

# Integración de Bases de Conocimiento Datos

2do Cuatrimestre de 2020

Clase 2: Agente Inteligentes



DEPARTAMENTO  
DE COMPUTACION

Facultad de Ciencias Exactas y Naturales - UBA

Profesora: Vanina Martinez

[mvmartinez@dc.uba.ar](mailto:mvmartinez@dc.uba.ar)

# Inteligencia Artificial

- Tomaremos la siguiente como definición operativa de Inteligencia Artificial:  
*“ ... es el estudio del diseño e implementación de **Agentes Inteligentes** y la o las teoría asociada con esta actividad.”*
- Surgen dos ámbitos en relación a la noción de agente:
  - ❖ El Agente en sí mismo, y los
  - ❖ Conjuntos de Agentes
- Tendremos computación *“Personal”* en el agente individual y una computación *“Social”* para conjuntos de agentes.



# ¿Qué es un Agente?

- Son entidades, *físicas* o *virtuales*, *activas* y *persistentes* que:
  - *perciben*
  - *razonan*
  - *actúan* y ...
  - *se comunican*
- Adicionalmente, se sugiere que deben poseer *autonomía* y tener *metas propias*.

# Características Generales

- Son capaces de *actuar* sobre el entorno.
- Pueden *comunicarse* con otras entidades similares.
- Son capaces de *percibir* (en forma limitada) el entorno en el que desarrollan su actividad.
- Poseen una *representación* parcial del entorno que depende de propios *sensores* directos y de las *comunicaciones* que posiblemente establece con otros agentes.
- Poseen un conjunto de *tendencias* que gobiernan su comportamiento, que pueden tomar la forma de objetivos y/o funciones de supervivencia o de satisfacción de objetivos.



# Características Generales

- Su comportamiento tiende a la *satisfacción de objetivos*, teniendo en cuenta *sus habilidades* y *recursos disponibles* y *dependiente de sus percepciones, comunicaciones* y su *representación*.
- Poseen *recursos* propios, i.e. recursos de los que solo el agente puede disponer.
- Es fundamental es que su *comportamiento* tiende a lograr sus *objetivos* utilizando *habilidades* y *recursos propios*, o que se encuentran disponibles en su *entorno*.
- Poseen *habilidades* y puede ofrecer *servicios*.
- Es posible que posean la habilidad de *reproducirse*.

# Noción de Agencia

Una *noción débil* de Agencia incluye:

- *Autonomía*.
- *Habilidad social* a través de un *lenguaje de comunicación* entre agentes.
- *Reactividad* (en alguna medida).
- *Pro-actividad*

# Noción de Agencia

Una *noción fuerte* de Agencia que involucra actitudes mentales tales como:

- *Conocimiento y Creencias*
- *Intencionalidad*
- *Reconocimiento de sus Obligaciones*
- *Consideración de las Emociones*

# Cuestiones Fundamentales

---

- *Teorías de Agencia*
- *Arquitecturas de Agentes*
- *Lenguajes de Agentes*



# Teorías de Agencia

- Teorías de Agencia: es fundamentalmente una especificación que responde a la pregunta de **como conceptualizar a un agente en términos de sus propiedades y la representación de las mismas**.
- *Teorías de Agencia*
  - ¿Qué es un agente?
  - ¿Qué propiedades debe tener?
  - ¿Cómo se representan dichas propiedades formalmente?
  - ¿Cómo se razona acerca de ellas?

# Teorías Propiedades: Intensionalidad

- Un sistema **intencional** es una entidad cuyo comportamiento puede ser predicho atribuyéndole creencias, deseos y capacidad de razonamiento.
- Existen **grados** de intencionalidad.
- Un sistema Intencional de primer orden **tiene creencias y deseos** pero **no creencias y deseos acerca de creencias y deseos**.
- Un Sistema Intencional de segundo orden **tiene creencias y deseos acerca de creencias y deseos (propios y de otros agentes)**.

# Teorías Propiedades: Actitudes

- ¿Cuáles son las actitudes más interesantes para la representación de agentes?
- Dos de las más importantes son las denominadas *Actitudes Informacionales* y las *Pro-actitudes*.
- Actitudes Informacionales: *Creencia* y *Conocimiento*.
- Pro-actitudes: *Deseos*, *Intencionalidad*, *Obligación*, *Compromiso*, *Selectividad*.

# Arquitecturas

- Representan el movimiento desde la especificación hacia la implementación y responden a la pregunta de **como construir un agente**.
- *Arquitecturas de Agentes*
  - ¿Cómo deben construirse los agentes de manera que posean las propiedades que deseamos?
  - ¿Qué estructuras de Software, o Hardware, son necesarias para soportar esta arquitectura?

# Arquitecturas

- Uno de los problemas clásicos en el diseño de una arquitectura es balancear la parte de *percepción / acción* y el razonamiento sobre como actuar, (*i.e.*, control de crisis vs. acción cuidadosa).
- Si el razonamiento domina a la percepción tenemos una *Arquitectura Deliberativa*.
- Si la percepción domina al razonamiento tenemos una *Arquitectura Reactiva*.

•Tacto

Módulo de sensores y control de efectores

•Olfato

•Gusto

•Lenguaje

•Audición

•Emociones

•Vista

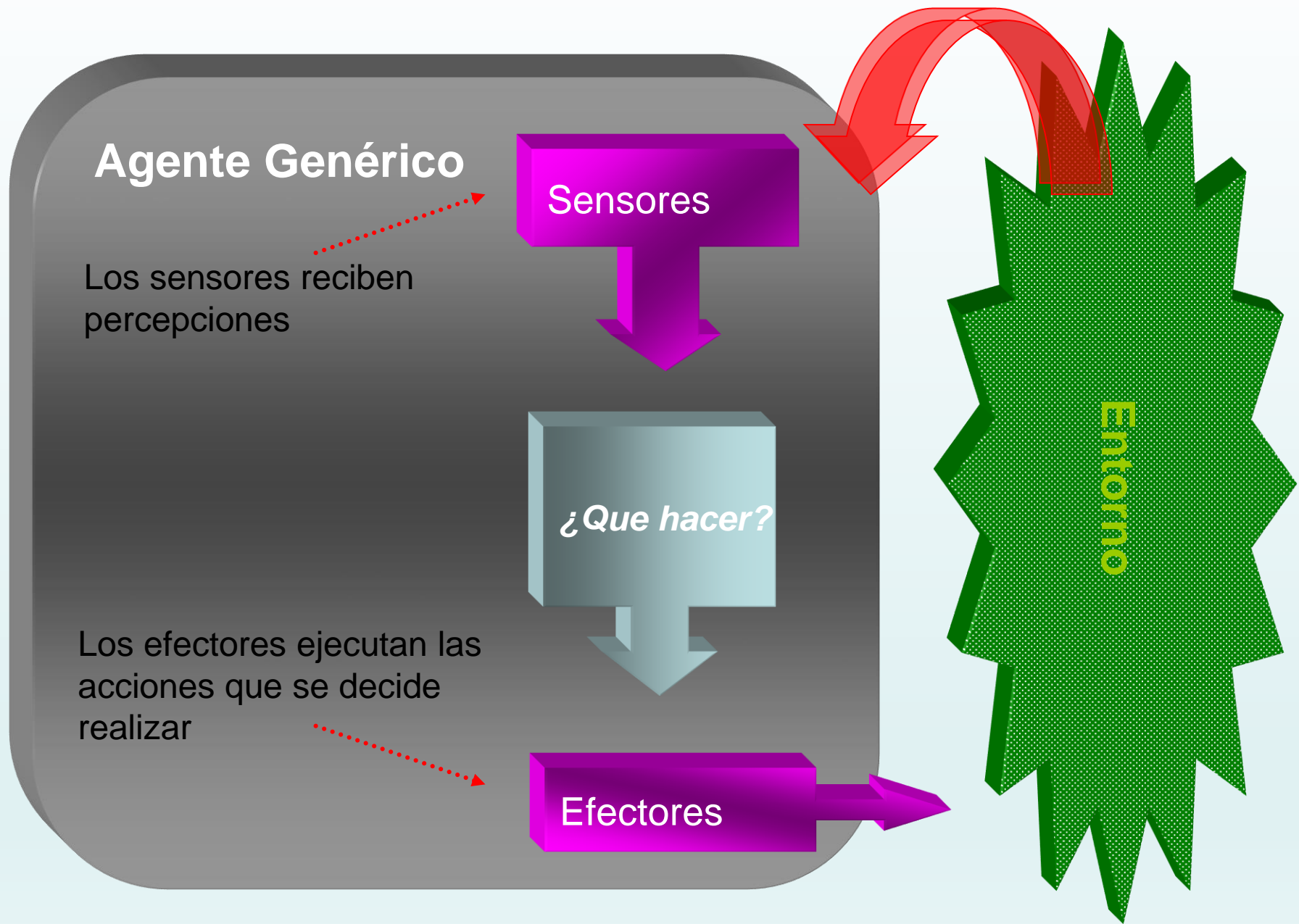
•Comportamiento  
Cognitivo

•Control  
Muscular

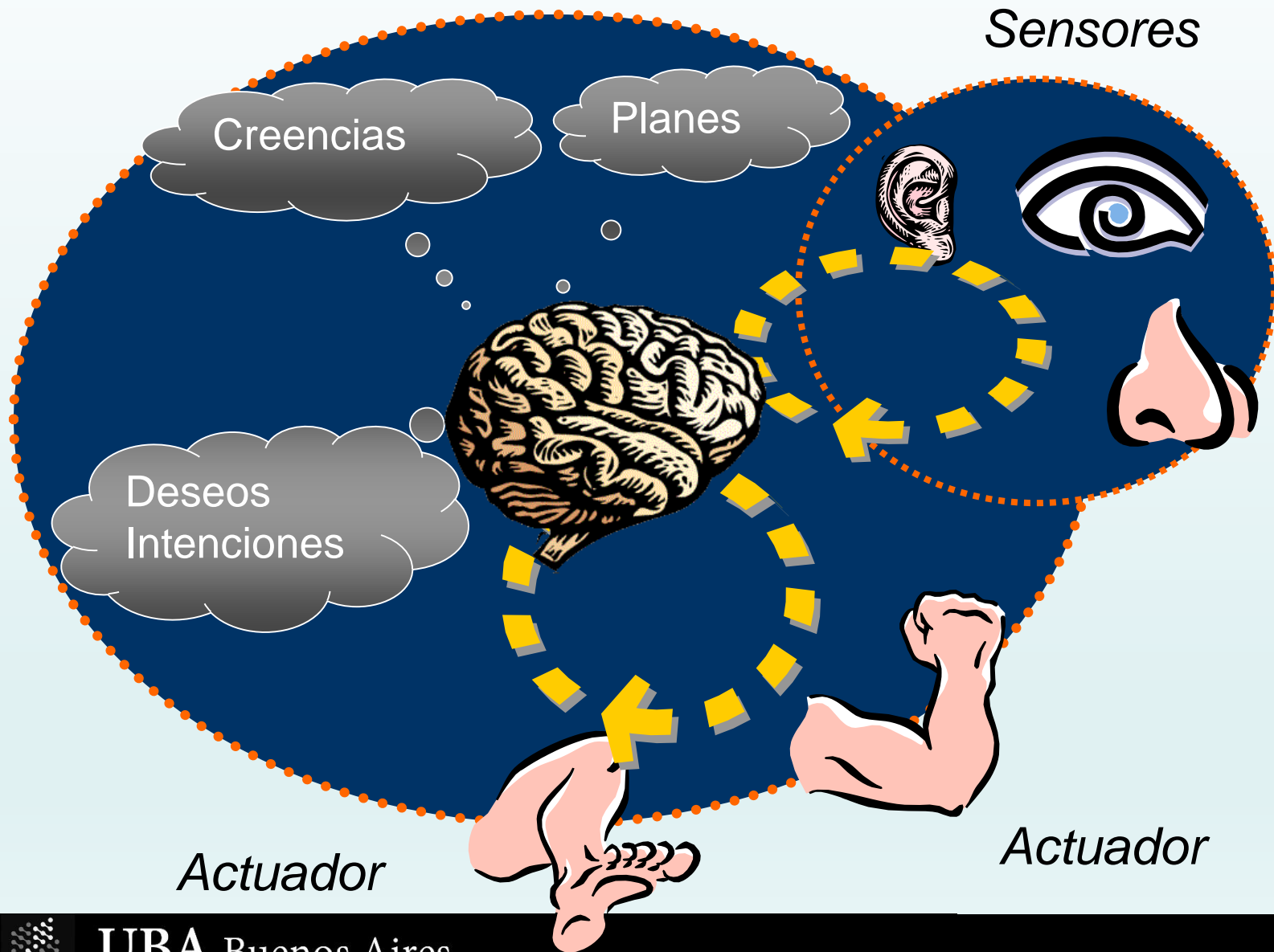
•Pensamiento

•Memoria



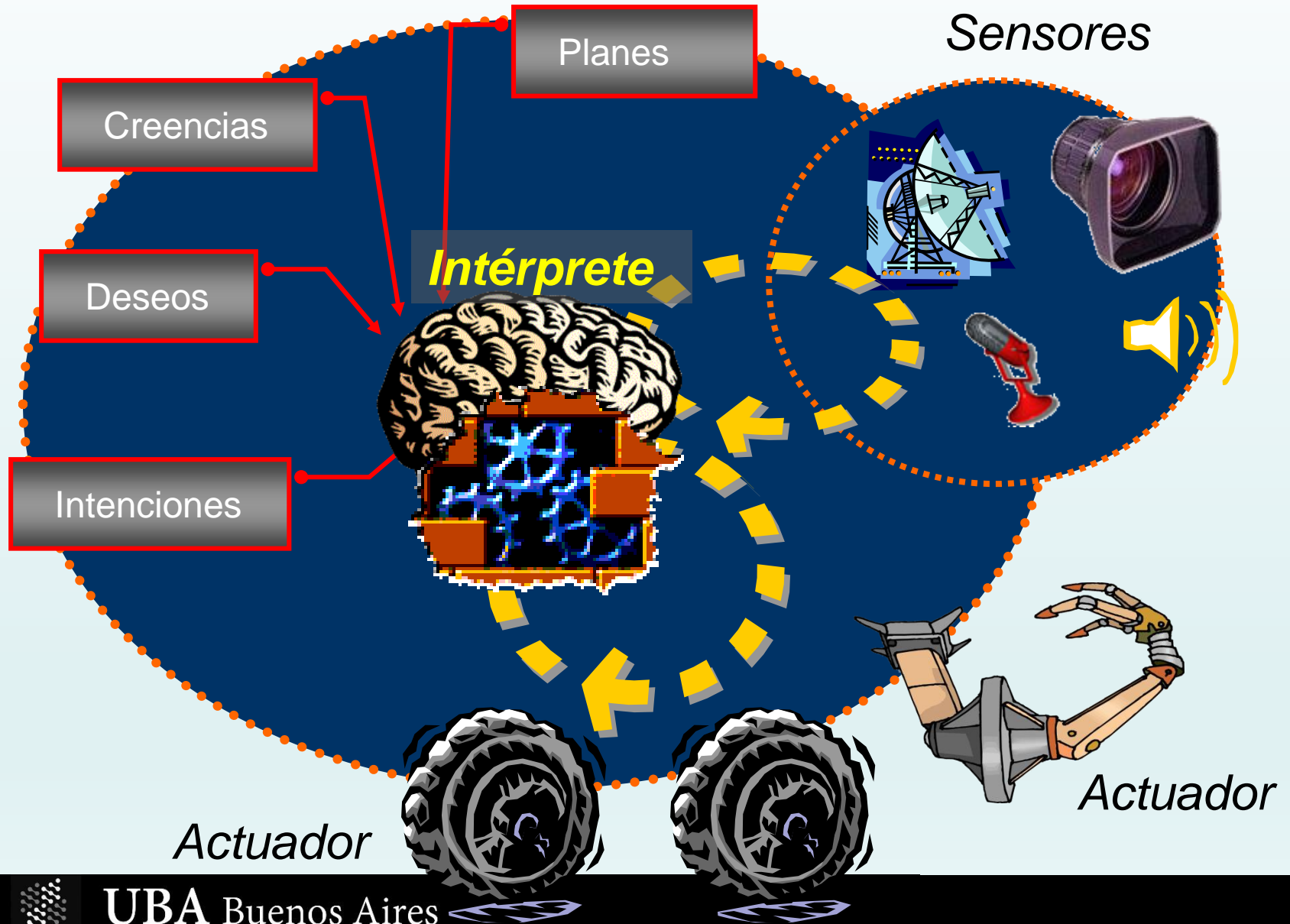


# Arquitectura





# Arquitectura



# Agente Reactivo Simple (Reflejo)

```
function simple-reflex-agent(percept)
```

```
  returns an action
```

```
static: rules, a set of condition-action  
rules
```

```
state  $\leftarrow$  interpret-input(percept)
```

```
rule  $\leftarrow$  rule-match(state, rules)
```

```
action  $\leftarrow$  rule-action(rule)
```

```
return action
```

Reglas  
Condición-acción

Sensores

Observaciones  
acerca del mundo

¿Que hacer?

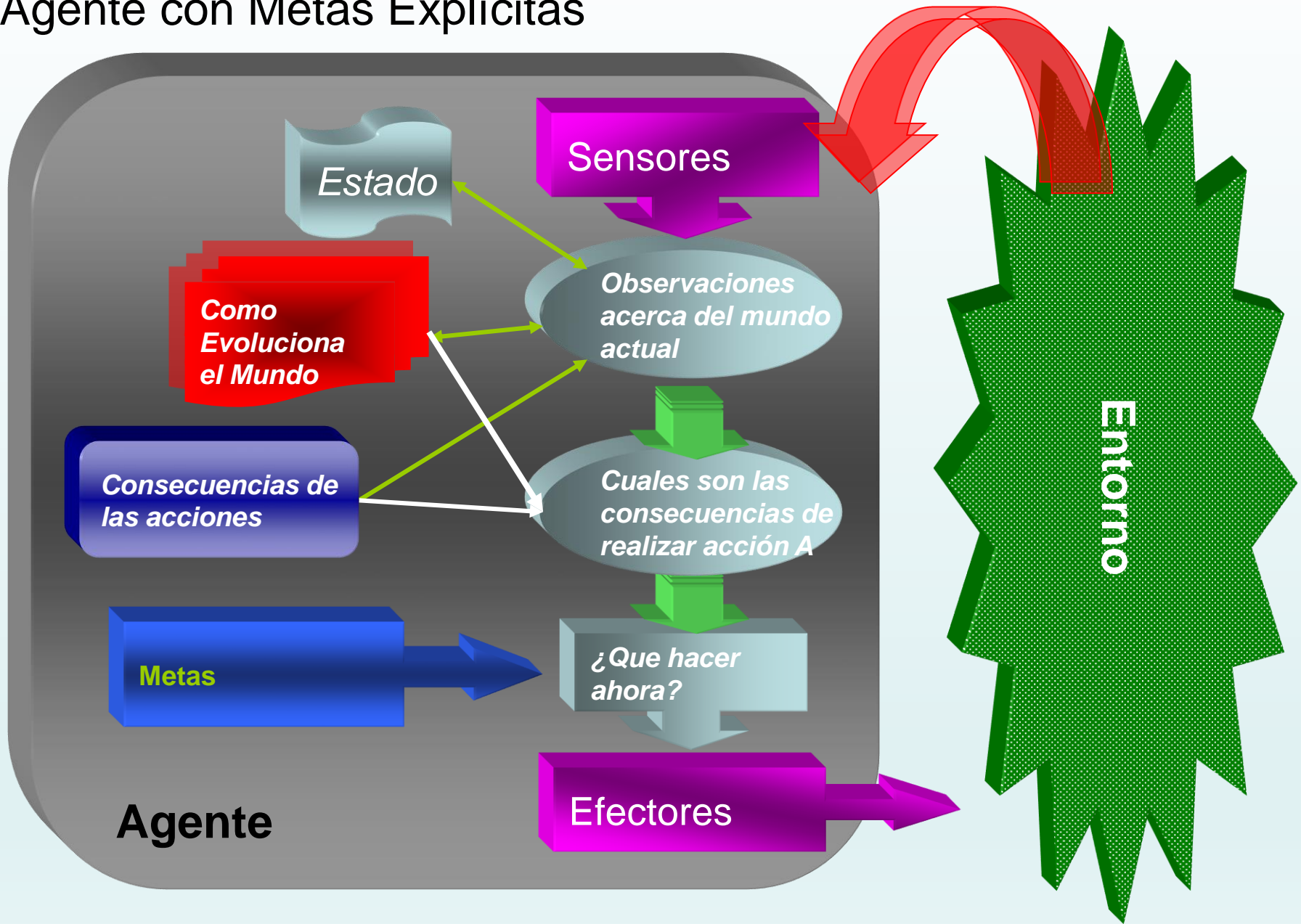
Efectores

Entorno

# Agente reactivo con estado interno



# Agente con Metas Explícitas





# Agente basado en Utilidades



# Cuestiones Fundamentales

- *Lenguajes de Agentes*
  - ¿Cómo deben programarse estos agentes?
  - ¿Cuáles deben ser las primitivas para esta tarea?
  - ¿Cómo es posible hacer que estos lenguajes provean un marco efectivo?

# Agentes y Objetos

# Agentes: Dos Aspectos de Autonomía

*Autonomía Dinámica:* Pueden decidir actuar, Reactivo vs.

Proactivo, determinada por la estructura interna del Agente.

*Autonomía No Predecible:* Pueden decir NO, Predecible vs.

Impredecible, determinado por un observador externo.

- La metáfora de Agencia asigna **características antropomórficas** al ámbito de las unidades computacionalmente hábiles: *metas, propósitos e intenciones*.
- Permite lograr una ventaja de diseño para el estudio de los *sistemas complejos*.



# Objetos

- Pensarlas como *entidades computacionales* que *encapsulan un estado*, que *pueden realizar acciones* (o ejecutar métodos) sobre ese estado y que *se comunican* por medio de mensajes.
- Encapsulados, *objetos tienen cierto control sobre su estado*.
- Sin embargo, si un objeto tiene métodos públicos estos pueden ser invocados por otros objetos del sistema y en tal sentido pierde el control sobre su comportamiento.
- Esta característica permite diseñar un sistema que, por construcción, tiene una *meta* que *todos* los objetos colaboran a lograr y por la que están incluidos en tal sistema.

# Agentes y Objetos

- La *computación basada en objetos es interactiva*, pero los objetos carecen de ciertas propiedades deseables con respecto a la forma como se produce la interacción.
- Agregar esas propiedades representa un paso evolutivo natural del modelo computacional orientado a objetos al que se le agregan las nuevas características.
- Sin embargo, no es una extensión ya que se pueden perder algunas propiedades, por ejemplo *el determinismo*.

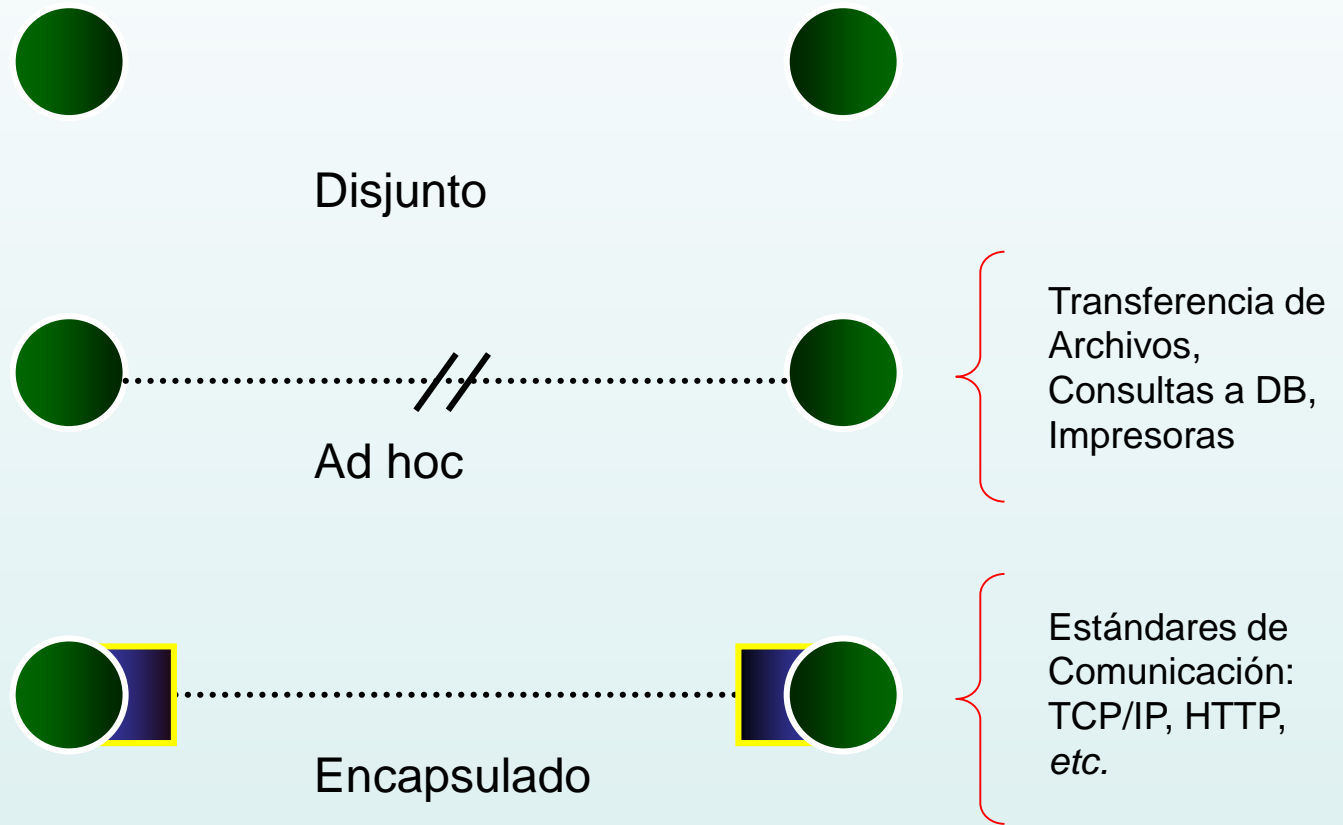
# Agentes vs. Objetos

- Los Agentes poseen una *noción más fuerte de autonomía* al poder elegir si ejecutar o no una acción a solicitud de otro agente.
- Los Agentes son capaces de gran flexibilidad en su comportamiento que puede ser *reactivo, proactivo, social*.
- En el modelo genérico de objetos no existe nada similar.
- Un Sistema Multi-Agente es inherentemente multi-hebra (*multi-threaded*) dado que cada agente posee al menos una hebra de control.

# Evolución

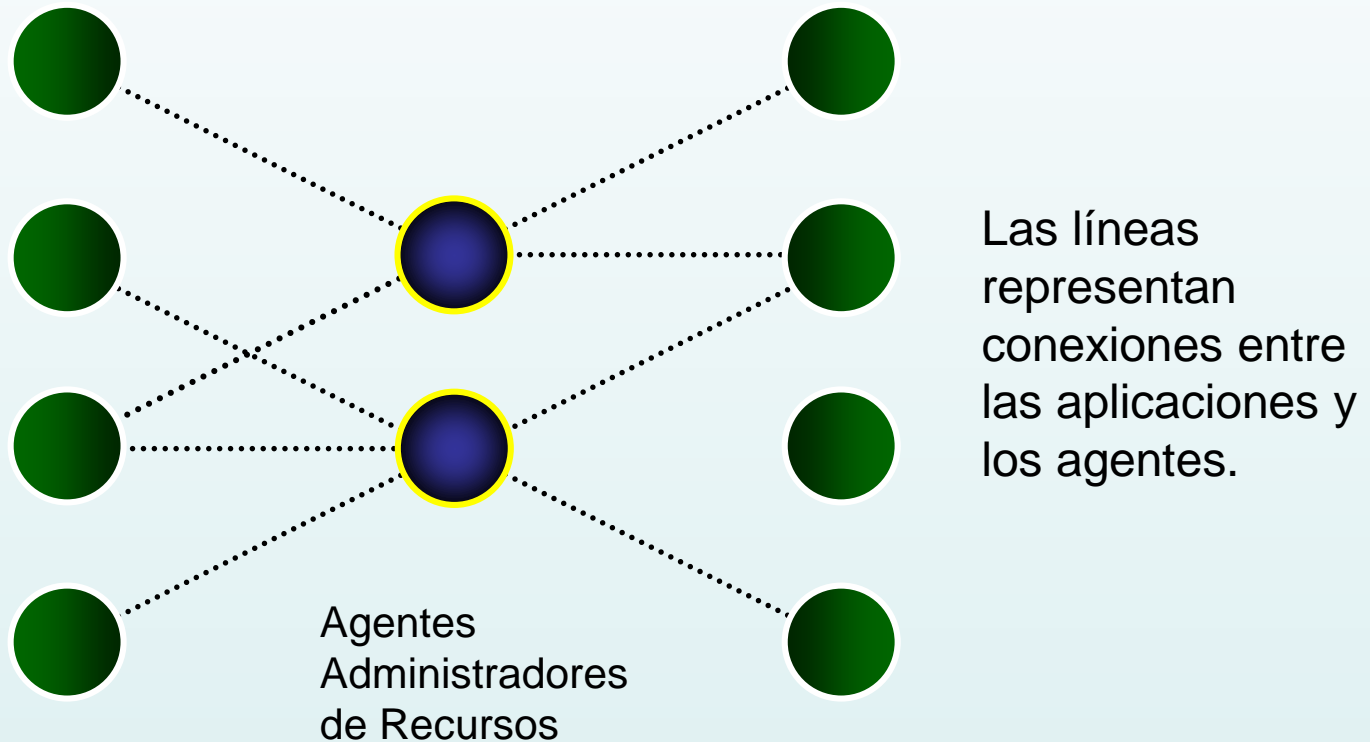
- El trabajo de investigación y desarrollo en Inteligencia Artificial Distribuida evolucionó de manera natural hacia los *Sistemas Multi-agente*.
- Esto estuvo motivado fundamentalmente por la creciente sofisticación lograda en los elementos distribuidos (las fuentes de conocimiento) que participan en la resolución de un problema.
- El foco de la investigación actual está en formalizar la interacción de una colección de agentes para resolver un problema en común.

# Brodie, M.L. 1989, *Future Intelligent Information Systems: AI and Database Technologies Working Together.*



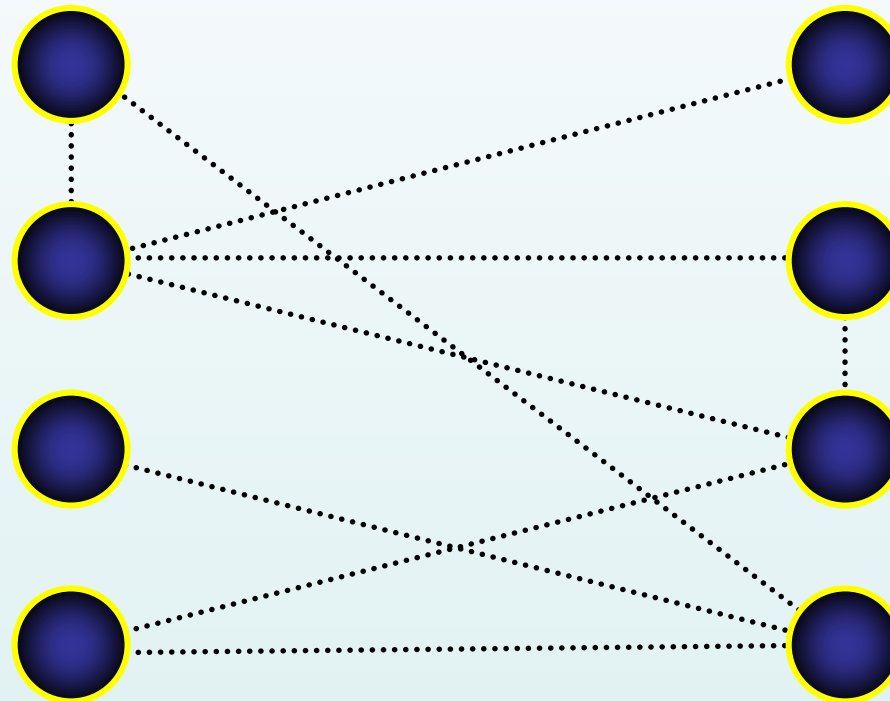
La Evolución en la Conectividad de Sistemas  
(Sin Cooperación)

Brodie, M.L. 1989, *Future Intelligent Information Systems: AI and Database Technologies Working Together.*



La Evolución en la Conectividad de Sistemas  
(Agentes como Planificadores Centralizados)

Brodie, M.L. 1989, *Future Intelligent Information Systems: AI and Database Technologies Working Together.*



Las líneas representan conexiones activas entre los agentes.

La Evolución en la Conectividad de Sistemas  
(Sistemas que Cooperan como Agentes)

# Sistemas Multi-Agente ( MAS )

- En los Sistemas Multi-Agente la investigación y las aplicaciones se concentran en la *coordinación del comportamiento inteligente* entre una colección de *agentes* inteligentes y autónomos (posiblemente pre-existentes).
- Entre los problemas que aparecen está el de lograr la *coordinación de su conocimiento, sus metas, sus habilidades y planes* de manera tal de lograr actuar en acuerdo para resolver problemas.
- Los agentes en un *MAS* pueden trabajar hacia una meta única o hacia metas independientes cuya resolución puede requerir interacción.





# Sistemas Multi-Agente ( MAS )

- Al igual que en los módulos de un Sistema de Resolución Distribuida de Problemas, los agentes en un MAS deben compartir conocimiento sobre el problema y las posibles soluciones.
- Sin embargo, los agentes deben *razonar acerca del proceso de coordinación* entre el conjunto de los agentes.
- En estos sistemas la tarea de coordinación puede ser muy difícil dado que pueden existir situaciones en las que no existe la posibilidad de acceder a *“objetos de conocimiento”* globales.

# Sistemas Multi-Agente ( MAS )

- Estos “*objetos de conocimiento*” globales pueden incluir *Control Global*, *Consistencia Global del Conocimiento*, *Metas Globales Compartidas* o *Criterios Globales de Éxito*.
- La dificultad puede incluir la falta de una *Representación Global del propio Sistema*, lo que hace que razonar acerca del sistema sea muy complicado.
- Estos sistemas reciben comúnmente el nombre de *Sistemas Abiertos* (Open Systems).

# Sistemas Multi-Agente ( MAS )

- Los Sistemas Abiertos (Open Systems) se caracterizan por:
  - *Concurrencia* en el manejo de la información.
  - *Asincronía* entre los componentes y del sistema con el entorno.
  - *Control Descentralizado*, con decisiones locales.
  - *Información potencialmente inconsistente*.
  - *Relaciones cercanas entre los componentes*.
  - *Operación Continua*.

# Sistemas Multi-Agente

Son sistemas con los siguientes componentes esenciales:

- Existe un *entorno*  $E$  que define el espacio de la actividad.
- Un conjunto de *objetos*  $O$  situados en  $E$ .

Estos objetos pueden ser percibidos, creados, modificados o destruidos por los agentes.

- Un conjunto de *agentes*  $A$ , tal que  $A \subseteq O$ .
- Un conjunto  $R$  de *relaciones* en  $O$ .
- Un conjunto  $Op$  de *operaciones*.

Estas operaciones hacen posible que los agentes perciban, produzcan, consuman y manipulen objetos.

# Resolución de Problemas usando Búsqueda

# Resolución de Problemas

- Un agente inteligente puede resolver problemas considerando las diferentes **secuencias de acciones** que puede realizar.
- Un agente de **resolución de problemas**: exhibe comportamiento orientado a alcanzar metas particulares.
- Este tipo de agente debe tener una **Representación** adecuada de su entorno.
- Debe conocer las **Acciones** que tiene disponibles.
- Debe poder **Razonar** acerca del efecto de sus acciones sobre el entorno.

El estado del entorno es discreto...



# Resolución de Problemas

- El razonamiento en este caso se reduce a la consideración de las *acciones* y su *efecto* sobre el entorno en un tipo de simulación introspectiva.
- El agente elegirá la *secuencia de acciones* que lo lleve a una de las metas deseadas.
- La determinación de cual de las *metas posibles* tratará de alcanzar involucrará la idea de *costo*.
- El proceso de seleccionar la secuencia de acciones se denomina *Búsqueda*.



# Búsqueda: Definiciones

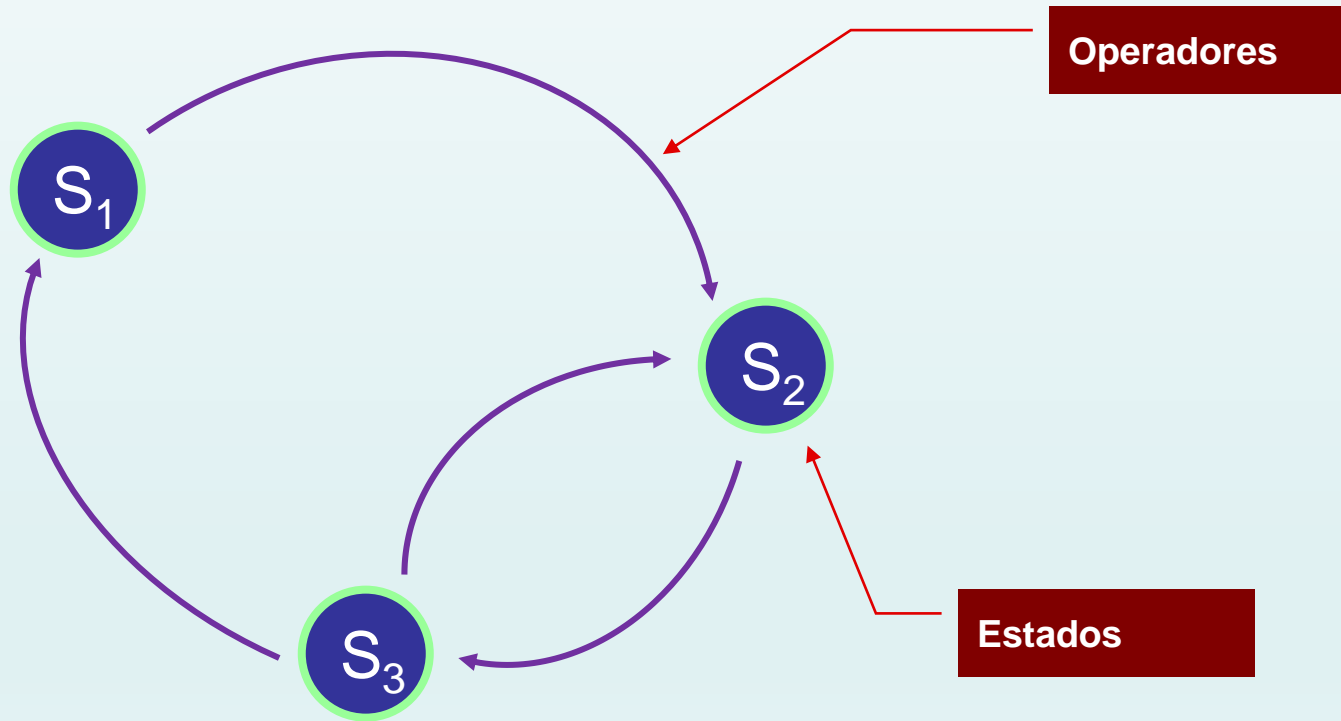
- Representación usando **Grafos**
- Propiedades de los **Grafos** y **Búsqueda**
- Métodos de **búsqueda a ciegas**:
  - Primero en profundidad (*Depth First*)
  - Primero a lo ancho (*Breadth First*)
  - Profundidad incrementada gradualmente (*Iterative Deepening*)
  - Ramificación incrementada gradualmente (*Iterative Broadening*)
  - Bi-direccional





# Resolución de Problemas y Búsqueda

- Se utilizan *Estados* y *Operadores* que transforman un estado en otro.



# Resolución de Problemas y Búsqueda

Una *Solución de un Problema* es una secuencia de operadores que transforman un *Estado Inicial* en un *Estado Meta*.



En general, ni el *Estado Inicial* ni el *Estado Meta* son necesariamente únicos, pero supondremos por convención que hay un solo estado inicial.

# Definiciones y Conceptos Básicos

- **Estado:** es una representación finita del mundo en un momento dado en un dado formalismo.
- **Operador:** una función que transforma un estado en otro (también se lo llama *regla*, *transición*, *función sucesor* *producción*, *acción*).
- **Estado inicial:** el estado del mundo al comenzar.
- **Estado Meta:** el estado deseado del mundo (puede no ser único).
- **Test de Meta:** test para determinar si se ha alcanzado la meta.

# Definiciones y Conceptos Básicos

- *Estado Alcanzable*: es un estado para el que existe una secuencia de operadores que partiendo de un estado inicial llega a él.
- *Espacio de Estados*: es el conjunto de todos los estados alcanzables desde el estado inicial.
- *Meta Alcanzable*: es una meta que es un estado alcanzable.
- *Función de Costo*: es una función que asigna un costo a la aplicación de los operadores.
- *Perfomance*: Costo de la secuencia de operadores, Costo de encontrar dicha secuencia.

# Grafo de búsqueda

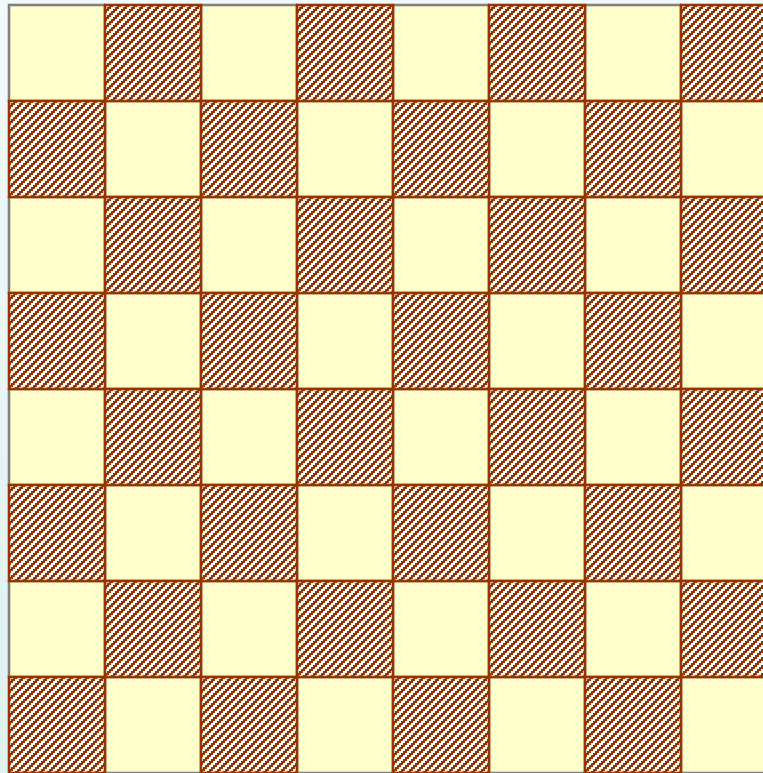
- Formado por *Nodos* que representan los *Estados* y por *Arcos* que representan la aplicación de los *Operadores*.
- Este Grafo se genera de manera *dinámica* a medida que la búsqueda avanza.
- Este grafo puede ser un *árbol* (grafo dirigido acíclico) o un *grafo genérico*.
- El número de estados posibles puede ser infinito.
- El *factor de ramificación* está determinado por el número de operadores que pueden aplicarse en cada estado.
- El *nivel de profundidad* de un nodo es el número de arcos desde el estado inicial hasta dicho nodo.

# Formulación del Problema

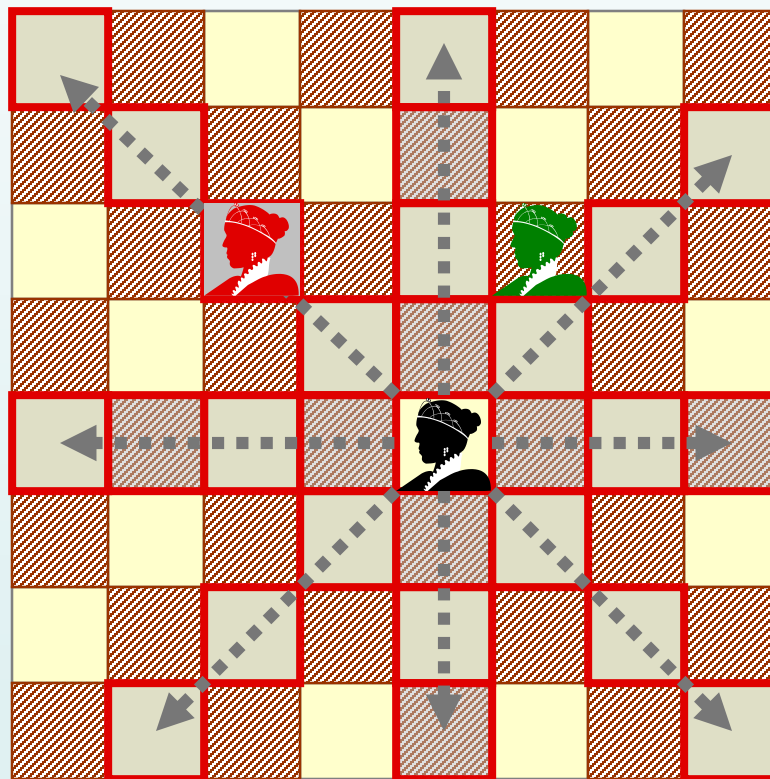
---

- La primera tarea es formular el problema en términos de **estados** y **operadores**.
- No siempre es posible lograr una formulación adecuada.
- En general, existen muchas formulaciones diferentes que dependen de la como se considera el problema.
- De manera que también la forma de plantear los problemas es extremadamente relevante.

# Ejemplo: 8 Reinas

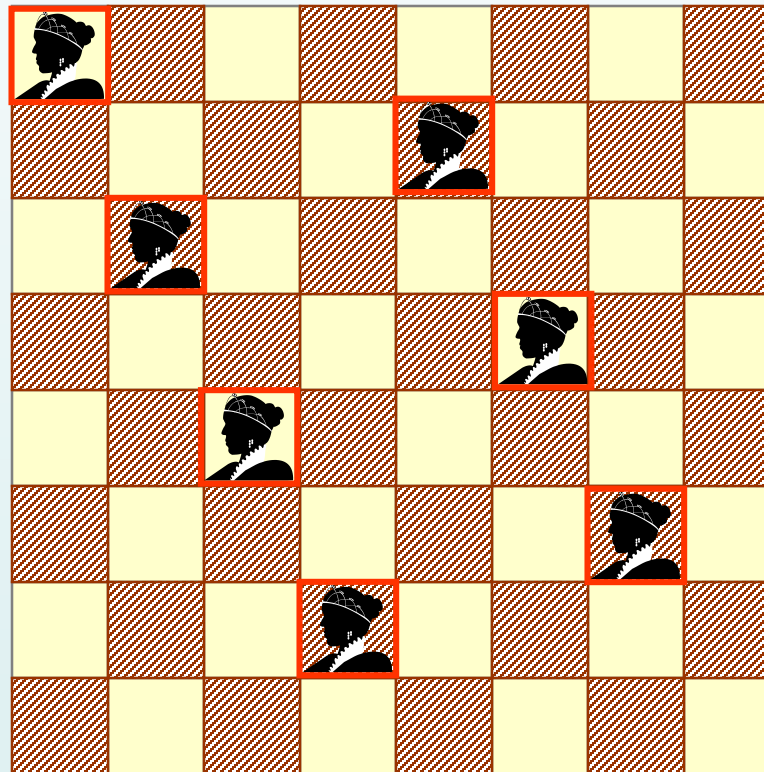


# Ejemplo

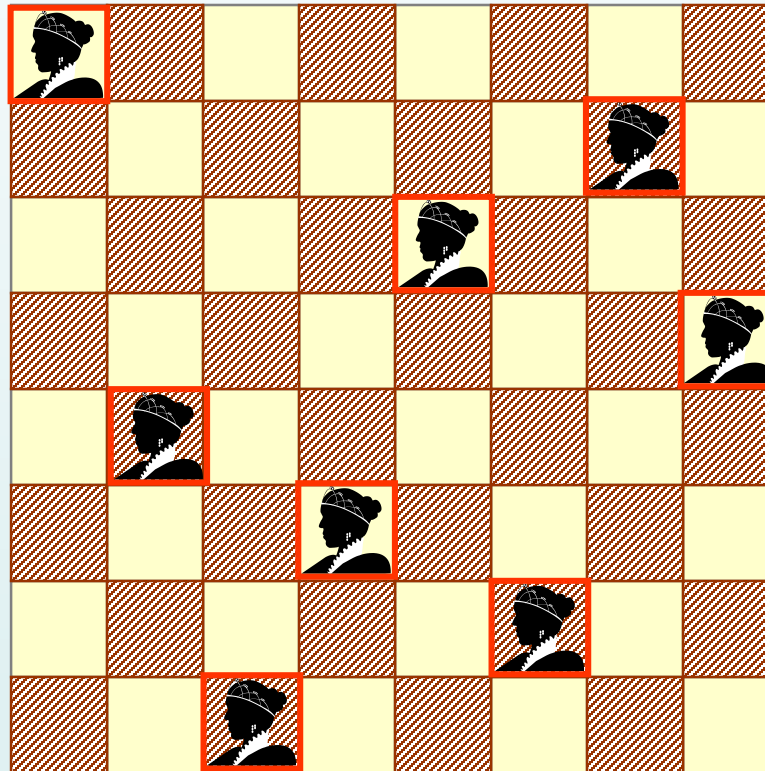




# Ejemplo: 8 Reinas

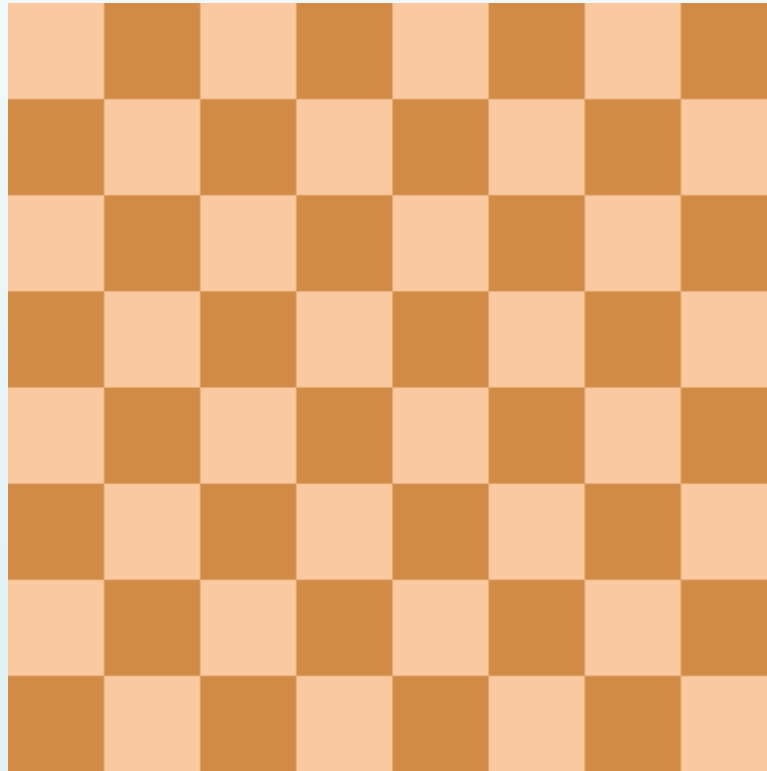


# Ejemplo: 8 Reinas



•Una solución

# Ejemplo: 8 Reinas



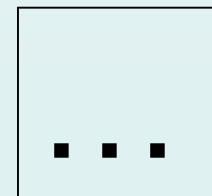
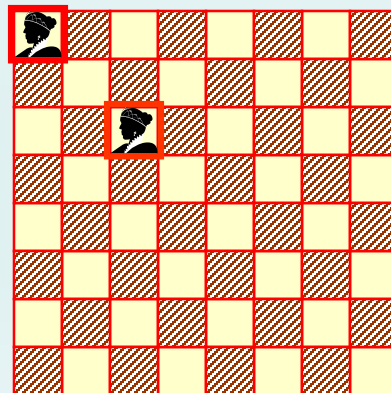
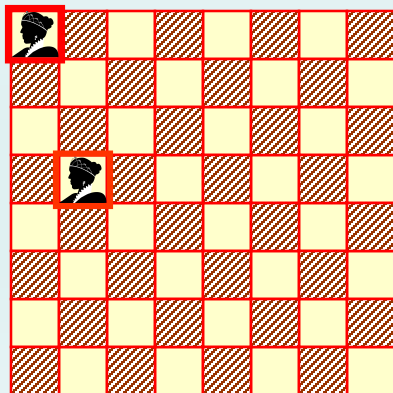
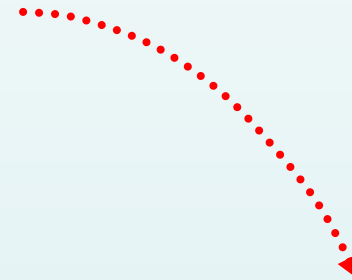
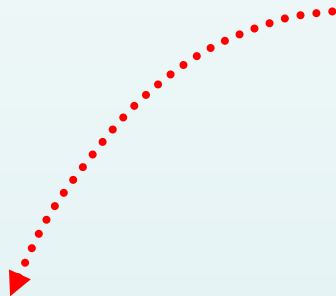
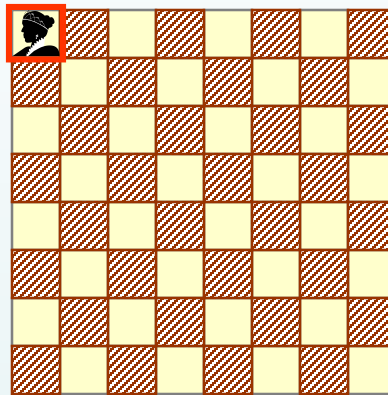
Encontrar una solución (fuerza bruta)

*Estado:* de 1 a 8  
reinas en el tablero

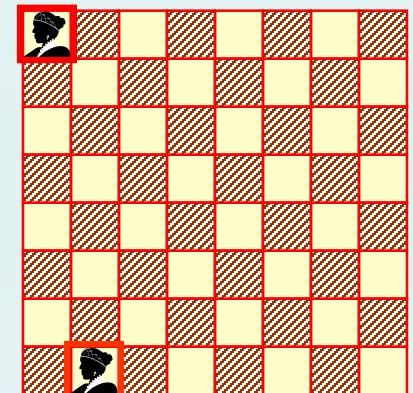
*Espacio de Estados:*  
todas las posibles  
configuraciones de 8  
reinas en el tablero  
( $64^8$ )

*Operador:* colocar 1  
reina en el tablero

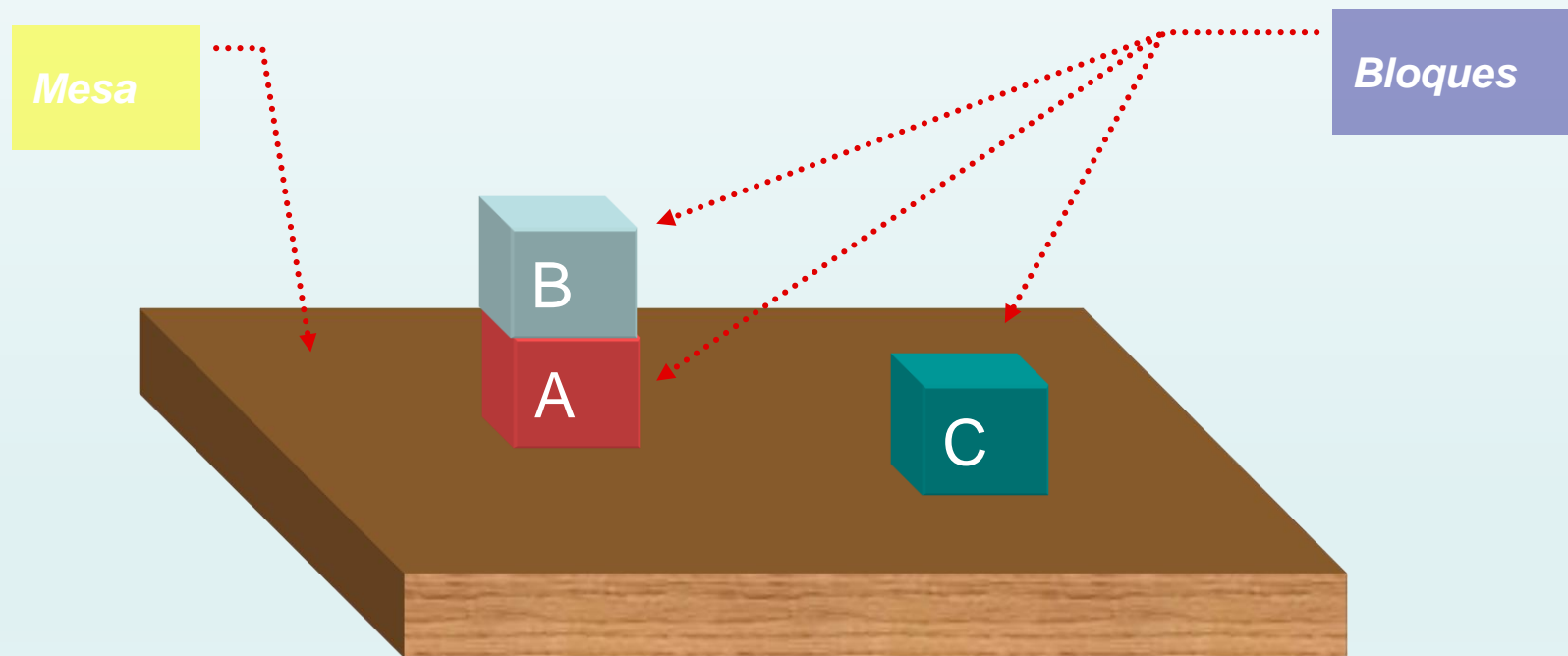
*Meta:* 8 reinas en el  
tablero sin jaque

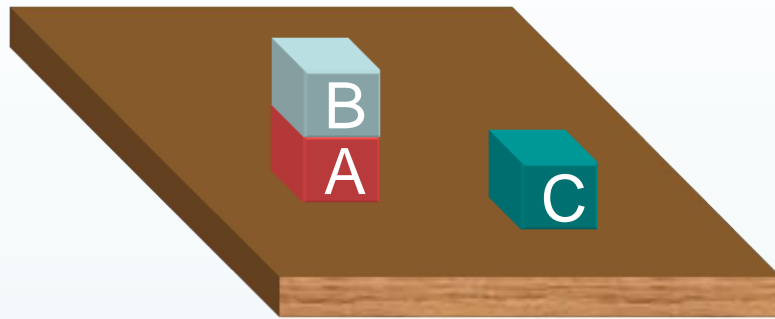


42  
sucesores

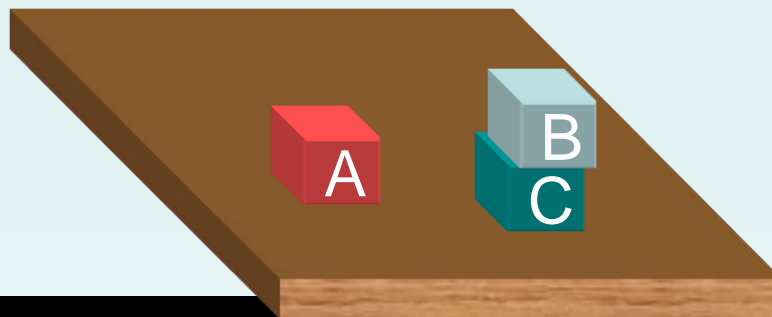
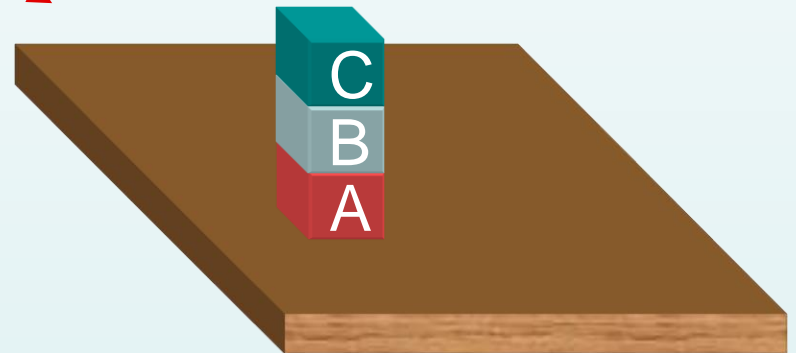
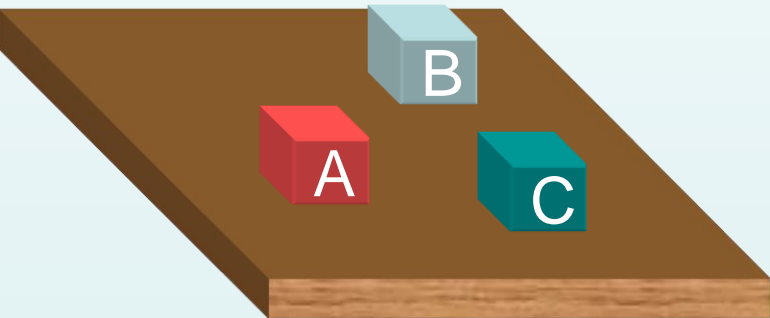
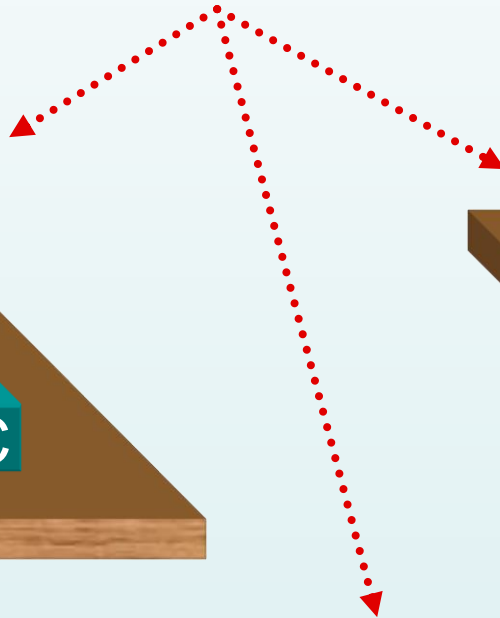


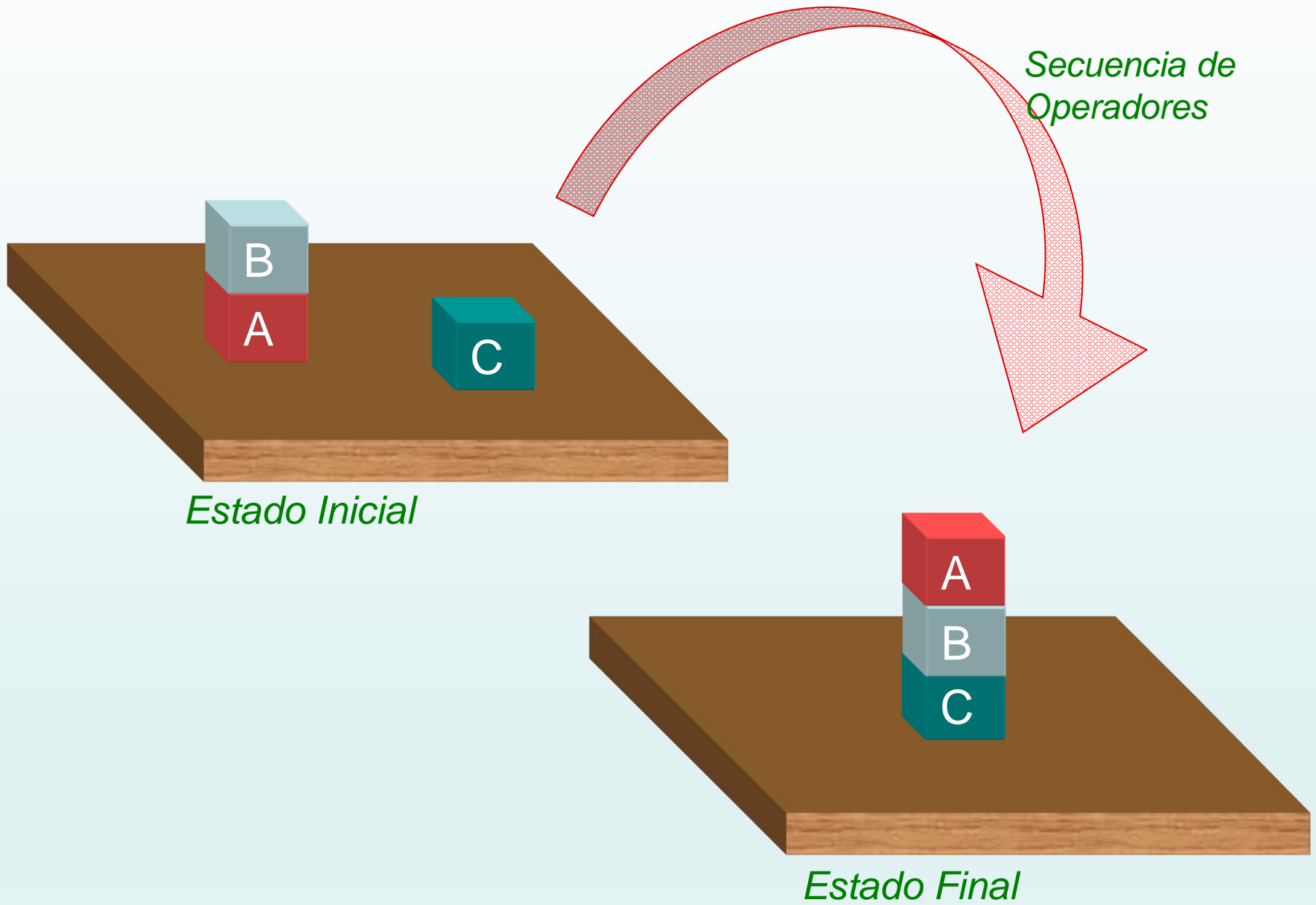
# Ejemplo: El Mundo de Bloques





*Posibles  
Movimientos*



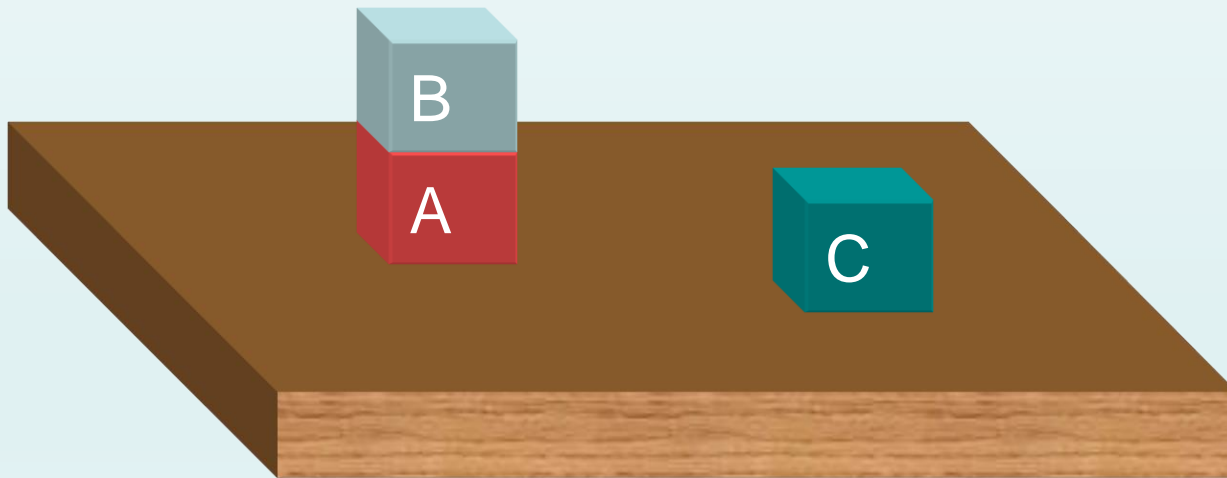


*Estado:* una configuración de los bloques sobre la mesa.

*Espacio de Estados:*  
todas las posibles  
configuraciones de los  
bloques sobre la mesa.

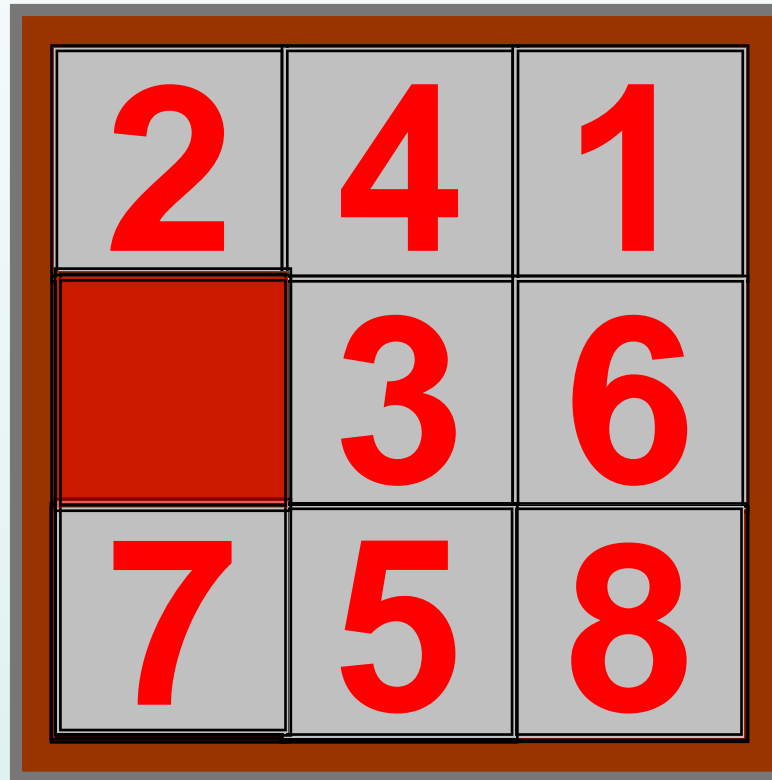
*Operadores:*  
reposicionamiento de los  
bloques sobre la mesa o  
sobre otro bloque, a partir  
de su posición sobre la  
mesa o sobre otro bloque.

*Meta:* un estado específico.





# 8-puzzle



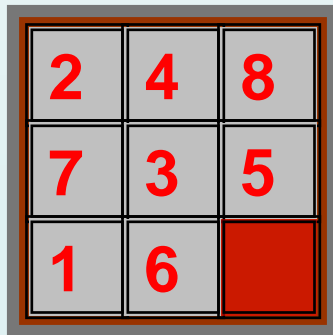
Un caso en donde la representación es importante

*Estado:* Cada uno de los posibles arreglos de los números en el cuadrado.

*Espacio de Estados:*  
Todas las posibles configuraciones.

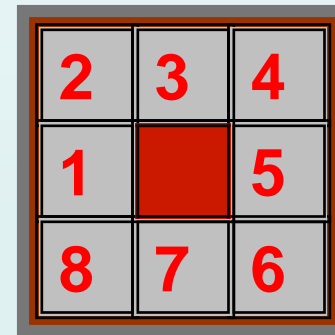
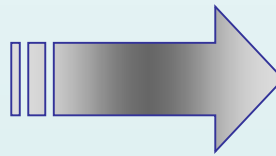
*Meta:* Un Estado específico.

*Operadores:* Deslizar un número al lugar vacío o...  
Intercambiar el lugar vacío con una de las posiciones adyacentes.



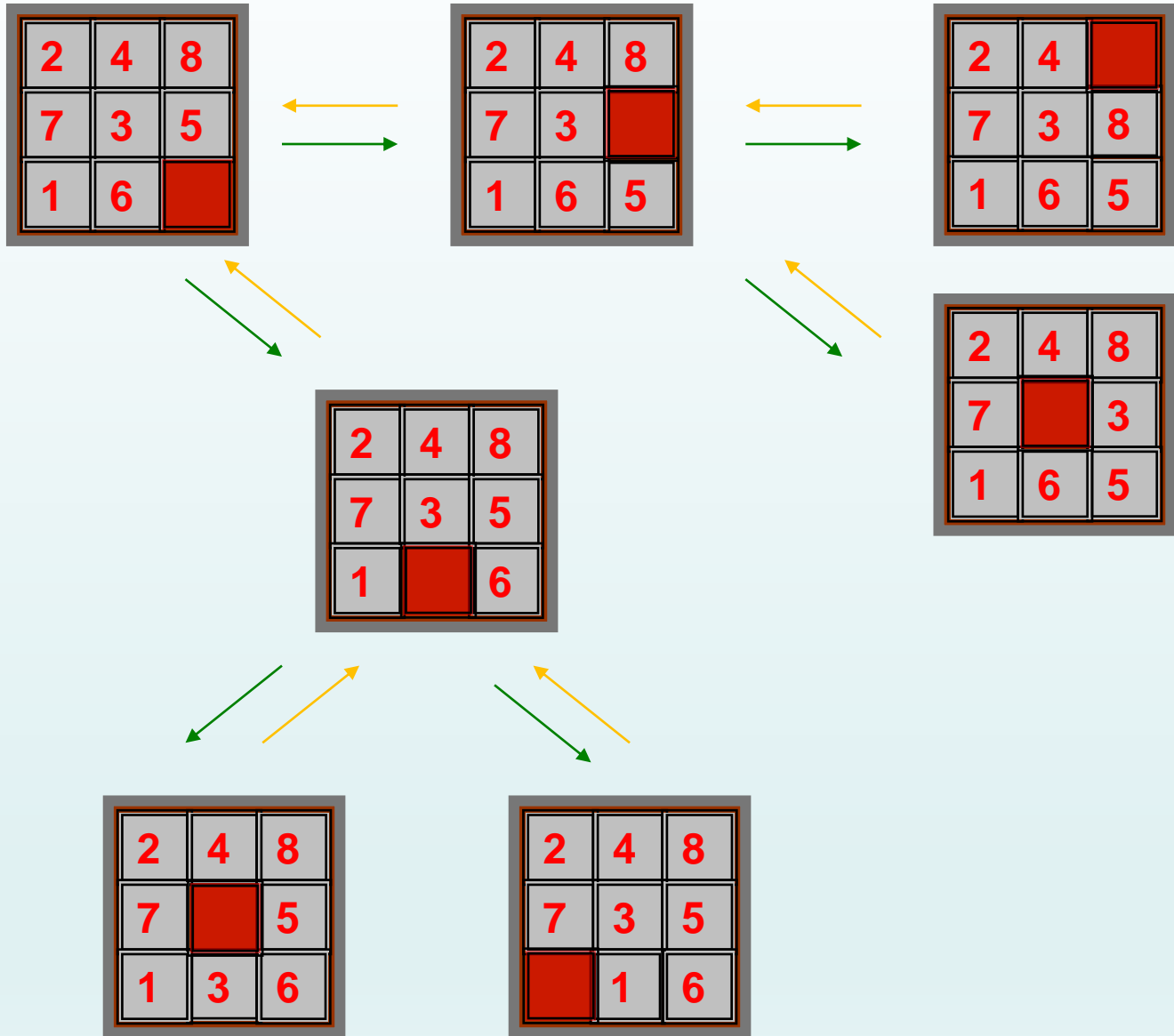
2	4	8
7	3	5
1	6	

*Estado Inicial*

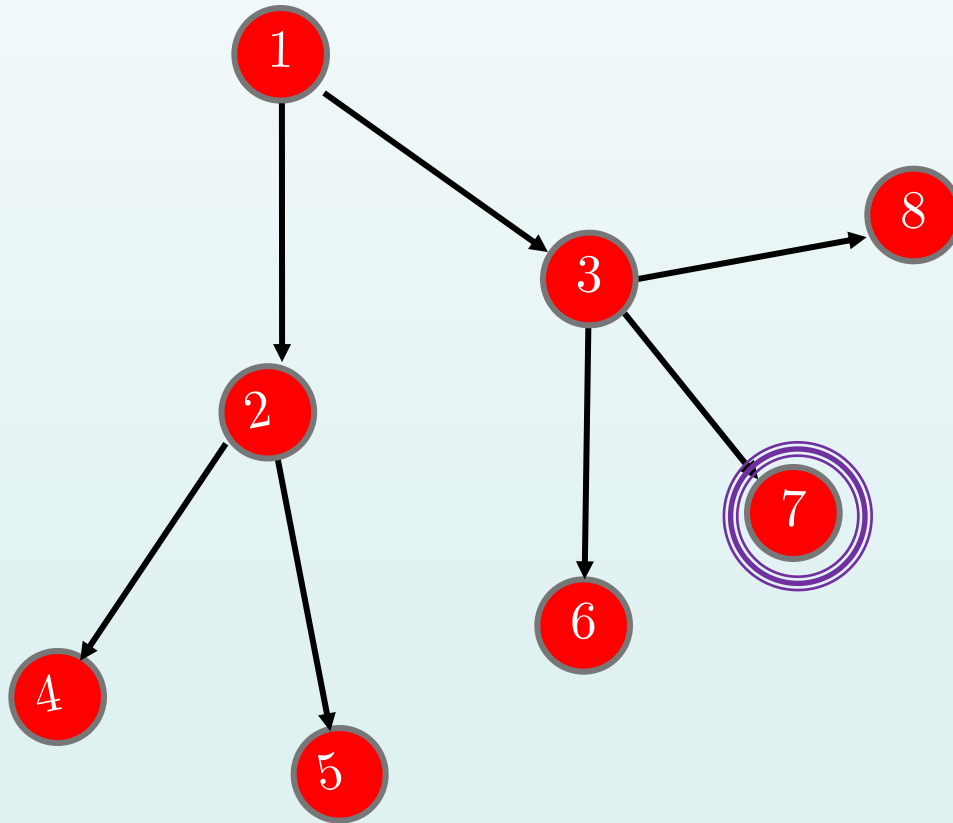


2	3	4
1		5
8	7	6

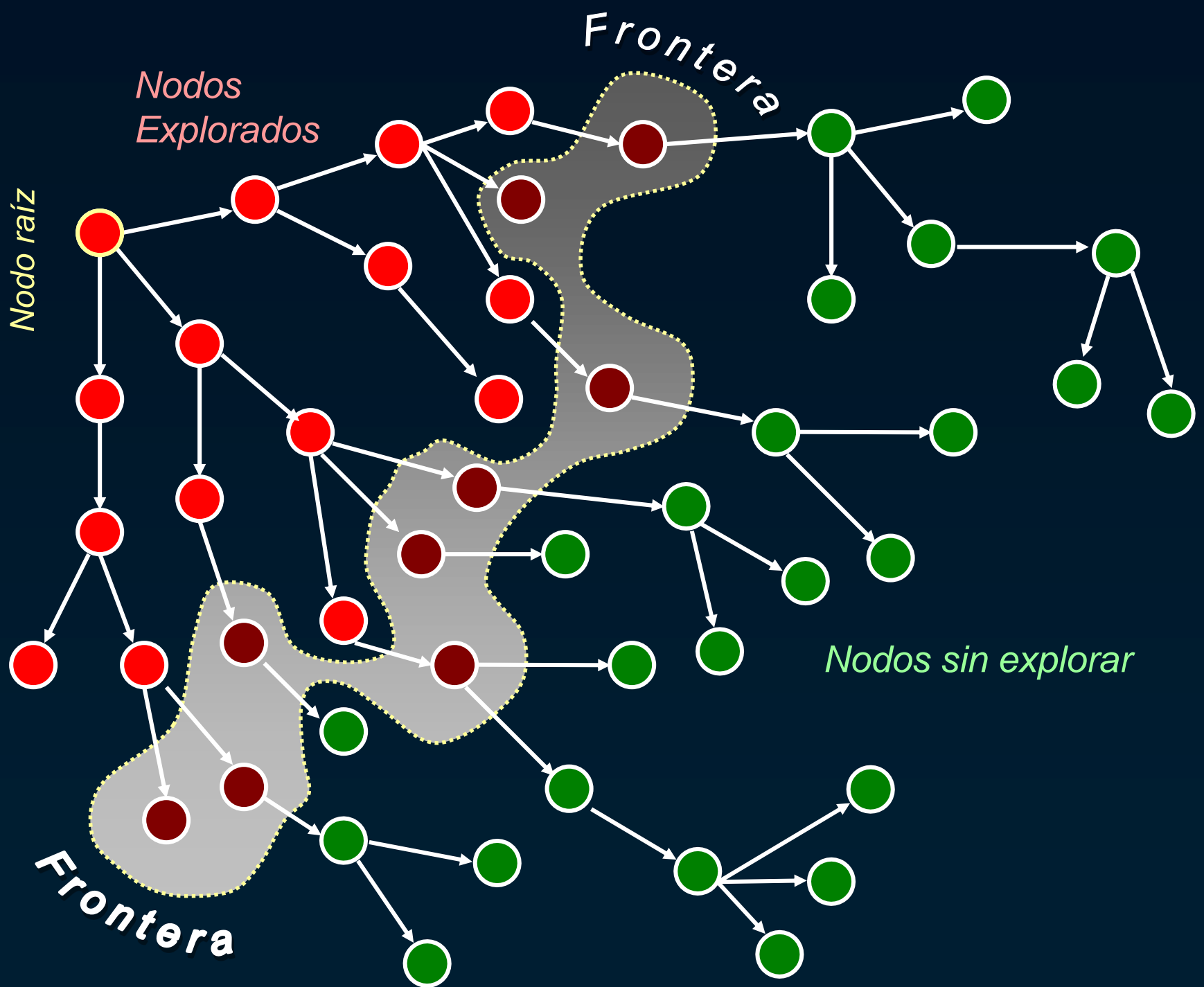
*Estado Final*



# Buscando ...



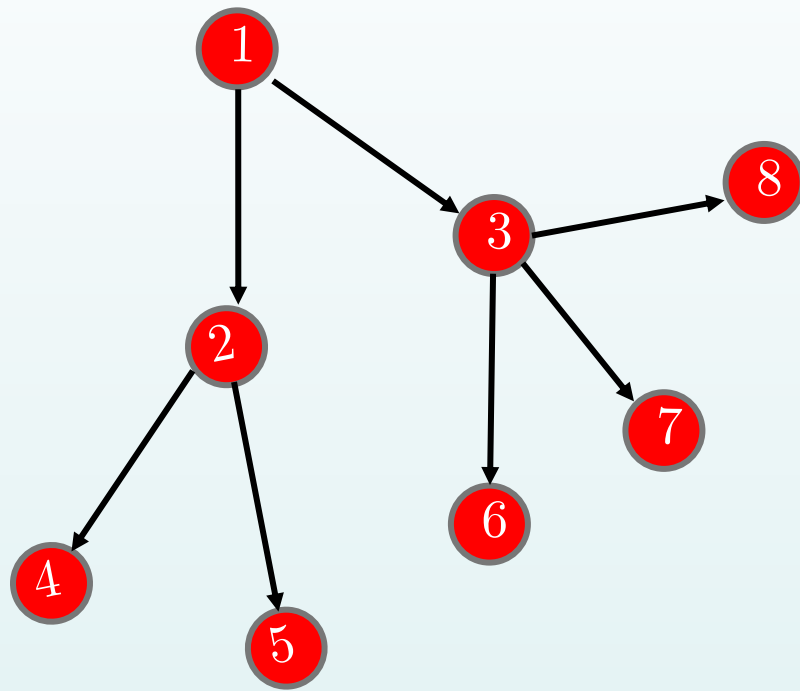
```
vecinos(1, [2, 3]).  
vecinos(2, [4, 5]).  
vecinos(3, [6, 7, 8]).  
vecinos(4, [ ]).  
vecinos(5, [ ]).  
vecinos(6, [ ]).  
vecinos(7, [ ]).  
vecinos(8, [ ]).  
es_meta( 7 ).
```



# Estrategias de Búsqueda

- A partir del problema se define el *grafo* y la *meta* que se desea alcanzar.
- Al definir el grafo queda definido cuales son los *vecinos* de un nodo dado.
- Una *estrategia de búsqueda* define la forma en como se seleccionan elementos de la frontera.
- La forma de seleccionar el nodo a *expandir* y la forma como agregar un nodo a la frontera están indefinidas y al hacerlo aparecen diferentes estrategias de búsqueda.

# Ejemplo



Definición de *vecinos* y *es\_meta*

*vecinos*(1, [2, 3]).

*vecinos*(2, [4, 5]).

*vecinos*(3, [6, 7, 8]).

*vecinos*(4, [ ]).

*vecinos*(5, [ ]).

*vecinos*(6, [ ]).

*vecinos*(7, [ ]).

*vecinos*(8, [ ]).

*es\_meta*( 7 ).

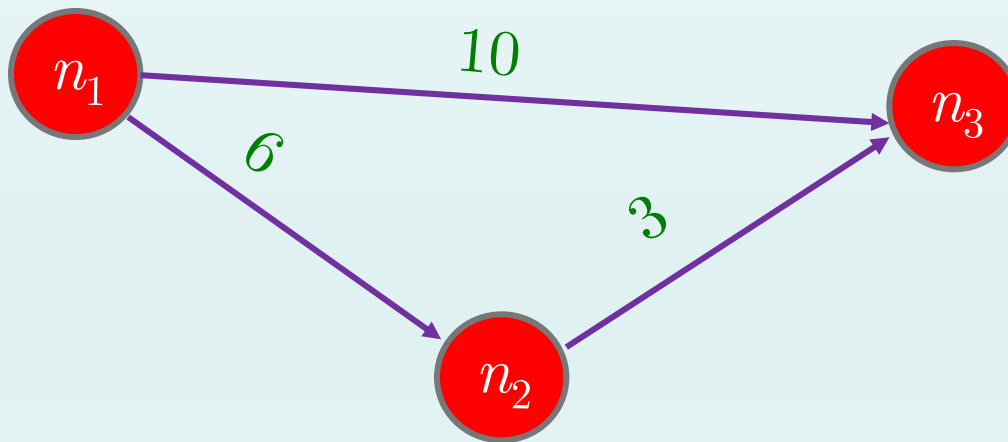
Ejercicio 1: escriba un algoritmo en pseudocódigo de *busqueda genérico* que utilice estas estructuras, simulando la exploración del espacio de búsqueda mediante la generación de sucesores de los nodos/estados ya explorados.

Asuma esta signatura: *funcion* *TreeSearch*(*Raiz*, *ArbolBusqueda*, *Estrategia*), la función devuelve, la *solución* (camino a la meta) o *failure*.

La estrategia define como se hace la expansión (queremos que sea genérica)

# Costos

- En algunos casos se asigna a los arcos en los grafos una cierta “*longitud*” o “*peso*”.
- Esto permite calcular el “*costo*” de un paso desde la raíz a un nodo meta.
- Existen diferentes maneras de representar este costo.
- Por ejemplo,  $\text{costo}(n_1, n_2)$  representa el costo de recorrer el arco que va de  $n_1$  a  $n_2$  es 6.



$$\text{costo}(n_1, n_2) = 6$$

$$\text{costo}(n_2, n_3) = 4$$

$$\text{costo}(n_1, n_3) = 9$$



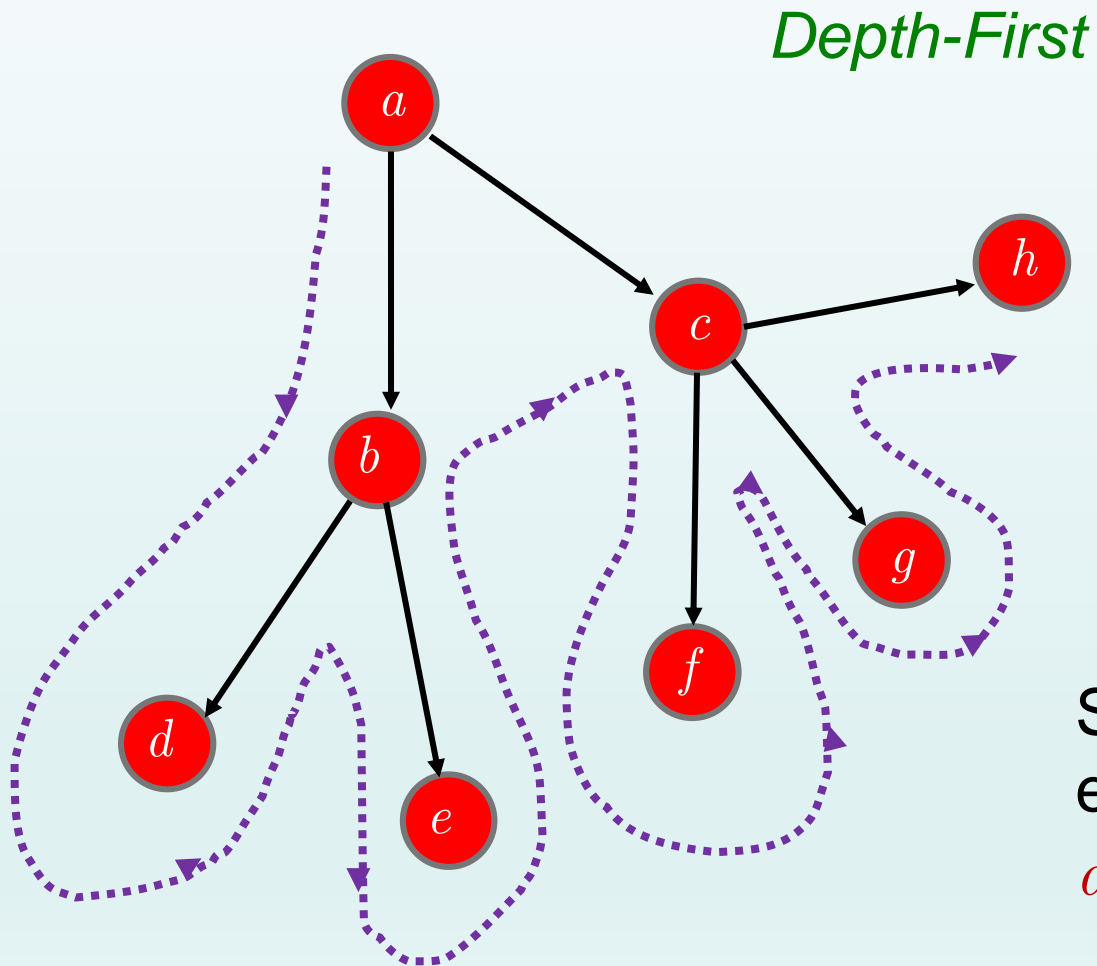
# Estrategias de Búsqueda: Propiedades

- *Compleitud*: garantiza encontrar la meta si esta existe o retorna falla si no existiera.
- *Optimalidad*: garantiza encontrar siempre la “*mejor*” meta.
- *Complejidad Temporal*: número de operaciones aplicadas en la búsqueda.
- *Complejidad Espacial*: número de nodos almacenados durante la búsqueda.

# Búsqueda: Definiciones

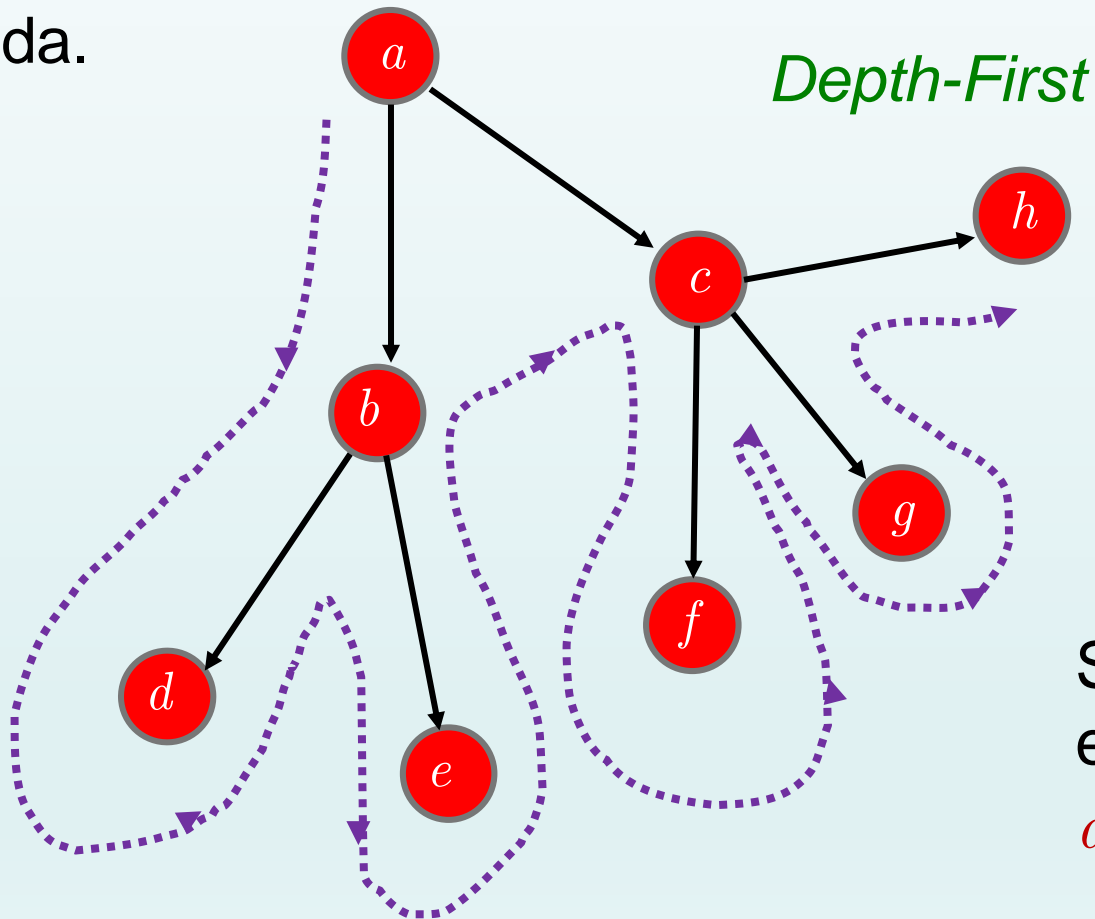
- Métodos de **búsqueda a ciegas**:
  - Primero en profundidad (*Depth First*)
  - Primero a lo ancho (*Breadth First*)
  - Profundidad incrementada gradualmente (*Iterative Deepening*)
  - Ramificación incrementada gradualmente (*Iterative Broadening*)
  - Bi-direccional

# Búsqueda Primero en Profundidad



# Búsqueda Primero en Profundidad

Ejercicio 2: Revise el algoritmo de búsqueda genérico que definió previamente para implementar esta técnica de búsqueda.

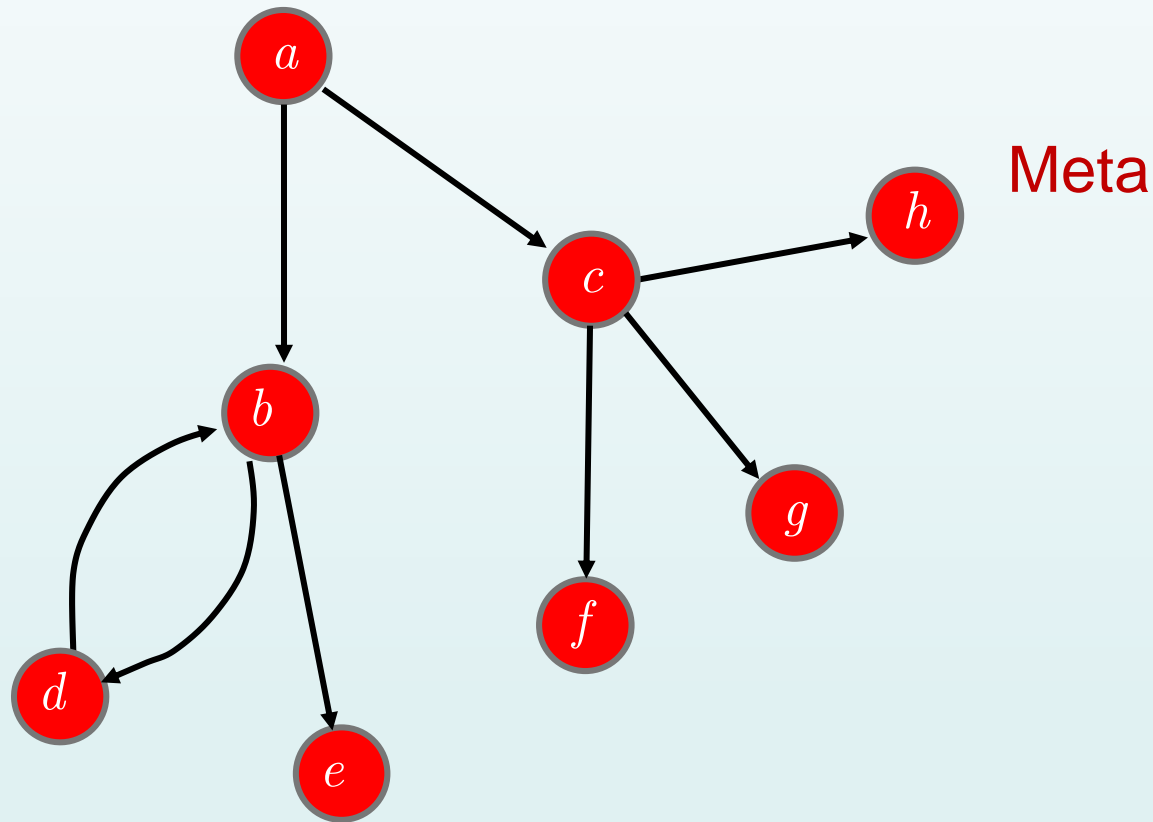


# Análisis de Primero en Profundidad

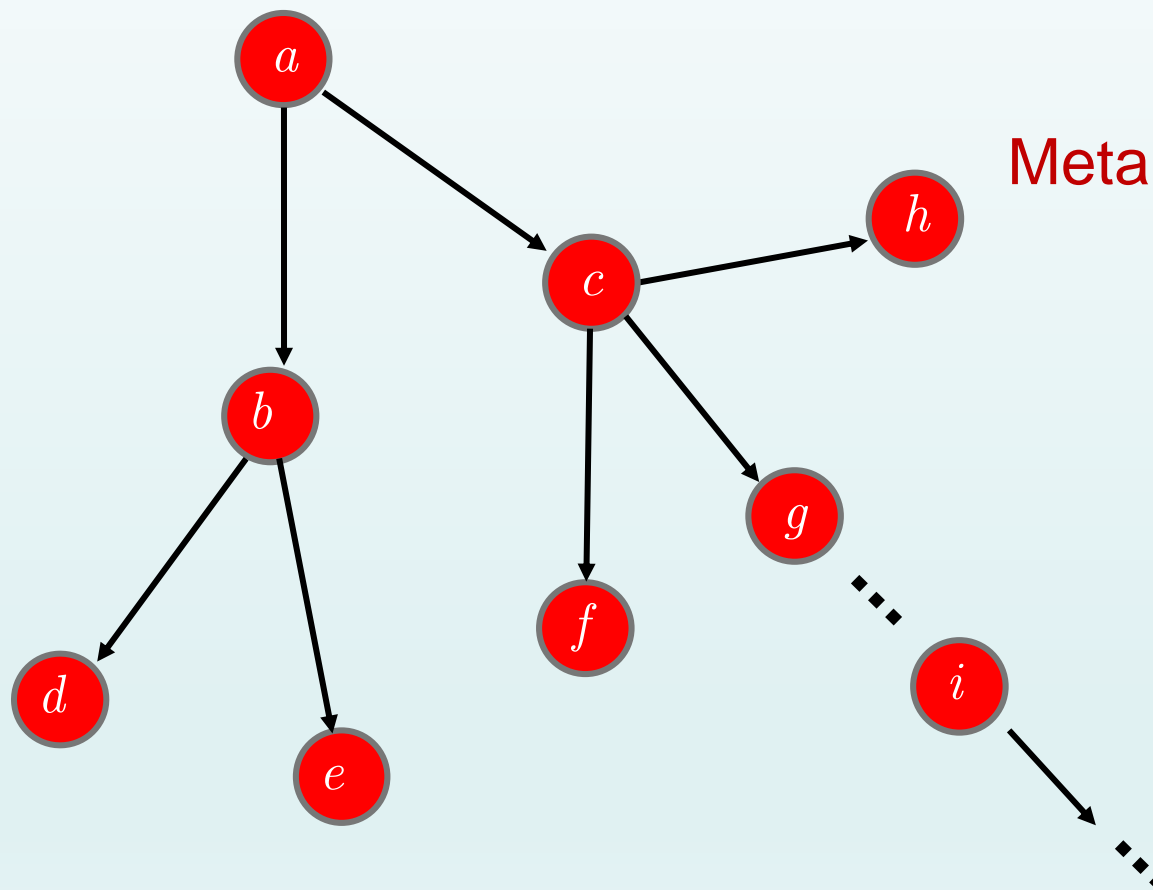
- *Complejidad*: no
- *Optimalidad*: no
- *Complejidad Temporal*:  $b^m$
- *Complejidad Espacial*:  $b \cdot m$

$b$  es el factor de ramificación  
 $m$  es la máxima profundidad

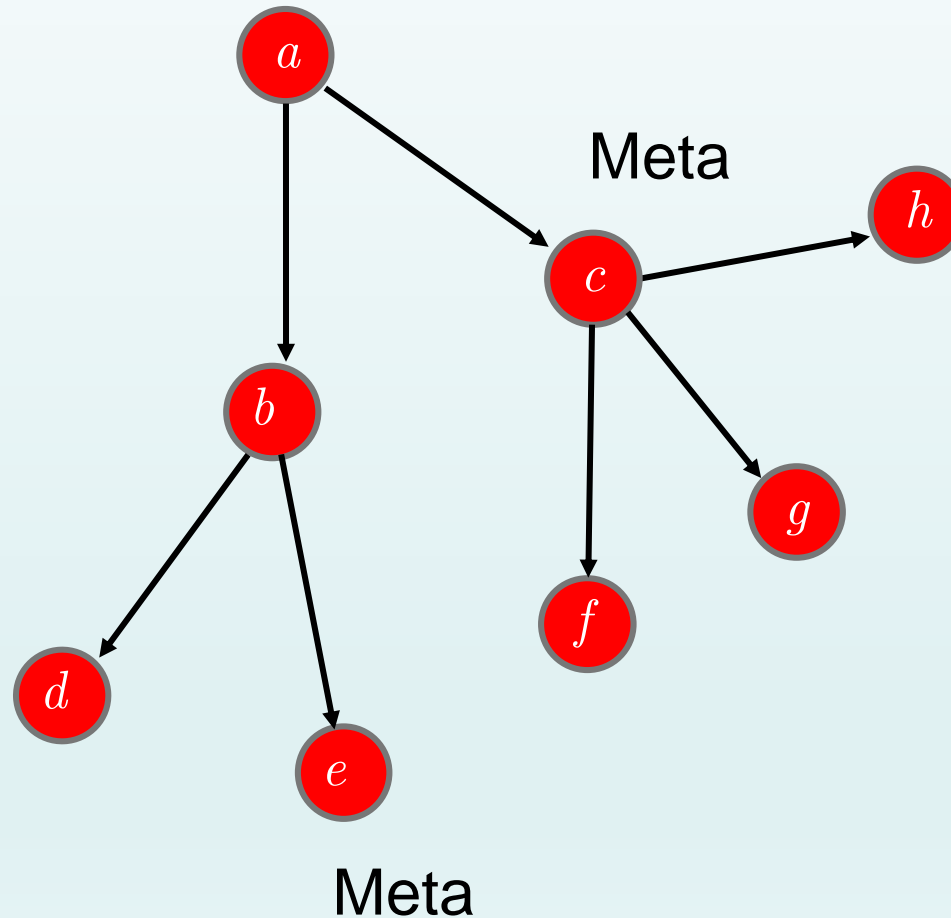
# Completitud



# Completitud



# Optimalidad

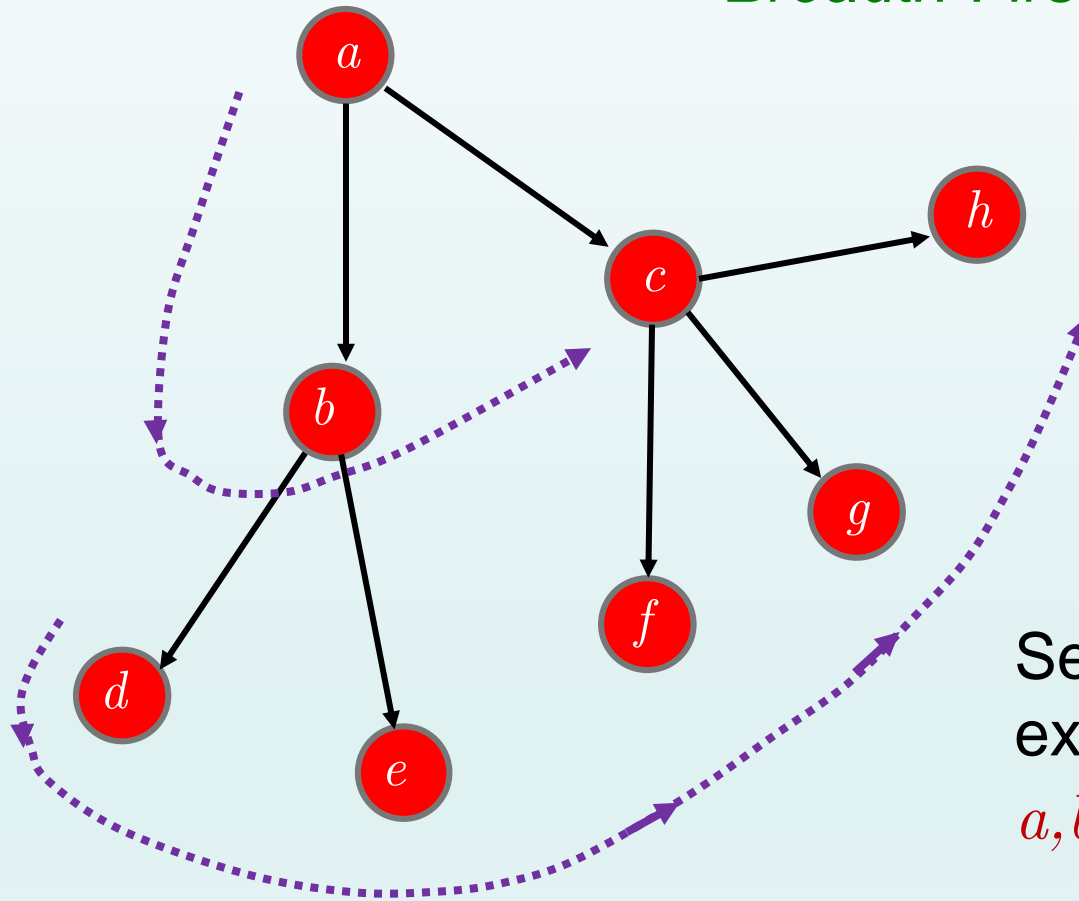


Supongamos  
que el costo de  
los arcos es  
uniforme



# Búsqueda Primero a lo Ancho

*Breadth-First*



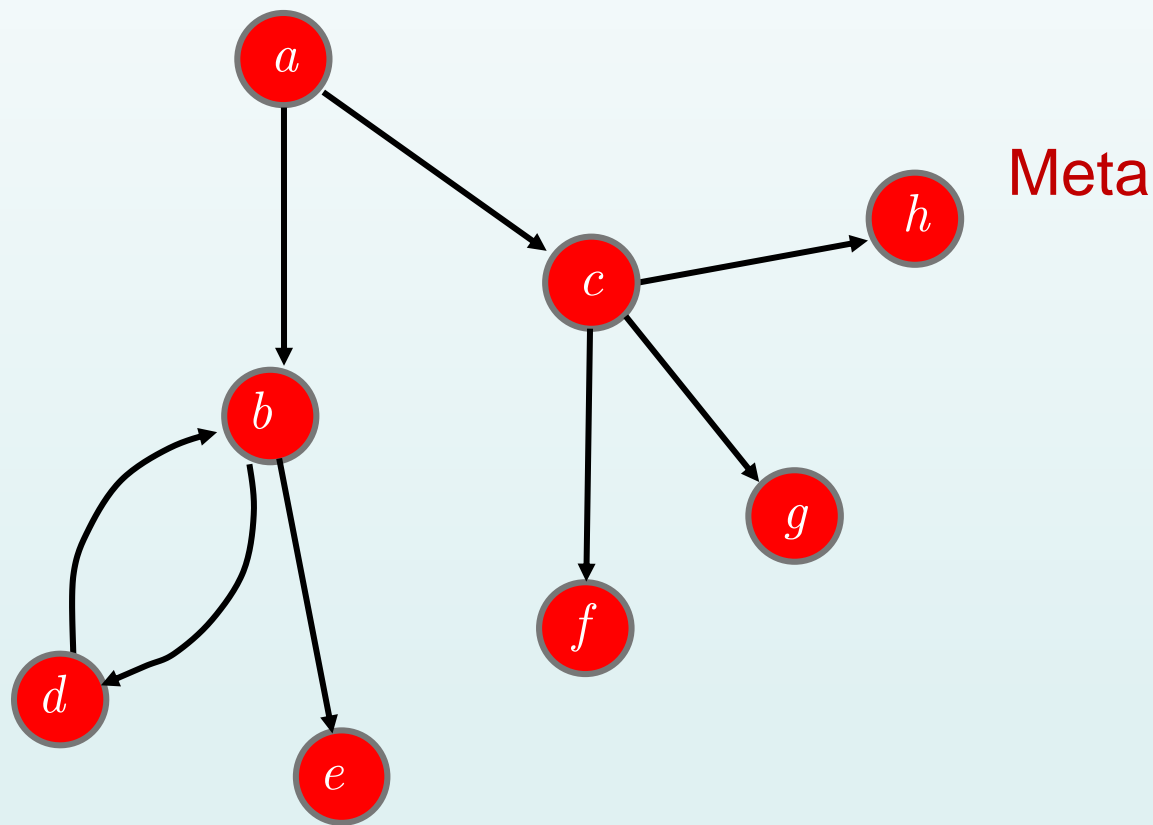
Secuencia de  
exploración  
*a, b, c, d, e, f, g, h*

# Análisis de Primero a lo Ancho

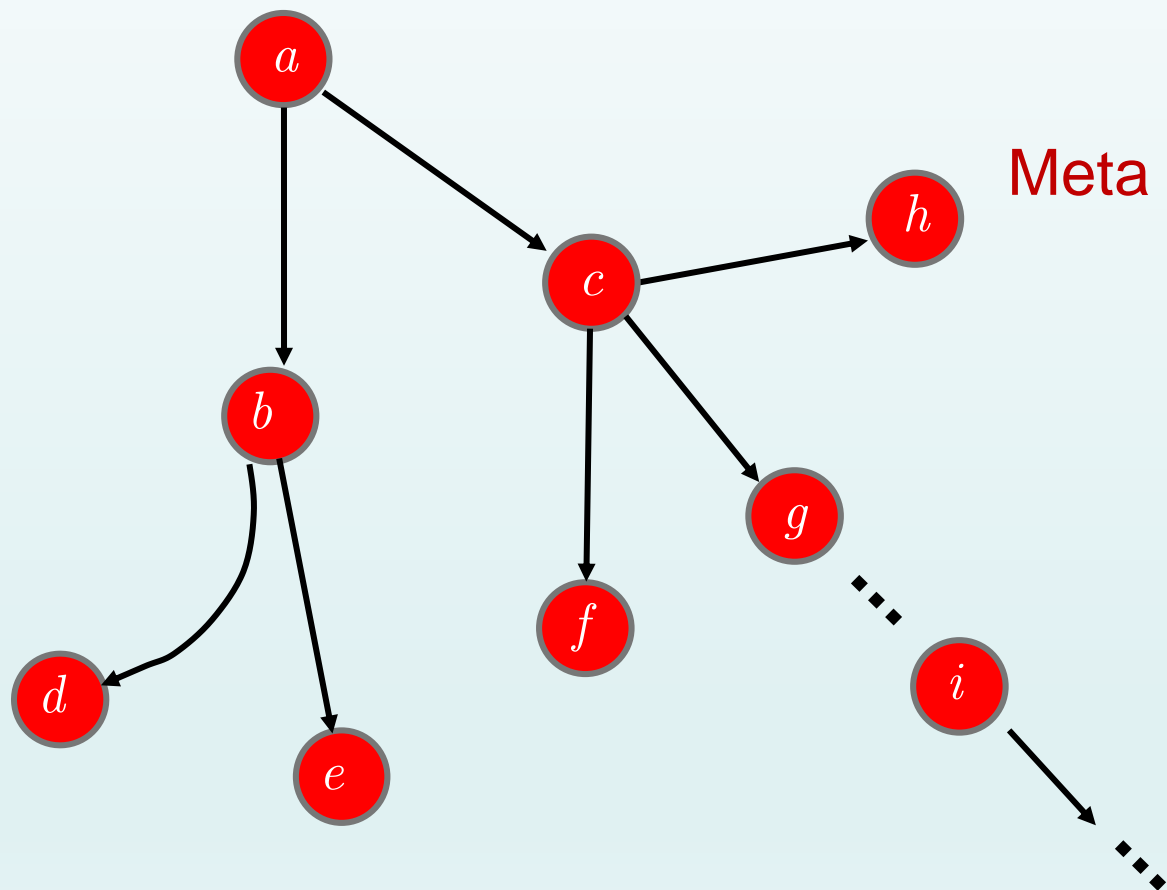
- *Compleitud:* si
- *Optimalidad:* si
- *Complejidad Temporal:*  $b^d$
- *Complejidad Espacial:*  $b^d$

$b$  es el factor de ramificación  
 $d$  es la profundidad de la meta

# Completitud



# Compleitud

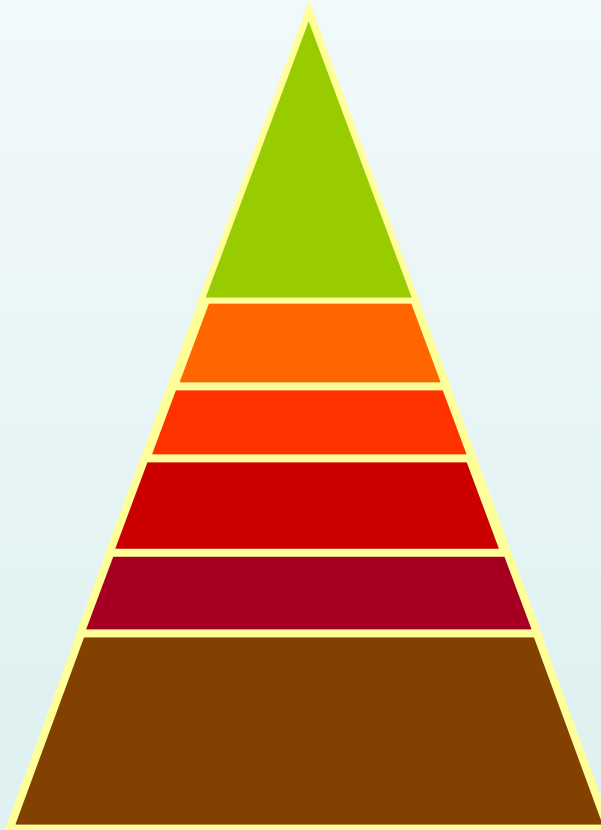


# Otros Métodos de Búsqueda

---

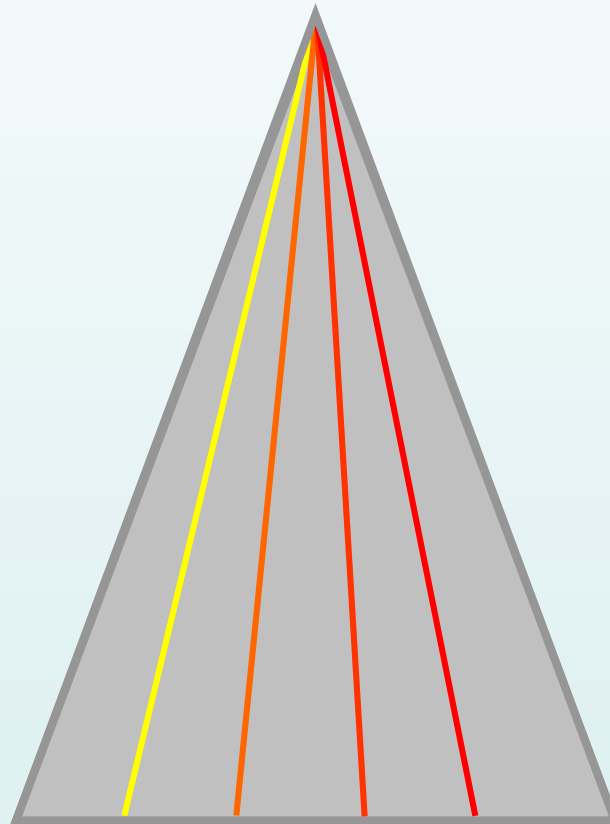
- Profundización Iterativa.
- Expansión Iterativa (iterative deepening)
- Bidireccional

# Profundización Iterativa

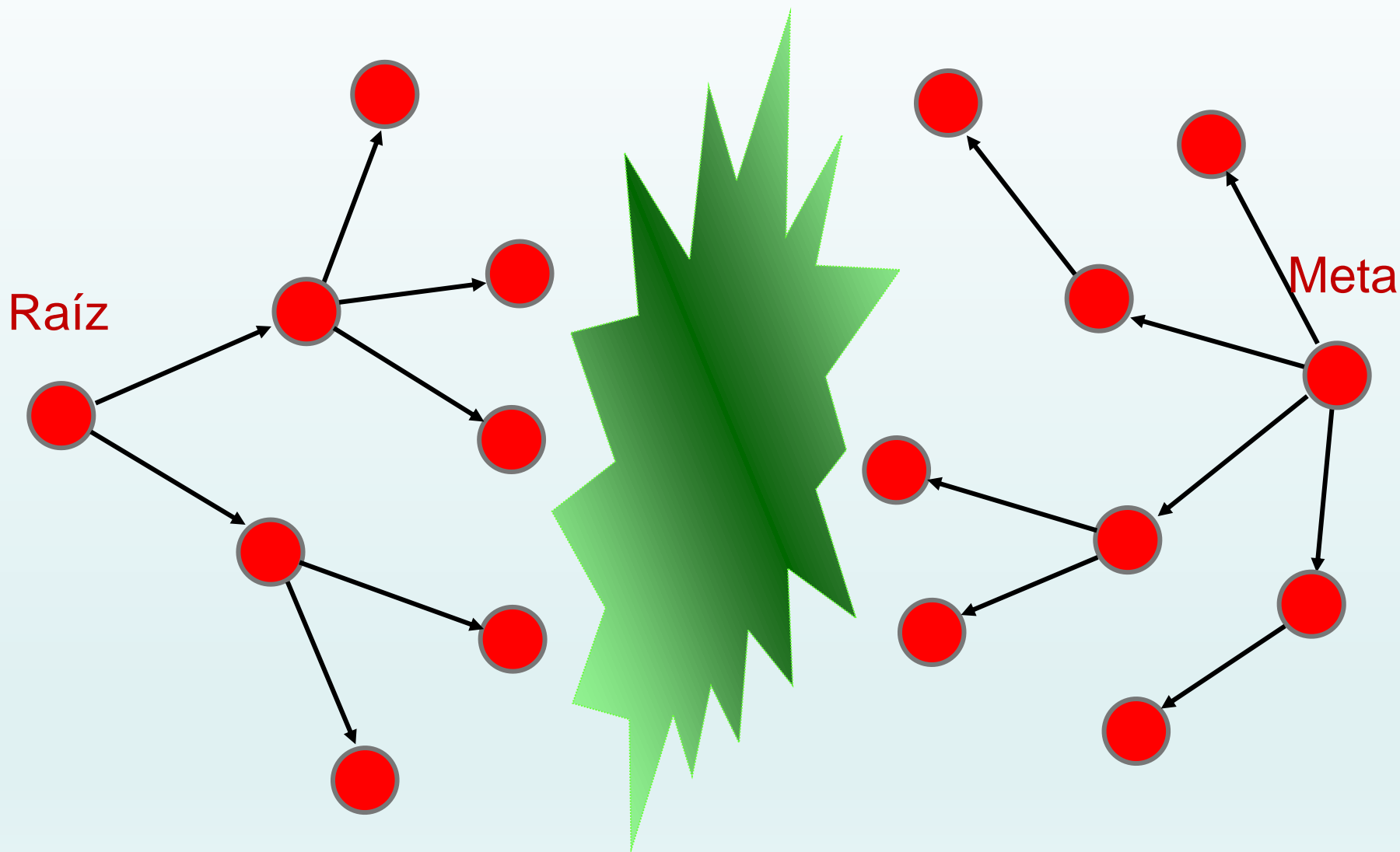


# Expansión Iterativa

Paper original de Iterative Broadening



# Bidireccional





# Analisis de los algoritmos

Criterion	Breadth-First	Depth-First	Iterative Deepening
Complete?	Yes	No	Yes
Time	$O(b^{d+1})$	$O(b^m)$	$O(b^d)$
Space	$O(b^{d+1})$	$O(bm)$	$O(bd)$
Optimal?	Yes	No	Yes



# Búsqueda Informada

- Los métodos de búsqueda que hemos presentado hasta aquí realizan un *recorrido exhaustivo del espacio de búsqueda total*.
- De esta manera, *si el espacio es finito*, la meta se encuentra o el proceso se completa al fallar en encontrarla.
- Pero aún en el caso finito, esto resulta imposible para problemas de tamaño suficiente para ser de *interés práctico* dado que sus espacios de búsqueda son enormes.
- Veremos métodos de búsqueda orientados a considerar este tipo de problemas poco tratables.
- El compromiso que se acepta es tal que:
  - *No es posible encontrar soluciones aún cuando existan.*
  - *Las soluciones que se encuentran pueden no ser optimales.*
- Sin embargo, estos métodos tienen gran utilidad práctica.



# Búsqueda Informada

- Los métodos de búsqueda vistos son *exhaustivos* porque son *métodos generales*, *independientes del problema*: *Métodos de Búsqueda a Ciegas* o *No Informados*.
- Para lograr resultados prácticos: considerar la *estructura particular* del espacio búsqueda para una *clase particular de problemas*.
- Se introduce una *noción de medida, o métrica* que permite al agente de búsqueda estimar la distancia desde el estado en que se encuentra hasta la meta.
- El uso de esta métrica permitirá elegir los nodos que se explorarán en el paso siguiente.
- La función que calcula esas métricas se denomina *Heurística*.

# Heurística

- Una *heurística* (del griego *Heuriskein*, “descubrir”) es una técnica que permite mejorar la eficiencia de la búsqueda.
- Pero para lograr la eficiencia a veces se sacrifica la *completitud* (*siempre encontrar la meta*) y/o la *optimalidad* (*encontrar la mejor meta*).
- Una heurística es una estimación *adecuada* del costo o longitud del paso desde un estado a una meta.
- Diremos que *subestima* la distancia si su estimación a la meta es menor o igual que la distancia real.
- La heurística debe calcularse siempre (para cada nodo expandido), es necesario *balancear* la ganancia en eficiencia de búsqueda con el gasto al calcular la heurística.

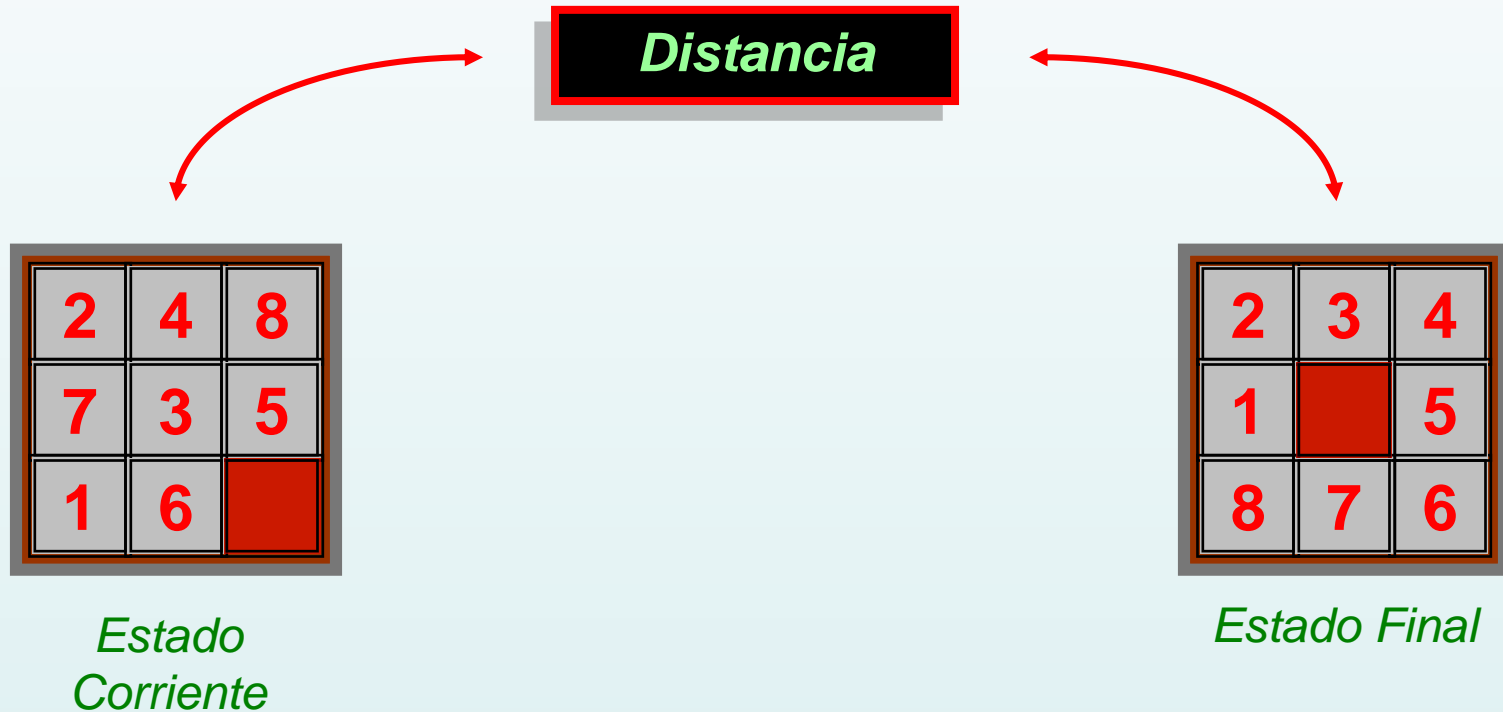
# Ejemplo



$$h_1 \left( \begin{array}{|c|c|c|} \hline 2 & 4 & 8 \\ \hline 7 & 3 & 5 \\ \hline 1 & 6 & \text{red square} \\ \hline \end{array} \right) = ?$$

Estimando la distancia

# Ejemplo



$$h_1 \left( \begin{array}{|c|c|c|} \hline 2 & 4 & 8 \\ \hline 7 & 3 & 5 \\ \hline 1 & 6 & \\ \hline \end{array} \right) = \text{la cantidad de piezas fuera de lugar} \\ = 7$$

# Ejemplo

2	4	8
7	3	5
1	6	

*Estado 1*

$$h_1 \left( \begin{array}{|c|c|c|} \hline 2 & 4 & 8 \\ \hline 7 & 3 & 5 \\ \hline 1 & 6 & \\ \hline \end{array} \right) = 7$$

2	4	8
7	3	5
1		6

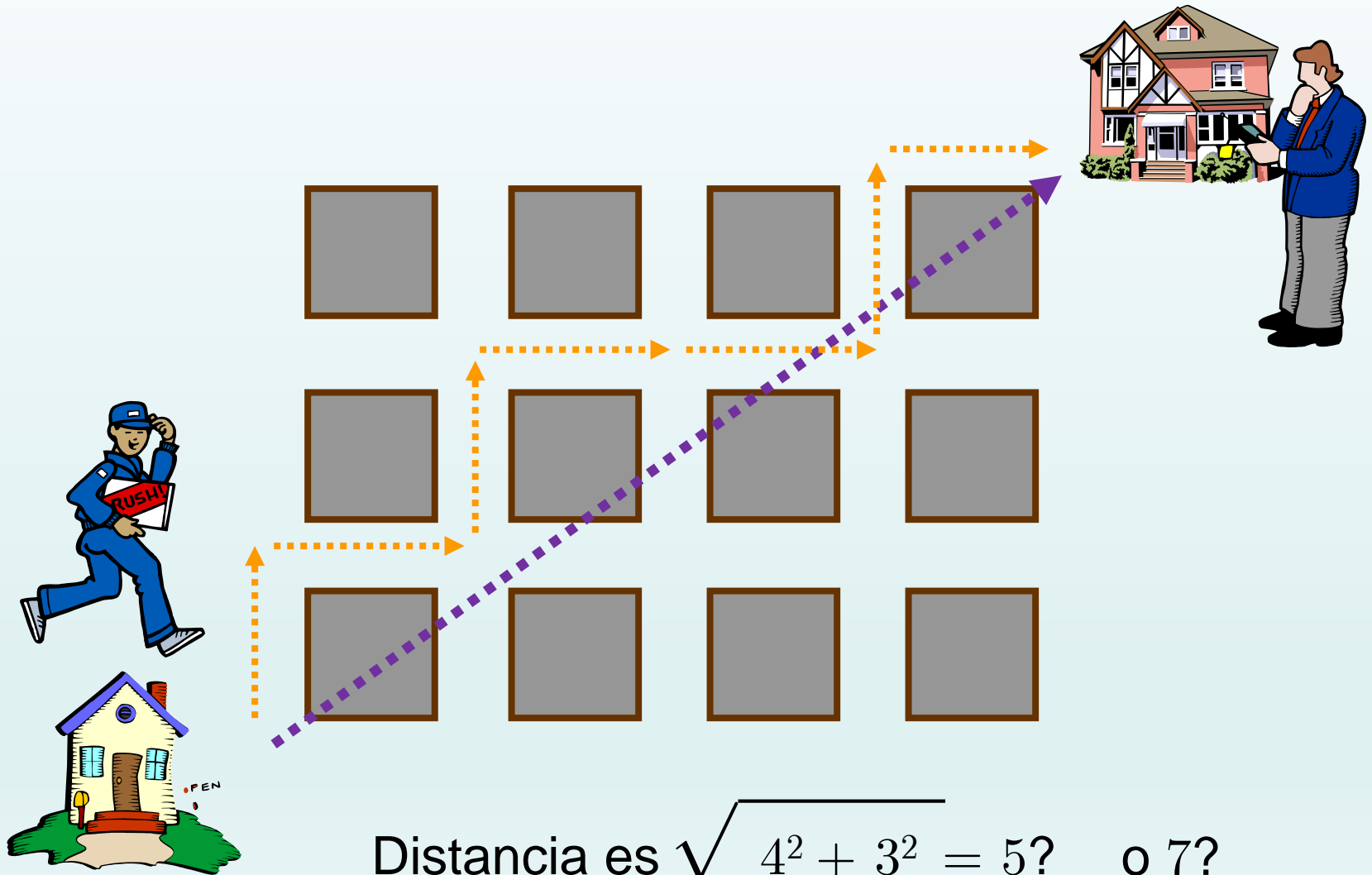
*Estado 2*

$$h_1 \left( \begin{array}{|c|c|c|} \hline 2 & 4 & 8 \\ \hline 7 & 3 & 5 \\ \hline 1 & & 6 \\ \hline \end{array} \right) = 6$$

2	3	4
1		5
8	7	6

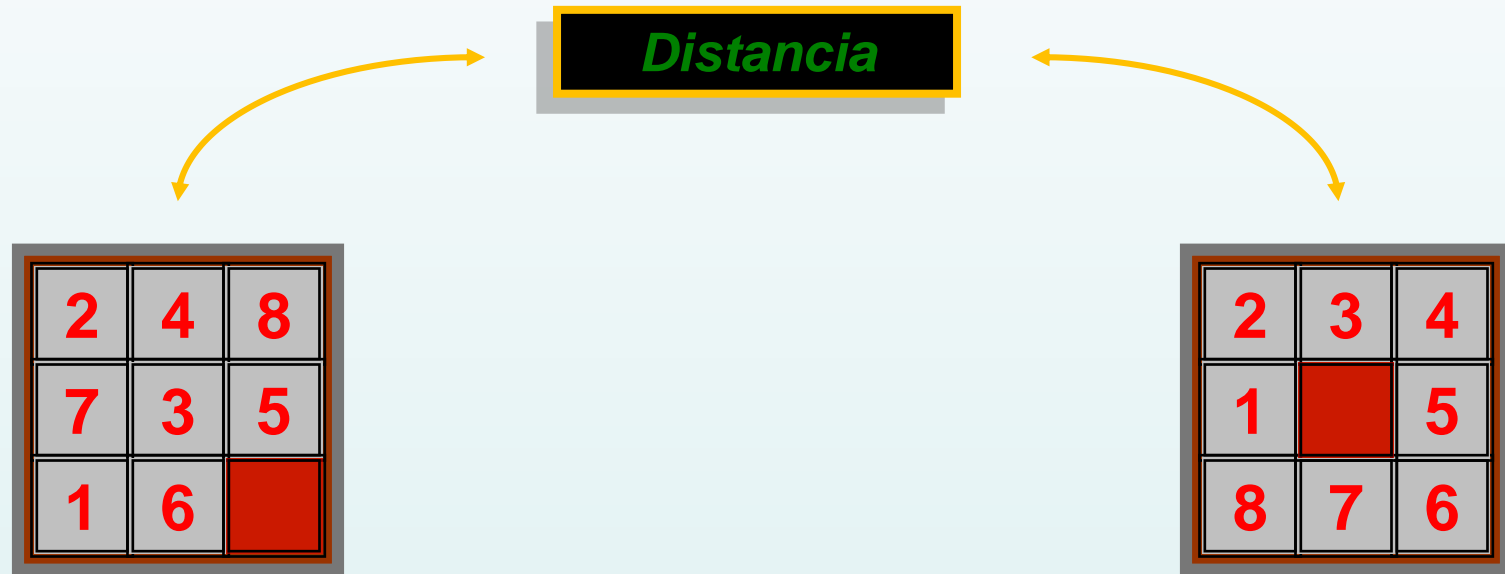
*Estado Final*

# Distancia de Manhattan





# Ejemplo



*Estado  
Corriente*

*Estado Final*

$$h_2 \left( \begin{array}{|c|c|c|} \hline 2 & 4 & 8 \\ \hline 7 & 3 & 5 \\ \hline 1 & 6 & \text{red square} \\ \hline \end{array} \right) = ?$$

¿Otra heurística?

# Ejemplo



$$h_2 \left( \begin{array}{|c|c|c|} \hline 2 & 4 & 8 \\ \hline 7 & 3 & 5 \\ \hline 1 & 6 & \\ \hline \end{array} \right) = \text{cantidad de movimientos necesarios para poner cada pieza en su lugar}$$
$$= 10$$

# Ejemplo

2	4	8
7	3	5
1	6	

*Estado 1*

$$h_2 \left( \begin{array}{|c|c|c|} \hline 2 & 4 & 8 \\ \hline 7 & 3 & 5 \\ \hline 1 & 6 & \\ \hline \end{array} \right) = 10$$

2	3	4
1		5
8	7	6

*Estado Final*

2	4	8
7	3	5
1		6

*Estado 2*

$$h_2 \left( \begin{array}{|c|c|c|} \hline 2 & 4 & 8 \\ \hline 7 & 3 & 5 \\ \hline 1 & & 6 \\ \hline \end{array} \right) = 9$$

# Mejor heurística?

- Contar las piezas fuera de lugar en el 8 puzzle ( $h_1$ ) es una heurística más *gruesa* que la distancia de Manhattan ( $h_2$ ), es decir, la distancia de Manhattan distingue más estados.

$$h_1 \left( \begin{array}{|c|c|c|} \hline 2 & 3 & 7 \\ \hline 1 & 4 & 5 \\ \hline 8 & \text{red} & 6 \\ \hline \end{array} \right) = 2$$

$$h_1 \left( \begin{array}{|c|c|c|} \hline 2 & 3 & 4 \\ \hline 1 & 5 & 7 \\ \hline 8 & \text{red} & 6 \\ \hline \end{array} \right) = 2$$

$$h_2 \left( \begin{array}{|c|c|c|} \hline 2 & 3 & 7 \\ \hline 1 & 4 & 5 \\ \hline 8 & \text{red} & 6 \\ \hline \end{array} \right) = 5$$

$$h_2 \left( \begin{array}{|c|c|c|} \hline 2 & 3 & 4 \\ \hline 1 & 5 & 7 \\ \hline 8 & \text{red} & 6 \\ \hline \end{array} \right) = 3$$

# Búsqueda Informada (Heurística)

- Es posible aplicar estas ideas para mejorar el Proceso de Búsqueda.
- Ahora vamos a utilizar la medida de la distancia estimada a la meta para realizar la *selección del nodo a expandir*.
- Una forma de lograr esto es *mantener ordenada* la frontera de búsqueda de acuerdo a la distancia estimada a la meta.

# Búsqueda Primero el Mejor

## Frontera:

*a*

*b, i, c*

*d, i, c*

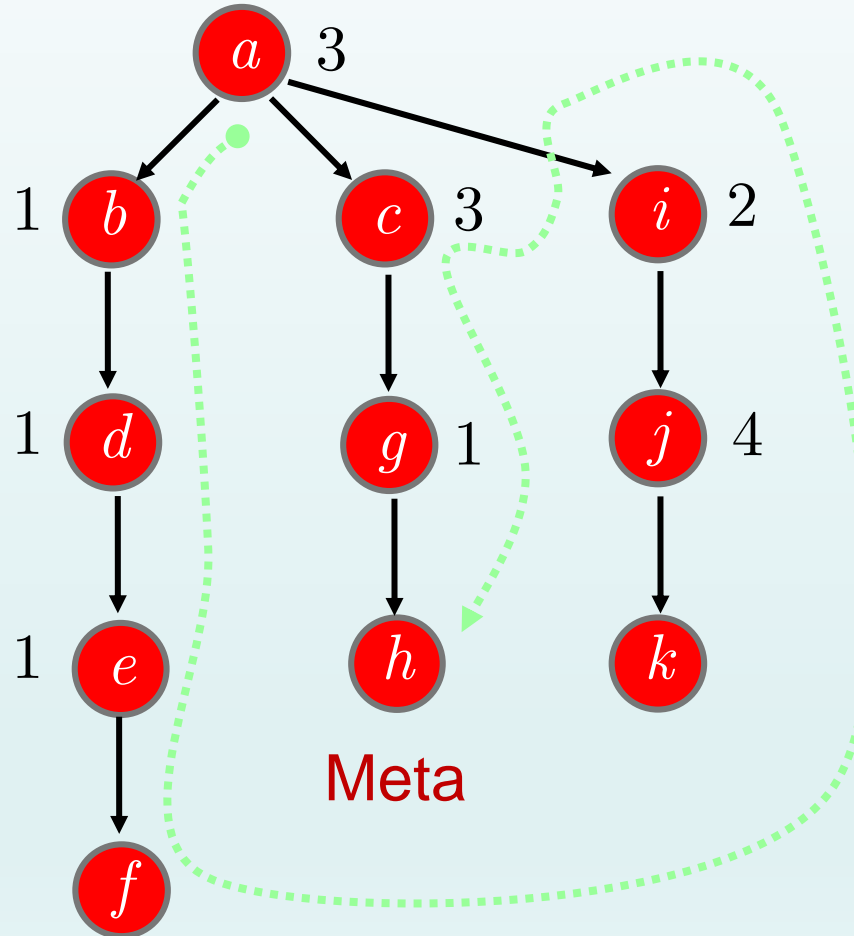
*e, i, c*

*f, i, c*

*c, j*

*g, j*

*h, j*

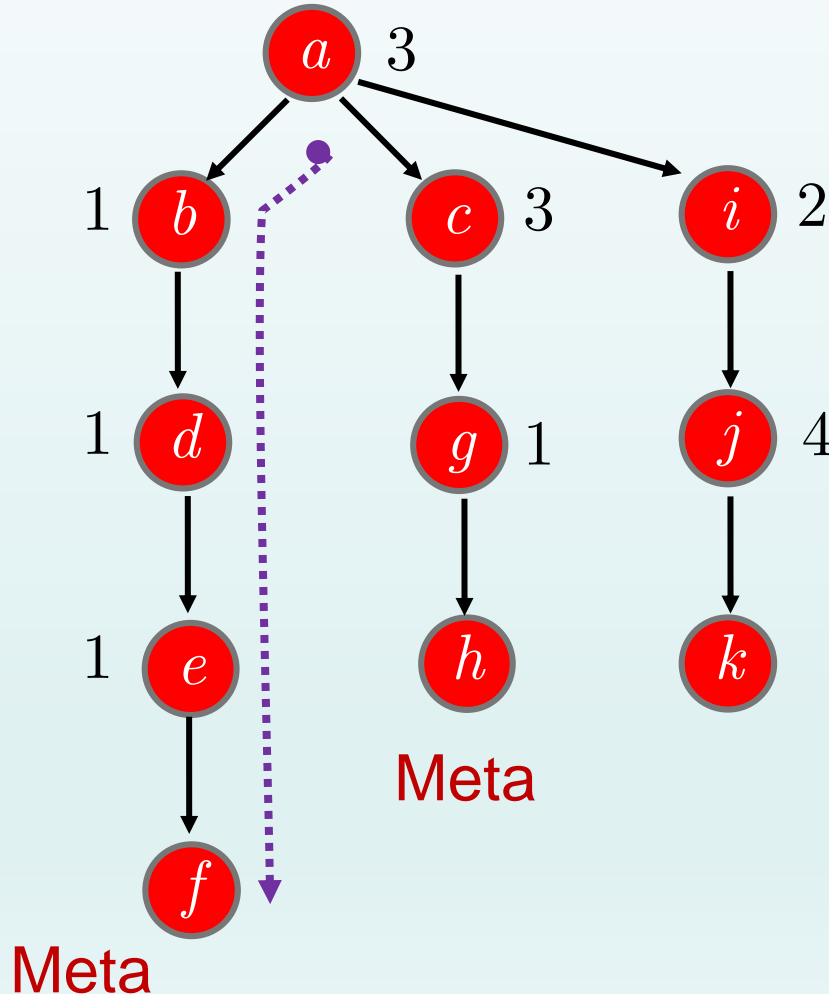


Los números representan el valor de la heurística para el nodo.

# Búsqueda Primero el Mejor

## Frontera:

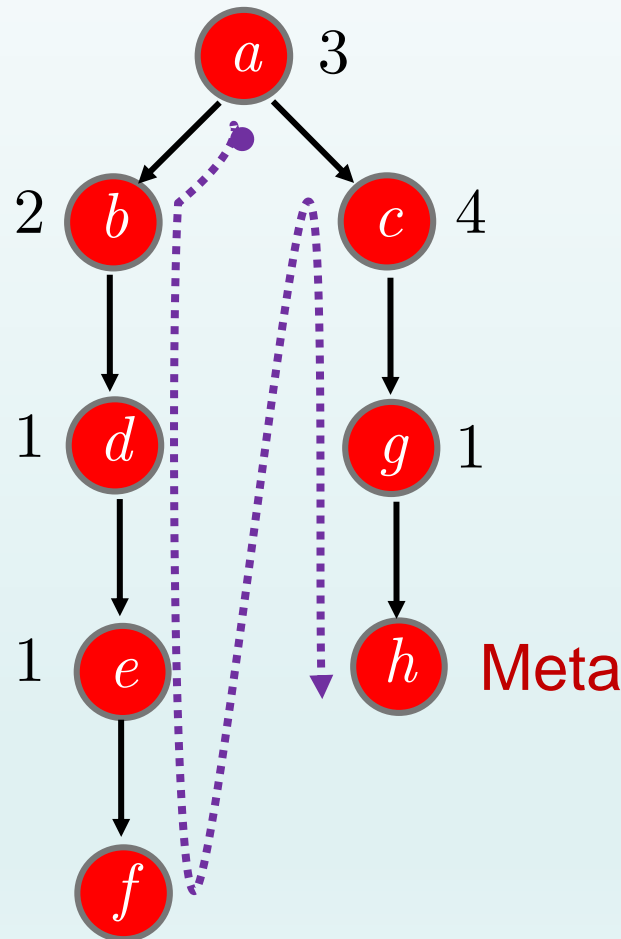
*a*  
*b, i, c*  
*d, i, c*  
*e, i, c*  
*f, i, c*



Los números representan el valor de la heurística para el nodo.

# Primero en Profundidad Heurístico

Ordenar los vecinos en la frontera por valor de la heurística, los vecinos se agregan a la frontera de manera que el mayor se selecciona **primero**. Ordena los vecinos localmente pero luego se compromete a ese camino.

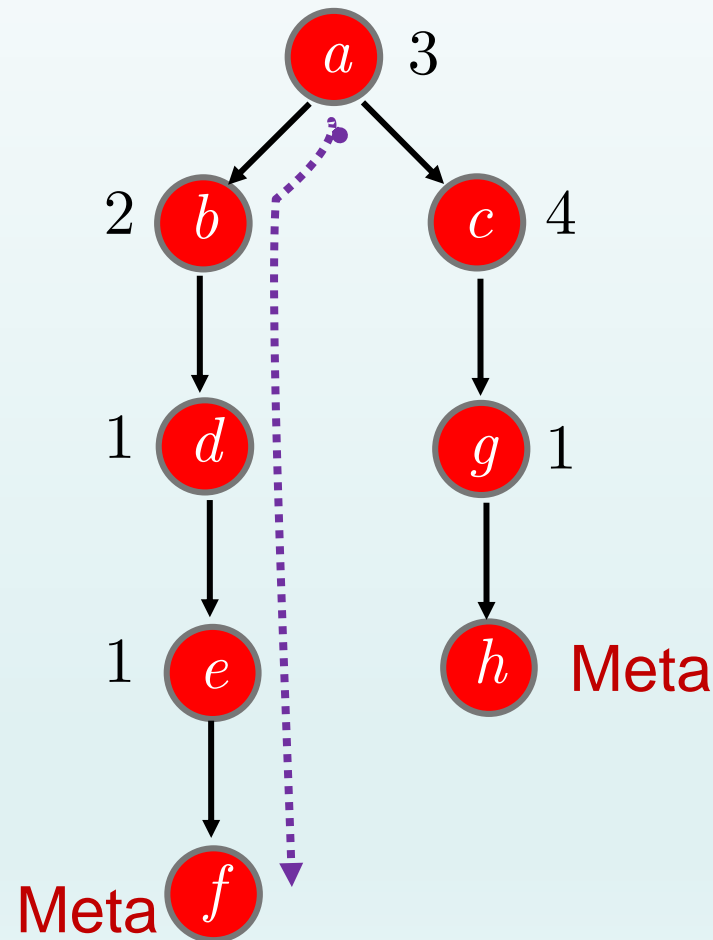


Los números representan el valor de la heurística para el nodo.



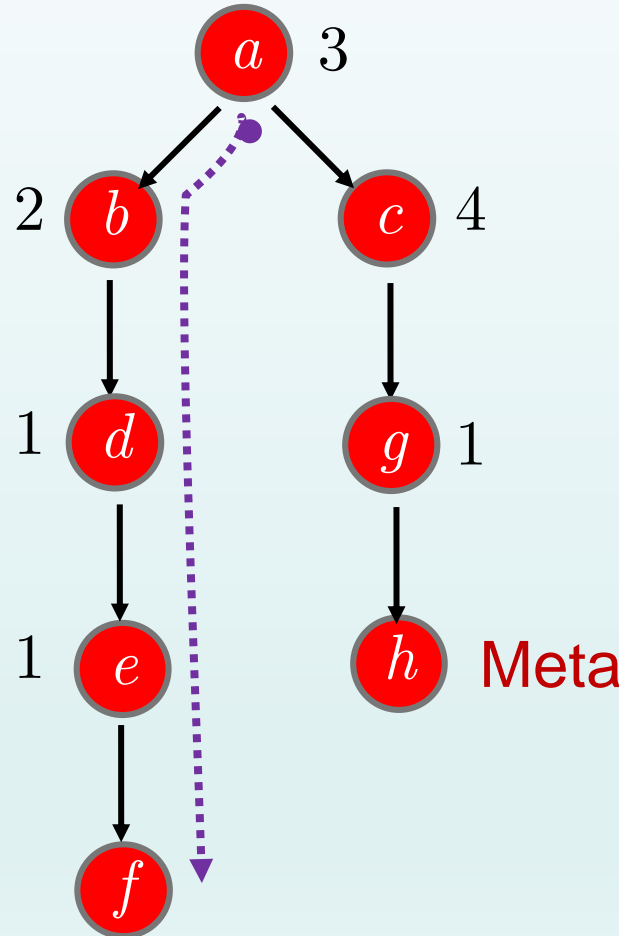
# Primero en Profundidad Heurístico

- No es optimal!



# Búsqueda en Trepada (Greedy)

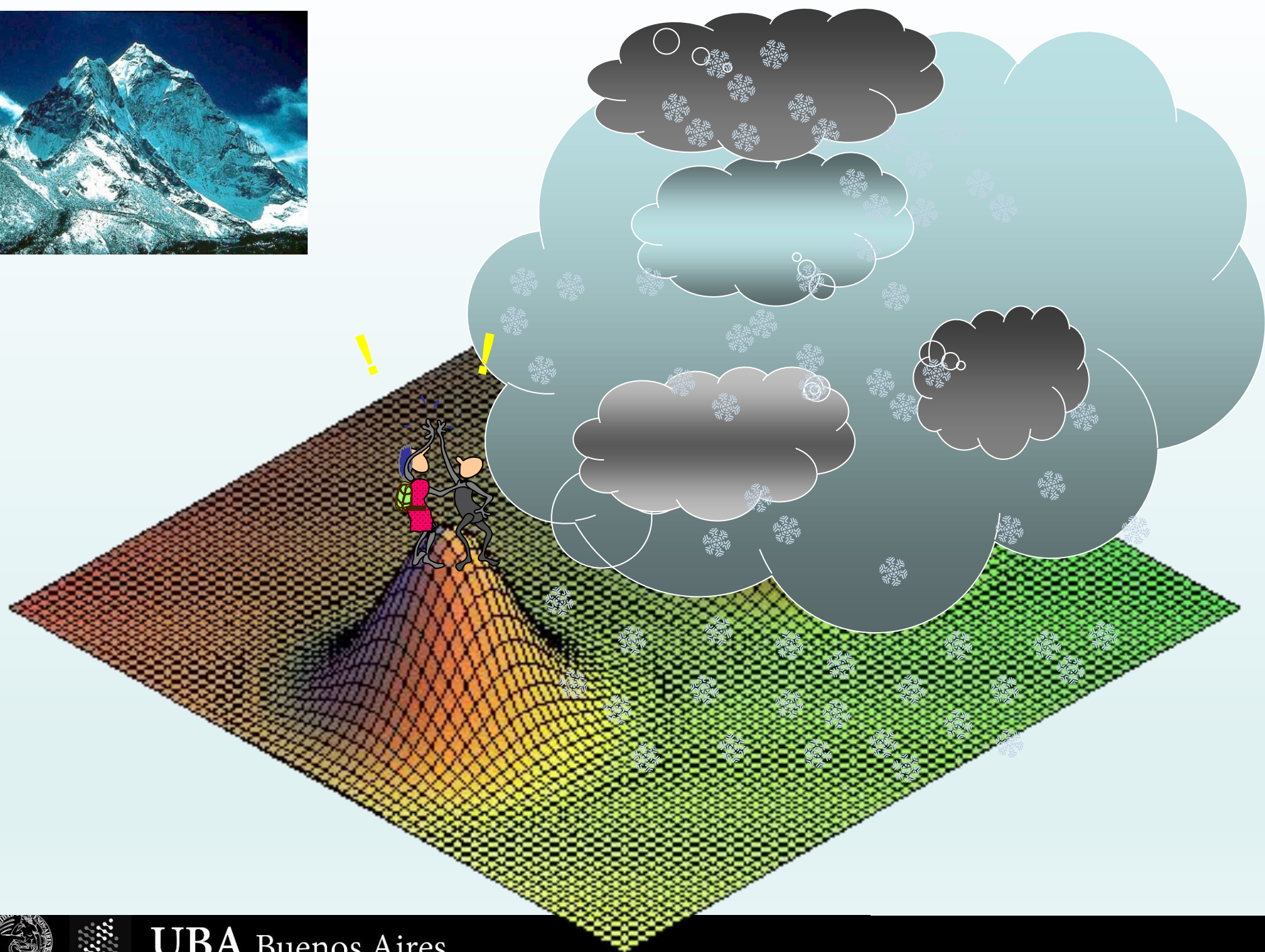
- Selecciona el mejor cada vez y se compromete a esa elección, no hay backtracking!



Los números representan el valor de la heurística para el nodo.

# Problemas en la Búsqueda Informada





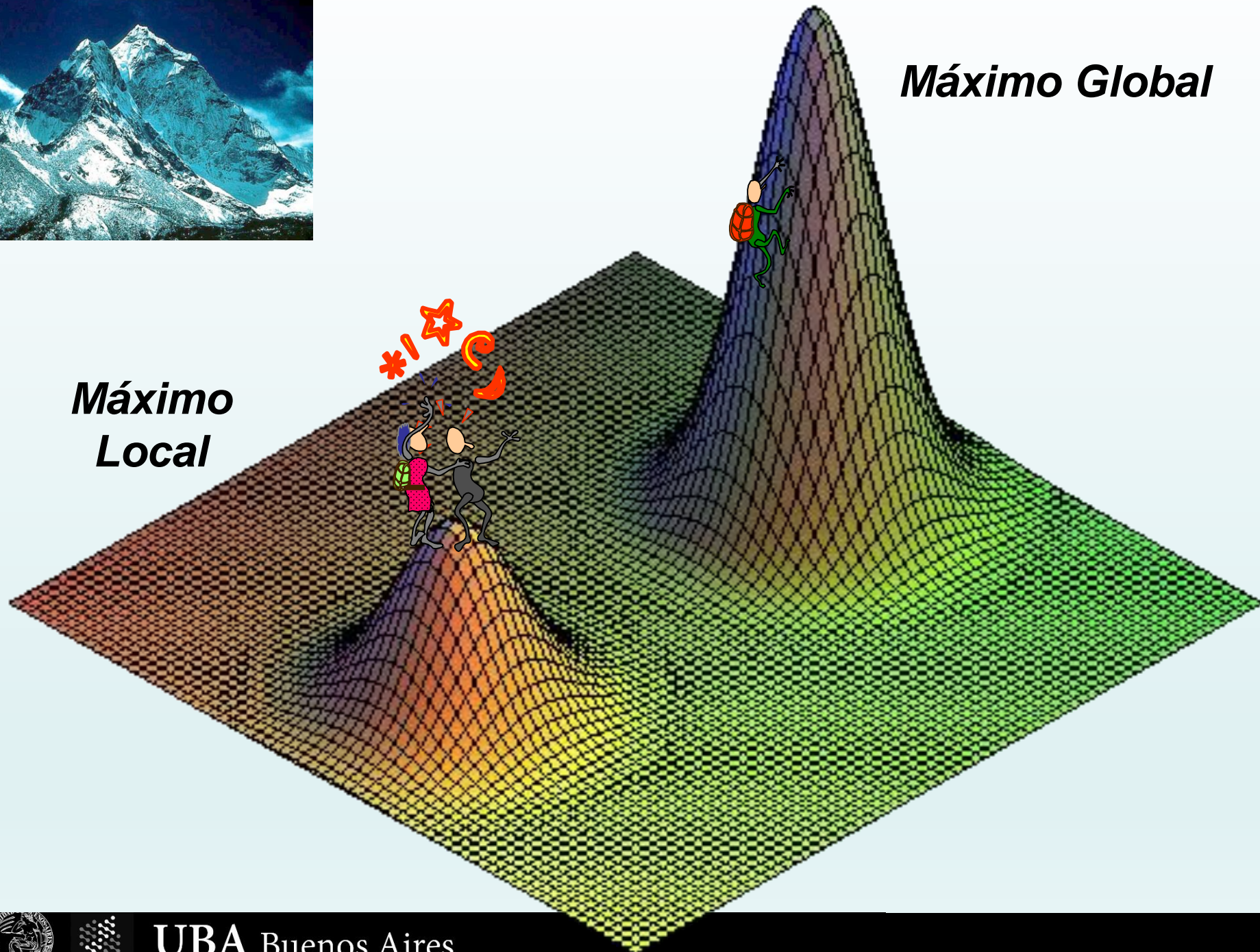
UBA Buenos Aires



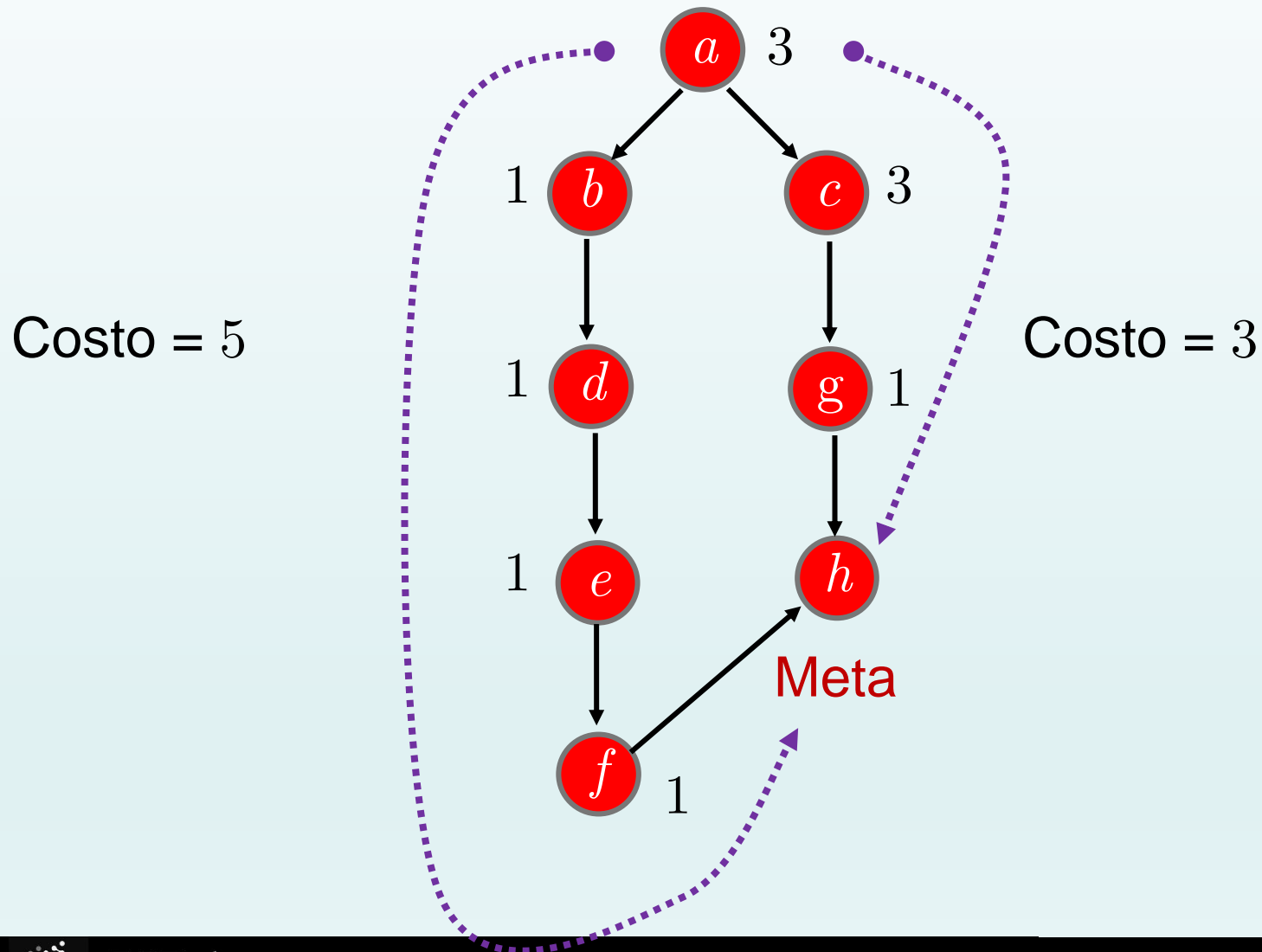


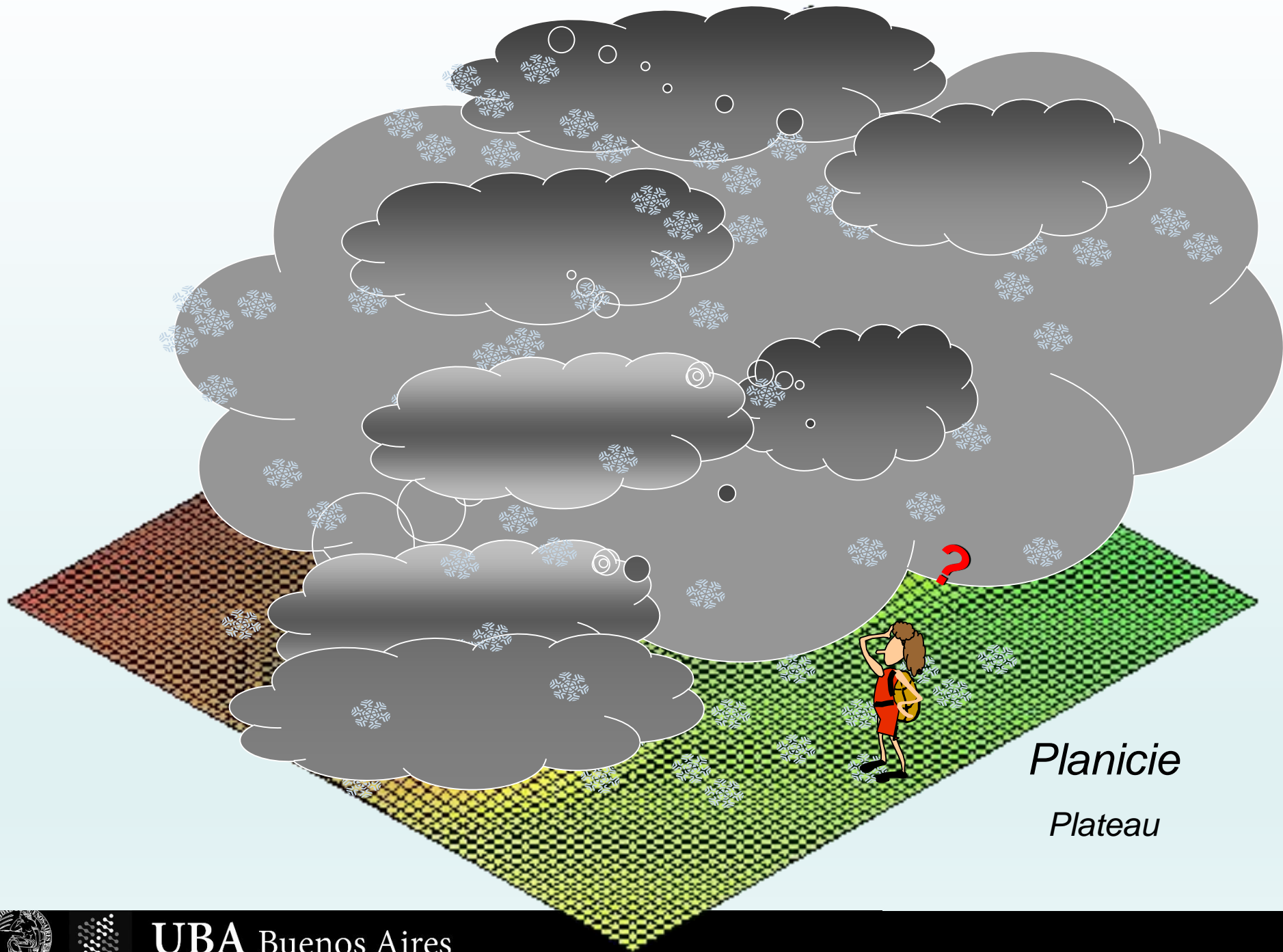
***Máximo Global***

***Máximo Local***



# Máximo Local vs. Máximo Global

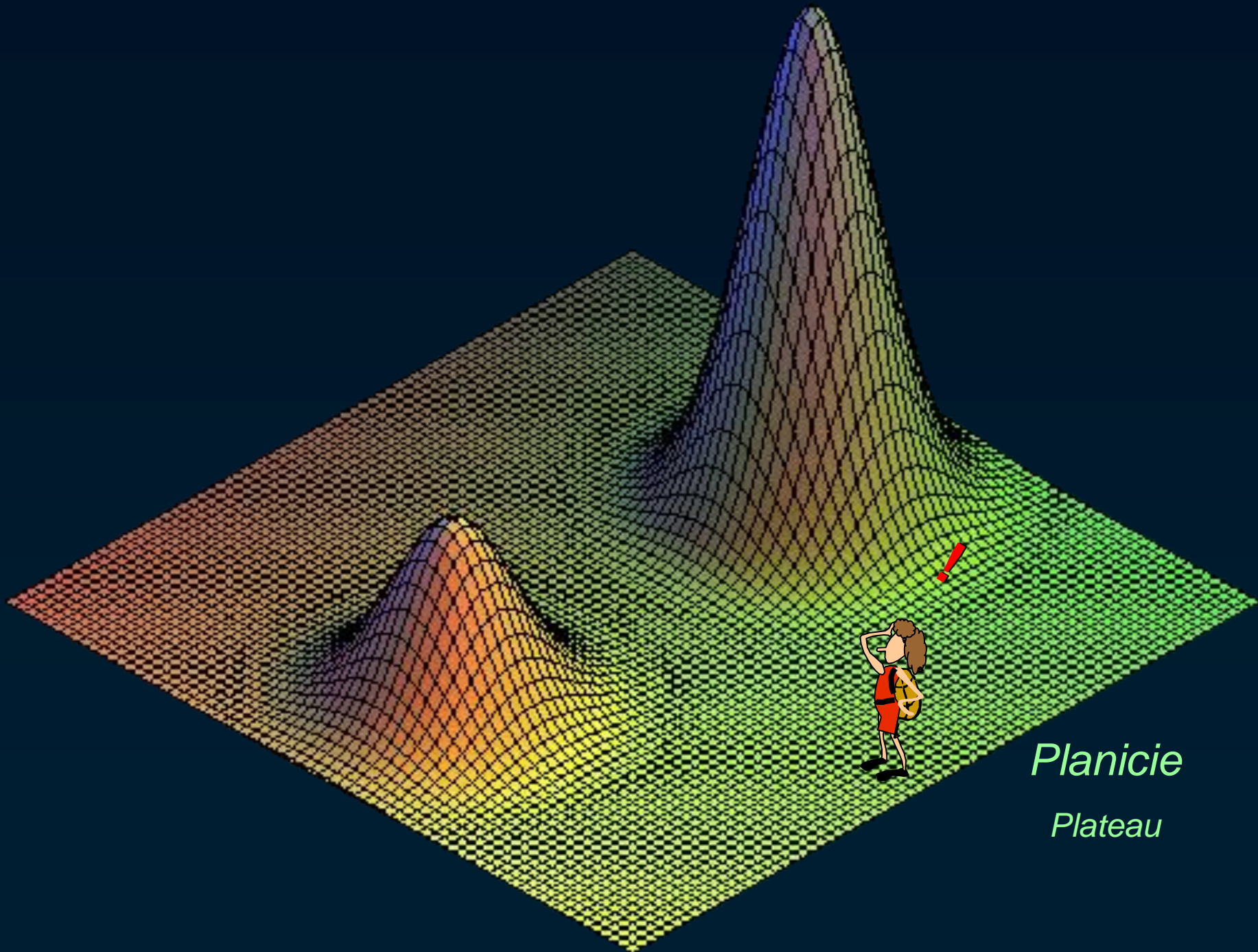




*Planicie*  
*Plateau*







*Planicie*

*Plateau*



# Planicie

2	4	8
7	3	5
6	1	

*Estado 1*

$$h_1(\text{Estado 1}) = 7$$

2	4	8
7	3	5
6		1

*Estado 2*

$$h_1(\text{Estado 2}) = 7$$

2	4	8
7	3	
6	1	5

*Estado 3*

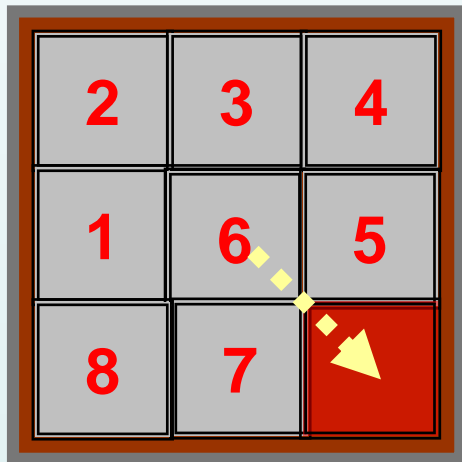
$$h_1(\text{Estado 3}) = 7$$

2	3	4
1		5
8	7	6

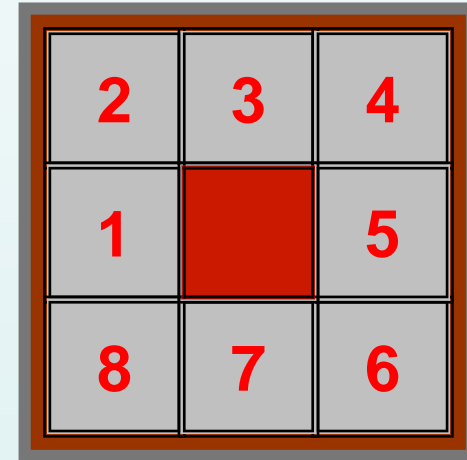
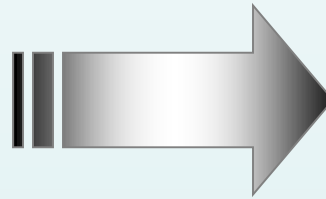
*Estado Final*



# El Problema del Borde (Ridge)



*Estado corriente*



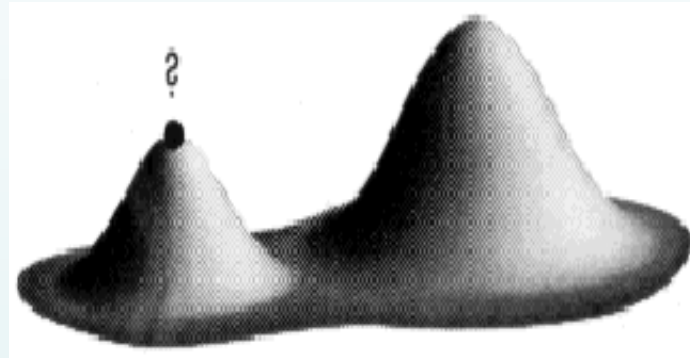
*Estado Final*

# Los tres problemas de la búsqueda greedy

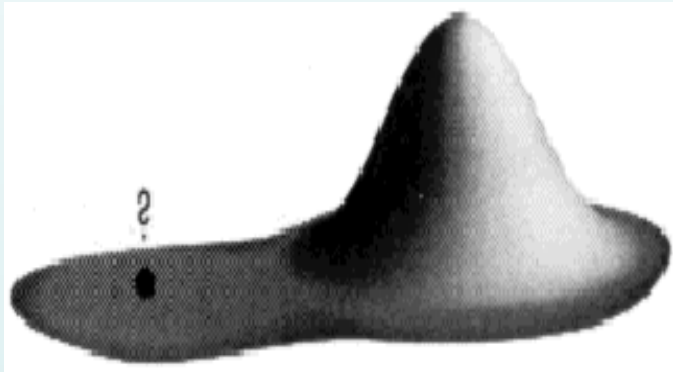
**Essentials of Artificial Intelligence**

**Matthew Ginsberg**

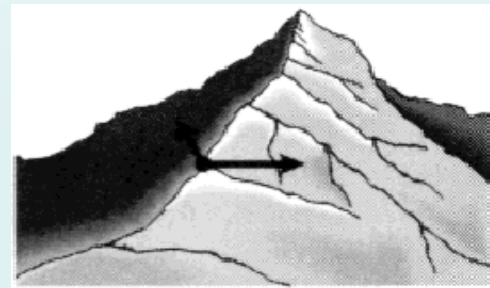
**Morgan Kaufmann, 1993**



*Máximo Local*



*Planicie*



*Borde*

# Como mejorar la situación

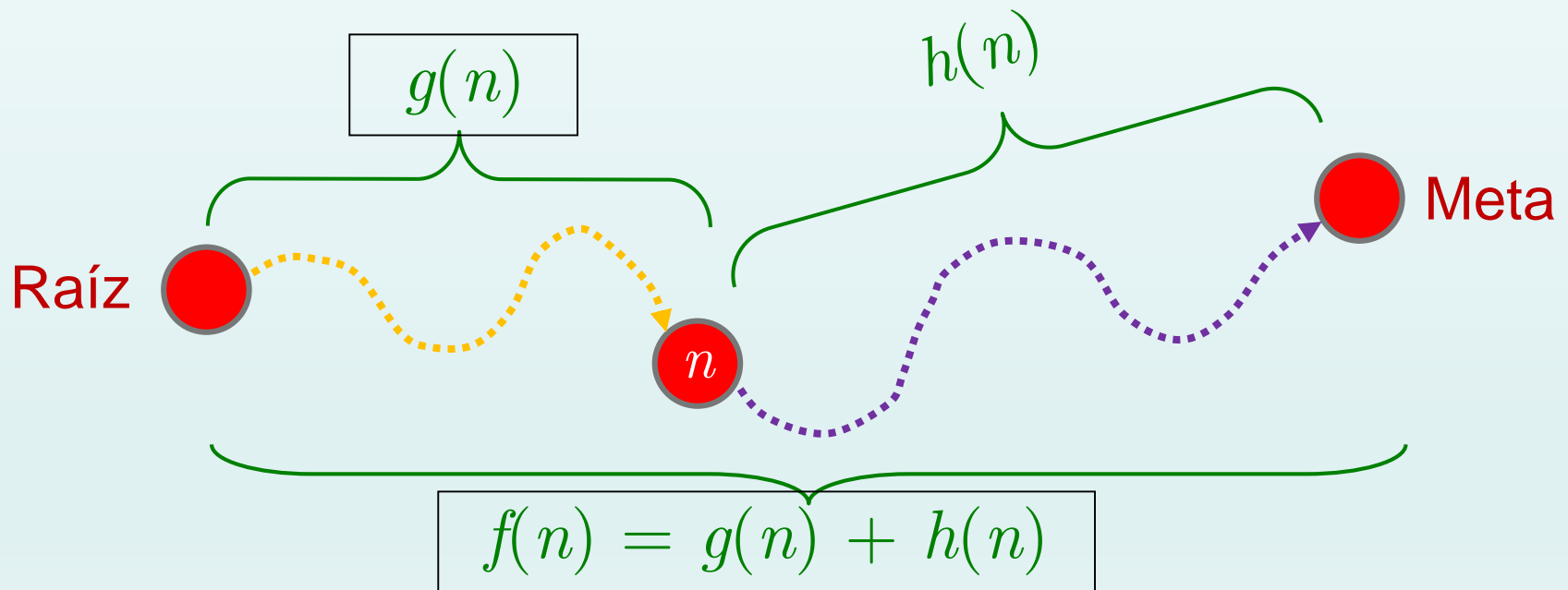
- Los problemas exhibidos en los ejemplos anteriores son motivados por la forma en que se utiliza la información heurística.
- En los algoritmos que presentamos la información que guía la búsqueda solo tiene en cuenta la predicción de la distancia a la meta.

# El Algoritmo $A^*$

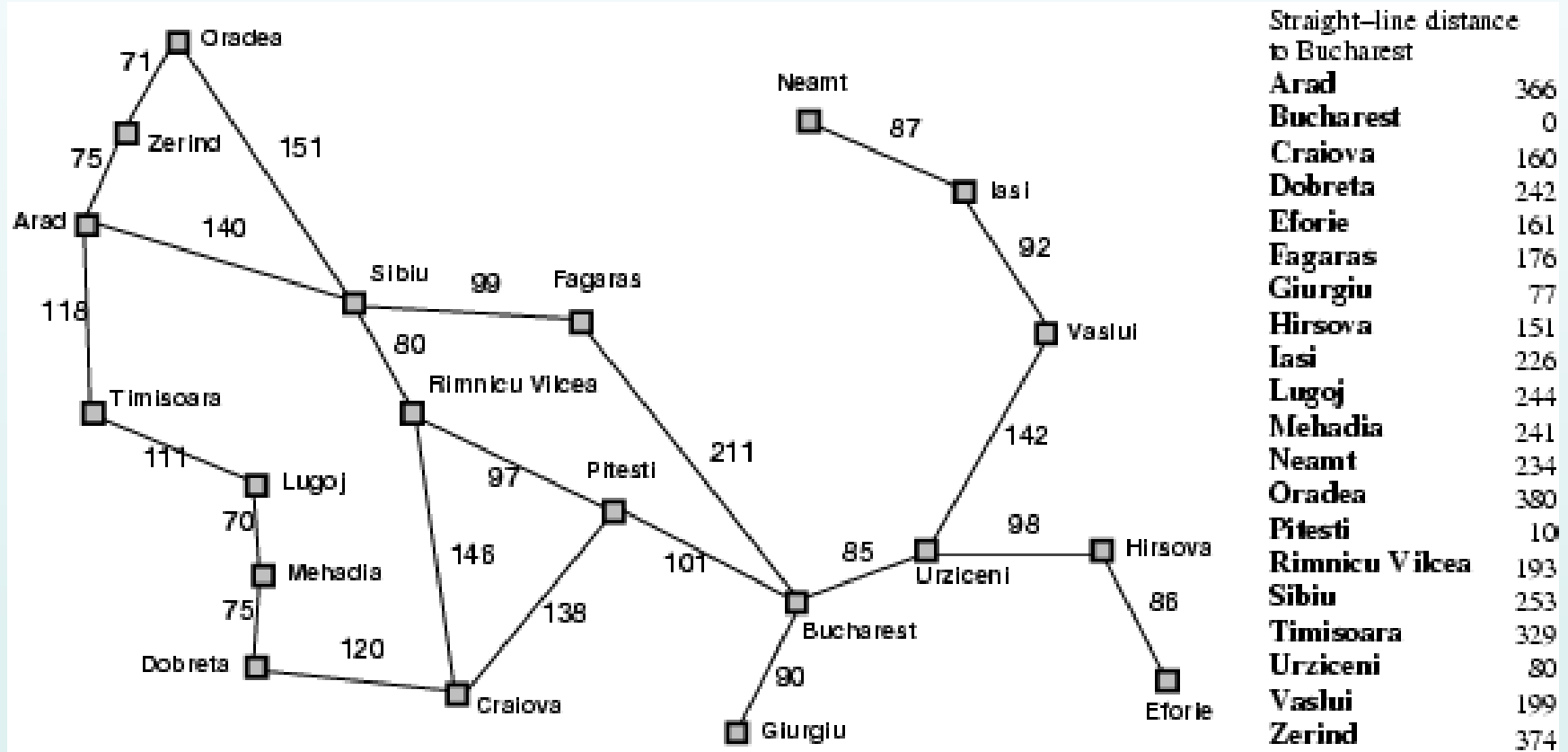
- Este algoritmo combina dos estrategias:
  - *Primero el menor costo hasta ese punto.*
  - *Primero el mejor estimado.*
- Es decir, se combina la información del costo exacto invertido hasta ese punto en la búsqueda y la predicción heurística sobre la distancia a la meta.
- Se selecciona la mejor estimación para una solución.

# El Algoritmo $A^*$

Dado un nodo  $n$  en la frontera sea  $f(n)$  la combinación del costo exacto para llegar hasta el nodo  $n$  desde el nodo raíz,  $g(n)$ , con la predicción heurística de la distancia a la meta,  $h(n)$ .




# Romania with step costs in km





# A\* search example

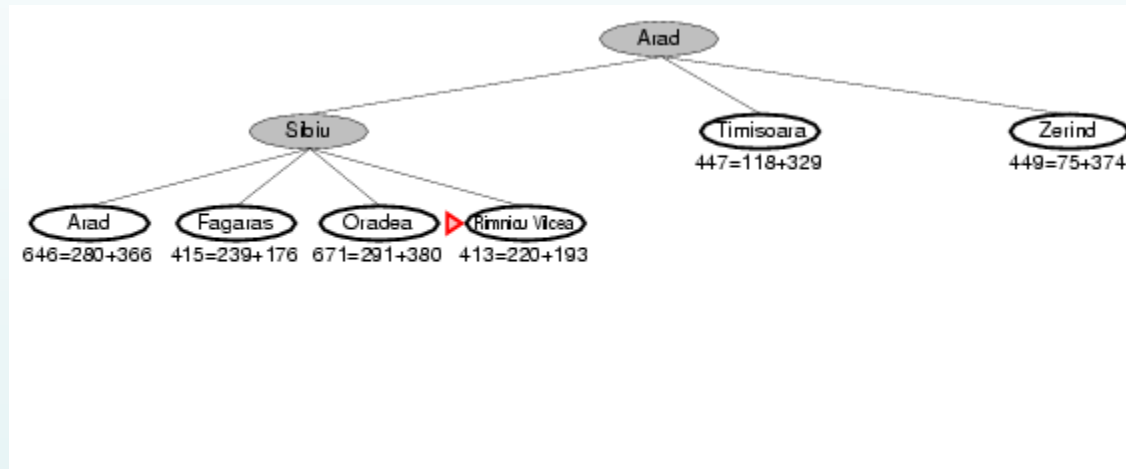


Arad  
366=0+366

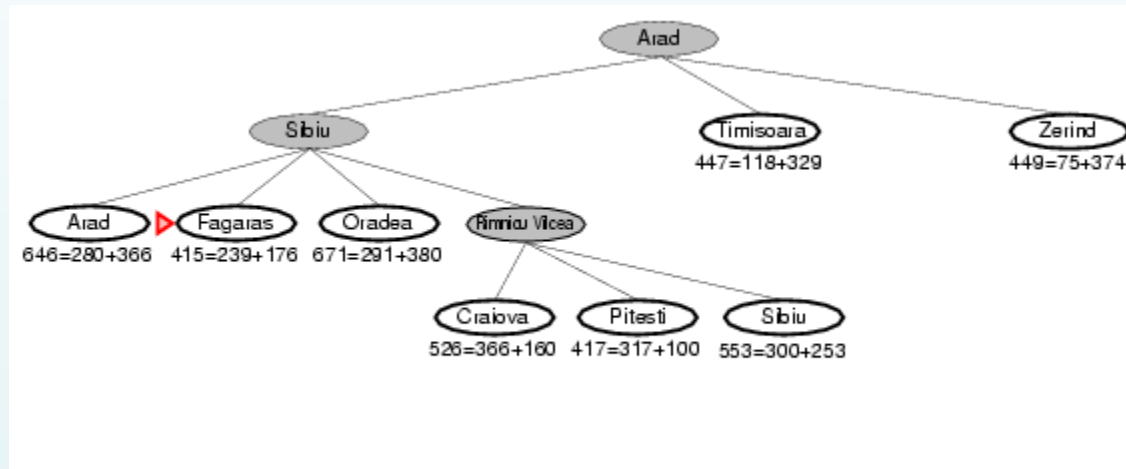
# A\* search example



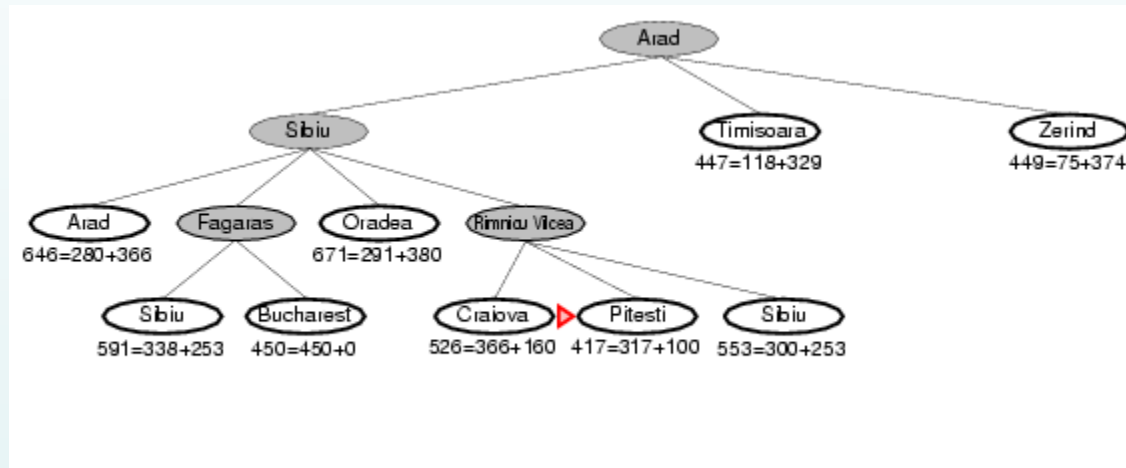
# A\* search example



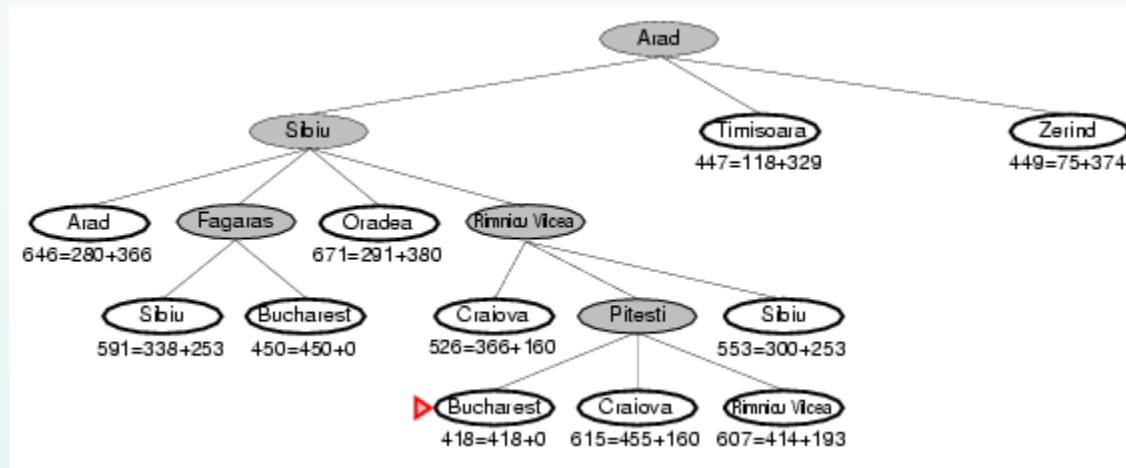
# A\* search example



# A\* search example



# A\* search example



# Admisibilidad del Algoritmo $A^*$

- La propiedad de encontrar siempre el paso optimal, si este existe, se denomina *admisibilidad*.
- Esto significa que aún en el caso de que el espacio de búsqueda sea infinito, si la solución existe, se encontrará la solución optimal.
- La siguiente proposición enuncia las condiciones para que esta propiedad se verifique.

# Admisibilidad del Algoritmo $A^*$

*Proposición (La admisibilidad de  $A^*$ ):* Si la solución existe,  $A^*$  siempre encontrará una solución y además la primera solución encontrada será optimal, si las siguientes condiciones se verifican:

- i. **el factor de ramificación es finito** (es decir, cada nodo tiene una cantidad finita de vecinos)*
- ii. El costo de los arcos es estrictamente positivo (es decir, existe  $\varepsilon > 0$  tal que el costo de cualquier arco es mayor que  $\varepsilon$ ), y*
- iii.  $h(n)$  es una cota inferior de mínimo costo del paso más corto desde el nodo  $n$  a un nodo meta.*

Dem: Ver Computational Intelligence, Poole et al., pág. 136-137.





# Sobre la subestimación de $h$

$$f(a) = 3$$

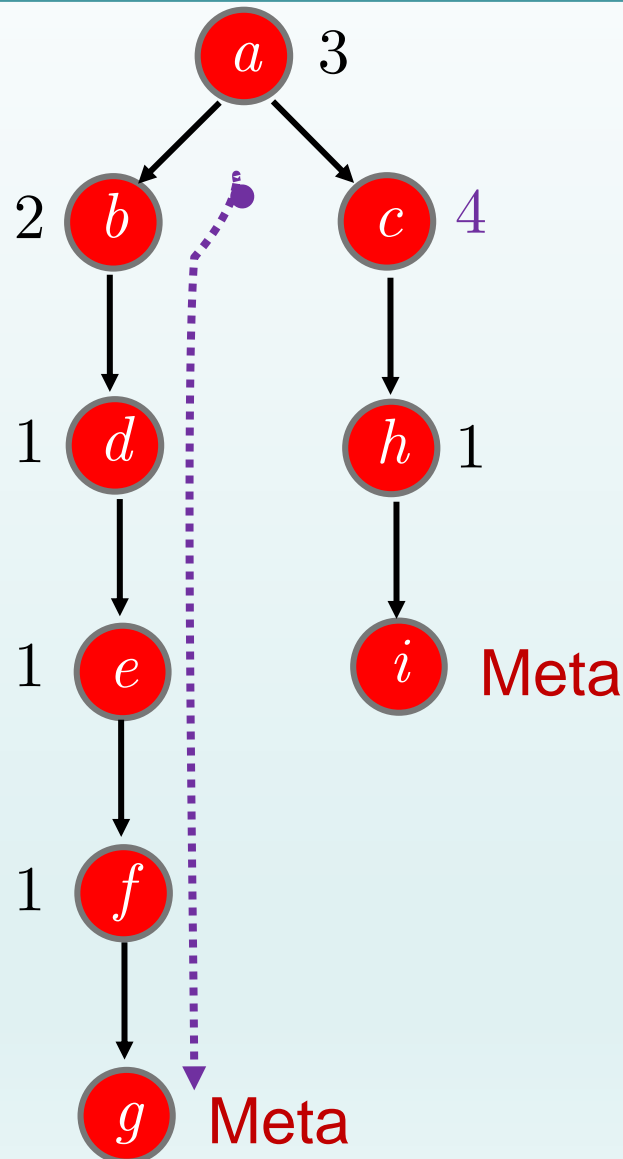
$$f(b) = 1 + 2 = 3$$

$$f(c) = 1 + 4 = 5$$

$$f(d) = 2 + 1 = 3$$

$$f(e) = 3 + 1 = 4$$

$h(c)$  no está  
subestimando el  
costo de llegar  
desde  $c$  a la  
meta!



# Flexibilización de Problemas

- Una problema flexibilizado/relajado es un problema con menos restricciones en sus acciones.
- El costo de una **solución optimal** a un problema relajado es una **heurística admisible** para el problema original.

## Ejemplo:

- Consideremos la versión de 8-puzzle donde una pieza se puede mover a cualquier lado, entonces  $h_1(n)$  [el nro de piezas fuera de lugar] nos da la solución mas corta.
- Si permitimos que las piezas pueden moverse a cualquier cuadrado adyacente, entonces  $h_2(n)$  [distancia Manhattan total] nos da la solución mas corta.
- Estas dos funciones son heurísticas admisibles para el problema original.



# Otras mejoras: Local beam search

Tener en cuenta  $k$  estados en lugar de solo uno.

- Arrancar con  $k$  estados generados de manera aleatoria.
- En cada iteración, se generan todos los sucesores de todos los estados  $k$ .
- Si alguno es un estado meta, entonces parar; sino elegir el mejor  $k$  de la lista completa y repetir.



# Ejercicios Tema 2

1. Implementar Algoritmo genérico de búsqueda (ver Slide 63)
2. Modifique minimamente el algoritmo del punto anterior para implementar los métodos de búsquedas “Primero en profundidad” y “Primero a lo ancho”.
3. ¿Es el método de búsqueda “Iterative deeping” completo? ¿Por qué?
4. Defina en sus propios términos los siguientes términos: autonomía, agente reactivo, agente basado en metas, agente basado en utilidades.
5. Escriba un programa en pseudo-código para los agentes basados en metas y basados en utilidad.
6. Consideremos un termostato simple que enciende una caldera cuando la temperatura esta al menos tres grados por debajo de la temperatura seteada, y la apaga cuando la temperatura esta al menos 3 grados por encima de la temperatura seteada. El termostato, ¿es una instancia de un agente reactivo? ¿un agente basado en metas? ¿o un agente basado en utilidad?