



TECNOLÓGICO  
NACIONAL DE MÉXICO



INSTITUTO TECNOLÓGICO DE CULIACÁN

ING. SISTEMAS COMPUTACIONALES

INTELIGENCIA ARTIFICIAL (11:00 A 12:00)

EVALUACIÓN UNIDAD 1

PUZZLE 8 EXPLICACIÓN DE CÓDIGO Y FUNCIONAMIENTO

JIMENEZ VELAZQUEZ ZETH ODIN ALFONSO

BRYAN JAVIER ANGULO SANDOVAL

---

## Funcionamiento paso a paso

### Se inicializa la búsqueda:

```
queue = Queue()
```

```
queue.put((self.state, []))
```

```
visited = set()
```

- **queue**: Cola FIFO (First In, First Out) que almacena los estados del tablero y la secuencia de movimientos realizados para llegar a ellos.

- **visited:** Conjunto para registrar los estados ya explorados y evitar ciclos innecesarios.

#### **Se exploran los estados mientras haya elementos en la cola:**

```
while not queue.empty():
```

```
    current, path = queue.get()
```

- Se extrae el primer estado almacenado en la cola (current) y su lista de movimientos (path).

#### **Se verifica si el estado es la solución:**

```
if current == GOAL_STATE:
```

```
    self.animate_solution(path)
```

```
    return
```

- Si el estado coincide con el objetivo (GOAL\_STATE = "123456780"), la solución se anima paso a paso.

#### **Se marcan los estados visitados y se generan nuevos movimientos:**

```
visited.add(current)
```

```
index = current.index("0") # Posición del espacio vacío
```

```
for direction, move in MOVES.items():
```

```
    new_index = index + move
```

- Se marca el estado como visitado.
- Se obtiene la posición del 0 (el espacio vacío) para calcular los movimientos posibles.

#### **Se generan los nuevos estados posibles y se agregan a la cola:**

```
if 0 <= new_index < 9 and not (index % 3 == 2 and direction == "Right") and not (index % 3 == 0 and direction == "Left"):
```

```
    new_state = list(current)
```

```
    new_state[index], new_state[new_index] = new_state[new_index], new_state[index]
```

```
    new_state = "".join(new_state)
```

```
    if new_state not in visited:
```

```
queue.put((new_state, path + [direction]))
```

- Se valida que el nuevo índice esté dentro del rango y no cruce los bordes del tablero.
- Se intercambia el 0 con la ficha correspondiente para generar un nuevo estado.
- Si el nuevo estado no ha sido visitado, se agrega a la cola junto con la lista de movimientos actualizada.

---

### **Animación de la solución**

Si se encuentra la solución, la IA llama a `animate_solution(path)`, que reproduce los movimientos encontrados en `path` cada 500ms:

```
def animate_solution(self, path):
```

```
    if not path:
```

```
        return
```

```
    self.move(path.pop(0))
```

```
    self.root.after(500, lambda: self.animate_solution(path))
```

- Se ejecuta un movimiento del camino encontrado y se programa el siguiente.

---

### **Capturas**



### Estado inicial

**Botón Reiniciar:** reinicia el estado actual y lo genera de manera aleatoria con la función Random.

**Boton Jugar Manual:** activa las flechas del teclado para mover el espacio vacio a voluntad con las flechas del teclado.

**Boton Resolver con IA:** Resuelve el pluzzle automáticamente mostrando paso por paso las piezas que se mueven hasta llegar al estado final.



**Estado final.**