INSTITUTO POLITÉCNICO NACIONAL

ESCOM

INTELIGENCIA ARTIFICIAL

LAB 4: BUSQUEDA INFORMADA

MARTÍNEZ CHÁVEZ JORGE ALEXIS

6CV3

04 OCTUBRE 2024

INTRODUCCIÓN

Al algoritmo de A* es un tipo de algoritmo informado, esto por que le ayudamos a saber o cuantificar si estamos mas cerca o mas lejos de la solución, para ello tenemos la función “f = g + h”, en donde “g” nos representa el valor que hemos tenido desde el inicio hasta el nodo donde nos encontramos, en nuestro caso es que nivel de hijo es; ahora la “h” es el valor de la heurística, esta la podemos diseñar según nos ayude, en el caso de los laberintos contabilizamos el numero de casillas de distancia que tenemos con respecto a la salida. Una vez hechas las operaciones, en nuestro caso tomamos el nodo con el “f” de menor valor para llegar a la solución óptima, esto debido que según nuestra función, es la que tiene menor valor.

ALGORITMO

Primero debemos de el nodo con el que vamos a trabajar, para ellos recibirá como parámetros al mismo nodo, la posición actual, el nodo padre, el costo desde el inicio y el valor de la heurística; además tendrá como parámetro calculado la función de A*, que es el valor del costo mas el valor de la heurística. Además para ayudarle al algoritmo, le añadimos 2 funciones principales, para poder hacer comparativas cuando dos nodos son iguales o la posición de uno es menor a la de otro cuando estan en la cola.

```python
# Nodo
class Nodo:
    def __init__(self, posicion, padre=None, g=0, h=0):
        self.posicion = posicion
        self.padre = padre
        self.g = g
        self.h = h
        self.f = g + h

    def __eq__(self, otro):
        return self.posicion == otro.posicion

    def __lt__(self, otro):
        return self.f < otro.f
```

Tambien definimos la fucnión de nuestra heusristica, en este caso vamos a utilizar la distencía directa entre 2 puntos, esto porque lo movimientos solo pueden ser verticales u horizontales, por tanto, no nos interesa saber la distancía lineal, si no la distancía en cantidad de celdas respecto a estos, dos movimientos.

```python
# Heurística
def heuristica(posicion_actual, posicion_final):
    return abs(posicion_actual[0] - posicion_final[0]) + abs(posicion_actual[1] - posicion_final[1])
```

Iniciamos llamando la función principal del algoritmo de A*, en donde se recibe como parametros el laberinto, la posición de inicio y la posición de fin. Se crea un nuevo nodo en donde se le envía la posición de inicio, no tiene padre, no tiene costo hasta ahí y el valor de su heuristica, tambien agregamos el nodo final con la posición de fin. Creamos una cola que sera en forma de tupla, con el nodo y con el valor de la función calculada, es una cola de prioridad por lo que se iran ordenando conforme al valor mas pequeño de la función calculada, esta cola sera nuestra cola de nodos frontera. Ademas creamos la lista donde guardaremos los nodos visitados.

```python
# Algoritmo A*
def resolver_laberinto_A_star(laberinto, inicio, fin):
    nodo_inicio = Nodo(inicio, None, 0, heuristica(inicio, fin))
    nodo_fin = Nodo(fin)

    cola = []
    heapq.heappush(cola, (nodo_inicio.f, nodo_inicio))

    visitados = set()
```

Ahora haremos el ciclo en donde debemos de ir sacando nodos del nodo frontera, se sacara el nodo con el menor valor de la función, este nodo lo almecenaremos en un nodo actual y mandamos a imprimir que el camino que se esta tomando. Hacemos una comparación para saber si hemos llegado a la solución, en caso de que si, debemos de romper el ciclo; en caso de que no, debemos de continuar con el ciclo, primero debemos de agregar este nodo a la lista de nodos visitados; depues debemos de generar los posibles caminos que es abajo (resta fila), arriba (suma fila), izquierda (resta columna) y derecha (suma columna). Mandamos a ejecutar estos posibles movimientos para saber si son movimientos validos y nos regresa los que si, ahora debemos de comparara que estos moviemintos no esten dentro de la lista de nodos visitados, en caso de que no, entonces ahora los agregamos en la cola de nodos frontera creando cada uno de los nodos en donde como nodo padre les enviamos el nodo actual, le mandamos el valor de camino actual mas uno y el calculo de su heuristica.

```
    if nodo_actual.posicion == nodo_fin.posicion:
        print("¡Laberinto resuelto!")
        return reconstruir_camino(nodo_actual)

    visitados.add(nodo_actual.posicion)

    fila, columna = nodo_actual.posicion
    movimientos = [(fila-1, columna), (fila+1, columna), (fila, columna-1), (fila, columna+1)]

    for movimiento in movimientos:
        if es_movimiento_valido(laberinto, movimiento) and movimiento not in visitados:
            nuevo_nodo = Nodo(movimiento, nodo_actual, nodo_actual.g + 1, heuristica(movimiento, fin))

            heapq.heappush(cola, (nuevo_nodo.f, nuevo_nodo))

print("No se encontró solución.")
return None
```

Para poder imprimir el laberinto, primero debemos de reconstruir el camino, para ello madnamos llamar la función en donde hace un ciclo en donde cada uno de los nodos tienen que ir viajando hacía el nodo padre, así hasta que no haya un nodo pade; para esto va creando un arreglo con las posicones visitadas y las regresa en reversa para ir de inicio hacía fin.

```
# Reconstrucción del camino final
def reconstruir_camino(nodo):
    camino = []
    actual = nodo
    while actual is not None:
        camino.append(actual.posicion)
        actual = actual.padre
    camino.reverse()
    return camino
```

Una vez que hayamos tenido el camino lo debemos de imprimir, para eso llamamos otra función en donde recorremos esta posiciones visitados y vamos imprimiendo una P en lugar del 0, para indicar el caminos que recorrimos.

```python
# Función para imprimir el laberinto
def imprimir_laberinto(laberinto, solucion=[]):
    for i in range(len(laberinto)):
        for j in range(len(laberinto[i])):
            if (i, j) in solucion:
                print("P", end=" ")
            else:
                print(laberinto[i][j], end=" ")
        print()
    print()
```

Ahora para poder haber generado cada una de los nodos, mencionamos que debemos de validar el movimiento, es decir, que este dentro del camino y que no lo estorbe ninguna pared, para ello simplemente hacemos un if para comprobar que si tanto en fila como en columna esta dentro de los rango y que el valor sea “0” que nos indica que no hay pared.

```python
# Movimiento válido
def es_movimiento_valido(laberinto, posicion):
    fila, columna = posicion
    if 0 <= fila < len(laberinto) and 0 <= columna < len(laberinto[0]) and laberinto[fila][columna] == 0:
        return True
    return False
```

Por utlimo cuando se termino de hacer toda la ejecución debemos de comprobar si se llego a la solución, en caso de que si se haya llegado, entonces mandamos impirmir el laberinto solucionado.

```python
# Imprimir la solución final
if camino_solucion:
    print("Se encontró un camino. El laberinto resuelto es:")
    imprimir_laberinto(laberinto, camino_solucion)
else:
    print("No se encontró ninguna solución para el laberinto.")
```

Ahora ponemos a prueba el primer laberinto, el cual tuvo un tiempo de ejecución de: 0.0059967041015625 segundos.

```
PS C:\Users\jorge\OneDrive\Documentos\JORGE\ESCOM\9_SEMESTRE> & C:/Users/jorge/AppData/Local/Programs/Python/Python310/python.exe c:/Users/jorg
ive/Documentos/JORGE/ESCOM/9_SEMESTRE/IA/LABORATORIO/4_BUSQUEDA_INFORMADA/laberinto1.py
1 P 1 1 1
1 0 0 0 1
1 1 1 0 1
1 0 0 0 0
1 1 1 1 1

1 P 1 1 1
1 P 0 0 1
1 1 1 0 1
1 0 0 0 0
1 1 1 1 1

1 P 1 1 1
1 P P 0 1
1 1 1 0 1
1 0 0 0 0
1 1 1 1 1

1 P 1 1 1
1 P P P 1
1 1 1 0 1
1 0 0 0 0
1 1 1 1 1

1 P 1 1 1
1 P P P 1
1 1 1 P 1
1 0 0 0 0
1 1 1 1 1
```

```
1 P 1 1 1
1 P P P 1
1 1 1 P 1
1 0 0 P 0
1 1 1 1 1

1 P 1 1 1
1 P P P 1
1 1 1 P 1
1 0 0 P P
1 1 1 1 1

¡Laberinto resuelto!
Se encontró un camino. El laberinto resuelto es:
1 P 1 1 1
1 P P P 1
1 1 1 P 1
1 0 0 P P
1 1 1 1 1



El tiempo de ejecución fue:  0.0059967041015625
PS C:\Users\jorge\OneDrive\Documentos\JORGE\ESCOM\9_SEMESTRE>
```

Ahora ponemos a prueba el segundo laberinto el cual es mas grande y complejo, el cual tuvo un tiempo de ejecución de:

```
PS C:\Users\jorge\OneDrive\Documentos\JORGE\ESCOM\9_SEMESTRE> & C:/Users/jorge/AppData/Local/Programs/Python/Python310/python.exe c:/Users/jorge/Onel
ive/Documentos/JORGE/ESCOM/9_SEMESTRE/IA/LABORATORIO/4_BUSQUEDA_INFORMADA/laberinto2.py
1 P 1 1 0 0 0 0 1
1 0 0 0 1 0 1 1 0 1
1 1 1 0 1 0 1 0 0 1
1 0 0 0 0 0 1 1 0 1
1 0 1 1 1 1 1 0 0 1
1 0 0 0 1 0 0 0 1 1
1 1 1 0 1 1 1 0 1 1
1 0 1 0 0 0 1 0 0 1
1 0 1 1 1 0 1 1 0 1
1 1 1 1 1 1 1 1 0 1
```


1 P 1 1 1 0 0 0 0 1
1 P 0 0 1 0 1 1 0 1
1 1 1 0 1 0 1 0 0 1
1 0 0 0 0 0 1 1 0 1
1 0 1 1 1 1 1 0 0 1
1 0 0 0 1 0 0 0 1 1
1 1 1 0 1 1 1 0 1 1
1 0 1 0 0 0 1 0 0 1
1 0 1 1 1 0 1 1 0 1
1 1 1 1 1 1 1 1 0 1
1 P 1 1 1 0 0 0 0 1
1 P P 0 1 0 1 1 0 1
1 1 1 0 1 0 1 0 0 1
1 0 0 0 0 0 1 1 0 1
1 0 1 1 1 1 1 0 0 1
1 0 0 0 1 0 0 0 1 1
1 1 1 0 1 1 1 0 1 1
1 0 1 0 0 0 1 0 0 1
1 0 1 1 1 0 1 1 0 1
1 1 1 1 1 1 1 1 0 1
1 P 1 1 1 0 0 0 0 1
1 P P P 1 0 1 1 0 1
1 1 1 0 1 0 1 0 0 1
1 0 0 0 0 0 1 1 0 1
1 0 1 1 1 1 1 0 0 1
1 0 0 0 1 0 0 0 1 1
1 1 1 0 1 1 1 0 1 1
1 0 1 0 0 0 1 0 0 1
1 0 1 1 1 0 1 1 0 1
1 1 1 1 1 1 1 1 0 1


1 P 1 1 1 0 0 0 0 1
1 P P P 1 0 1 1 0 1
1 1 1 P 1 0 1 0 0 1
1 0 0 0 0 0 1 1 0 1
1 0 1 1 1 1 1 0 0 1
1 0 0 0 1 0 0 0 1 1
1 1 1 0 1 1 1 0 1 1
1 0 1 0 0 0 1 0 0 1
1 0 1 1 1 0 1 1 0 1
1 1 1 1 1 1 1 1 0 1
1 P 1 1 1 0 0 0 0 1
1 P P P 1 0 1 1 0 1
1 1 1 P 1 0 1 0 0 1
1 0 0 P 0 0 1 1 0 1
1 0 1 1 1 1 1 0 0 1
1 0 0 0 1 0 0 0 1 1
1 1 1 0 1 1 1 0 1 1
1 0 1 0 0 0 1 0 0 1
1 0 1 1 1 0 1 1 0 1
1 1 1 1 1 1 1 1 0 1
1 P 1 1 1 0 0 0 0 1
1 P P P 1 0 1 1 0 1
1 1 1 P 1 0 1 0 0 1
1 0 0 P P 0 1 1 0 1
1 0 1 1 1 1 1 0 0 1
1 0 0 0 1 0 0 0 1 1
1 1 1 0 1 1 1 0 1 1
1 0 1 0 0 0 1 0 0 1
1 0 1 1 1 0 1 1 0 1
1 1 1 1 1 1 1 1 0 1


1 P 1 1 1 0 0 0 0 1
1 P P P 1 0 1 1 0 1
1 1 1 P 1 0 1 0 0 1
1 0 0 P P P 1 1 0 1
1 0 1 1 1 1 1 0 0 1
1 0 0 0 1 0 0 0 1 1
1 1 1 0 1 1 1 0 1 1
1 0 1 0 0 0 1 0 0 1
1 0 1 1 1 0 1 1 0 1
1 1 1 1 1 1 1 1 0 1
1 P 1 1 1 0 0 0 0 1
1 P P P 1 0 1 1 0 1
1 1 1 P 1 0 1 0 0 1
1 0 P P 0 0 1 1 0 1
1 0 1 1 1 1 1 0 0 1
1 0 0 0 1 0 0 0 1 1
1 1 1 0 1 1 1 0 1 1
1 0 1 0 0 0 1 0 0 1
1 0 1 1 1 0 1 1 0 1
1 1 1 1 1 1 1 1 0 1
1 P 1 1 1 0 0 0 0 1
1 P P P 1 0 1 1 0 1
1 1 1 P 1 P 1 0 0 1
1 0 0 P P P 1 1 0 1
1 0 1 1 1 1 1 0 0 1
1 0 0 0 1 0 0 0 1 1
1 1 1 0 1 1 1 0 1 1
1 0 1 0 0 0 1 0 0 1
1 0 1 1 1 0 1 1 0 1
1 1 1 1 1 1 1 1 0 1


1 P 1 1 1 0 0 0 0 1
1 P P P 1 0 1 1 0 1
1 1 1 P 1 0 1 0 0 1
1 P P P 0 0 1 1 0 1
1 0 1 1 1 1 1 0 0 1
1 0 0 0 1 0 0 0 1 1
1 1 1 0 1 1 1 0 1 1
1 0 1 0 0 0 1 0 0 1
1 0 1 1 1 0 1 1 0 1
1 1 1 1 1 1 1 1 0 1
1 P 1 1 1 0 0 0 0 1
1 P P P 1 P 1 1 0 1
1 1 1 P 1 P 1 0 0 1
1 0 0 P P P 1 1 0 1
1 0 1 1 1 1 1 0 0 1
1 0 0 0 1 0 0 0 1 1
1 1 1 0 1 1 1 0 1 1
1 0 1 0 0 0 1 0 0 1
1 0 1 1 1 0 1 1 0 1
1 1 1 1 1 1 1 1 0 1
1 P 1 1 1 0 0 0 0 1
1 P P P 1 0 1 1 0 1
1 1 1 P 1 0 1 0 0 1
1 P P P 0 0 1 1 0 1
1 P 1 1 1 1 1 0 0 1
1 0 0 0 1 0 0 0 1 1
1 1 1 0 1 1 1 0 1 1
1 0 1 0 0 0 1 0 0 1
1 0 1 1 1 0 1 1 0 1
1 1 1 1 1 1 1 1 0 1


1 P 1 1 1 0 0 0 0 1
1 P P P 1 0 1 1 0 1
1 1 1 P 1 0 1 0 0 1
1 P P P 0 0 1 1 0 1
1 P 1 1 1 1 1 0 0 1
1 P 0 0 1 0 0 0 1 1
1 1 1 0 1 1 1 0 1 1
1 0 1 0 0 0 1 0 0 1
1 0 1 1 1 0 1 1 0 1
1 1 1 1 1 1 1 1 0 1
1 P 1 1 1 0 0 0 0 1
1 P P P 1 0 1 1 0 1
1 1 1 P 1 0 1 0 0 1
1 P P P 0 0 1 1 0 1
1 P 1 1 1 1 1 0 0 1
1 P P 0 1 0 0 0 1 1
1 1 1 0 1 1 1 0 1 1
1 0 1 0 0 0 1 0 0 1
1 0 1 1 1 0 1 1 0 1
1 1 1 1 1 1 1 1 0 1
1 P 1 1 1 0 0 0 0 1
1 P P P 1 0 1 1 0 1
1 1 1 P 1 0 1 0 0 1
1 P P P 0 0 1 1 0 1
1 P 1 1 1 1 1 0 0 1
1 P P P 1 0 0 0 1 1
1 1 1 0 1 1 1 0 1 1
1 0 1 0 0 0 1 0 0 1
1 0 1 1 1 0 1 1 0 1
1 1 1 1 1 1 1 1 0 1


1 P 1 1 1 0 0 0 0 1
1 P P P 1 0 1 1 0 1
1 1 1 P 1 0 1 0 0 1
1 P P P 0 0 1 1 0 1
1 P 1 1 1 1 1 0 0 1
1 P P P 1 0 0 0 1 1
1 1 1 P 1 1 1 0 1 1
1 0 1 0 0 0 1 0 0 1
1 0 1 1 1 0 1 1 0 1
1 1 1 1 1 1 1 1 0 1
1 P 1 1 1 0 0 0 0 1
1 P P P 1 0 1 1 0 1
1 1 1 P 1 0 1 0 0 1
1 P P P 0 0 1 1 0 1
1 P 1 1 1 1 1 0 0 1
1 P P P 1 0 0 0 1 1
1 1 1 P 1 1 1 0 1 1
1 0 1 P 0 0 1 0 0 1
1 0 1 1 1 0 1 1 0 1
1 1 1 1 1 1 1 1 0 1
1 P 1 1 1 0 0 0 0 1
1 P P P 1 0 1 1 0 1
1 1 1 P 1 0 1 0 0 1
1 P P P 0 0 1 1 0 1
1 P 1 1 1 1 1 0 0 1
1 P P P 1 0 0 0 1 1
1 1 1 P 1 1 1 0 1 1
1 0 1 P P 0 1 0 0 1
1 0 1 1 1 0 1 1 0 1
1 1 1 1 1 1 1 1 0 1

```
1 P 1 1 1 0 0 0 0 1
1 P P P 1 0 1 1 0 1
1 1 1 P 1 0 1 0 0 1
1 P P P 0 0 1 1 0 1
1 P 1 1 1 1 1 0 0 1
1 P P P 1 0 0 0 1 1
1 1 1 P 1 1 1 0 1 1
1 0 1 P P P 1 0 0 1
1 0 1 1 1 0 1 1 0 1
1 1 1 1 1 1 1 1 0 1

1 P 1 1 1 0 0 0 0 1
1 P P P 1 0 1 1 0 1
1 1 1 P 1 0 1 0 0 1
1 P P P 0 0 1 1 0 1
1 P 1 1 1 1 1 0 0 1
1 P P P 1 0 0 0 1 1
1 1 1 P 1 1 1 0 1 1
1 0 1 P P P 1 0 0 1
1 0 1 1 1 P 1 1 0 1
1 1 1 1 1 1 1 1 0 1

1 P 1 1 1 P 0 0 0 1
1 P P P 1 P 1 1 0 1
1 1 1 P 1 P 1 0 0 1
1 0 0 P P P 1 1 0 1
1 0 1 1 1 1 1 0 0 1
1 0 0 0 1 0 0 0 1 1
1 1 1 0 1 1 1 0 1 1
1 0 1 0 0 0 1 0 0 1
1 0 1 1 1 0 1 1 0 1
1 1 1 1 1 1 1 1 0 1
```

```
1 P 1 1 1 P P 0 0 1
1 P P P 1 P 1 1 0 1
1 1 1 P 1 P 1 0 0 1
1 0 0 P P P 1 1 0 1
1 0 1 1 1 1 1 0 0 1
1 0 0 0 1 0 0 0 1 1
1 1 1 0 1 1 1 0 1 1
1 0 1 0 0 0 1 0 0 1
1 0 1 1 1 0 1 1 0 1
1 1 1 1 1 1 1 1 0 1

1 P 1 1 1 P P P 0 1
1 P P P 1 P 1 1 0 1
1 1 1 P 1 P 1 0 0 1
1 0 0 P P P 1 1 0 1
1 0 1 1 1 1 1 0 0 1
1 0 0 0 1 0 0 0 1 1
1 1 1 0 1 1 1 0 1 1
1 0 1 0 0 0 1 0 0 1
1 0 1 1 1 0 1 1 0 1
1 1 1 1 1 1 1 1 0 1

1 P 1 1 1 P P P P 1
1 P P P 1 P 1 1 0 1
1 1 1 P 1 P 1 0 0 1
1 0 0 P P P 1 1 0 1
1 0 1 1 1 1 1 0 0 1
1 0 0 0 1 0 0 0 1 1
1 1 1 0 1 1 1 0 1 1
1 0 1 0 0 0 1 0 0 1
1 0 1 1 1 0 1 1 0 1
1 1 1 1 1 1 1 1 0 1
```

```
1 P 1 1 1 P P P P 1
1 P P P 1 P 1 1 P 1
1 1 1 P 1 P 1 0 0 1
1 0 0 P P P 1 1 0 1
1 0 1 1 1 1 1 0 0 1
1 0 0 0 1 0 0 0 1 1
1 1 1 0 1 1 1 0 1 1
1 0 1 0 0 0 1 0 0 1
1 0 1 1 1 0 1 1 0 1
1 1 1 1 1 1 1 1 0 1

1 P 1 1 1 P P P P 1
1 P P P 1 P 1 1 P 1
1 1 1 P 1 P 1 0 P 1
1 0 0 P P P 1 1 0 1
1 0 1 1 1 1 1 0 0 1
1 0 0 0 1 0 0 0 1 1
1 1 1 0 1 1 1 0 1 1
1 0 1 0 0 0 1 0 0 1
1 0 1 1 1 0 1 1 0 1
1 1 1 1 1 1 1 1 0 1

1 P 1 1 1 P P P P 1
1 P P P 1 P 1 1 P 1
1 1 1 P 1 P 1 0 P 1
1 0 0 P P P 1 1 P 1
1 0 1 1 1 1 1 0 0 1
1 0 0 0 1 0 0 0 1 1
1 1 1 0 1 1 1 0 1 1
1 0 1 0 0 0 1 0 0 1
1 0 1 1 1 0 1 1 0 1
1 1 1 1 1 1 1 1 0 1
```

```
1 P 1 1 1 P P P P 1
1 P P P 1 P 1 1 P 1
1 1 1 P 1 P 1 0 P 1
1 0 0 P P P 1 1 P 1
1 0 1 1 1 1 1 0 P 1
1 0 0 0 1 0 0 0 1 1
1 1 1 0 1 1 1 0 1 1
1 0 1 0 0 0 1 0 0 1
1 0 1 1 1 0 1 1 0 1
1 1 1 1 1 1 1 1 0 1

1 P 1 1 1 P P P P 1
1 P P P 1 P 1 1 P 1
1 1 1 P 1 P 1 P P 1
1 0 0 P P P 1 1 0 1
1 0 1 1 1 1 1 0 0 1
1 0 0 0 1 0 0 0 1 1
1 1 1 0 1 1 1 0 1 1
1 0 1 0 0 0 1 0 0 1
1 0 1 1 1 0 1 1 0 1
1 1 1 1 1 1 1 1 0 1

1 P 1 1 1 P P P P 1
1 P P P 1 P 1 1 P 1
1 1 1 P 1 P 1 0 P 1
1 0 0 P P P 1 1 P 1
1 0 1 1 1 1 1 P P 1
1 0 0 0 1 0 0 0 1 1
1 1 1 0 1 1 1 0 1 1
1 0 1 0 0 0 1 0 0 1
1 0 1 1 1 0 1 1 0 1
1 1 1 1 1 1 1 1 0 1
```

```
1 P 1 1 1 P P P P 1
1 P P P 1 P 1 1 P 1
1 1 1 P 1 P 1 0 P 1
1 0 0 P P P 1 1 P 1
1 0 1 1 1 1 1 P P 1
1 0 0 0 1 0 0 P 1 1
1 1 1 0 1 1 1 0 1 1
1 0 1 0 0 0 1 0 0 1
1 0 1 1 1 0 1 1 0 1
1 1 1 1 1 1 1 1 0 1

1 P 1 1 1 P P P P 1
1 P P P 1 P 1 1 P 1
1 1 1 P 1 P 1 0 P 1
1 0 0 P P P 1 1 P 1
1 0 1 1 1 1 1 P P 1
1 0 0 0 1 0 0 P 1 1
1 1 1 0 1 1 1 P 1 1
1 0 1 0 0 0 1 0 0 1
1 0 1 1 1 0 1 1 0 1
1 1 1 1 1 1 1 1 0 1

1 P 1 1 1 P P P P 1
1 P P P 1 P 1 1 P 1
1 1 1 P 1 P 1 0 P 1
1 0 0 P P P 1 1 P 1
1 0 1 1 1 1 1 P P 1
1 0 0 0 1 0 0 P 1 1
1 1 1 0 1 1 1 P 1 1
1 0 1 0 0 0 1 P 0 1
1 0 1 1 1 0 1 1 0 1
1 1 1 1 1 1 1 1 0 1
```

```
1 P 1 1 1 P P P P 1
1 P P P 1 P 1 1 P 1
1 1 1 P 1 P 1 0 P 1
1 0 0 P P P 1 1 P 1
1 0 1 1 1 1 1 P P 1
1 0 0 0 1 0 0 P 1 1
1 1 1 0 1 1 1 P 1 1
1 0 1 0 0 0 1 P P 1
1 0 1 1 1 0 1 1 0 1
1 1 1 1 1 1 1 1 0 1

1 P 1 1 1 P P P P 1
1 P P P 1 P 1 1 P 1
1 1 1 P 1 P 1 0 P 1
1 0 0 P P P 1 1 P 1
1 0 1 1 1 1 1 P P 1
1 0 0 0 1 0 0 P 1 1
1 1 1 0 1 1 1 P 1 1
1 0 1 0 0 0 1 P P 1
1 0 1 1 1 0 1 1 P 1
1 1 1 1 1 1 1 1 0 1

1 P 1 1 1 P P P P 1
1 P P P 1 P 1 1 P 1
1 1 1 P 1 P 1 0 P 1
1 0 0 P P P 1 1 P 1
1 0 1 1 1 1 1 P P 1
1 0 0 0 1 0 0 P 1 1
1 1 1 0 1 1 1 P 1 1
1 0 1 0 0 0 1 P P 1
1 0 1 1 1 0 1 1 P 1
1 1 1 1 1 1 1 1 P 1
```

```
¡Laberinto resuelto!
Se encontró un camino. El laberinto resuelto es:
1 P 1 1 1 P P P P 1
1 P P P 1 P 1 1 P 1
1 1 1 P 1 P 1 0 P 1
1 0 0 P P P 1 1 P 1
1 0 1 1 1 1 1 P P 1
1 0 0 0 1 0 0 P 1 1
1 1 1 0 1 1 1 P 1 1
1 0 1 0 0 0 1 P P 1
1 0 1 1 1 0 1 1 P 1
1 1 1 1 1 1 1 1 P 1




El tiempo de ejecución fue:  0.0865175724029541
PS C:\Users\jorge\OneDrive\Documentos\JORGE\ESCOM\9_SEMESTRE>
```

ENLACE

A continuación, dejamos el repositorio completo, la practica se encuentra dentro de la carpeta LAB_4_BUSQUEDA_INFORMADA y a continuación dejo el enlace: https://github.com/Jorge300403/IA_6CV3_MartinezChavez/tree/main/LAB_4_BUSQUEDA_INFORMADA