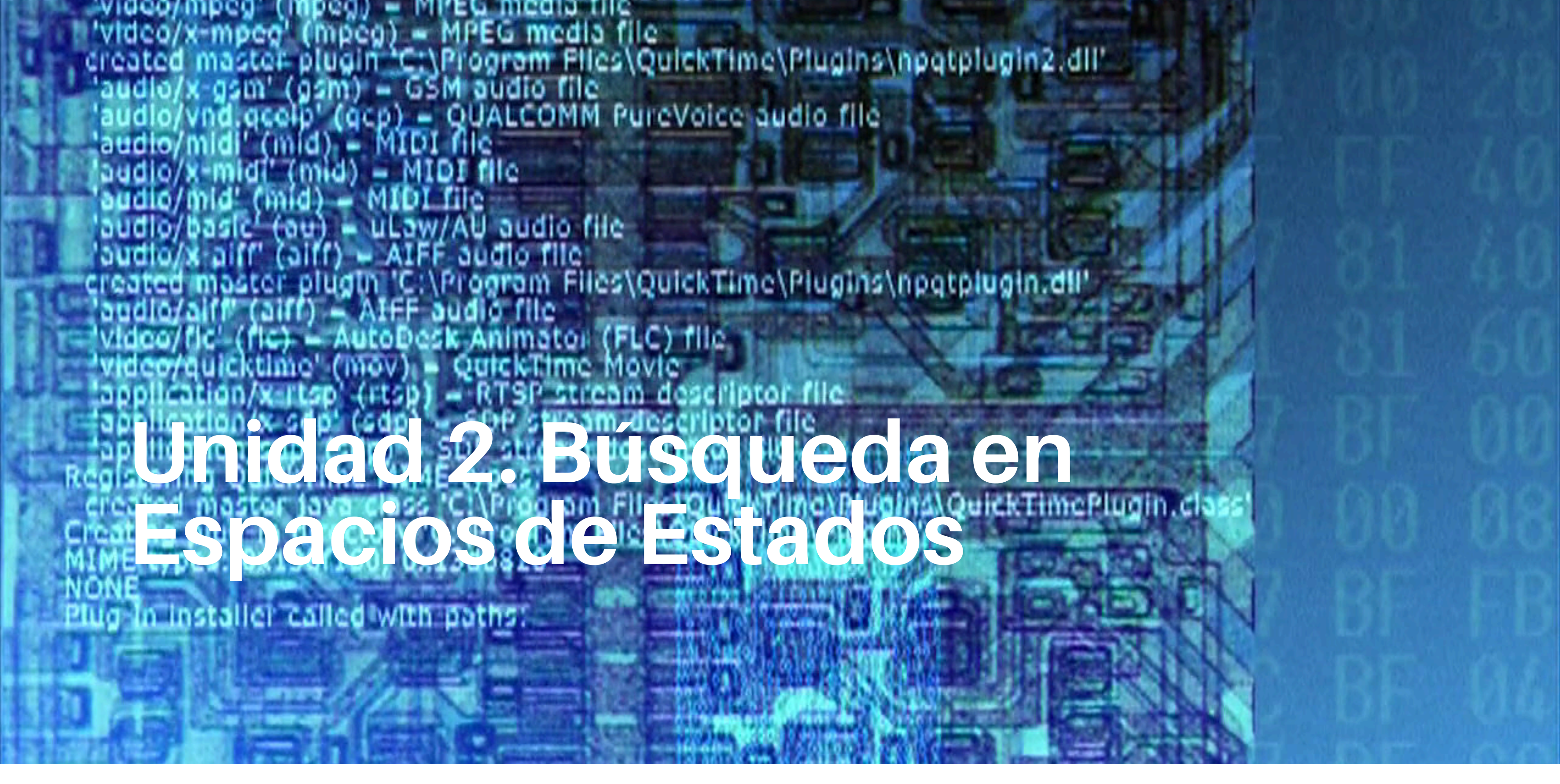


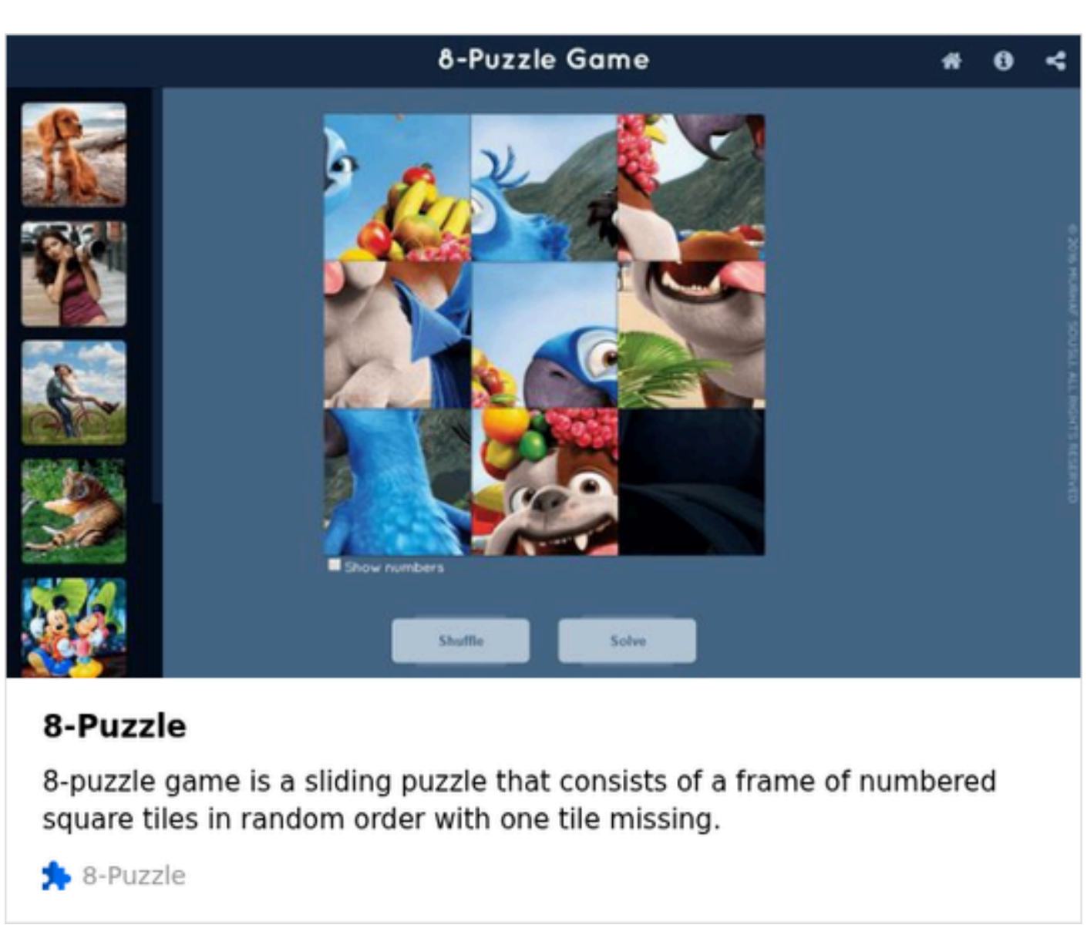
Unidad 2. Búsqueda en Espacios de Estados

¿Qué es un espacio de estados?

Un espacio de estados es una representación matemática de un problema donde cada estado es una configuración posible del sistema, y las acciones permiten moverse entre estados.

Componentes clave de un espacio de estados:

1. **Estados:** Representación de cada posible situación del problema.
2. **Estado inicial:** Punto de partida del sistema.
3. **Estado meta:** Objetivo o solución del problema.
4. **Operadores (acciones):** Transformaciones que llevan de un estado a otro.
5. **Camino de solución:** Secuencia de acciones desde el estado inicial hasta el estado meta.



8-Puzzle

8-puzzle game is a sliding puzzle that consists of a frame of numbered square tiles in random order with one tile missing.

8-Puzzle

8-puzzle es un ejemplo de un problema de *espacio de estados*.

Ejemplos Clásicos de Problemas Modelados como Espacios de Estados

Problema	Estados Posibles	Operadores (Acciones)	Estado Meta
Problema del 8-puzzle	Distribuciones de fichas en una cuadrícula 3x3	Mover una ficha vacía en 4 direcciones	Tablero ordenado
Juego del Ajedrez	Posiciones de las piezas en el tablero	Movimiento permitido de cada pieza	Jaque mate
Búsqueda de rutas	Localización en un mapa	Moverse entre ciudades conectadas	Llegar al destino

Representación Formal de un Espacio de Estados

Un espacio de estados se modela matemáticamente como un **grafo dirigido**, donde:

- Los nodos representan los estados.
- Las aristas representan las acciones que permiten cambiar de un estado a otro.

Ejemplo en **notación formal**:

- **Estados:** $S = \{s_0, s_1, s_2, \ldots, s_n\}$
- **Operadores:** $O = a_1, a_2, \ldots, a_m$
- **Función de transición:** $f(s,a) \rightarrow s\prime$
- **Estado meta:** $S_{goal} \subset S$

*El estado *s′* resulta de aplicar la acción *a* en *s*.

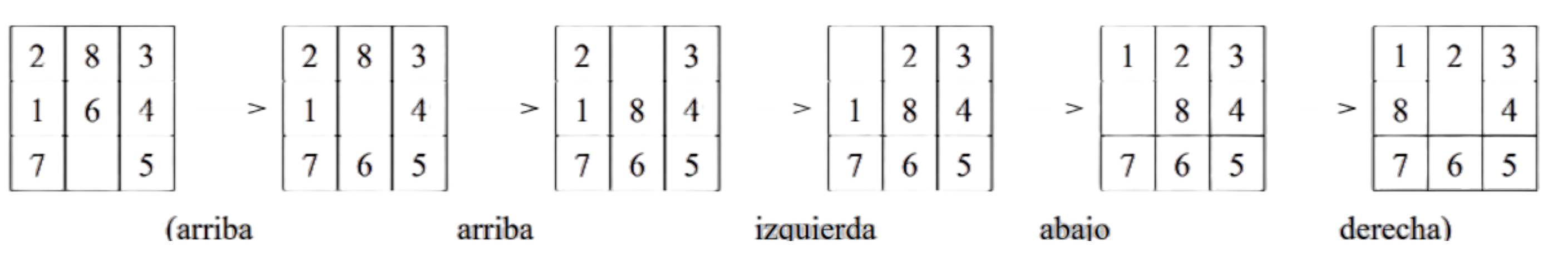
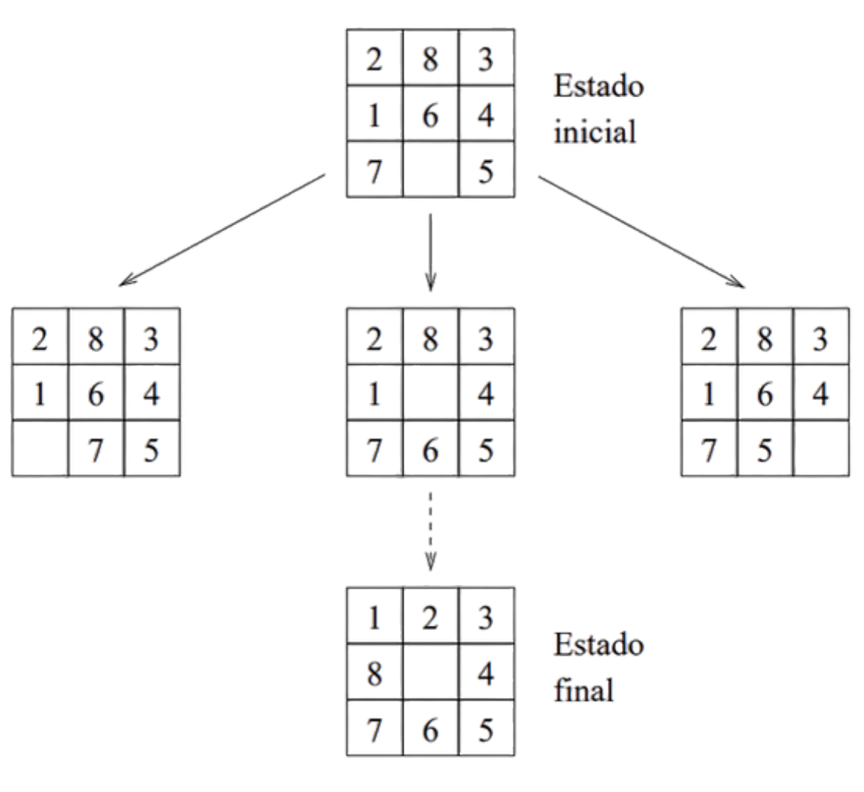


Gráfico en secuencia de estados del 8-puzzle.

Gráfico de estados en 8-puzzle.



Aplicaciones de la Búsqueda en Espacios de Estados

Los espacios de estados son útiles en muchas aplicaciones, por ejemplo:

En búsqueda de rutas
GPS y optimización de caminos

Resolución de problemas en Videojuegos
IA en Ajedrez o Go.

Planeación Automática
Sistemas que planifican acciones en robótica.

Algoritmos de Búsqueda No Informada

La búsqueda no informada o ciega, explora el espacio de estados sin usar información adicional sobre la solución. Solo conoce el **estado inicial**, **el estado meta** y **las posibles transiciones**.

Búsqueda en profundidad - DFS Depth-First Search

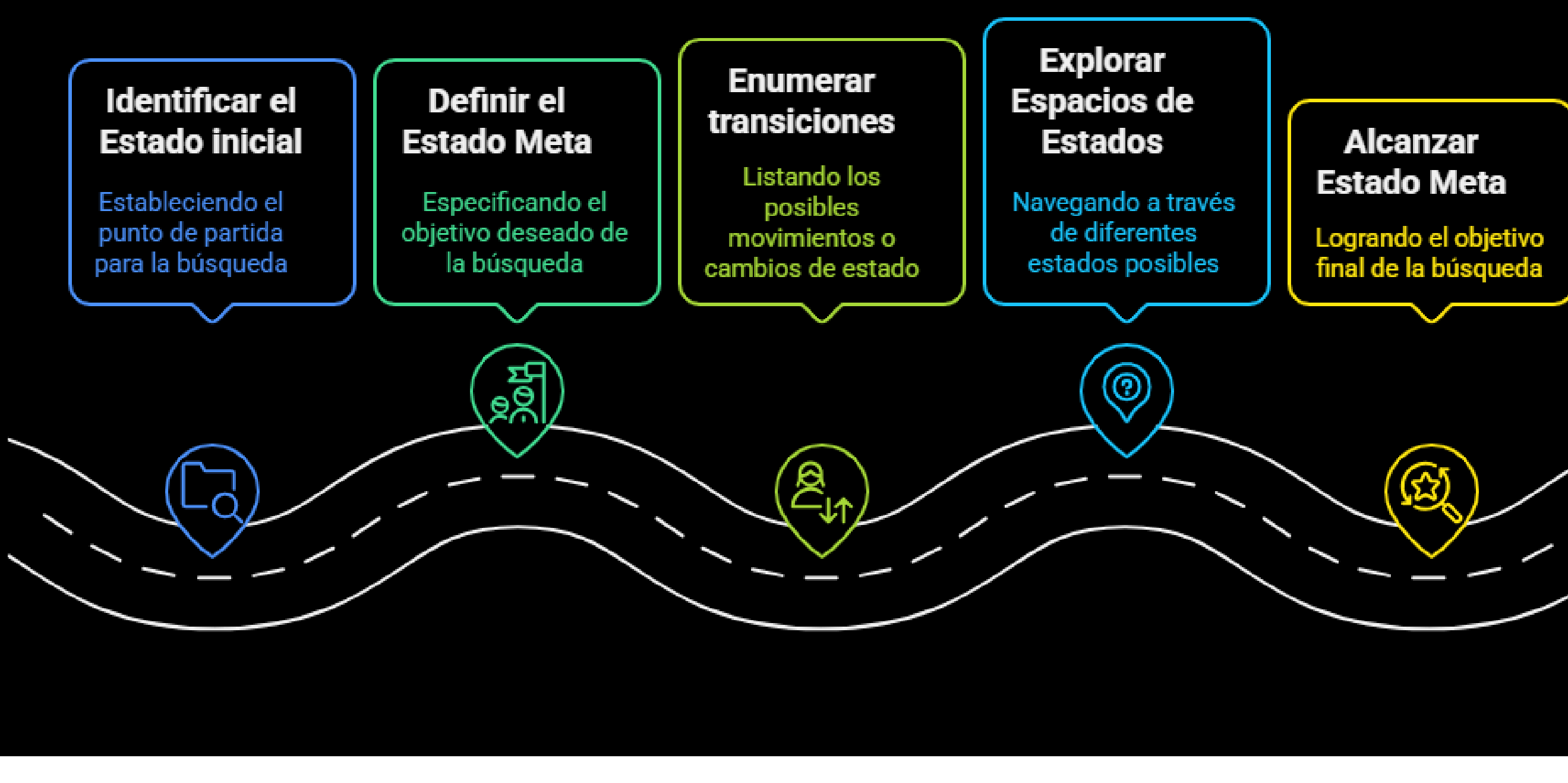
- Explora un camino completamente hasta llegar a un callejón sin salida, y luego retrocede para explorar otro camino.
- Utiliza LIFO (Last in, First out) para gestionar los nodos pendientes.

Ejemplo: En 8-Puzzle, exploraría cada camino hasta llegar a un estado bloqueado, y solo entonces probaría otras ramas.

Búsqueda en anchura - BFS Breadth-First Search

- Explora el espacio de estados por niveles, avanzando en todas las direcciones antes de descender a niveles más profundos.
- Utiliza FIFO (First in, First Out) para gestionar los nodos pendientes.

Ejemplo: En 8-Puzzle, buscaría todos los movimientos posibles desde el estado inicial antes de avanzar a los siguientes estados.



Identificar el Estado inicial
Estableciendo el punto de partida para la búsqueda

Definir el Estado Meta
Especificando el objetivo deseado de la búsqueda

Enumerar transiciones
Listando los posibles movimientos o cambios de estado

Explorar Espacios de Estados
Navegando a través de diferentes estados posibles

Alcanzar Estado Meta
Logrando el objetivo final de la búsqueda

Diferencias entre Bread-First Search y Deep-First Search

Criterio	BFS (Anchura)	DFS (Profundidad)
Estructura de datos	Cola (FIFO)	Pila (LIFO)
Óptimo	Encuentra la solución más corta (en términos de número de pasos) si el costo es uniforme.	Puede encontrar una solución, pero no necesariamente la más corta.
Complejidad Temporal	O(b^d) Dónde b = factor de ramificación y d = profundidad del nodo más cercano.	O(b^m) Dónde b = factor de ramificación y d = profundidad máxima del árbol.
Uso recomendado	Problemas donde la solución más corta es importante	Problemas con profundidades desconocidas o grandes espacios de estados

Búsqueda No Informada en 8-Puzzle

El 8-puzzle se representa como una cuadrícula 3x3 con 8 fichas que numeraremos del 1 al 8, más un hueco (espacio en blanco) que llamamos '0'.

Estado Inicial		
1	0	3
4	2	5
7	8	6

Estado Final		
1	2	3
4	5	6
7	8	0

Podemos guardar este estado inicial como un **string** **"103425786"**, donde cada carácter corresponde a una casilla.

Movimientos: se puede intercambiar el hueco '0' con alguna ficha adyacente (arriba, abajo, izquierda o derecha), generando un nuevo estado.

Reglas lógicas

BFS (Breadth First Search) explora primero todos los estados a la misma “distancia” (en número de movimientos) del estado inicial, antes de pasar a los estados de distancia mayor.

- Usa una cola FIFO para los estados por expandir.
- Se sacan los estados más antiguos de la cola (los que ingresaron primero) para expandir sus sucesores.
- Si un estado se repite (ya se generó antes), se ignora el duplicado.

Solución de 8- puzzle con Breadth-First Search

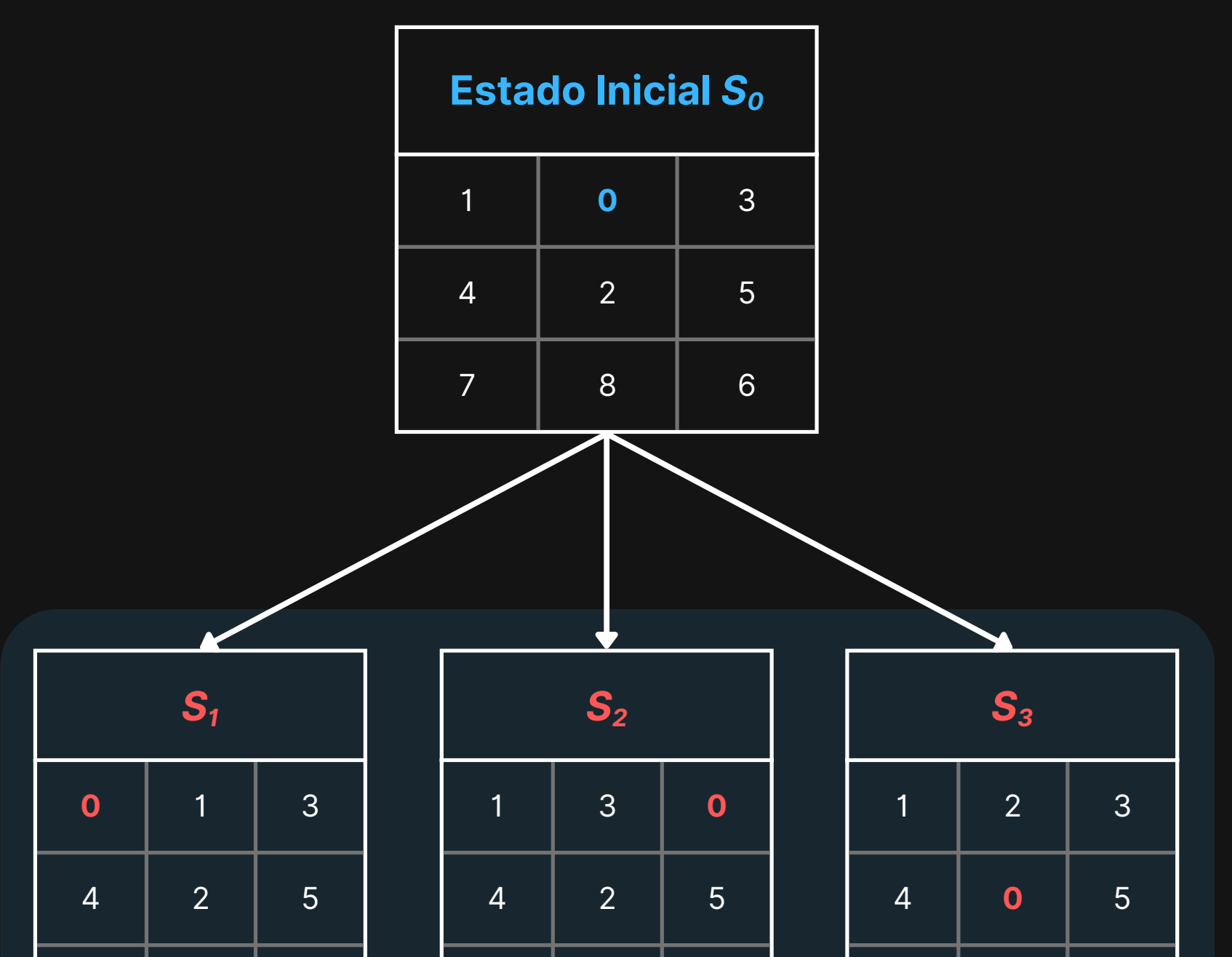
1. Supongamos un estado inicial *S_0*, el primero en la cola de estados FIFO sin predecesores (o su predecesor es *‘None’*). Es decir, la fila o cola *‘=[S_0]’*.

2. Se toma el primer estado de la cola (*S_0*) y se generan sus sucesores.

- Se determina la posición del *0* y se intercambia con los posibles vecinos.
- Cada sucesor *S_i* se incluye en la cola, siempre que no se haya visitado antes.
- Se indica su predecesor como *S_0* y la cola de estados quedaría: *cola=[S_1, S_2, S_3]*.

3. Se repite nuevamente tomando el primer estado de la cola.

- Ahora se toma *S_1* y se generan sus sucesores, expandiendo y añadiéndolos al final de la cola.
- Luego se toma *S_2* de la cola y se repite el proceso hasta llegar al esado final.



Estado Inicial S_0

1	0	3
4	2	5
7	8	6

S_1

0	1	3
4	2	5
7	8	6

S_2

1	3	0
4	2	5
7	8	6

S_3

1	2	3
4	0	5
7	8	6

Sucesores

Cada vez que se **expande un estado**, se coloca sus sucesores al final de la cola. Es decir, que primero se completará la expansión de todos los estados a **distancia 1** del inicial *S_0* antes de pasar con los estados a **distancia 2** (los sucesores de los sucesores), y así sucesivamente.

Solución de 8- puzzle con Deep-First Search

Reglas Lógicas

DFS (Depth First Search) sigue un camino lo más lejos posible, entrando rama tras rama hasta que no pueda avanzar más o encuentre la meta.

- Usa típicamente una **pila** (LIFO: Last-In, First-Out).
- Toma el **estado más reciente** que se haya insertado y no se haya expandido aún. **Explora todos sus sucesores**, y los sucesores de estos, etc.
- Al quedar sin caminos que explorar en la rama, **retrocede (backtracking)** a la rama anterior y prueba otra rama.

Paso a paso DFS

Siguiendo con el estado inicial ***S_0***, se inicia pila=[***S_0***].

1. Se extrae el último estado de la pila (al principio sería el estado inicial).
2. Generación de sucesores de ***S_0***, es decir, ***S_1***, ***S_2***, ***S_3***... que se insertan en la pila. El orden de inserción puede variar, pero aquel que sea el último será el primero en expandirse (LIFO).
3. Expansión sucesiva:
   a. Suponiendo pila=[***S_1, S_2, S_3***] se expande el tope de la pila, es decir, el estado ***S_3***.
   b. Generamos sus sucesores, por ejemplo ***S_{3a}, S_{3b}***, ... y se añaden en la pila.
   c. Se toma nuevamente el tope de la pila y se repite el proceso hasta que no haya más estados sucesores y se expanda el siguiente estado en la pila.

Algoritmos de Búsqueda Informada

A diferencia de la búsqueda no informada, emplea **información adicional (heurísticas)** para decidir qué caminos explorar primero.

📌 **¿Qué es una heurística?**

Es una **estrategia o técnica** que guía la búsqueda de soluciones, permitiendo abordar problemas complejos al proporcionar estimaciones informadas, **sin evaluar exhaustivamente todas las posibilidades**.

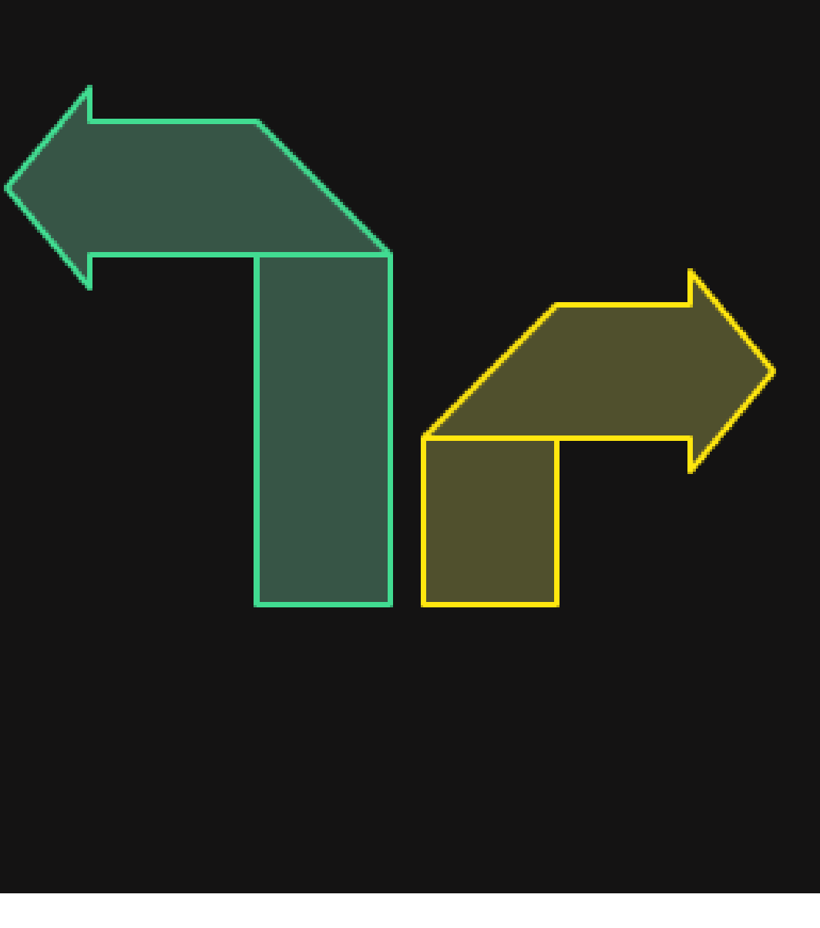
Número de Piezas Fuera de Lugar
Cuenta las piezas mal colocadas para evaluar el desorden

Distancia Manhattan
Calcula la distancia en cuadrícula para estimar movimientos

Distancia Euclidiana
Mide la distancia en línea recta para estimar

Algoritmos de Búsqueda Informada

Los métodos de búsqueda informada implementan una función de evaluación *f(n)* que utiliza la heurística definida el cálculo en cada estado.



Búsqueda A*

Óptimo y completo con heurística admisible, considera costo y heurística.

$f(x) = g(n) + h(x)$

Búsqueda Greedy

Rápido pero no garantiza la optimalidad, propenso a mínimos locales.

$f(x) = h(x)$

Diferencias entre A∗y Greedy

Criterio	A∗	Greedy
Estrategia	Elige el estado considerando costo y heurística.	Elige el estado con la mejor heurística.
Búsqueda Completa	Sí.	No, puede quedar atrapado en bucles.
Óptimo	Sí (con heurística admisible).	No necesariamente.
Velocidad	Más lento, pero encuentra el mejor camino.	Más rápido, pero menos confiable.
Uso recomendado	Cuando se requiere precisión y solución óptima.	Cuando importa la velocidad y no la perfección.