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?

- En una estrategia de búsqueda a ciegas, el agente no cuenta con información más allá de la provista por el contexto del problema, lo que lleva a que este no pueda elegir una ruta conveniente u óptima, simplemente puede expandir estados o comprobar si estos son o no el objetivo.
- En una estrategia de búsqueda informada el agente cuenta con información que va más allá de la definición del problema, lo que le permite optimizar rutas hacia el/los objetivo/s.

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

Una Heurística es una regla o conjunto de reglas que un agente sigue durante la búsqueda del objetivo, aunque no siempre llegue a la solución óptima.

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

Si es posible que un algoritmo de búsqueda no tenga solución. Un ejemplo puede ser una búsqueda en GPS de una ruta que lleve a un auto al otro lado de un lago muy grande, no existirá dicha ruta y por lo tanto la búsqueda no tendrá solución.

4. 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.

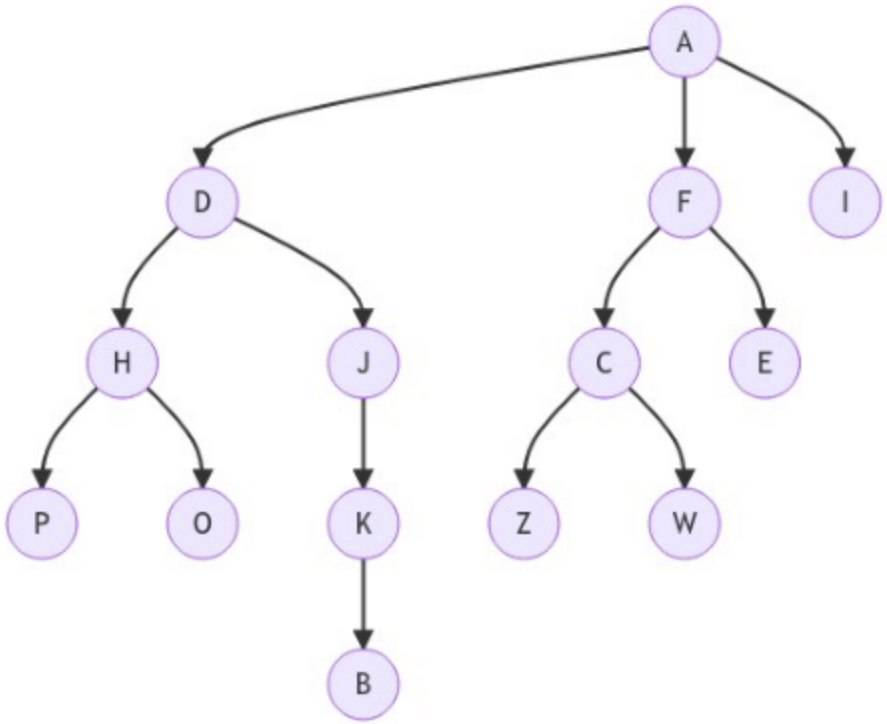
4.3 Primero en Profundidad con Profundidad Limitada Iterativa (comenzando por un nivel de profundidad 1).

In [10]:
```
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.

Secuencias:

- 4.1 Primero en Amplitud

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

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

Estado Actual	↓							
Límite 1								
A	D	F	I					
D	F	I						
F	I							
I								

Estado Actual	↓							
Límite 2								
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								

Estado Actual		↓							
Límite 3									
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									

Estado Actual		↓							
Límite 4									
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									

Ejercicios de Implementación

5. 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.`

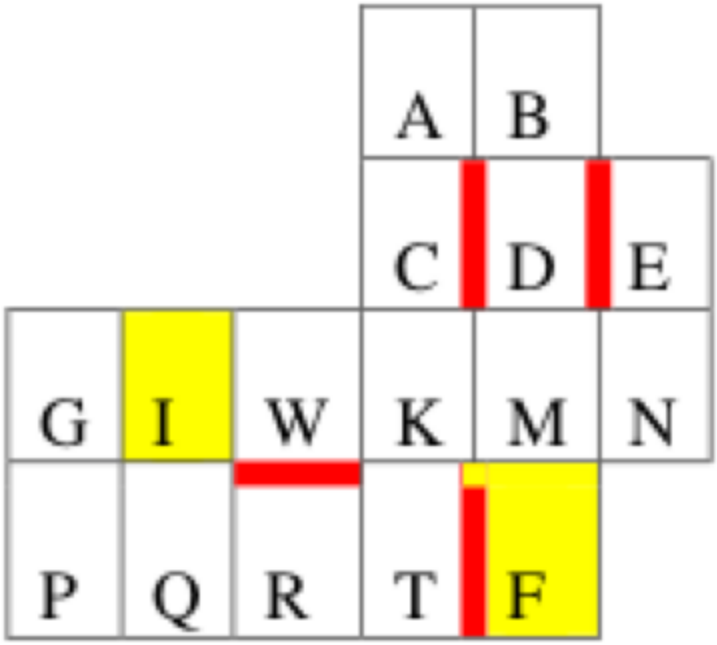
   `5.3 La estrategia A*.`

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.

In [11]:
```
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()
```

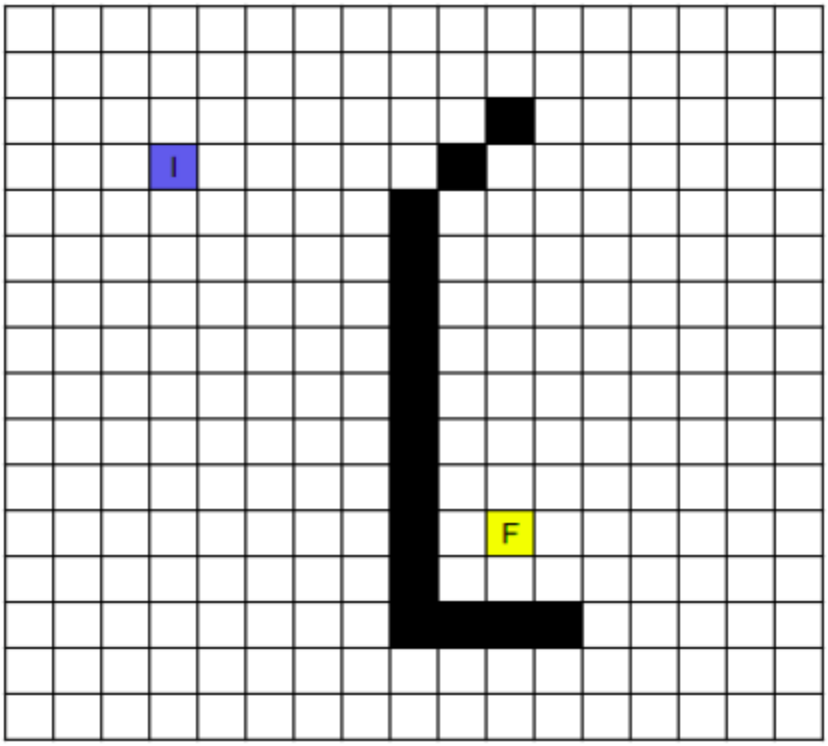


In [ ]:
```

```

6. 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 [12]:
```
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()
```



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