Introducción a la Inteligencia Artificial: El papel de la heurística

Ignacio Alejandro Ordaz Estrada

N.C. 17120188

-> ¿Qué es la heurística y cual es su papel en la resolución de problemas?

Practicamente se puede definir una heurística como: *Caminos cortos y eficientes que tu cerebro genera ante cierta resolución de problemas, los cuales son generados en base a lo ya aprendido anteriormente*.

Esto no significa que sea la mejor opción siempre, pueden darse casos en los que pensamos que es lo mejor pero en algun punto genera un problema y esa heurística no fue la mejor opción.

La importancia de dichas heurísticas se basa en que al ir tratando de resolver ciertos problemas nuevos los asociamos con algo ya conocido y el saber que si funciona y que no, hace mas corto el tiempo para encontrar la solución al nuevo problema. Como ejemplo se puede usar una division simple de 12/3.

- Paso 1 : recordamos que 3 * 2 = 6 y queda lejos de ser un resultado, por lo tanto no vamos a comenzar los intentos con un 2.
- Paso 2: 3 * 3 = 9, misma situación.
- Paso 3: 3 * 4 = 12, esta es nuestra solución.

... que pasa aquí?, ya sabíamos previamente las tablas de multiplicar y aplicamos un camino con heuristica sabiendo que el 4 era su cociente y no se tuvo que hacer de número por número.

De esta misma manera aplica en los problemas que decidas plantearte a resolver, se utiliza lo aprendido previamente para hacer una solución más rápida, lo cual reduce nuestro trabajo.

-> Resolver con recursividad.

Para ver la solución del problema visitar el link que contiene un repositorio de github el alumno **Ignacio Alejandro Ordaz Estrada** IA_Laberinto_Ignacio-Ordaz

-> Proponer algoritmo de solución. Aquí solo será superficial en el siguiente punto se explica a fondo, por practicidad se hara copy paste de partes del codigo.

- Declarar matriz laberinto con sus respectivos muros, inicio y final

```
laberinto = [
    ['1','1','1','1','1','1','1','1','1'],
    ['O',' ',' ',' ',' ',' ','1',' ','1'],
    ['1','1','1',' ','1','1','1',' ','1'],
    ['1',' ',' ',' ','1',' ','1',' ','1'],
    ['1',' ','1','1','1',' ','1',' ','1'],
    ['1',' ',' ',' ',' ',' ',' ',' ','1'],
    ['1',' ','1','1','1',' ','1',' ','1'],
    ['X',' ','1',' ',' ',' ','1',' ','1'],
    ['1','1','1','1','1','1','1','1','1']
]
```

- Declarar 2 variables para la posicion incial, fila y columna.

```
fila=1
columna=0
```

- Crear 2 funciones mostrar_laberinto y resolver
    - Mostrar laberinto tendrá 2 condiciones para imprimir el laberinto inicial o el resuelto.

```
def mostrar_laberinto(laberinto):

    if(m==0):
        print("Laberinto inicial")
        for fila in laberinto:
            for columna in fila:
                print(columna, end=' ')
            print(" ")

    elif(m==1):
        print("Laberinto final")
        for fila in laberinto:
            for columna in fila:
                print(columna, end=' ')
            print(" ")
```

- Resolver es la función primordial del programa

```
def resolver(laberinto, fila, columna):
    if laberinto[fila][columna] == 'X':
        laberinto[fila][columna]='-'
        return True

    if laberinto[fila][columna] == '1' or laberinto[fila][columna] ==
'v':
        return False

    laberinto[fila][columna] = 'v'

    if ((fila > 0 and resolver(laberinto, fila - 1, columna)) or
        (columna < len(laberinto[fila]) - 1 and resolver(laberinto,
fila, columna + 1)) or
        (fila < len(laberinto) - 1 and resolver(laberinto, fila + 1,
columna)) or
        (columna > 0 and resolver(laberinto, fila, columna - 1))):

        laberinto[fila][columna]='-'
        return True

    laberinto[fila][columna] = ' '

    return False
```

- Llamar a las funciones de acuerdo a lo que se necesita y enviar variables necesarias en la llamada de cada una. entre estas modificar el contador para que imprima el laberinto resuelto

```
  mostrar_laberinto(laberinto)

  resolver(laberinto,fila,columna)
  m=1

  mostrar_laberinto(laberinto)
```

-> Describir punto anterior

Creo que lo único que necesita una eplicación mas a fondo es la parte de resolver así que aqui la documentare:

- Resolver es la función primordial del programa

  def resolver(laberinto, fila, columna):

```
   # si estamos en el inicio agregamos el caracter '-' a la matriz y
devolvemos true
   if laberinto[fila][columna] == 'X':
       laberinto[fila][columna]='-'
       return True

   # Si la posición actual es pared o ya pasamos por ahi, devolvemos
False
   if laberinto[fila][columna] == '1' or laberinto[fila][columna] ==
'v':
       return False

   # cambiamos el valor por una 'v' de visitado
   laberinto[fila][columna] = 'v'

   # comprobar si podemos avanzar hacaia algun punto (arriba, derecha,
abajo, izquierda)
   if ((fila > 0 and resolver(laberinto, fila - 1, columna)) or
       (columna < len(laberinto[fila]) - 1 and resolver(laberinto,
fila, columna + 1)) or
       (fila < len(laberinto) - 1 and resolver(laberinto, fila + 1,
columna)) or
       (columna > 0 and resolver(laberinto, fila, columna - 1))):
       # Si pudimos avanzar modificamos con el carcater '-' de que por
ahi va el camino
       laberinto[fila][columna]='-'
       return True

   # Si no se puede avanzar la dejamos vacia
   laberinto[fila][columna] = ' '

   return False
```