



# *Inteligencia Artificial*

## Espacio de estados y búsqueda

## 3. Espacio de estados y búsqueda

### 3.1 Métodos de búsqueda no informados (Uninformed Search Methods):

- Búsqueda en anchura (Breadth-First Search)
- Búsqueda en profundidad (Depth-First Search)
- British Museum
- Búsqueda de coste uniforme (Uniform-Cost Search)

### 3.2 Métodos de búsqueda informados (Informed Search Methods):

- Heurísticos
- Búsqueda voraz (Greedy Search)
- Búsqueda A\* (A\* or A star search)
- Grafos AND / OR

### 3.3 Búsqueda adversarial

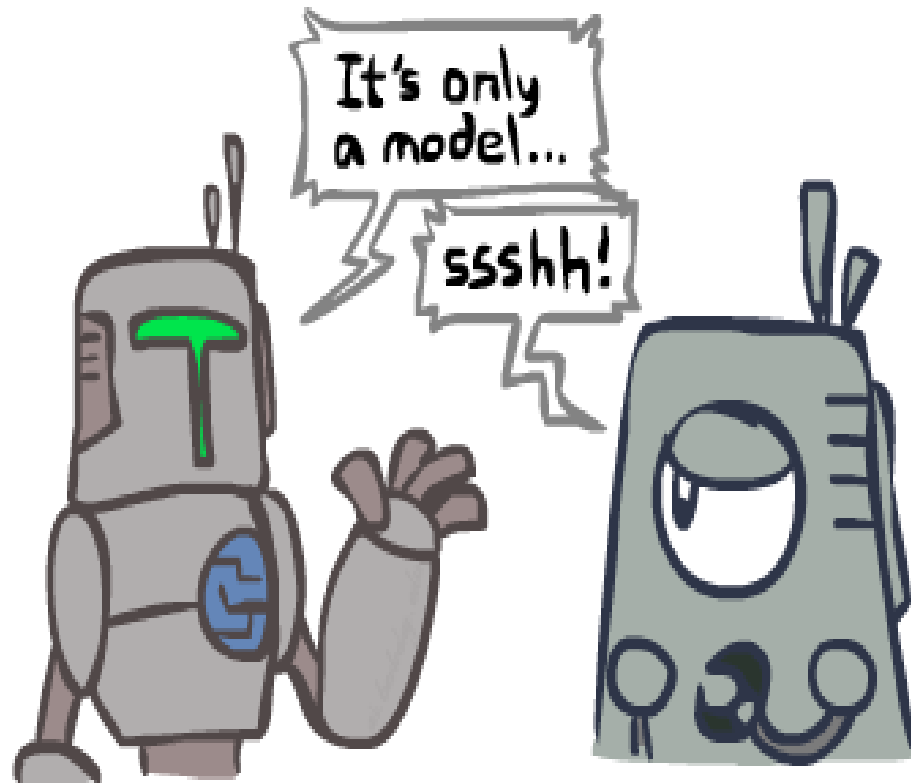
- Minimax
- Alfa-beta
- Expectimax

# Búsqueda informada (Informed Search)



# Búsqueda en modelos

- La búsqueda opera sobre **modelos del mundo**:
  - El agente **no** prueba todos los planes **del mundo real**
  - La planificación se hace en modo “**simulación**”
  - La búsqueda **es tan buena como** nuestros **modelos...**



# Heurísticos de búsqueda

## ➤ Un heurístico es:

- Conjunto de reglas que **nos sirven** para escoger aquellas ramas del espacio de estados que **habitualmente** llevan hacia una solución aceptable del problema.
- Se **obtiene de la experiencia** considerando las características específicas de cada problema.
  - Ejemplo: buscar una oficina de turismo al llegar a una ciudad.
    - *Heurístico?*
      - *tomar la calle que más me acerque a la catedral*

## ➤ Objetivo:

- Restringir la búsqueda

## ➤ Adecuados para espacio de estados amplios

# Heurísticos de búsqueda

## ➤ Características:

- **Algunos algoritmos no garantizan** que se encuentre una solución, **aunque exista**.
- **En caso de encontrar** una solución, **no siempre aseguran** que ésta sea la de longitud mínima o la de coste óptimo (la mejor).



- **En algunas ocasiones** (que, en general, no se podrán determinar a priori), **encontrará una solución** (aceptablemente buena) **en un tiempo razonable**.
  - Pierden completitud (no se analizan todos los estados posibles del mundo)
    - **Aumenta eficiencia**

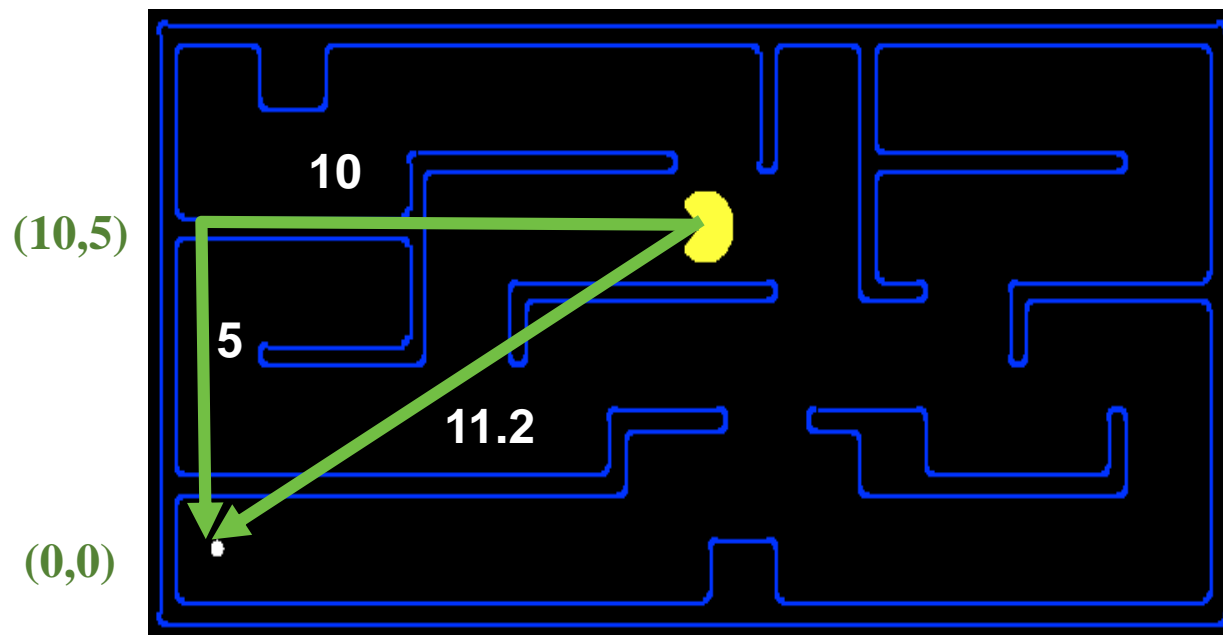
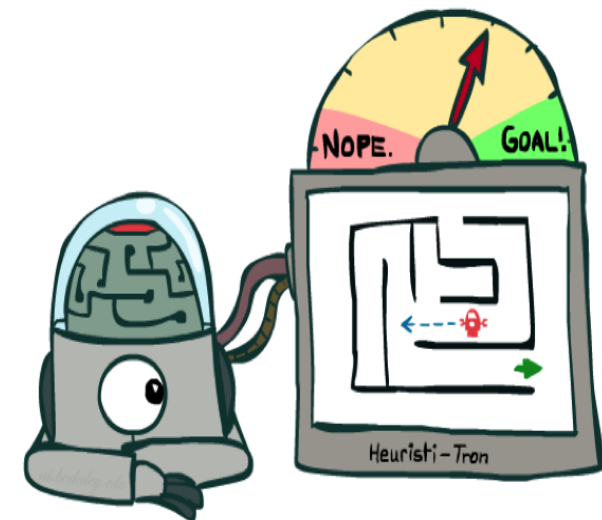
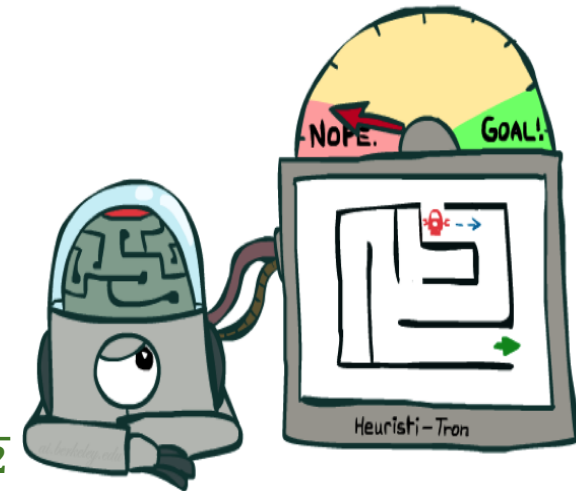
# Heurísticos de búsqueda

➤ Un heurístico es:

- Una función que **estima cómo de cerca se encuentra un estado del objetivo**

– Ejemplos para pathing: *Heurístico?*

- *distancia Euclídea*:  $d_E(P_1, P_2) = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$
- *distancia de Manhattan*:  $d_M(i, j) = |x_i - x_j| + |y_i - y_j|$

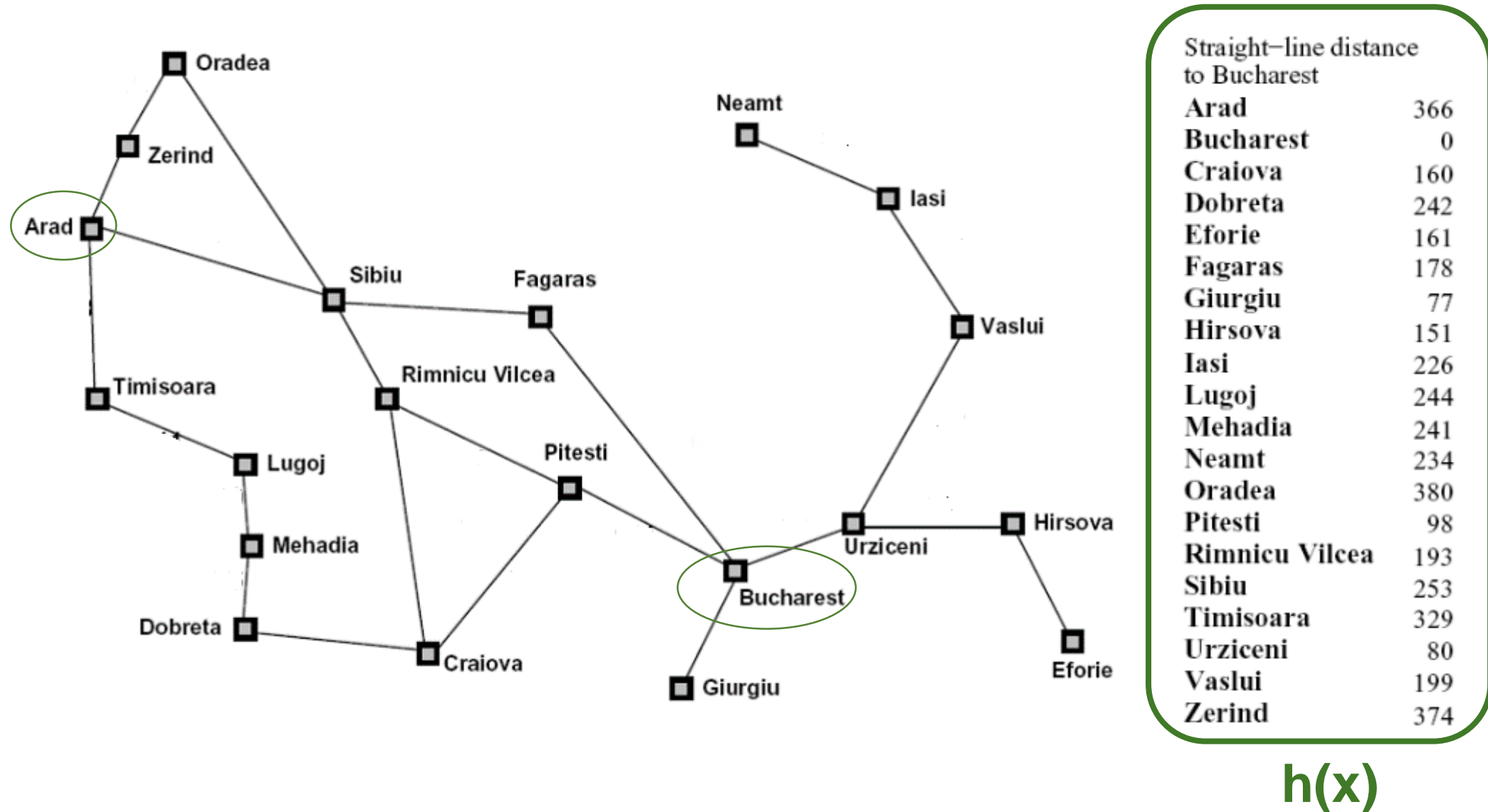


# Búsqueda voraz (Greedy Search)





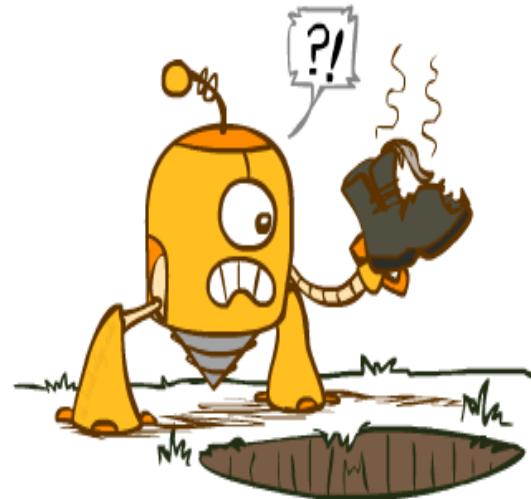
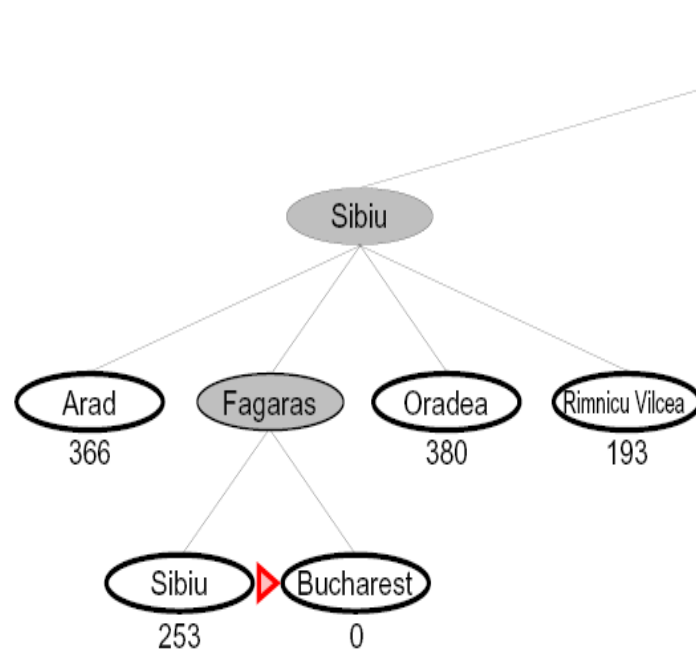
# Ejemplo: función Heurística



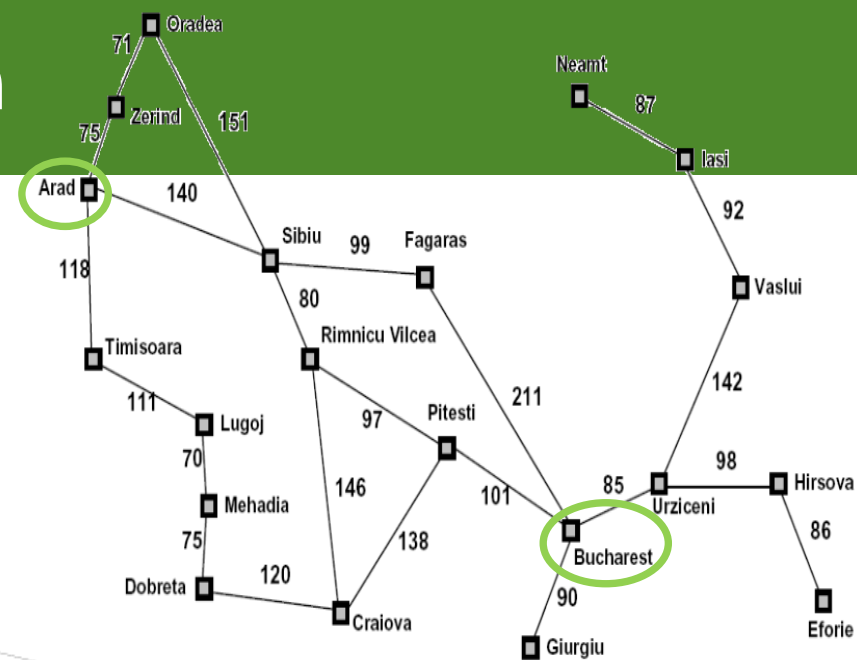
# Greedy Search

➤ Expandir el nodo que parece más cercano...

- $f(n) = h(n)$



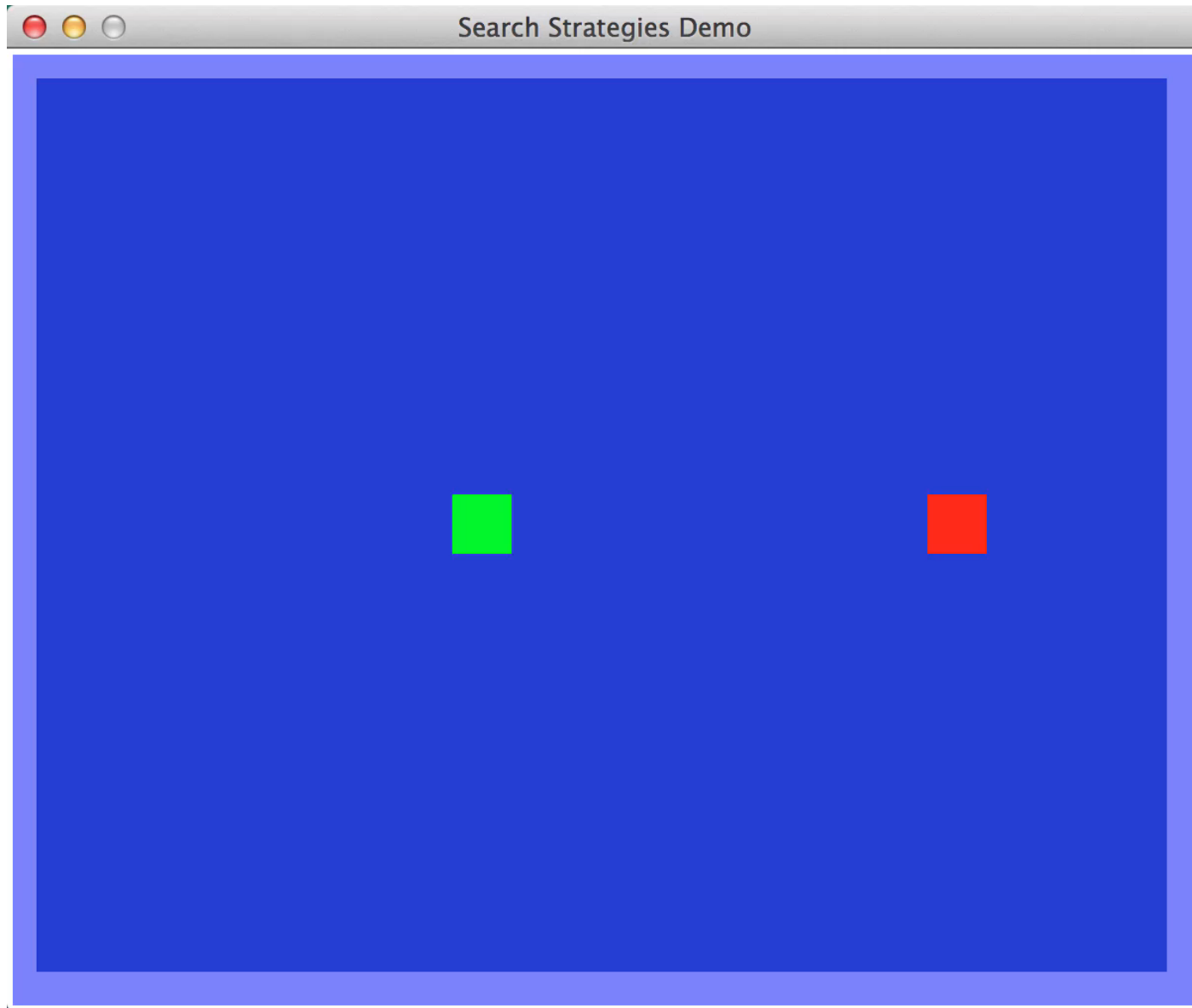
➤ Qué puede fallar?



Straight-line distance  
to Bucharest

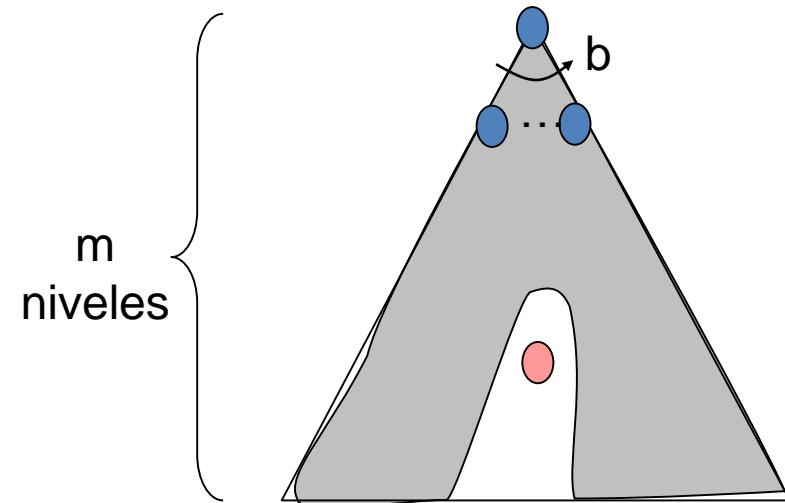
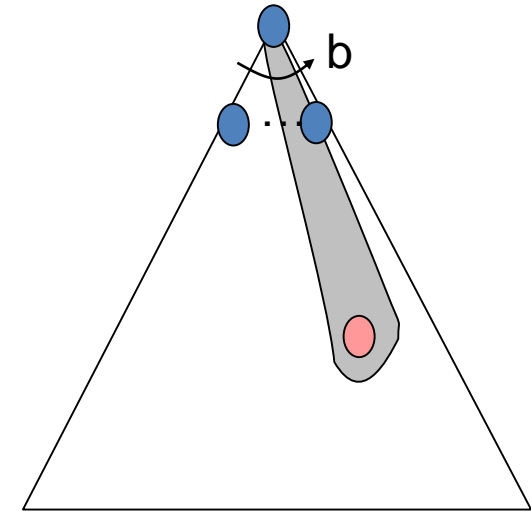
Arad	366
Bucharest	0
Craiova	160
Dobreta	242
Eforie	161
Fagaras	178
Giurgiu	77
Hirsova	151
Iasi	226
Lugoj	244
Mehadia	241
Neamt	234
Oradea	380
Pitesti	98
Rimnicu Vilcea	193
Sibiu	253
Timisoara	329
Urziceni	80
Vaslui	199
Zerind	374

# Video of Demo Greedy (Empty)

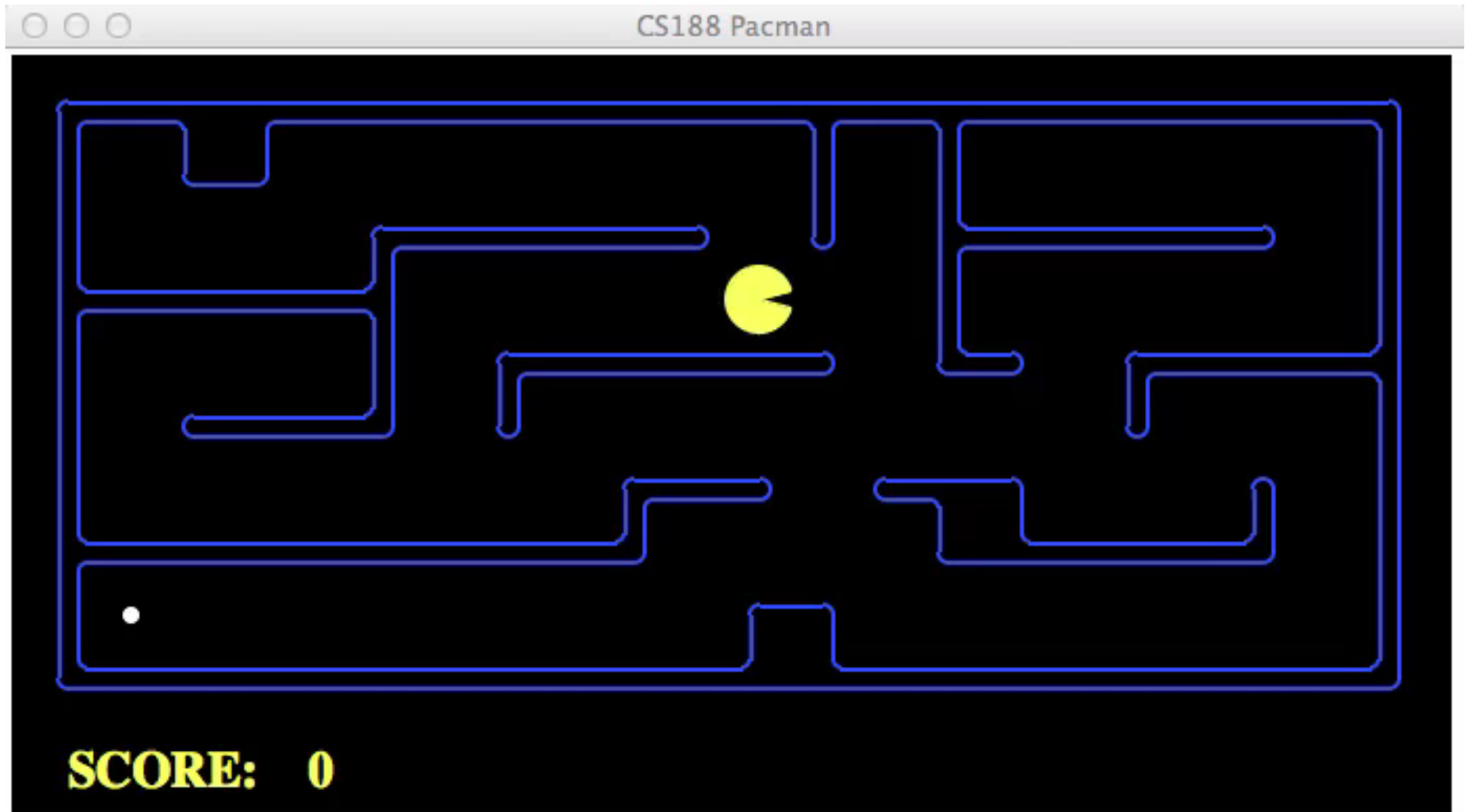


# Greedy Search

- **Estrategia:** **expandir el nodo** que pensamos **que está más cerca** de un estado objetivo
  - **Heurístico:** estimación de la distancia al objetivo más cercano desde cada estado
- **En el mejor de los casos:**
  - Nos lleva directamente a un objetivo (**quizás no el mejor**)
- **Caso peor:**
  - como DFS mal guiado =  $O(b^m)$
  - guarda todos los nodos

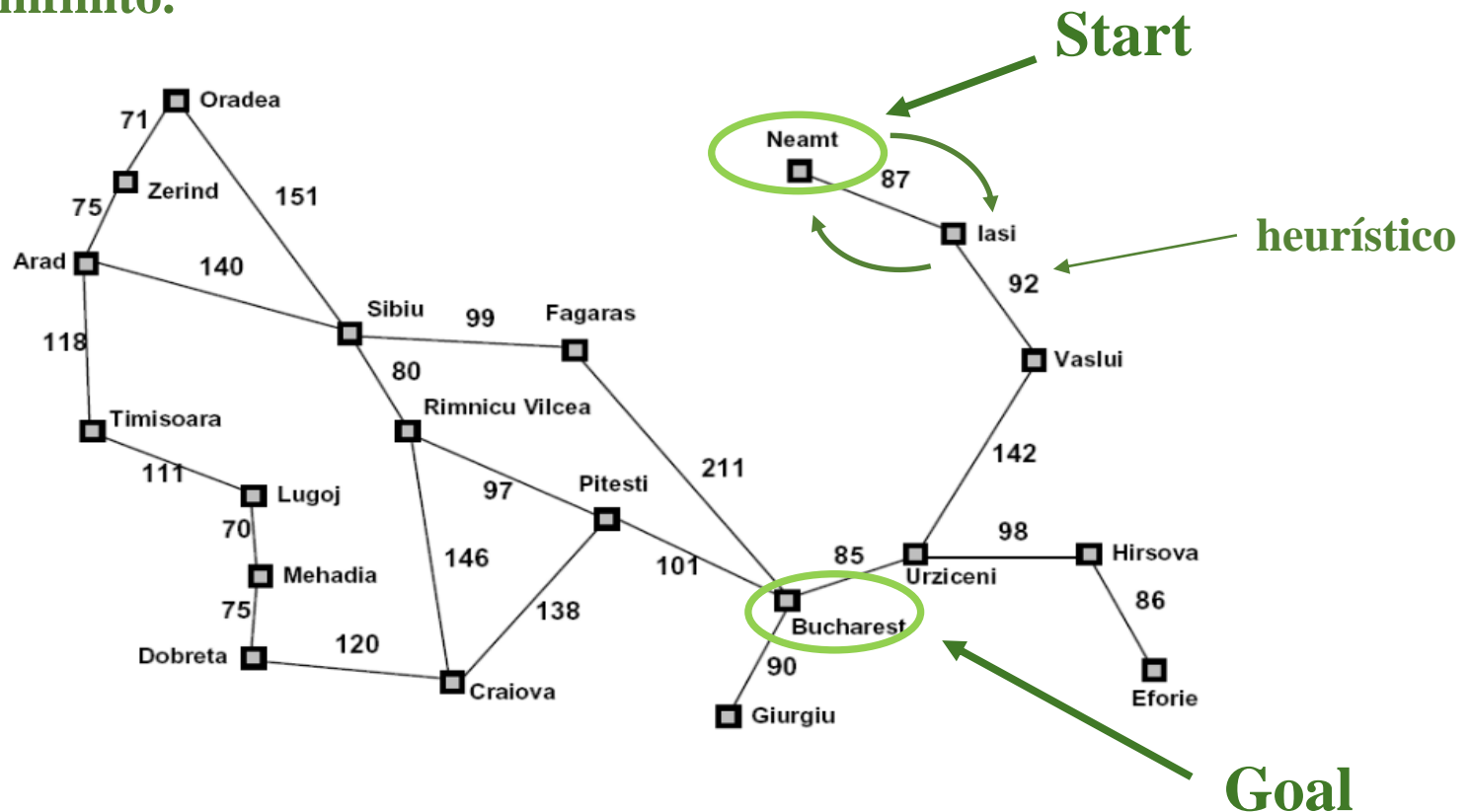


# Video of Demo Contours Greedy (Pacman Small Maze)



# Greedy Search

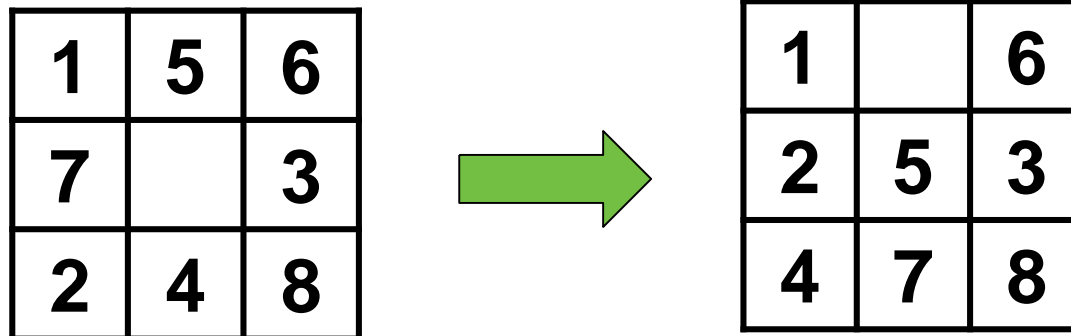
- Óptimo?: **NO**
- Completo?
  - **No** en la búsqueda en **árbol**. Incluso con un espacio de estados finito, **puede entrar en un bucle infinito**.



# Problema puzzle

## ➤ Heurístico a utilizar?

- Número de fichas que hay desordenadas en cada estado ➡ **H=4**



- Suma de la distancia a la solución de cada número (Distancia de Manhattan):
  - $5 (1) + 7 (2) + 2 (1) + 4 (1) \rightarrow \mathbf{H=5}$

# Métodos que realizan una búsqueda heurística

- Algoritmos que *quizás* encuentran una *solución cualquiera*:
  - Memoria limitada: **eliminan ramas del espacio del estado** que pueden llevar a la solución:
    - **Hill Climbing**
    - **Best first**
    - **Beam search**



# Hill climbing - versión 1

➤ Algoritmo: No garantiza que se encuentre una solución, aunque exista.

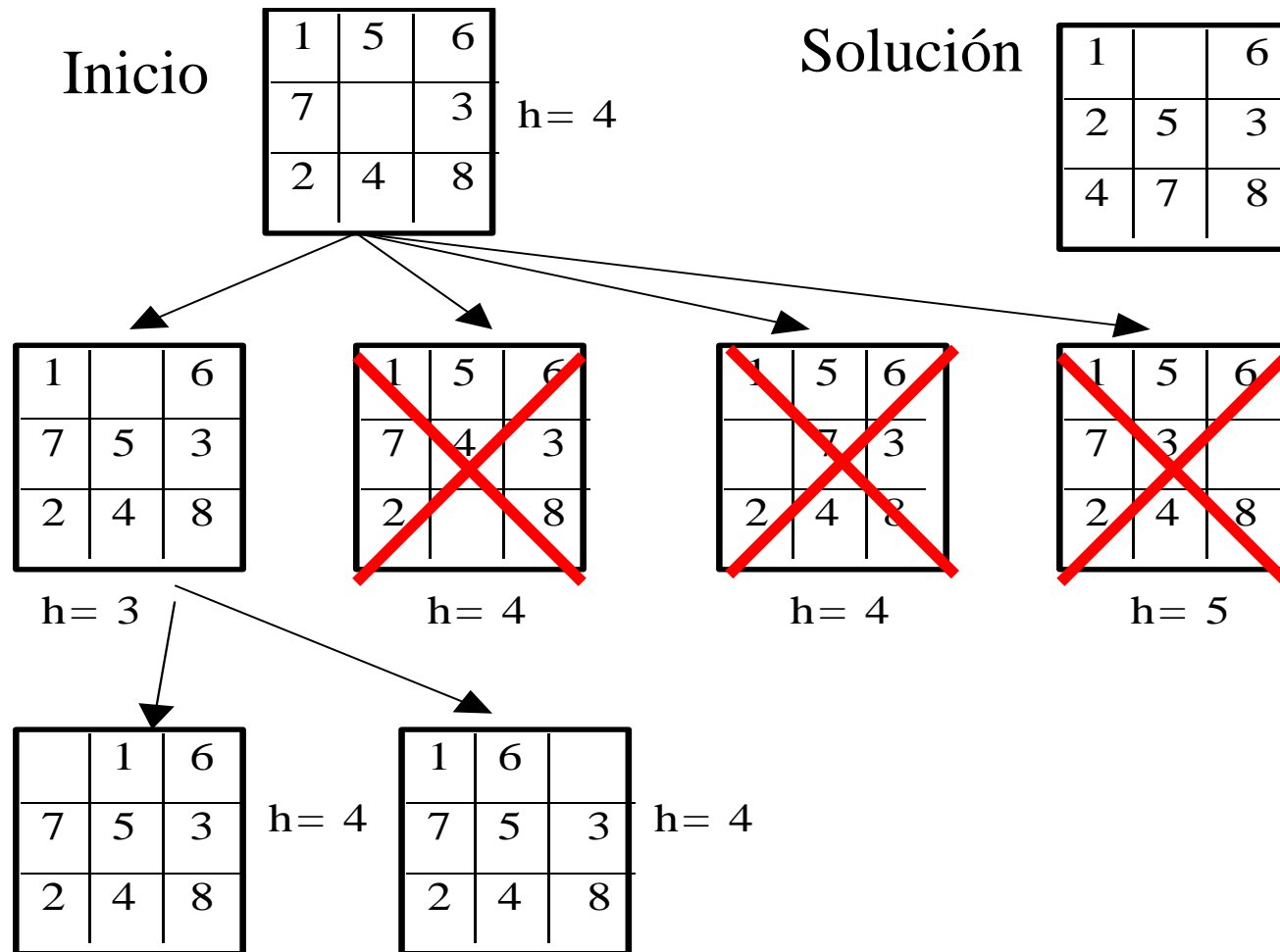
1- Construir una lista con el nodo raíz como único elemento.

2- Hasta que la lista esté vacía o el primer (y único) elemento de la lista sea el elemento objetivo:

- 2.1- Eliminar el primer elemento de la lista (nodo padre) y ordenar sus hijos (si los hubiera) estimando cuál de ellos está más cerca de la solución.
- 2.2- Si el primer nodo hijo,  $\mathbf{x}_1$ , de los ordenados está **MÁS CERCA** de la solución de lo que estaba el nodo padre, es decir  $\mathbf{h}(\text{padre}) > \mathbf{h}(\mathbf{x}_1)$  añadirlo al principio de la lista (el único en la lista), si no, acabar.

3- Si se ha encontrado un nodo objetivo, anunciar éxito, si no, fallo.

# Árbol de búsqueda. Ejemplo HC versión 1



**Fallo:** en este caso el algoritmo fracasa  $\rightarrow 3 < 4$

# Hill climbing - versión 1

## ➤ Problema:

- Pequeñas colinas

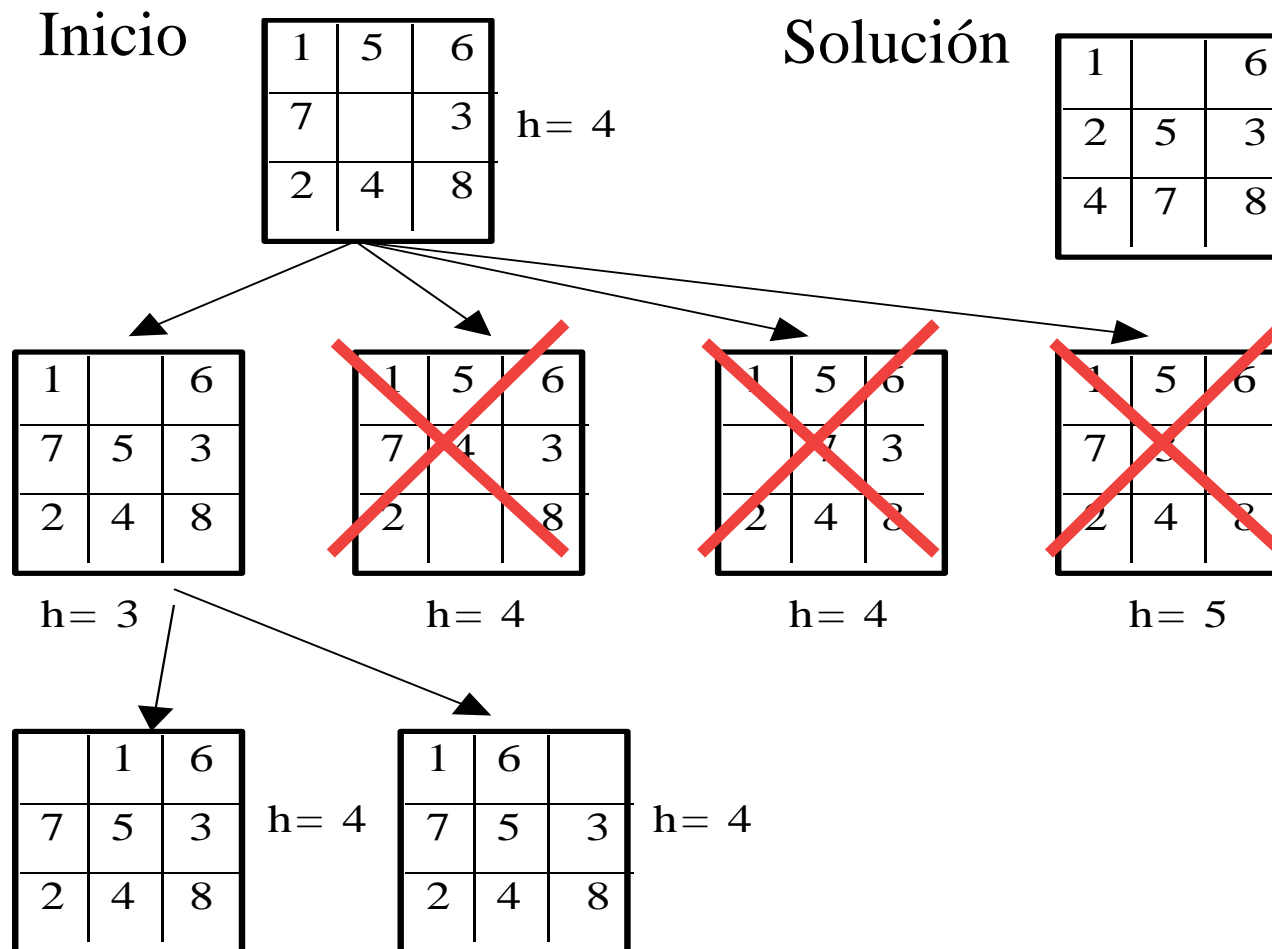


# Hill climbing - versión 2

➤ Algoritmo: No garantiza que se encuentre una solución, aunque exista.

- 1- Construir una lista con el nodo raíz como único elemento.
- 2- Hasta que la lista esté vacía o el primer (y único) elemento de la lista sea el elemento objetivo
  - 2.1- Eliminar el primer elemento de la lista (nodo padre) y ordenar sus hijos (si los hubiera) estimando cuál de ellos está más cerca de la solución
  - 2.2- Añadir el primer hijo de los ordenados,  $x_1$ , al principio de la lista (el único de la lista). *Elimina la restricción de mejorar el heurístico*
- 3- Si se ha encontrado un nodo objetivo, anunciar éxito, si no, fallo.

# Árbol de búsqueda. Ejemplo HC versión 2

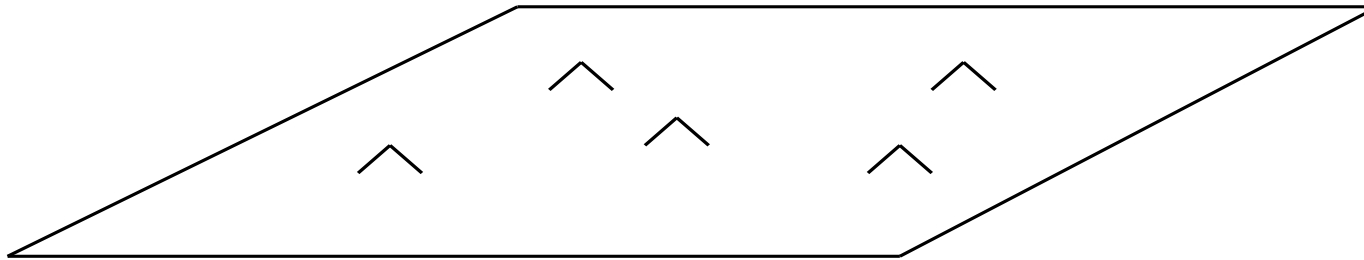


En este punto el algoritmo **NO** fracasa y seguiría tratando de buscar una solución

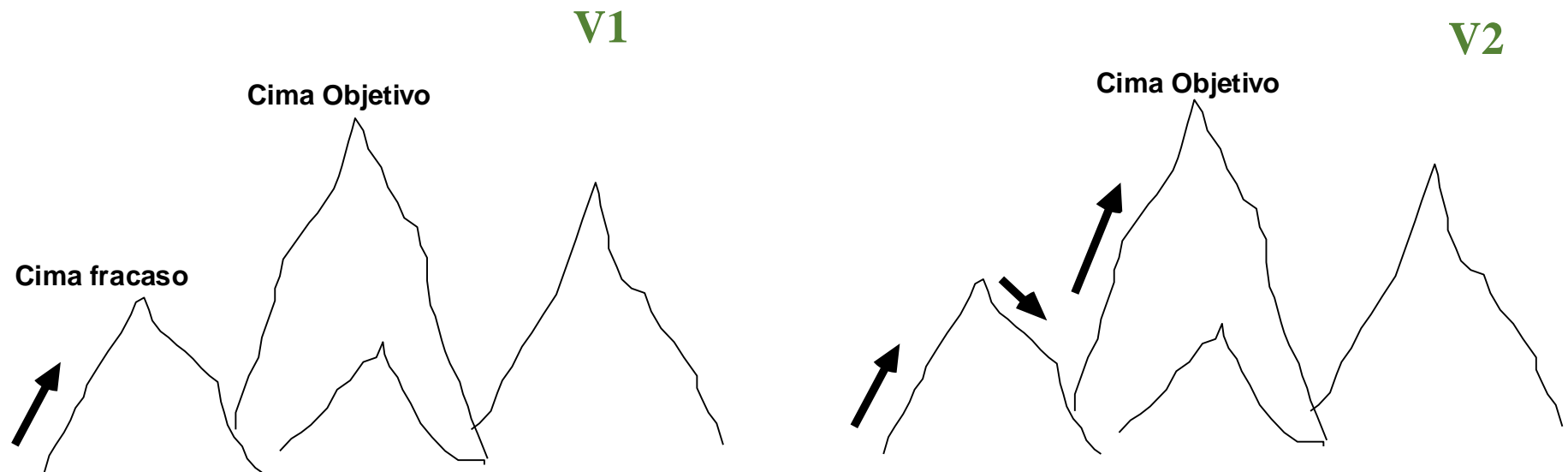
# Hill climbing - versión 2

## ➤ Problema:

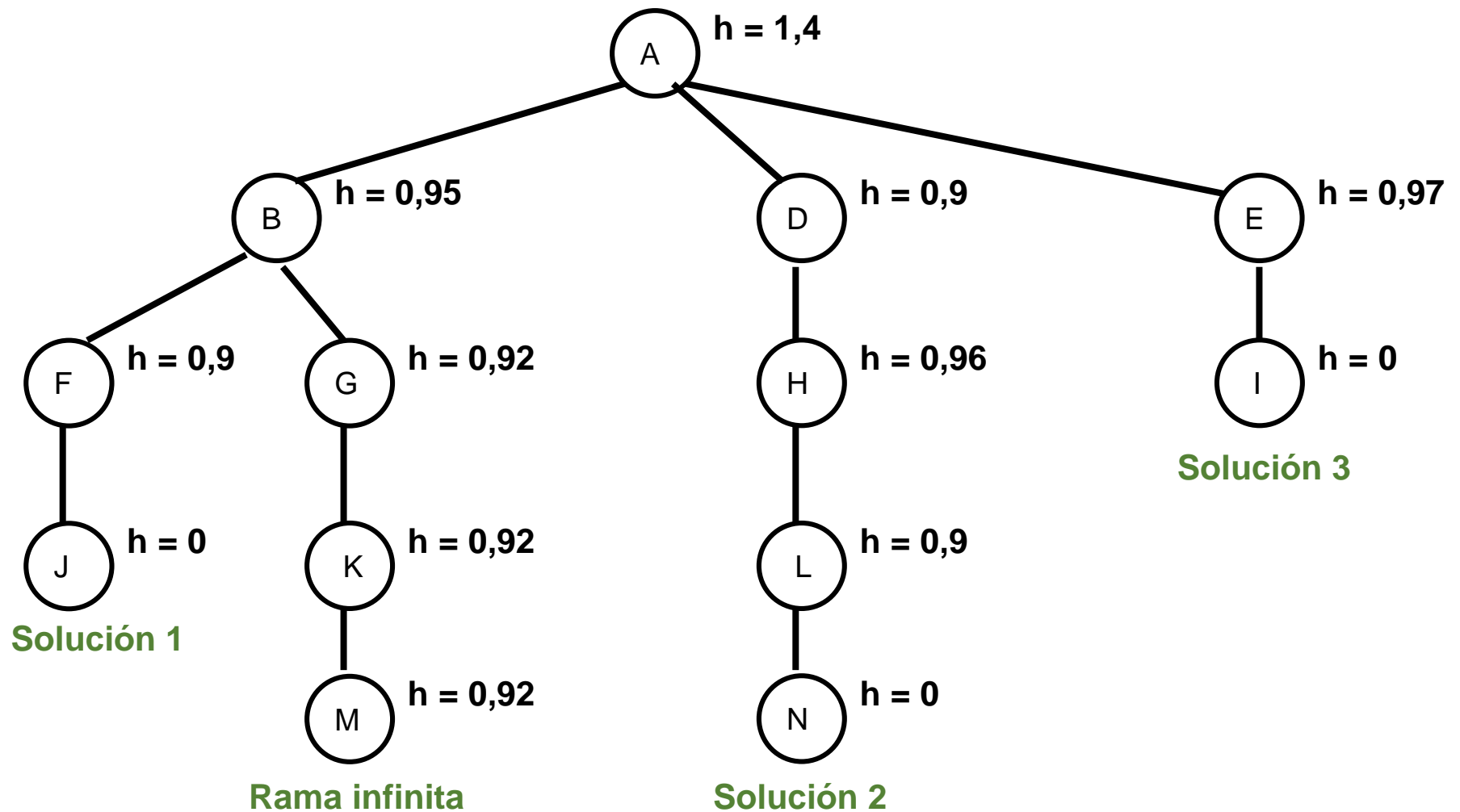
- Meseta → **todos los hijos mismo valor heurístico!**



# Comparativa Hill climbing - V1 y V2

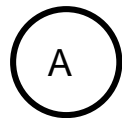


# Hill climbing V1 / V2

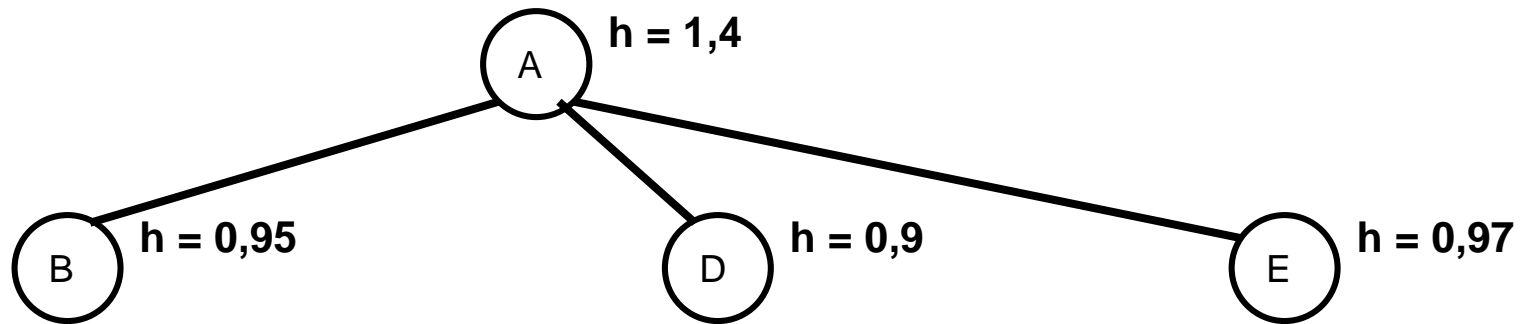




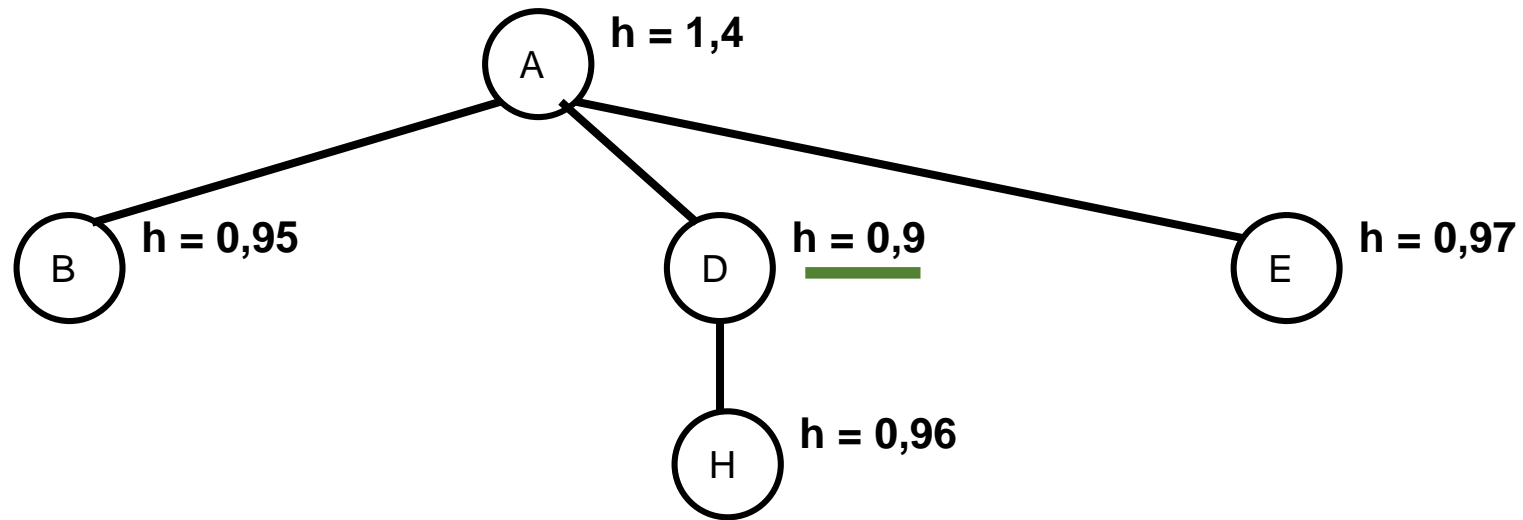
# Hill climbing V1

  $h = 1,4$

# Hill climbing V1

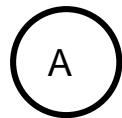


# Hill climbing V1

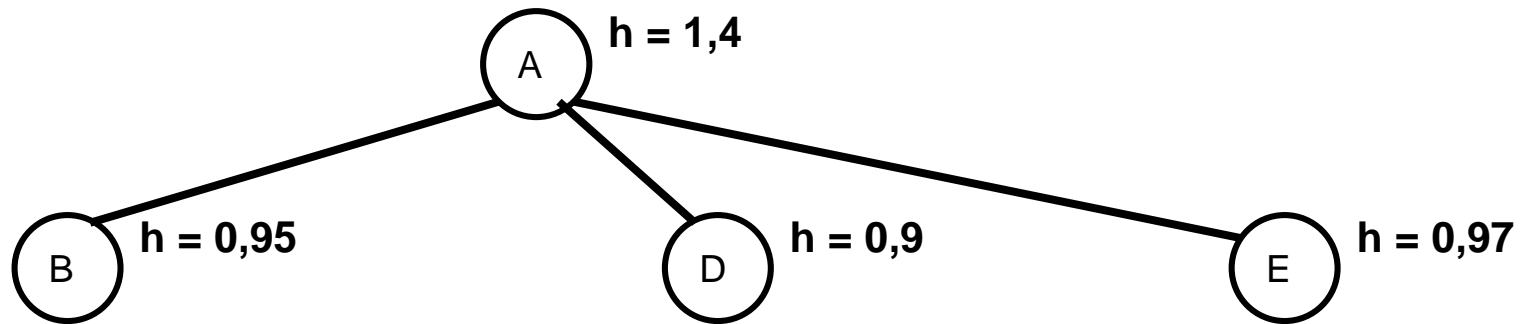


**Fallo:** en este caso el algoritmo fracasa:  $0.9 < 0.96$

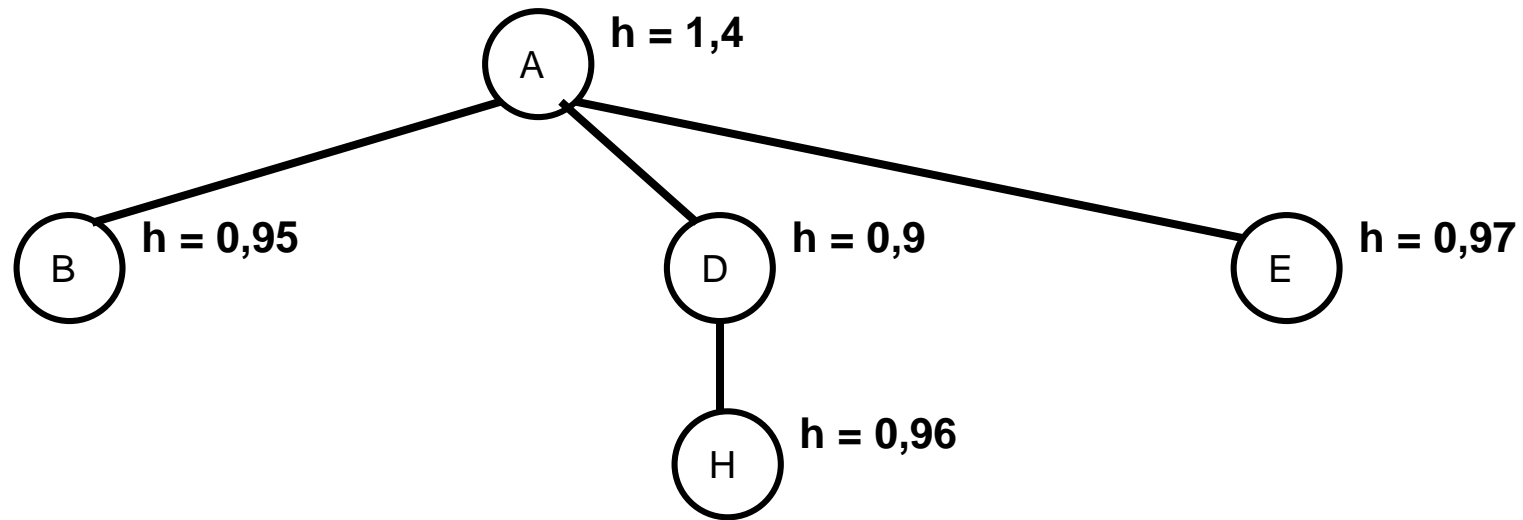
# Hill climbing V2

  $h = 1,4$

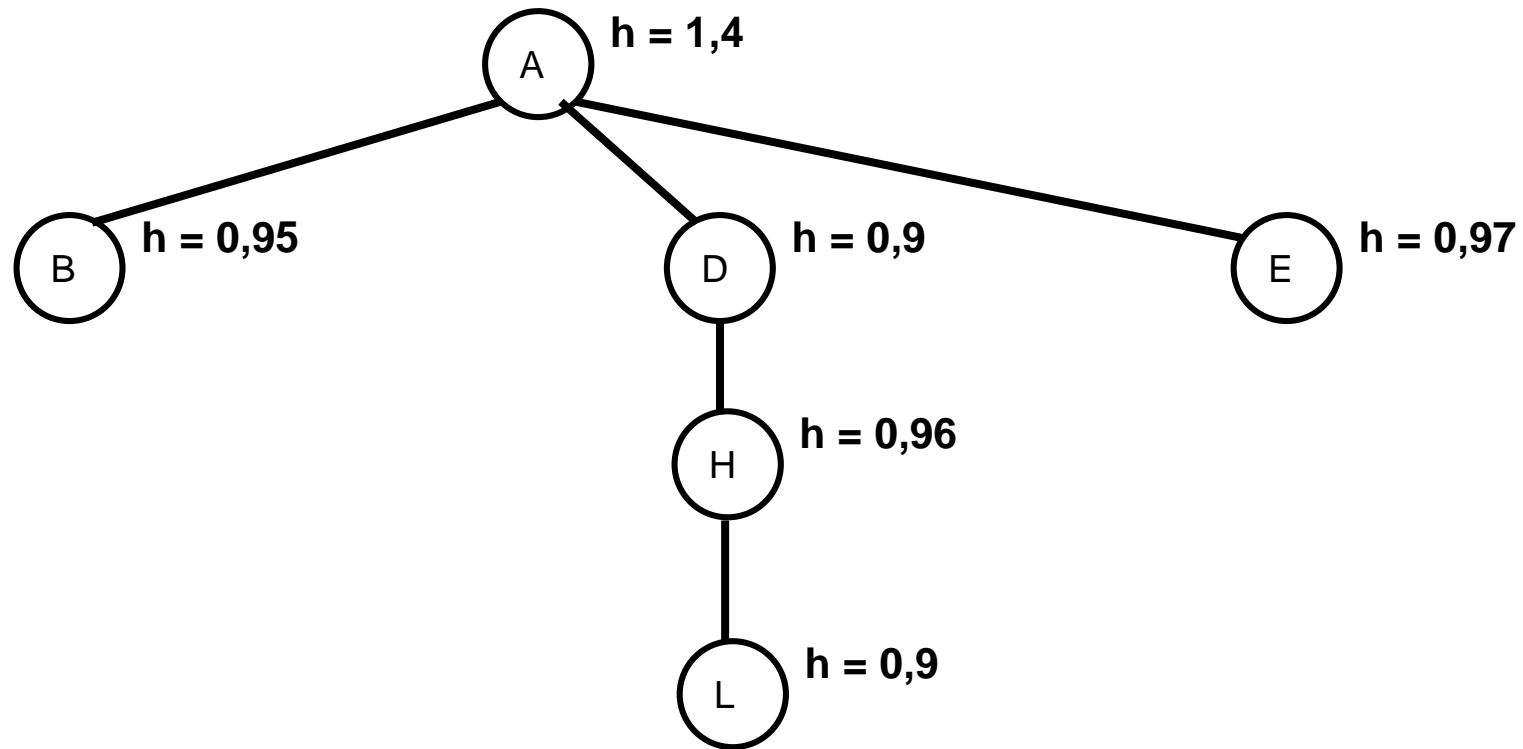
# Hill climbing V2



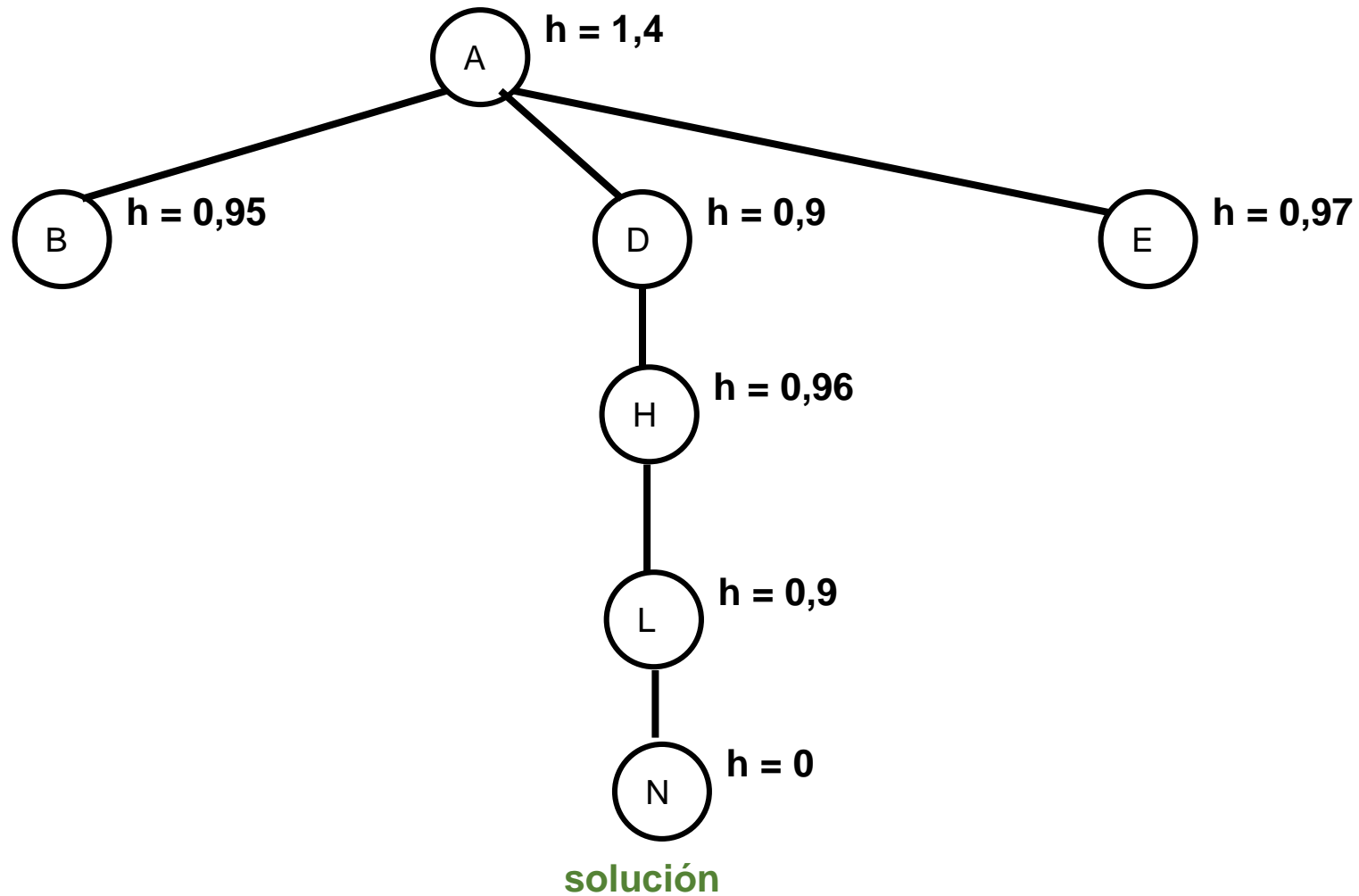
# Hill climbing V2



# Hill climbing V2



# Hill climbing V2





# Conclusiones Hill Climbing

## ➤ Problemas:

- Puede haber puntos en los que el algoritmo se estanque
  - **Máximos locales**: todos los hijos de un estado son peores que él y no es el estado objetivo. Un máximo **local** es un estado mejor que cualquier otro estado vecino, pero peor que otros más lejanos. El algoritmo para sin dar solución en **V1**
  - **Mesetas**: todos los hijos tienen mismo valor heurístico. Una meseta es una región del espacio de estados donde todos los estados tienen el mismo valor heurístico. **El algoritmo para sin dar solución.**
    - Y si seguimos adelante?
      - Si sigue, la heurística no informa ⇒ **búsqueda ciega**

## ➤ Comportamiento de Hill Climbing?

- **Búsqueda en profundidad!**

# Best first (el mejor primero)

## ➤ Algoritmo

- 1- Construir una lista con el nodo raíz como único elemento.
- 2- Hasta que la lista esté vacía o el primer elemento de la lista sea el elemento objetivo:
  - 2.1 - Eliminar el primer elemento de la lista, añadir los hijos de este elemento (si los hubiera) y ordenar la lista estimando lo que falta hasta la solución.
- 3- Si se ha encontrado el nodo objetivo, anunciar éxito, si no, fallo.

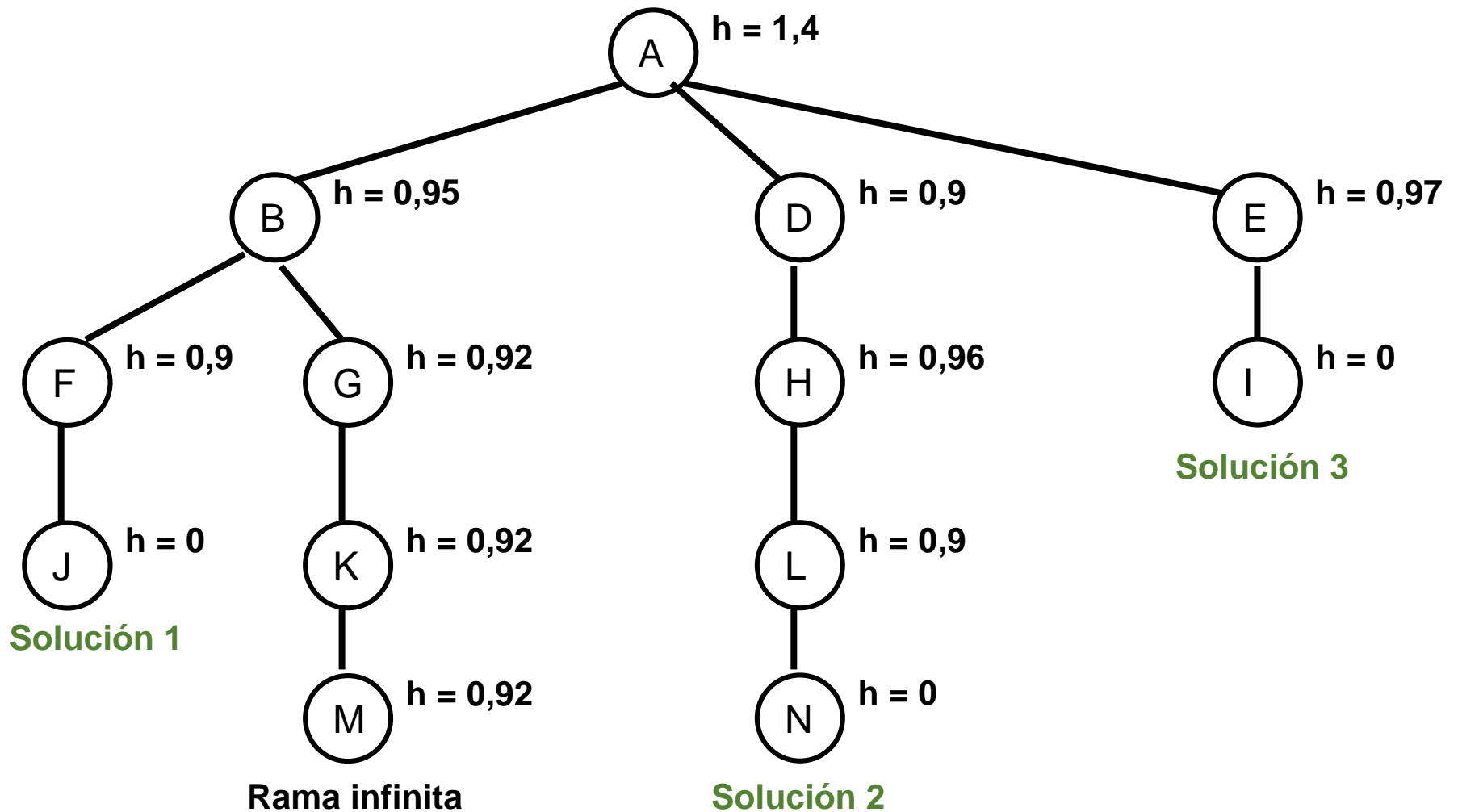
## ➤ Problema

- Problemático para problemas con ramificación alta

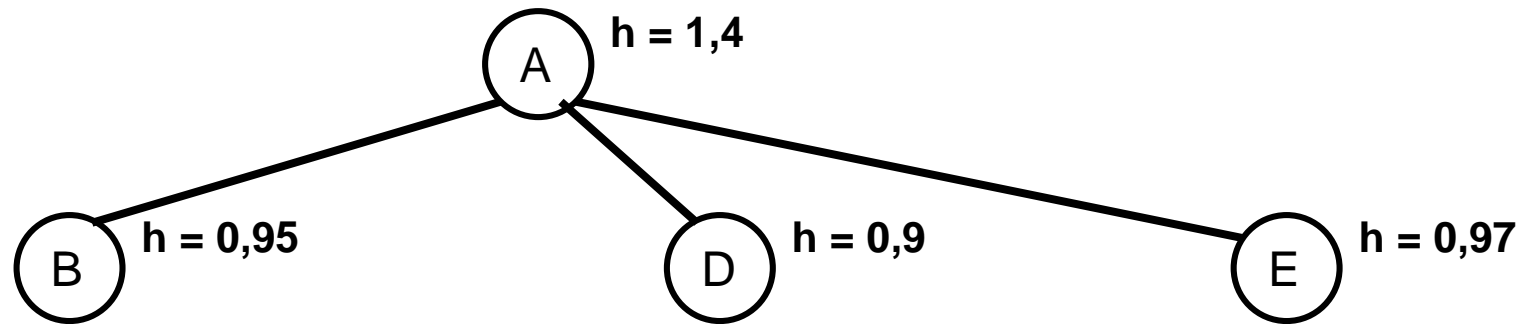
Qué ocurre cuando todos los valores de los heurísticos de los hijos sea iguales?

**Búsqueda en profundidad!**

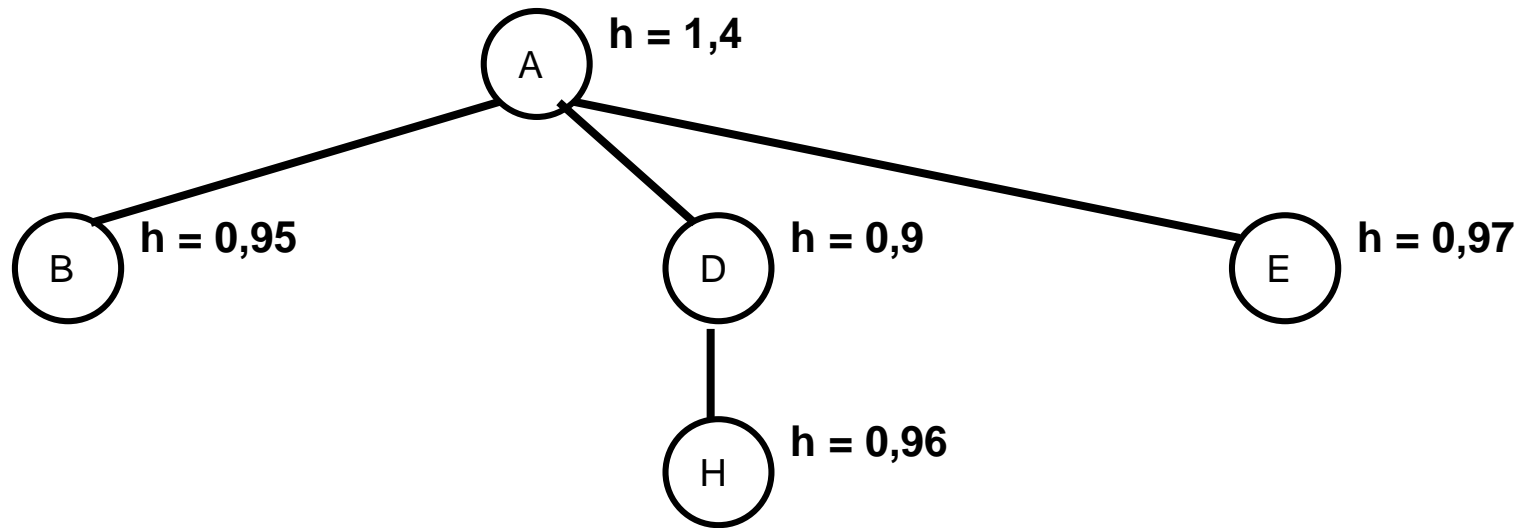
# Best First



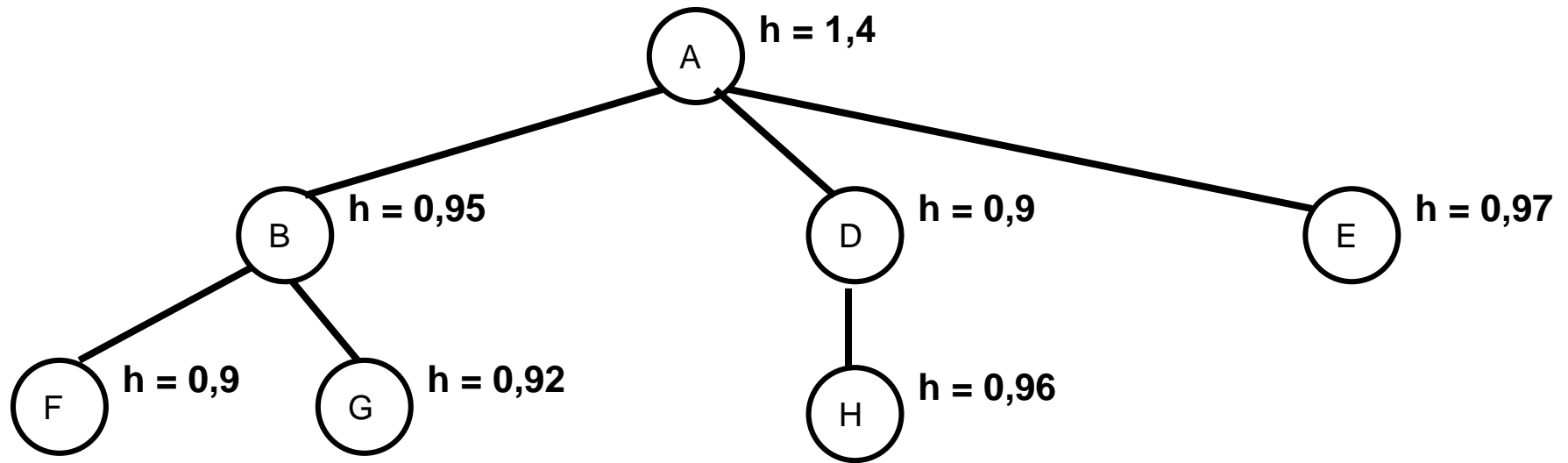
# Best First



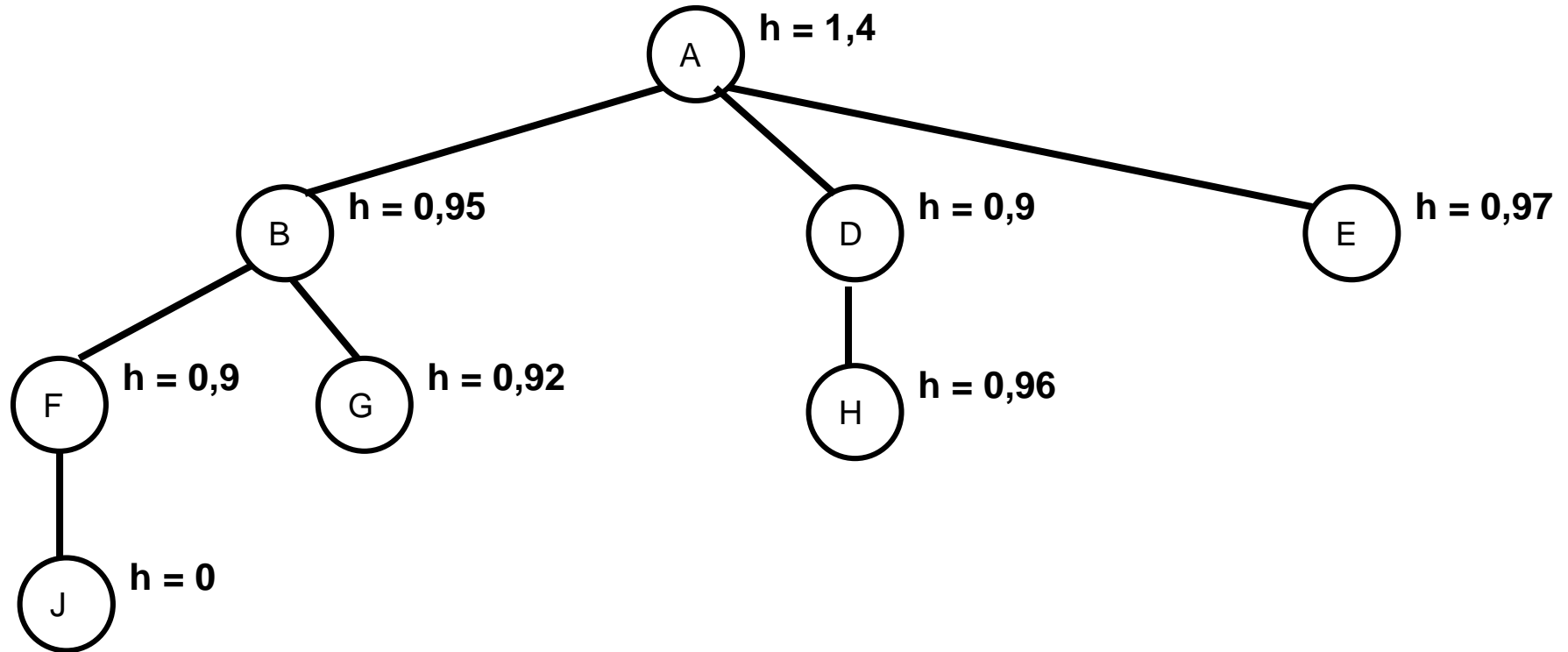
# Best First



# Best First



# Best First



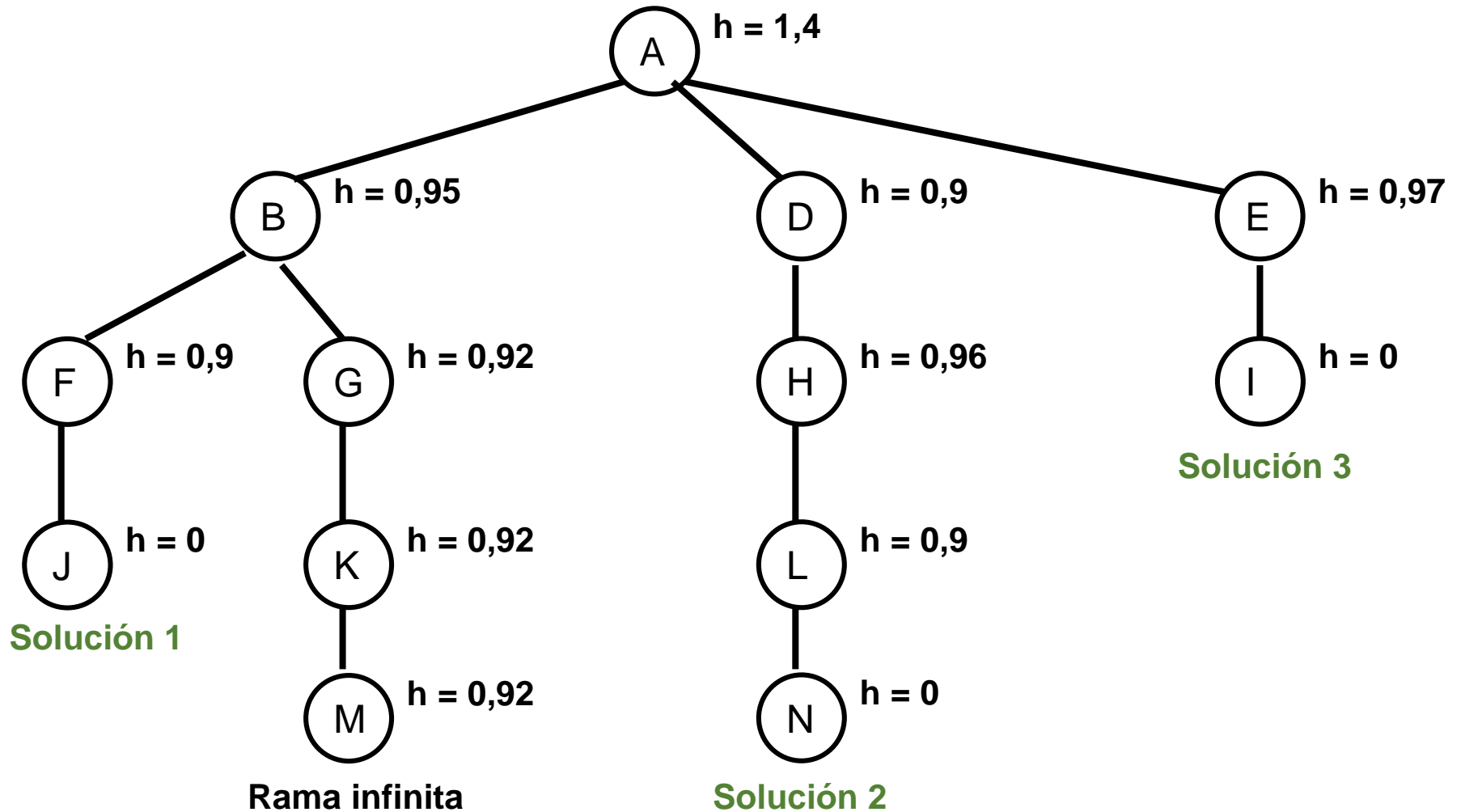
**Solución 1**

# Beam search

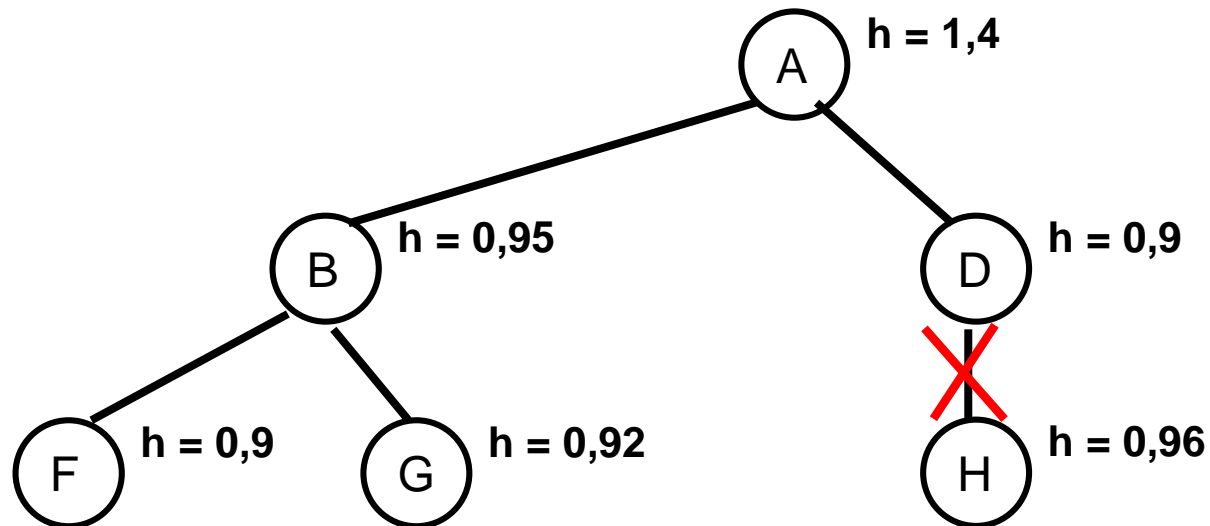
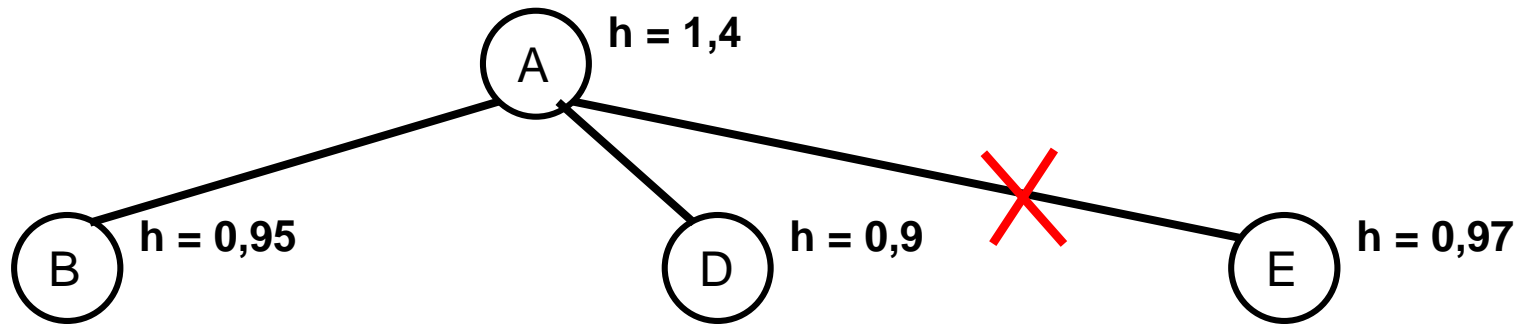
- **Alternativa** entre Hill Climbing y Best-first
- Mantiene las **X** mejores alternativas
- **Algoritmo:**
  - 1-** Determinar X como el n° máximo de nodos por nivel a expandir; Lista\_X = (nodo raíz); Lista\_hijos = ( )
  - 2-** Hasta que Lista\_X esté vacía o Lista\_X contenga un nodo objetivo
    - 2.1-** Si Lista\_X contiene una solución:
      - 2.1.1-** Salir del bucle anunciando éxito para primer nodo solución
    - 2.2-** Si no:
      - 2.2.1-** Eliminar todos los elementos de Lista\_X y añadir sus hijos (**si los hubiera**) a Lista\_hijos, **ordenados crecientemente según h**
    - 2.3-** Meter en Lista\_X los X primeros nodos de Lista\_hijos
  - 3-** Si Lista\_X está vacía, **anunciar fracaso**



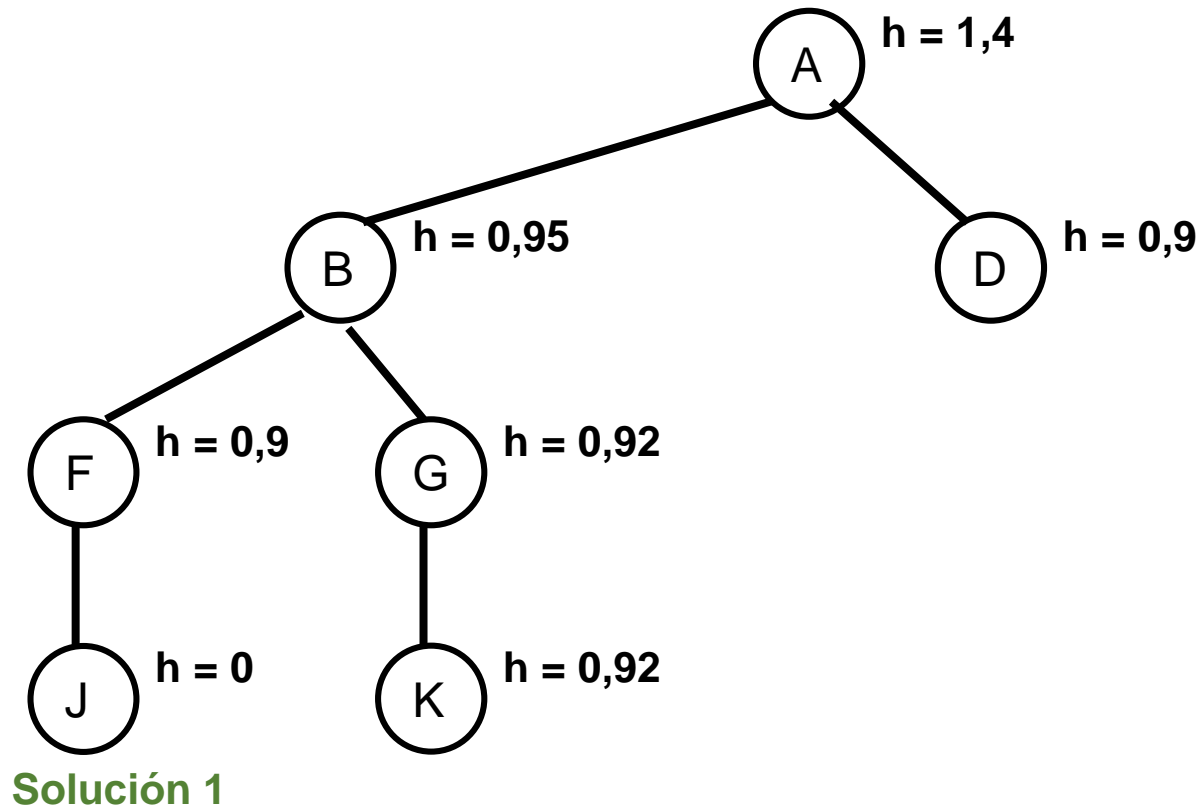
# Beam search (X=2)



# Beam search (X=2)



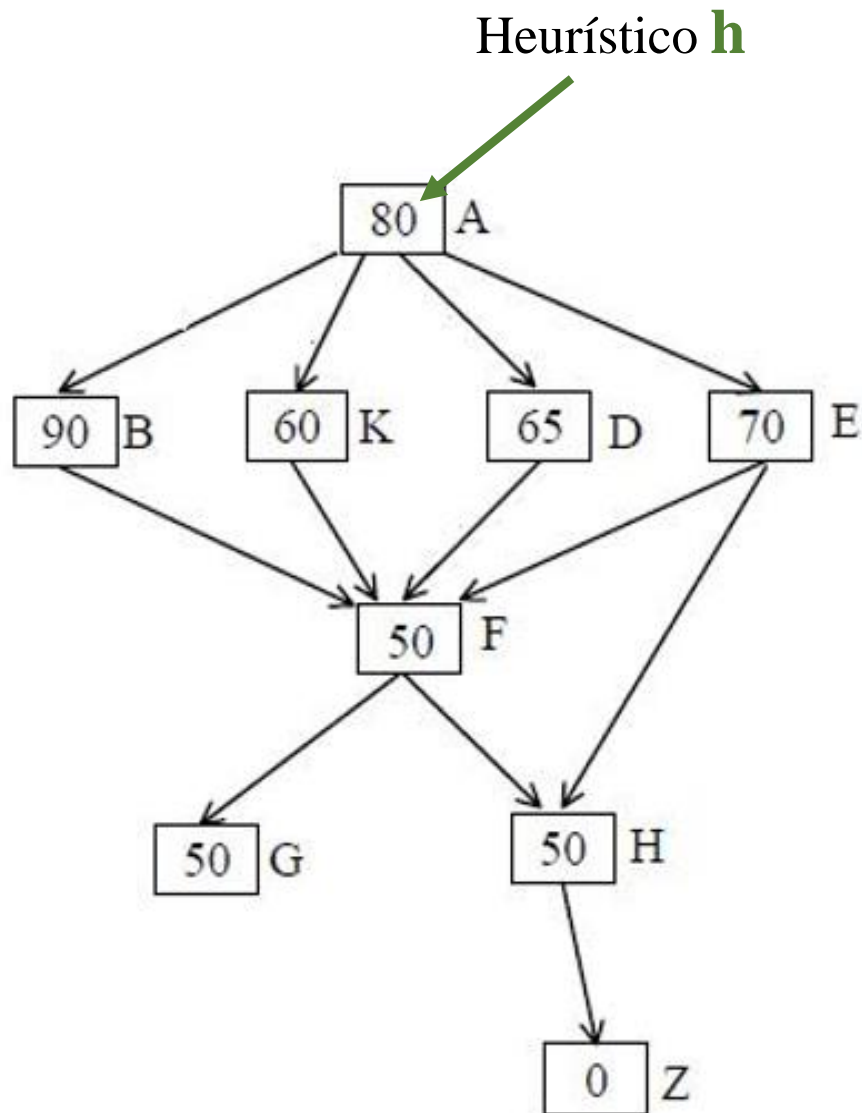
# Beam search (X=2)



# Beam search

- Qué ocurre cuando **X es 1**?
  - tenemos la técnica **Hill Climbing** (examina una sola opción, no hace falta una cola de estados)
- Y si **X es tan grande** que trata todos los nodos?
  - nos encontraremos ante la técnica de **búsqueda en anchura**
- Problemas?
  - puede ocurrir el problema de la meseta, por lo que **puede no encontrar ninguna solución (no es completo)**

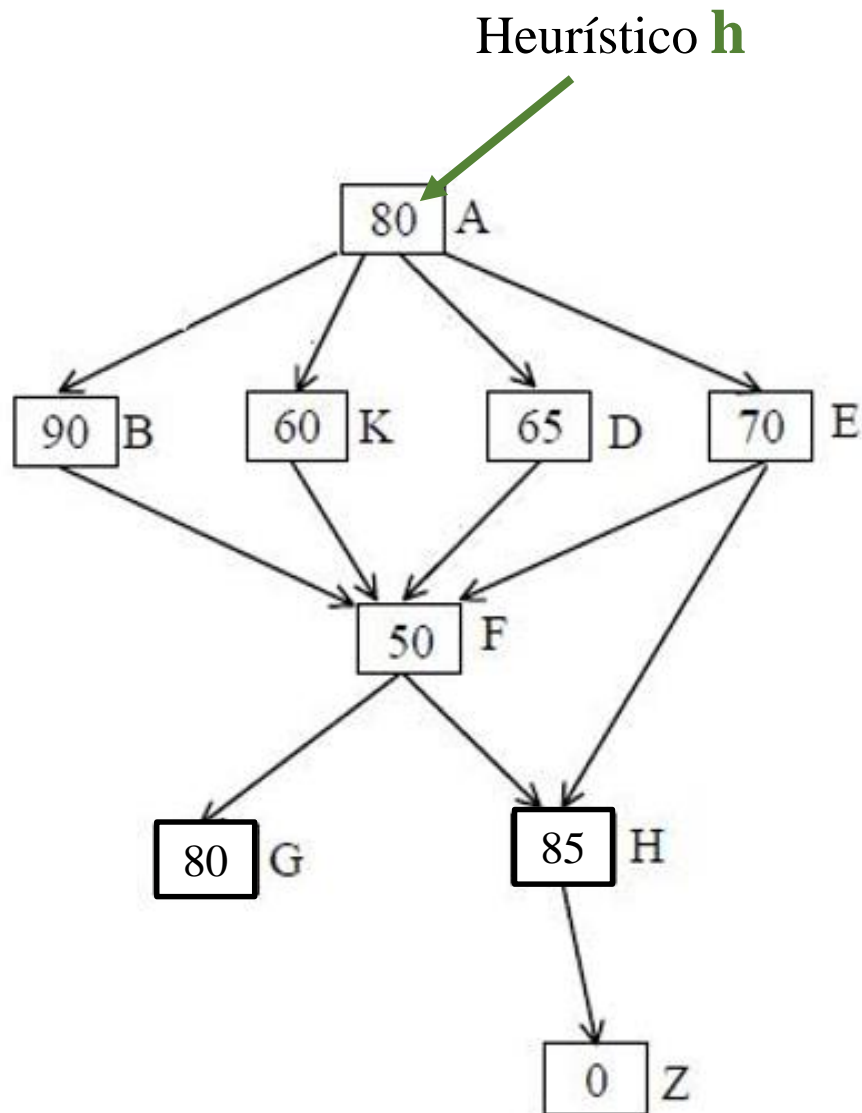
# Ejercicio 1



Estado inicial: **A**  
Estado final: **Z**

- Hill climbing, versión 1
- Hill climbing versión 2
- Beam search, X=2
- Best first

# Ejercicio 2



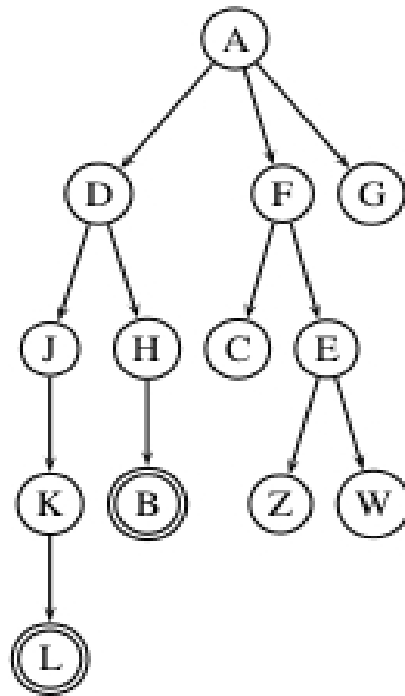
Estado inicial: **A**

Estado final: **Z**

- Hill climbing, versión 1
- Hill climbing versión 2
- Beam search, X=2
- Best first

# Ejercicio 3

- Dado el árbol de la figura 2 donde B y L son los 2 únicos nodos objetivo y A es el nodo inicial.



Indica en qué orden se visitarían los nodos, distinguiendo nodos generados de nodos expandidos, para el siguiente algoritmo:

1. Best First (heurístico = orden alfabético)

# Ejercicio 4

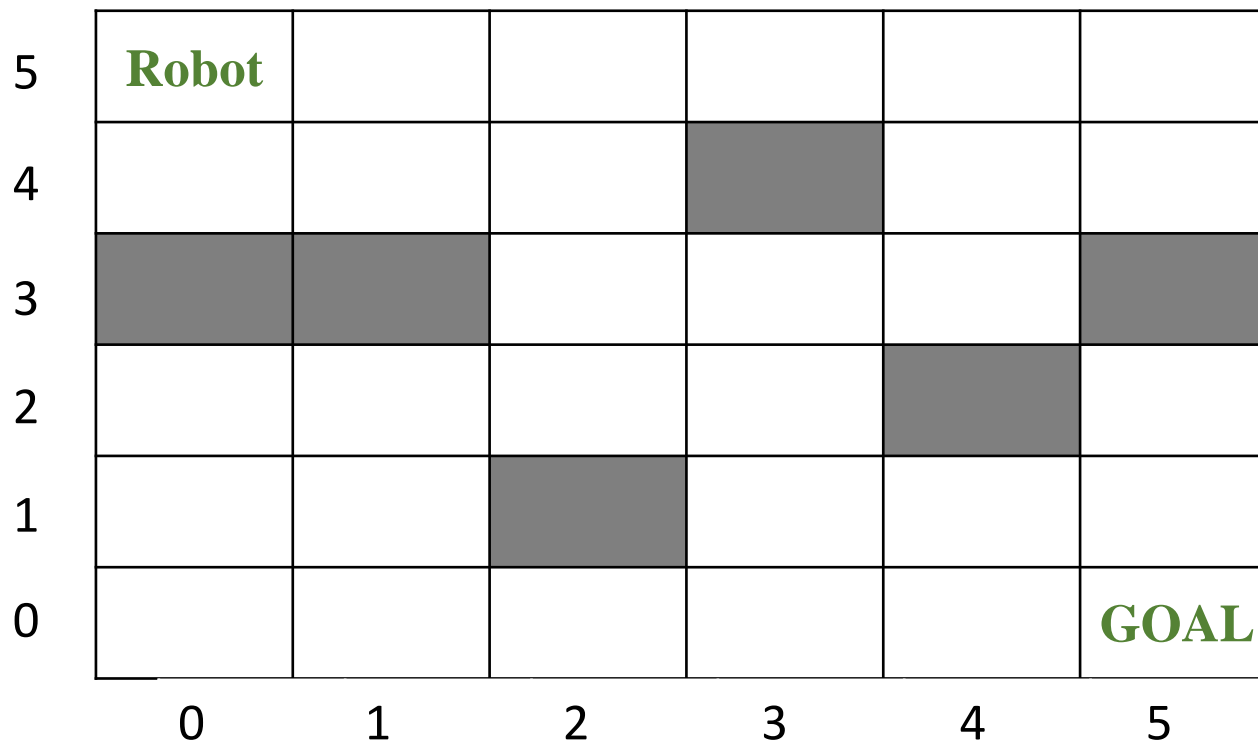
- Beam search ( $X = 3$ )
- **Heurístico:** cantidad de fichas mal colocadas

1	5	6
7		3
2	4	8



# Ejercicio 5

- Se tiene un robot autónomo en una habitación cuadrada de 6x6 casillas. El robot es capaz de realizar desplazamientos verticales y horizontales y reconoce un obstáculo si esta en una de las casillas adyacentes.



- Suponiendo que el robot utiliza algoritmos de búsqueda para planificar sus movimientos contesta a las siguientes preguntas:

# Ejercicio 5

➤ El Robot tiene que ir de una esquina de la habitación a la otra.

¿Qué heurístico podría utilizar el robot para guiarse desde su posición actual hasta el otro extremo de la habitación?

a) ¿Qué camino tomaría el robot si su movimiento estuviese determinado por un algoritmo de búsqueda hill climbing con el heurístico del apartado anterior? Suponer que los estados sucesores se generan aplicando los operadores de desplazamiento en este orden: [**derecha, abajo, izquierda, arriba**] (**nota: si el valor del heurístico es igual, entonces la elección dependerá del orden de los operadores**)

b) Y si el orden de los operadores fuesen: [**arriba, izquierda, abajo, derecha**] ¿Qué camino tomaría?

c) Siguiendo el primer orden de operadores ¿Qué camino tomaría el robot si sus movimientos estuviesen guiados por un algoritmo de beam search con  $X=3$ ?