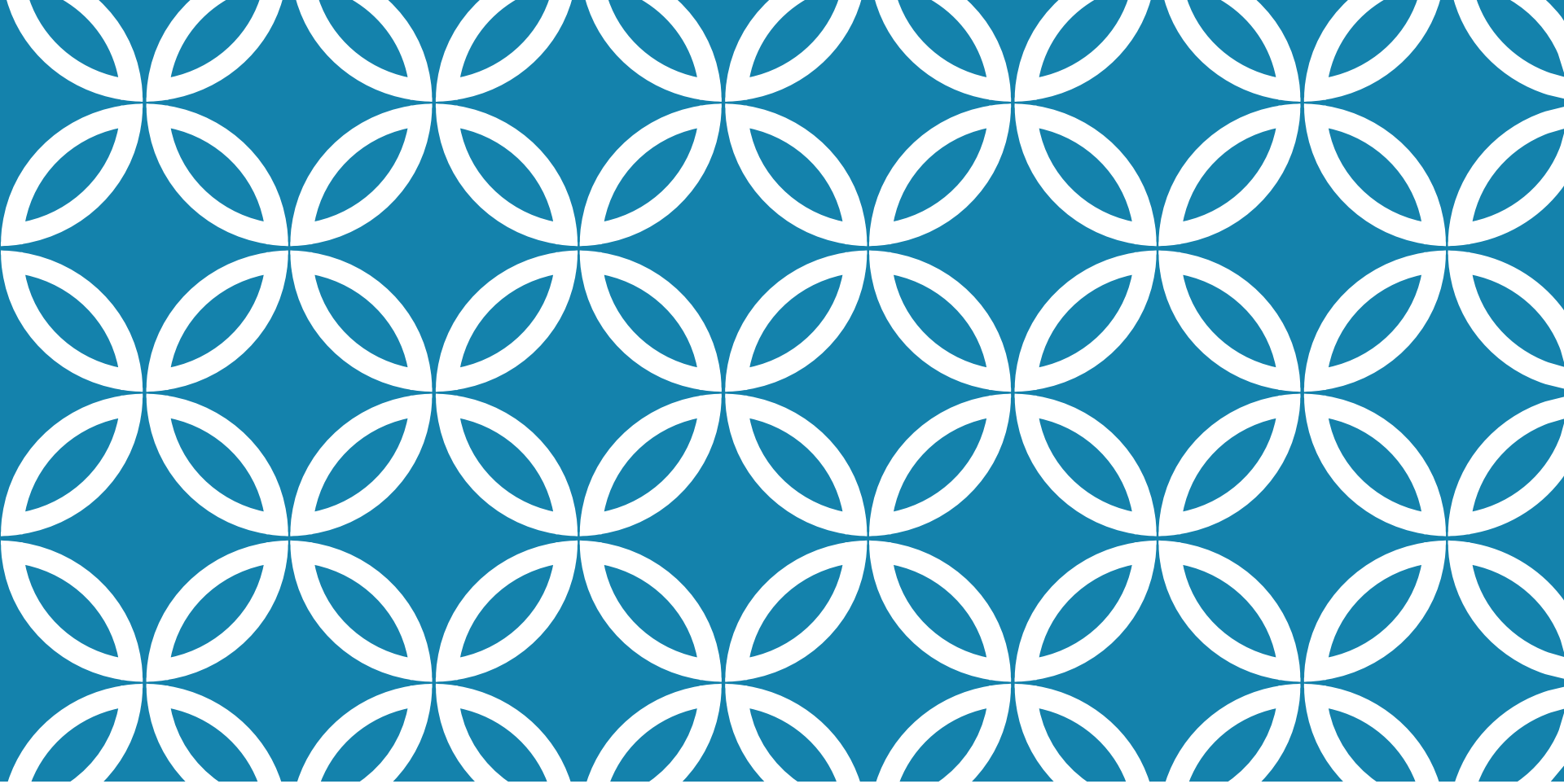


BÚSQUEDA NO INFORMADA

José Manuel Molina López
Grupo de Inteligencia Artificial Aplicada

CONTENIDO

Búsqueda I. Introducción al concepto de búsqueda

- Motivación
- Ejemplos de problemas
- Representación de los problemas

Búsqueda II. Búsqueda sin información

- Búsqueda en amplitud
- Búsqueda en profundidad
- Variantes

Motivación

Ejemplos de problemas

Representación de los problemas

EL CONCEPTO DE BÚSQUEDA

MOTIVACIÓN

Definición de búsqueda
Definición de un problema
Restricciones
El concepto de Información

DEFINICIÓN DE BÚSQUEDA

Problema: encontrar una secuencia de operaciones que nos conducen desde un estado inicial a un estado solución.

Entorno:

Conjunto de estados	Acciones u operadores, función sucesor	Vecindario: conjunto de estados al que se llega desde un estado dado	Camino: secuencia de estados conectado por una secuencia de acciones	Coste (de un camino)

INSTANCIA DE UN PROBLEMA

En cada instancia de un problema:

- Estado inicial
- Estados objetivo o meta (a veces, función objetivo)
  - Explícito (solución al 8-puzzle)
  - Implícito (dar jaque mate, minimización de una función)

RESTRICCIONES

Se asumen una serie de supuestos:

- Entorno estático
- Entorno discreto (se escoge entre un número de opciones fijas)
- Determinismo (resultado de una opción bien definido)

EJEMPLOS DE PROBLEMAS

8-puzzle

EL problema de las 8 reinas

Torres de Hanoi

Cálculo de rutas

TSP

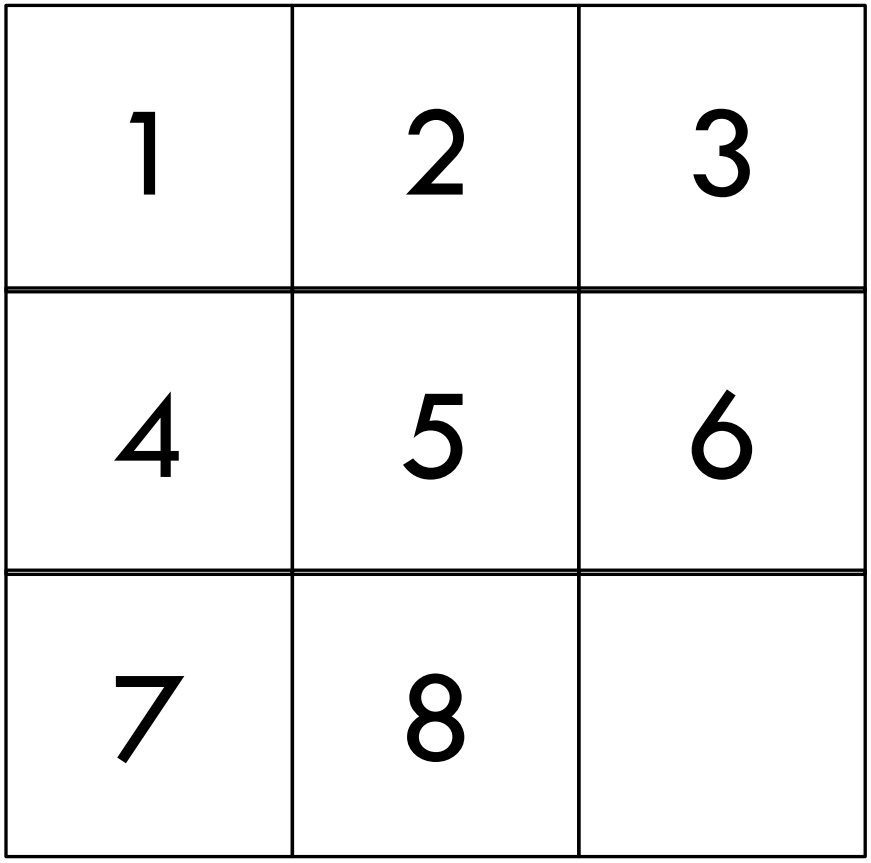
PROBLEMA DEL 8-PUZZLE

Un marco contiene nueve espacios y ocho fichas deslizantes numeradas del 1 al 8.

La posición inicial es cualquiera.

Las acciones posibles en cada estado son mover las fichas adyadentes al espacio vacío.

El objetivo es ordenar las fichas. El estado objetivo es único.



PROBLEMA DE LAS 8 REINAS

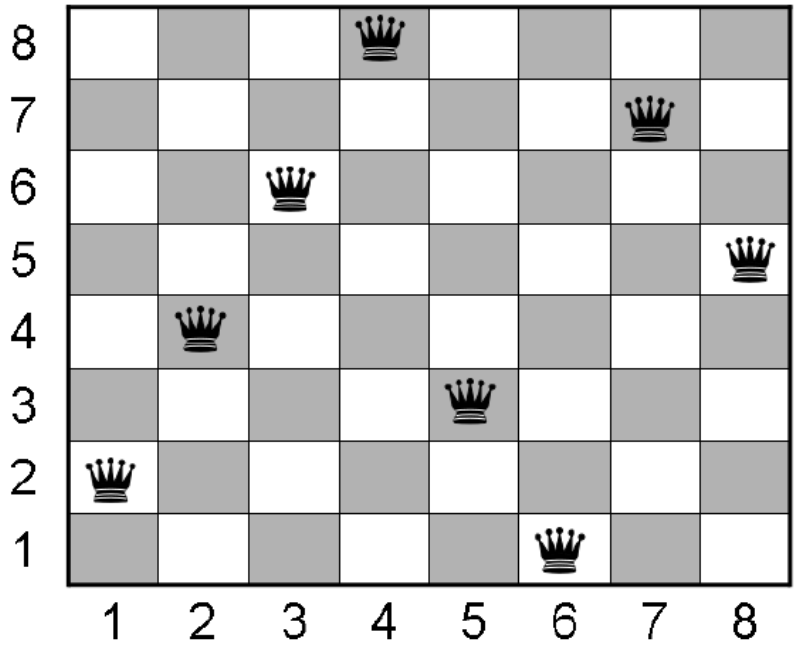
Objetivo: Colocar 8 reinas en un tablero de ajedrez de manera que ninguna reina ataque a ninguna otra (una reina ataca a otra si está en su misma fila, columna o diagonal)

Dos posibles formulaciones del problema:

- Formulacion completa de estados: comienza con las 8 reinas en el tablero y las mueve
- Formulación incremental: comienza con el tablero vacío, y añade una reina cada vez.

Objetivo:

- No importa el camino
- No hay descripción explícita, sí una función que comprueba si se ha llegado

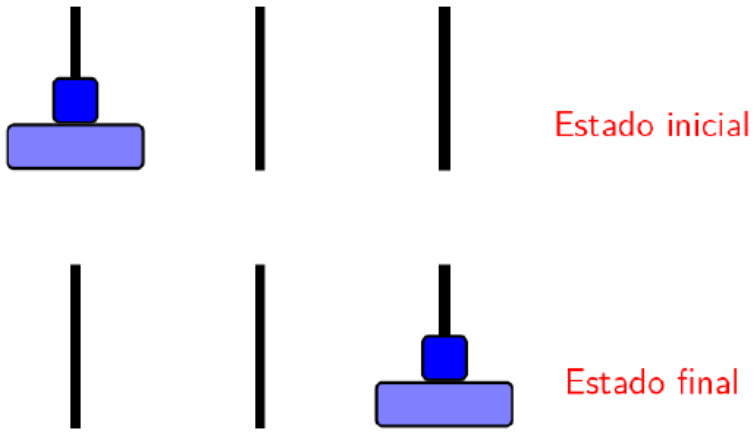


TORRES DE HANOI

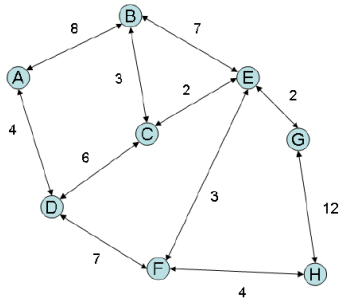
Objetivo: Mover las piezas de manera que nunca una pieza más pequeña esté debajo de una pieza mayor

Objetivo:

- Importa el camino
- El estado inicial está perfectamente definido
- Hay una descripción explícita del estado final



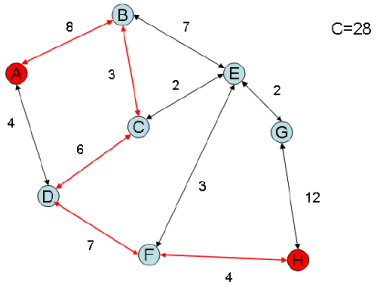
PROBLEMAS DE RUTAS

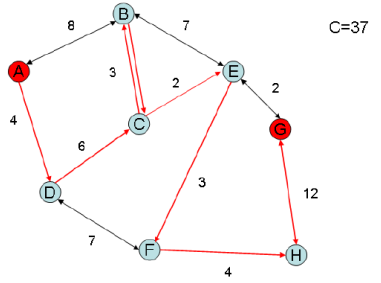


Planificación de rutas: dado un conjunto de medios de transporte y tiempos (coste), encontrar rutas que permitan llegar en un tiempo dado (o en el mínimo tiempo)

Problema del viajante (TSP): visitar todos los nodos una vez en el mínimo tiempo. Uso en distribución o encaminamiento.

Movimiento de robots: dada una serie de tareas a realizar en distintos puntos, determinar la secuencia de movimientos que cumple las restricciones (precedencia de tareas, etc.).





REPRESENTACIÓN DE LOS PROBLEMAS DE BÚSQUEDA

Árboles
Propiedades de los árboles
Comparación de problemas

ÁRBOLES

El problema se puede representar mediante un grafo

La búsqueda consiste en recorrer el grafo en un orden determinado

La búsqueda se puede representar con un árbol

Si el grafo contiene ciclos, el árbol de búsqueda es infinito (ej. con acciones reversibles)

Para evitar los ciclos, en el árbol se pueden controlar los estados repetidos

ÁRBOLES

Se representa en el nodo superior el estado inicial

Se representan las acciones con arcos orientados

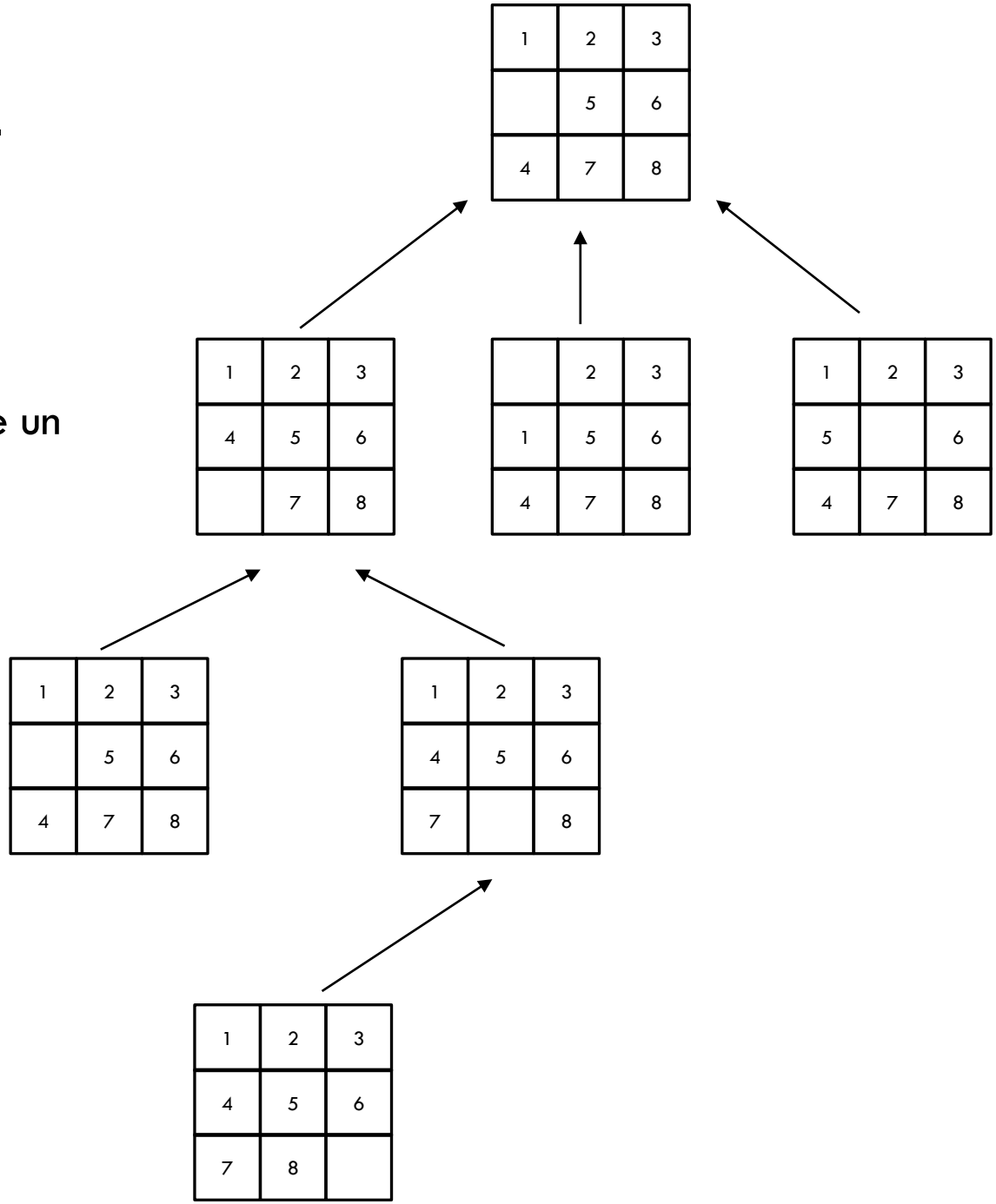
Los sucesores de un nodo son los conectados con el mismo

Los sucesores de un nodo se procesan de izquierda a derecha.

El orden de generación de los sucesores afecta la búsqueda.

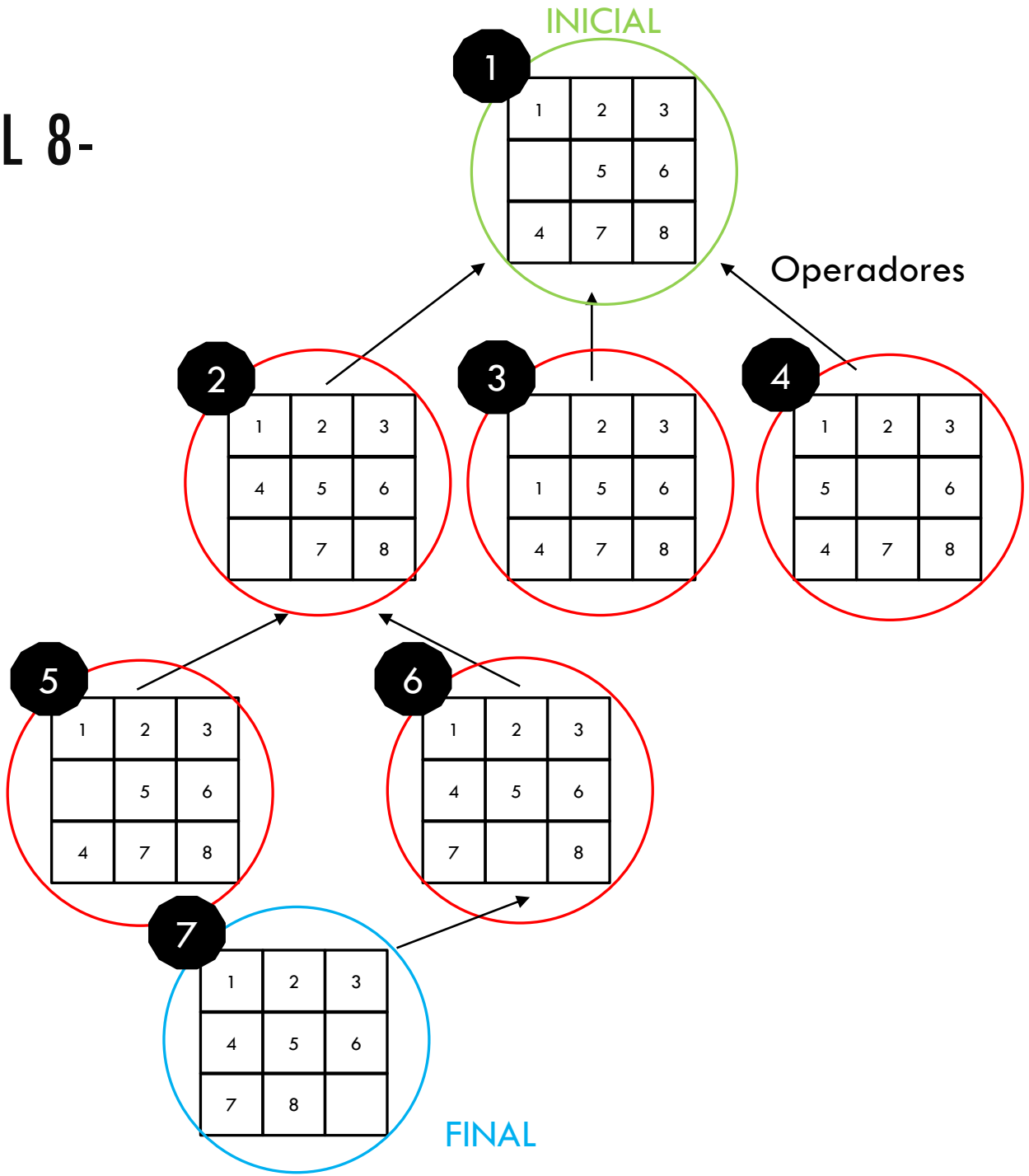
PROBLEMA DEL 8-PUZZLE

Representación mediante un árbol



PROBLEMA DEL 8-PUZZLE

Estados



PROPIEDADES DE LOS ÁRBOLES

fr: Factor de ramificación (branching). Número de opciones en

los nodos del árbol de búsqueda.

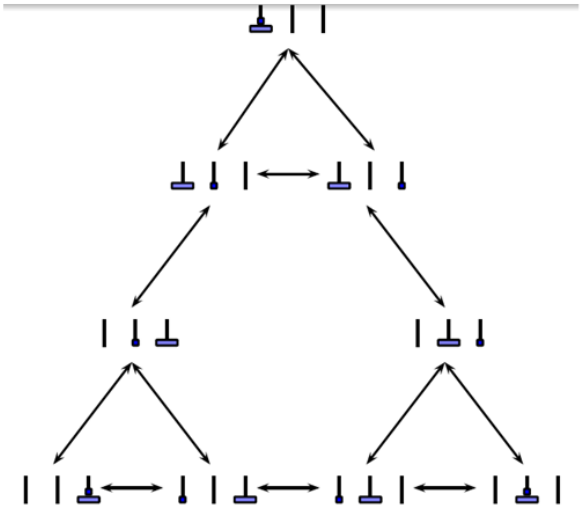
p: Profundidad de la solución (depth). Mínimo número de

acciones hasta una solución desde un inicio dado.

m: Máxima longitud de un camino

TORRES DE HANOI

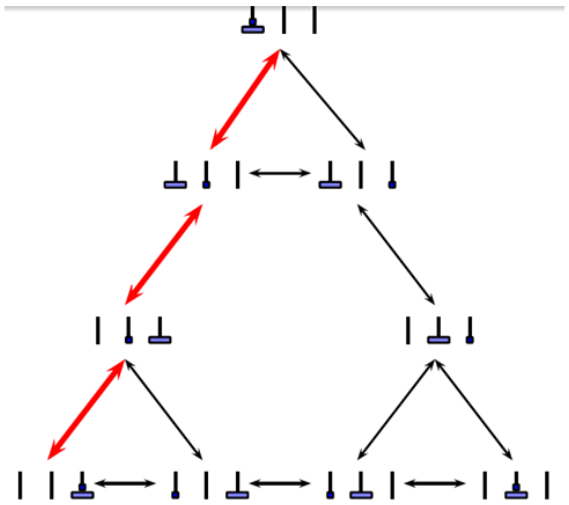
Representación mediante un árbol



El factor de ramificación (máximo) **fr** es una propiedad del grafo de estados

TORRES DE HANOI

Representación mediante un árbol



La profundidad (p=3) es una propiedad del problema a resolver

COMPARAR PROBLEMAS

Comparamos los algoritmos mediante las siguientes medidas:

- Completitud: Encuentra una solución
- Optimalidad: Encuentra la mejor solución
- Complejidad temporal: Tiempo que tarda en función del tamaño del problema
- Complejidad espacial: Memoria que requiere en función del tamaño del problema

Las medidas de complejidad se calculan para el caso peor

Dominio	Número de Estados	Tiempo (10^7 nodos/s)
8-puzzle	$181,440$	0.01 segundos
15-puzzle	10^{13}	11.5 días
24-puzzle	10^{25}	$31,7 \times 10^9$ años
Hanoi(3,2)	9	9×10^{-7} segundos
Hanoi(3,4)	81	8.1×10^{-6} segundos
Hanoi(3,8)	6561	6.5×10^{-4} segundos
Hanoi(3,16)	$4,3 \times 10^7$	4.3 segundos
Hanoi(3,24)	$2,284 \times 10^{11}$	0.32 días
Cubo Rubik $2 \times 2 \times 2$	10^6	0.1 segundos
Cubo Rubik $3 \times 3 \times 3$	4.32×10^{19}	31,000 años

COMPARAR PROBLEMAS

Amplitud
Profundidad
Variaciones

BÚSQUEDA SIN INFORMACIÓN

BÚSQUEDA EN AMPLITUD

Algoritmo
Características
Coste

ALGORITMO EN AMPLITUD

1 Crear lista ABIERTA con el nodo inicial I (estado inicial)

2 ÉXITO=Falso

3 Hasta que ABIERTA esté vacía O ÉXITO

- Quitar de ABIERTA el primer nodo N
- Si N tiene sucesores ENTONCES EXPANDE(N):
  - Generar los sucesores de N (sin repeticiones – ver nota 1)
  - Crear punteros desde los sucesores hacia N
  - Si algún sucesor es nodo meta – ver nota 2 - Entonces EXITO=Verdadero
  - Si no Añadir los sucesores al final de ABIERTA

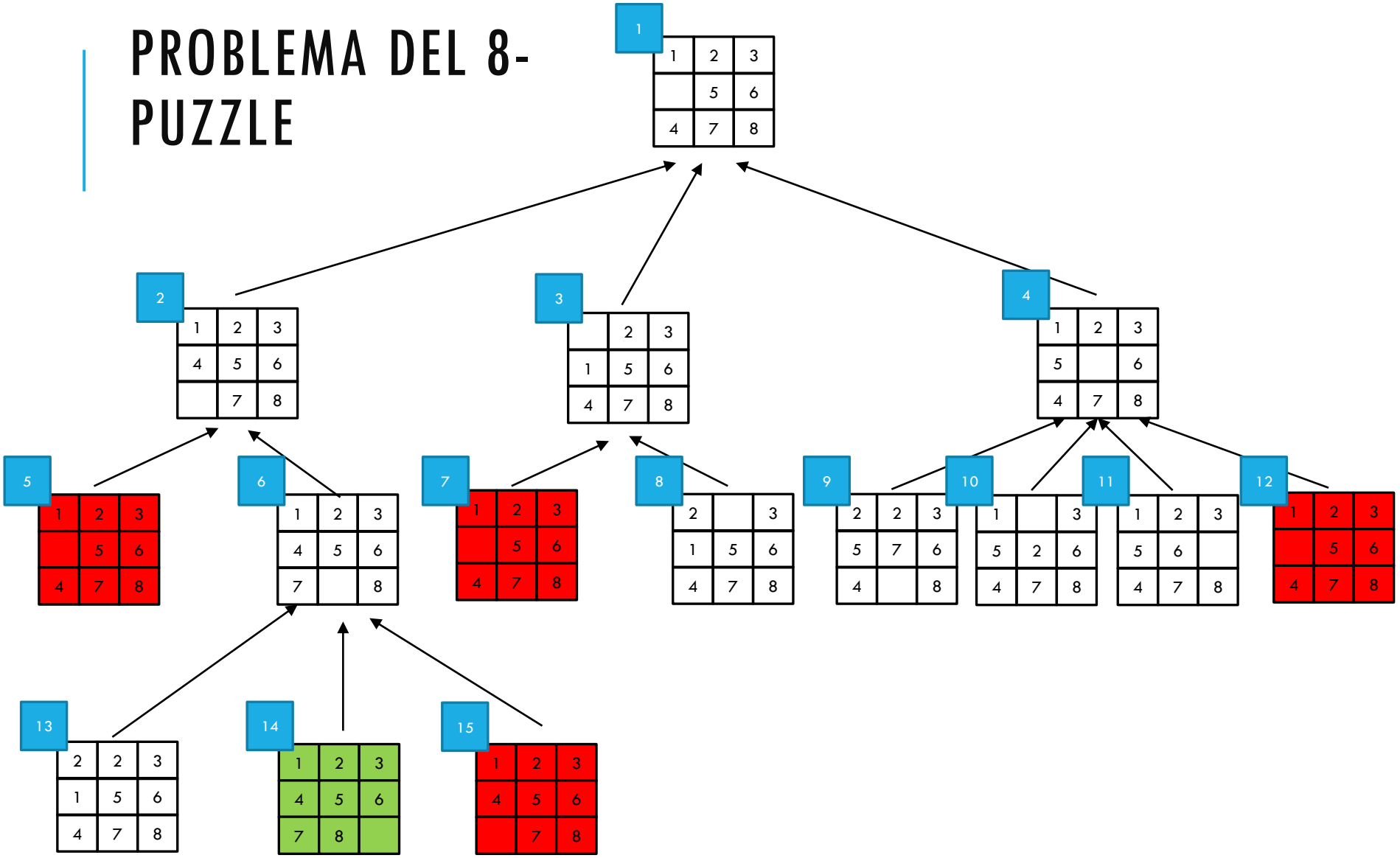
4 Si ÉXITO ENTONCES Solución=camino desde I a N por los punteros

Si no Solución=fracaso

Nota 1 Control de ciclos: no se generan sucesores que hayan sido ya generados con anterioridad (no hace falta escribirlos, pero podemos hacerlo por claridad)

Nota 2 Control de objetivo: algunas implementaciones comprobarán solamente el nodo que se va a expandir, no todos los que se generan

PROBLEMA DEL 8-PUZZLE



CARACTERÍSTICAS

Completa. Si existe, encuentra siempre solución

Óptima. Encuentra siempre la solución menos profunda. Con costes uniformes, es la óptima.

El control de ciclos se introduce por eficiencia.

COSTES

Se asume que se visita cada nodo y en ese acto se expande.

La complejidad temporal y espacial es la misma, porque hay que retener en memoria todos los nodos que se visitan.

El primer nivel genera fr nodos

El segundo nivel son $fr \times fr$ nodos

En el nivel p (profundidad de la solución) son fr^p nodos

Complejidad espacial y temporal $\approx O(fr^p)$

BÚSQUEDA EN PROFUNDIDAD

Algoritmo
Características
Coste

ALGORITMO EN PROFUNDIDAD

El método básico es primitivo, pero se pueden realizar mejoras

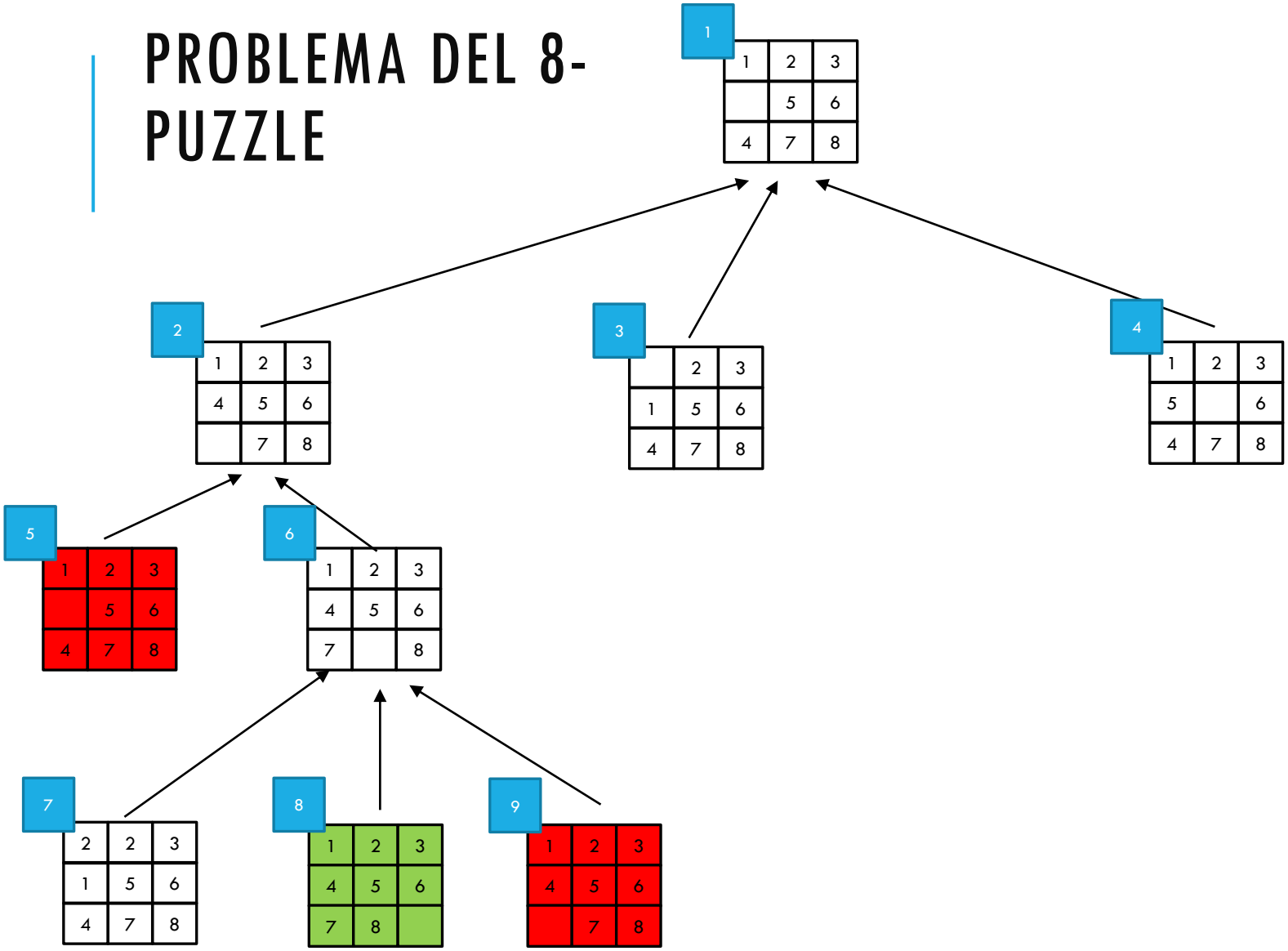
Se usa una lista ABIERTA pero se va expandiendo el nodo más a la izquierda

Usa memoria mínima (en realidad la versión que damos usa un poco más que las versiones más reducidas de PROFUNDIDAD)

Si un nodo no tiene sucesores, acabaría sin llegar a la solución (no sería completo)

Si el grafo tiene ciclos, puede no acabar nunca

PROBLEMA DEL 8-PUZZLE



ALGORITMO EN PROFUNDIDAD CON RETROCESO

RETROCESO: se vuelve al siguiente nodo en la lista ABIERTA si se da alguna de las siguientes circunstancias:

- No hay sucesores del nodo que se expande
- Control de ciclos (varias posibilidades):

  1 En nodos ya generados (más memoria)

  2 Sólo en lista ABIERTA

  3 En camino actual: almacena y comprueba el camino actual, únicamente se evita expandir nodos que están repetidos en este camino

Retroceso también si:

- Se ha llegado al límite de profundidad (Profundidad con límite incremental: se va incrementando este límite)
- Se sabe que el estado no conduce a la solución: método de Ramificación y Poda (Branch and Bound)

ALGORITMO EN PROFUNDIDAD CON RETROCESO

(Con retroceso, detección de ciclos y profundidad máxima)

1 Crear lista ABIERTA con el nodo inicial I y su profundidad (0)

2 ÉXITO=Falso y M=Profundidad Máxima

3 Hasta que ABIERTA esté vacía O ÉXITO

Quitar de ABIERTA el primer nodo N de profundidad P

Si (P < M) Y N tiene sucesores

Generar los sucesores de N (sin ciclos1)

Crear punteros desde los sucesores hacia N

Si algún sucesor es nodo meta

Entonces EXITO=Verdadero

Si no Añadir los sucesores al principio de ABIERTA con profundidad P+1

4 Si EXITO

Entonces Solución=camino desde I a N por los punteros

Si no Solución=fracaso

1 Control de ciclos: si no se especifica asumiremos que no se generan nodos que están en el mismo camino (método 2)

CARACTERÍSTICAS

No Completa. No garantiza encontrar la solución. Pero se hace completa si se añade detección de ciclos y retroceso.

No Óptima. Encuentra la solución más próxima a la rama por la que busca, no necesariamente la que se encuentra a menor profundidad.

Eficiente, cuando la meta está alejada del estado inicial, o hay problemas de memoria

COSTES

Cuando se llega al final del árbol se pueden descartar nodos visitados

Complejidad espacial: el camino hasta el punto en el que se busca, y las ramas aún no visitadas: $fr \times m + 1 \approx O(fr \times m)$

Complejidad temporal: en el peor de los casos, el nodo solución es el último: $O(fr^m)$ (m es la máxima profundidad de un camino)

UNA COMPARACIÓN ENTRE BÚSQUEDA EN AMPLITUD Y PROFUNDIDAD CON RETROCESO

Amplitud:

http://www.youtube.com/watch?v=tDtMj9wWtEk

Profundidad:

http://www.youtube.com/watch?v=AKgo5I5eYAg

VARIACIONES

Profundidad Incremental
Bidireccional
Dikjstra

MODIFICACIONES EN LOS ALGORITMOS BÁSICOS

Profundidad incremental

Bidireccional ($O(fr^{p/2})$) cuando los estados finales son explícitos

Dijkstra

- Búsqueda en amplitud
- g(n): coste de alcanzar el nodo n desde el nodo raíz
- Expande el nodo con mínima g(n)

Ramificación y acotación (Branch and Bound B&B)

- Normalmente búsqueda en profundidad
- Cuando se encuentra una solución, su coste se utiliza como límite para podar nodos posteriores

BÚSQUEDA CON COSTES: DIKJSTRA

Si se introducen costes, la solución óptima no depende de la profundidad sino de los costes del camino (suma de costes de las acciones).

Se pueden modificar los algoritmos de búsqueda para que escojan el camino de menor coste acumulado en cada momento (ABIERTA ordenada por mínimo coste).

Algoritmo completo.

Garantiza la optimalidad en coste.