

Inteligencia Artificial 2025 - Grupo 5

Trabajo práctico N°2



Integrantes:

- Medina, Gabriel 14086
- Vasquez, Nicolas 14097
- Albanez, Luciano 13185
- Fattiboni, Álvaro 13201

Inteligencia Artificial 2025 - Grupo 5

1. ¿Qué diferencia hay entre una estrategia de búsqueda Informada y una estrategia de búsqueda No Informada?

Las estrategias de búsqueda “No Informadas” (también llamadas “ciegas”) no disponen de información adicional sobre el problema más allá de la definición de los estados inicial y final, y las acciones posibles. Por lo tanto, exploran el espacio de estados de manera sistemática pero sin orientación, lo que puede resultar en un mayor consumo de tiempo y recursos.

En cambio, las estrategias de búsqueda “Informadas” utilizan información adicional (generalmente a través de una función heurística) para guiar el proceso de exploración hacia los estados más prometedores. Esto permite reducir el número de nodos explorados y, en consecuencia, encontrar soluciones de manera más eficiente.

2. ¿Qué es una heurística y para qué sirve?

Una “heurística” es una función o regla que proporciona una estimación del costo o distancia desde un estado dado hasta el objetivo. No garantiza exactitud, pero ofrece una guía para priorizar qué caminos explorar primero.

Su principal utilidad es mejorar la eficiencia de los algoritmos de búsqueda, ya que orienta el proceso hacia las soluciones más probables, reduciendo así el tiempo y los recursos necesarios para resolver el problema.

3. ¿Es posible que un algoritmo de búsqueda no tenga solución?

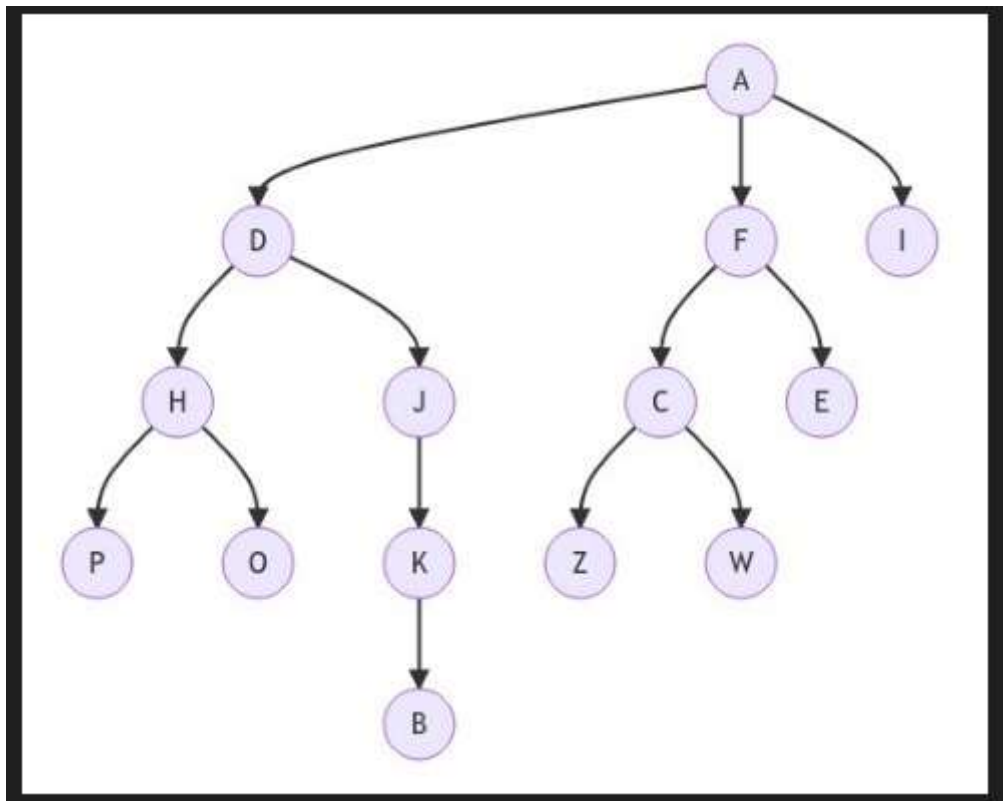
Sí, es posible. Un algoritmo de búsqueda puede no encontrar solución en las siguientes situaciones:

- Cuando el problema planteado no tiene un estado objetivo alcanzable desde el estado inicial.
- Cuando el espacio de estados es infinito o demasiado grande y el algoritmo no puede explorar todas las posibilidades.
- Si el algoritmo está mal diseñado o presenta limitaciones en memoria, tiempo o profundidad, puede no llegar a encontrar la solución incluso si esta existe.

En conclusión, la existencia de una solución depende tanto de la naturaleza del problema como de las capacidades y restricciones del algoritmo empleado.

Inteligencia Artificial 2025 - Grupo 5

4.



4.1) Primero en Amplitud

- La exploración se hace nivel por nivel.
- Se empieza en la raíz y se visitan todos los nodos hijos antes de bajar a la siguiente profundidad.
- Se usa una cola (FIFO).

Tiene la ventaja de que encuentra siempre la solución más corta (en cantidad de pasos) con la desventaja de que necesita mucha memoria si el árbol es muy grande

Inteligencia Artificial 2025 - Grupo 5

Estado Actual								
A	D	F	I					
D	F	I	H	J				
F	I	H	J	C	E			
I	H	J	C	E				
H	J	C	E	P	O			
J	C	E	P	O	K			
C	E	P	O	K	Z	W		
E	P	O	K	Z	W			
P	O	K	Z	W				
O	K	Z	W					
K	Z	W	B					
Z	W	B						
W	B							
B								

4.2) Primero en Profundidad

- La exploración se hace siguiendo un camino hasta el final antes de retroceder.
- Se usa una pila (LIFO) (o recursividad).

Tiene la ventaja de usar poca memoria pero la desventaja de poder quedarse atascado en ramas profundas sin encontrar la solución.

Estado Actual								
A	D	F	I					
D	H	J	F	I				
H	P	O	J	F	I			

Inteligencia Artificial 2025 - Grupo 5

P	O	J	F	I				
O	J	F	I					
J	K	F	I					
K	B	F	I					
B	F	I						
F	C	E	I					
C	Z	W	E	I				
Z	W	E	I					
W	E	I						
E	I							
I								

4.3) Primero en Profundidad con Profundidad Limitada Iterativa

- Combinación de los 2 anteriores
- Se hace DFS (Depth-First Search) con un límite de profundidad 1. Luego se aumenta el límite a 2, y se repite DFS, pero esta vez explorando hasta el nivel 2. Después límite 3, y así sucesivamente

Tiene la ventaja de que

- Como BFS (Breadth-First Search), garantiza encontrar la solución más cercana (mínima profundidad).
- Como DFS, solo necesita memoria proporcional a la profundidad, no a todo el árbol.

La desventaja es que repetiría mucho trabajo

Límite 1

Estado Actual								
A	D	F	I					
D	F	I						
F	I							

Inteligencia Artificial 2025 - Grupo 5

I								
---	--	--	--	--	--	--	--	--

Límite 2

Estado Actual								
A	D	F	I					
D	H	J	F	I				
H	J	F	I					
J	F	I						
F	C	E	I					
C	E	I						
E	I							
I								

Límite 3

Estado Actual								
A	D	F	I					
D	H	J	F	I				
H	P	O	J	F	I			
P	O	J	F	I				
O	J	F	I					
J	K	F	I					
K	F	I						
F	C	E	I					
C	Z	W	E	I				
Z	W	E	I					
W	E	I						
E	I							
I								

Inteligencia Artificial 2025 - Grupo 5

Límite 4

Estado Actual								
A	D	F	I					
D	H	J	F	I				
H	P	O	J	F	I			
P	O	J	F	I				
O	J	F	I					
J	K	F	I					
K	B	F	I					
B	F	I						
F	C	E	I					
C	Z	W	E	I				
Z	W	E	I					
W	E	I						
E	I							
I								

Límite 1, 2, 3 y 4

Estado Actual								
A	D	F	I					
D	F	I						
F	I							
I								
A	D	F	I					
D	H	J	F	I				

Inteligencia Artificial 2025 - Grupo 5

H	J	F	I					
J	F	I						
F	C	E	I					
C	E	I						
E	I							
I								
A	D	F	I					
D	H	J	F	I				
H	P	O	J	F	I			
P	O	J	F	I				
O	J	F	I					
J	K	F	I					
K	F	I						
F	C	E	I					
C	Z	W	E	I				
Z	W	E	I					
W	E	I						
E	I							
I								
A	D	F	I					
D	H	J	F	I				
H	P	O	J	F	I			
P	O	J	F	I				
O	J	F	I					
J	K	F	I					
K	B	F	I					
B	F	I						
F	C	E	I					
C	Z	W	E	I				

Inteligencia Artificial 2025 - Grupo 5

Z	W	E	I					
W	E	I						
E	I							
I								

Ej 5)

Programas

- **DFS (Depth-First Search)**
- **Greedy Best-First Search**
- **A*** (con todas las casillas de coste unitario, es decir, busca el camino con menor número de pasos)

La interfaz gráfica permite ejecutar y animar cada búsqueda paso a paso o de manera automática al seleccionar el algoritmo de búsqueda a observar.

Estructura

a) Representación del tablero

- Cada casilla (A, B, C, ..., N, W, F) está definida con coordenadas (x , y) que controlan su posición en el lienzo.
- Las conexiones entre casillas se definen en un diccionario **adj** (adyacencias).
- Se incluye una **barrera en W** dibujada en la parte inferior para representar la restricción visual del tablero original.

b) Algoritmos de búsqueda

Cada algoritmo está implementado como **generador de eventos** (**yield**) que devuelve pasos intermedios. Esto permite animar su ejecución en la GUI.

Inteligencia Artificial 2025 - Grupo 5

- **DFS**

- Usa una **pila (stack)**.
- Expande nodos en orden alfabético al insertar vecinos.
- Muestra cómo se profundiza primero antes de retroceder.

- **Greedy Best-First**

- Usa una **cola de prioridad (heap)**.
- Prioriza nodos con menor heurística **$h(n)$** = distancia Manhattan hasta el objetivo **F**.
- Ignora costes acumulados.

- **A***

- También usa un heap, pero calcula **$f(n) = g(n) + h(n)$**
- **$g(n)$** : número de pasos desde el inicio (cada transición vale 1).
- **$h(n)$** : Manhattan hasta **F**.
- Toma en cuenta el coste de pasar por las casillas

c) Interfaz gráfica (Tkinter)

- El tablero se dibuja en un **Canvas**.
- Cada nodo es un círculo con su etiqueta.
- Colores dinámicos:
 - **Azul**: frontera (nodos en la estructura de control).
 - **Verde**: nodo en expansión.
 - **Gris**: visitados.
 - **Amarillo**: camino final.
- Controles disponibles:
 - Selección de algoritmo (DFS, Greedy, A*).
 - Botones: **Run**, **Step**, **Reset**.

Inteligencia Artificial 2025 - Grupo 5

- Selector de velocidad de animación.

Funcionamiento de A*

En esta versión:

- A* quiere encontrar el camino mas rapido pero tambien mas barato.
- Lo que significa que nunca va a pasar por w por su altísimo precio.

Ventajas del programa

- Permite **comparar diferentes algoritmos** sobre el mismo tablero.
- Ofrece una **visualización interactiva** que facilita comprender cómo exploran el grafo.
- Separación clara entre la lógica del grafo, los algoritmos y la visualización.

Posibles mejoras

- Guardar la animación o el tablero final como imagen.
- Mostrar estadísticas completas (número de nodos expandidos, coste del camino, longitud en pasos).
- Soporte para tableros definidos por el usuario.

Inteligencia Artificial 2025 - Grupo 5

DFS (Depth-First Search)

- **Estrategia:** Explora siempre lo más profundo posible antes de retroceder.
- **Estructura de control:** Usa una **pila (stack)**.
- **Orden de expansión:** Inserta los vecinos en orden alfabético, por lo que la exploración sigue ese criterio.
- **Ventajas:**
 - Consume poca memoria, ya que solo almacena la ruta actual.
 - Puede encontrar rápidamente una solución, aunque no necesariamente la mejor.
- **Desventajas:**
 - No garantiza encontrar el camino más corto.
 - Puede explorar ramas muy largas o infinitas si existieran ciclos.

En este tablero: DFS puede dar una solución válida a **F**, pero no necesariamente será la más corta.

Método Greedy Best-First Search

1. Estrategia

- En cada paso selecciona el nodo con el **menor valor de heurística $h(n)$** .
- El coste de **W** es **30**, pero eso no nos interesa en este método ya que lo que queremos es simplemente llegar lo más rápido posible al resultado.
- Expande ese nodo y agrega sus sucesores a la cola.
- Se repite hasta alcanzar el objetivo.

Inteligencia Artificial 2025 - Grupo 5

2. Función heurística

La heurística utilizada es la **distancia Manhattan** entre el nodo actual y el nodo objetivo **F**.

$$h(n) = |x_n - x_f| + |y_n - y_f| \quad h(n) = |x_n - x_f| + |y_n - y_f|$$

3. Consideración de costes

A diferencia de A^* , Greedy no utiliza el coste acumulado del camino ($g(n)$). Esto significa que si una casilla tiene un coste mayor (por ejemplo $W = 30$), al algoritmo no le va a importar este dato y puede pasar por un camino costoso.

4. Ventajas

- Explora rápidamente hacia el objetivo.
- Consume menos recursos que A^* .
- Puede evitar caminos costosos si la heurística lo refleja correctamente.

5. Desventajas

- No garantiza encontrar la solución óptima.
- Puede quedar atrapado en caminos subóptimos si la heurística está mal definida.

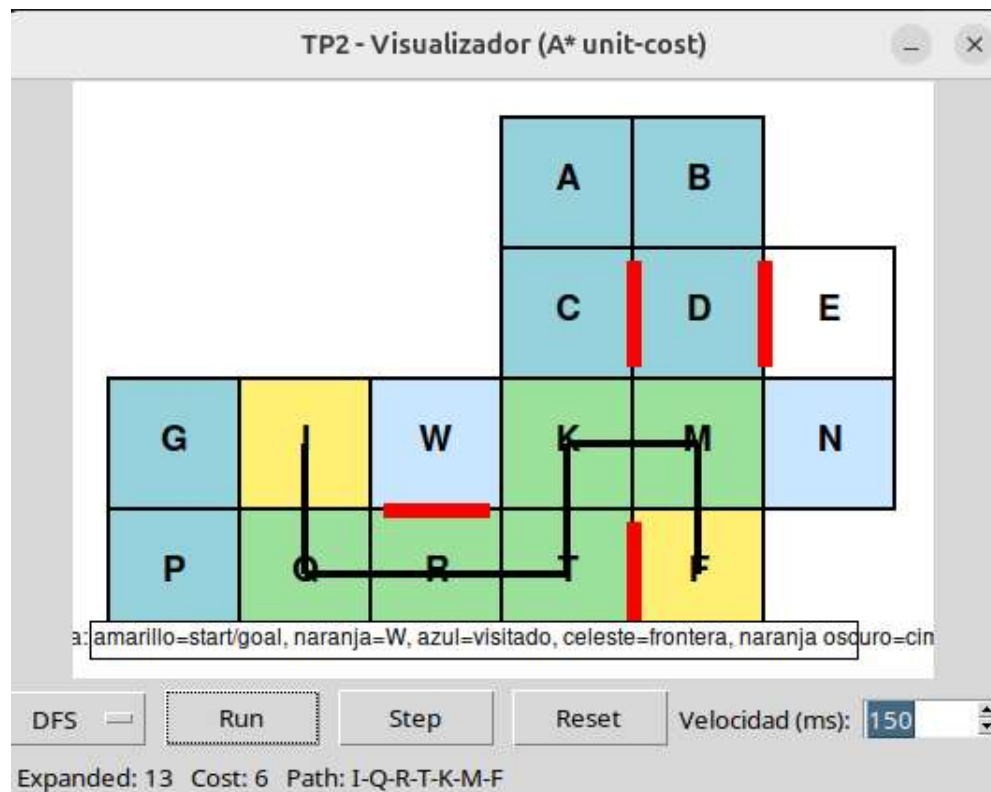
6. En este tablero

- A diferencia de A^* , Greedy no “se preocupa” por el coste de cada casilla en su elección inmediata, aunque no evalúe el coste total del camino.

Inteligencia Artificial 2025 - Grupo 5

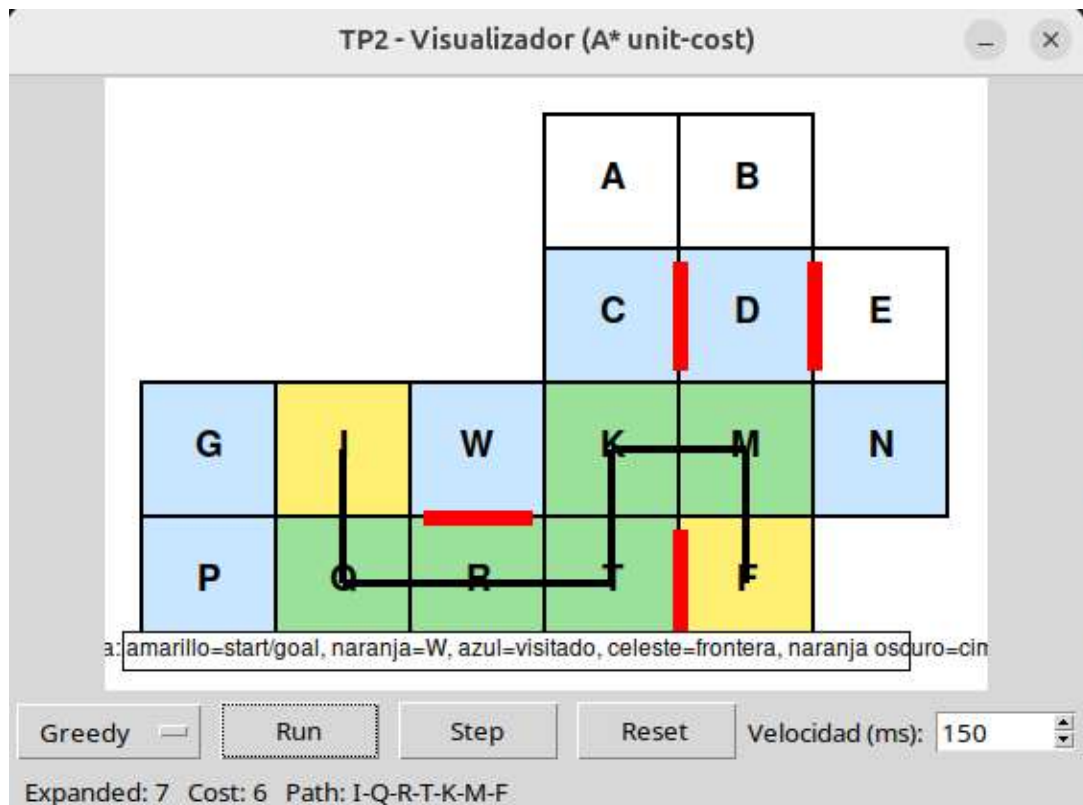
Resultados

DFS:

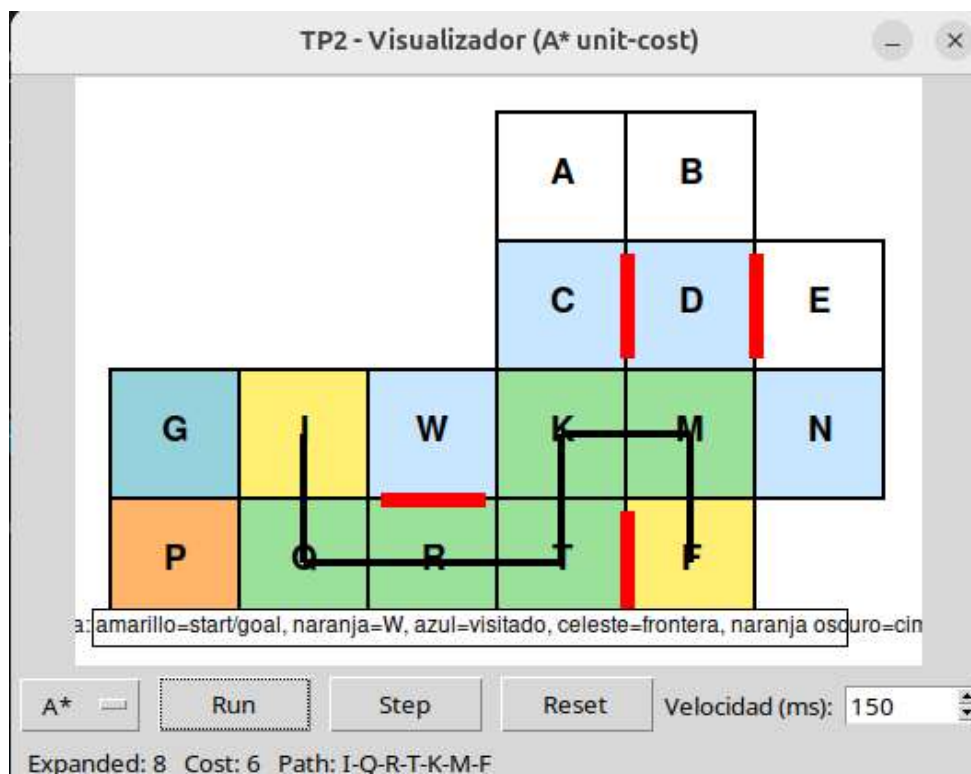


Inteligencia Artificial 2025 - Grupo 5

Greedy:



A*:



Inteligencia Artificial 2025 - Grupo 5

Observaciones:

Si bien tanto A* como DFS lograron el camino esperado no fue así con Greedy. Esto se debe a que la heurística Manhattan que usamos para tomar la distancia que hay de Q y W a la meta daba igual ya que la misma no tiene en cuenta la pared a la izquierda de F y ante dicho empate elige Q por orden alfabético. Así Q y W tienen un valor de 3 y se elegía el camino de Q que termina siendo el mismo que toma A* ante el gran costo de W.

EJ 6)

Informe sobre el algoritmo A* implementado en Pygame

1. Propósito del programa

El código implementa el **algoritmo de búsqueda A*** (A-star), un método de búsqueda de caminos en grafos que garantiza encontrar la ruta óptima (más corta) entre dos puntos, siempre que la heurística utilizada sea admisible. El entorno gráfico se construye con **Pygame**, permitiendo la interacción del usuario para establecer:

- **Punto de inicio** (start)
- **Punto de destino** (end)
- **Obstáculos** (barriers)

La visualización en tiempo real muestra cómo A* explora el espacio hasta encontrar la mejor ruta disponible.

2. Herramientas clave utilizadas

a) Librerías

- **pygame**: renderizado gráfico y manejo de eventos.

Inteligencia Artificial 2025 - Grupo 5

- **queue.PriorityQueue**: estructura de datos para manejar la frontera en A*, donde siempre se expande el nodo con menor $f(n)$.
- **tkinter.messagebox**: muestra mensajes emergentes cuando no existe solución.
- **typing**: aporta tipado estático opcional para mayor claridad del código.

b) Clase **Spot**

Cada celda del grid se representa con la clase **Spot**, que encapsula:

- **Estado**: posición en el grid, color actual (blanco, barrera, inicio, destino, etc.), lista de vecinos.
- **Comportamiento**: métodos para cambiar su estado (**make_start()**, **make_barrier()**, **make_path()**...), dibujarse (**draw()**), y actualizar sus vecinos transitables (**update_neighbors()**).

Esto convierte cada celda en un **nodo** del grafo implícito sobre el cual trabaja A*.

c) Funciones auxiliares

- **h(p1, p2)**: heurística Manhattan (distancia en cuadrícula ortogonal).
- **reconstruct_path(came_from, current, draw_cb)**: reconstruye el camino final a partir del diccionario **came_from**.
- **make_grid()** y **draw_grid()**: generan y dibujan la grilla.
- **draw()**: refresca la pantalla en cada iteración.
- **get_clicked_pos()**: traduce coordenadas del mouse a posiciones de celda.

d) Algoritmo principal **algorithm()**

Implementa la lógica de A*, usando:

- **open_set (PriorityQueue)**: lista de nodos frontera, ordenados por su costo estimado $f(n) = g(n) + h(n)$.
- **came_from**: diccionario que almacena la ruta previa.
- **g_score**: costo real desde el inicio.

Inteligencia Artificial 2025 - Grupo 5

- **f_score**: costo estimado total.

El bucle principal expande nodos, actualiza costos, agrega vecinos a la frontera, y termina cuando alcanza el nodo destino o cuando se agotan las opciones.

e) Interacción

- **Clic izquierdo**: marcar inicio, fin u obstáculos.
 - **Clic derecho**: borrar una celda.
 - **Barra espaciadora**: ejecutar A*.
 - **Tecla C**: reiniciar el tablero.
 - **Ventana emergente**: indica cuando no existe solución.
-

3. Estructura del programa

El código está organizado en secciones claras:

1. **Imports y configuración inicial** (colores, ventana, librerías).
 2. **Clase Spot**: definición de nodos del grid.
 3. **Funciones auxiliares**:
 - Heurística (**h**)
 - Reconstrucción de ruta (**reconstruct_path**)
 - Generación/dibujo del grid (**make_grid**, **draw_grid**, **draw**, **get_clicked_pos**)
 4. **Algoritmo A*** (**algorithm**).
 5. **Bucle principal main()**: maneja la interacción del usuario y el flujo del programa.
 6. **Punto de entrada**: **main(WIN, WIDTH)** ejecuta todo cuando se corre el script directamente.
-

Inteligencia Artificial 2025 - Grupo 5

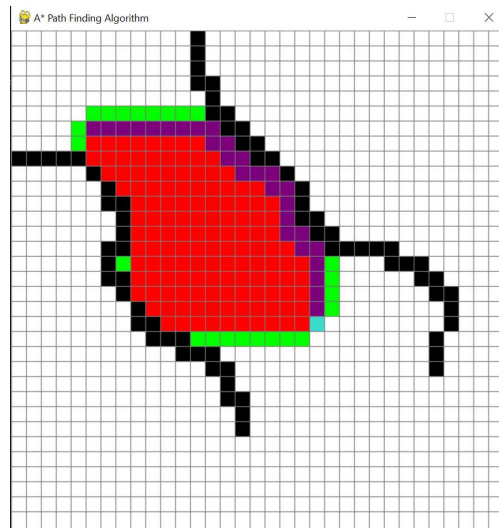
4. Conclusiones

Este programa es un ejemplo didáctico y visual de cómo funciona el algoritmo A*:

- Demuestra la importancia de una buena **estructura de datos** (diccionarios, colas de prioridad).
- Hace uso de **clases** para organizar la lógica del grafo.
- Implementa **heurística Manhattan**, adecuada para grids sin diagonales.
- Permite comprender tanto la teoría detrás de A* como su aplicación práctica.

En resumen, combina conceptos de **algoritmos de búsqueda**, **programación orientada a objetos** y **visualización interactiva**.

A continuación se muestran algunos resultados de la ejecución del programa, junto con posibles “excepciones”.



Inteligencia Artificial 2025 - Grupo 5

Los posibles “errores” son dos: o el inicio está rodeado de barreras o el fin está rodeado de barreras.

