Inteligencia Artificial 1

Universidad Nacional de Cuyo - Facultad de Ingeniería

Trabajo Práctico N°8

Trabajo Práctico N°2: Algoritmos de Búsqueda

Grupo 2: Avila J., Barrios F., Patricelli N.

Septiembre 2025

---

1 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*.

2 Ejercicios Teóricos

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

La principal diferencia entre una estrategia de búsqueda informada y una no informada es que la primera utiliza conocimiento adicional para guiar su camino hacia la solución, mientras que la segunda opera “a ciegas” sin ninguna información extra.

2.2 ¿Qué es una heurística y para qué sirve?

Una heurística es un atajo mental o una regla práctica para resolver un problema de forma rápida, aunque no siempre garantice la solución perfecta. Es una estimación inteligente que te ayuda a tomar decisiones.

Sirve para reducir el tiempo y el esfuerzo de búsqueda en problemas complejos. En el caso del algoritmo A*, la heurística estima qué tan cerca estás de la meta, permitiéndole ignorar caminos menos prometedores y encontrar una solución más eficientemente.

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

Sí, es posible. Un algoritmo de búsqueda no encontrará una solución si no existe un camino viable hacia el objetivo, por ejemplo, si está bloqueado por obstáculos. También puede fallar si el problema es tan complejo que el algoritmo se queda atrapado en un bucle infinito o simplemente no puede explorar todo el espacio de búsqueda en un tiempo razonable. En estos casos, el algoritmo termina sin poder dar una respuesta.

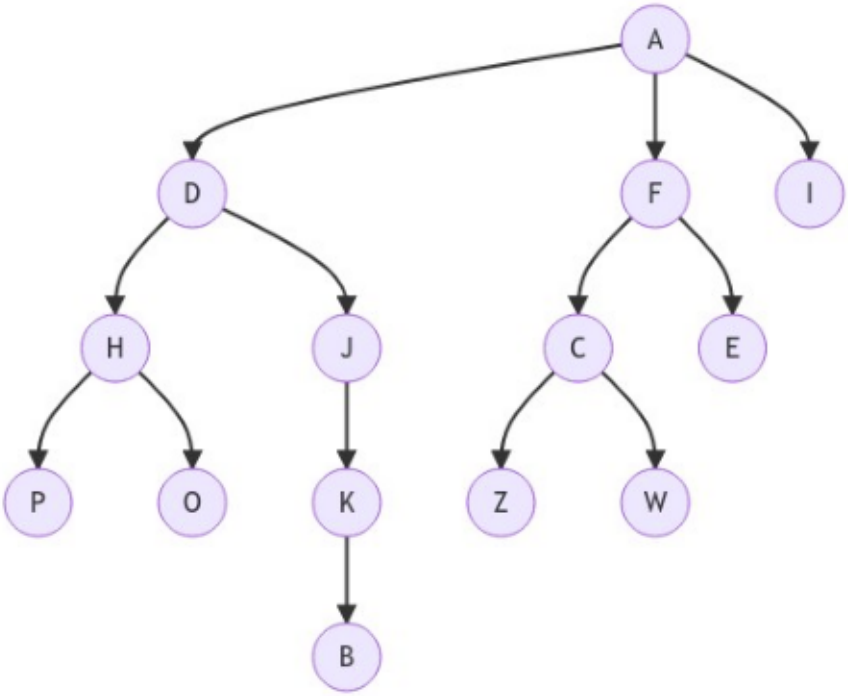
2.4 Describa en qué secuencia será recorrido el Árbol de Búsqueda representado en la imagen cuando se aplica un Algoritmo de Búsqueda con la estrategia:

4.1 Primero en Amplitud.

4.2 Primero en Profundidad.

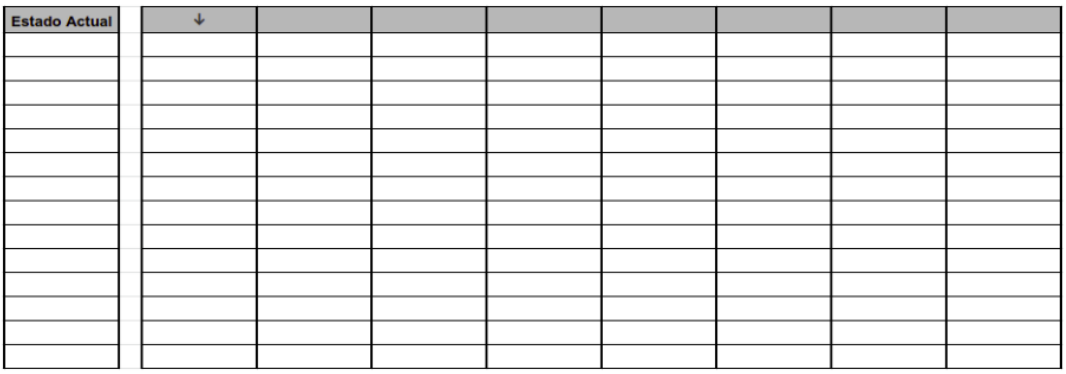
4.3 Primero en Profundidad con Profundidad Limitada Iterativa (comenzando por un nivel c

```python
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.

```
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 densidad deseada
plt.imshow(img)
plt.axis('off')
plt.show()
```



2.4.1 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						

En cada paso de estado actual, se realiza un test objetivo y, en el caso de nos ser el objtivo, el nodo actual se expande a su frontera. La búsqueda primero en amplitud utiliza estructura de filas FIFO, ya que los primeros nodos que se expandieron son los primeros en evaluarse de izquierda a derecha.

2.4.2 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					

La búsqueda primero en profundidad utiliza estructura de pilas LIFO, ya que los últimos nodos que se expanden en cada estado (los que nacen del nodo del estado actual) son los primeros en evaluarse en el siguiente estado, de izquierda a derecha.

2.4.3 Primero en Profundidad con Profundidad Limitada Iterativa.

Limite	Estado Actual	↓				
0	A					
1	A	D	F	I		
	D	F	I			
	F	I				
	I					
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					
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			

Limite	Estado Actual	↓
	E	I
	I	

Este tipo de búsqueda combina la búsqueda primero en amplitud, con primero en profundidad. En realidad, en cada iteración se realiza una búsqueda primero en profundidad de las ramas de izquierda a derecha, pero hasta donde marca el límite actual, y luego se recorre todo el arbol en su amplitud. En la siguiente iteracion se amplia el límite y se vuelve a realizar la búsqueda hasta este nuevo límite.

3 Ejercicios de Implementación

3.1 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.`

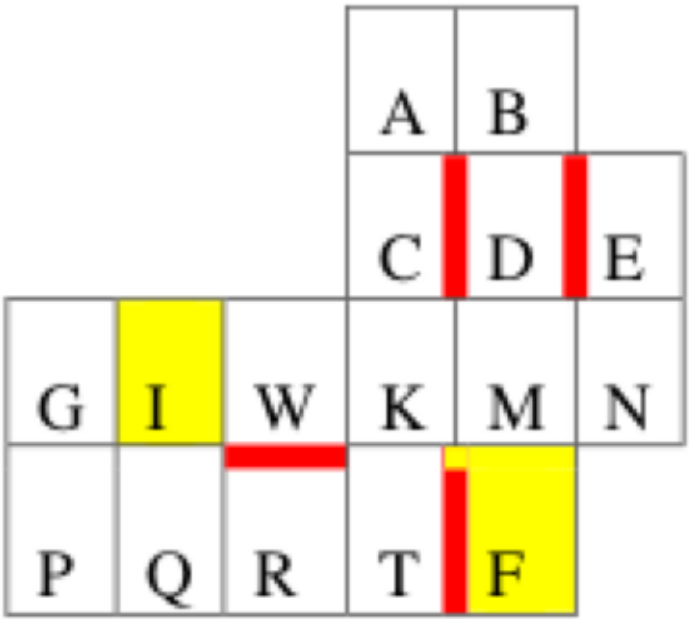
`5.2 La estrategia Avara.`

`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.

```
url = "https://drive.google.com/uc?export=view&id=1FajYiBQ5O7o6yiE7MndL-PQXyoyELtuD"
response = requests.get(url)
img = Image.open(BytesIO(response.content))
plt.imshow(img)
plt.axis('off')
plt.show()
```

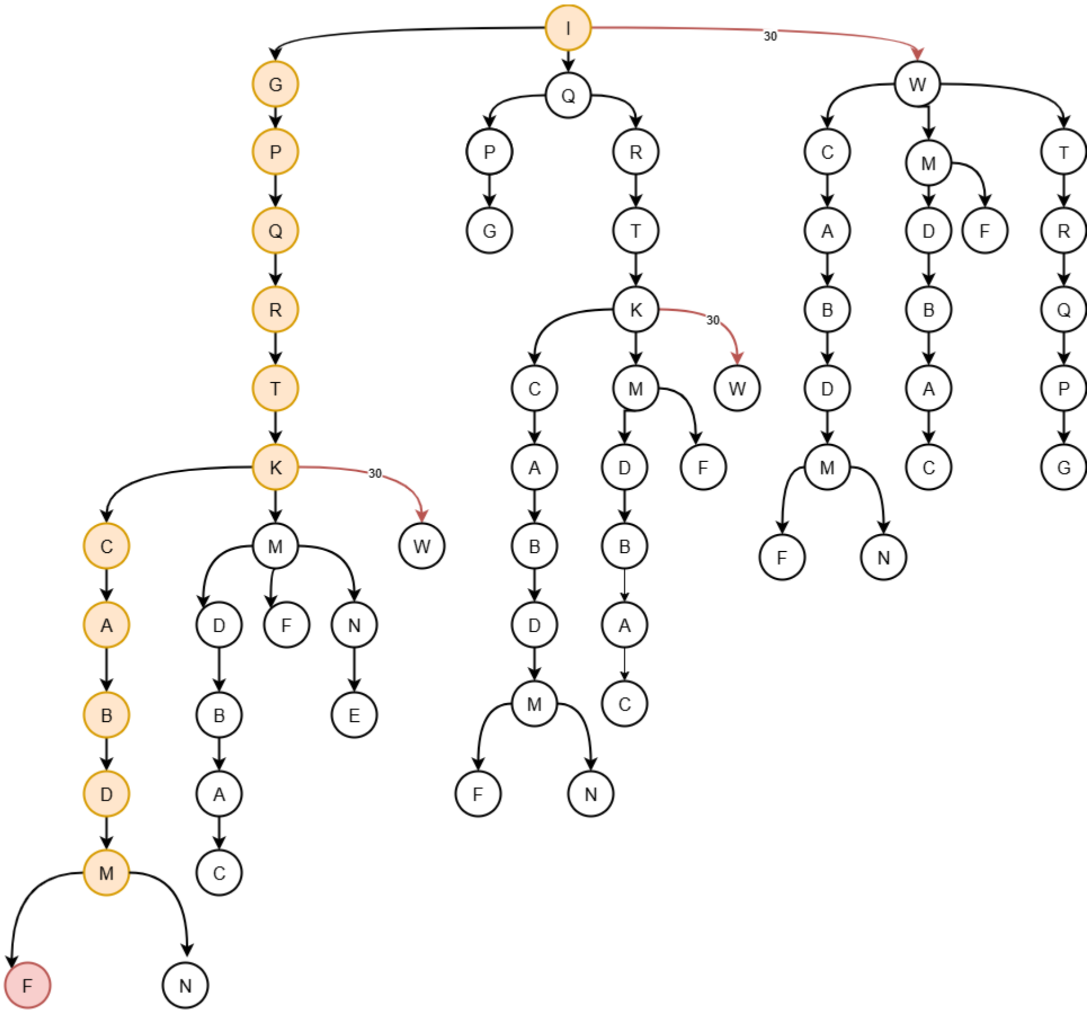


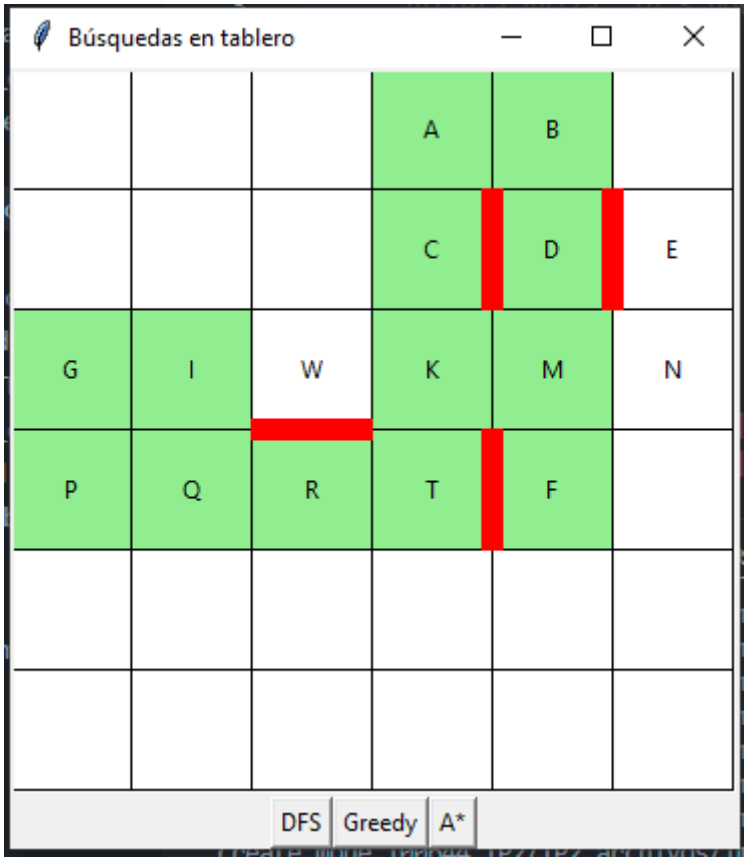
Resultado de nuestro codigo con los distintos algoritmos:

```
Primero en Profundidad: ['I', 'G', 'P', 'Q', 'R', 'T', 'K', 'C', 'A', 'B', 'D', 'M', 'F']
Costo: 12
Largo del camino: 12

Greedy: ['I', 'W', 'K', 'M', 'F']
Costo: 33
Largo del camino: 4

A*: ['I', 'Q', 'R', 'T', 'K', 'M', 'F']
Costo: 6
Largo del camino: 6
```

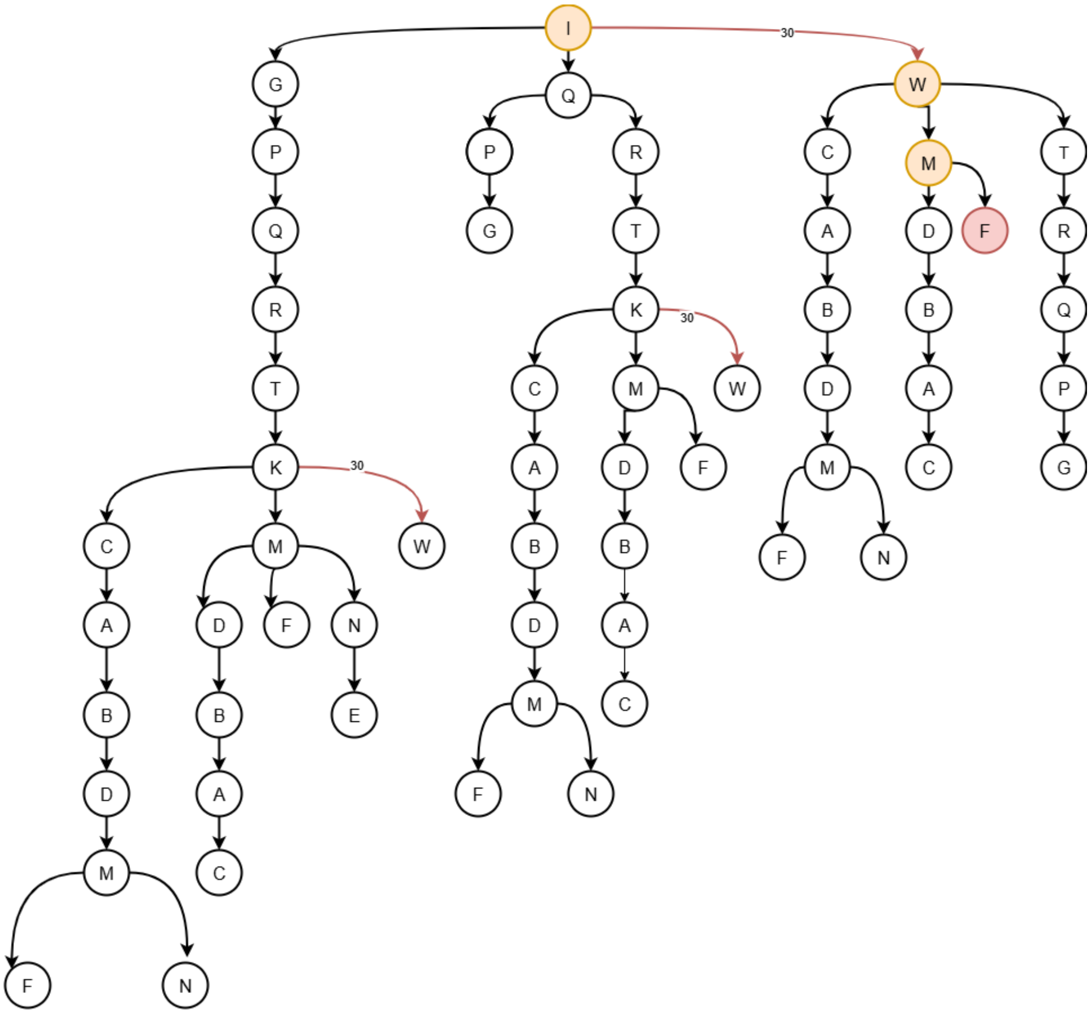
Arboles de busqueda obtenidos con los distinto algoritmos:

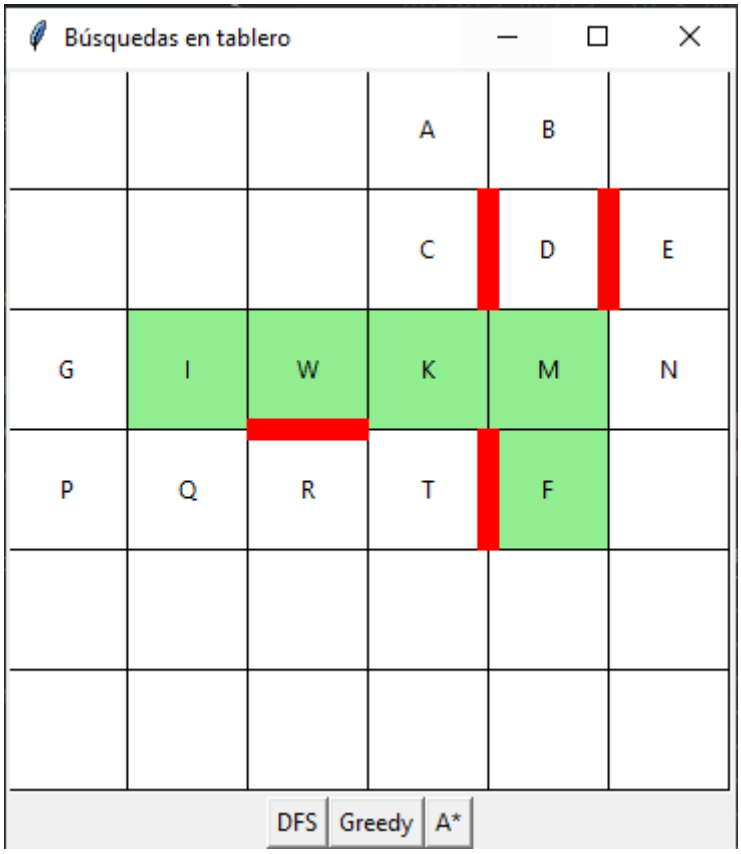
-Primero en profundidad


I
30
G
Q
W
P
P
R
C
M
T
Q
G
T
A
D
F
R
R
K
30
B
B
Q
T
C
M
W
D
A
P
K
30
A
D
F
M
C
G
C
M
W
B
B
F
N
A
D
F
N
D
A
B
B
E
M
C
D
A
F
N
M
C
F
N

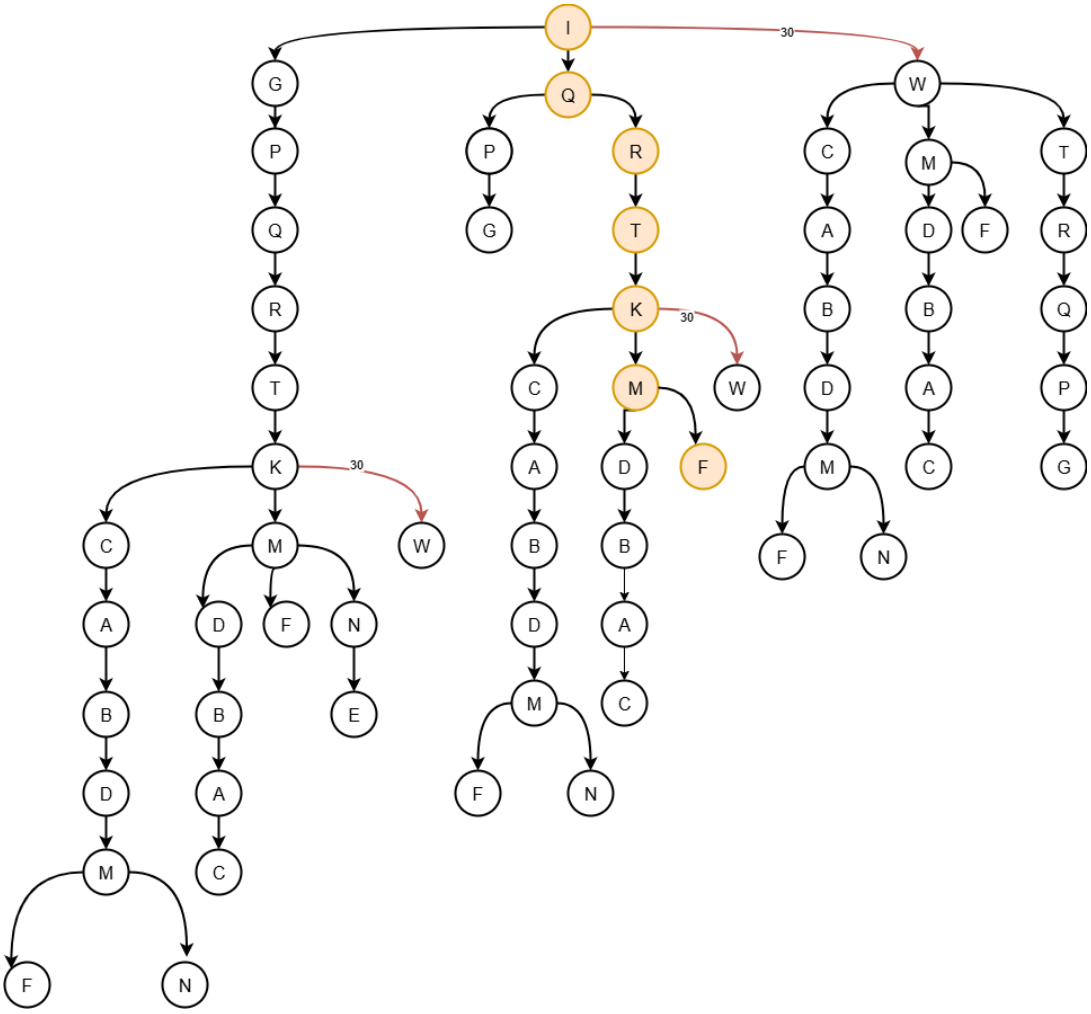


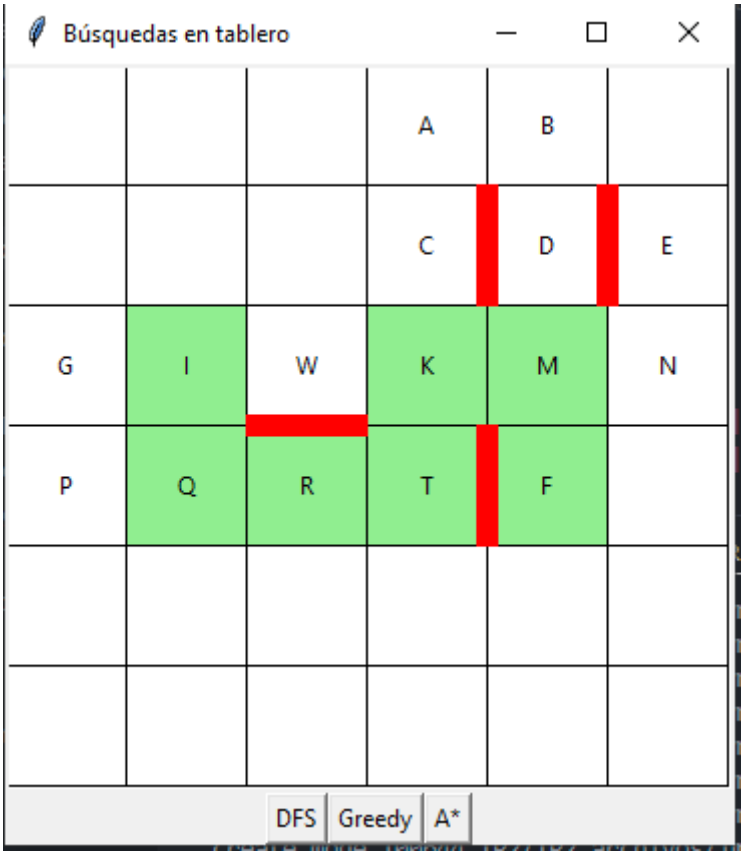
-Estrategia avara:


I
30
G
Q
W
P
P
R
C
M
T
Q
G
T
A
D
F
R
R
K
30
B
B
Q
T
C
M
W
D
A
P
K
30
A
D
F
M
C
G
C
M
W
B
B
F
N
A
D
F
N
D
A
B
B
E
M
C
D
A
F
N
M
C
F
N



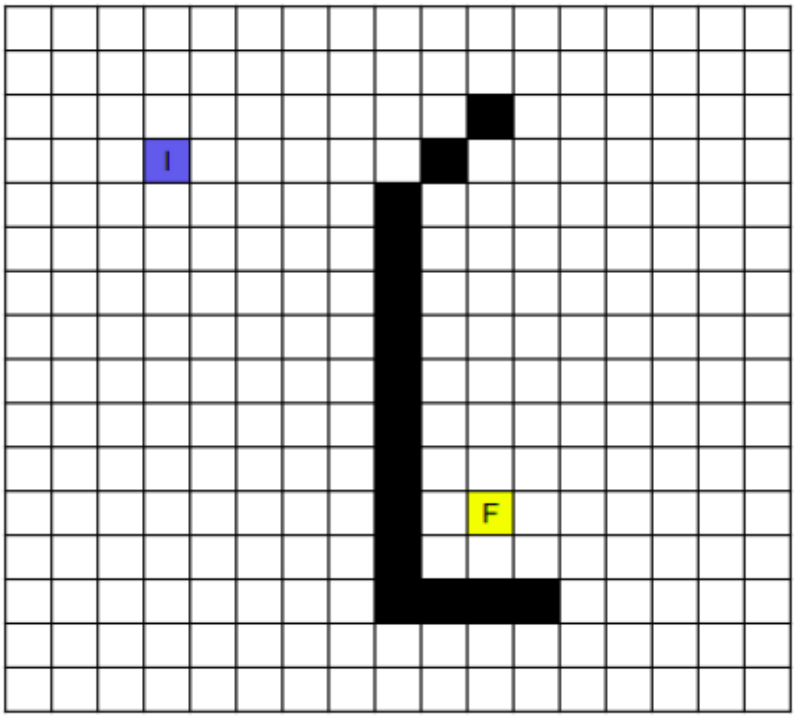
-Estrategia A estrella:


I
30
G
Q
W
P
R
C
M
T
P
G
T
A
D
F
R
Q
R
K
30
B
B
Q
T
C
M
W
D
A
P
K
30
A
D
F
M
C
G
C
M
W
B
B
F
N
A
D
F
N
D
A
B
B
E
M
C
D
A
F
N
B
A
M
C
F
N

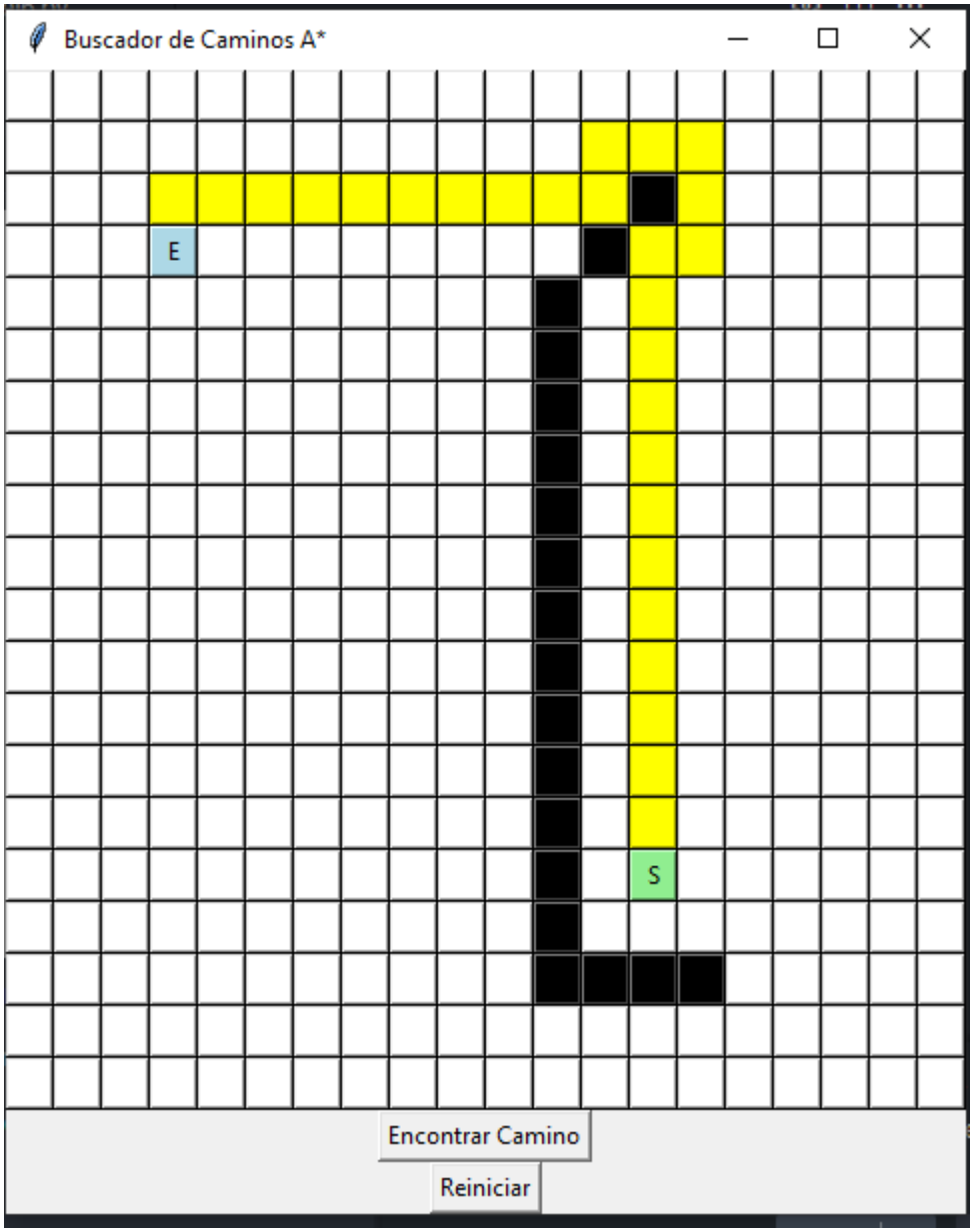


3.2 Desarrolle un agente que emplee una estrategia de búsqueda A* para ir de una casilla a otra evitando la pared representada, pudiendo seleccionar ustedes mismos el inicio y el final. Muestre en una imagen el camino obtenido.

```
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()
```



Captura del resultado del agente con las condiciones iniciales dadas por la consigna.



4 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