

# Inteligencia Artificial

Sistemas inteligentes artificiales



# Procedimientos para la solución de problemas



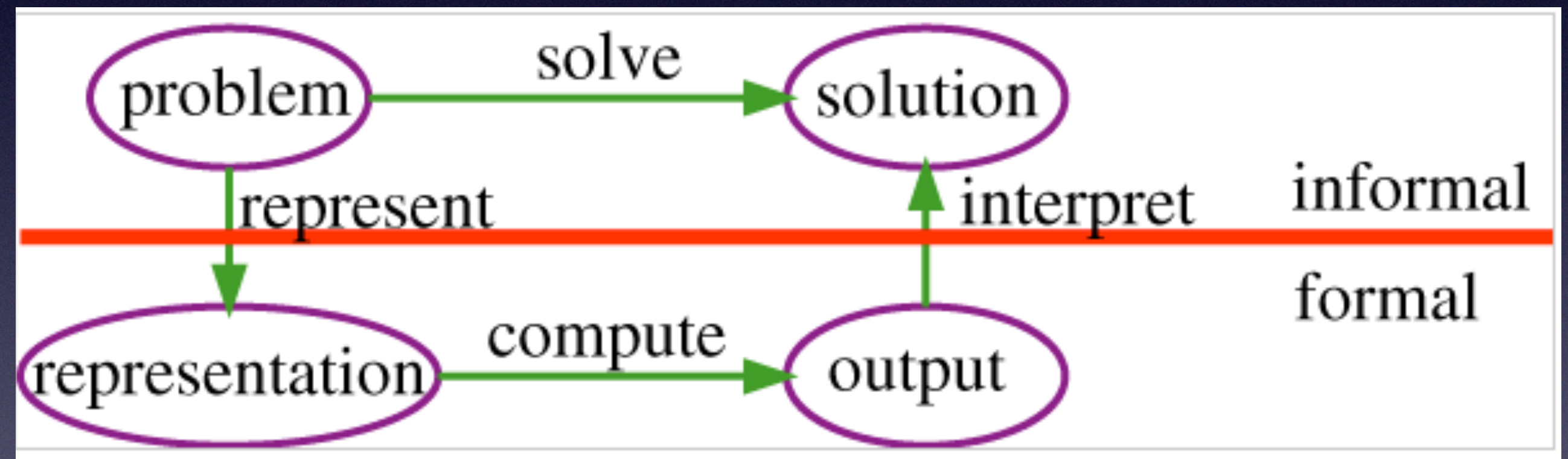
# Temas

- Resolución de problemas
- Búsqueda
- Tipos y métodos de búsqueda
- Juegos complejidad



# Resolución de problemas 1

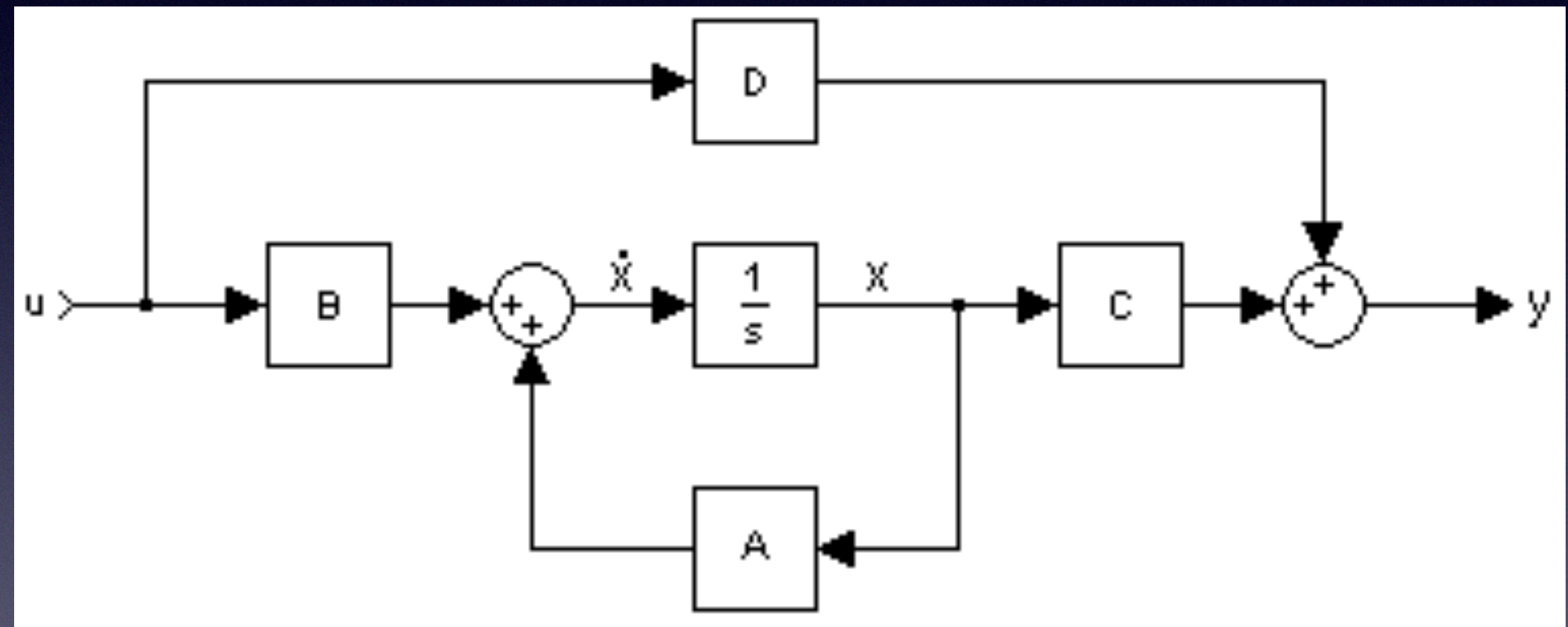
- Se puede definir la resolución de problemas como:
- El proceso que, partiendo de una situación inicial y utilizando un conjunto de procedimientos/reglas/acciones seleccionados a priori, es capaz de explicitar el conjunto de pasos que nos llevan a una situación posterior que llamamos solución.





# Resolución de problemas 2

- Espacio de estados
- Una de las aproximaciones más generales y sencillas de formalizar un problema y sus posibles mecanismos de solución es por medio de lo que se denomina espacio de estados.
- Estado: representación de los elementos que describen el problema en un momento dado.





# Resolución de problemas 3

- Estados:
- Las variables de estado son el subconjunto más pequeño de variables de un sistema que pueden representar su estado dinámico completo en un determinado instante. Las variables de estado deben ser independientes entre sí. En el caso de la representación lineal de un sistema, las variables de estado deben ser linealmente independientes entre sí: una variable de estado no puede ser una combinación lineal de otras variables de estado.

Tipo de sistema	Modelo de espacio de estados
Continuo e invariante en el tiempo	$\dot{\mathbf{x}}(t) = \mathbf{A}\mathbf{x}(t) + \mathbf{B}\mathbf{u}(t)$ $\mathbf{y}(t) = \mathbf{C}\mathbf{x}(t) + \mathbf{D}\mathbf{u}(t)$
Continuo y variante en el tiempo	$\dot{\mathbf{x}}(t) = \mathbf{A}(t)\mathbf{x}(t) + \mathbf{B}(t)\mathbf{u}(t)$ $\mathbf{y}(t) = \mathbf{C}(t)\mathbf{x}(t) + \mathbf{D}(t)\mathbf{u}(t)$
Discreto e invariante en el tiempo	$\mathbf{x}(k+1) = \mathbf{A}\mathbf{x}(k) + \mathbf{B}\mathbf{u}(k)$ $\mathbf{y}(k) = \mathbf{C}\mathbf{x}(k) + \mathbf{D}\mathbf{u}(k)$
Discreto y variante en el tiempo	$\mathbf{x}(k+1) = \mathbf{A}(k)\mathbf{x}(k) + \mathbf{B}(k)\mathbf{u}(k)$ $\mathbf{y}(k) = \mathbf{C}(k)\mathbf{x}(k) + \mathbf{D}(k)\mathbf{u}(k)$
Transformada de Laplace del sistema continuo e invariante en el tiempo	$s\mathbf{X}(s) = \mathbf{A}\mathbf{X}(s) + \mathbf{B}\mathbf{U}(s)$ $\mathbf{Y}(s) = \mathbf{C}\mathbf{X}(s) + \mathbf{D}\mathbf{U}(s)$
Transformada Z del sistema discreto e invariante en el tiempo	$z\mathbf{X}(z) = \mathbf{A}\mathbf{X}(z) + \mathbf{B}\mathbf{U}(z)$ $\mathbf{Y}(z) = \mathbf{C}\mathbf{X}(z) + \mathbf{D}\mathbf{U}(z)$



# Resolución de problemas 3

- La elección de los estados en casos concretos, no solo determina qué información se almacenará de las diversas situaciones por las que pasa el problema, sino que en muchos casos es determinante también para decidir cuáles son las reglas u operaciones básicas que se permiten para realizar transformaciones entre estados.

$$\dot{\mathbf{x}}(t) = A(t)\mathbf{x}(t) + B(t)\mathbf{u}(t)$$

$$\mathbf{y}(t) = C(t)\mathbf{x}(t) + D(t)\mathbf{u}(t)$$

donde

$\mathbf{x}(t) \in \mathbb{R}^n$  es el **vector de estados**

$\mathbf{y}(t) \in \mathbb{R}^q$  es el **vector de salidas**

$\mathbf{u}(t) \in \mathbb{R}^p$  es el **vector de entradas**

$A(t) \in \mathbb{R}^{n \times n}$  es la **matriz de estados**

$B(t) \in \mathbb{R}^{n \times p}$  es la **matriz de entrada**

$C(t) \in \mathbb{R}^{q \times n}$  es la **matriz de salida**

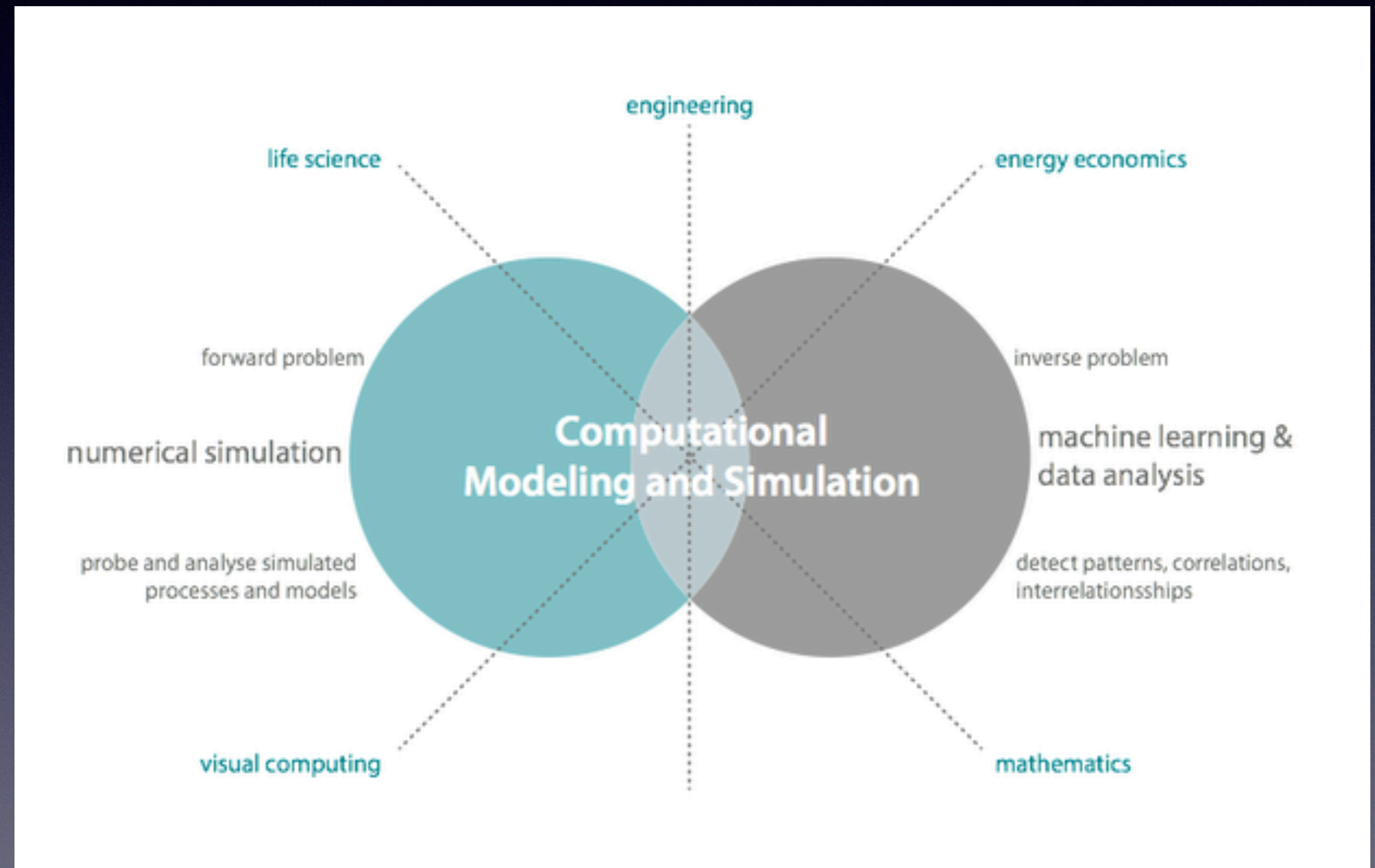
$D(t) \in \mathbb{R}^{q \times p}$  es la **matriz de transmisión directa**

$$\dot{\mathbf{x}}(t) := \frac{d\mathbf{x}(t)}{dt}$$



# Modelado

- Modelado computacional de un problema: proceso de transformar el problema original ("mundo real") en un espacio de estados que es manipulable por medios automáticos.



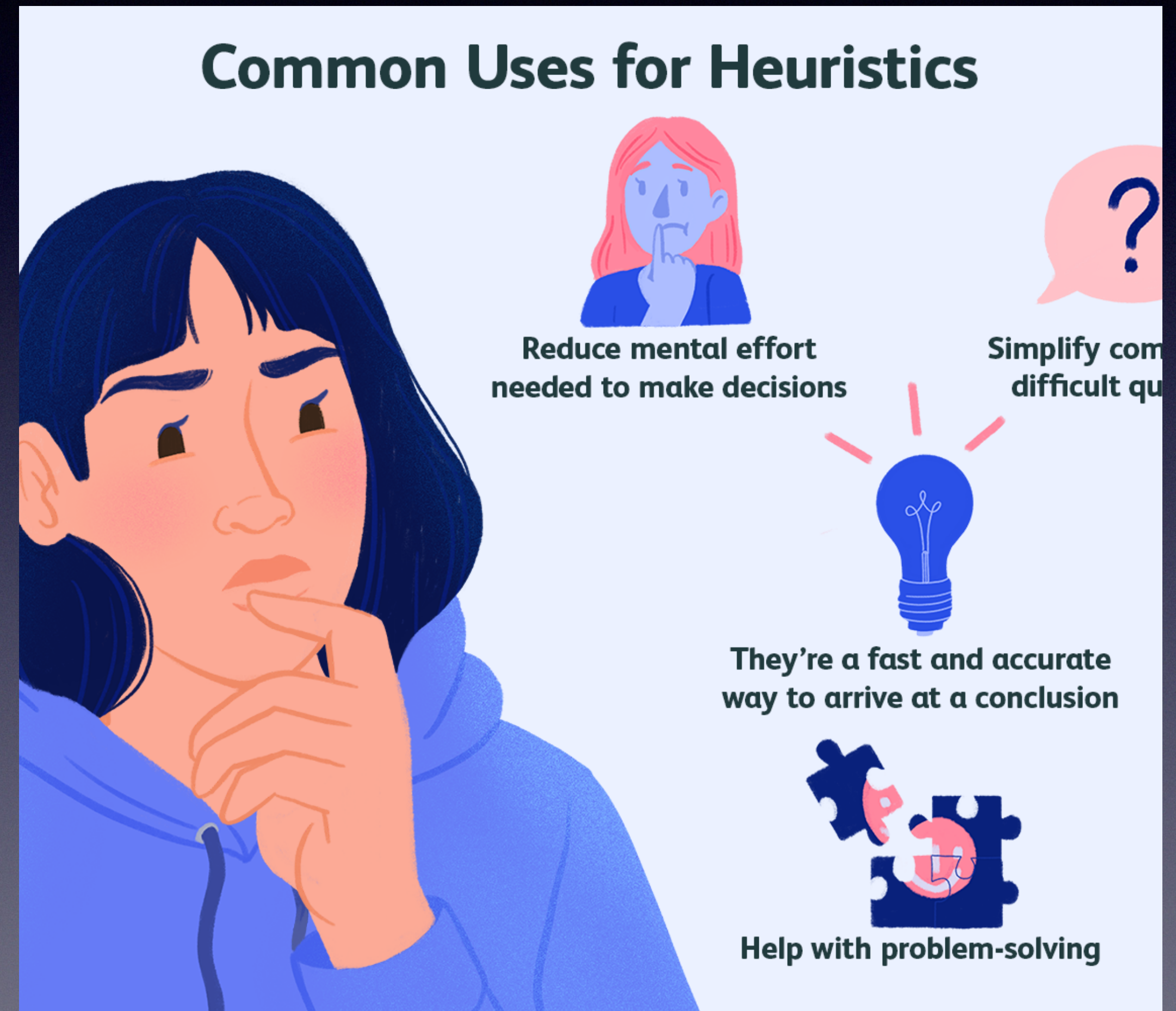


# Metaheurísticas



# Tipos de metaheurísticas

1. Metaheurísticas de relajación
2. Metaheurísticas constructivas
3. Metaheurísticas de búsquedas
4. Metaheurísticas para los procedimientos evolutivos





# Metaheurísticas de relajación

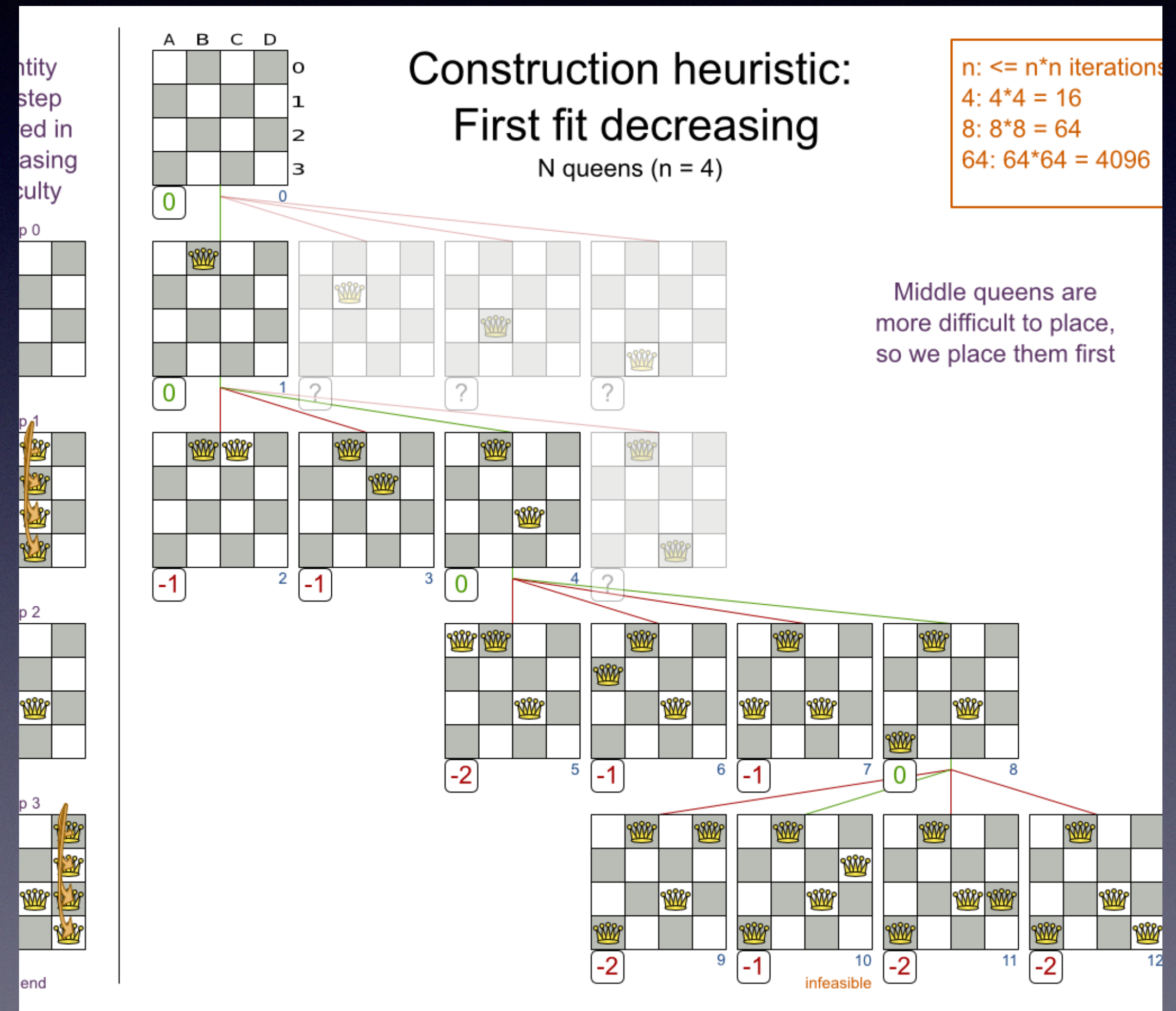
- Se refieren a procedimientos de resolución de problemas que utilizan relajaciones del modelo original (es decir, modificaciones del modelo que hacen al problema más fácil de resolver), cuya solución facilita la solución del problema original.





# Metaheurísticas constructivas

- Se orientan a los procedimientos que tratan de la obtención de una solución a partir del análisis y selección paulatina de las componentes que la forman.





# Metaheurísticas de búsquedas

- Guían los procedimientos que usan transformaciones o movimientos para recorrer el espacio de soluciones alternativas y explotar las estructuras de entornos asociadas.

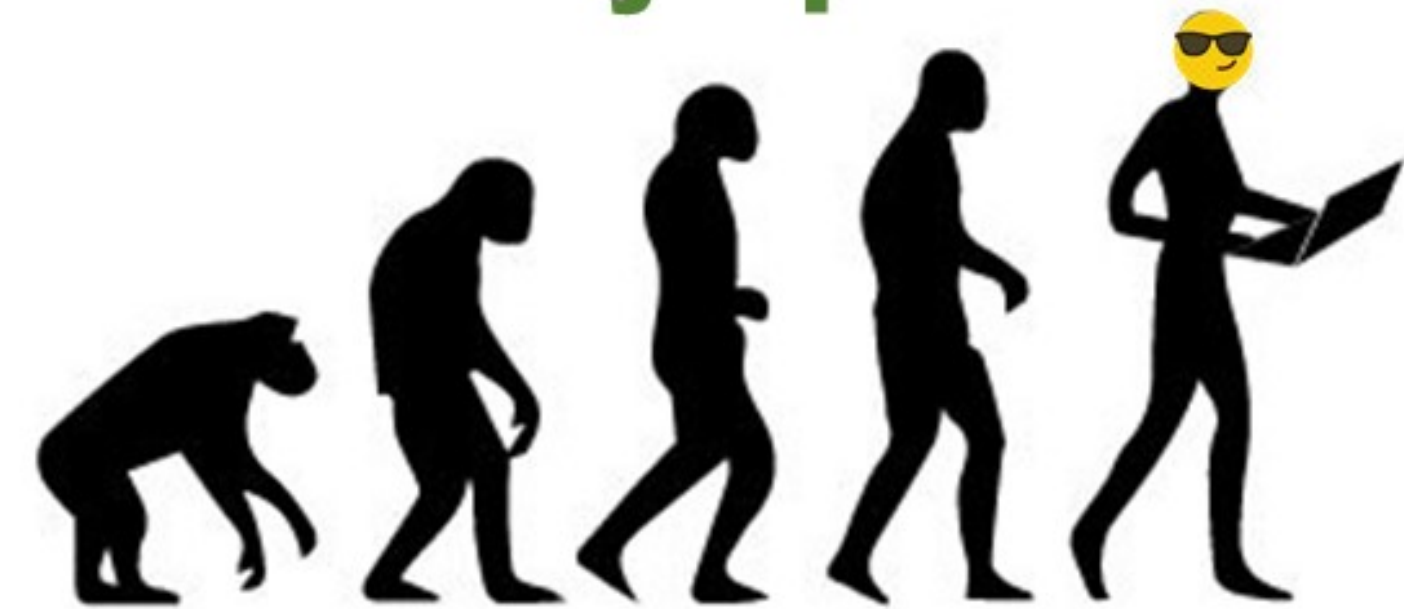




# Metaheurísticas evolutivas

- Están enfocadas a los procedimientos basados en conjuntos de soluciones que evolucionan sobre el espacio de soluciones para acercarse a la solución deseada

**Evolutionary Optimization**



**Python**

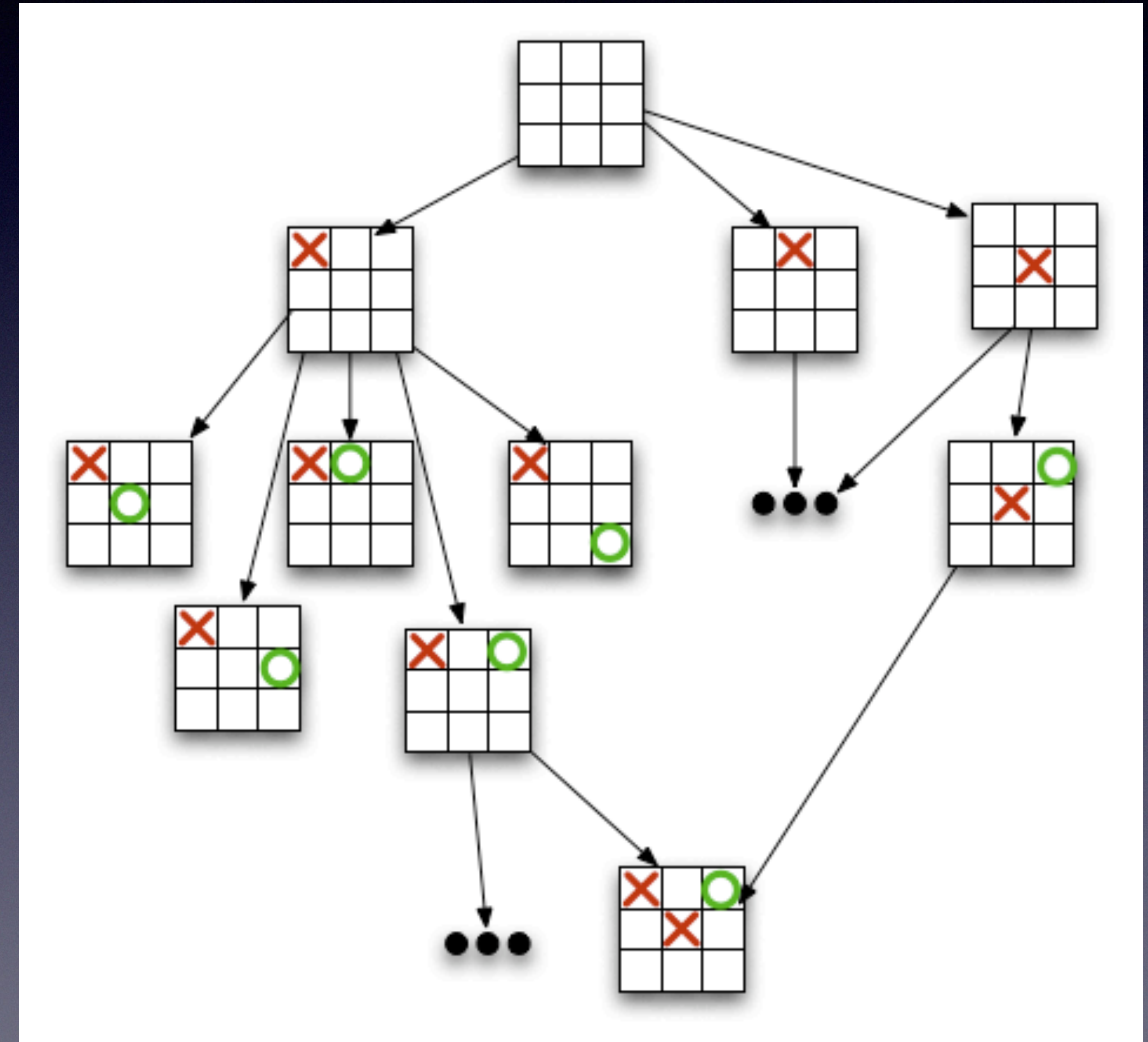


Búsqueda



# Problema de búsqueda básico 1

- Imagina un espacio de estados como un terreno por el cual nos podemos mover
- Partimos de un punto concreto del terreno (estado inicial) y podemos aplicar las reglas válidas para ir saltando de un estado a otro
- Se puede identificar la resolución del problema (llegar hasta un estado final válido) con el problema de buscar un camino adecuado entre un estado inicial y un estado final.
- Es por ello que muy habitualmente se habla del Problema de Búsqueda en el Espacio de Estados





# Problema de búsqueda básico 2

- Un problema de búsqueda básico es una 4-tupla  $(X, S, G, d)$ , donde:
- $X$  es un conjunto de estados (el que hemos estado llamando Espacio de Estados).
- $S \subseteq X$ , es un conjunto no vacío de estados iniciales (no tiene por qué existir un solo estado de partida).





# Problema de búsqueda básico 3

- $G \subseteq X$ , es un conjunto no vacío de estados finales (G se usa como inicial de Goals, objetivos en inglés, y al igual que no existe un único estado de partida, puede ocurrir que haya muchos estados finales válidos para el mismo problema).
- $d: X \rightarrow P(X)$  es una función de transición. Para cada  $x \in X$ ,  $d(x)$  determina el conjunto de estados sucesores de  $x$  ( $P(X)$  representa las partes de  $X$ , es decir, el conjunto de los posibles subconjuntos de  $X$ , ya que desde un estado concreto podemos llegar a varios posibles estados aplicando distintas operaciones o reglas permitidas).





# Métodos de búsqueda



# Tipos de búsquedas

- Algoritmo de búsqueda es un conjunto de instrucciones que están diseñadas para localizar un elemento con ciertas propiedades dentro de una estructura de datos

## Tipos de Métodos de Búsqueda Lineal

### Directos

- Gradiente
- Newton
- Quasi-Newton
- Secante

### Interpolación Polinómica

- Cuadrática
- Cúbica
- DSC (Davies, Swann y Campey)

### • Basados en intervalos

- Bisección
- Búsqueda de Fibonacci
- Búsqueda Dorada

### • Métodos Inexactos

- Armijo
- Goldstein



# Búsqueda informada vs No informada

- Un problema típico de la Inteligencia Artificial consiste en buscar un estado concreto entre un conjunto determinado, al que se le llama espacio de estados.
- Imaginemos, por ejemplo, una habitación en la que hay un libro. Un robot se desea desplazar por la habitación con el fin de llegar a dicho libro. ¿De qué manera lo hará?
- En este punto es donde entran en juego las estrategias y los algoritmos de búsqueda.





# Algoritmos de búsqueda

- Algoritmos no informados o ciegos: en general más ineficientes en tiempo y memoria que otros métodos.
- Algoritmos informados
- Algoritmos heurísticos: destacan las Búsquedas Primero el Mejor (Algoritmo voraz o Greedy y Algoritmo de búsqueda A\*) y de Mejora Iterativa (Algoritmo Escalada Simple -Hill Climbing- y Escalada por Máxima Pendiente)
- Algoritmos de Búsqueda con adversario: destacan el Minimax y el Poda alfa-beta.

## Procedimiento Búsqueda Local

```
s = genera una solución inicial
while s no es ótimo local do
     $s' \in N(s)$  con  $f(s) < f(s')$ 
    (solución mejor dentro de la vecindad de s)
     $s \leftarrow s'$ 
end
return s
```



# Complejidad en juegos



# ¿Cómo la medimos?

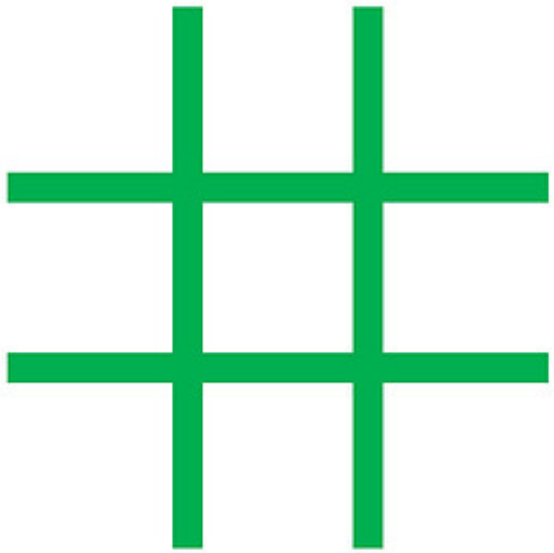
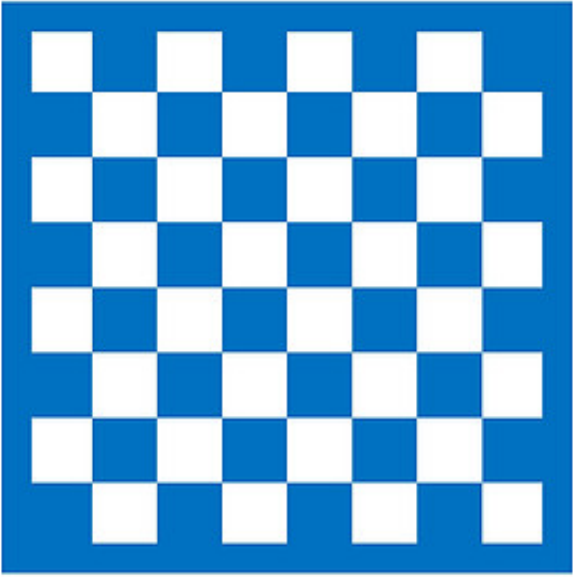
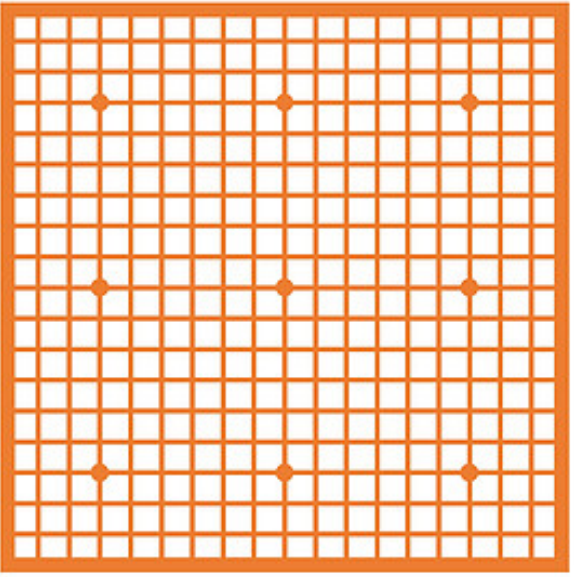
1. Complejidad estado-espacio
2. Tamaño del árbol del juego
3. Complejidad de las decisiones
4. Complejidad del árbol del juego
5. Complejidad computacional





# Complejidad del estado-espacio

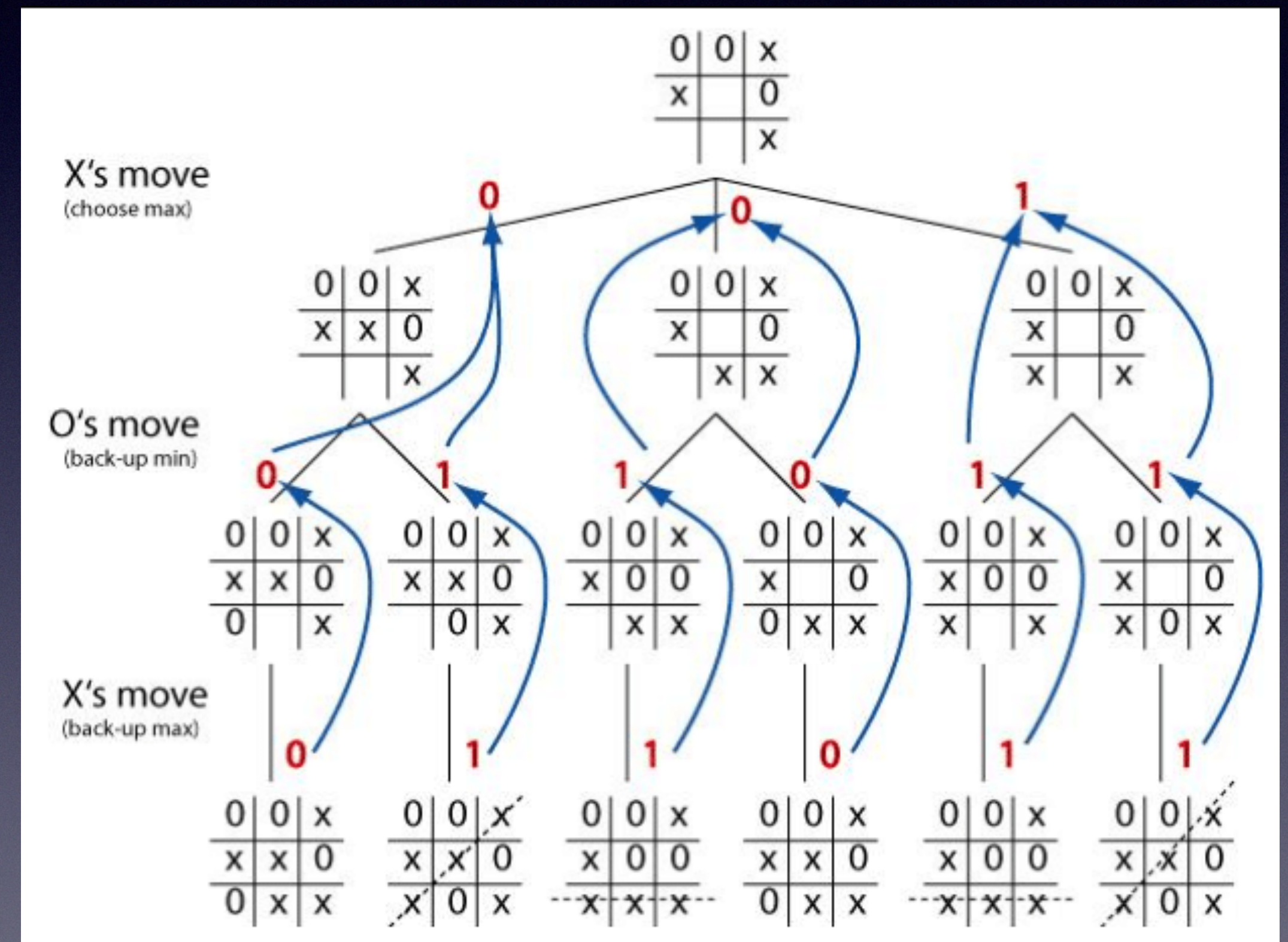
- La complejidad del estado-espacio de un juego es el número de posiciones legales del juego accesibles desde la posición inicial del juego

STATE SPACE COMPLEXITY		
TIC-TAC-TOE	CHESS	GO
		
4	43	172
(TEN THOUSAND)	(TEN TREDECILLION)	(TEN SEXQUINQUAGINTILLION)



# Tamaño del árbol del juego

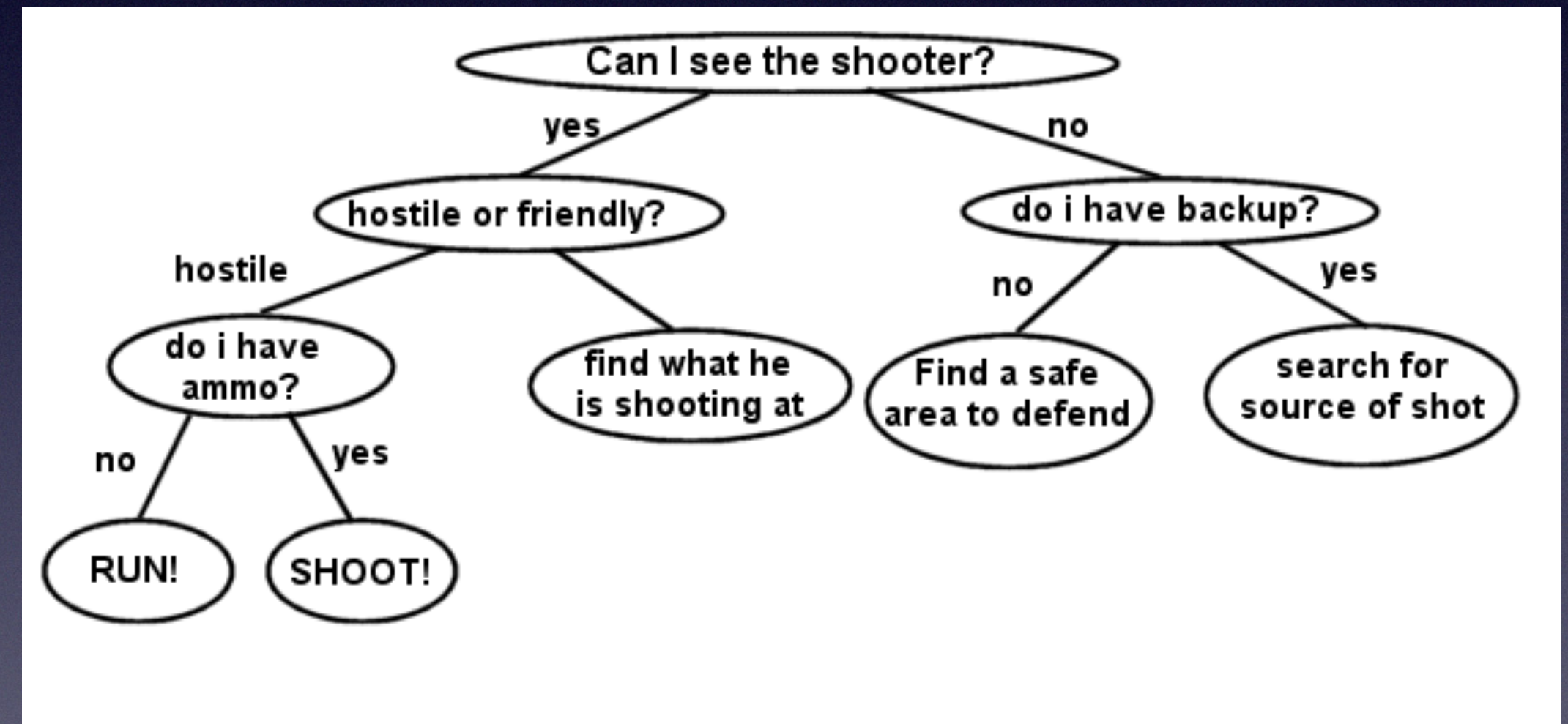
- Es el número total de las instancias posibles que puedan ser aplicadas al juego
- El número de los nodos hoja en el árbol del juego cuya raíz es la posición inicial.
- Un límite superior para el tamaño del árbol del juego puede ser computado a veces, simplificando el juego de manera que aumente sólo el tamaño del árbol del juego





# Complejidad de las decisiones

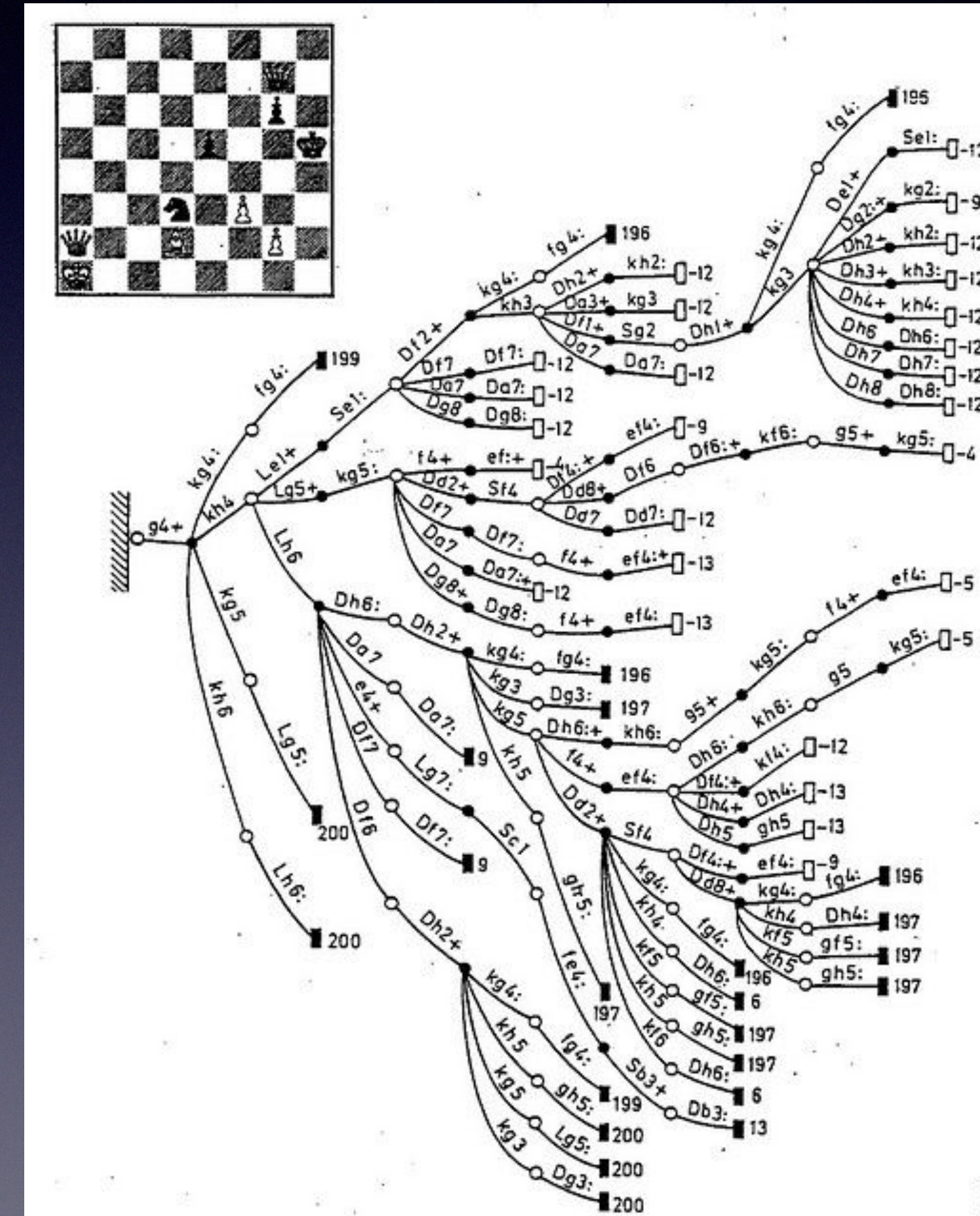
- Es el número de nodos hoja en el árbol de decisión más pequeño que establece el valor de la posición inicial.
- Todo árbol incluye todas las posibles decisiones para el jugador que mueve en segundo lugar, pero solo una posibilidad para cada decisión del jugador que comienza la partida.





# Complejidad del árbol del juego

- Es el número de nodos hoja del árbol de decisión completo en anchura que establece un valor en la posición inicial.
- Un árbol completo en anchura incluye todos los nodos de todos los niveles
- Ésta es una estimación del número de posiciones que podríamos tener que evaluar en una búsqueda usando el algoritmo minimax para determinar el valor de la posición inicial.
- Es difícil estimar la complejidad del juego-árbol, pero para algunos juegos un límite inferior razonable puede darse elevando el número de hijos medio de cada nodo al número de turnos que hay en el juego medio.





# Complejidad computacional

- El coste computacional de un juego describe la dificultad asintótica de un juego que crece de manera arbitraria, expresada en notación de cota superior o como miembro de una clase de complejidad.
- Este concepto no es aplicable a juegos concretos, pero sí a juegos que han sido formalizados para que puedan ser aleatoriamente grandes, generalmente jugándolos en un tablero de  $n$  por  $n$ .
- Desde el punto de vista de la complejidad computacional de un juego en un tablero de tamaño fijo, es un problema finito que se puede resolver en  $O(1)$ , por ejemplo, con una tabla de búsqueda de posiciones en el mejor movimiento en cada posición).





# Práctica de búsquedas