Tecnológico Nacional de México / Instituto Tecnológico de Culiacán.

Ingeniería en Sistemas Computacionales.

Inteligencia Artificial

Zuriel Dathan Mora Felix.

Modulo 1

Documentación del Código Puzzle-8.

Ontiveros Sánchez Jesús Daniel (22170750).

Ramírez Ruiz Santiago (22170783).

Horario de 9:00 a 10:00 AM.

Culiacán Sinaloa a 10 de septiembre del 2025

Explicación del código – Puzzle-8 (BFS)

Documento técnico breve que describe la estructura general del programa, el propósito de cada componente y el funcionamiento del algoritmo de búsqueda.

1) Propósito del programa

El programa resuelve el Puzzle-8 (tablero 3×3 con fichas del 1 al 8 y un hueco representado por 0) empleando Búsqueda en Anchura (BFS). Dado un estado inicial válido, encuentra si existe una secuencia de movimientos mínima para llegar al estado objetivo (1,2,3,4,5,6,7,8,0).

2) Representación y estructuras de datos

- Tablero / Estado: matriz de 9 enteros; 0 es el hueco.
- ESTADO_ACEPTACION: constante con el estado objetivo.
- deque: cola FIFO usada por BFS para los estados por explorar.
- explorados: conjunto de estados ya visitados, evita trabajo duplicado.
- viene_de: diccionario hijo → padre, para reconstruir el camino solución.

3) Funciones del código

- **imprimir_puzzle(estado)**

Muestra el tablero en formato 3×3, sustituyendo 0 por un espacio. Sirve para depurar y para presentar la solución paso a paso.

- **obtener_vecinos(estado)**

Genera todos los estados alcanzables en un movimiento desplazando el 0 en direcciones válidas (arriba, abajo, izquierda, derecha). Pasos:

1. Localiza el índice del 0 y lo convierte a coordenadas (fila, col).

2. Determina movimientos permitidos según bordes del tablero.

3. Intercambia 0 con la casilla destino para construir cada nuevo estado.

- **reconstruir_camino(viene_de, actual)**

Recorre hacia atrás el viene_de desde actual (objetivo) hasta el inicial, acumulando estados. Luego invierte la lista para obtener el camino en orden correcto.

- bfs(estado_inicial)

Se implementa la Búsqueda en Anchura:

1. Inicializa frontera con el estado_inicial y vacía explorados / viene_de.

2. Mientras haya elementos en frontera:

    - Extrae el primer estado (FIFO).
    - Si es el objetivo, reconstruye y devuelve el camino.
    - Si no, lo añade a explorados y expande vecinos con obtener_vecinos.
    - Encola cada vecino no visto y registra viene_de[vecino] = actual.

3. Si se agota la frontera, no hay solución (devuelve None).

- **mostrar_solucion(camino)**

Presenta el número de movimientos y cada paso del tablero usando imprimir_puzzle. Si camino es None, indica que no existe solución.

- **main()**

Interfaz de línea de comandos: valida la entrada del usuario (valido de 0..8), muestra el estado inicial, ejecuta bfs y llama a mostrar_solucion.

4) Funcionamiento del algoritmo (BFS)

BFS explora el espacio de estados por niveles (distancia en movimientos desde el inicial). Esto garantiza que el primer encuentro con el objetivo corresponde a una ruta de longitud mínima.

- **Frontera (cola FIFO):** contiene los estados descubiertos y pendientes de expansión.
- **Explorados:** evita re-visitar estados ya procesados.
- **Predecesores (viene_de):** permite backtracking eficiente para reconstruir la solución sin almacenar rutas completas en la cola.

Correcto y optimo:

- Correcto: si el objetivo es alcanzable, BFS lo encontrará porque explora sistemáticamente.
- Óptimo (en pasos): al expandir por niveles, el primer objetivo hallado usa el menor número de movimientos.

5) Desarrollo de ejecución resumido

1. **main()** solicita y valida la entrada.

2. **imprimir_puzzle()** muestra el estado inicial.

3. **bfs()** busca la solución:

   - usa **obtener_vecinos()** para expandir
   - marca **explorados**
   - registra padres en **viene_de**

4. **reconstruir_camino()** arma la ruta cuando se encuentra el objetivo.

5. **mostrar_solucion()** imprime pasos y conteo de movimientos.

6) Ejemplo corto

Entrada: 1 2 3 4 5 6 0 7 8

Salida:

Se encontraron 2 movimientos.

Paso 0:

1 2 3

4 5 6

  7 8

Paso 1:

1 2 3

4 5 6

7 8

Paso 2:

1 2 3

4 5 6

7 8