



# Inteligencia Artificial

Unidad 2

Practica 1

Prof. ISC. Stephanie  
Cordero Martínez

Herrera Mireles Flor Lucero

14/12/2023

## INDICE:

<b>1 CODIGO:</b> .....	3
<b>2. RESULTADO:</b> .....	5
<b>3. CONCLUSION:</b> .....	6

## 1 CODIGO:

```
Puzzle Practica.py X
C: > Users > lucero herrera m > Downloads > Puzzle Practica.py > ...

1  from collections import deque
2
3  # Clase que define un nodo en el 8-puzzle.
4  class Nodo:
5      def __init__(self, estado, padre, movimiento, profundidad, piezas_correctas):
6          self.estado = estado
7          self.padre = padre
8          self.movimiento = movimiento
9          self.profundidad = profundidad
10         self.piezas_correctas = piezas_correctas
11
12     def mover(self, direccion):
13         nuevo_estado = list(self.estado)
14         ind = nuevo_estado.index(0)
15         if direccion == "arriba" and ind not in [6, 7, 8]:
16             nuevo_estado[ind], nuevo_estado[ind + 3] = nuevo_estado[ind + 3], nuevo_estado[ind]
17         elif direccion == "abajo" and ind not in [0, 1, 2]:
18             nuevo_estado[ind], nuevo_estado[ind - 3] = nuevo_estado[ind - 3], nuevo_estado[ind]
19         elif direccion == "derecha" and ind not in [0, 3, 6]:
20             nuevo_estado[ind], nuevo_estado[ind - 1] = nuevo_estado[ind - 1], nuevo_estado[ind]
21         elif direccion == "izquierda" and ind not in [2, 5, 8]:
22             nuevo_estado[ind], nuevo_estado[ind + 1] = nuevo_estado[ind + 1], nuevo_estado[ind]
23         else:
24             return None
25         return tuple(nuevo_estado)
26
27     def encontrar_sucesores(self):
28         sucesores = []
29         movimientos = ["arriba", "abajo", "derecha", "izquierda"]
30         for movimiento in movimientos:
31             sucesor = self.mover(movimiento)
32             if sucesor:
```

```
Puzzle Practica.py X
C:\Users\lucero herrera m\Downloads> Puzzle Practica.py > $ Nodo > _init_

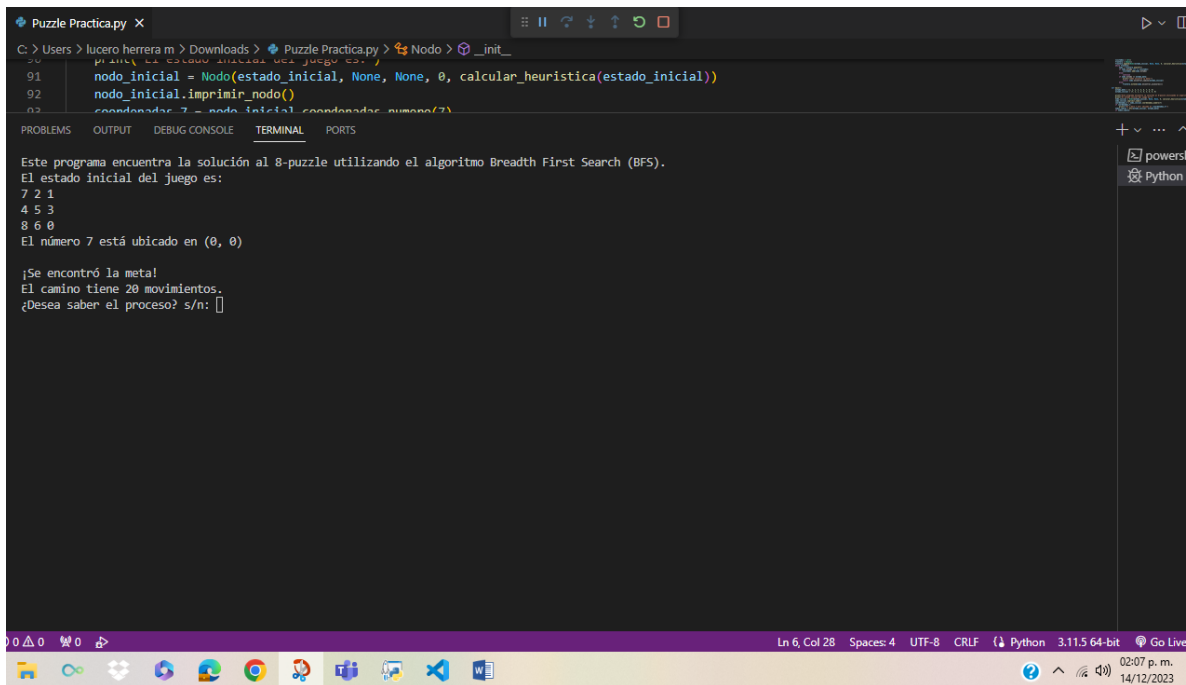
32         if sucesor:
33             sucesores.append(Nodo(sucesor, self, movimiento, self.profundidad + 1, calcular_heuristica(sucesor)))
34         return sucesores
35
36     def encontrar_camino(self, estado_inicial):
37         camino = []
38         nodo_actual = self
39         while nodo_actual.profundidad >= 1:
40             camino.append(nodo_actual)
41             nodo_actual = nodo_actual.padre
42         camino.reverse()
43         return camino
44
45     def imprimir_nodo(self):
46         renglon = 0
47         for i, pieza in enumerate(self.estado):
48             print(pieza, end=" ")
49             renglon += 1
50             if renglon == 3:
51                 print()
52                 renglon = 0
53
54     def coordenadas_numero(self, numero):
55         estado = list(self.estado)
56         if numero in estado:
57             indice = estado.index(numero)
58             fila = indice // 3
59             columna = indice % 3
60             return fila, columna
61         return None
62
63     def calcular_heuristica(estado):
```

```
Puzzle Practica.py X
C:\Users\lucero herrera m\Downloads> Puzzle Practica.py > $ Nodo > _init_

63     def calcular_heuristica(estado):
64         estado_correcto = (1, 2, 3, 4, 5, 6, 7, 8, 0)
65         piezas_correctas = sum(1 for a, b in zip(estado, estado_correcto) if a == b)
66         return piezas_correctas
67
68     # Algoritmo Breadth First Search.
69     def bfs(estado_inicial, estado_meta):
70         visitados = set()
71         frontera = deque()
72         frontera.append(Nodo(estado_inicial, None, None, 0, calcular_heuristica(estado_inicial)))
73         while frontera:
74             nodo = frontera.popleft()
75             if nodo.estado not in visitados:
76                 visitados.add(nodo.estado)
77             else:
78                 continue
79             if nodo.estado == estado_meta:
80                 print("\nSe encontró la meta!")
81                 return nodo.encontrar_camino(estado_inicial)
82             else:
83                 frontera.extend(nodo.encontrar_sucesores())
84
85     def main():
86         estado_meta = (1, 2, 3, 4, 5, 6, 7, 8, 0)
87         estado_inicial = (7, 2, 1, 4, 5, 3, 8, 6, 0)
88
89         print("Este programa encuentra la solución al 8-puzzle utilizando el algoritmo Breadth First Search (BFS).")
90         print("El estado inicial del juego es:")
91         nodo_inicial = Nodo(estado_inicial, None, None, 0, calcular_heuristica(estado_inicial))
92         nodo_inicial.imprimir_nodo()
93         coordenadas_7 = nodo_inicial.coordenadas_numero(7)
94         if coordenadas_7:
```

```
Puzzle Practica.py X
C:\Users\lucero herrera m\Downloads> Puzzle Practica.py > Nodo > _init_
91 print("¿El camino encontrado es el correcto?")
92 nodo_inicial = Nodo(estado_inicial, None, None, 0, calcular_heuristica(estado_inicial))
93 nodo_inicial.imprimir_nodo()
94 coordenadas_7 = nodo_inicial.coordenadas_numero(7)
95 if coordenadas_7:
96     print(f"El número 7 está ubicado en {coordenadas_7}")
97     nodos_camino = bfs(estado_inicial, estado_meta)
98     if nodos_camino:
99         print("El camino tiene", len(nodos_camino), "movimientos.")
100         imprimir_camino = input("¿Desea saber el proceso? s/n: ")
101
102         if imprimir_camino.lower() == "s":
103             print("\nEstado inicial:")
104             nodo_inicial.imprimir_nodo()
105             print("Piezas correctas:", calcular_heuristica(estado_inicial), "\n")
106             input("Presione \"enter\" para continuar.")
107
108             for nodo in nodos_camino:
109                 print("\nSiguiente movimiento:", nodo.movimiento)
110                 print("Estado actual:")
111                 nodo.imprimir_nodo()
112                 coordenadas_7 = nodo.coordenadas_numero(7)
113                 if coordenadas_7:
114                     print(f"El número 7 está ubicado en {coordenadas_7}")
115                     print("Piezas correctas en puzzle:", nodo.piezas_correctas, "\n")
116                     input("Presione \"enter\" para continuar.")
117             else:
118                 print("No hay un camino con esas condiciones.")
119
120 if __name__ == "__main__":
121     main()
```

## 2. RESULTADO:



```
Puzzle Practica.py X
C:\Users\lucero herrera m > Downloads > Puzzle Practica.py > Nodo > _init_
90 print("El estado inicial del juego es:")
91 nodo_inicial = Nodo(estado_inicial, None, None, 0, calcular_heuristica(estado_inicial))
92 nodo_inicial.imprimir_nodo()
93 coordenadas_7 = nodo_inicial.coordenadas_numero(7)

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
Este programa encuentra la solución al 8-puzzle utilizando el algoritmo Breadth First Search (BFS).
El estado inicial del juego es:
7 2 1
4 5 3
8 6 0
El número 7 está ubicado en (0, 0)

¡Se encontró la meta!
El camino tiene 20 movimientos.
¿Desea saber el proceso? s/n: []
```

### 3. CONCLUSION:

Durante este programa tuvimos que instalar Python para que el programa nos funcionara también tuvimos que entender la lógica de los números y los movimientos que se podían realizar como de izquierda a derecha y que nos mostrara la matriz

link

[https://github.com/LuceroHerreraM/Int\\_Art/blob/main/Unit%202/Puzzle%20Practica.py](https://github.com/LuceroHerreraM/Int_Art/blob/main/Unit%202/Puzzle%20Practica.py)