que para el archivo txt: El nombre del CSV debería ser el algoritmo_nombrelaberinto. Ejemplo: a_starlaberinto1.csv

Procure hacer uso de la programación orientada a objetos para facilitar la comprensión del código.

Puede utilizar la interfaz gráfica si así lo desea, aunque no es obligatorio. Puede mostrar el camino de la mejor ruta, no es necesario imprimir todas las posibilidades.

Reglas o restricciones del proyecto

- No pueden usar ChatGPT, ni copiar código de esta herramienta. Se analizará el código para identificar si fue plagiado. Debe crear su propia solución
- **Esta prohibido el uso de librerías como Pymaze**, ya que esto facilitaría demasiado el desarrollo del programa.

De descubrirse que hubo plagio de código, dada las condiciones descritas, se anulará completamente el proyecto de los estudiantes, obteniendo una nota de cero.

Evaluación del proyecto:

Descripción		
1. Leer un archivo CSV con el laberinto		**15pts**
2. Implementar una interfaz para escoger el algoritmo que se desea mostrar		**10 pts.**
3. Implemente cada algoritmo		**40 pts.**
	Dijkstra	5 pts.
	Bellman-Ford	5 pts.
	DFS	10 pts.
	BFS	10 pts.
	A*	10 pts.
4. **Dibujar la matriz resultante**		**15 pts.**
5. Tome estadísticas del tiempo y la memoria utilizada		**10 pts.**
6. Guarde en otro archivo CSV, la matriz resultante.		**10 pts.**
Total		**100 pts**

Valor porcentual total: 15%