Inteligencia Artificial

Espacio de estados y búsqueda

Basado en:

- Introduction to IA CS188, Univ. Berkeley

ROBOTIKA & SISTEMA AUTONOMOEN IKERKETA TALDEA

eman ta zabal zazu
Universidad del País Vasco
Euskal Herriko Unibertsitatea

Índice

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:

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

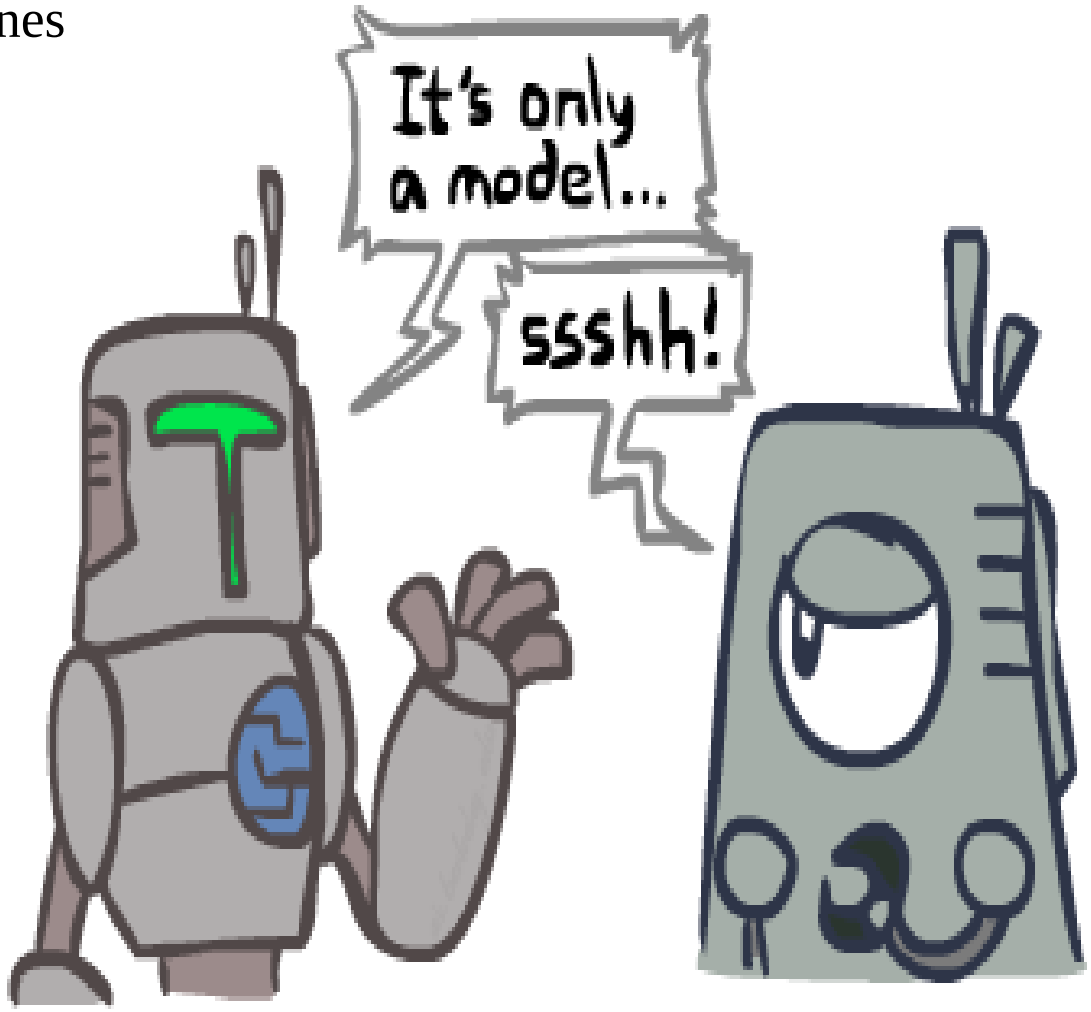
3.3 Búsqueda adversarial

- Minimax
- Alfa-beta
- Expectimax

Búsqueda informada (Informed Search)

Búsqueda y Modelos

- La búsqueda opera sobre modelos del mundo
  - El agente no prueba todos los planes en el mundo real
  - La planificación se hace en modo "simulación"
  - Nuestra 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 ⇨ **eficiencia**

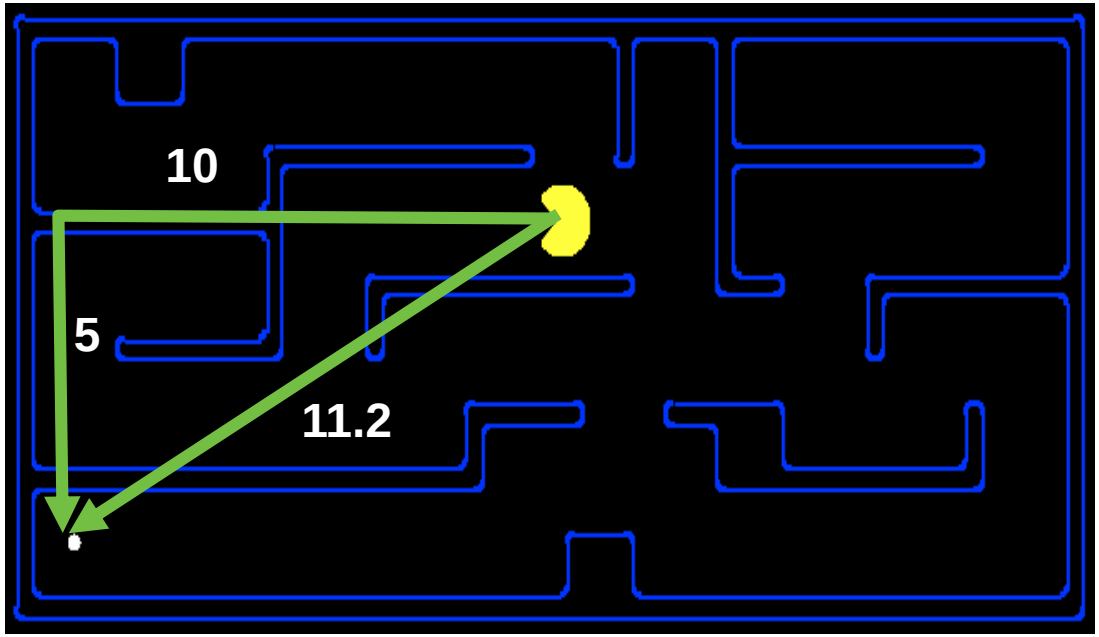
- **Adecuados para espacio de estados amplios**

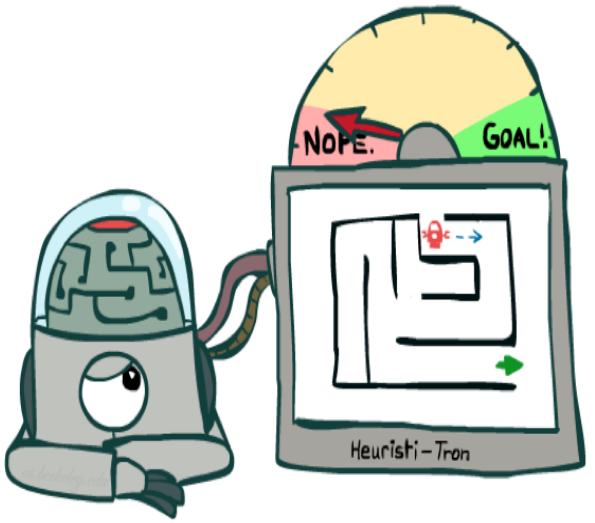
Heurísticos de búsqueda

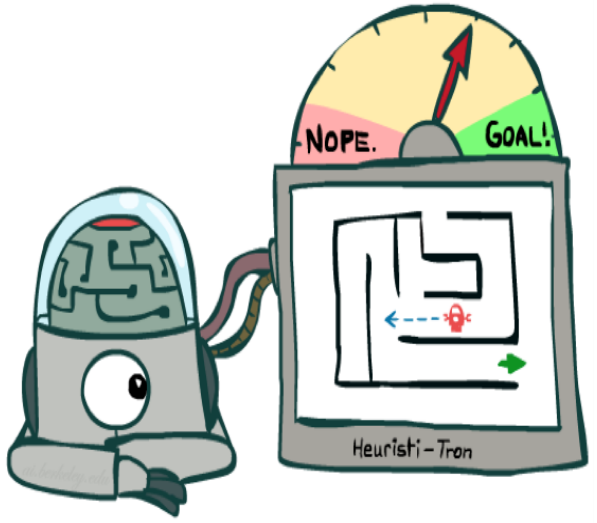
- Características:
  - **No** garantiza que se encuentre una solución, **aunque exista**.
  - En caso de encontrar una solución, **no siempre aseguran** que ésta sea la de longitud mínima o la de coste óptimo.
  - **En algunas ocasiones** (que, en general, no se podrán determinar a priori), encontrará 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: ***distancia de Manhattan***, ***distancia Euclidea*** para buscar un camino







Métodos que realizan una búsqueda heurística

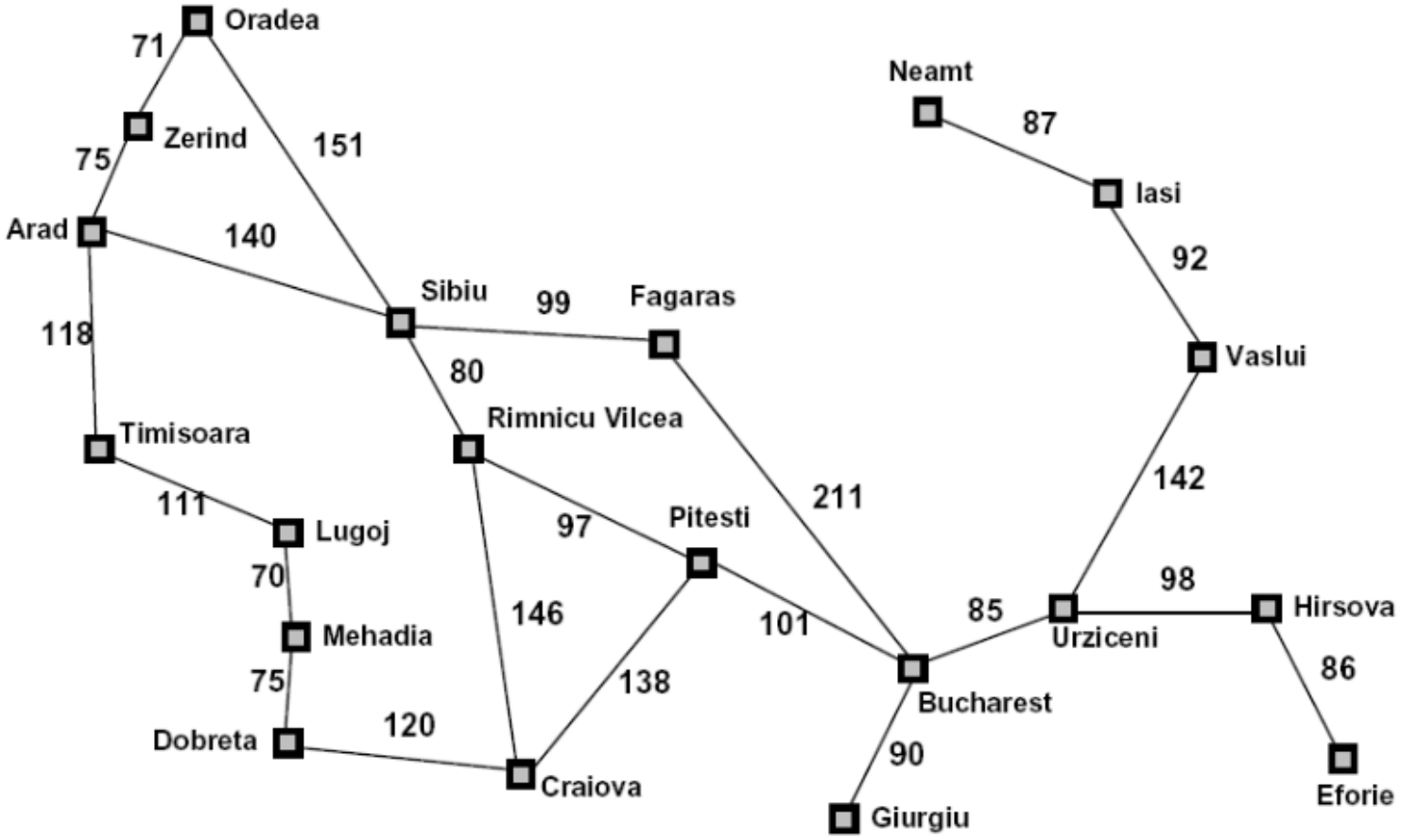
- Algoritmos que ***quizás*** encuentran una ***solución cualquiera***:
  - Memoria limitada: **eliminan ramas del espacio del estado** que pueden llevar a la solución (Greedy Search):

    - **Hill Climbing** → comportamiento en profundidad
    - **Beam search** → comportamiento en anchura
    - **Best first** → comportamiento en profundidad

- Algoritmos que encuentran la ***solución óptima***:

    - Ramificar y Acotar (Brand and Bound)
    - A*
    - Grafos AND/OR

Ejemplo: función Heurística

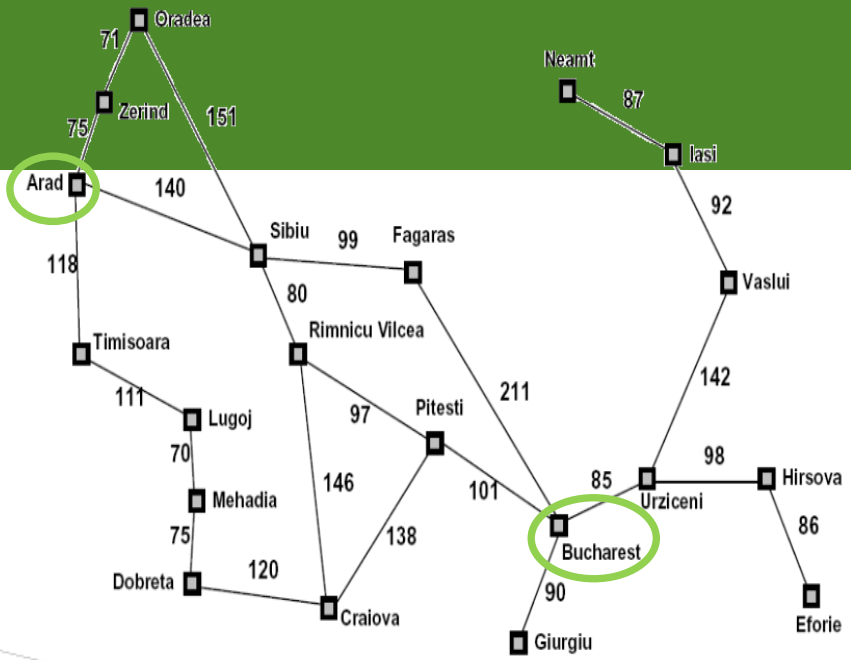


Straight−line distance to Bucharest	
Arad	366
Bucharest	0
Craiova	160
Dobreta	242
Eforie	161
Fagaras	178
Giurgiu	77
Hirsova	151
Iasi	226
Lugoj	244
Mehadia	241
Neamt	234
Oradea	380
Pitesti	98
Rimnicu Vilcea	193
Sibiu	253
Timisoara	329
Urziceni	80
Vaslui	199
Zerind	374

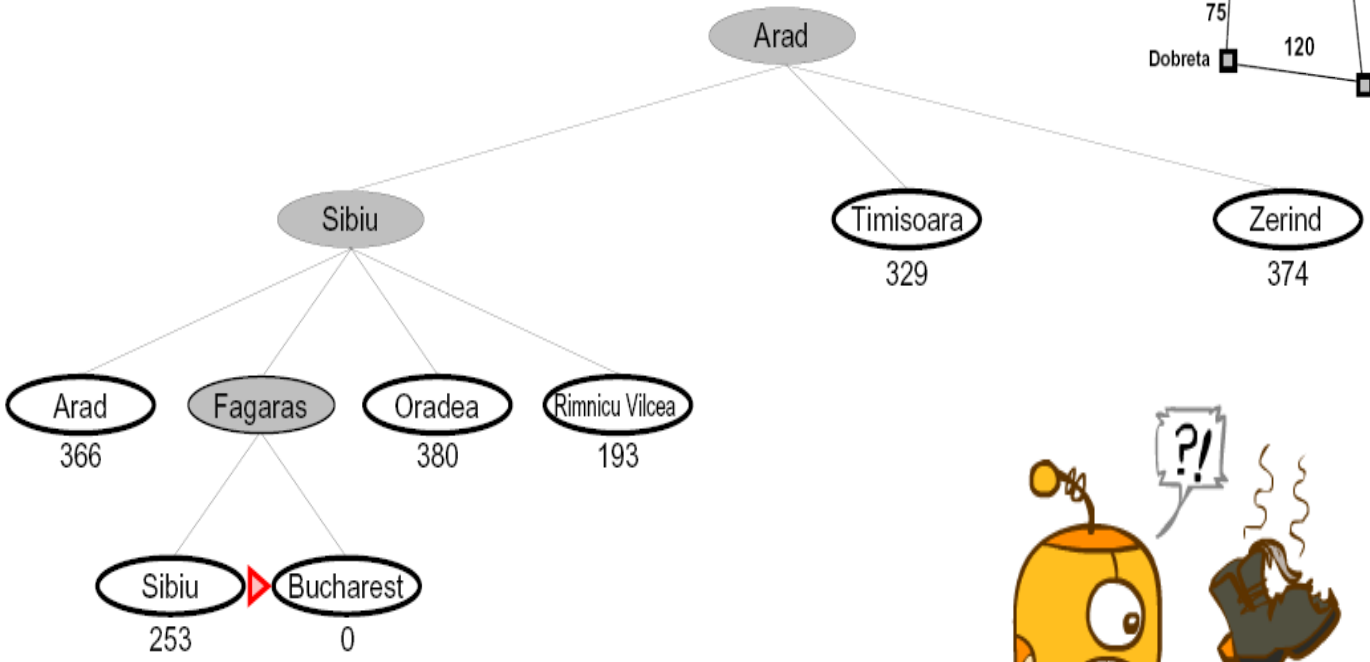
h(x)

Búsqueda voraz (Greedy Search)

Greedy Search



- Expandir el nodo que parece más cercano…
  - f(n) = h(n)

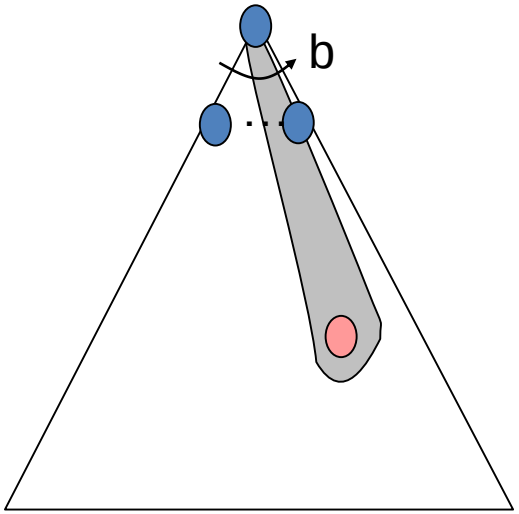


Straight–line distance to Bucharest	
Arad	366
Bucharest	0
Craiova	160
Dobreta	242
Eforie	161
Fagaras	178
Giurgiu	77
Hirsova	151
Iasi	226
Lugoj	244
Mehadia	241
Neamt	234
Oradea	380
Pitesti	98
Rimnicu Vilcea	193
Sibiu	253
Timisoara	329
Urziceni	80
Vaslui	199
Zerind	374

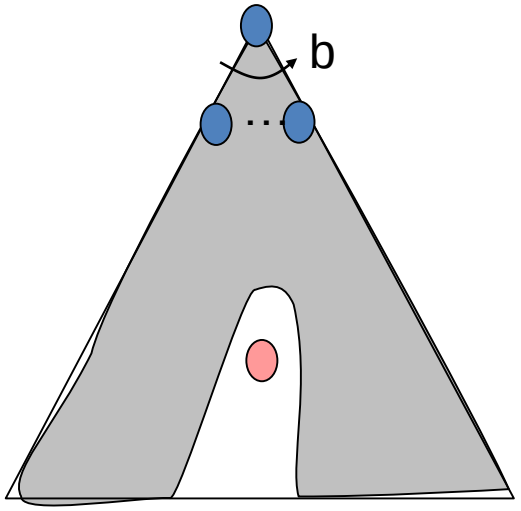
- Qué puede fallar?

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



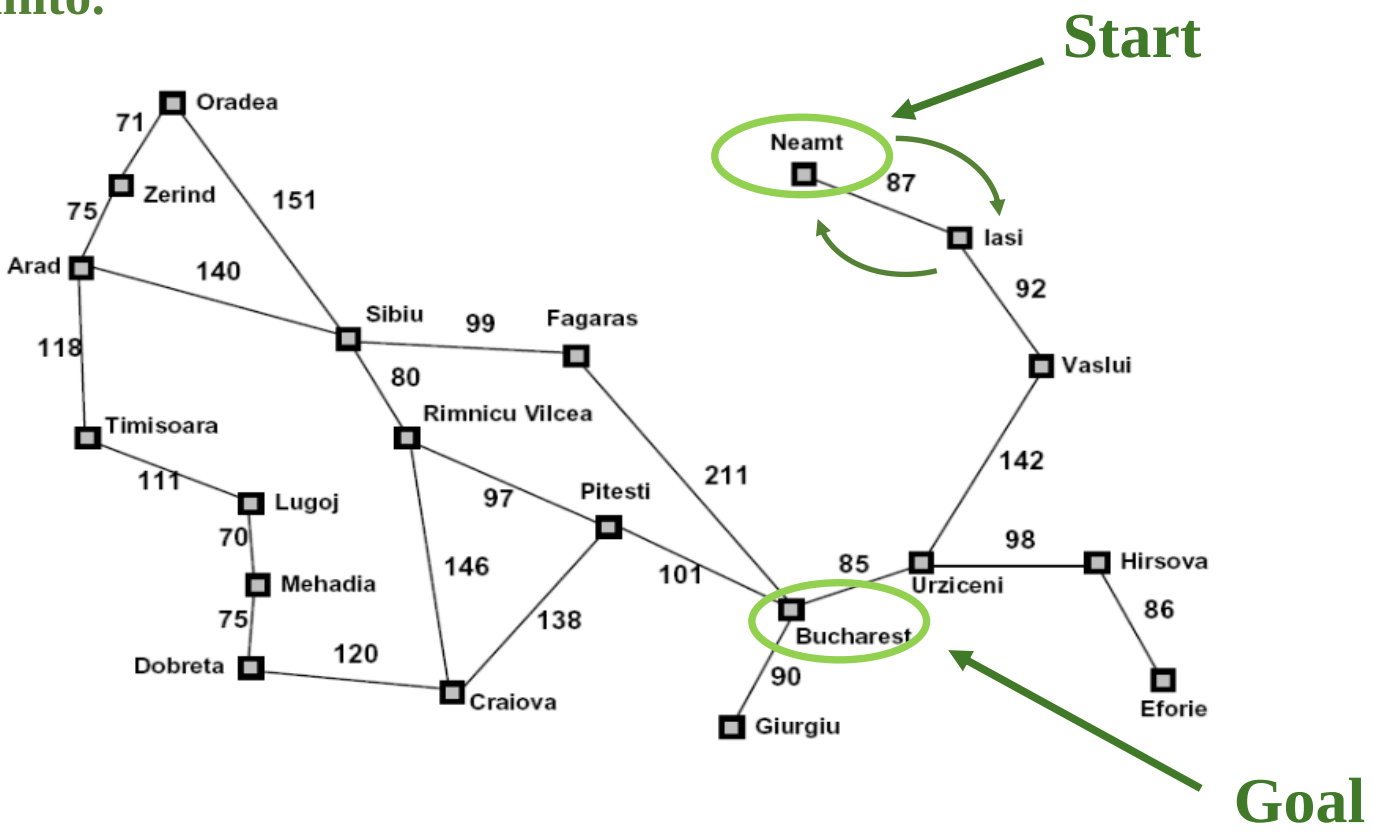
- **Un caso frecuente**:
  - Nos lleva directamente a un objetivo (quizás incorrecto)



- **Caso peor**:
  - como DFS mal guiado = **$O(b^m)$**
  - guarda todos los nodos

Greedy Search

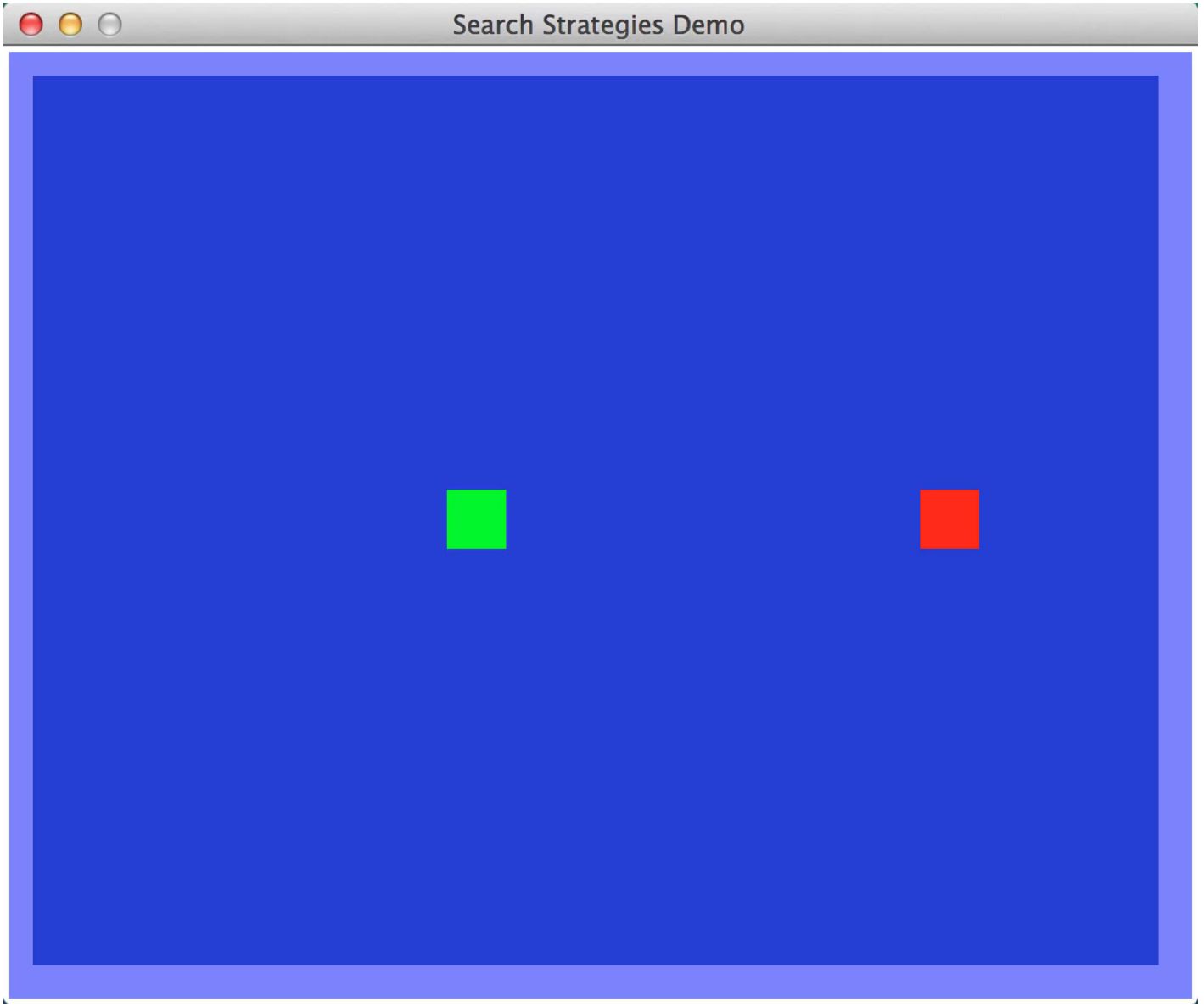
- Óptimo?: **NO**
- Completo?
  - No en la búsqueda en árbol. Incluso con un espacio de estados finito, **puede entrar en un bucle infinito.**



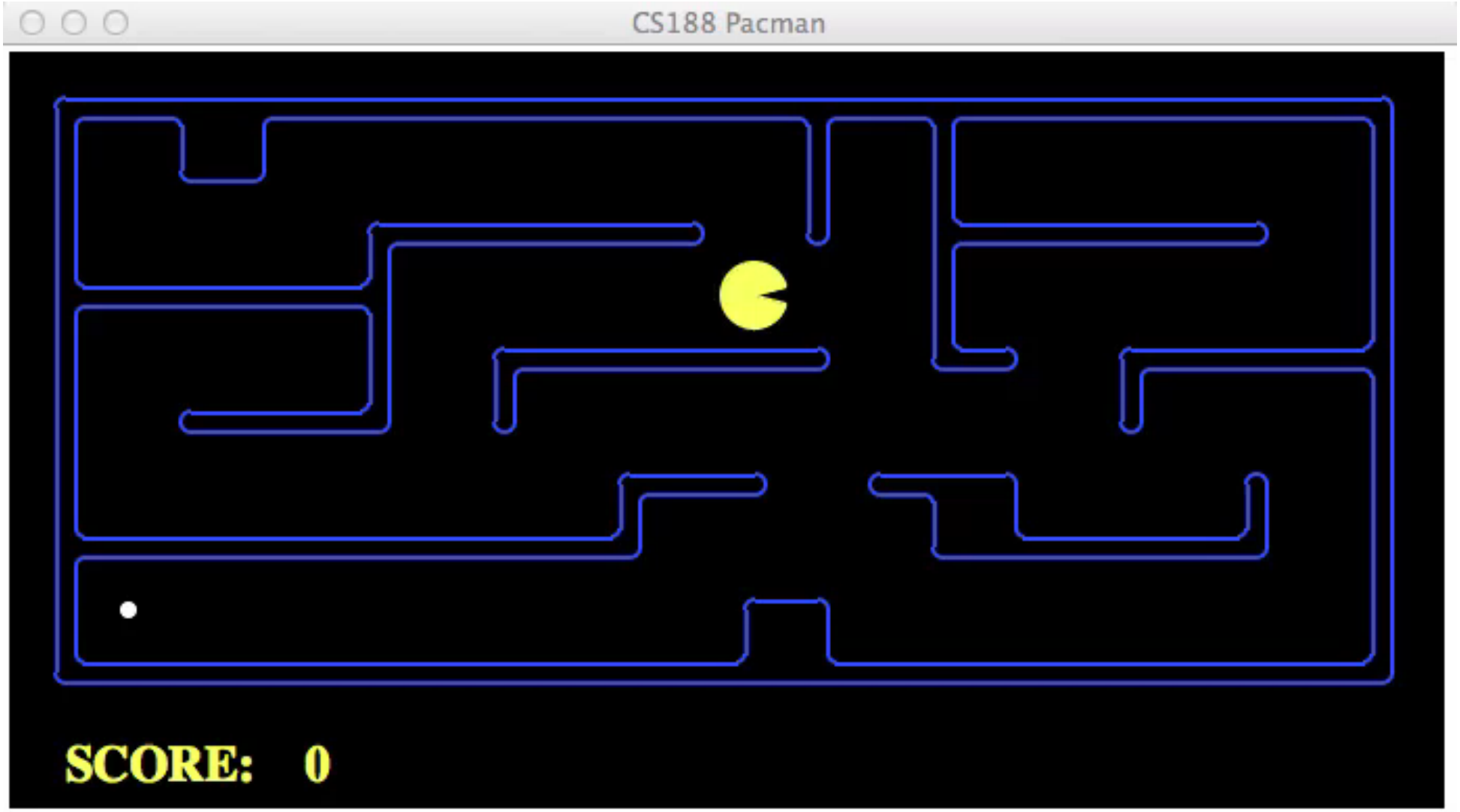
- Completo?: **Sí** en búsqueda en grafo **si el espacio es finito!**

[Demo: contours greedy empty (L3D1)]
[Demo: contours greedy pacman small maze (L3D4)]

Video of Demo Greedy (Empty)

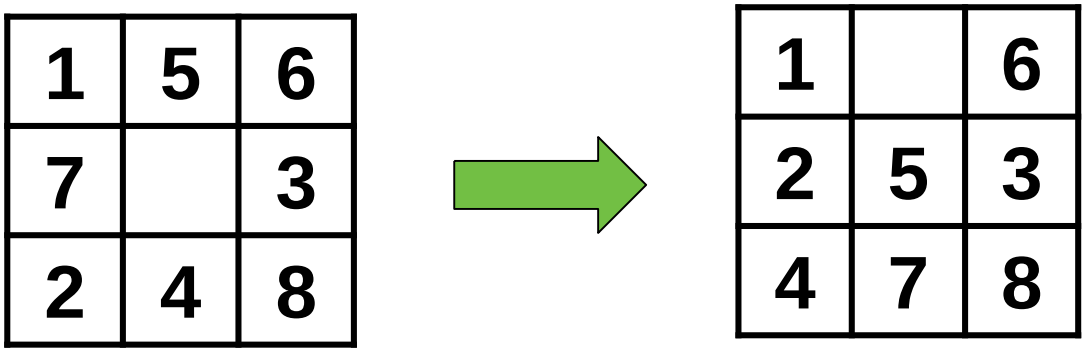


Video of Demo Contours Greedy (Pacman Small Maze)



Problema puzzle

- Heurístico a utilizar:
  - Número de fichas que hay desordenadas en cada estado ⇨ **H=4**



  - Otro: Suma de la distancia a la solución de cada número (Distancia de Manhatan):
    - 1(5) + 1(2) + 1(4) + 2(7) ⇨ **H=5**

Hill climbing - versión 1

Algoritmo

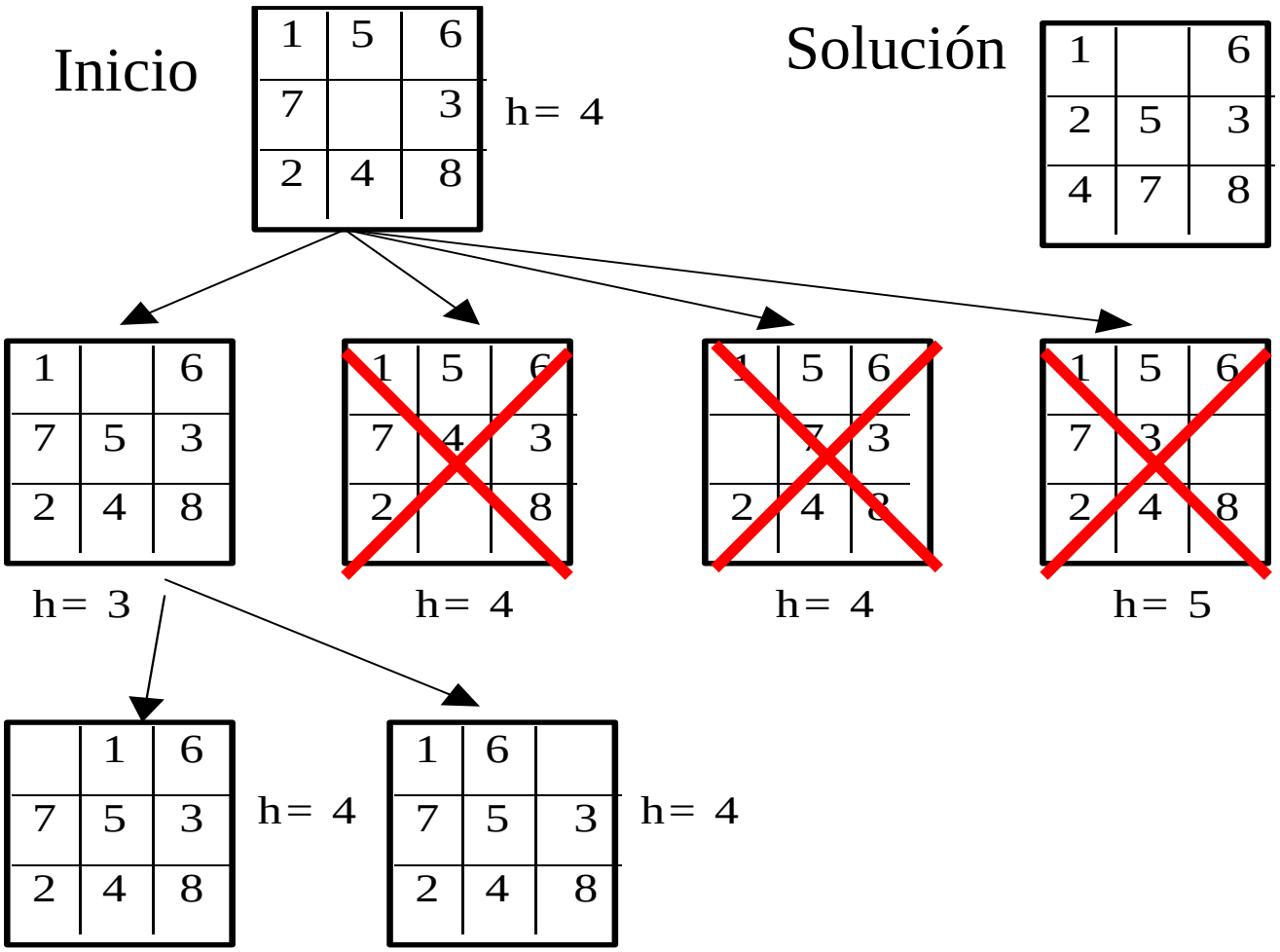
1- Construir una lista con el nodo raíz como único elemento.

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, **x_1**, de los ordenados está **MÁS CERCA** de la solución de lo que estaba el nodo padre, es decir **$h(padre) > h(x_1)$** 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 → 3 < 4

Hill climbing - versión 1

- Problema:
  - Pequeñas colinas

Hill climbing - versión 2

Algoritmo

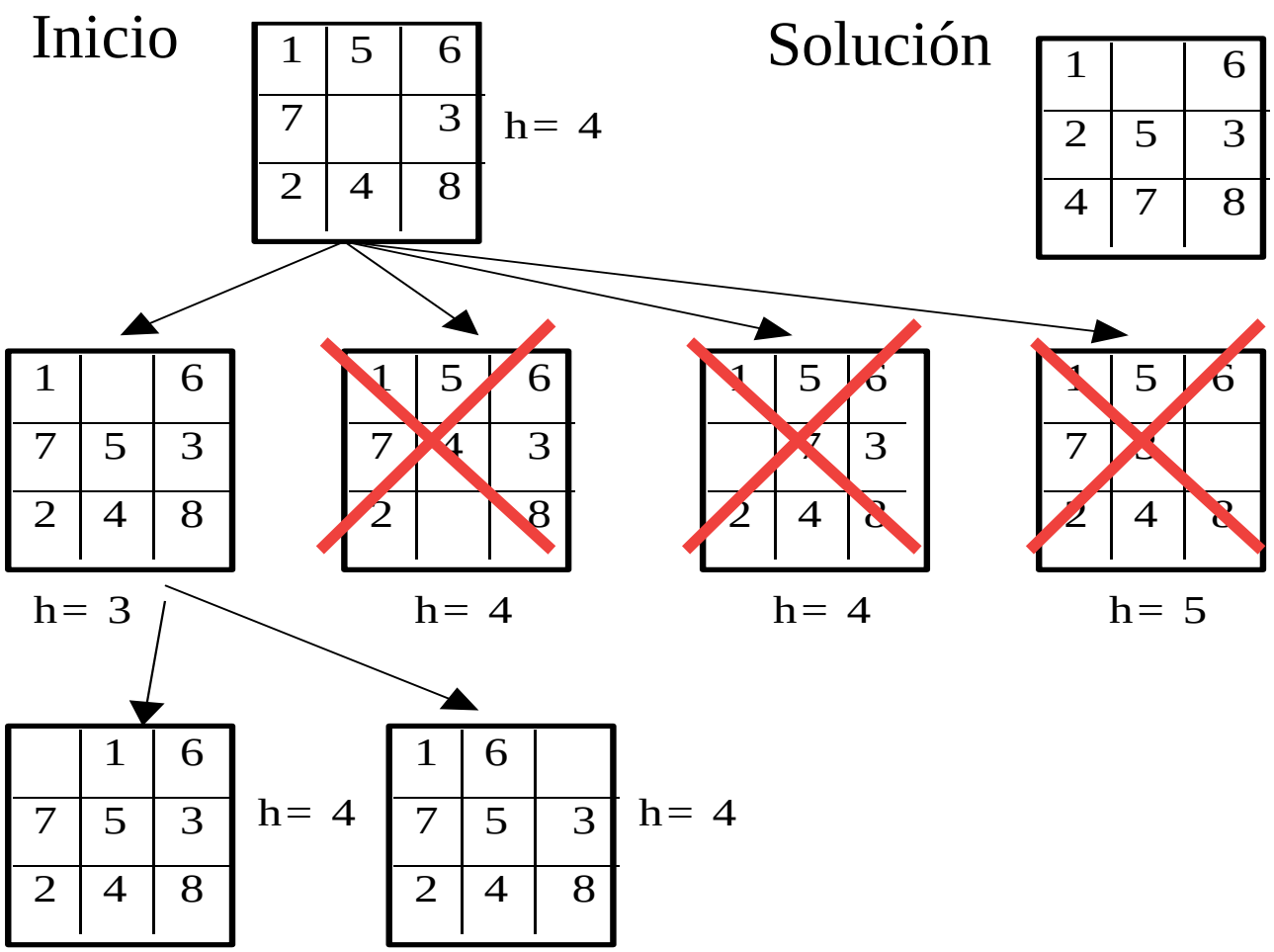
1- Construir una lista con el nodo raíz como único elemento.

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, $\mathbf{x_1}$, 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.

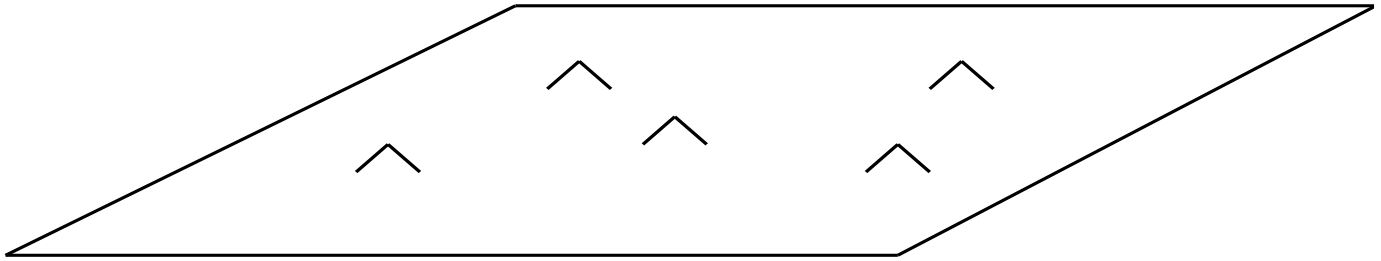
Árbol de búsqueda. Ejemplo HC versión 2



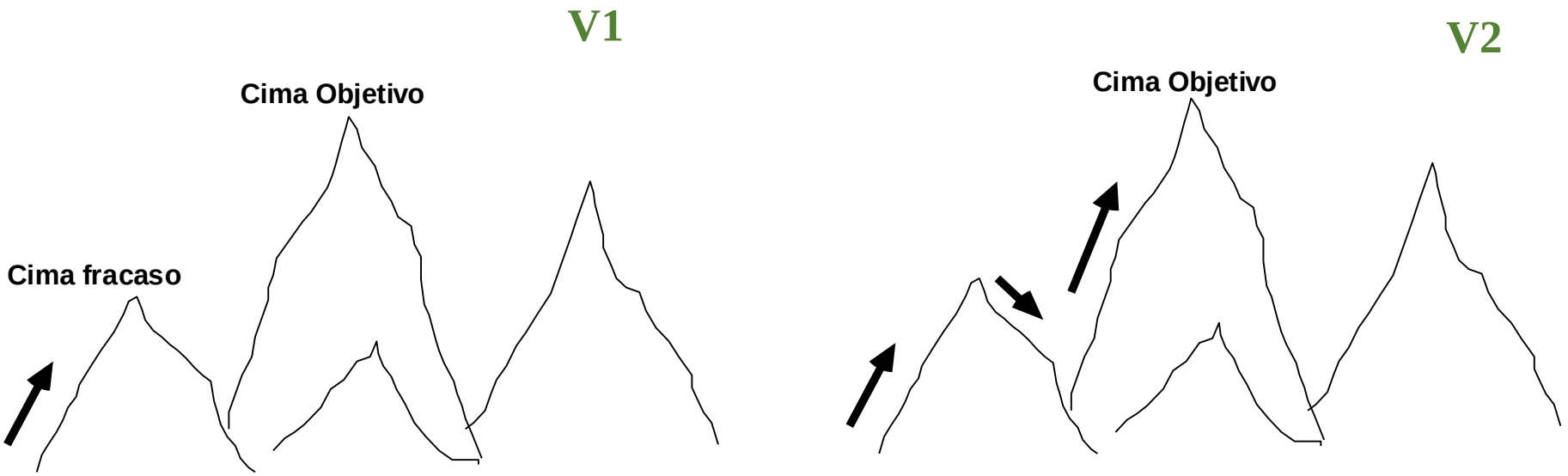
En este punto el algoritmo **NO** fracasa y seguiría tratando de buscar una solución

Hill climbing - versión 2

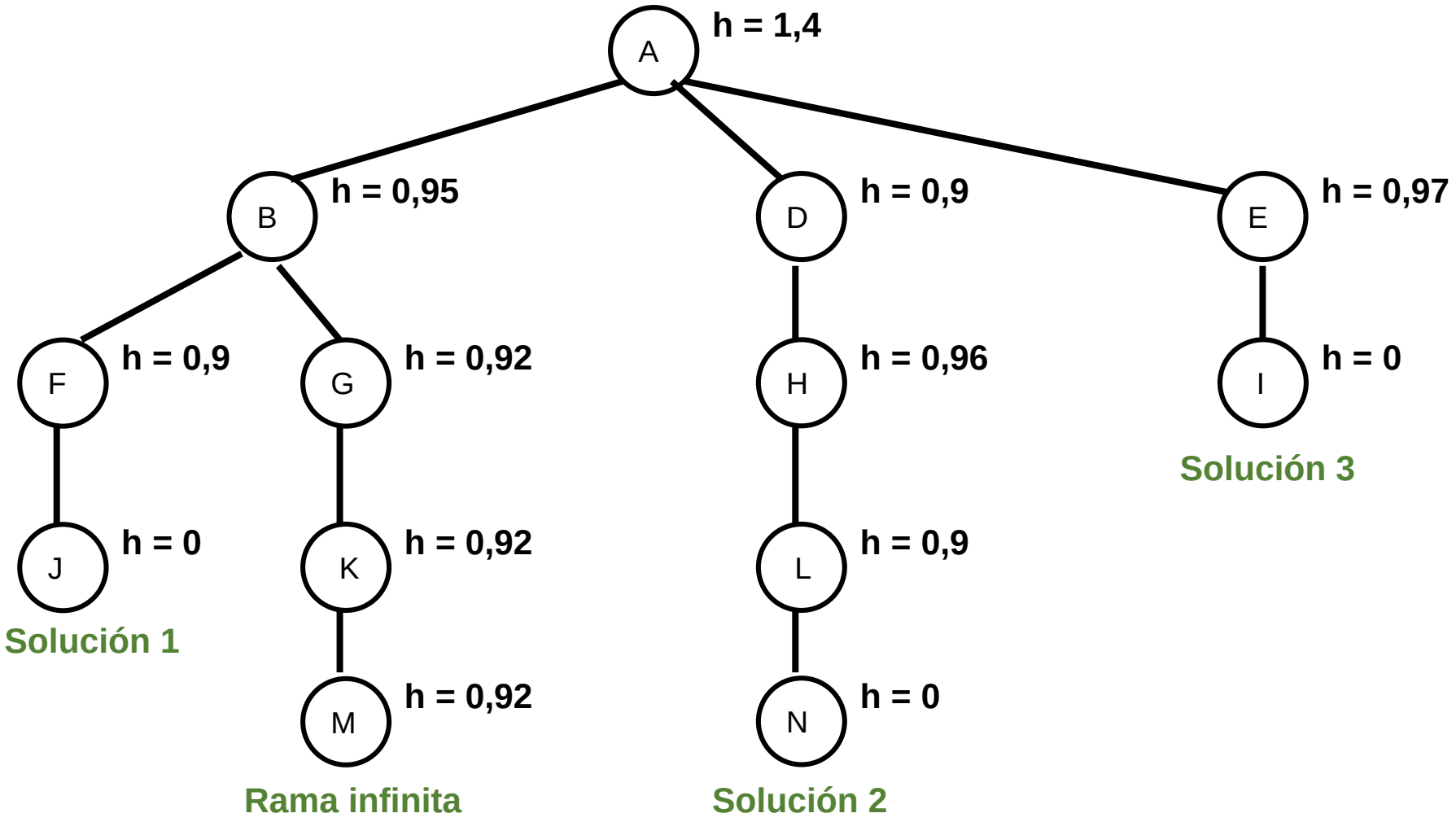
- Problema:
  - Meseta → **todos los hijos mismo valor heurístico**!



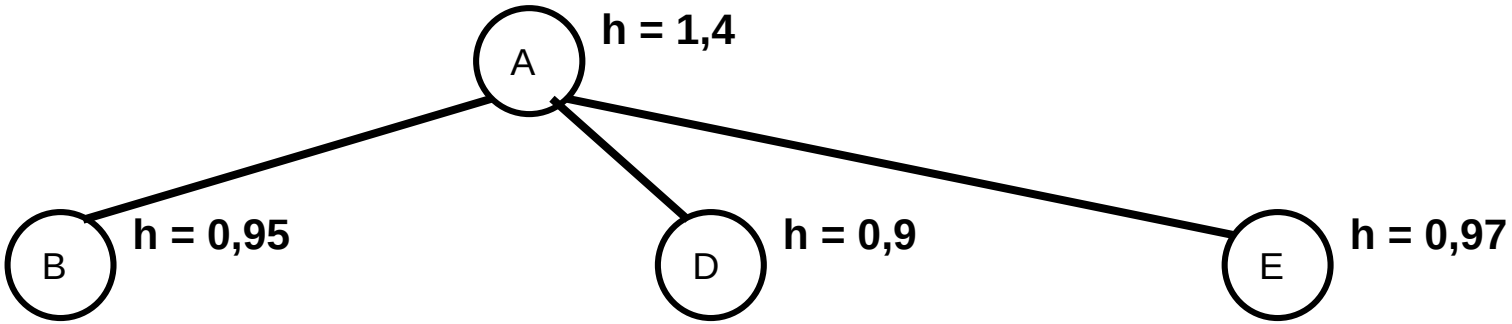
Comparativa Hill climbing - V1 y V2



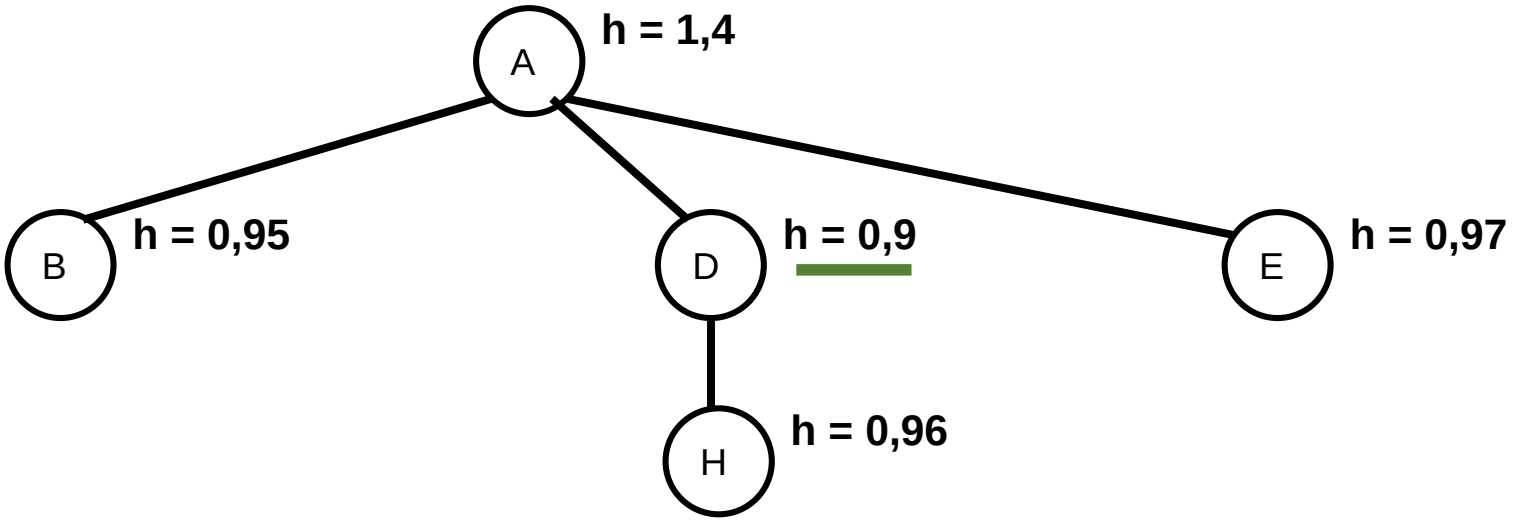
Hill climbing V1 / V2



Hill climbing V1

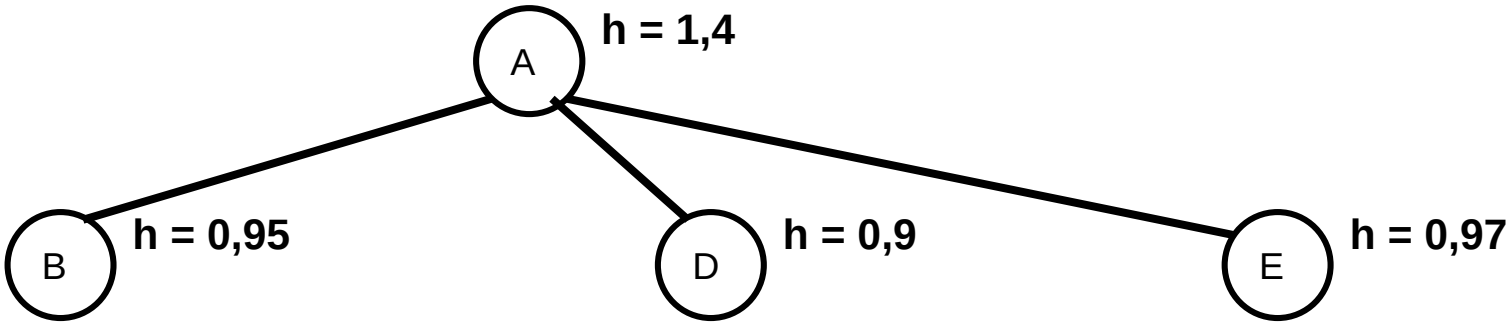


Hill climbing V1

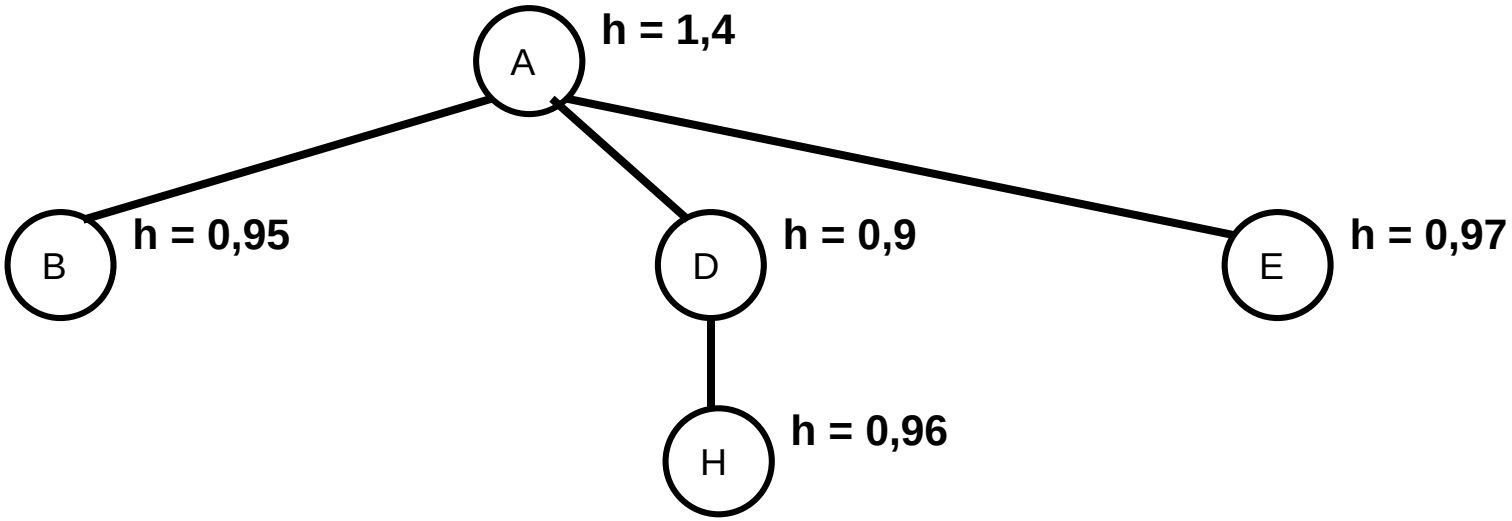


Fallo: en este caso el algoritmo fracasa: 0.9 < 0.96

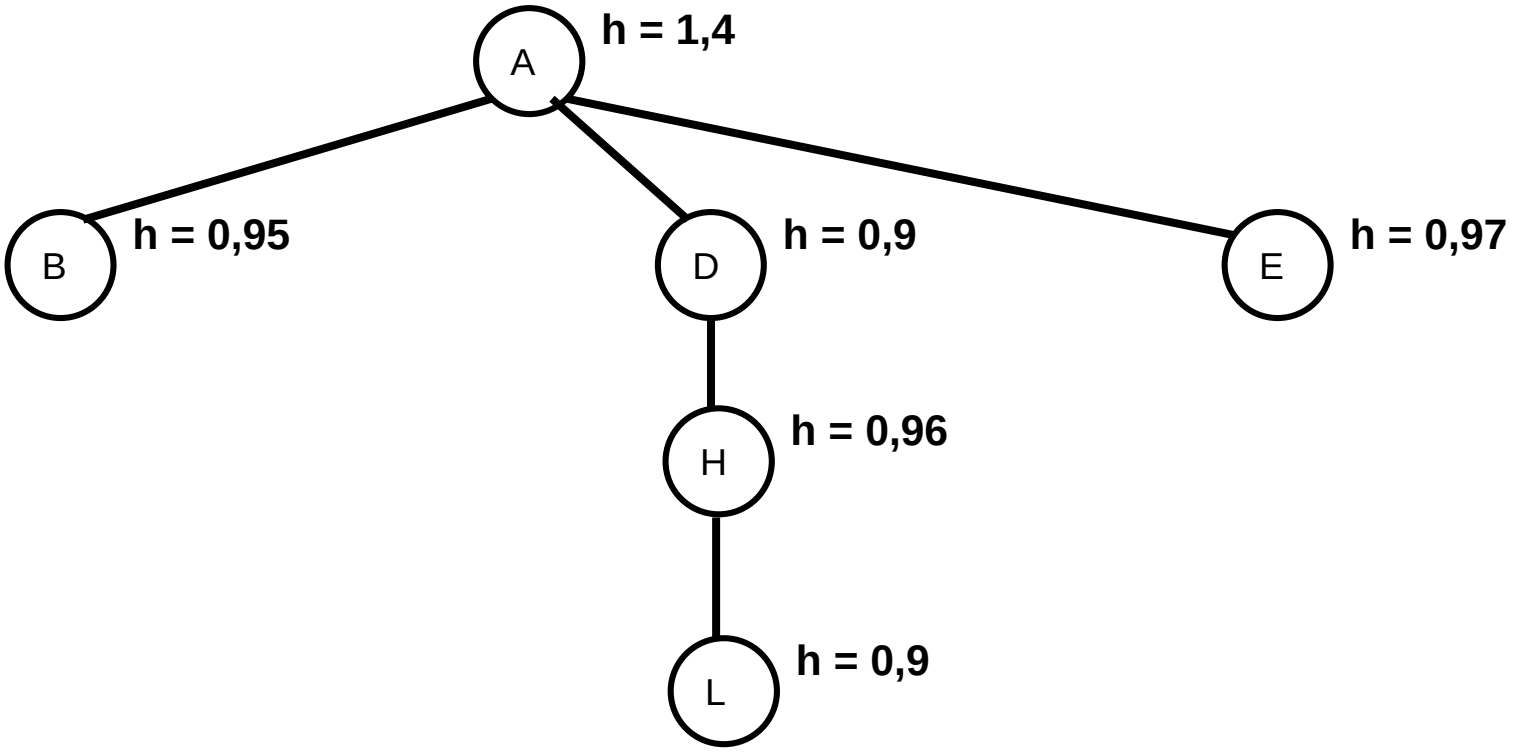
Hill climbing V2



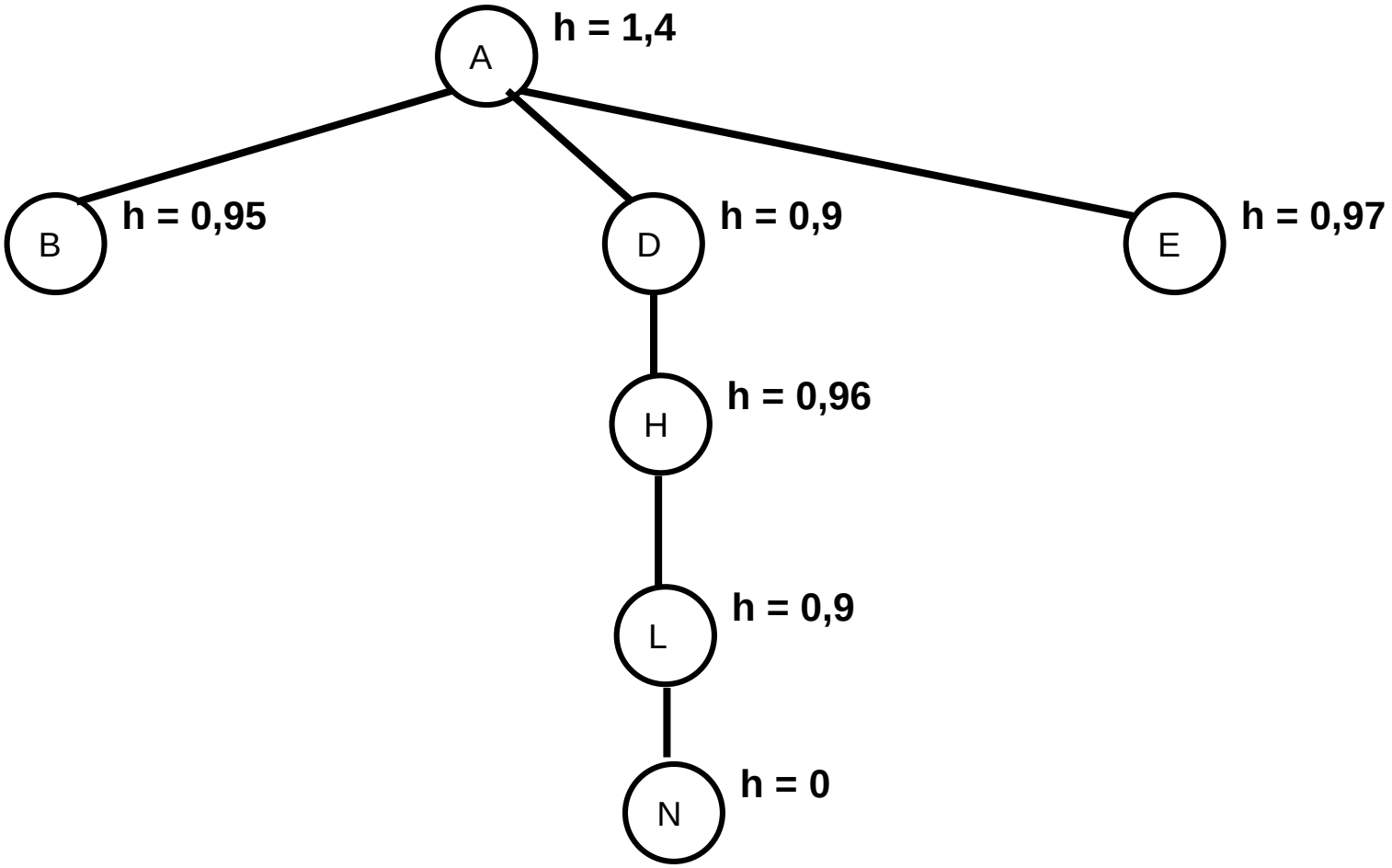
Hill climbing V2



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**
  - **Mesetas**: todos los hijos tienen mismo valor heurístico que el padre. Una meseta es una región del espacio de estados donde todos los estados tienen el mismo valor heurístico. **El algoritmo para sin dar solución**. Si sigue, la heurística no informa ⇒ **búsqueda ciega**

Best first (el mejor primero)

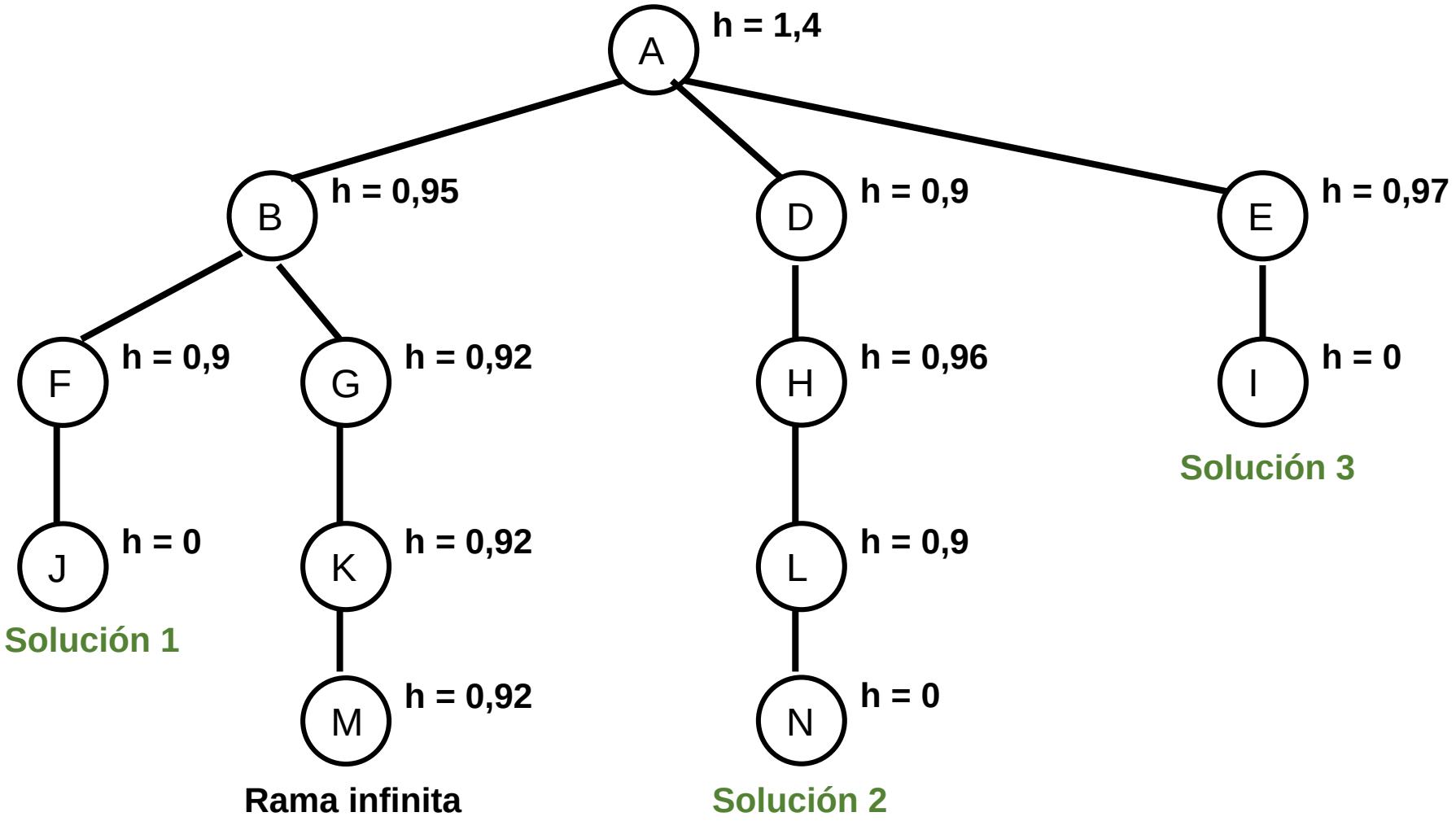
Algoritmo

- Construir una lista con el nodo raíz como único elemento.
- Hasta que la lista esté vacía o el primer elemento de la lista sea el elemento objetivo
  - 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.
- Si se ha encontrado el nodo objetivo, anunciar éxito, si no, fallo.

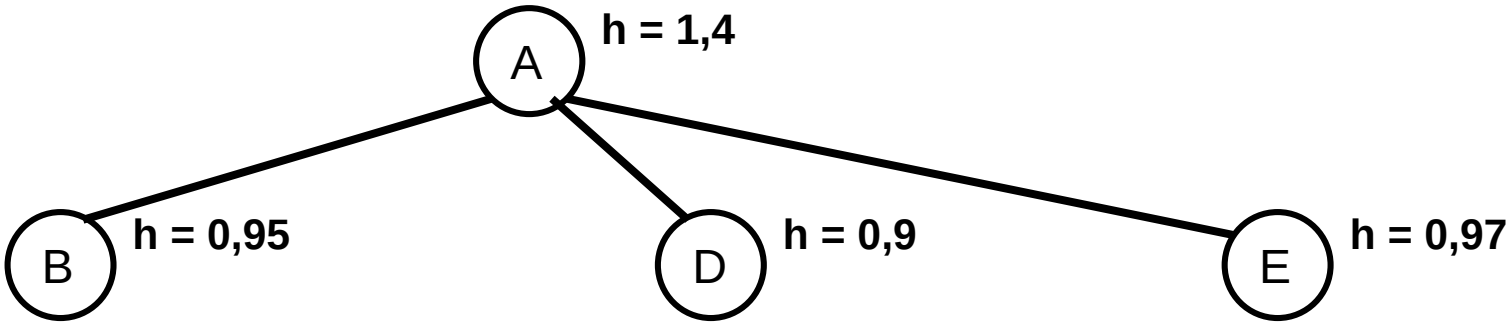
Problema

- Problemático para problemas con ramificación alta

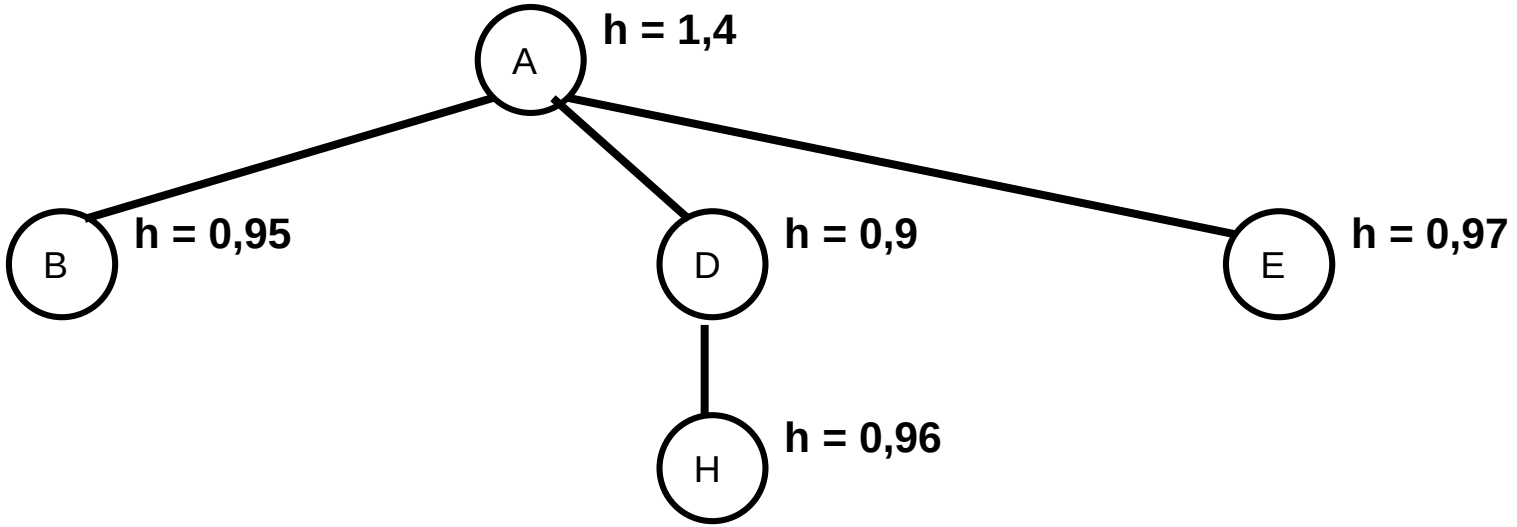
Best First


A
h = 1,4
B
h = 0,95
D
h = 0,9
E
h = 0,97
F
h = 0,9
G
h = 0,92
H
h = 0,96
I
h = 0
Solución 3
J
h = 0
K
h = 0,92
L
h = 0,9
Solución 1
M
h = 0,92
N
h = 0
Rama infinita
Solución 2

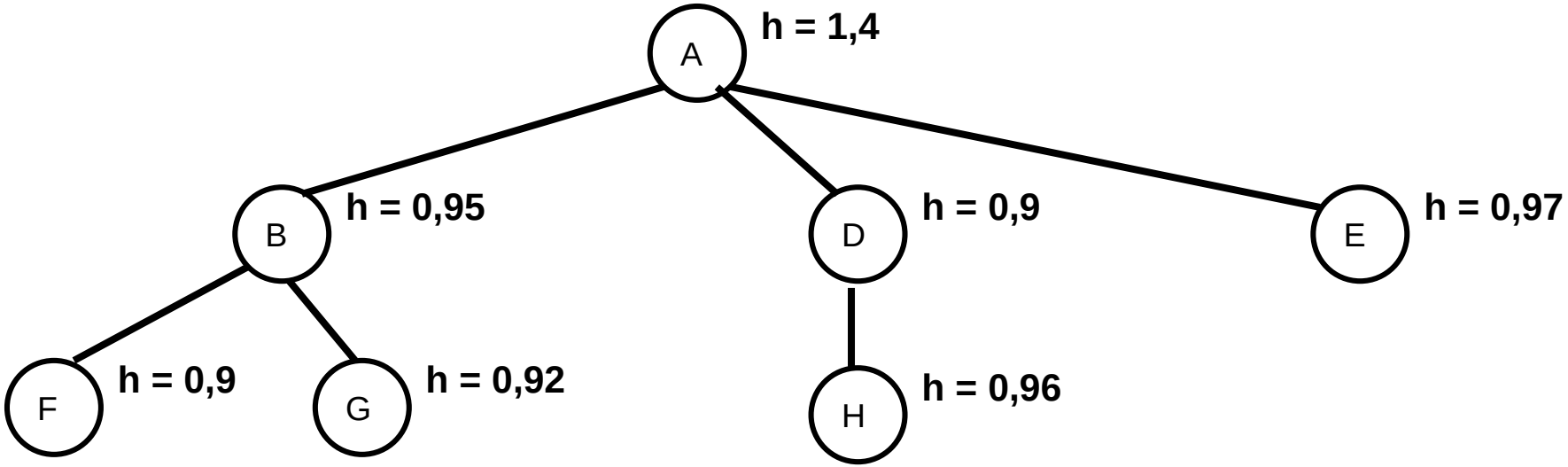
Best First



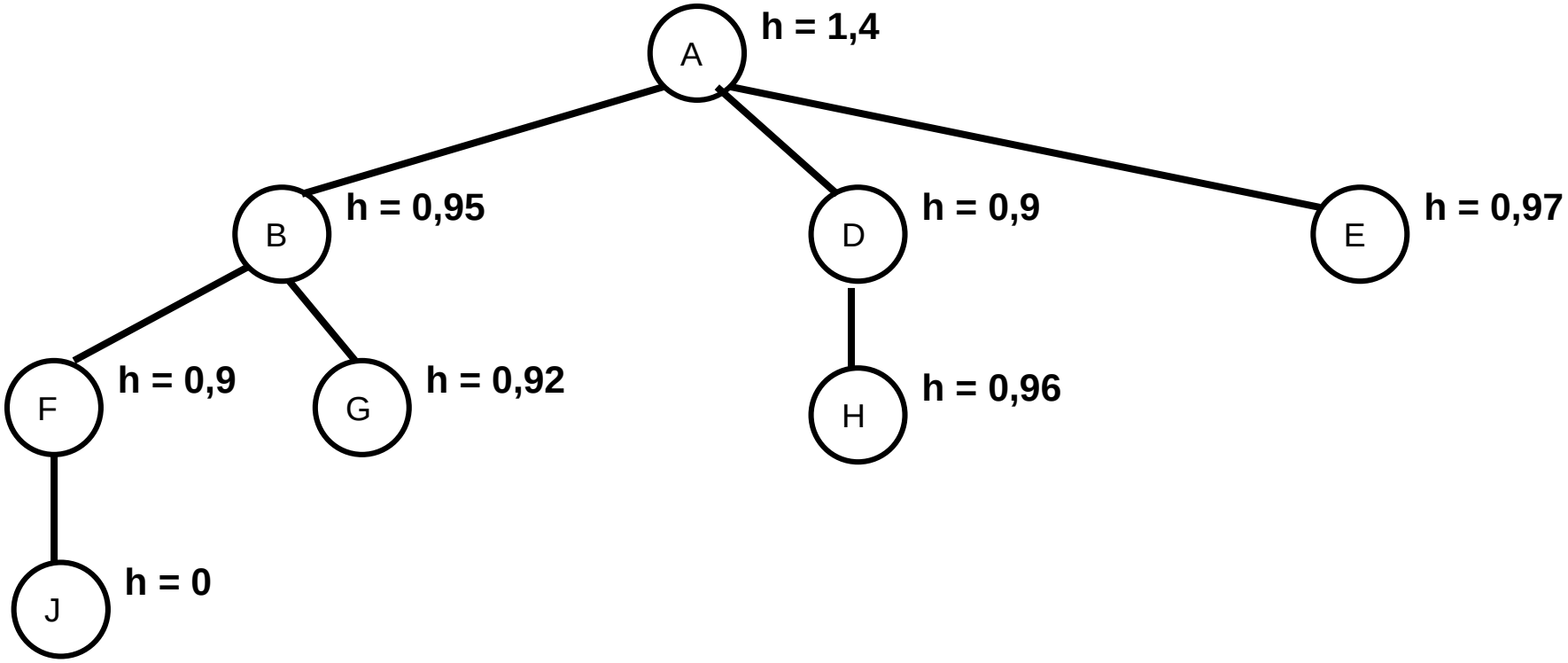
Best First



Best First



Best First



Solución 1

Beam search

- **Alternativa** entre Hill Climbing y Best-first
- Basado en búsqueda en anchura.
- Mantiene las **X** mejores alternativas
- **Algoritmo**:
  - Determinar X como el nº máximo de nodos por nivel a expandir; Lista_X = (nodo raíz); Lista_hijos = ( )
  - Hasta que Lista_X esté vacía o Lista_X contenga un nodo objetivo
    - » Si Lista_X contiene una solución:
      - salir del bucle anunciando éxito para primer nodo solución
    - » Si no:
      - Eliminar todos los elementos de Lista_X y añadir sus hijos (**si los hubiera**) a Lista-hijos ordenados crecientemente **según h**
    - » Meter en Lista_X los X primeros nodos de Lista-hijos
  - **Si Lista_X está vacía, anunciar fracaso**

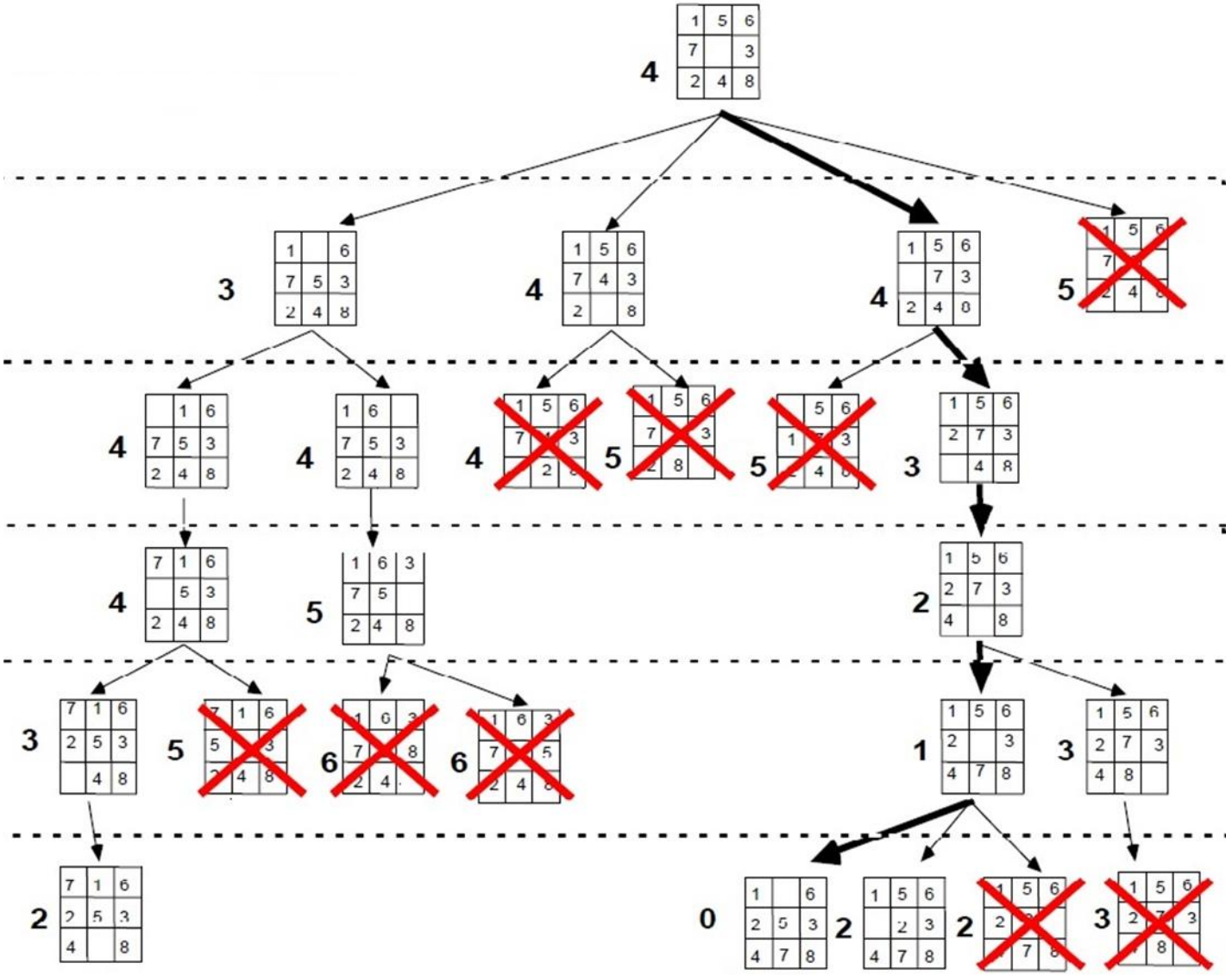
Beam search

- Cuando **X vale 1**, tenemos la técnica Hill Climbing (examina una sola opción, no hace falta una cola de estados)
- Si **X es tan grande que trata todos los nodos**, nos encontraremos ante la técnica de búsqueda en anchura
- Puede acabar en máximos locales, por lo que **puede no encontrar ninguna solución** (**no es completo**)

Beam search, X = 3

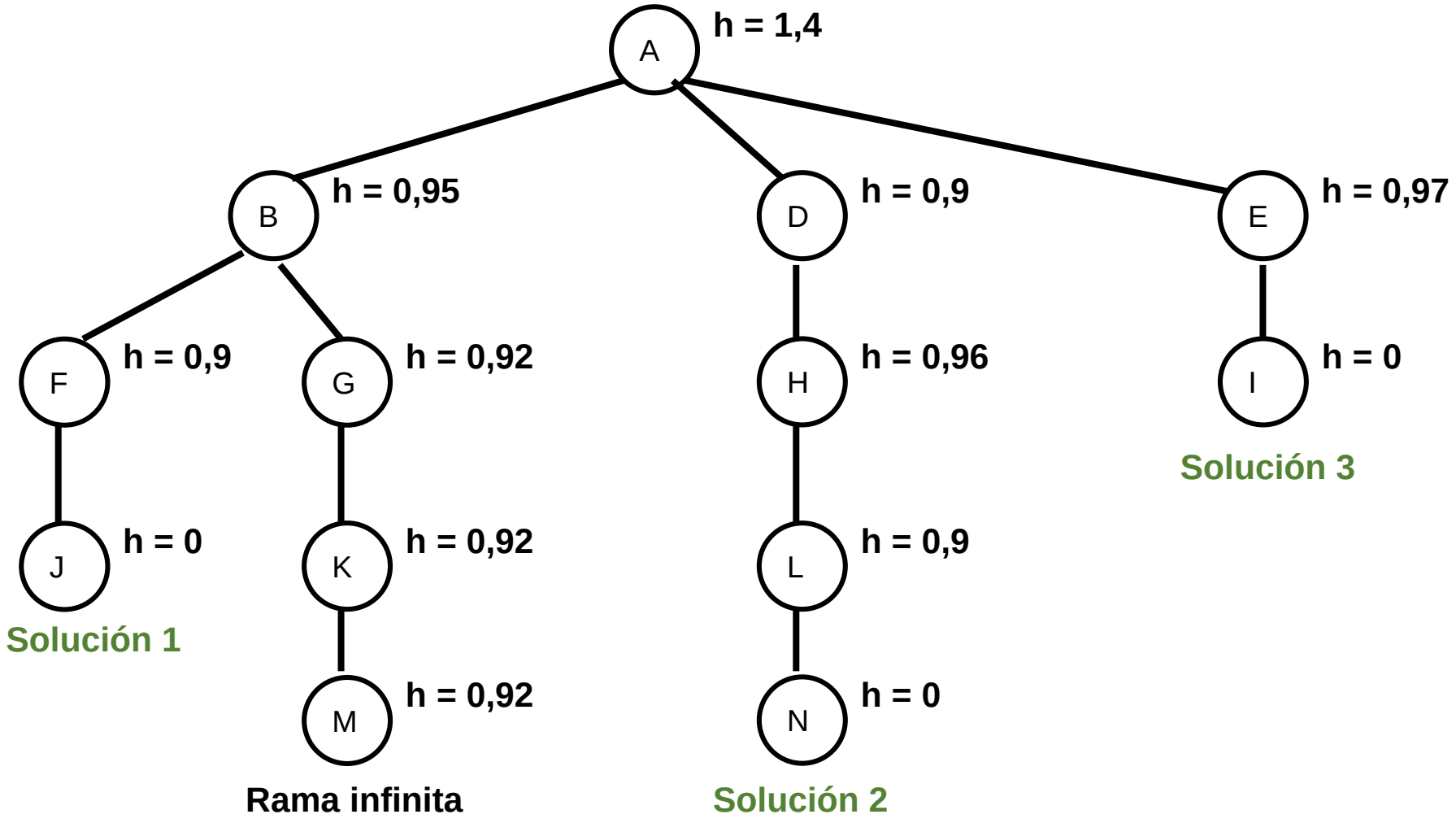
1	5	6
7		3
2	4	8

Beam search, X = 3

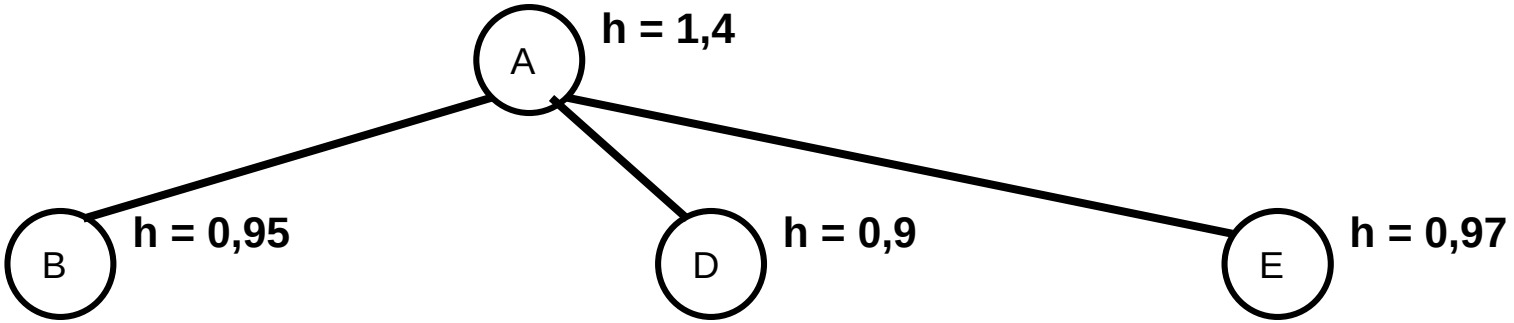


Solución

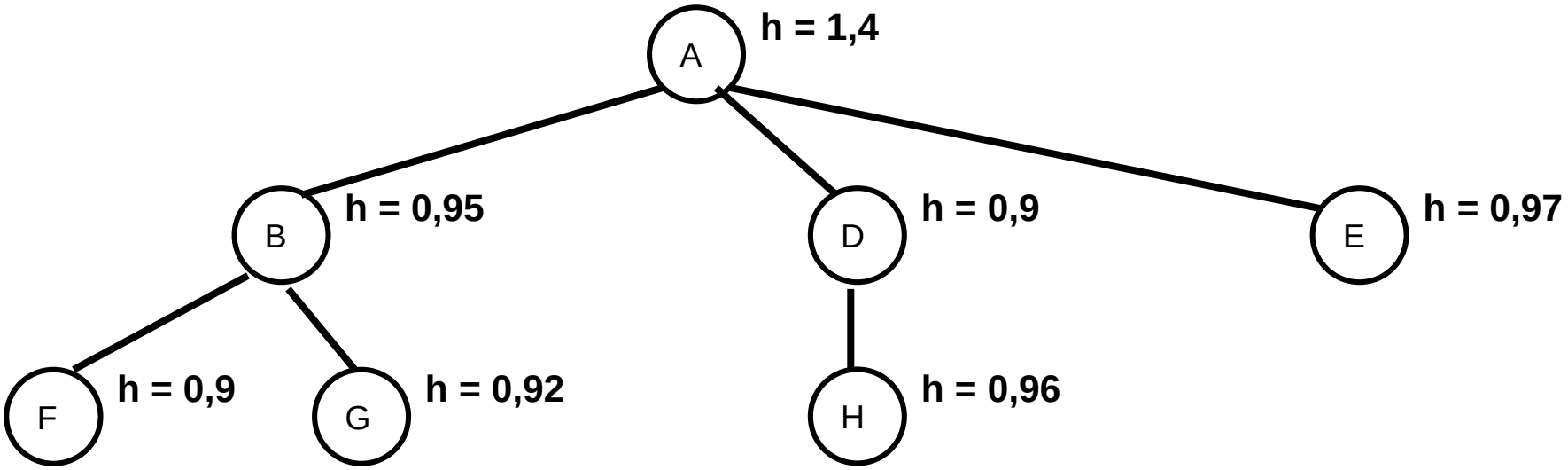
Beam search, X = 2



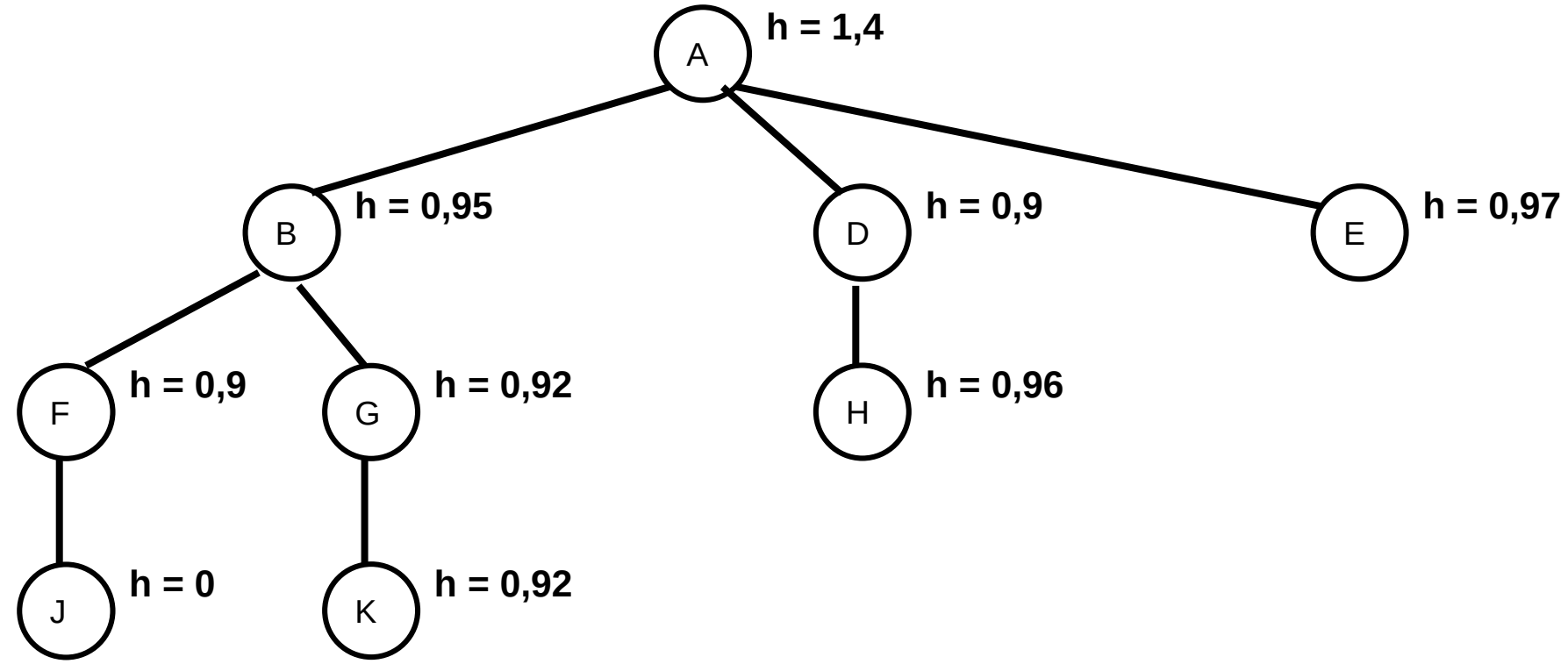
Beam search, X = 2



Beam search, X = 2

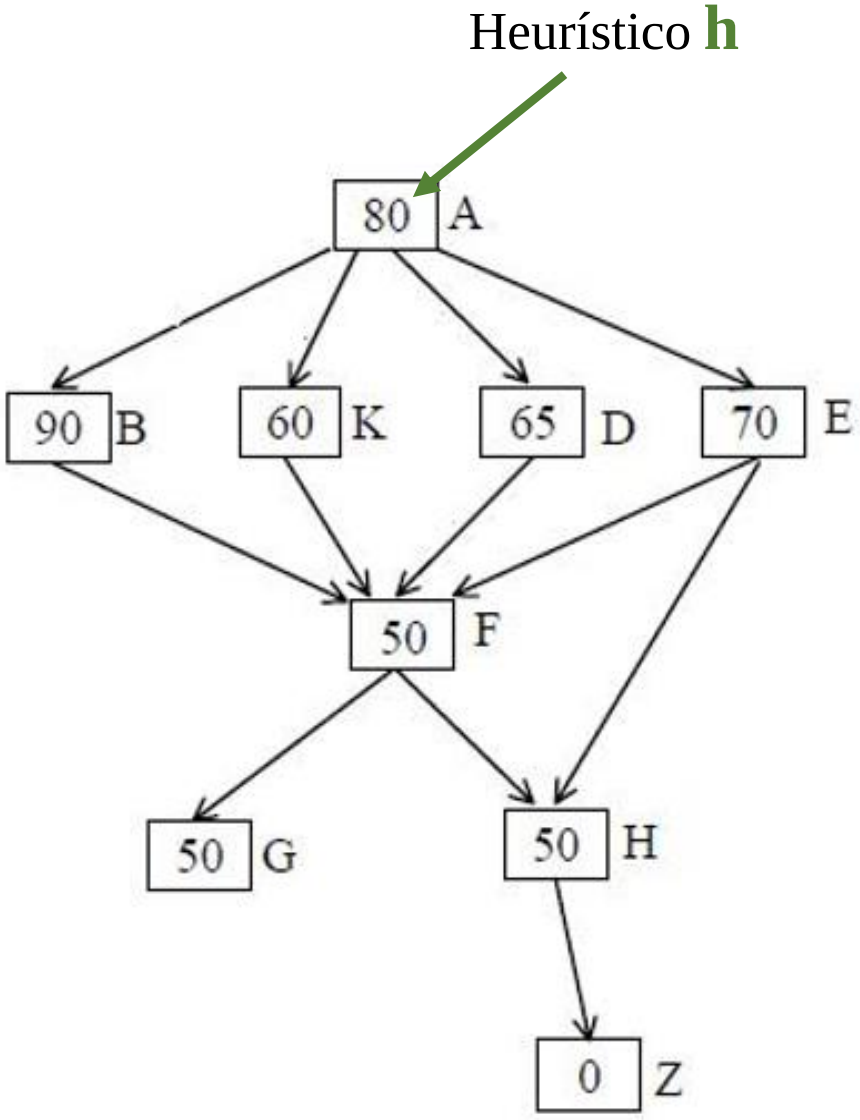


Beam search, X = 2



Solución 1

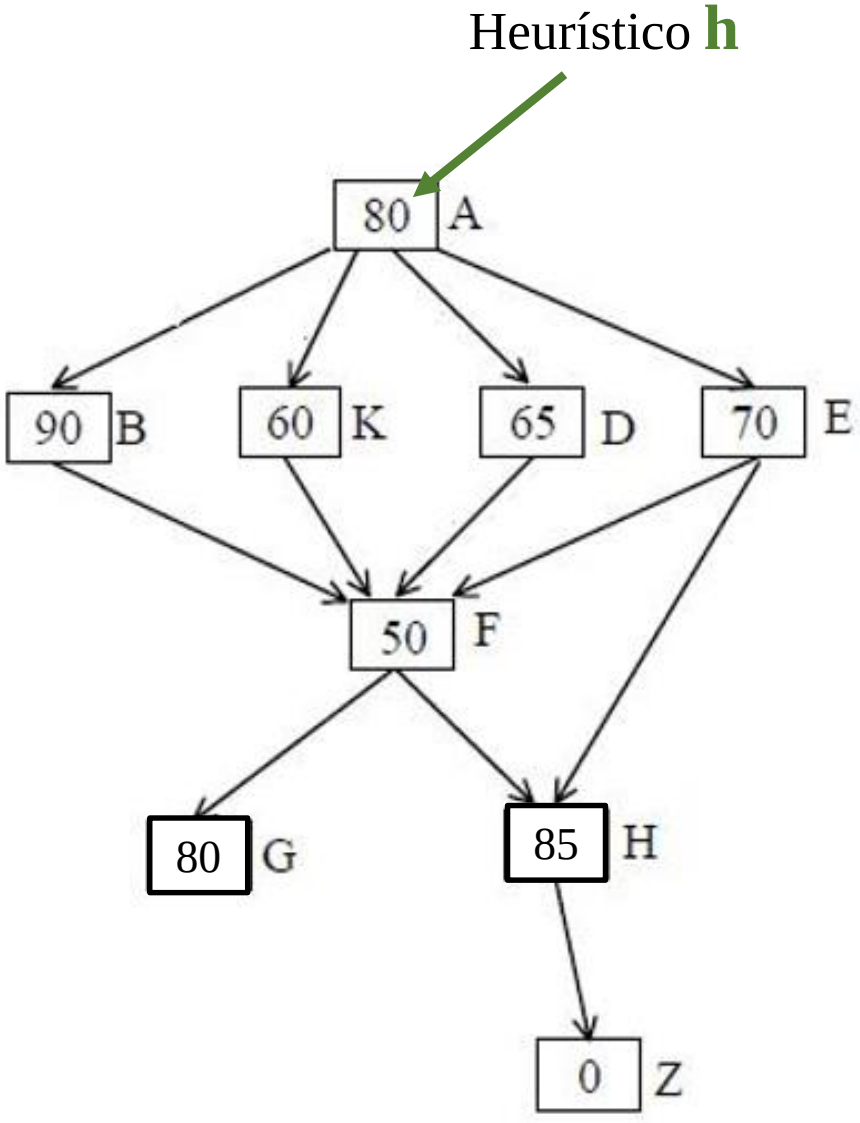
Ejercicio 1



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

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

Ejercicio 2

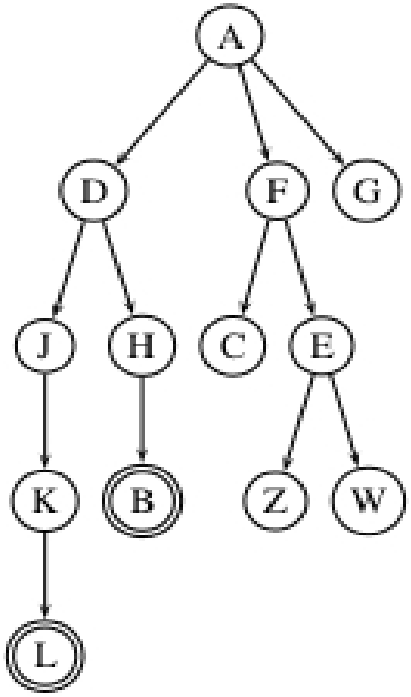


Estado inicial: **A**
Estado final: **Z**

- 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 meta y A es el nodo inicial.

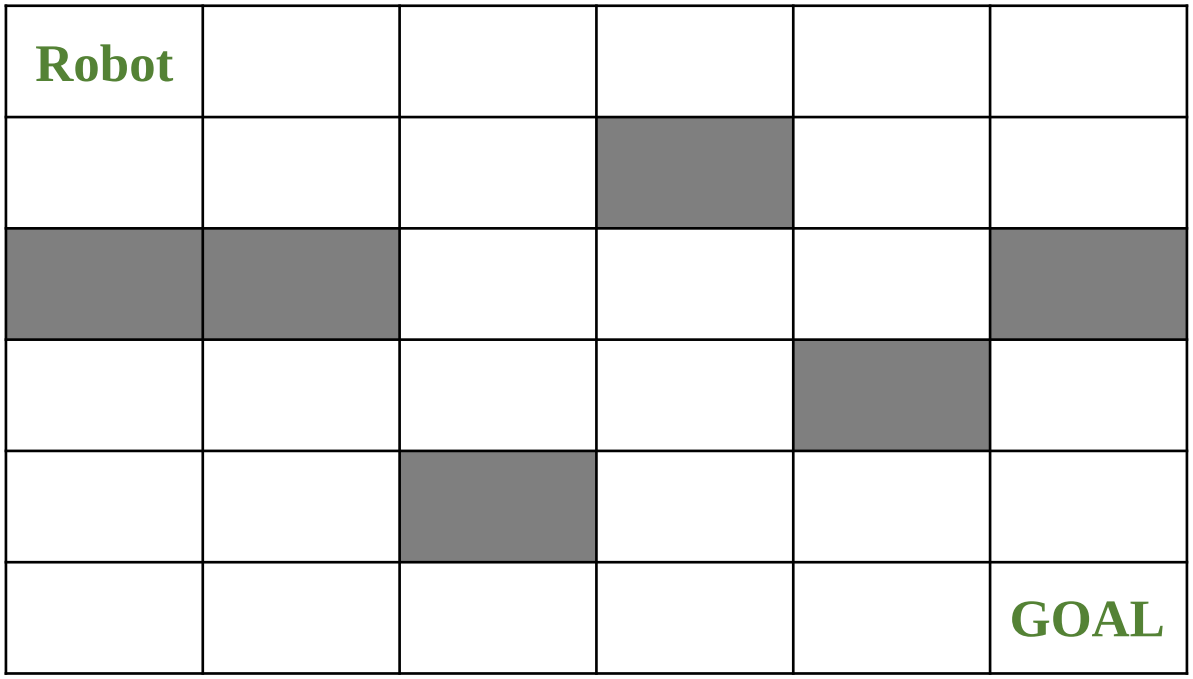


Indica en qué orden se visitarían los nodos, distinguiendo nodos generados de nodos expandidos, para los siguientes algoritmos:

1. Mejor primero (heuristico = orden alfabético)

Ejercicio 4

- 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 4

- 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?