

Octubre 2014



# Resolución de Problemas en IA

*Prof. María Auxiliadora Pérez*  
*E-mail : maperez@ucla.edu.ve*

# Inteligencia Artificial

En los años 60, Alan Newell y Herbert Simon, trabajando en la demostración de teoremas y el ajedrez por computador logran crear un programa llamado GPS (General Problem Solver: solucionador general de problemas).



Éste era una sistema en el que el usuario definía un entorno, en función de una serie de objetos y los operadores que se podían aplicar sobre ellos.



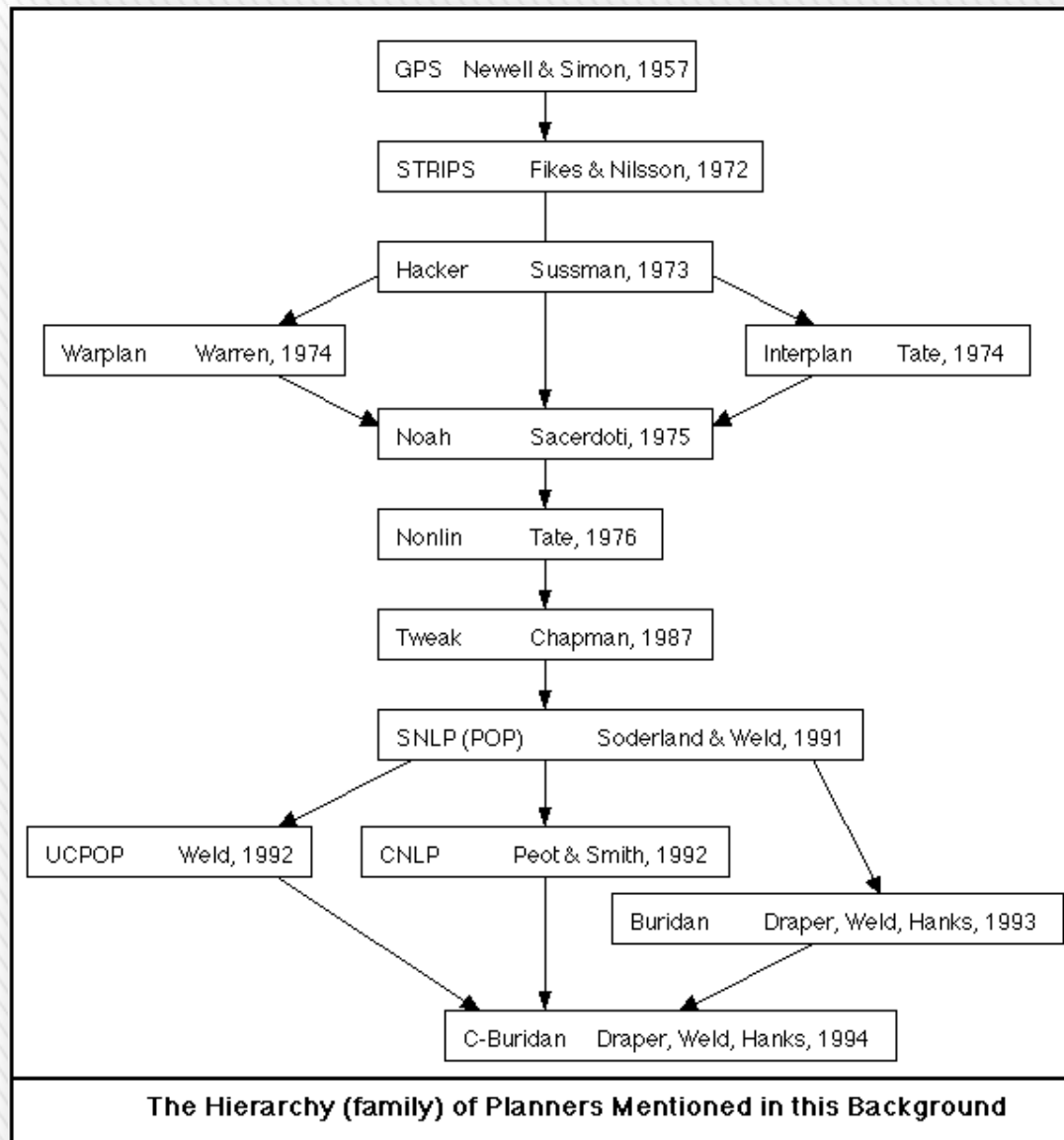
Este programa era capaz de trabajar con las torres de Hanoi, así como con criptoaritmética y otros problemas similares, operando, claro está, con microcosmos formalizados que representaban los parámetros dentro de los cuales se podían resolver problemas.



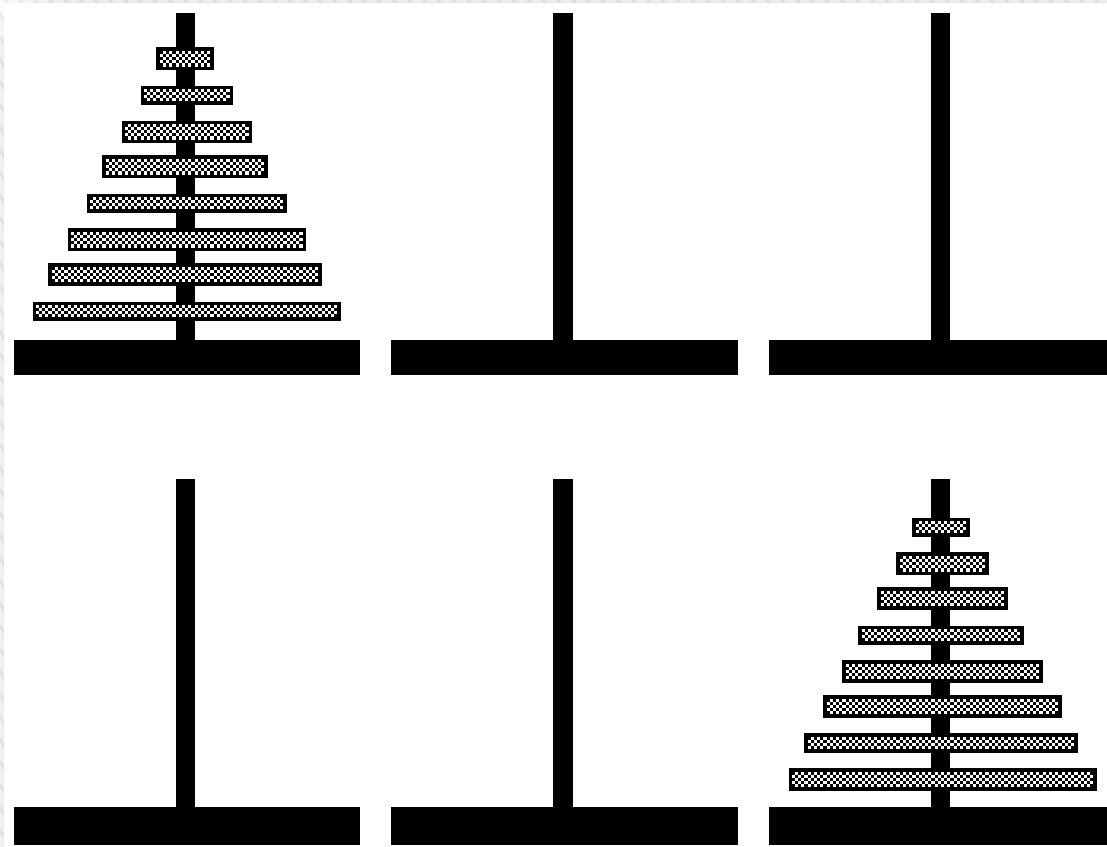
Lo que no podía hacer el GPS era resolver problemas ni del mundo real, ni tomar decisiones importantes. El GPS manejaba reglas heurísticas (aprender a partir de sus propios descubrimientos) que la conducían hasta el destino deseado mediante el método del ensayo y el error.



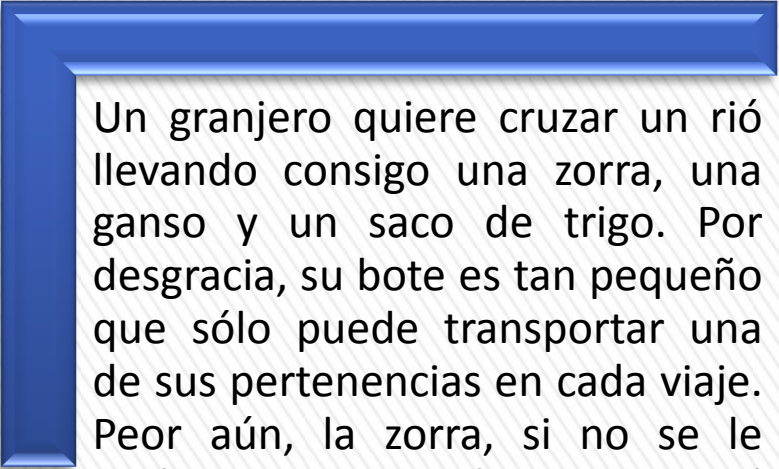
# Inteligencia Artificial



# Torres de Hanoi



# El granjero, la zorra, el ganso y el trigo



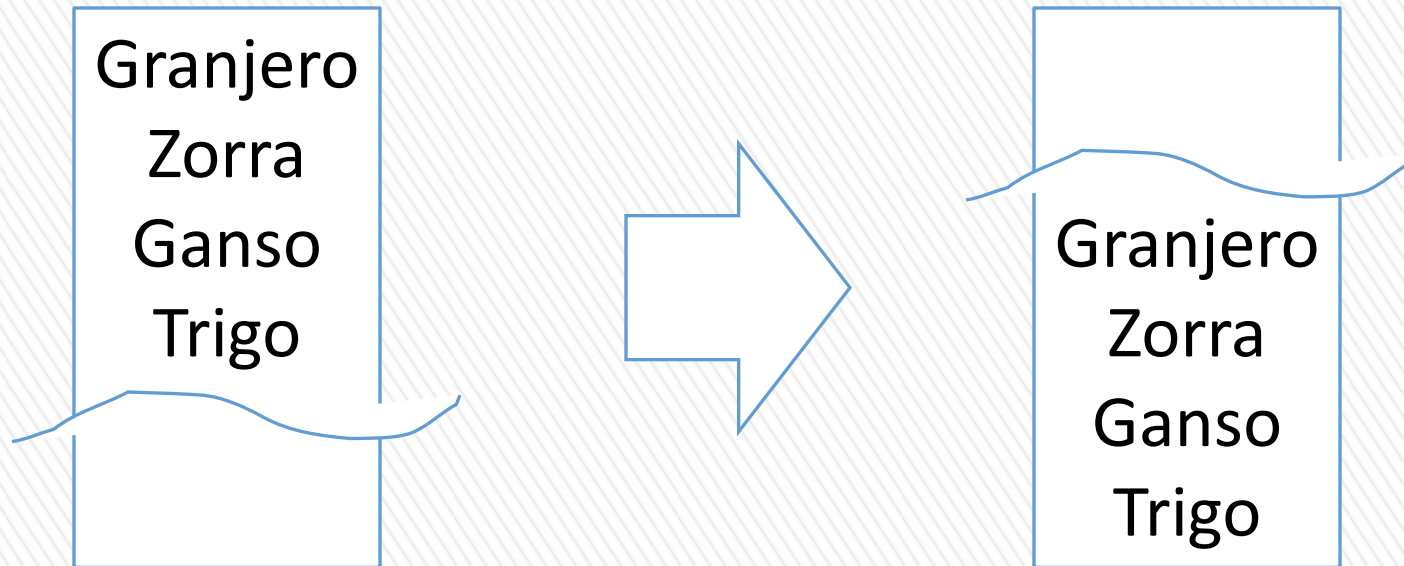
Un granjero quiere cruzar un río llevando consigo una zorra, un ganso y un saco de trigo. Por desgracia, su bote es tan pequeño que sólo puede transportar una de sus pertenencias en cada viaje. Peor aún, la zorra, si no se le vigila, se come al ganso, y el ganso, si no se le cuida, se come el trigo; de modo que el granjero no debe dejar a la zorra sola con el ganso o al ganso solo con el trigo.



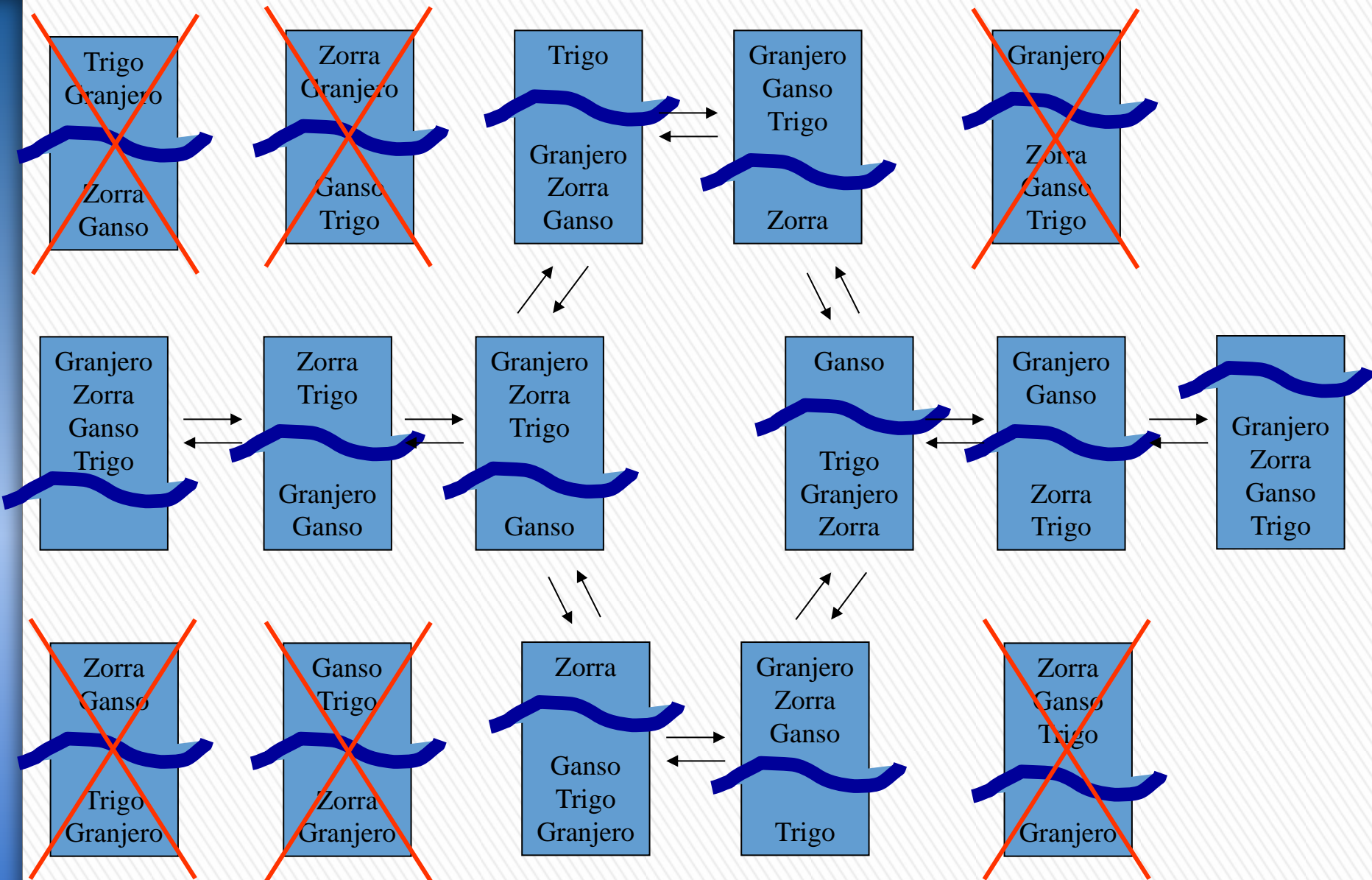
**¿qué puede hacer?**



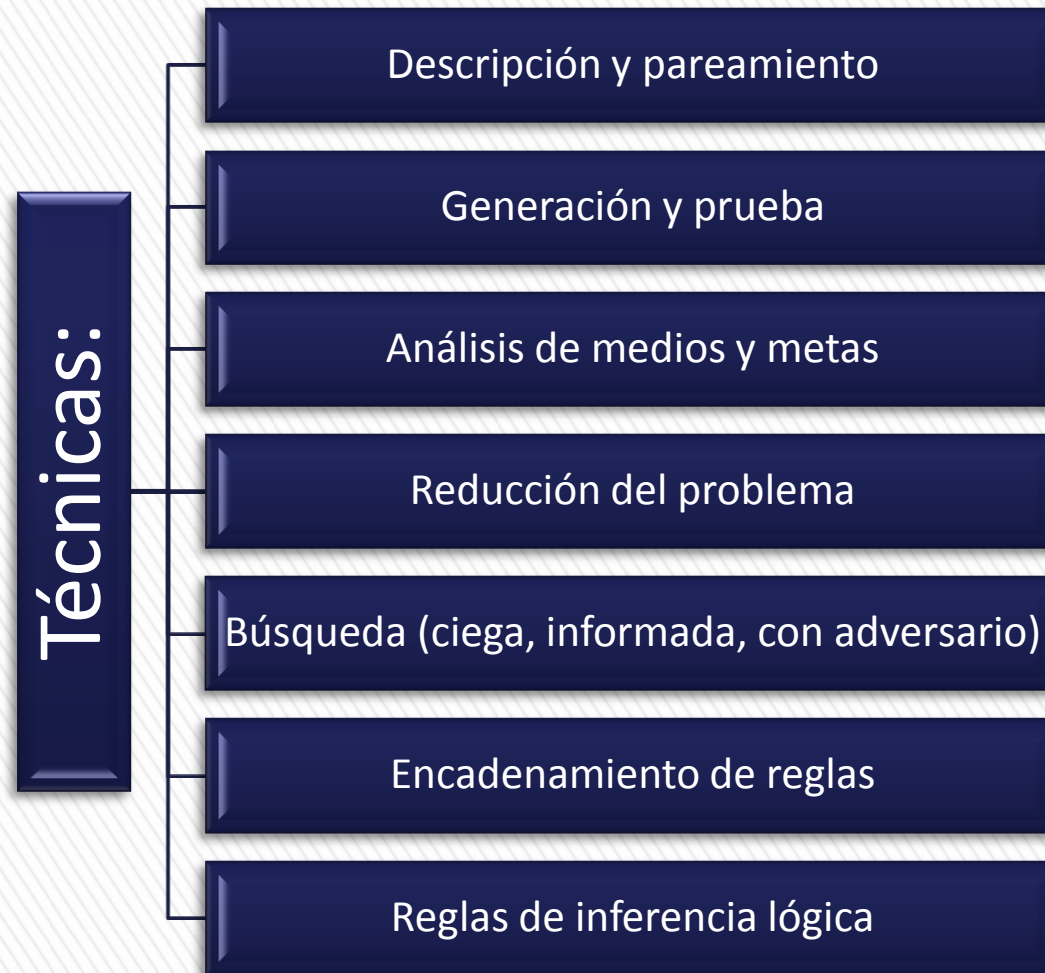
# ¿El problema?





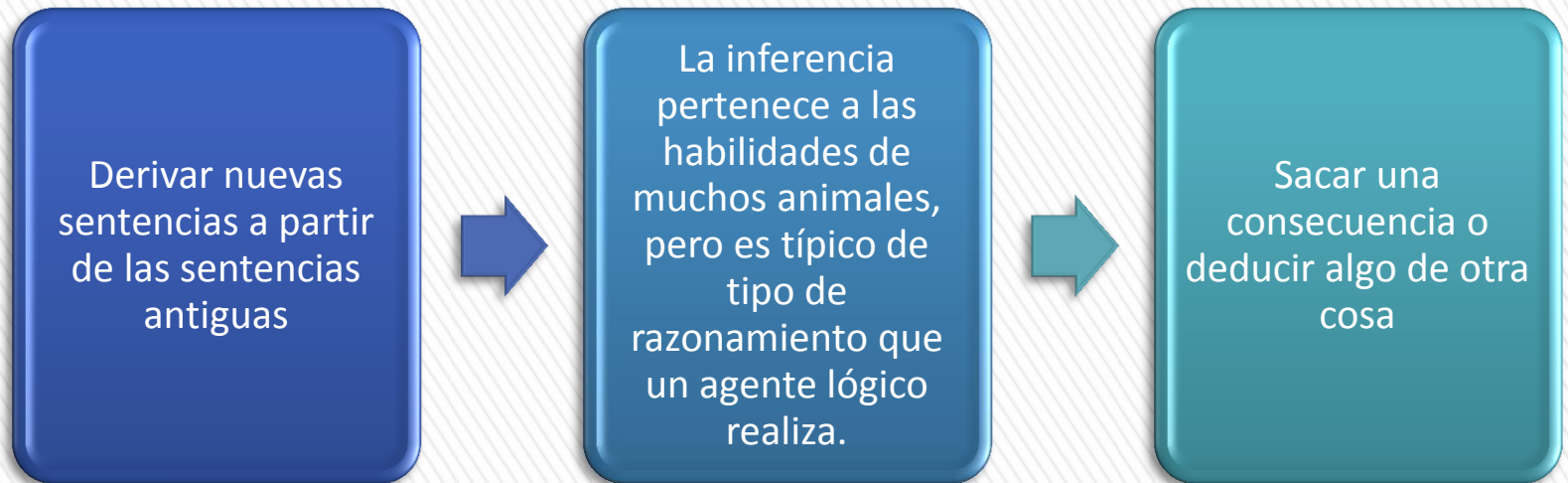


# Resolución de Problemas

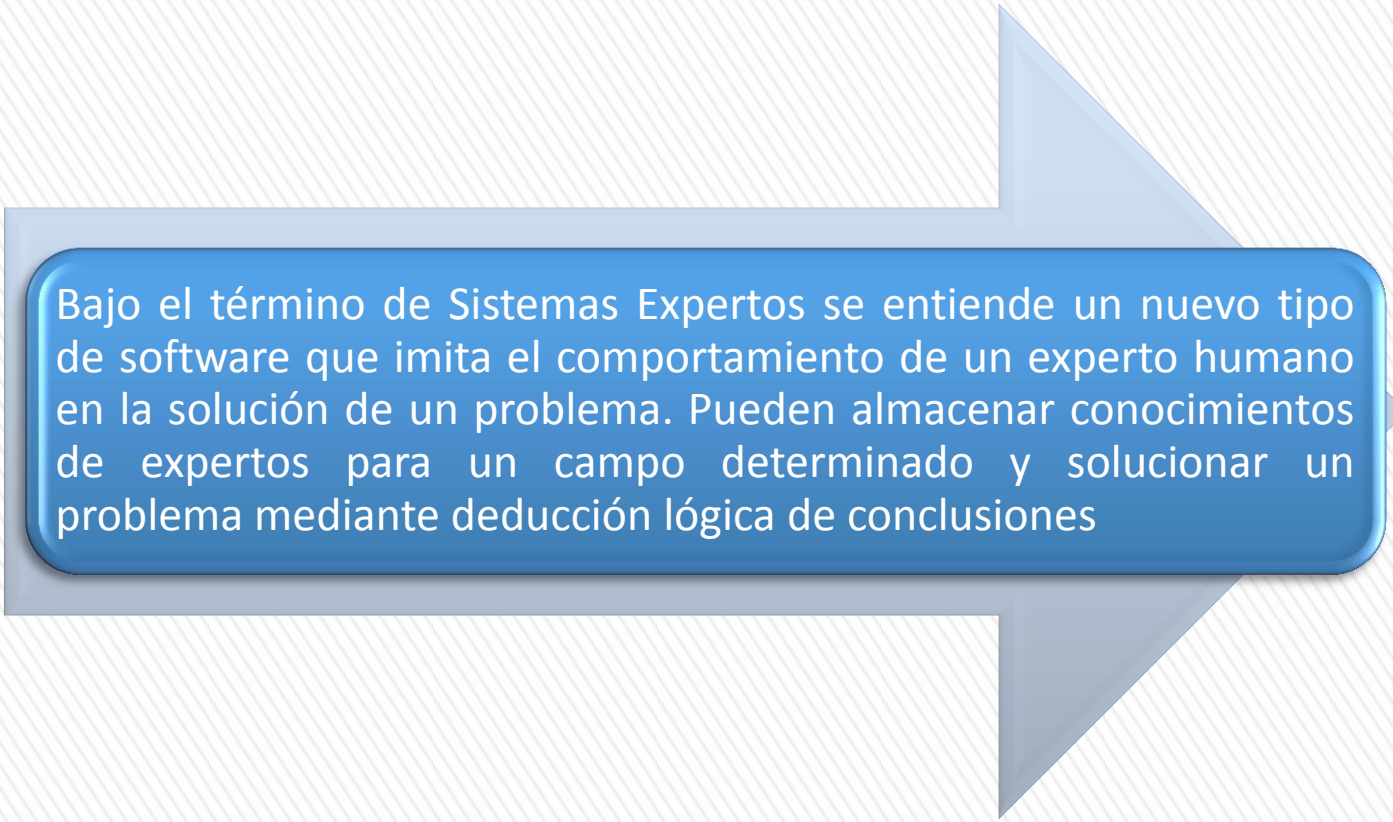




# Inferencia Lógica



# Qué es un Sistema Experto?



Bajo el término de Sistemas Expertos se entiende un nuevo tipo de software que imita el comportamiento de un experto humano en la solución de un problema. Pueden almacenar conocimientos de expertos para un campo determinado y solucionar un problema mediante deducción lógica de conclusiones



# Sistemas Expertos




Los Sistemas Expertos son uno de los puntos que componen las investigaciones en el campo de la IA. Un sistema que trabaje con técnicas de IA deberá estar en situación de combinar información de forma "inteligente", alcanzar conclusiones y justificarlas (al igual que el resultado final). Los Sistemas Expertos son una expresión de los sistemas basados en el conocimiento. Con la aplicación de técnicas de Inteligencia Artificial finaliza la transición del procesamiento de datos al procesamiento de conocimientos.



# Sistemas Expertos

Los sistemas expertos se aplican por norma general en problemas que implican un procedimiento basado en el conocimiento y comprende las siguientes capacidades:




Utilización de normas o estructuras que contengan conocimientos y experiencias de expertos especializados.




Deducción lógica de conclusiones.

Capaz de interpretar datos ambiguos.

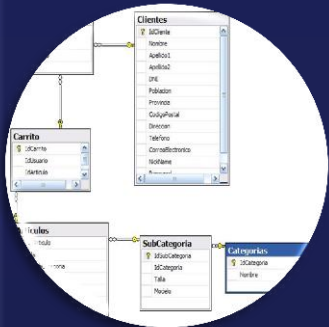
Manipulación de conocimientos afectados por valores de probabilidad.



Una característica decisiva de los Sistemas Expertos es la separación entre conocimiento (reglas, hechos) por un lado y su procesamiento por el otro. A ello se añade una interface de usuario y un componente explicativo.

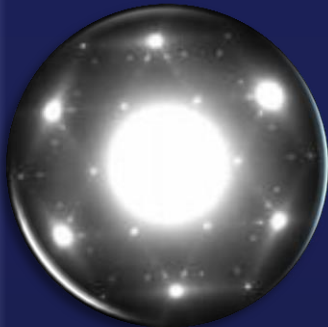


# Componentes de un Sistema Experto



## Base de Conocimientos

Conocimiento de los hechos y de las experiencias de los expertos en un dominio determinado



## Mecanismo de Inferencia

Permite simular la estrategia de solución de un experto



## Componente Explicativo

Explica al usuario la estrategia de solución encontrada y el porqué de las decisiones tomadas.



## Interface de Usuario

Realizar una consulta en un lenguaje lo más natural posible

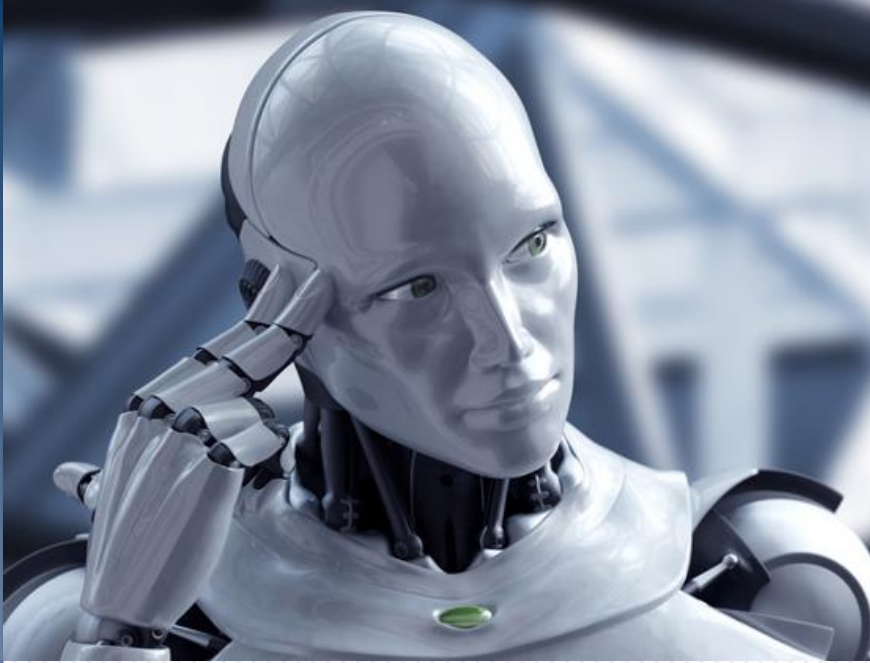


## Componente de Adquisición

Estructuración e implementación del conocimiento en la base de conocimientos







Octubre 2014



# Estrategias de Búsqueda



# Problema de Búsqueda

**El problema de las jarras de agua:** “Se tienen dos jarras sin marcas de medición y con capacidades 4 y 3 litros. Asumiendo que se tiene agua suficiente, cómo medir exactamente 2 litros en la jarra con capacidad 4?”

## Representación:

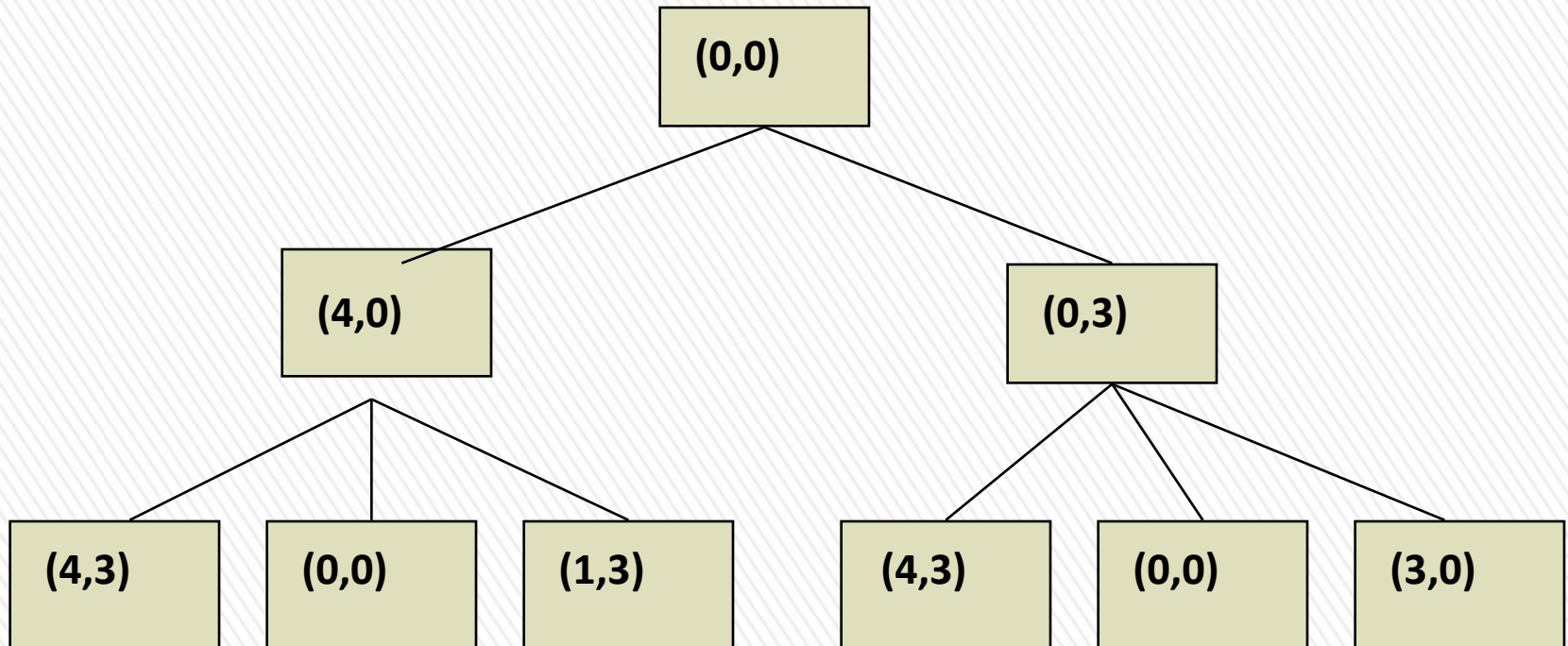
*Espacio de Estados:* Pares ordenados  $(x,y)$   
 $(c4,c3)$

*Estado inicial:*  $(0,0)$

*Estado final:*  $(2,n)$



# Problema de Búsqueda



# Estrategias de Búsqueda

Dado un estado inicial y varios posibles estados sucesores:

## **Estrategias sistemáticas:**

No tienen preferencias entre ellos.  
Difieren entre sí en el orden de expansión de nodos.

## **Estrategias heurísticas :**

Hacen uso del conocimiento del dominio para elegir entre ellos, generando mayor efectividad en la búsqueda.



# Métodos de búsqueda a ciegas

Métodos que  
no usan  
ningún  
conocimiento  
específico  
acerca del  
problema

Primero en profundidad

---

Primero en amplitud

---

Búsqueda No-determinística

---

Profundización Iterativa

---

Búsqueda Bi-direccional

---



## Estrategias Sistemáticas

- Primero en Amplitud (BFS)

- **Orden de expansión de nodos:**

Nodo raíz , sus sucesores (nivel  $d$ ), los sucesores de ellos (nivel  $d + 1$ )

- **Algoritmo:** general con inserción posterior en la cola



## Estrategias Sistemáticas

### • Primero en Amplitud (BFS)

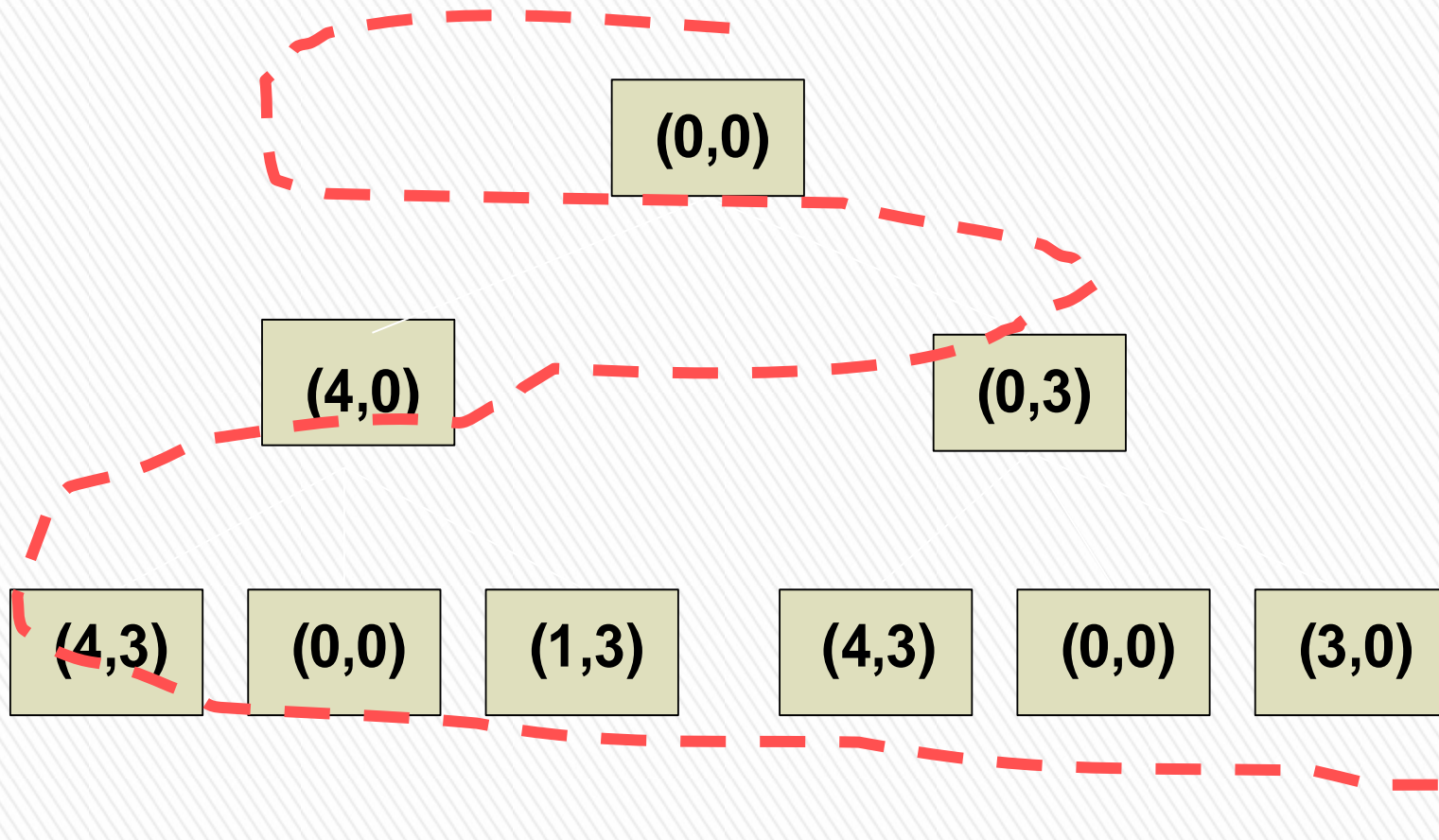
1. Inserte en una cola el elemento raíz (nodo de partida)
2. Hasta que el elemento frontal sea el nodo meta, o se vacié la cola
  - a) Si nodo frontal tiene hijos, insertar todos sus hijos al final de la cola.
  - b) Eliminar nodo frontal.
3. Si el nodo meta se alcanza, mencione éxito, de lo contrario, notifique el fracaso.





## Estrategias Sistemáticas

- Primero en Amplitud (BFS)



## Estrategias Sistemáticas

- Primero en Amplitud (BFS)

- » Completitud ☒ si  $r$  es finito
- » Complejidad en tiempo  $O(r^p)$
- » Complejidad en espacio  $O(r^p)$
- » Optimalidad ☒ siempre que el costo sea no-decreciente



• Costo Uniforme

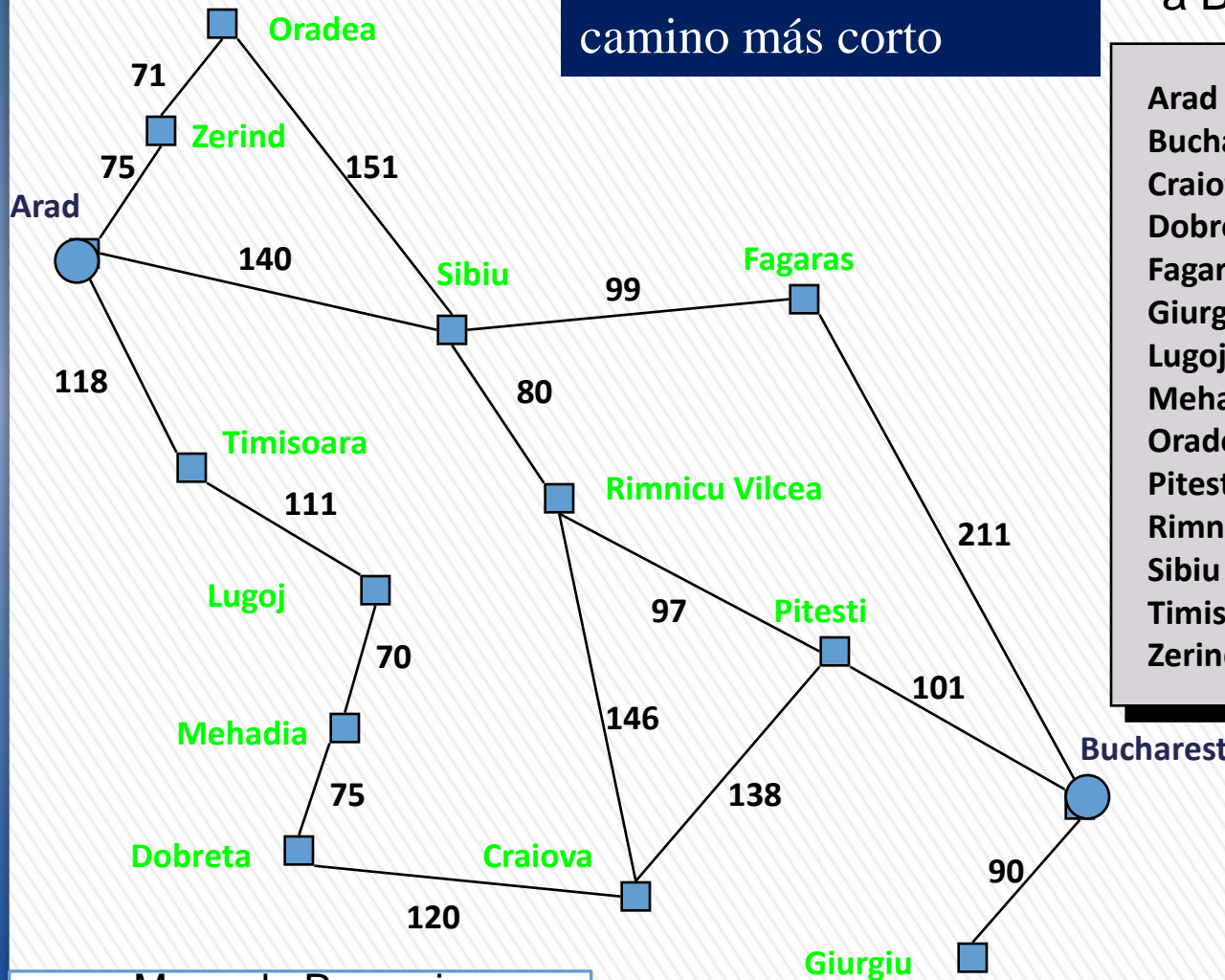
- » Modifica a primero-en-amplitud expandiendo primero el nodo de la agenda con menor costo ( $g(n)$ )
- » Si  $(g(n)) = (\text{profundidad}(n))$ , ambas son equivalentes



# Costos de recorridos en Rumania en Kms

**Objetivo:** Ir de Arad a Bucharest utilizando el camino más corto

Dist. en línea recta a Bucharest (kms.)

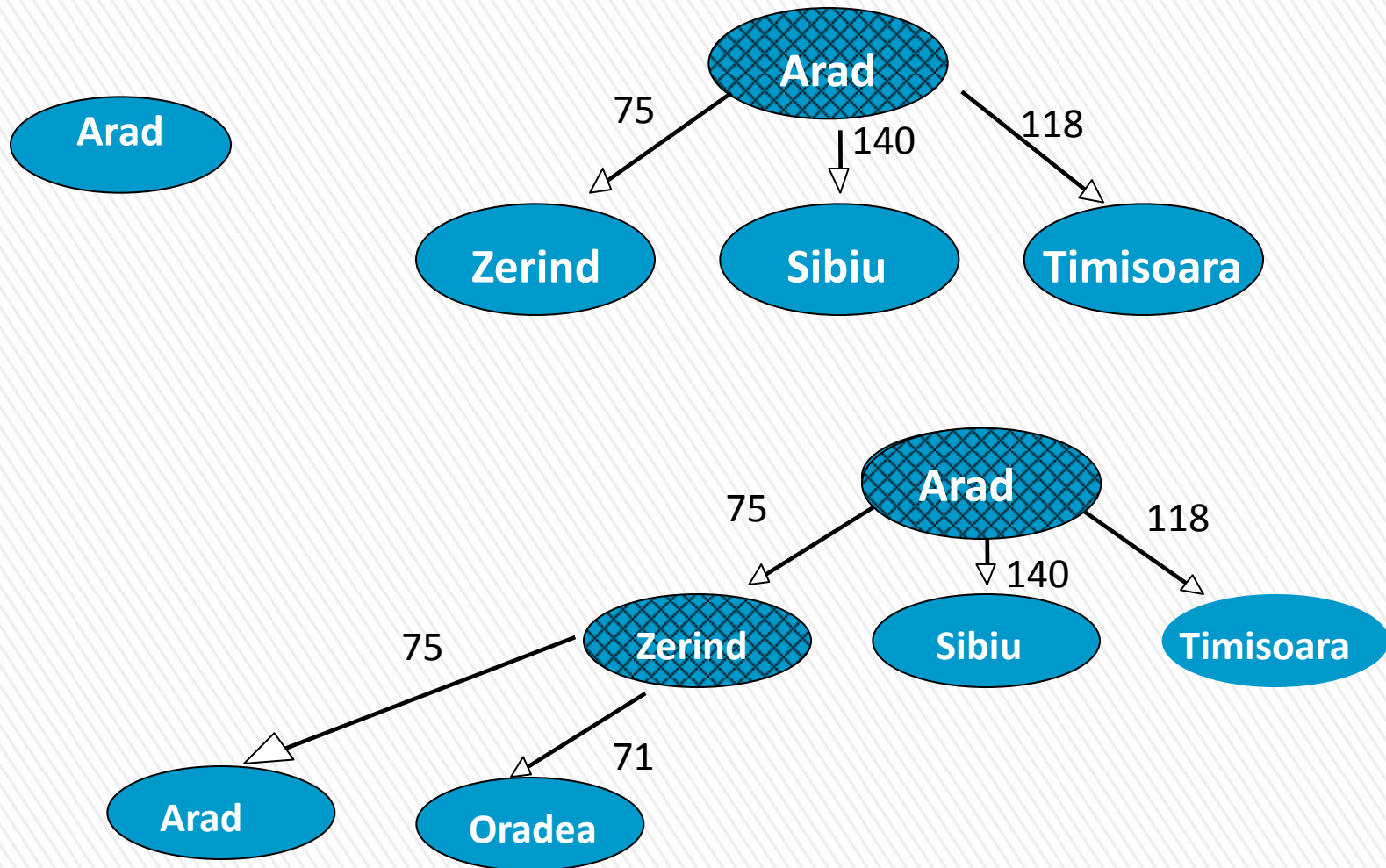


Arad	366
Bucharest	0
Craiova	160
Dobreta	242
Fagaras	178
Giurgiu	77
Lugoj	244
Mehadia	241
Oradea	380
Pitesti	98
Rimnicu Vilcea	193
Sibiu	253
Timisoara	329
Zerind	374

Mapa de Rumania  
(Dist. entre Ciudades)

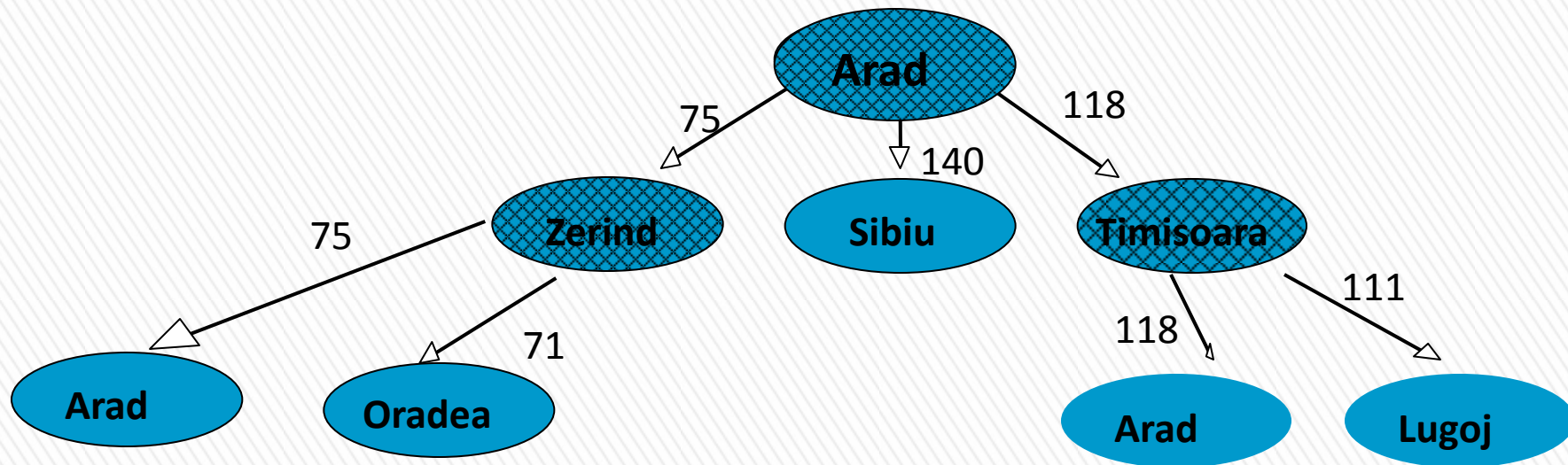
# Estrategias Sistemáticas

## • Costo Uniforme





## Estrategias Sistemáticas

### • Costo Uniforme





- » Completitud  si costo no decrece
- » Complejidad en tiempo  $O(r^p)$
- » Complejidad en espacio  $O(r^p)$
- » Optimalidad 



## Estrategias Sistemáticas



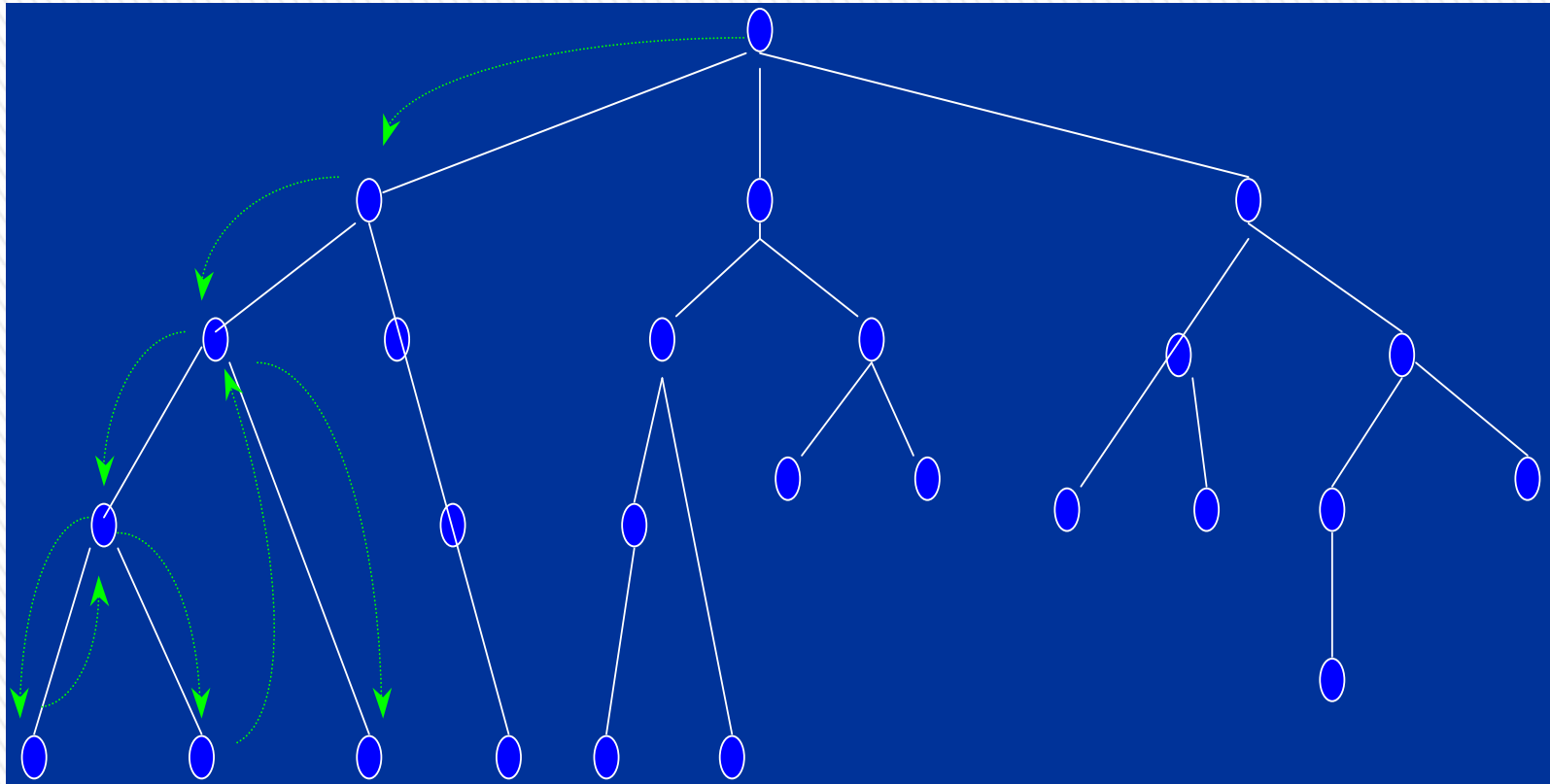
### Primero en Profundidad (DFS)

1. Inserte en una pila el elemento raíz (nodo de partida)
2. Hasta que el elemento tope sea el nodo meta, o se vacié la pila
  - ☐ Si nodo tope tiene hijos, insertar el hijo siguiente aun no visitado, según ordenamiento.
  - ☐ Si no, entonces eliminar nodo tope.
3. Si el nodo meta se alcanza, mencione éxito, de lo contrario, notifique el fracaso.



# Estrategias Sistemáticas

## Primero en Profundidad (DFS)



## Estrategias Sistemáticas



### Primero en Profundidad (DFS)



## Problemas:

- Puede caer en ciclos infinitos
- Requiere de espacio de búsqueda finito, no-cíclico (o chequear por estados repetidos)



## Estrategias Sistemáticas



- Completitud Sólo si espacio es finito
- Complejidad en tiempo  $O(r^m)$
- Complejidad en espacio  $O(r \cdot m)$
- Optimalidad No





- » Controla límite de profundidad en nivel de búsqueda: **diámetro**, ampliándolo iterativamente
- » Combina beneficios de primero en amplitud (óptimo y completo) y en profundidad (modestos requerimientos de memoria)
- » Expande nodos como en primero en amplitud, pero algunos nodos se expanden varias veces





Estrategias  
Sistemáticas



- Profundización Iterativa

```
function PROFUNDIZACION-ITERATIVA (problema)  
  returns una secuencia de solución  
inputs: problema, un problema  
for profundidad  $\leftarrow 0$  to  $\infty$  do  
  resultado  $\leftarrow$  PROFUNDIZACION-  
    ITERATIVA (problema, profundidad)  
  if resultado  $\neq$  corte then return resultado  
end
```



Estrategias  
Sistemáticas



- Profundización Iterativa

$L = 0$

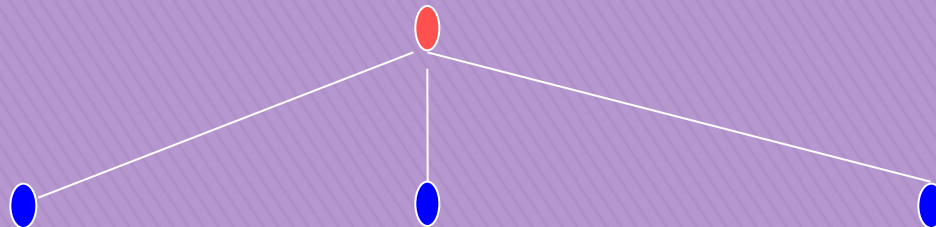


Estrategias  
Sistemáticas



- Profundización Iterativa

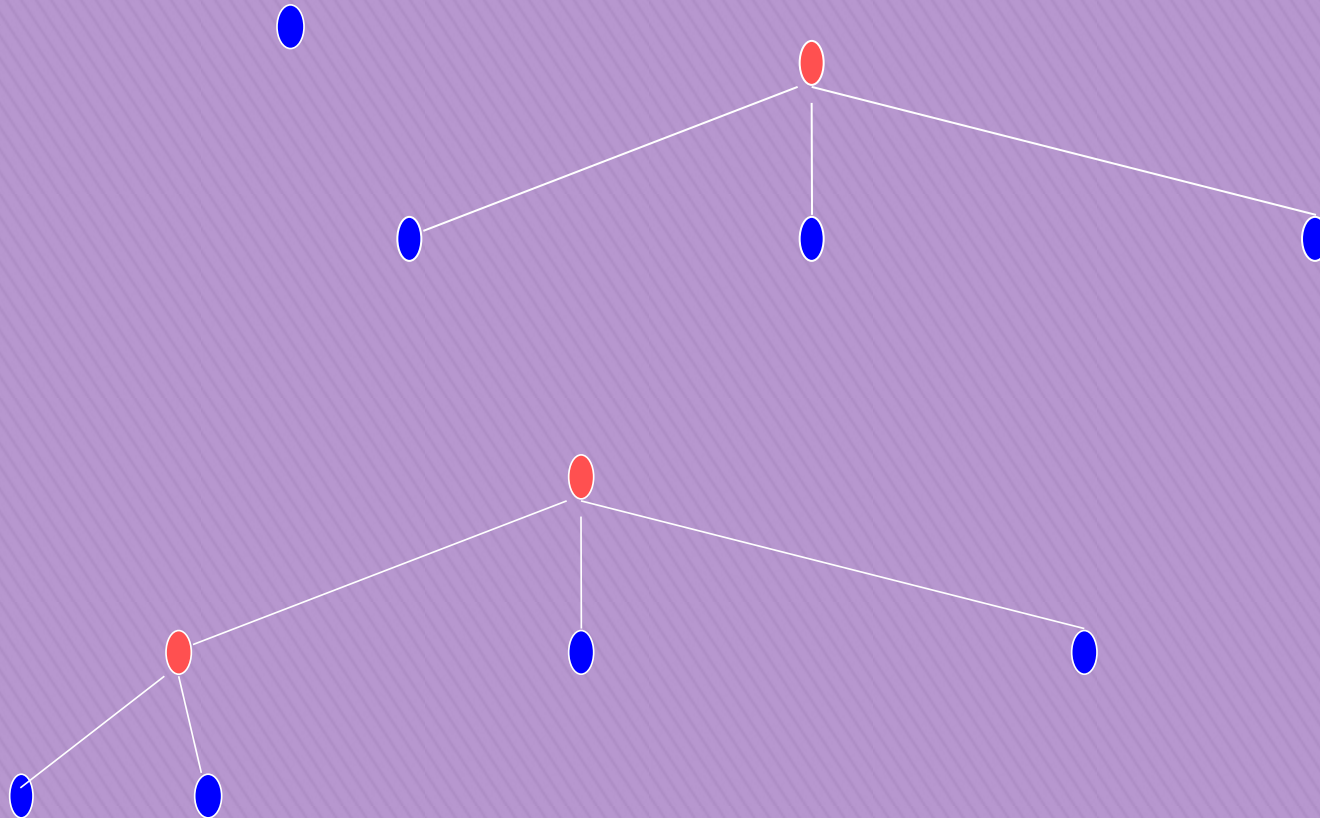
$L = 1$



Estrategias  
Sistemáticas

- Profundización Iterativa

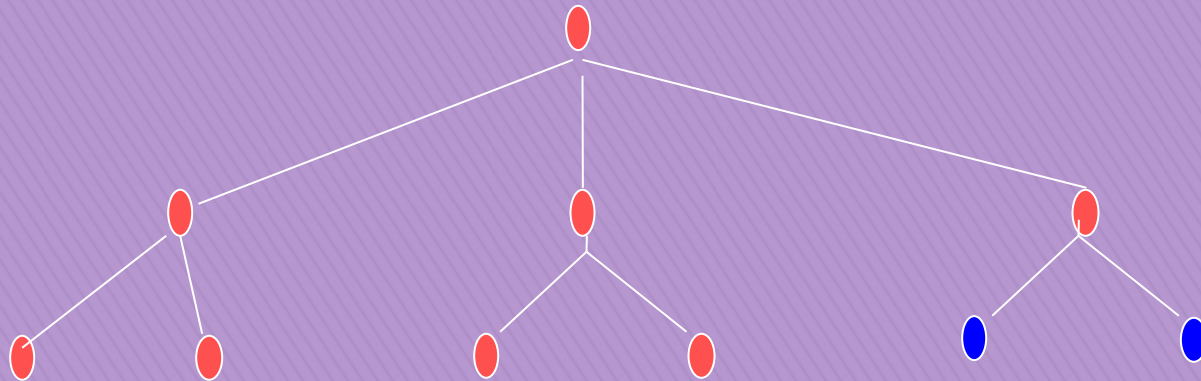
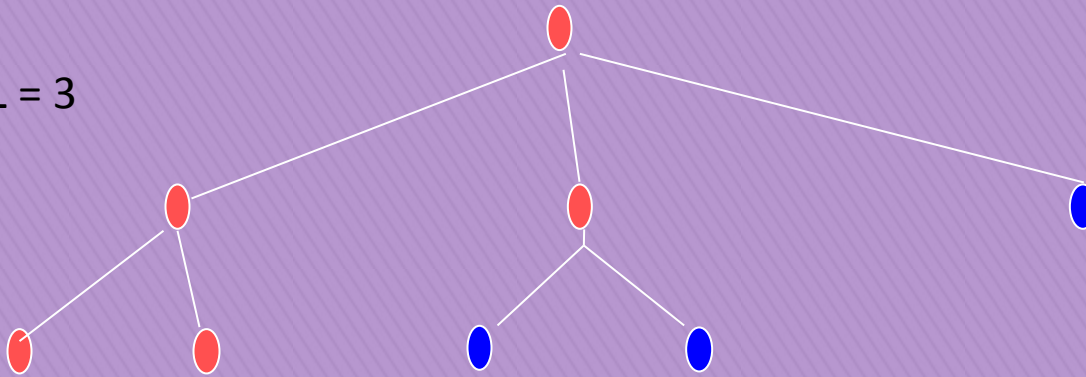
$L = 2$



Estrategias  
Sistemáticas

- Profundización Iterativa

$L = 3$





- Completitud Sí
- Complejidad en tiempo  $O(r^p)$
- Complejidad en espacio  $O(r^p)$
- Optimalidad Sí, si costo de  $c/\text{tramo}=1$





## Estrategias Heurísticas

- Cuando se dispone de más información que el estado inicial, los operadores y el test de estado objetivo, el tamaño del espacio de búsqueda usualmente puede reducirse.
- A mejor información disponible, más eficientes procesos de búsqueda.
- Tales métodos o estrategias son basados en información heurística.



# Estrategias Heurísticas

- **Heurísticas:**

(Del v. griego *heuriskein*: encontrar, descubrir). Reglas empíricas o técnicas de juicio que, en algunos casos, conducen a una solución, aunque no garantizan el éxito.

- **Ejemplos:**

- Vendedor Viajero: Dist. a estado objetivo
- 8-puzzle: N° fichas fuera de sitio o Manhattan



## Estrategias Heurísticas

- Su eficacia depende en gran medida de la forma en que exploten el conocimiento del dominio particular del problema.
- Proporcionan un marco donde situar el conocimiento del dominio específico.
- Forman el núcleo de la mayoría de los sistemas de IA.



## Estrategias Heurísticas

- Su eficacia depende en gran medida de la forma en que exploten el conocimiento del dominio particular del problema.
- Proporcionan un marco donde situar el conocimiento del dominio específico.
- Forman el núcleo de la mayoría de los sistemas de IA.



# Estrategias Heurísticas

- Algoritmo General

```
function BUSQUEDA-GENERAL (problema, INSERTA-FN  
  returns una solución, o falla  
nodos ← HACER-COLA (HACER-NODO  
  (ESTADO-INICIAL[problema]))  
loop do  
  if nodos está vacío then return falla  
  nodo ← REMUEVE-FRENTE (nodos)  
  if test-meta [ problema] aplicado a  
    ESTADO(nodo) tiene éxito  
  then return nodo  
  nodos ← INSERTA-FN (nodos, EXPANDE (nodo,  
    OPERADORES[problema]))  
end
```



- Utiliza una función heurística  $h(n)$  para estimar el costo necesario del camino más económico para alcanzar el nodo objetivo a partir del nodo actual.
- Una buena función heurística para problemas de búsqueda de ruta es la **distancia en línea recta al objetivo**.



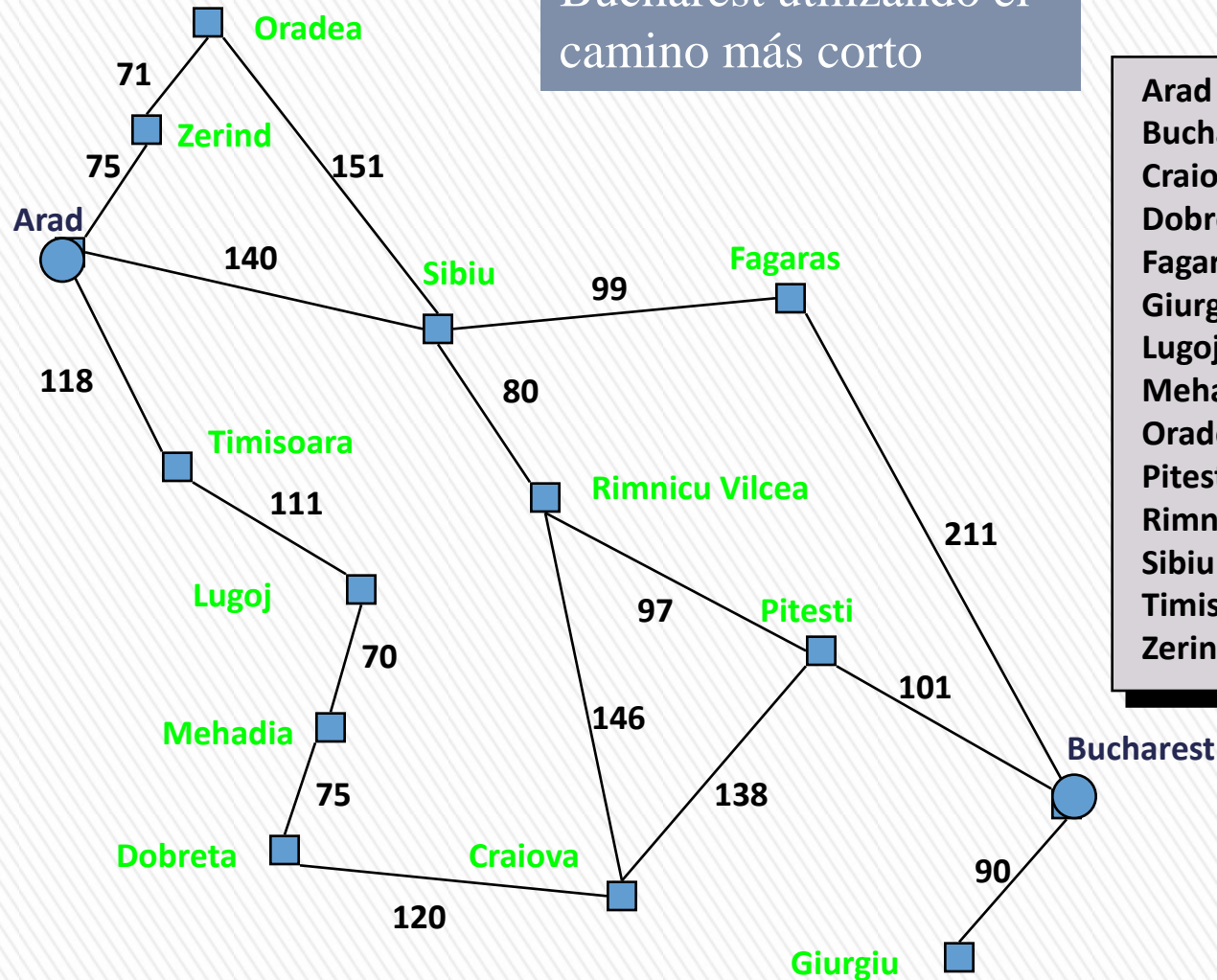


# Costos de recorridos en Rumania en Kms

Mapa de Rumania  
(Dist. entre Ciudades)

**Objetivo:** Ir de Arad a  
Bucharest utilizando el  
camino más corto

Dist. en línea recta  
a Bucharest (kms.)



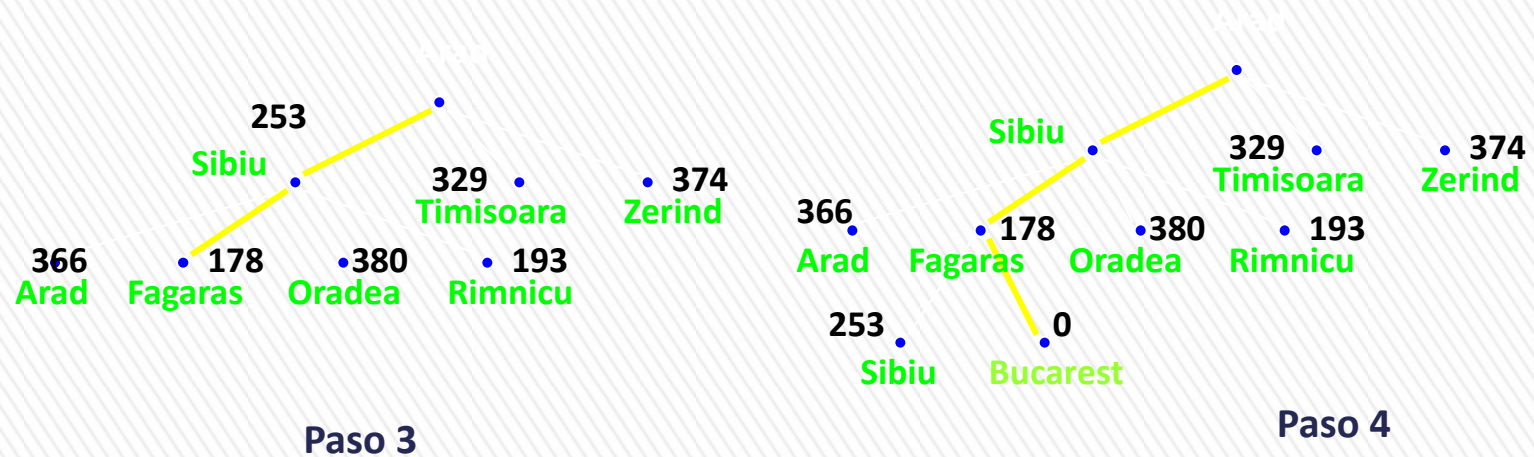
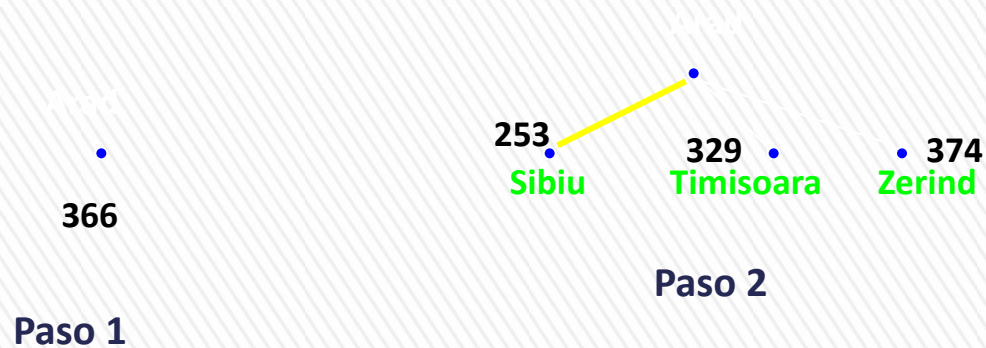
Arad	366
Bucharest	0
Craiova	160
Dobreta	242
Fagaras	178
Giurgiu	77
Lugoj	244
Mehadia	241
Oradea	380
Pitesti	98
Rimnicu Vilcea	193
Sibiu	253
Timisoara	329
Zerind	374



# Estrategias Heurísticas

## Primero el Mejor

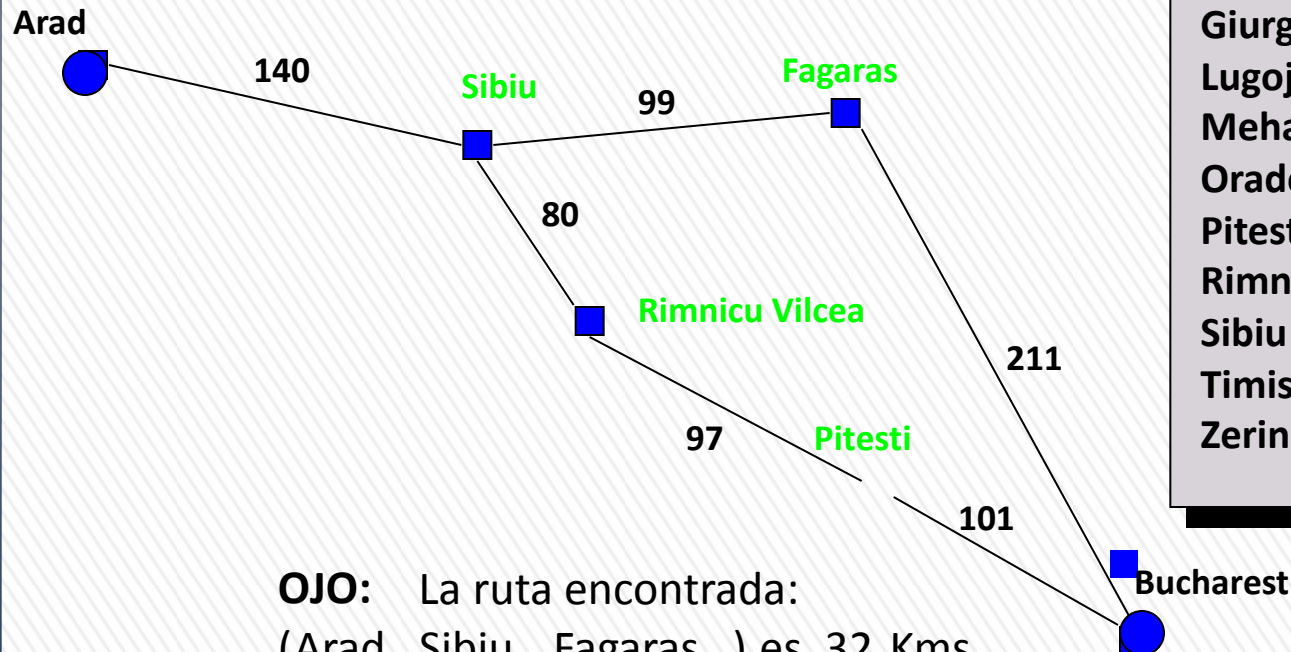
Función heurística:  
Dist . en línea recta  
al objetivo



# Análisis del Resultado

**Objetivo:** Ir de Arad a Bucharest utilizando el camino más corto

Mapa de Rumania  
(Dist. entre Ciudades)



Dist. en línea recta  
a Bucharest (Kms)

Arad	366
Bucharest	0
Craiova	160
Dobreta	242
Fagaras	178
Giurgiu	77
Lugoj	244
Mehadia	241
Oradea	380
Pitesti	98
Rimnicu Vilcea	193
Sibiu	253
Timisoara	329
Zerind	374

**OJO:** La ruta encontrada:  
(Arad , Sibiu , Fagaras ) es 32 Kms .  
más larga que la ruta:  
(Arad , Sibiu , Rimnicu , Pitesti )



- **Completitud** No, puede quedar atrapado en loops. Ej Giurgiu  $\leftarrow$  Craiova. Sólo si chequea repeticiones
- **Complejidad en tiempo**  $O(r^m)$
- **Complejidad en espacio**  $O(r^m)$
- **Optimalidad** No



# Estrategias Heurísticas

- A Optima (A\*)

- Utiliza en su algoritmo una **función de evaluación**: estimación del costo necesario para alcanzar un estado objetivo por el camino que se ha seguido para generar el nodo actual

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

**f(n)** representa el costo estimado de la solución más económica a través del nodo **n**



# Estrategias Heurísticas

- A Optima ( $A^*$ )

$A^*$  es una estrategia completa y óptima **siempre que la función heurística  $h(n)$  sea admisible**; esto es, nunca sobre-estime el costo para alcanzar el objetivo.

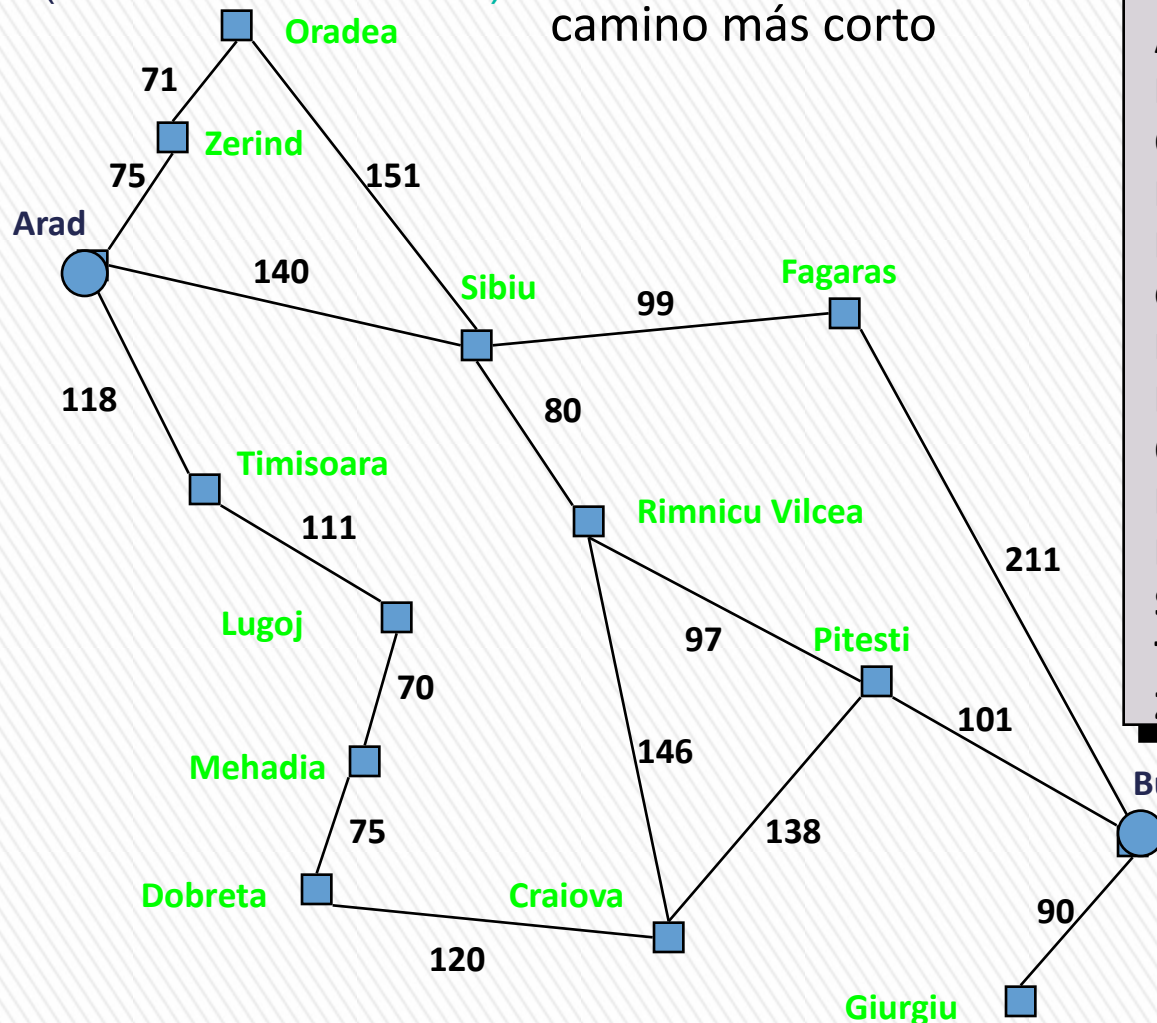
Es decir,... sea **optimista** !!!!





# Costos de recorridos en Rumania en Kms

Mapa de Rumania  
(Dist. entre Ciudades)



**Objetivo:** Ir de Arad a Bucharest utilizando el camino más corto

Dist. en línea recta  
a Bucharest (kms.)

Arad	366
Bucharest	0
Craiova	160
Dobreta	242
Fagaras	178
Giurgiu	77
Lugoj	244
Mehadia	241
Oradea	380
Pitesti	98
Rimnicu Vilcea	193
Sibiu	253
Timisoara	329
Zerind	374



# Estrategias Heurísticas

- A Optima (A\*)

Función de evaluación:  
 $f(n) = h(n) + g(n)$

Arad  
366 + 0  
= 366  
**Paso 1**

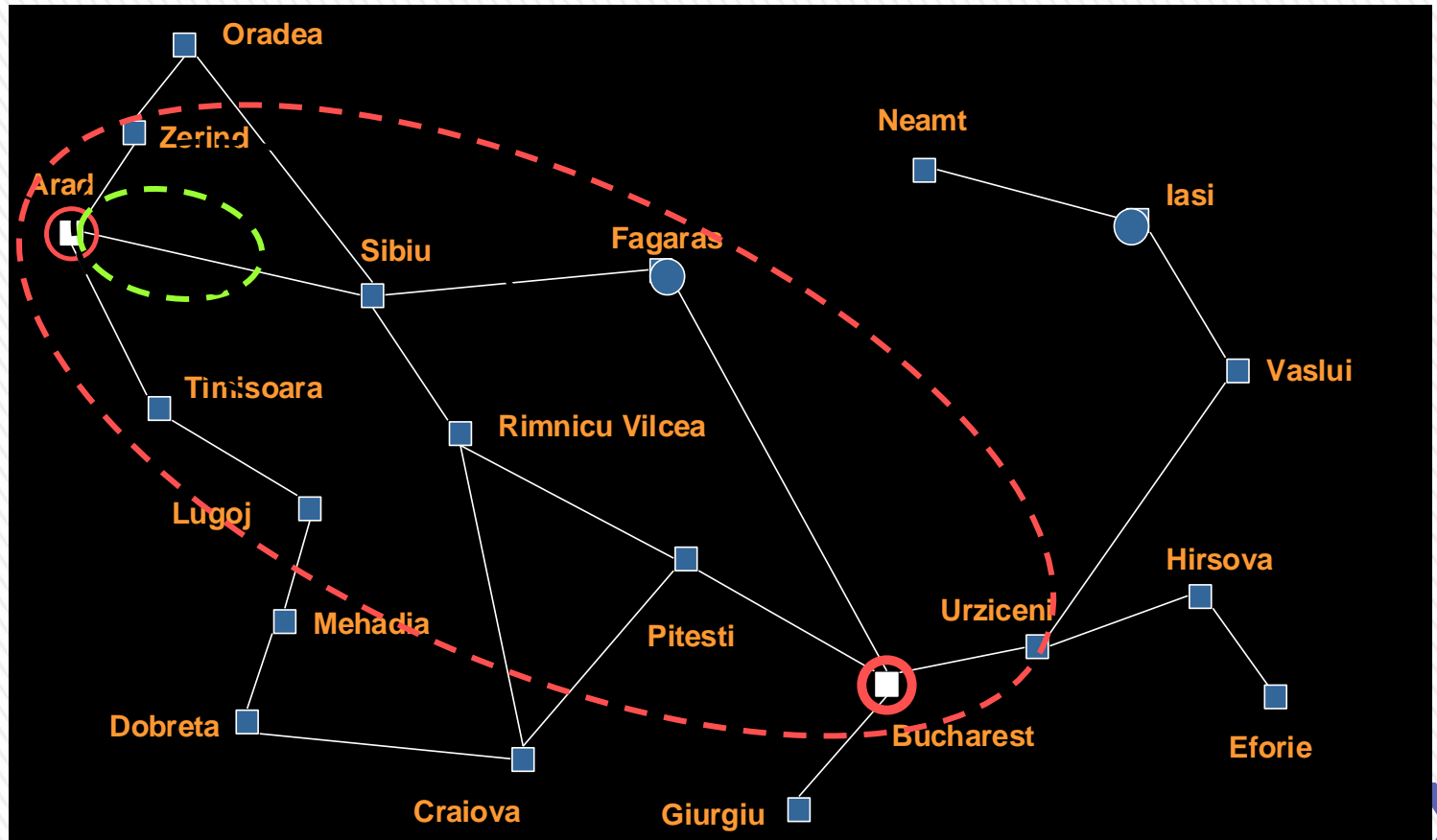
Arad  
Sibiu 253 + 140 = 393  
Timisoara 329 + 118 = 447  
Zerind 374 + 75 = 449  
**Paso 2**

Sibiu  
Arad 366 + 280 = 616  
Fagaras 178 + 239 = 417  
Oradea 380 + 146 = 526  
Timisoara 329 + 118 = 447  
Zerind 374 + 75 = 449  
Rimnicu 193 + 220 = 413  
**Paso 3**

Arad  
Sibiu 366 + 280 = 616  
Fagaras 178 + 239 = 417  
Oradea 380 + 146 = 526  
Timisoara 329 + 118 = 447  
Zerind 374 + 75 = 449  
Rimnicu  
Craiova 160 + 366 = 526  
Pitesti 98 + 317 = 415  
Sibiu 253 + 300 = 553  
**Paso 4**

# Estrategias Heurísticas

- A Optima ( $A^*$ )



# Estrategias Heurísticas

- A Optima ( $A^*$ )

- Completitud    Sí, en grafos localmente finitos
- Complejidad en tiempo    expon. en error en  $h(n)$
- Complejidad en espacio    guarda todos los nodos en memoria
- Optimalidad    Sí



# Heurísticas

## • Satisfacción de Restricciones

- Cripto-aritmética: Asignar valores (0 - 9) a la letras para que la operación dada sea correcta
- 8 Reinas : Colocar 8 reinas en un tablero de ajedrez (8 x 8) de modo que ninguna esté en la misma fila, columna o diagonal que otra
- Coloriz. de mapas: Colorear mapa con  $n$  colores, tal que países vecinos no estén de igual color



# Heurísticas

## • Satisfacción de Restricciones

### Qué tienen en común estos problemas?

- Un conjunto de variables toman valor de un dominio dado
- Un conjunto de restricciones especifican las propiedades del problema
- Los estados están definidos por la asignación de valores de un conjunto de variables
- **Estado meta:** un conjunto de restricciones que estos valores deben cumplir





# Heurísticas

- Satisfacción de Restricciones

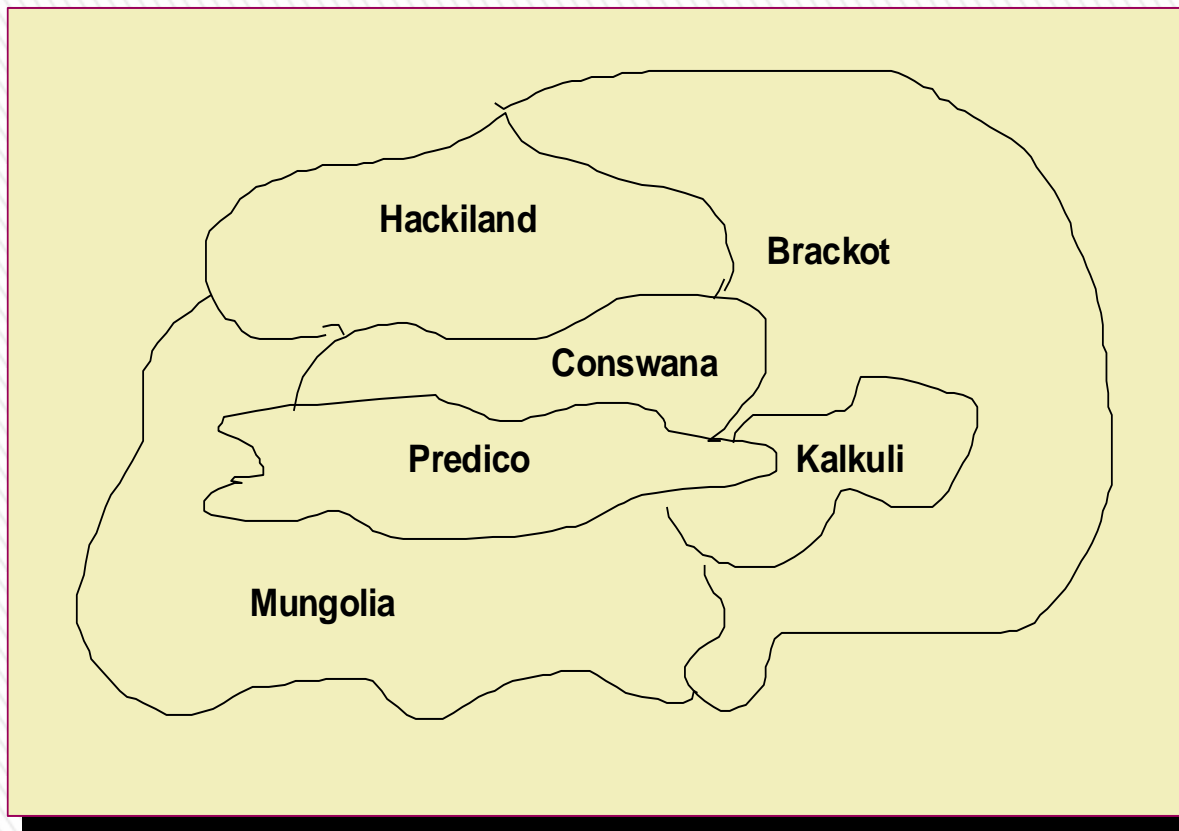
Cuáles son buenas heurísticas para este tipo de problemas?

**“Las que permiten elegir una variable para ser instanciada y escoger un valor para esa variable”**



## EJEMPLO:

### Problema de Colorización de Mapa



Colores

Azul

Rojo

Rosado

Verde



# Heurísticas

## • Satisfacción de Restricciones

- » **Variable más restringida:** en cada paso de la búsqueda, se asigna valor a la variable con menor número posibles valores.
- » **Variable más restrictiva:** asigna un valor a una variable que está involucrada en el mayor número de restricciones de otras variables aún no asignadas.
- » **Valor menos restrictivo:** escoge un valor que descarte el menor número de valores en las variables conectadas por restricciones a la variable actual.



## Búsqueda de Gradiente

- » Considera todos los estados posibles a partir del estado actual y elige el mejor de ellos como nuevo estado.
- » Dependiendo del objetivo de la búsqueda se denomina:
  - > Descenso de gradiente      mínimo
  - > Ascenso de colina      máximo



## Búsqueda de Gradiente

### ★ Evaluar el edo. inicial.

- ★ Si es el edo. Meta retornar EXITO y terminar.
- ★ Si no, hacerlo edo. actual.

🕒 Repetir hasta encontrar solución, o hasta que una iteración completa no modifique el estado actual



## Búsqueda de Gradiente

- a Hacer SUCC un posible sucesor del edo. actual
- b ) Hacer, para cada operador aplicado al edo. actual:
  - ★ Aplicar el operador y generar un nuevo estado
  - 🕒 Evaluar el nuevo estado. Si es un edo. meta, retornar EXITO y terminar . Si no, compararlo con SUCC. Si es mejor, asignar a SUCC ese nuevo estado
- c ) Si SUCC es mejor que el edo. actual, asignar SUCC al edo. actual





## Búsqueda de Gradiente

**Puede quedar atrapado en:**

- **Máximos locales:** generalmente aparecen en las cercanías de una solución .
- **Mesetas:** áreas planas del espacio de búsqueda.
- **Crestas:** tipo especial de máximos locales.
- **Soluciones?** Particulares y no seguras!!!

