Temas Tratados en el Trabajo Práctico 2

- Conceptos de Búsqueda no Informada y Búsqueda Informada.
- Concepto de Heurística.
- Abstracción de Problemas como Gráficos de Árbol.
- Estrategias de Búsqueda no Informada: Primero en Amplitud, Primero en Profundidad y Profundidad Limitada.
- Estrategias de Búsqueda Informada: Búsqueda Voraz, Costo Uniforme, A*.

Ejercicios Teóricos

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

La principal diferencia entre la búsqueda informada y la no informada radica en que la primera proporciona la guía sobre dónde y cómo encontrar la solución. Por el contrario, la segunda no aporta información adicional sobre el problema, salvo su especificación.

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

Es una estrategia de resolución de problemas que utiliza atajos mentales o reglas generales para encontrar soluciones de manera rápida y eficiente, aunque no siempre garantice la optimalidad o la exactitud.

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

Sí, es posible que un algoritmo de búsqueda no encuentre una solución. Esto puede ocurrir por varias razones. Los dos escenarios principales son la incompletitud del algoritmo o la imposibilidad inherente del problema.

1. Describa en qué secuencia será recorrido el Árbol de Búsqueda representado en la imagen cuando se aplica un Algoritmo de Búsqueda con la estrategia:

   4.1 Primero en Amplitud.

   4.2 Primero en Profundidad.

   4.3 Primero en Profundidad con Profundidad Limitada Iterativa (comenzando por un nivel de profundidad 1).

4.1. La Búsqueda Primero en Amplitud (BFS) explora el árbol por niveles, de arriba hacia abajo y de izquierda a derecha. Visita todos los nodos de un nivel antes de pasar al siguiente.
Secuencia: A, D, F, I, H, J, C, E, P, O, K, Z, W, B.

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 La Búsqueda Primero en Profundidad (DFS) explora el árbol yendo lo más profundo posible por una rama antes de retroceder y explorar otra. La secuencia de exploración sigue el camino más lejano posible desde la raíz antes de retroceder.
Secuencia: A, D, H, P, O, J, K, B, F, C, Z, W, E, I.

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								

4.3 La Búsqueda de Profundidad Limitada Iterativa (IDDFS) combina la búsqueda en anchura con la de profundidad. Realiza una serie de búsquedas en profundidad con un límite de profundidad que aumenta en cada iteración.
Nivel de Profundidad 1: A, D, F, I.

Estado Actual								
A	D	F	I					
D	F	I						
F	I							
I								

Nivel de Profundidad 2: A, D, H, J, F, C, E, I.

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								

Nivel de Profundidad 3: A, D, H, P, O, J, K, F, C, Z, W, E, I.

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							

Nivel de Profundidad 4: A, D, H, P, O, J, K, B, F, C, Z, W, E, I.

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								

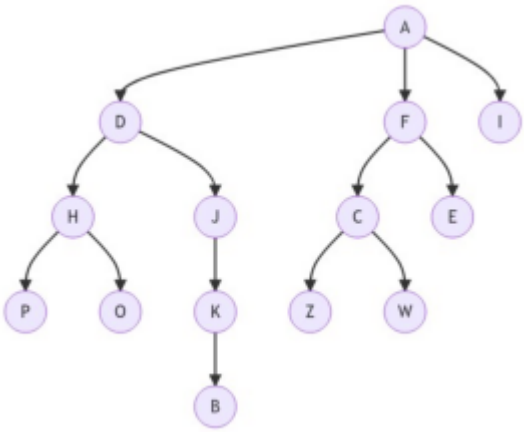
In [23]:

```
import requests
from PIL import Image
from io import BytesIO
import matplotlib.pyplot as plt

# URL directa de Google Drive
url = "https://drive.google.com/uc?export=view&id=1IJDEKWhfMEzXnzr28RgTNOuKBER2NsuP"

# Descargar la imagen
response = requests.get(url)
img = Image.open(BytesIO(response.content))

# Mostrar la imagen
plt.imshow(img)
plt.axis('off')  # Ocultar ejes
plt.show()
```

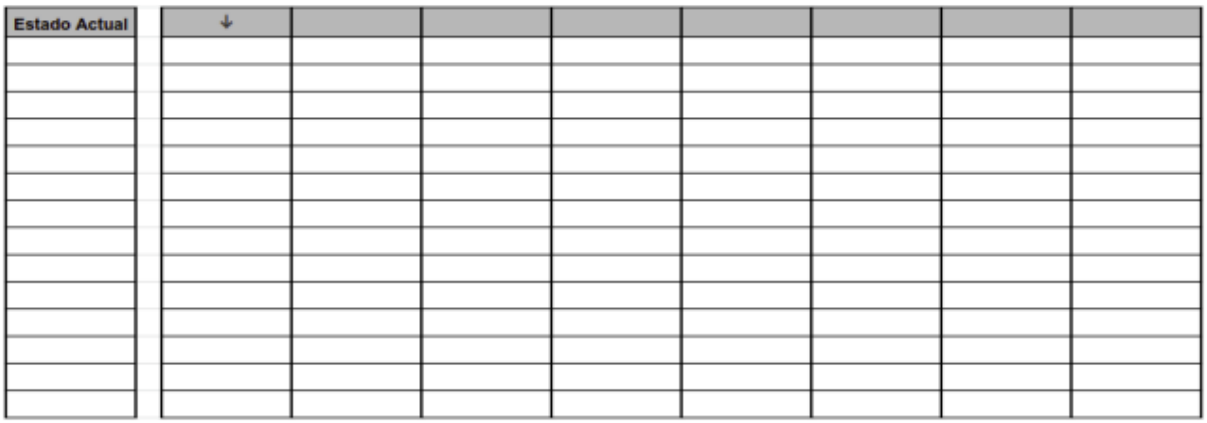


Muestre la respuesta en una tabla, indicando para cada paso que da el agente el nodo que evalúa actualmente y los que están en la pila/cola de expansión según corresponda.

In [24]:

```
url = "https://drive.google.com/uc?export=view&id=1fW_BgT5muzQffMVRcIiB2mM2Zf66Nb-m"
response = requests.get(url)
img = Image.open(BytesIO(response.content))

# Mostrar con figura más grande
plt.figure(figsize=(img.width / 80, img.height / 80))  # ajustá el divisor según den
plt.imshow(img)
plt.axis('off')
plt.show()
```



Ejercicios de Implementación

1. Represente el tablero mostrado en la imágen como un árbol de búsqueda y a continuación programe un agente capaz de navegar por el tablero para llegar desde la casilla I a la casilla F utilizando:

   `5.1 La estrategia Primero en Profundidad.`

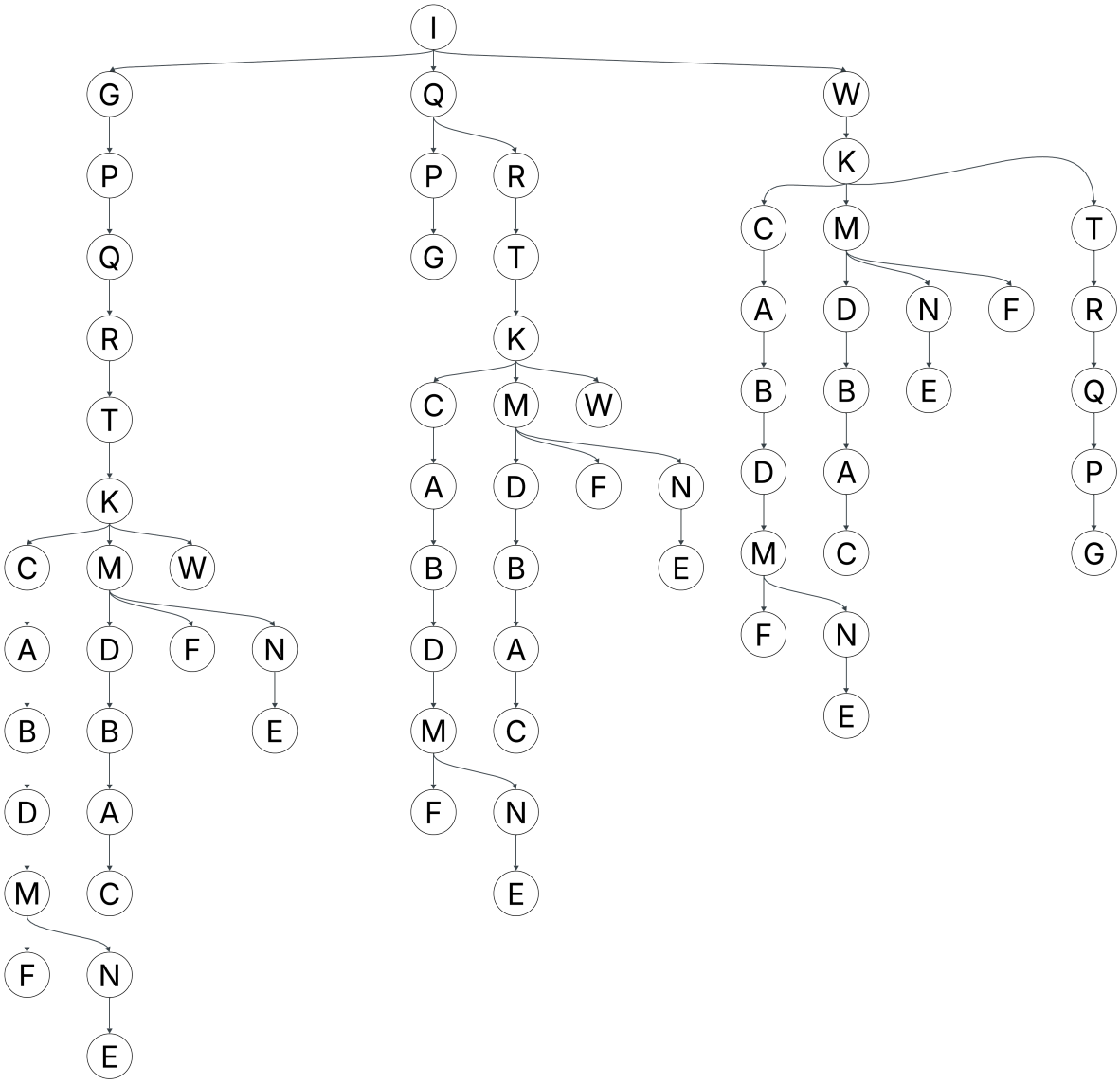
   `5.2 La estrategia Avara.`

   `5.3 La estrategia A*.`

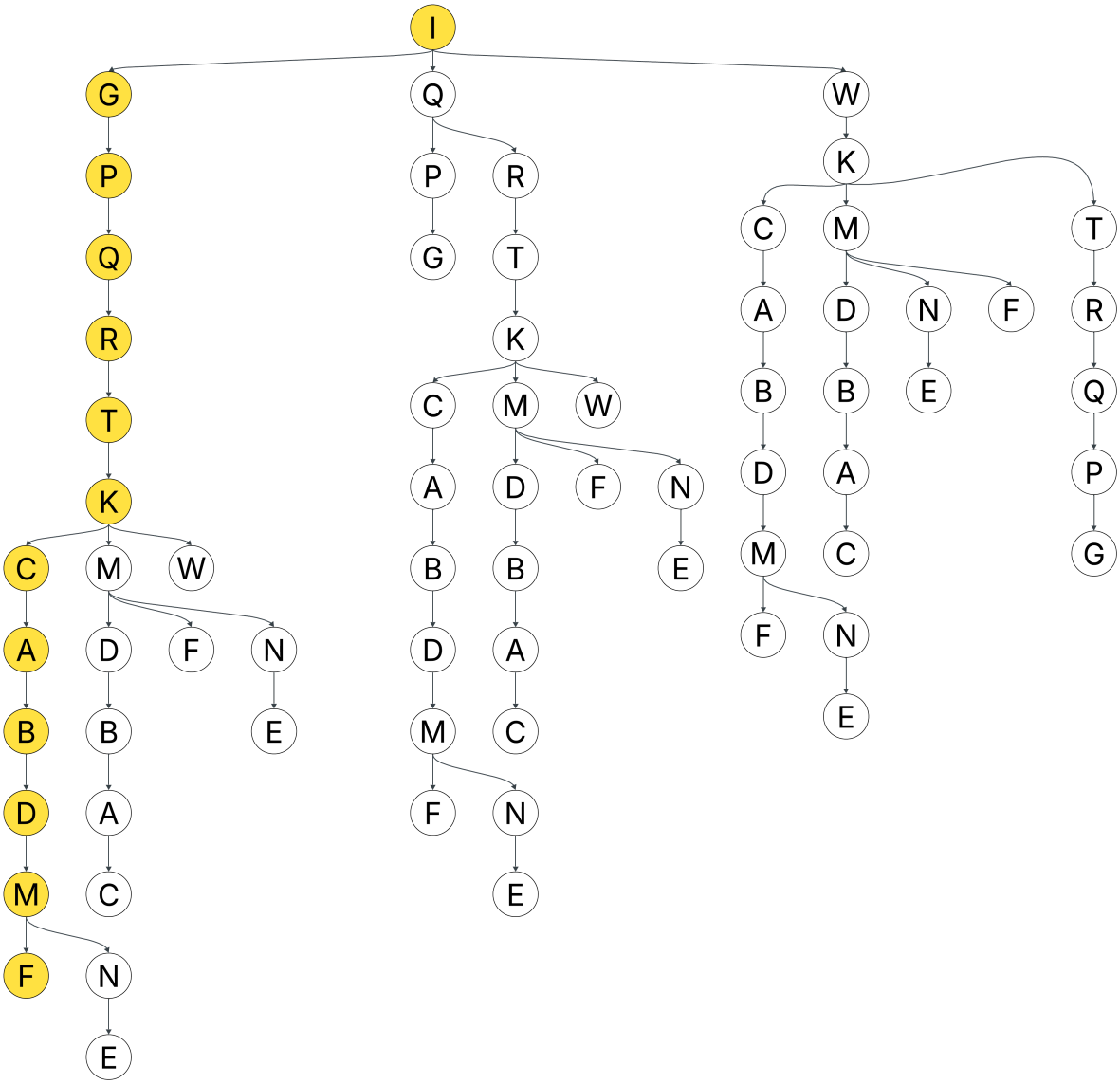
Considere los siguientes comportamientos del agente:

- El agente no podrá moverse a las casillas siguientes si las separa una pared.
- La heurística empleada en el problema es la Distancia de Manhattan hasta la casilla objetivo (el menor número de casillas adyacentes entre la casilla actual y la casilla objetivo).
- El costo de atravesar una casilla es de 1, a excepción de la casilla W, cuyo costo al atravesarla es 30.
- En caso de que varias casillas tengan el mismo valor para ser expandidas, el algoritmo eligirá en orden alfabético las casillas que debe visitar.

Arbol General

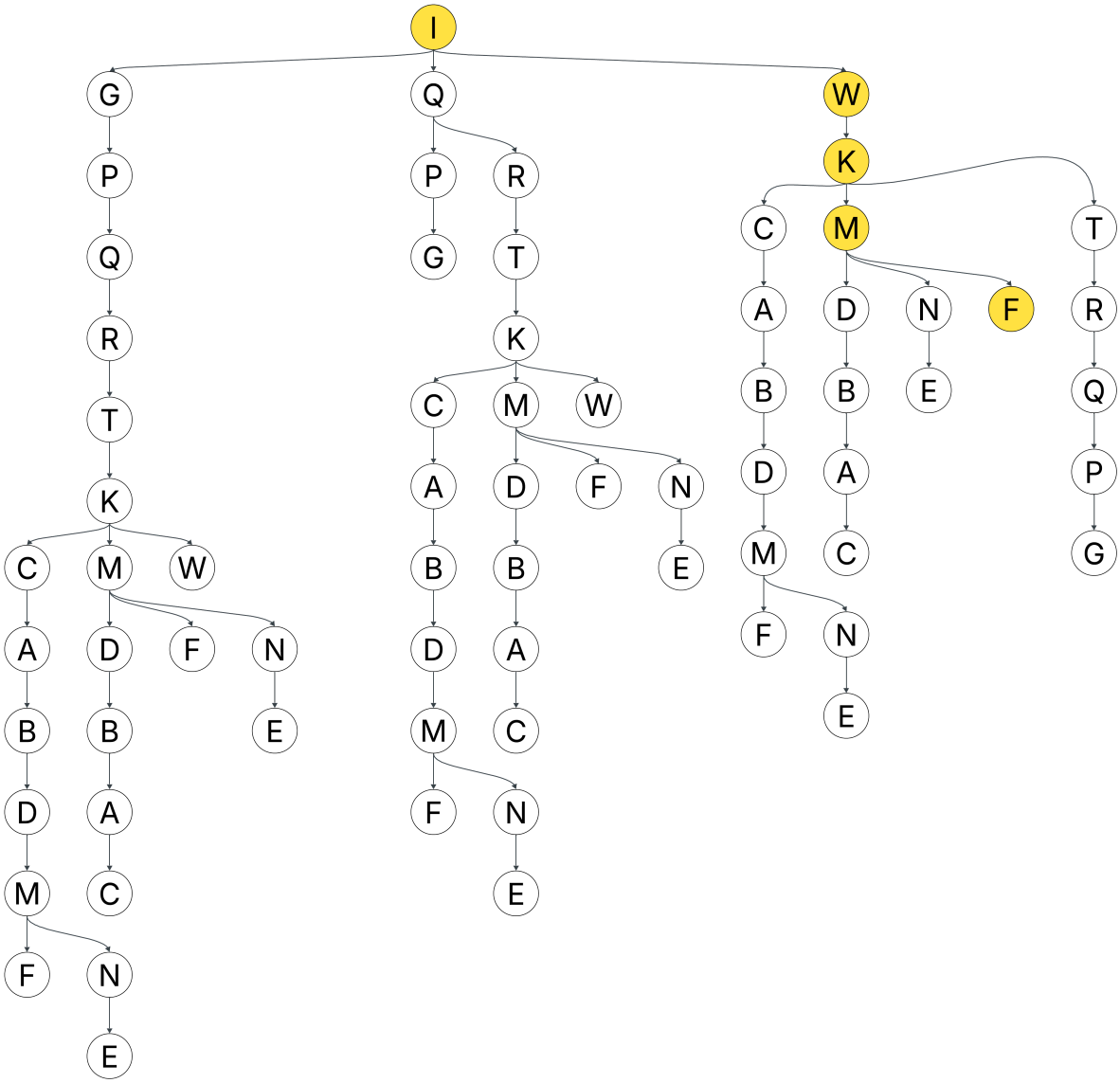


5.1 Estrategia Primero en Profundidad.

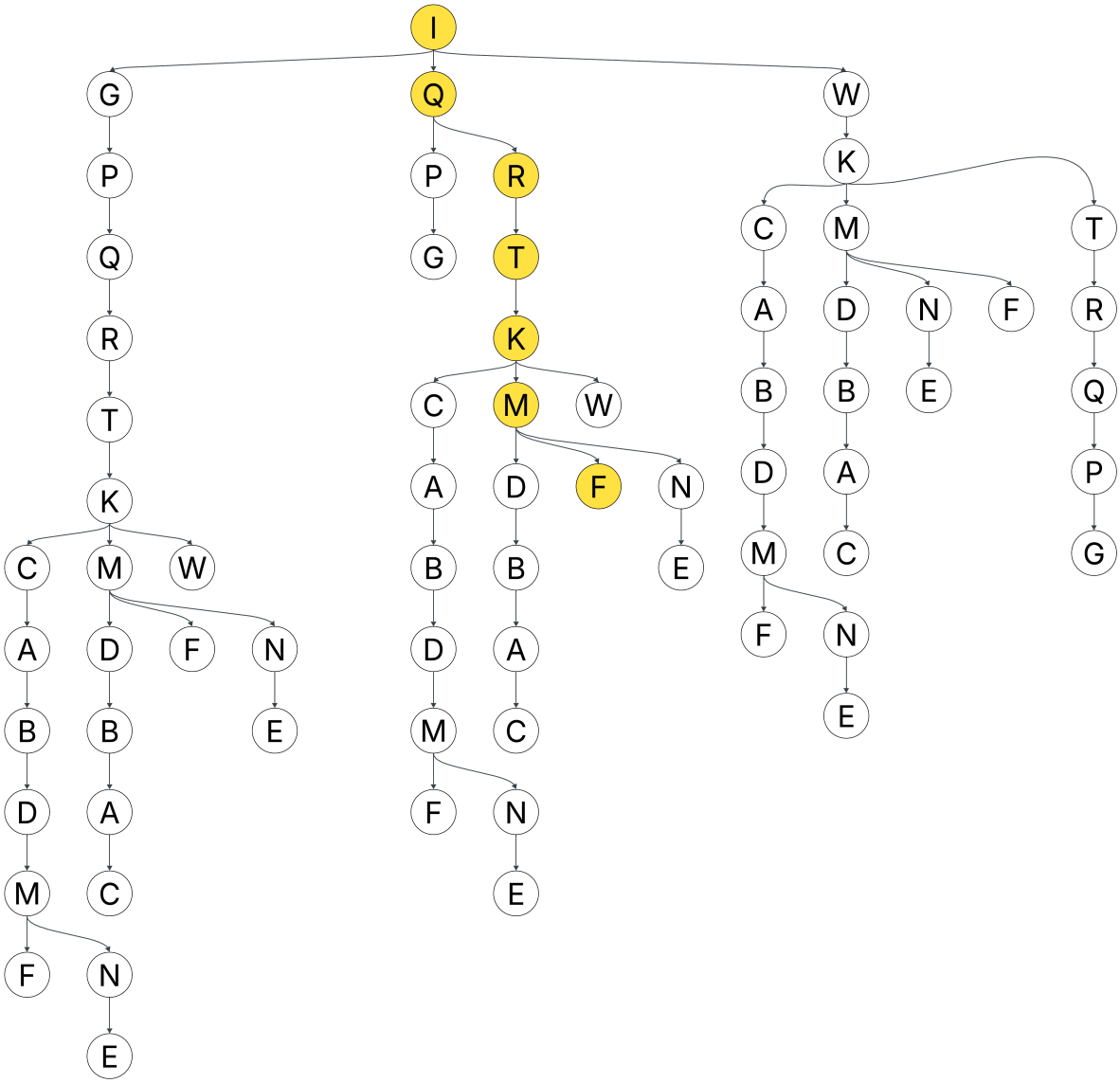


5.2 Estrategia Avara.

Aclaración: Este es el camino que se debería obtener, pero como la heurística está implementada con distancia manhattan, Q y W tienen la misma heurística y al estar ordenado alfabéticamente, Q se escoge primero y por tanto el camino resultante es igual al de la estrategia A *ilustrada más abajo.*

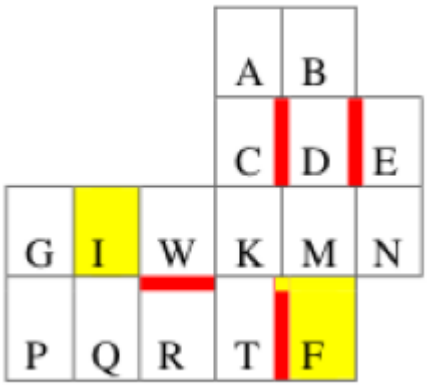


5.3 Estrategia A.



In [25]:

```
url = "https://drive.google.com/uc?export=view&id=1FajYiBQ507o6yiE7MndL-PQXyoyELtuD"
response = requests.get(url)
img = Image.open(BytesIO(response.content))
plt.imshow(img)
plt.axis('off')
plt.show()
```

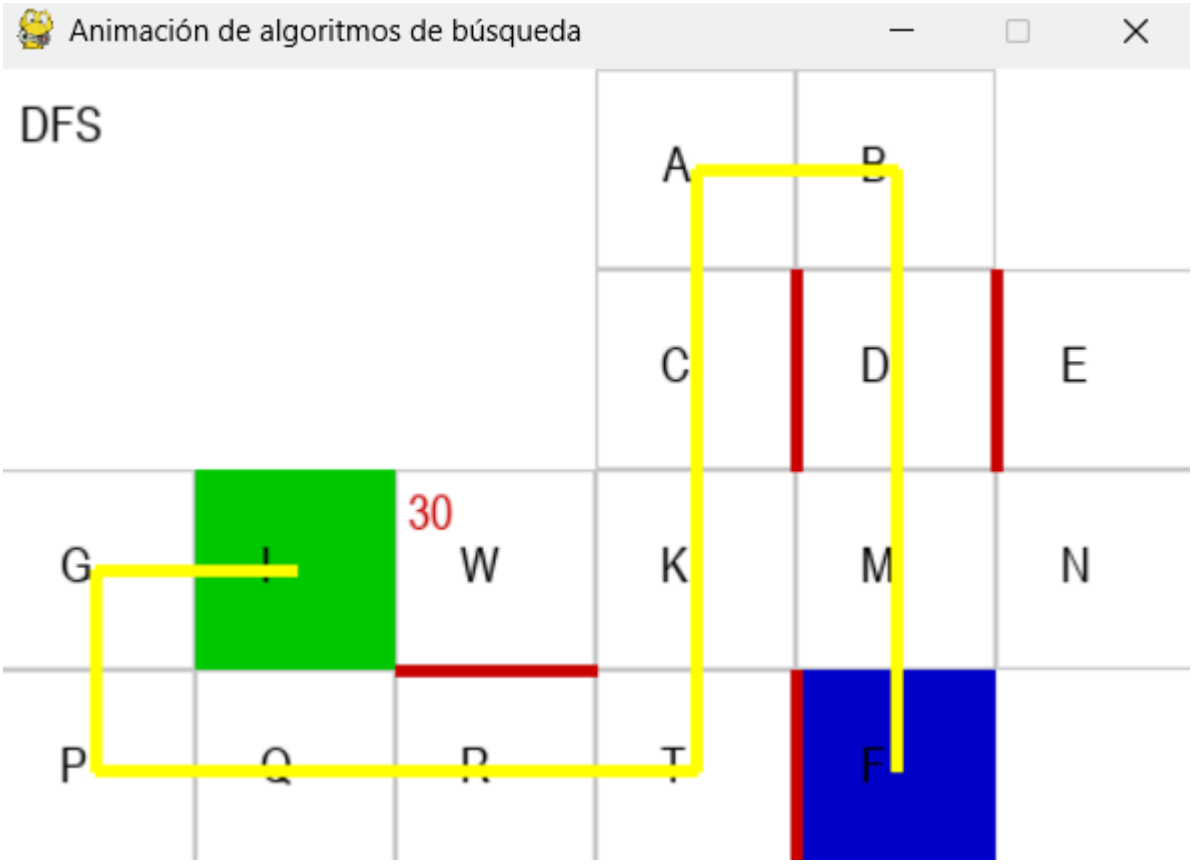


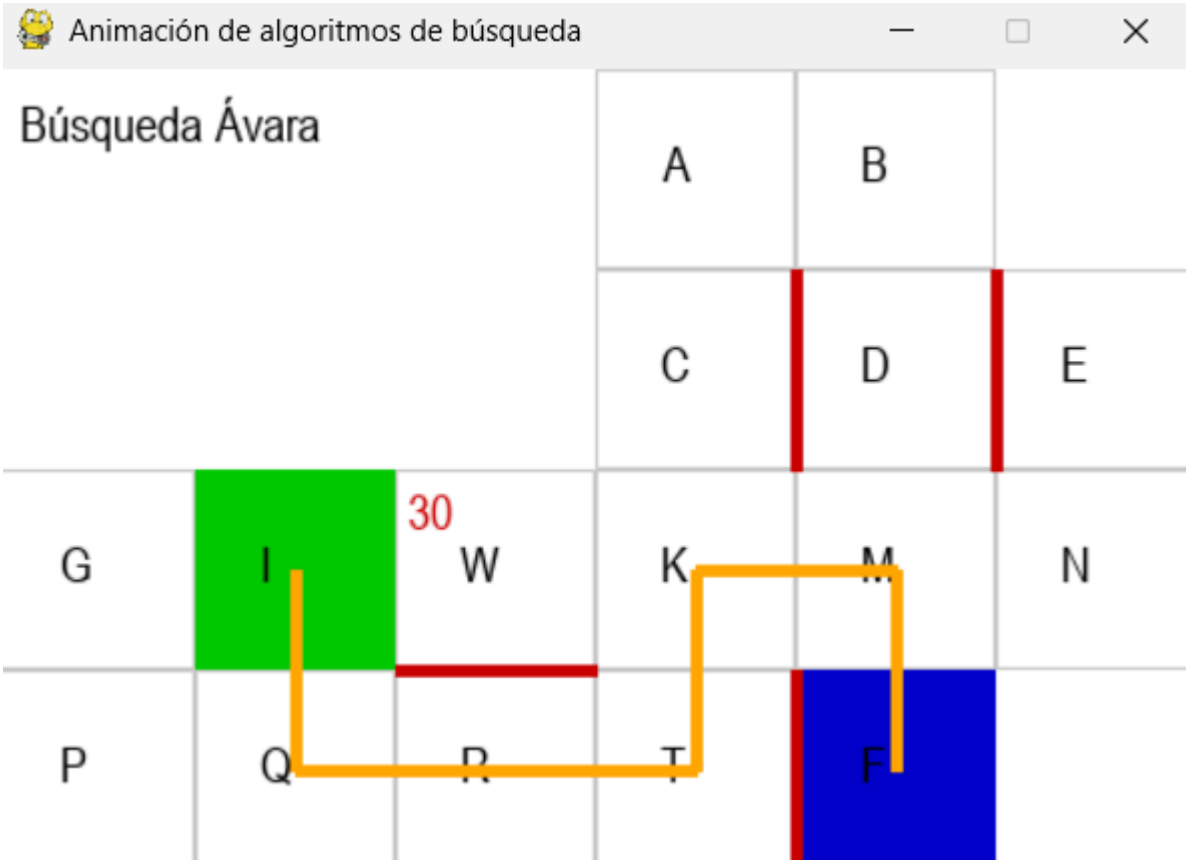
Desarrollado en el archivo Punto5.py

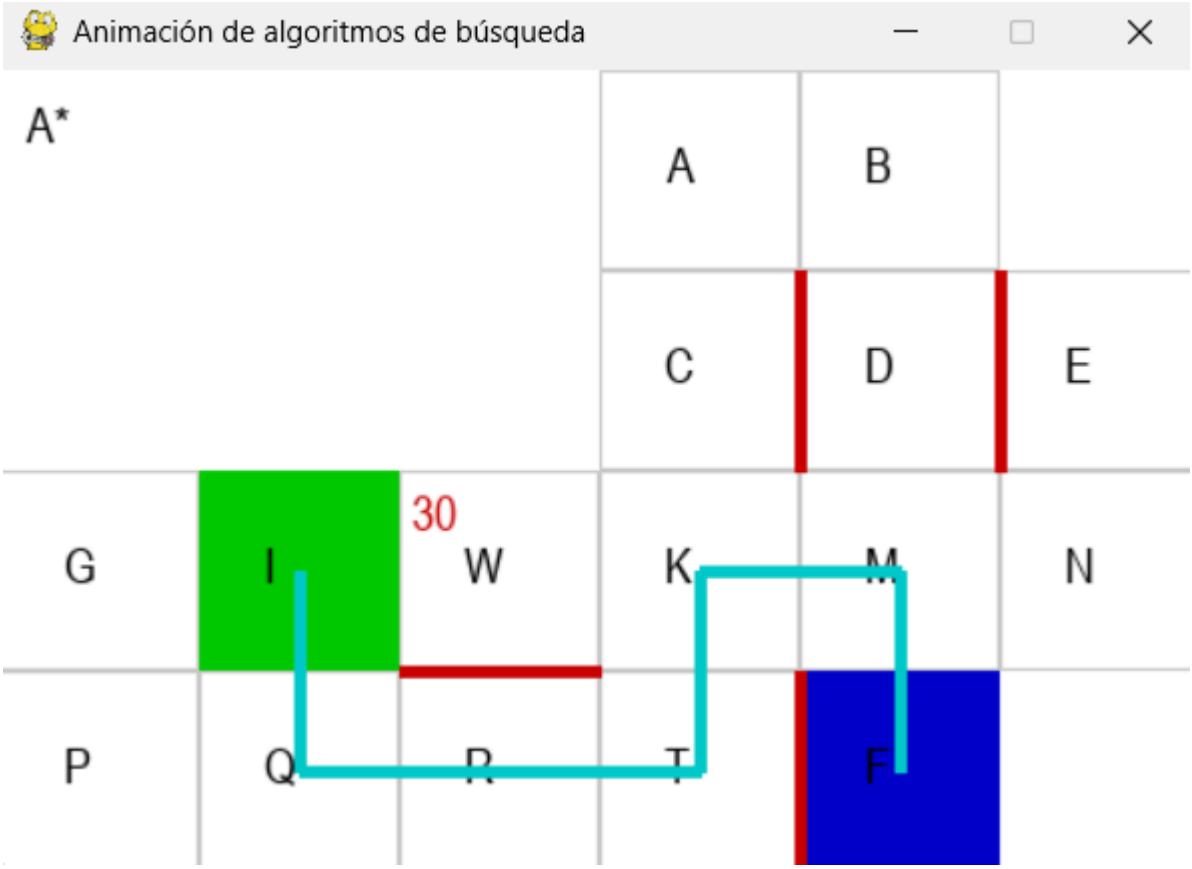
Resultados en consola

```
Búsqueda Primero en Profundidad: ['I', 'G', 'P', 'Q', 'R', 'T', 'K', 'C', 'A', 'B', 'D', 'M', 'F']
Búsqueda Avara: ['I', 'Q', 'R', 'T', 'K', 'M', 'F']
Búsqueda A*: ['I', 'Q', 'R', 'T', 'K', 'M', 'F']
```

Resultados con animación en pygame

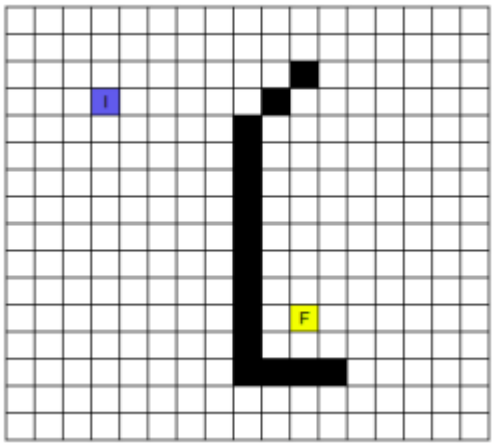






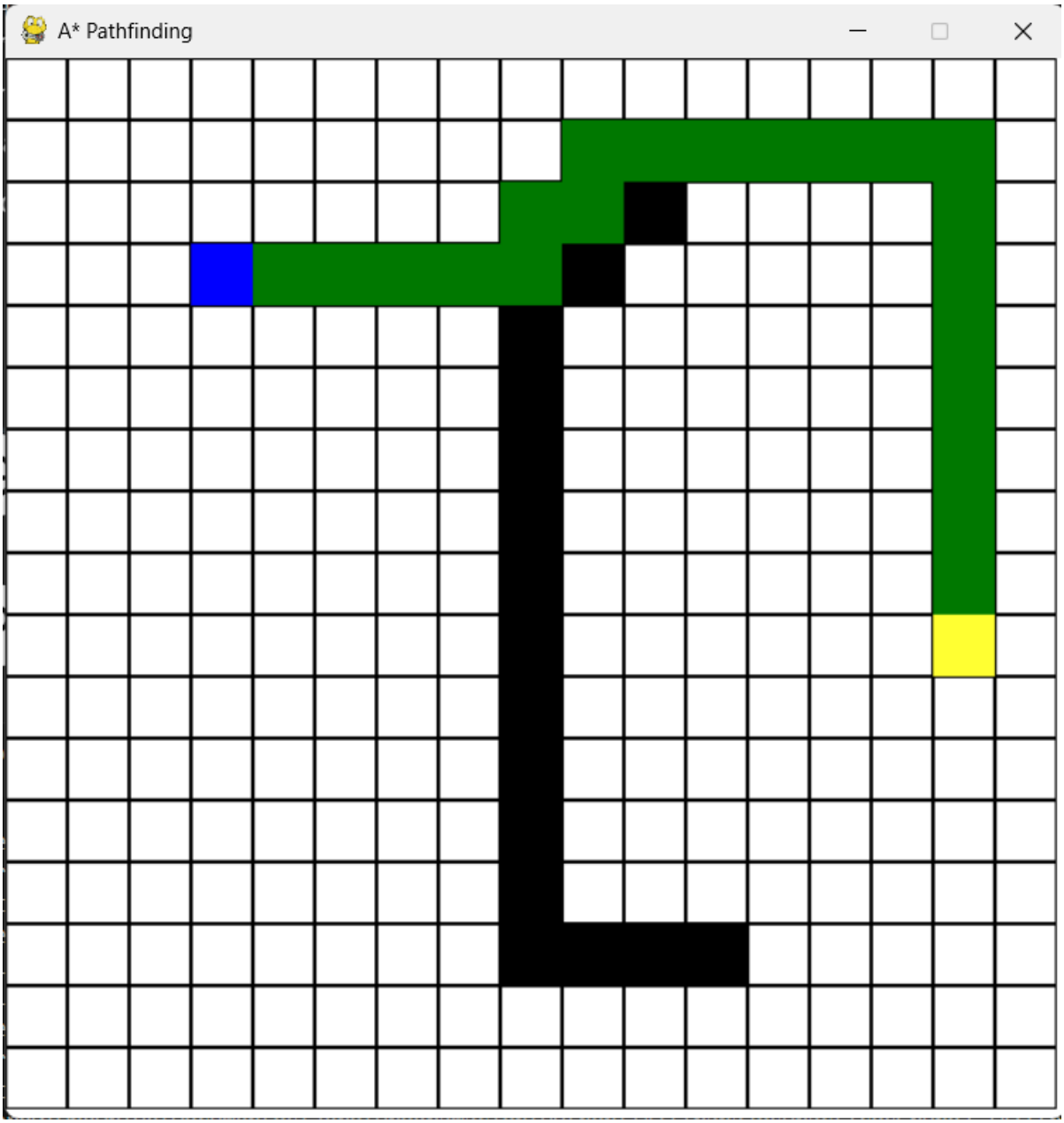
1. Desarrolle un agente que emplee una estrategia de búsqueda A* para ir de una casilla a otra evitando la pared representada, pudiendo seleccionar ustedes mismos el inicio y el final. Muestre en una imagen el camino obtenido.

In [28]:
```
url = "https://drive.google.com/uc?export=view&id=1fD2Ws5oqFU9_RTj-yX9BIvslXJiqcLCZ"
response = requests.get(url)
img = Image.open(BytesIO(response.content))
plt.imshow(img)
plt.axis('off')
plt.show()
```



Desarrollado en un archivo aparte Punto6.py

Ejemplo:



Bibliografía

Russell, S. & Norvig, P. (2004) *Inteligencia Artificial: Un Enfoque Moderno*. Pearson Educación S.A. (2a Ed.) Madrid, España

Poole, D. & Mackworth, A. (2023) *Artificial Intelligence: Foundations of Computational Agents*. Cambridge University Press (3a Ed.) Vancouver, Canada