Learning is a treasure that will follow its owner everywhere
— Chinese Proverb

Inteligencia Artificial

Búsqueda informada

Programa Ciencias de la Computación e Inteligencia Artificial

Escuela de Ciencias Exactas e Ingeniería

Universidad Sergio Arboleda

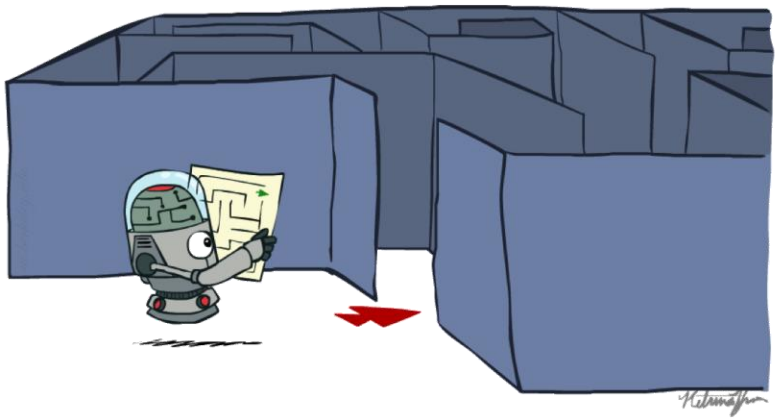
Contenido

- Revisión sesión anterior
- Heurísticas
- Algoritmos voraces
- Algoritmo A*
- Heurísticas admisibles y consistentes

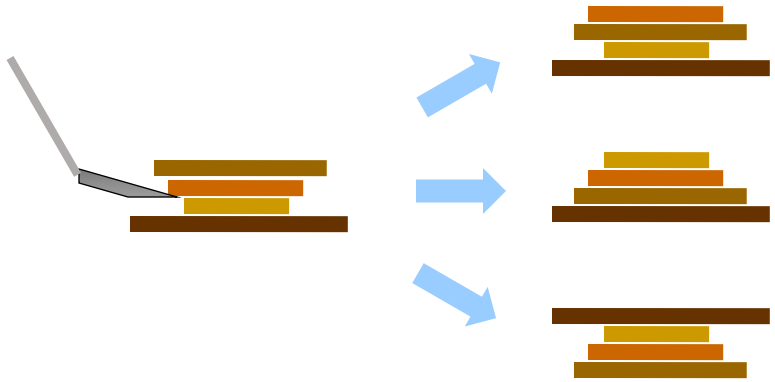
Revisión de la sesión anterior

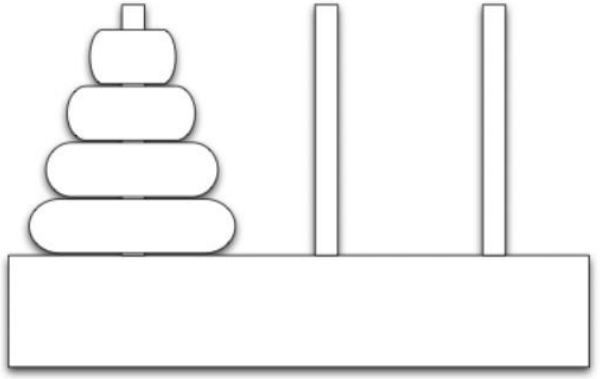
Búsqueda no informada

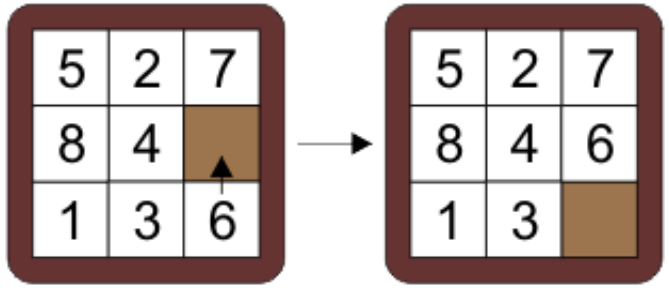
- **Problemas de búsqueda**
  - Estados (configuraciones del mundo)
  - Acciones y costos
  - Función de transición
  - Estados inicial y objetivo
  - Función de evaluación
- **Búsqueda en grafos**
  - Nodos: Representan un plan para alcanzar un estado
  - Los planes tienen un costo (suma del costo de las acciones)
- **Algoritmo de búsqueda**
  - Construye un árbol de forma sistemática
  - Selecciona los nodos de la frontera según una estrategia(FIFO, LIFO, PRIORIDAD)
  - ¿Óptimos y completos?



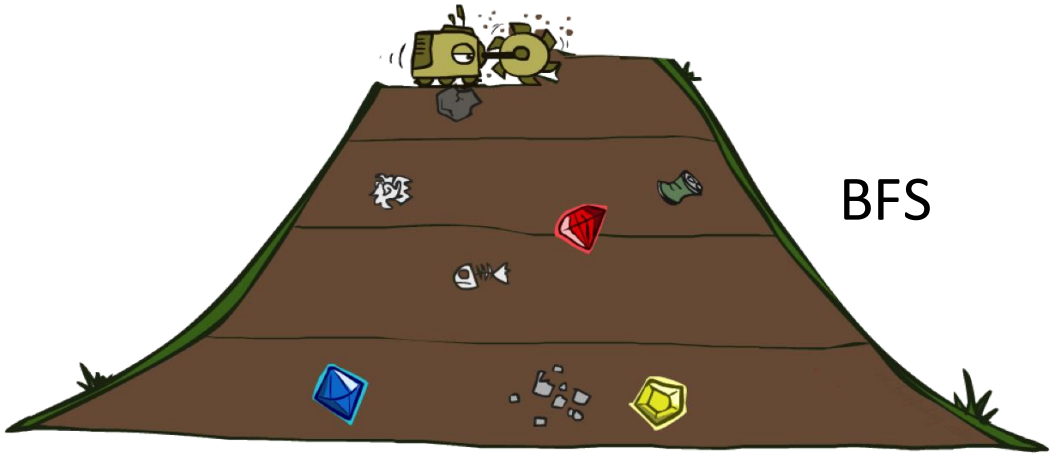
Problemas de búsqueda



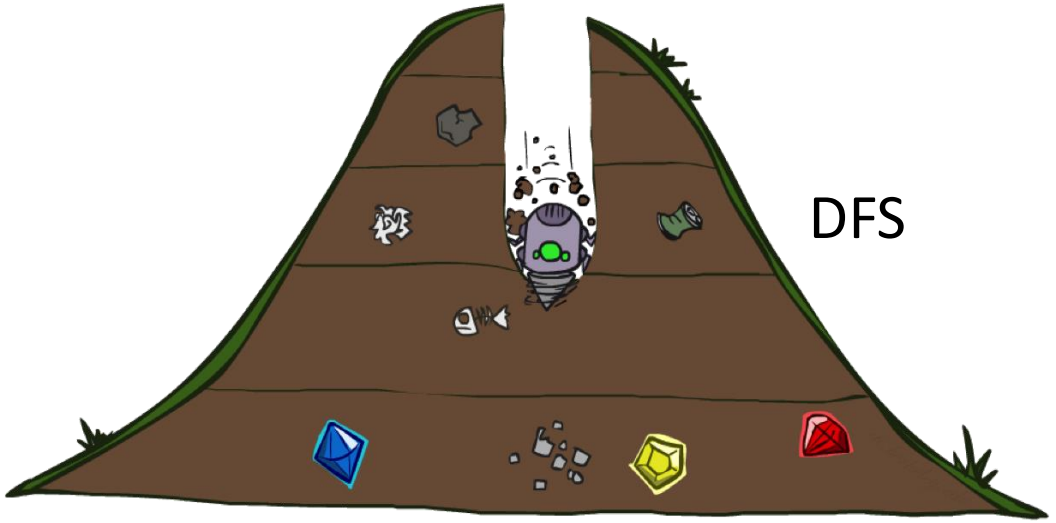




Búsqueda no informada



BFS

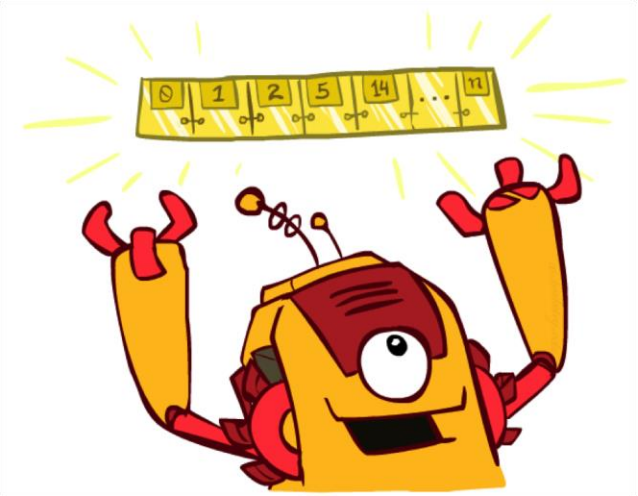


DFS

Búsqueda no informada

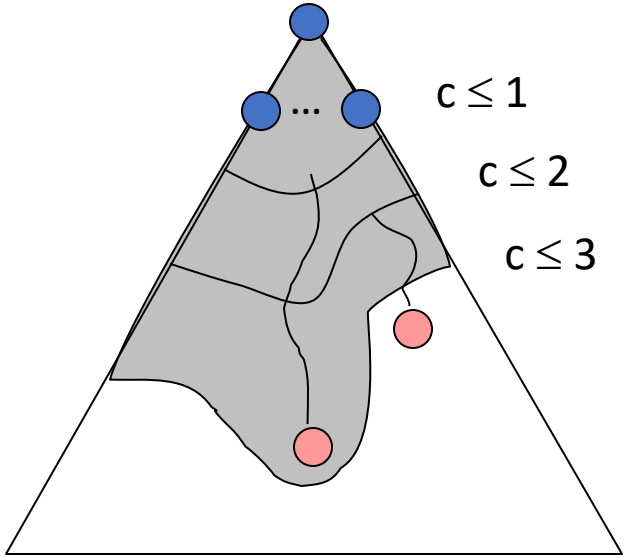
Los algoritmos **BFS**, **DFS** son exactamente iguales excepto por la forma como se extraen los nodos de la frontera

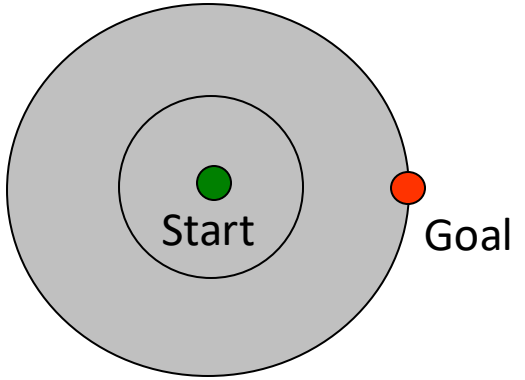
- En ambos casos la frontera puede representarse como cola de prioridad
- Deberíamos tener una única implementación del algoritmo de búsqueda que tenga como parámetro un objeto para representar la frontera



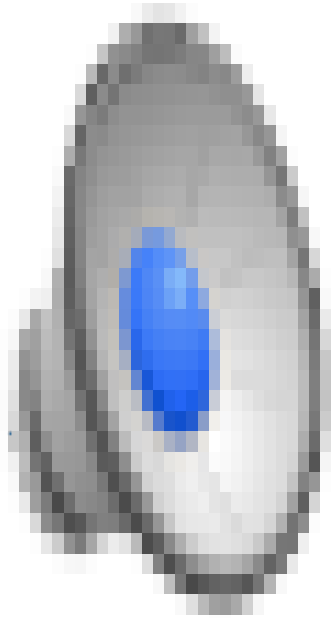
Búsqueda de costo uniforme (UCS)

- **Estrategia**: expandir el nodo con el menor costo acumulado
- **Ventajas**: UCS es completo y óptimo
- **Desventajas**:
  - Explora opciones en todas las direcciones
  - No tiene ninguna información acerca de la ubicación del objetivo

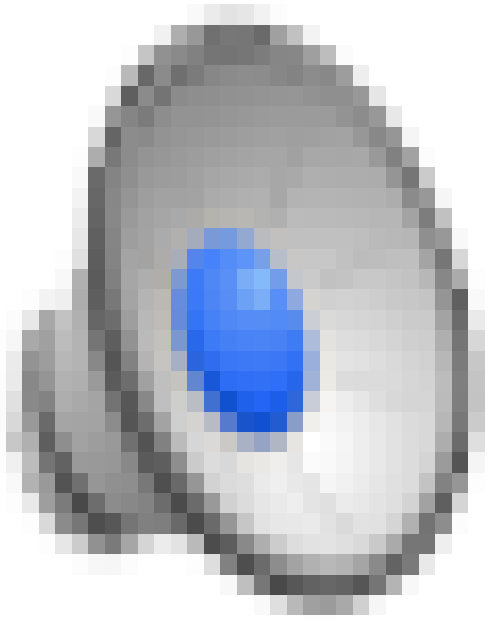




Búsqueda de costo uniforme (UCS)



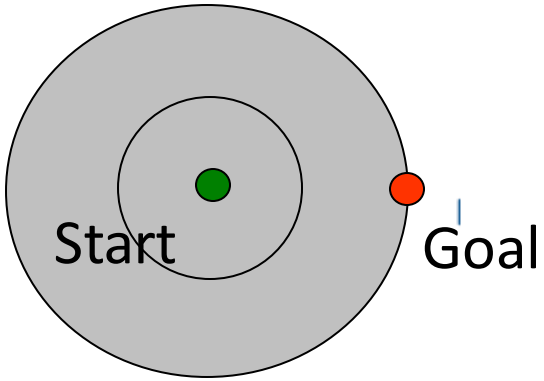
Búsqueda de costo uniforme (UCS)



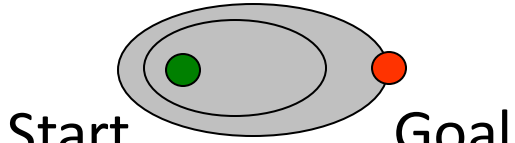
Búsqueda de costo uniforme (UCS)

¿Qué sería deseable en el caso de UCS?

- Qué la búsqueda estuviera dirigida hacia el lugar de la solución y no en todas las direcciones



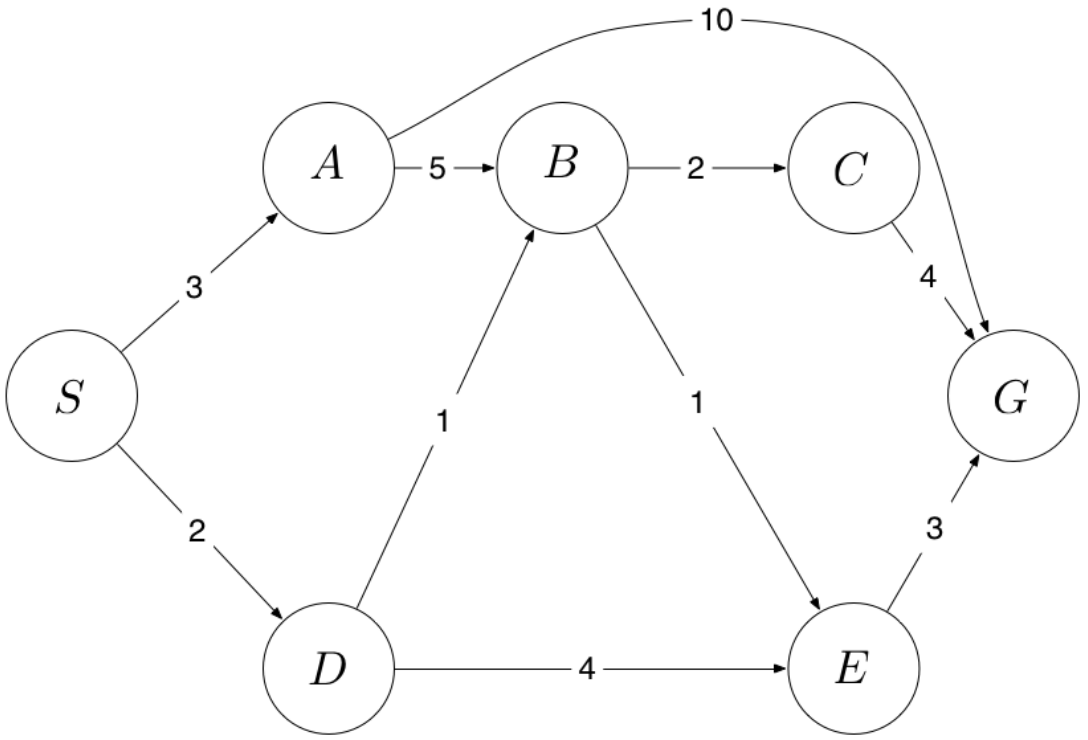
Búsqueda no informada



Búsqueda informada

Ejercicio 1

- Encuentre la solución para el siguiente problema de búsqueda utilizando BFS, DFS, UCS
- Suponga que los empates se rompen alfabéticamente
  - $S \rightarrow X \rightarrow A$ será expandido antes $S \rightarrow X \rightarrow B$
  - $S \rightarrow A \rightarrow Z$ será expandido antes $S \rightarrow B \rightarrow A$
- Para cada caso encuentre la solución y la secuencia de nodos expandidos



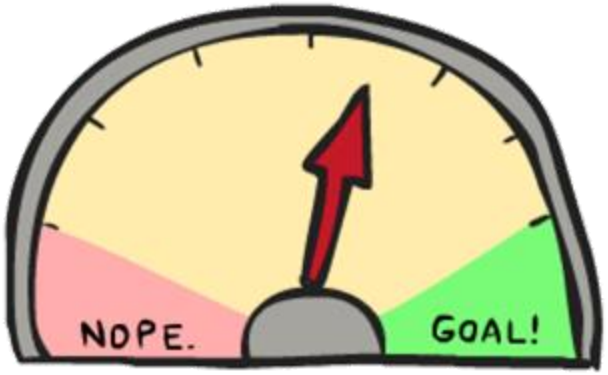
Búsqueda informada

Ideas iniciales

- Las estrategias de búsqueda no informada dependen únicamente del estado inicial, estado objetivo y las acciones.
- No utilizan ningún tipo de información sobre la naturaleza del problema que haga la búsqueda más eficiente
- Si tenemos información sobre en que dirección buscar, podemos incluirla como parte de nuestra estrategia de búsqueda

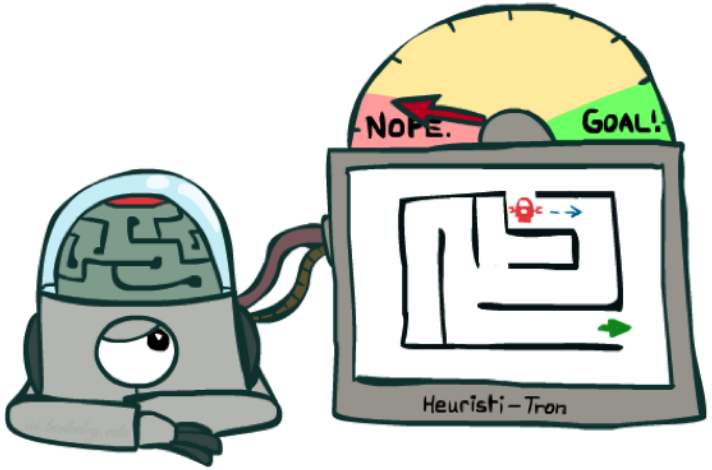
Heurísticas

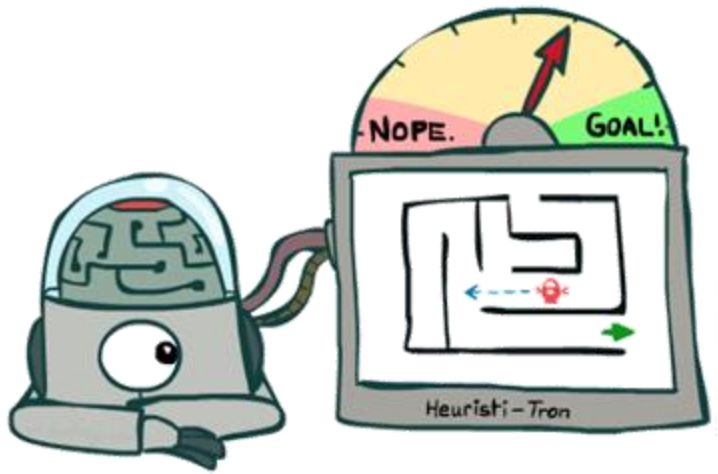
- Las heurísticas son criterios, métodos o principios para decidir cuál, de entre varias acciones, promete ser la mejor para alcanzar un objetivo
- El uso de heurísticas nos permite guiar nuestra búsqueda, permitiéndonos obtener una solución más rápidamente
- Las heurísticas están relacionadas con la naturaleza del problema (Basadas en la experiencia). Son funciones *ad hoc* diseñadas para un problema



Heurísticas

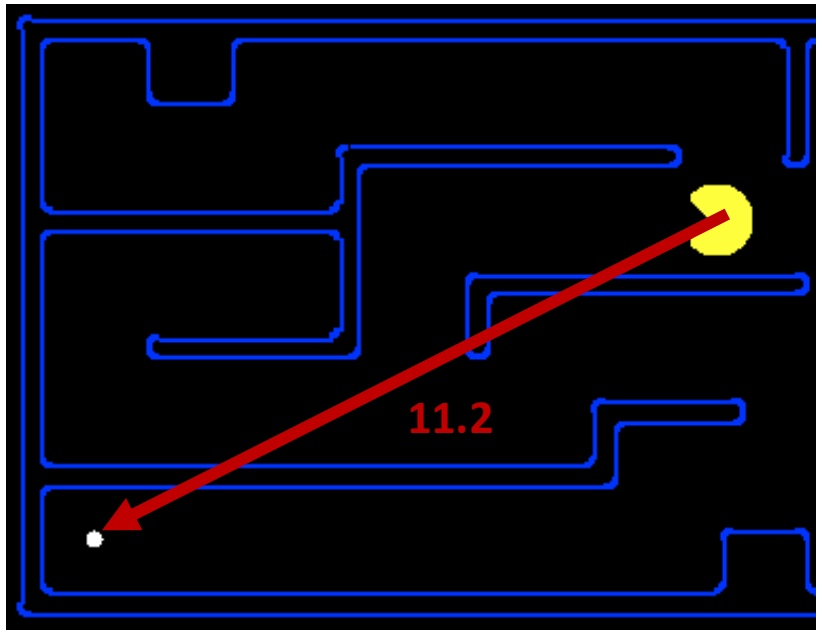
- Formalmente podemos definir una heurística como una función $h : S \rightarrow \mathbb{R}^+$ que asigna a cada estado $s \in S$ un estimado de la distancia entre s y el estado objetivo.
- Mientras más pequeño el valor de $h(s)$, más cercano estará s del estado objetivo. Si s es el estado objetivo, entonces $h(s) = 0$



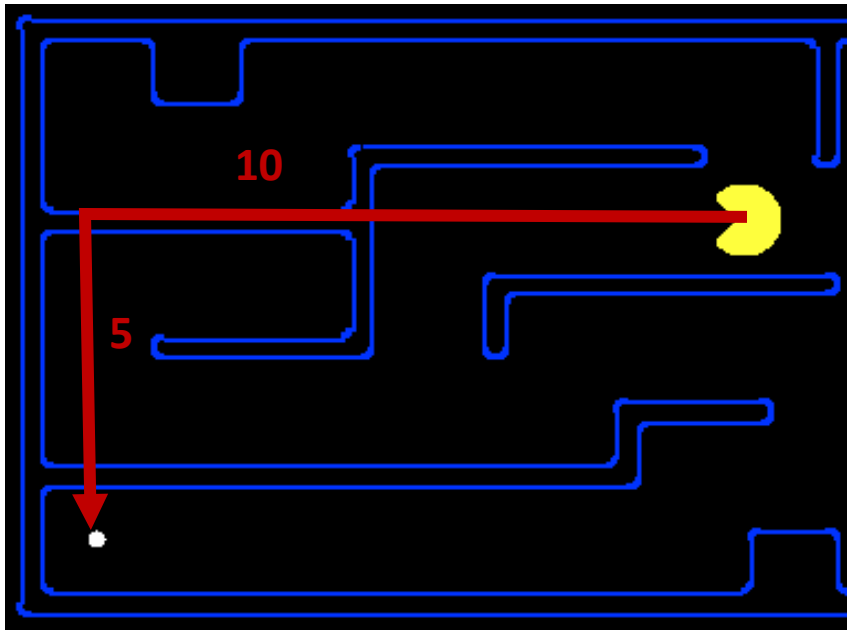


Heurísticas: ejemplos

La función heurística h puede ser definida de distintas formas según la naturaleza del problema (o según sea conveniente):



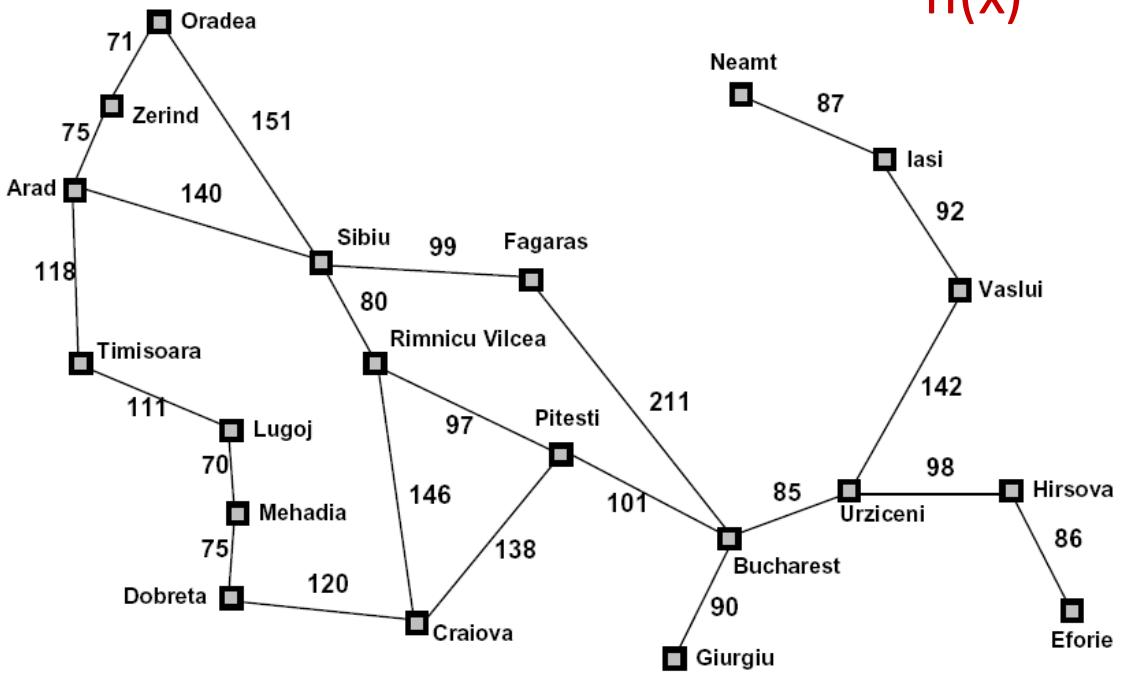
h(x) : Distancia Euclidiana



h(x) : Distancia de Manhattan

Heurísticas: ejemplos

Heurística: la distancia representada por una línea recta desde una ciudad hasta Bucharest

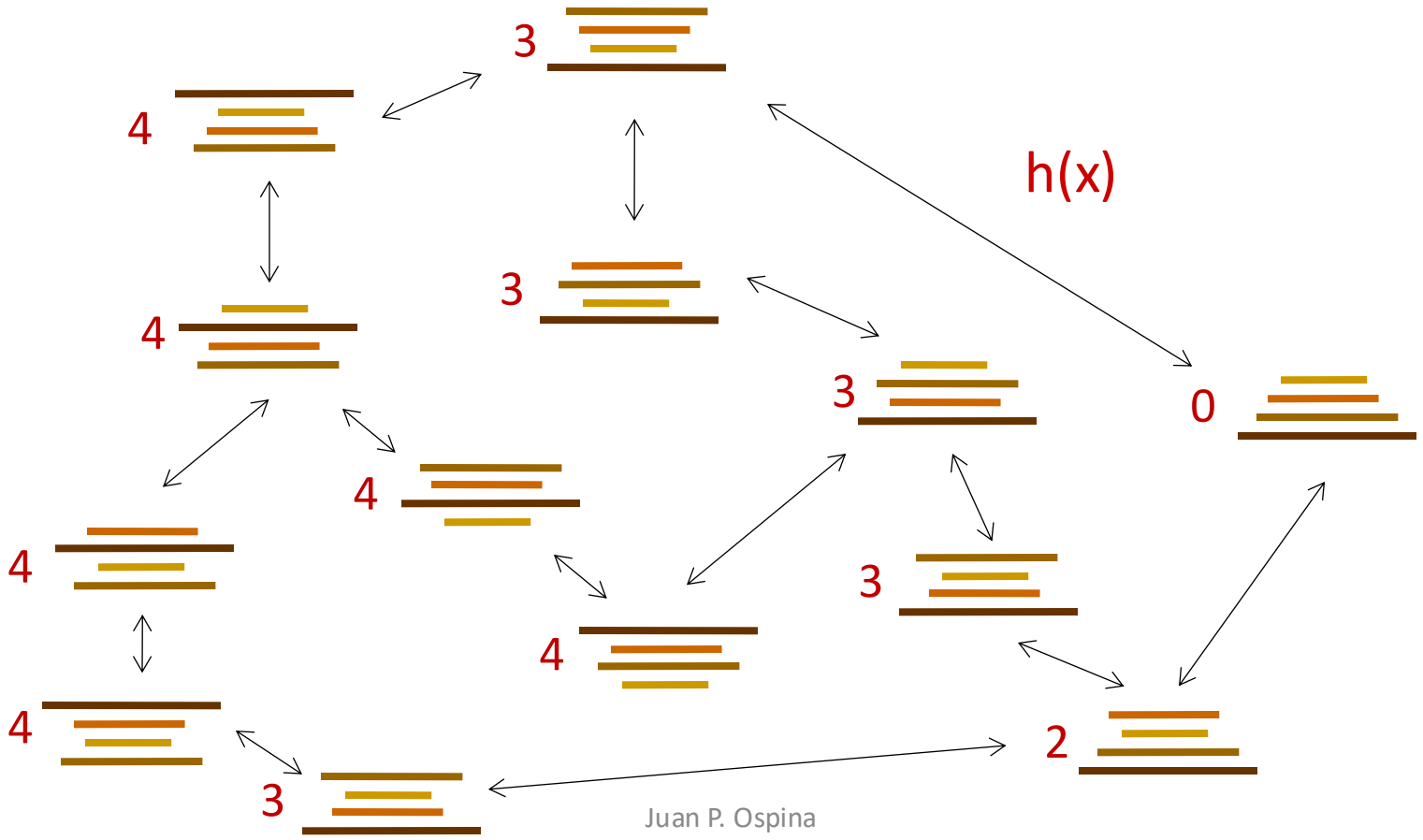


h(x)

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

Heurísticas: ejemplos

Heurística: el número del panqueque más grande que esté fuera de su posición



Heurísticas: ejemplos

initial state:

8		7
6	5	4
3	2	1

goal state:

1	2	3
4	5	6
7	8	

- Número de piezas mal ubicadas:

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

- La suma de las distancias de Manhattan entre todas las piezas y sus posiciones finales

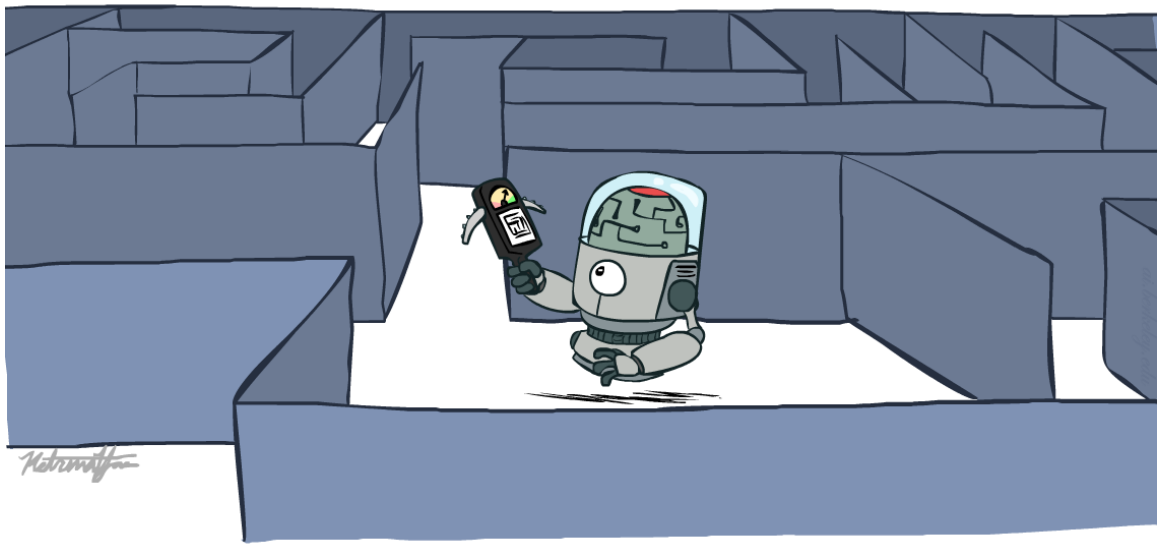
$$h_2\left(\begin{array}{|c|c|c|}\hline 8 & & 7 \\ \hline 6 & 5 & 4 \\ \hline 3 & 2 & 1 \\ \hline\end{array}\right) = 21$$

- Note que $h_2(s) \geq h_1(s)$

Búsqueda informada

Dependiendo de la forma como incorporemos la función heurística a nuestro método de búsqueda podremos definir dos algoritmos:

- Búsqueda Voraz
- Algoritmo A*



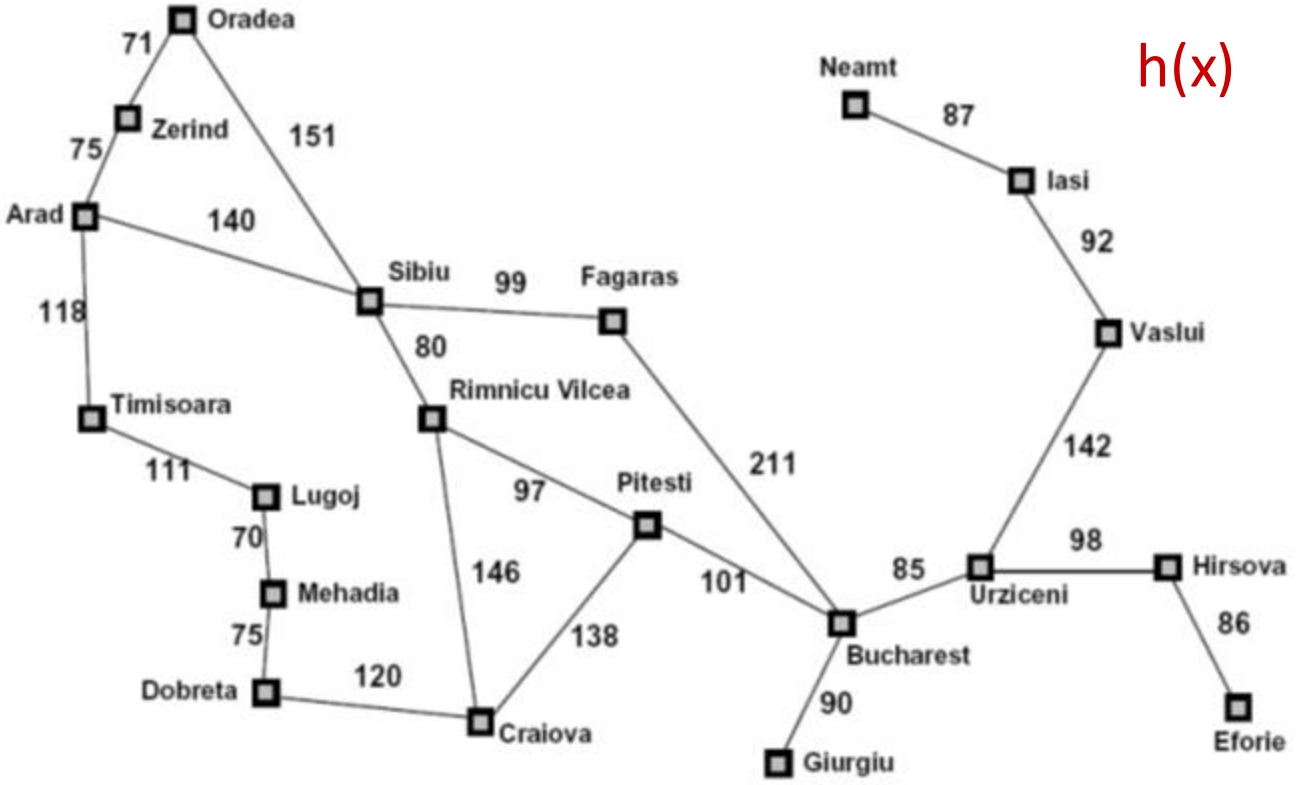
Búsqueda Voraz

Búsqueda Voraz

Cuando utilizamos directamente los valores proporcionados por nuestra función heurística para expandir los nodos estamos utilizando un método voraz **(greedy)**

Búsqueda Voraz

Heurística: la distancia representada por una línea recta desde una ciudad hasta Bucharest

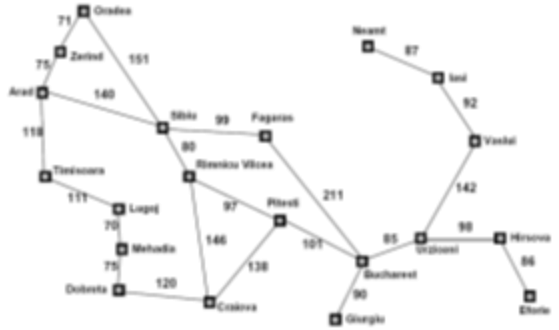


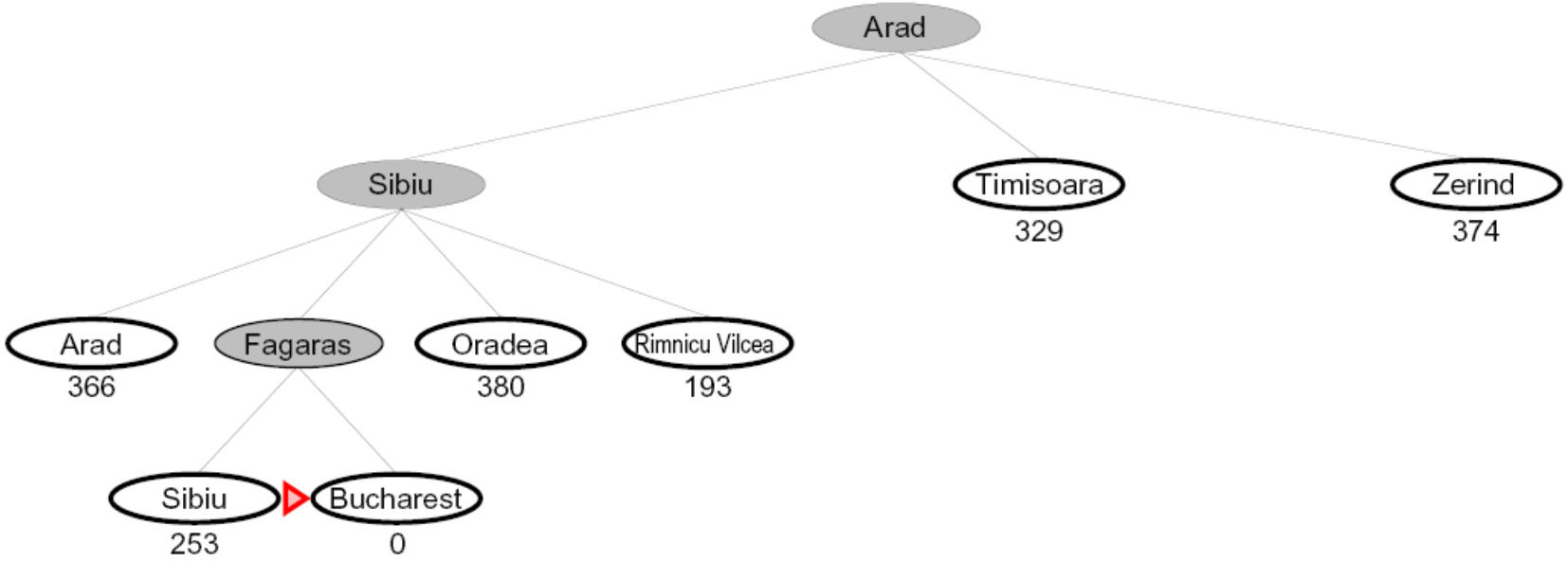
h(x)

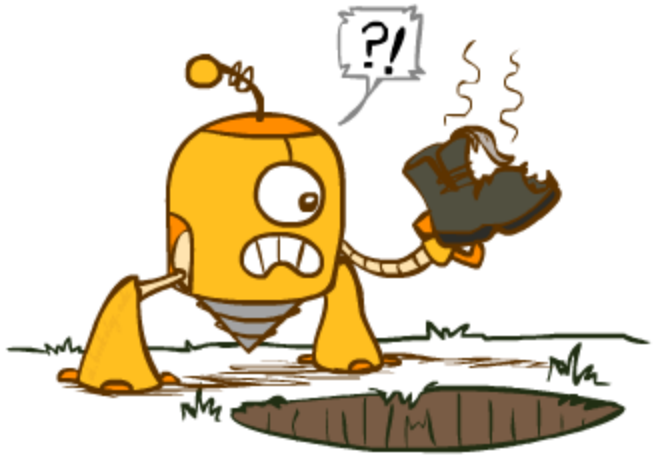
Straight−line distance to Bucharest	
Arad	366
Bucharest	0
Craiova	160
Dobreta	242
Eforie	161
Fagaras	178
Giurgiu	77
Hirsova	151
Iasi	226
Lugoj	244
Mehadia	241
Neamt	234
Oradea	380
Pitesti	98
Rimnicu Vilcea	193
Sibiu	253
Timisoara	329
Urziceni	80
Vaslui	199
Zerind	374

Búsqueda Voraz

Expandimos el nodo que siempre parece estar más cerca

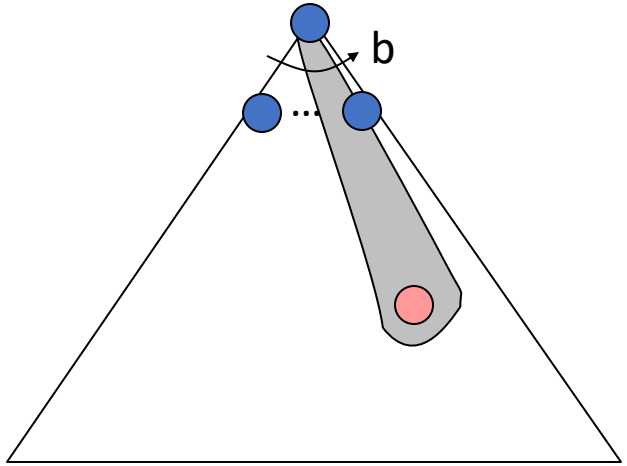


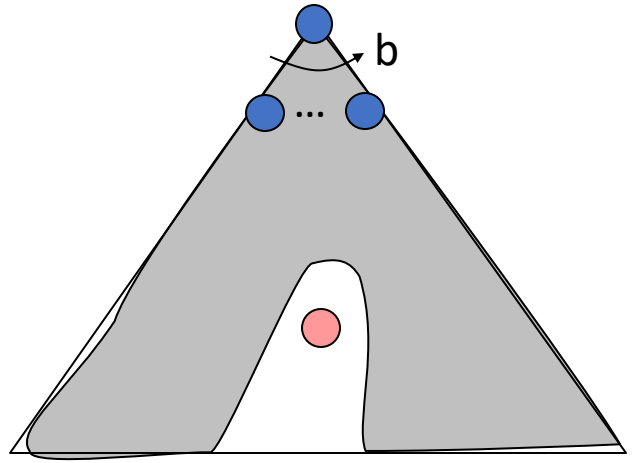




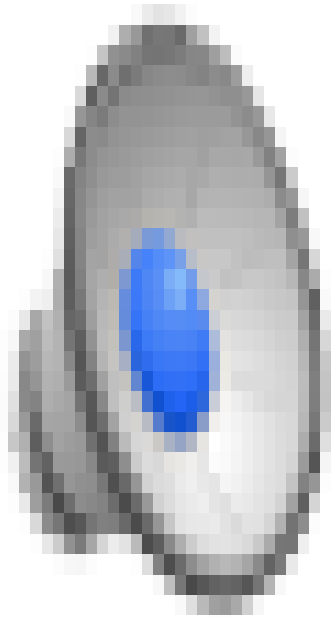
Búsqueda Voraz

- **Estrategia:** expandimos el nodo con el mejor valor heurístico
- **Expectativa:** saltar al nodo con el mejor valor heurístico nos llevará más rápido a la meta
- **Realidad:** puede terminar comportándose como una DFS
- **Incompleto:** a menos que se utilice una lista para referenciar los estados visitados
- **No óptimo:** No es posible garantizar que el costo de la solución

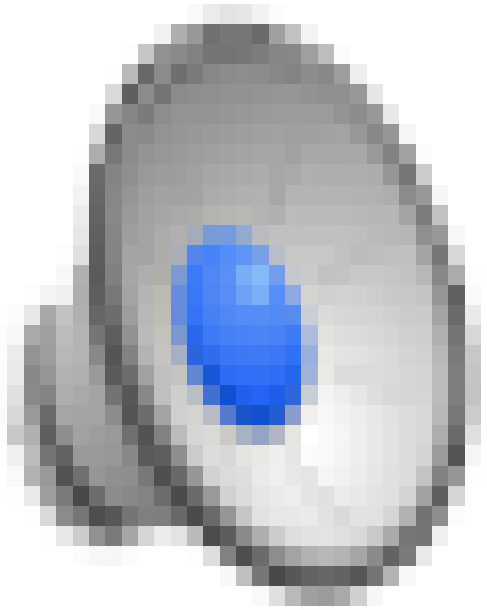




Búsqueda Voraz

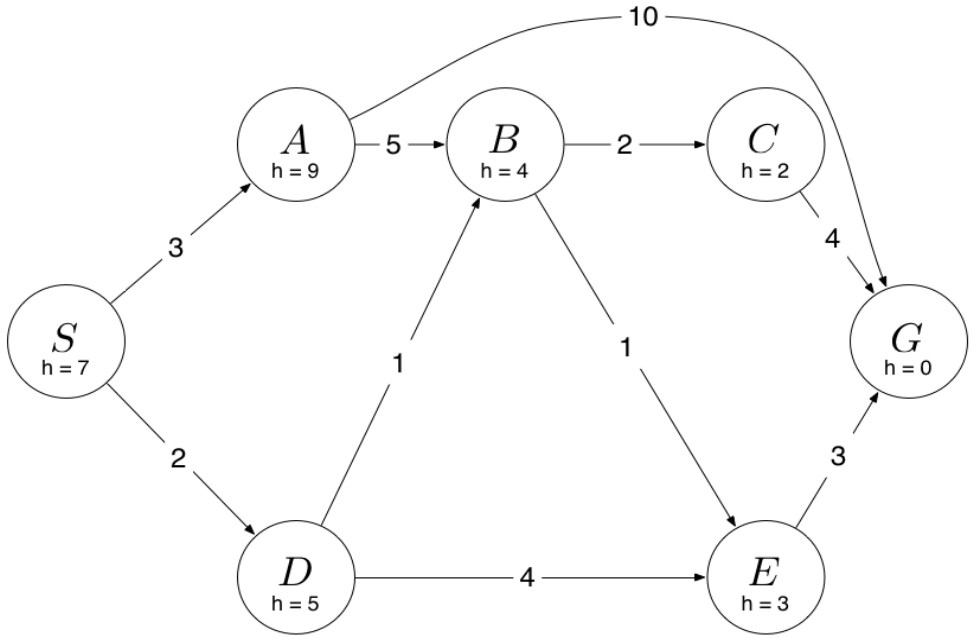


Búsqueda Voraz



Ejercicio 2

- Encuentre la solución para el siguiente problema utilizando búsqueda Voraz
- Suponga que los empates se rompen alfabéticamente
  - $S \rightarrow X \rightarrow A$ será expandido antes $S \rightarrow X \rightarrow B$
  - $S \rightarrow A \rightarrow Z$ será expandido antes $S \rightarrow B \rightarrow A$
- Para cada caso encuentre la solución y la secuencia de nodos expandidos



Algoritmo A*

Algoritmo A*

UCS

Greedy

Algoritmo A*

UCS

Greedy

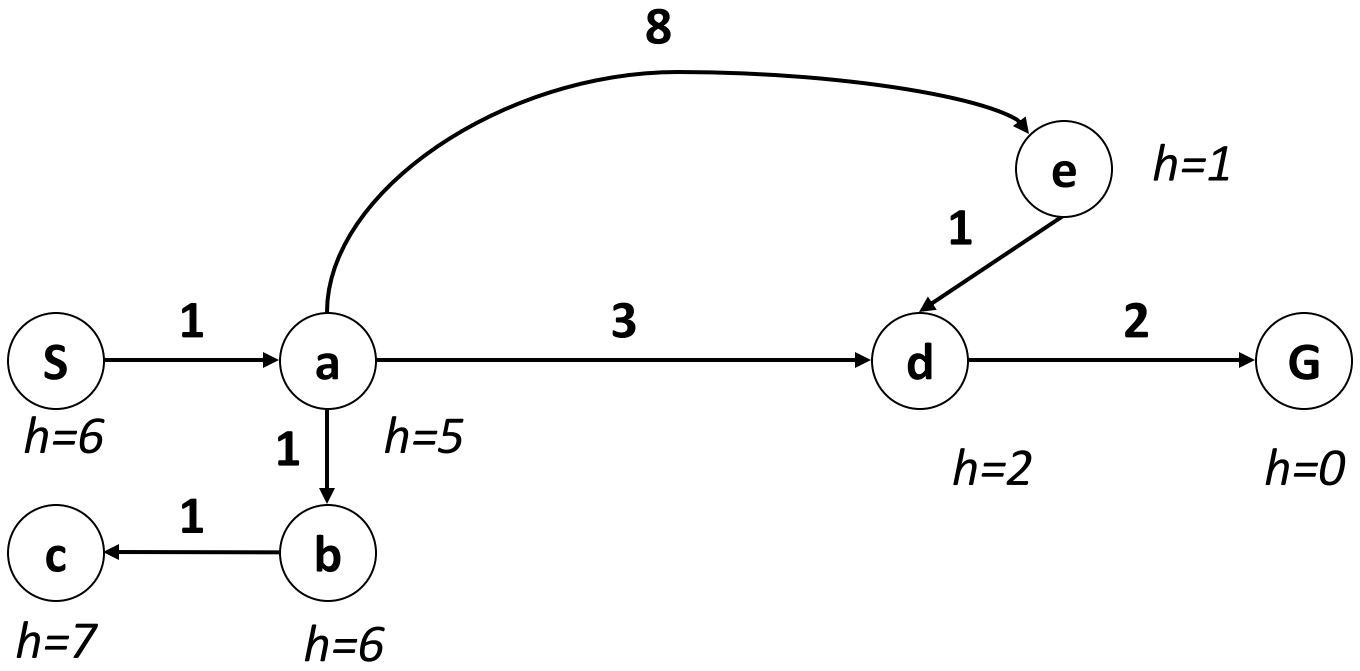
A*

Algoritmo A*

- Expandir el nodo $\boldsymbol{n}$ con más probabilidad de estar en la ruta óptima
- Expandir el nodo $\boldsymbol{n}$ con el menor costo $\mathbf{f}(\boldsymbol{n}) = \boldsymbol{g}(\boldsymbol{n}) + \boldsymbol{h}(\boldsymbol{n})$ donde:
  - $\boldsymbol{g}(\boldsymbol{n})$ es el costo real desde la raíz hasta $\boldsymbol{n}$
  - $\boldsymbol{h}(\boldsymbol{n})$ es costo estimado desde $\boldsymbol{n}$ hasta la solución más cercana
- **A* =** utiliza una cola de prioridad ordenada por $\mathbf{f}(\boldsymbol{n}) = \boldsymbol{g}(\boldsymbol{n}) + \boldsymbol{h}(\boldsymbol{n})$

Algoritmo A*

Encuentre la solución para el siguiente problema de búsqueda utilizando UCS, búsqueda voraz y A*

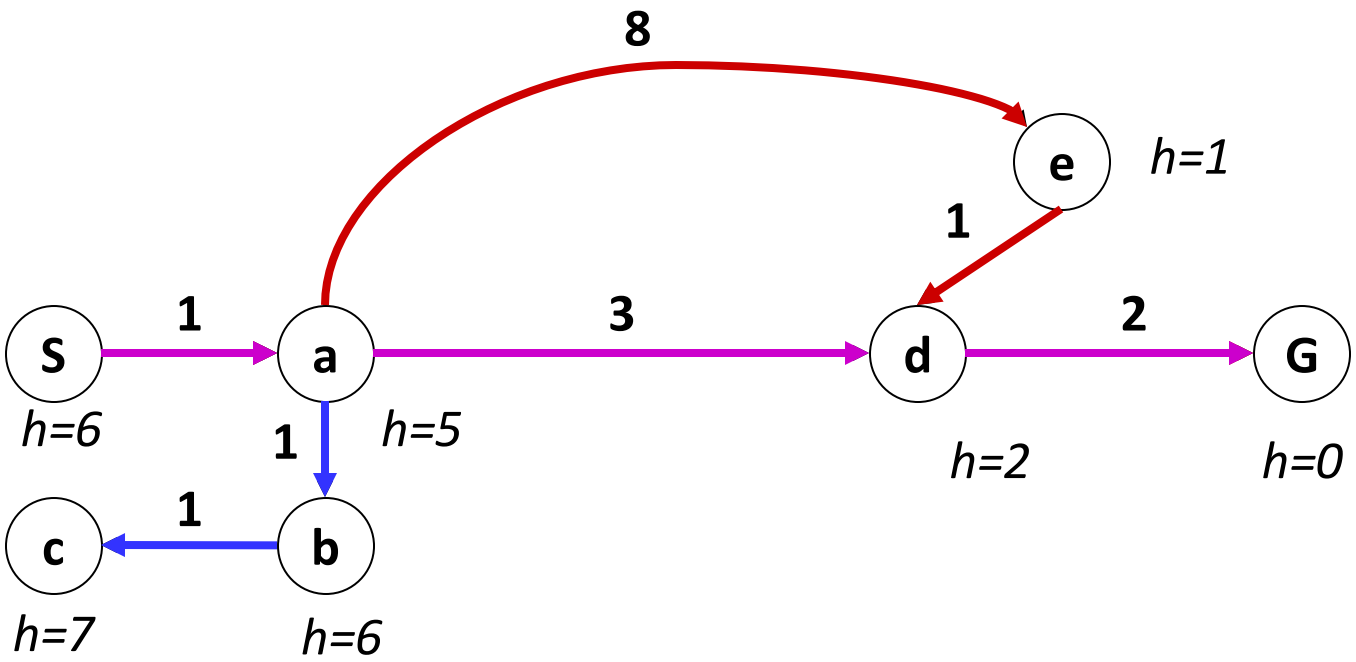


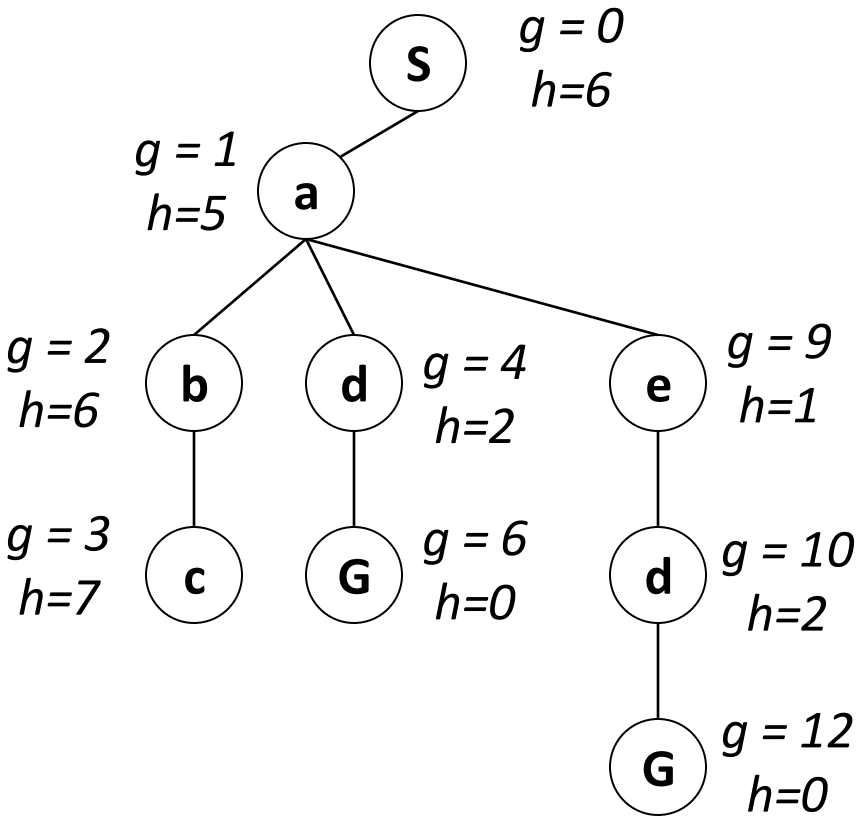
Algoritmo A*

Costo uniforme: expande nodos de acuerdo a $g(n)$

Búsqueda voraz: expande nodos de acuerdo a $\mathrm{h}(n)$

Algoritmo A*: expande nodos de acuerdo $\mathrm{f}(n) = g(n) + h(n)$





Algoritmo A*

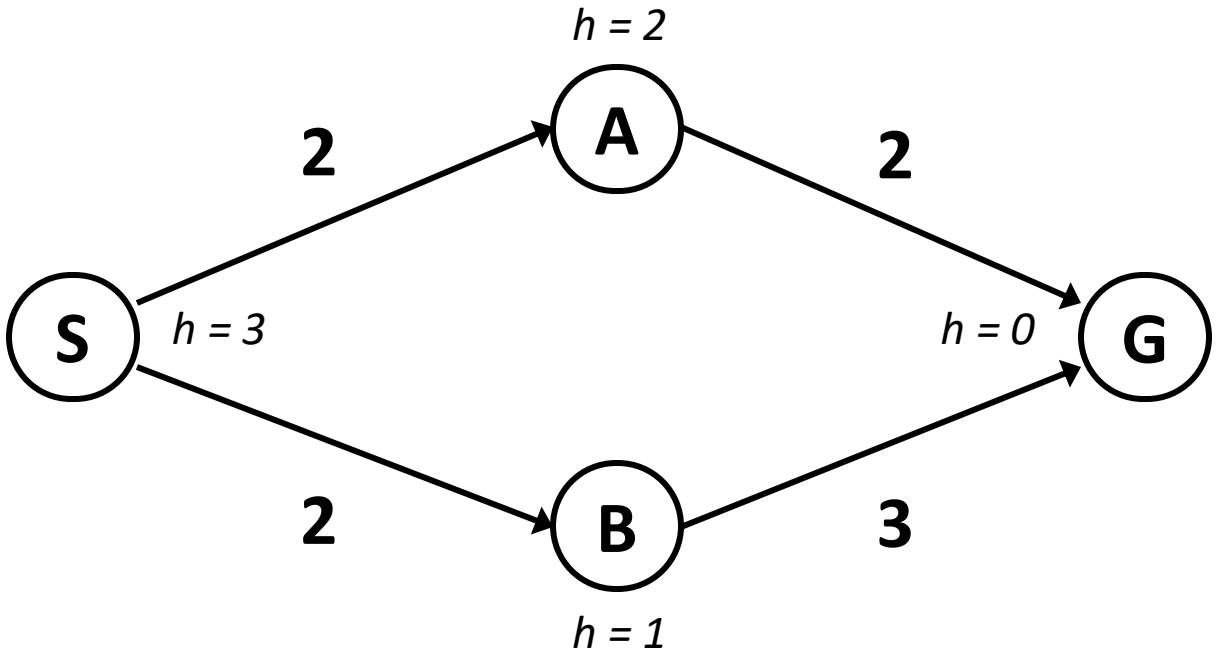
- Combina UCS y búsqueda Voraz
- La búsqueda por costo uniforme solo tiene en cuenta el costo desde la raíz al nodo actual para expandir un nodo: $g(n)$
- La búsqueda voraz solo considera el costo estimado del nodo actual al nodo objetivo para expandir un nodo: $\mathrm{h}(n)$
- El algoritmo A* utiliza la suma de ambos $\mathrm{f}(\mathrm{n}) = g(n) + h(n)$

Algoritmo A*

Algoritmo A*

Cuestiones de implementación

- ¿Cuándo debemos terminar la búsqueda?

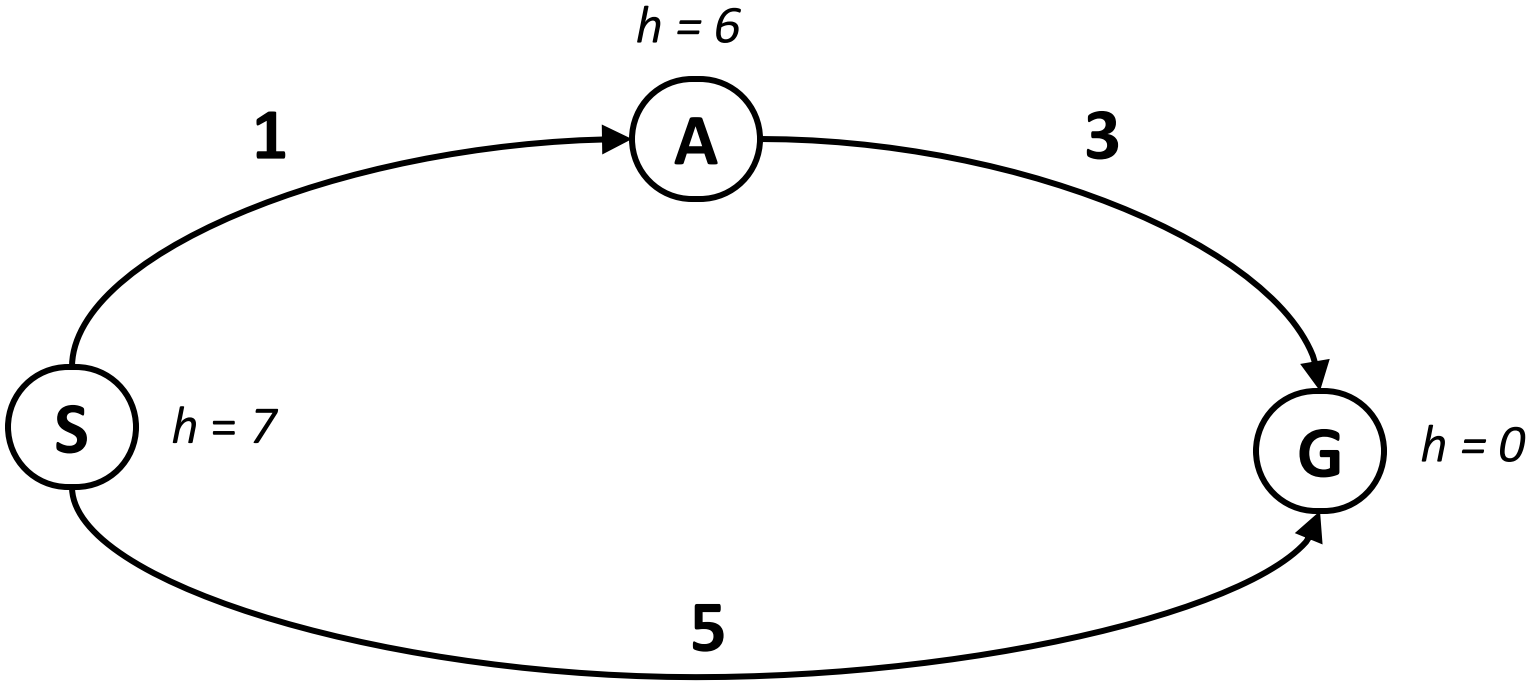


Cuando se expande el nodo y no cuando se agrega a la frontera

Algoritmo A*

Cuestiones de implementación

- ¿Se encuentra siempre la solución óptima?



Depende de la heurística utilizada

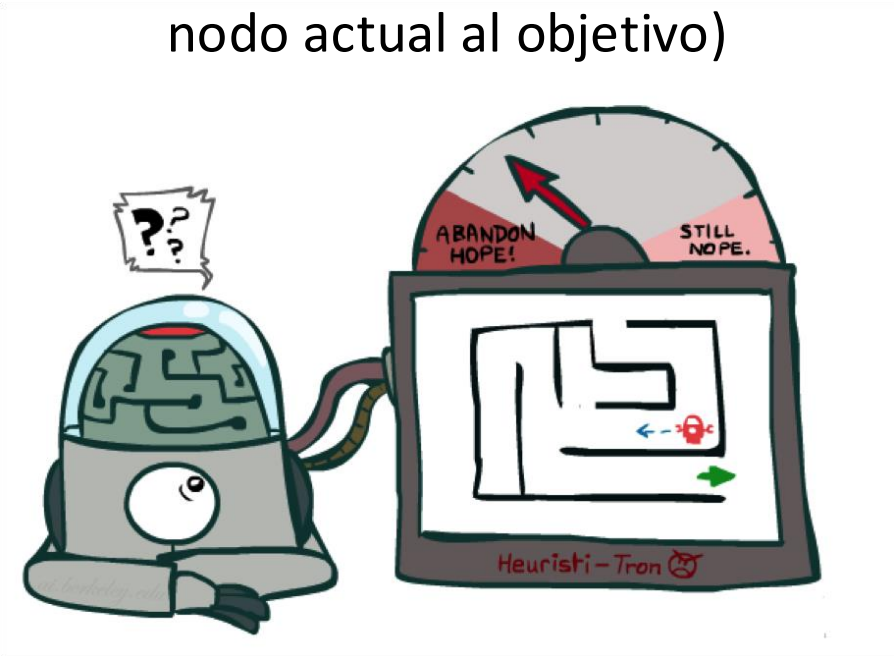
Algoritmo A*

Cuestiones de implementación

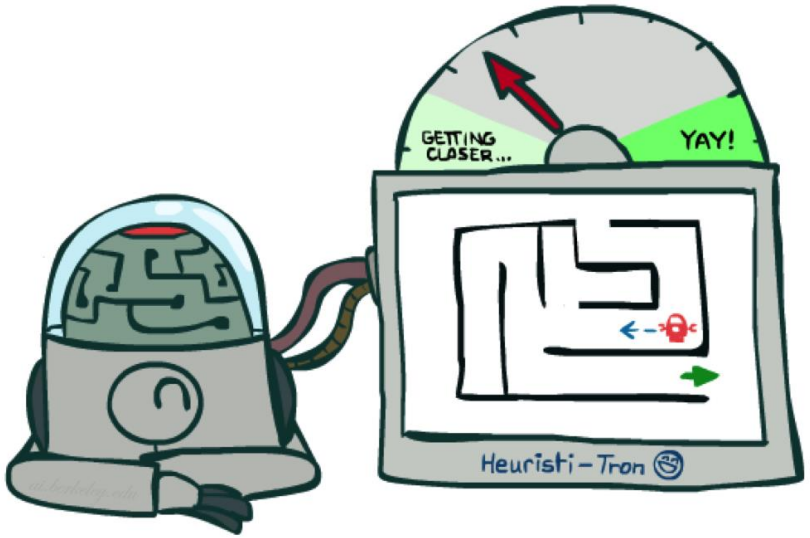
- ¿Cómo sabemos que una heurística es una buena heurística?

Heurísticas admisibles

Una heurística es inadmisible (pesimista) si atrapa buenos planes en la frontera. (sobreestima el costo del nodo actual al objetivo)



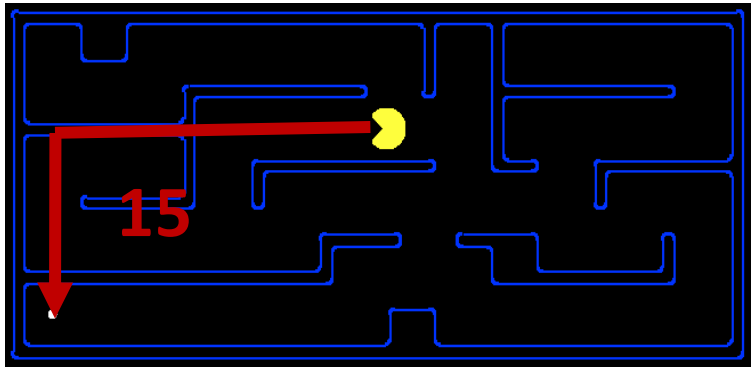
Una heurística es admisible **(optimista)** desacelera la elección de malos planes, pero nunca sobreestima el costo real de un camino

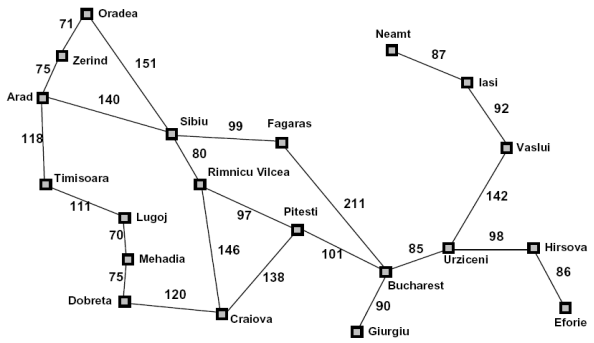


Heurísticas admisibles

Una heurística h es admisible si: $(\forall\ s\ \in S).\ 0 \leq h(n) \leq h^*(n)$

Donde $h^*(n)$ es el costo verdadero de la solución desde el n hasta el nodo objetivo





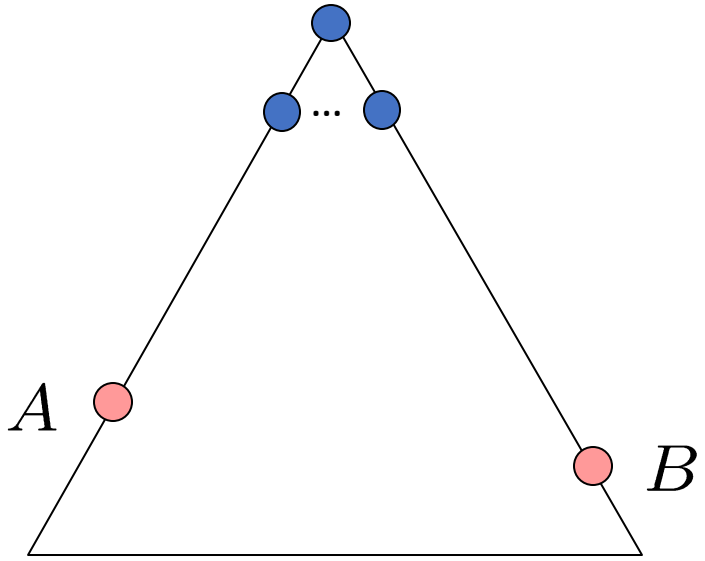
Straight−line distance to Bucharest	
Arad	366
Bucharest	0
Craiova	160
Dobreta	242
Eforie	161
Fagaras	178
Giurgiu	77
Hirsova	151
Iasi	226
Lugoj	244
Mehadia	241
Neamt	234
Oradea	380
Pitesti	98
Rimnicu Vilcea	193
Sibiu	253
Timisoara	329
Urziceni	80
Vaslui	199
Zerind	374

4

Encontrar heurísticas admisibles y fáciles de implementar es fundamental para el éxito del algoritmo de búsqueda

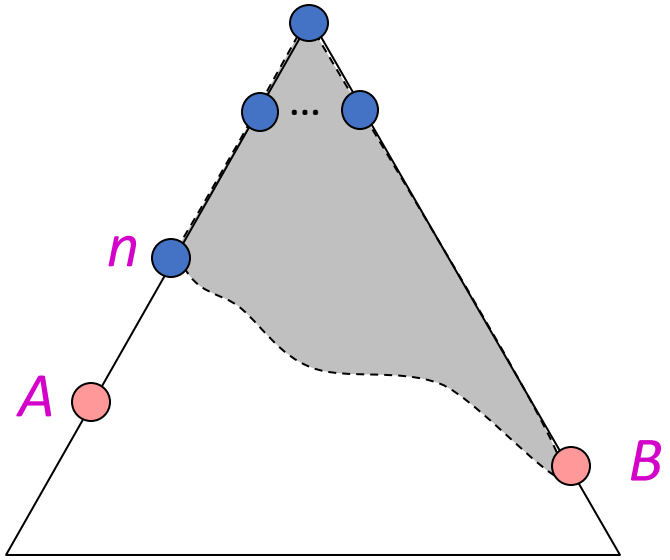
Optimalidad A*

- Para garantizar la optimalidad de A* debemos probar que el algoritmo expandirá A antes que B
- Asumimos que:
  - **A** es la solución óptima
  - **B** es una solución subóptima
  - **h** es admisible
- Debemos probar qué:
  - **A** será expandido antes que **B**



Optimalidad A*

- **Podemos razonar de la siguiente forma:**
  - Supongamos que existe un nodo $\boldsymbol{n}$ que se encuentra en la frontera
  - Probar que $\boldsymbol{f(n)}$ es menor o igual $\boldsymbol{f(A)}$
  - Probar que $\boldsymbol{f(A)}$ es menor o igual $\boldsymbol{f(B)}$
  - Probar que $\boldsymbol{n}$ se expande antes que $\boldsymbol{B}$

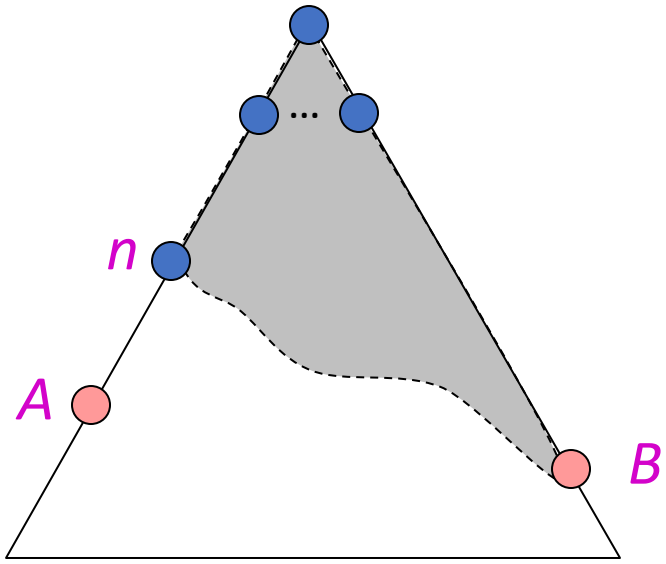


Optimalidad A*

A será expandido antes que **B**

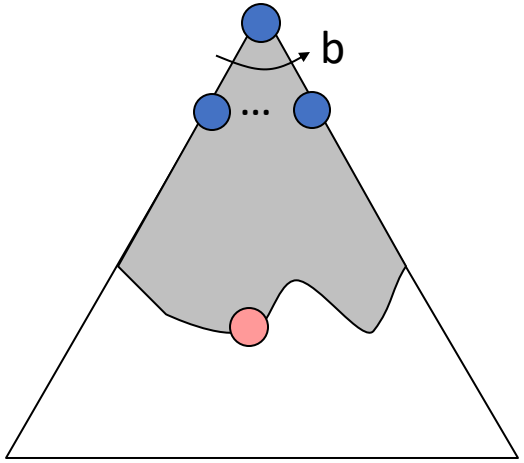
Demostración:

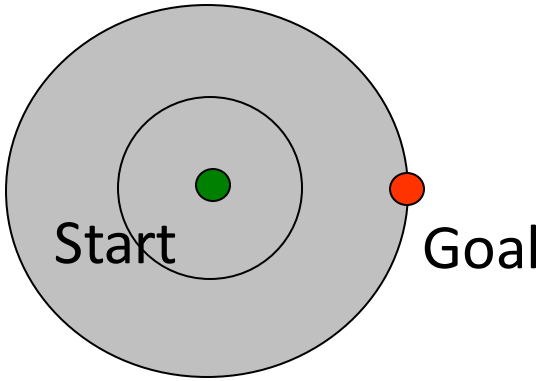
- Si **B** está en la frontera de búsqueda.
- Algún ancestro ***n*** de **A** debe estar en la frontera (puede ser el propio **A**)
- n se expandirá antes que B, ya que: $f(n) \leq f(a) < f(b)$
  - $f(n) \leq f(A)$. $\quad f(n) = g(n) + h(n) \leq g(A) = f(A)$
  - $f(A) \leq f(B)$. $\quad g(A) < g(B) \rightarrow f(A) < f(B)$
- Todos los ancestros de **A** se expanden antes que **B**
- **A** se expande antes que **B**



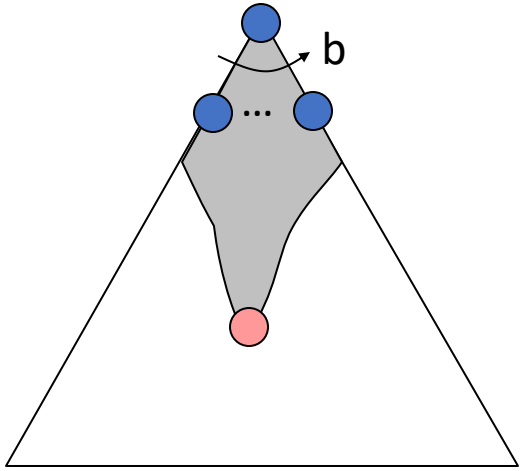
Algoritmo A* vs UCS

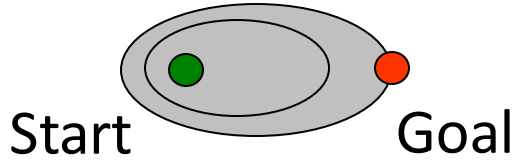
Costo uniforme



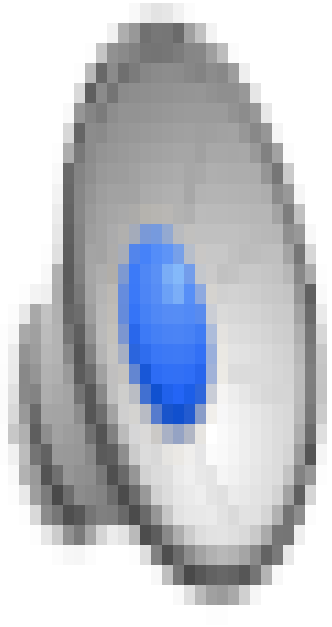


A*

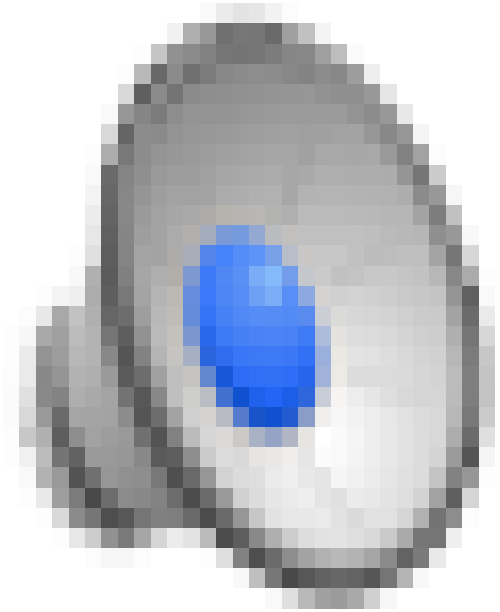




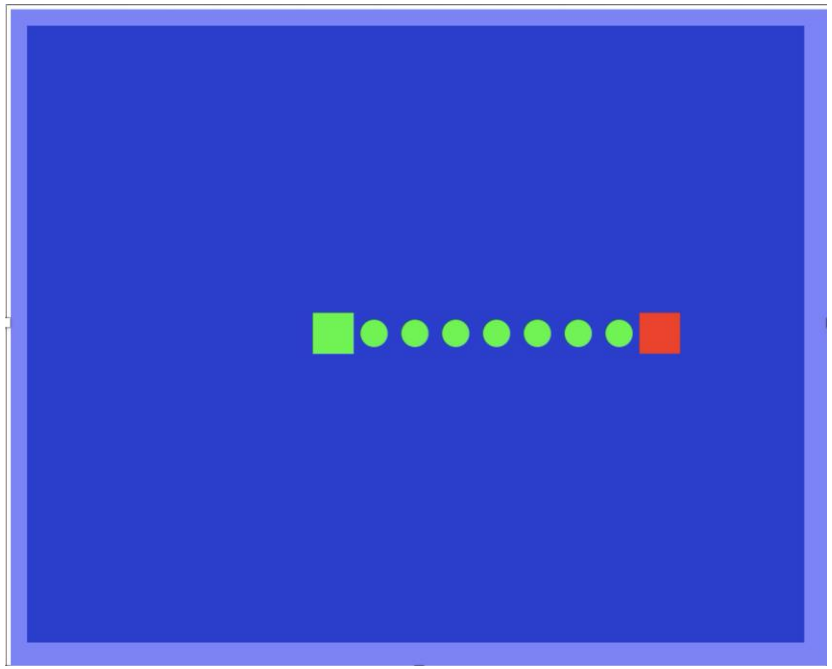
Algoritmo A*



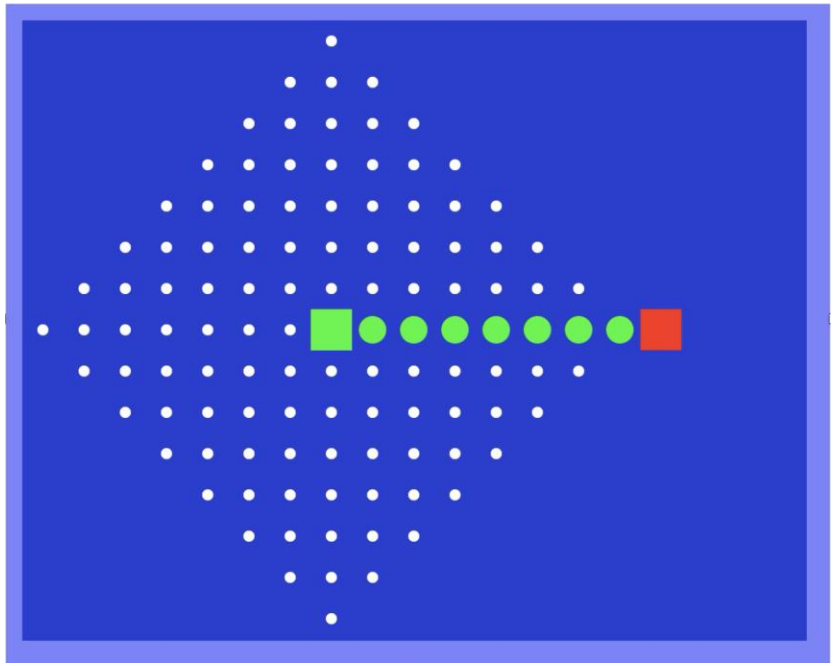
Algoritmo A*



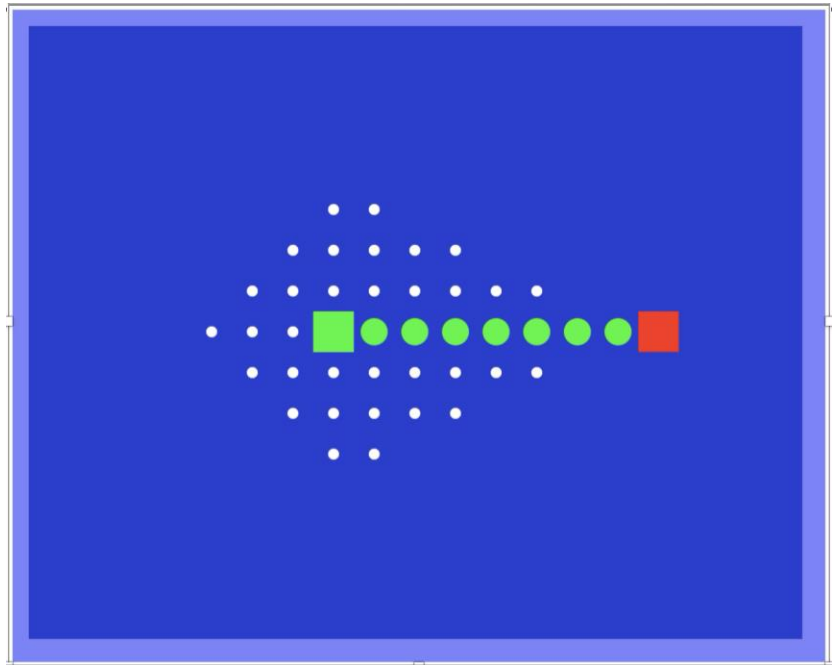
A* vs UCS vs Voraz



Voraz (h)



Costo uniforme (g)



A* (g+h)

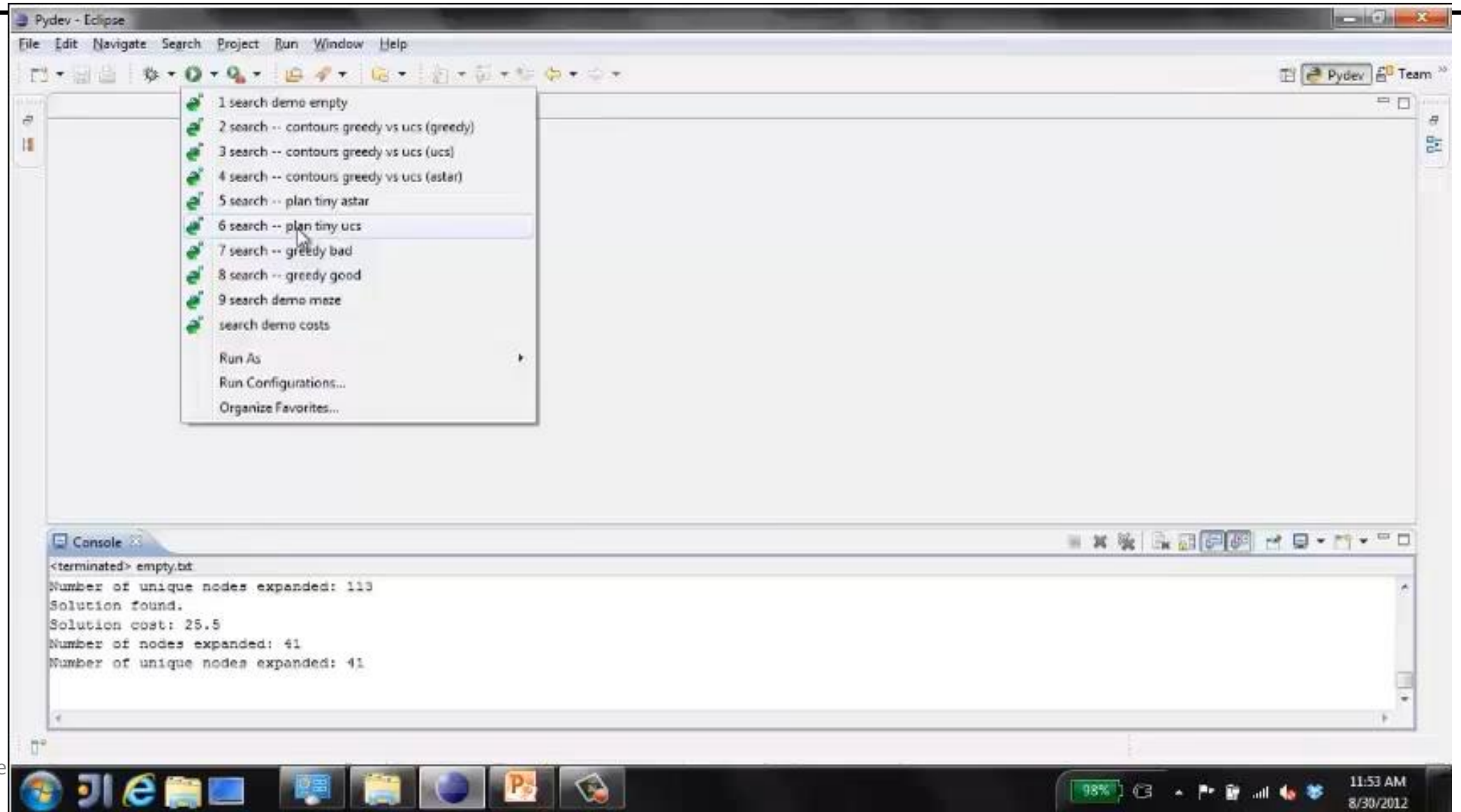
A* vs UCS vs Voraz

Voraz (h)

Costo uniforme (g)

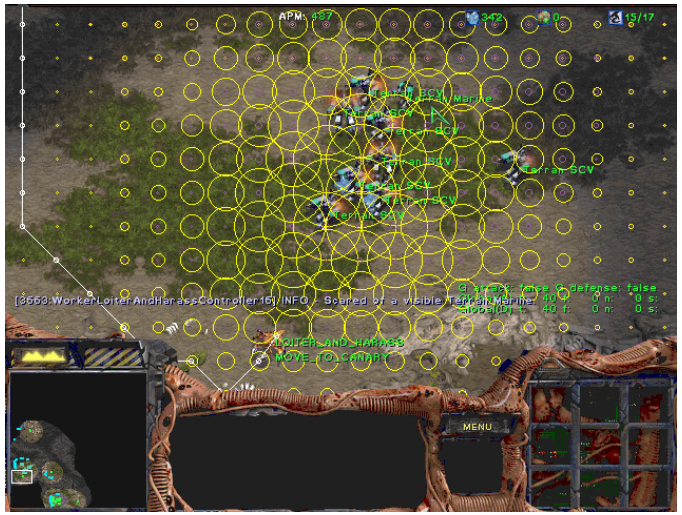
A* (g+h)

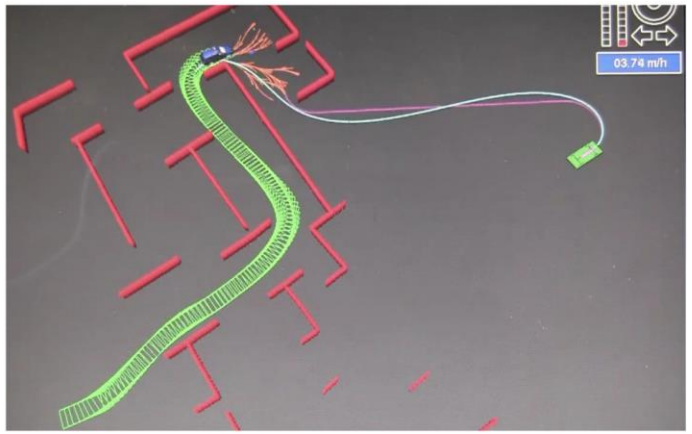
A* vs UCS


Pydev - Eclipse
File Edit Navigate Search Project Run Window Help
Pydev
Team
1 search demo empty
2 search -- contours greedy vs ucs (greedy)
3 search -- contours greedy vs ucs (ucs)
4 search -- contours greedy vs ucs (astar)
5 search -- plan tiny astar
6 search -- plan tiny ucs
7 search -- greedy bad
8 search -- greedy good
9 search demo maze
search demo costs
Run As
Run Configurations...
Organize Favorites...
Console
<terminated> empty.txt
Number of unique nodes expanded: 113
Solution found.
Solution cost: 25.5
Number of nodes expanded: 41
Number of unique nodes expanded: 41
98%
11:53 AM
8/30/2012

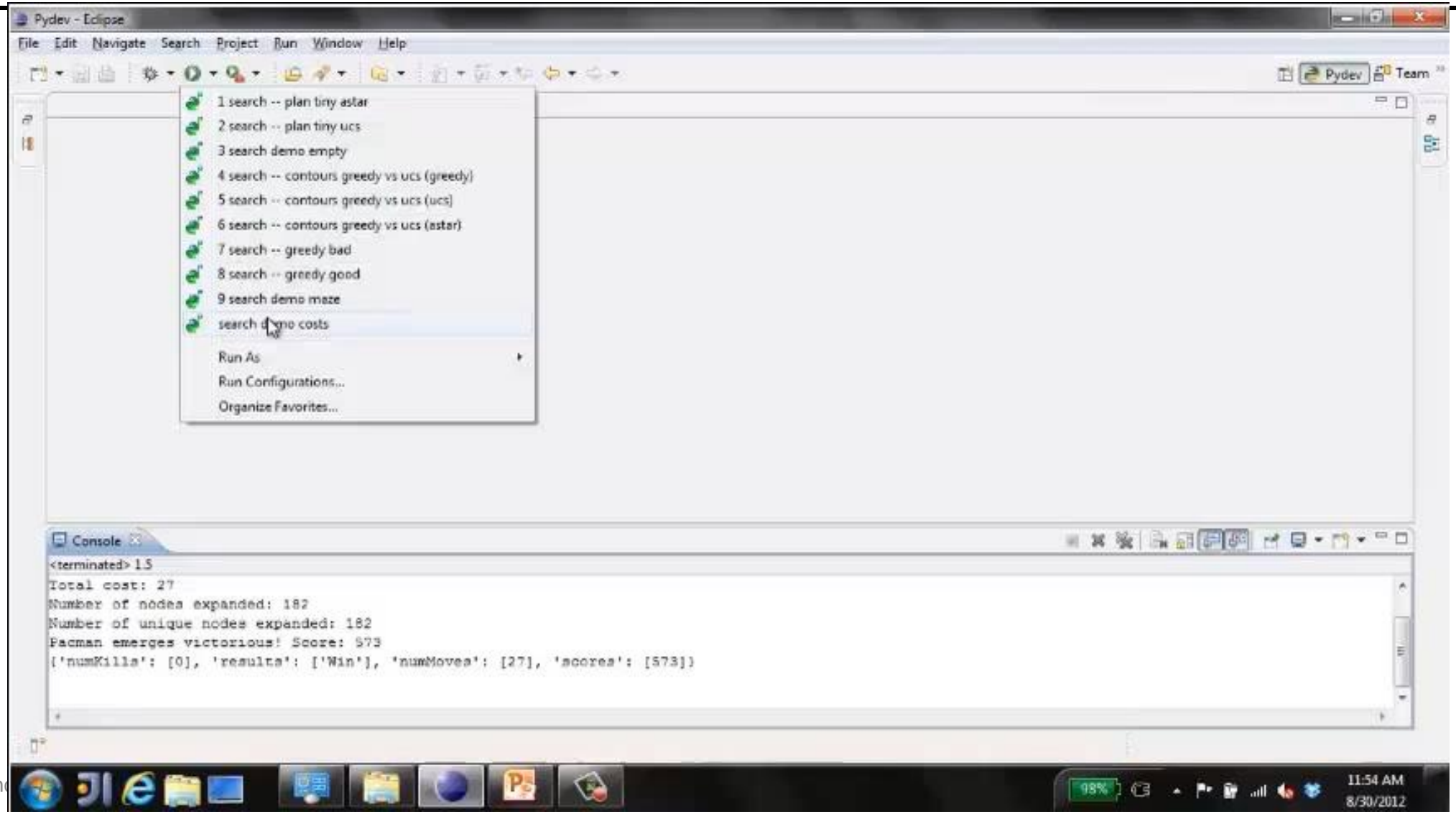
Aplicaciones A*

- Video Juegos
- Problemas de enrutamiento
- Problemas de asignación de recursos
- Movimiento de robots
- Análisis de lenguaje
- Traducción automática
- Reconocimiento de voz
- Síntesis de proteínas



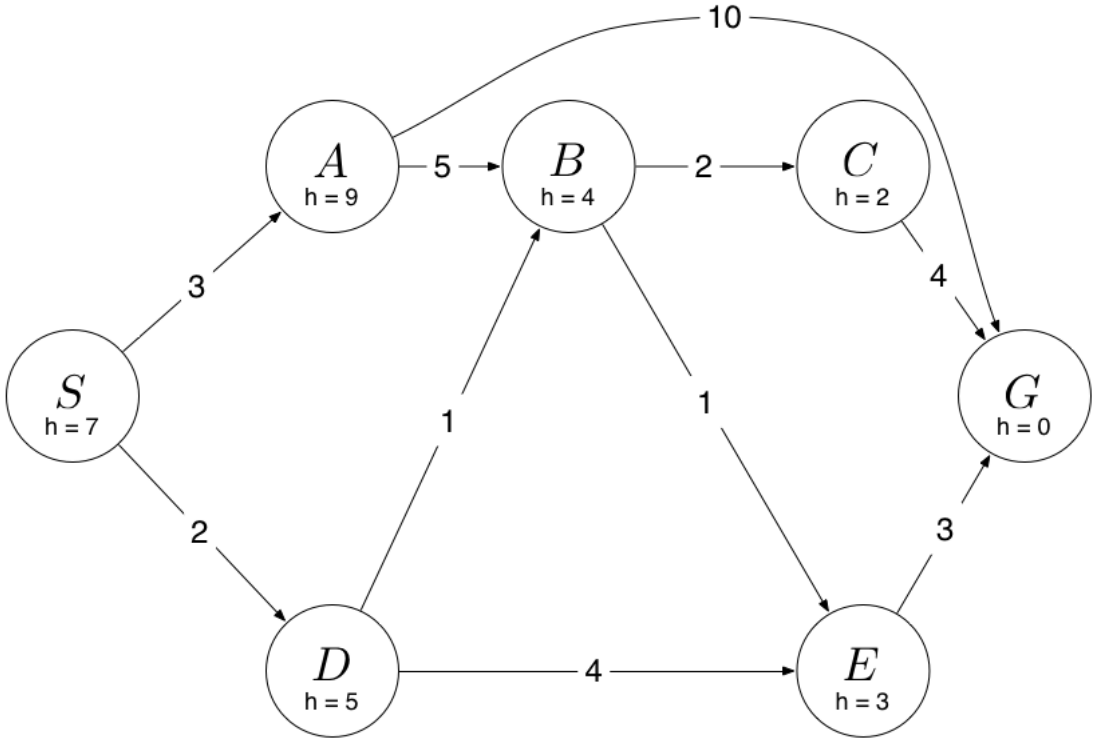


Algoritmos de búsqueda



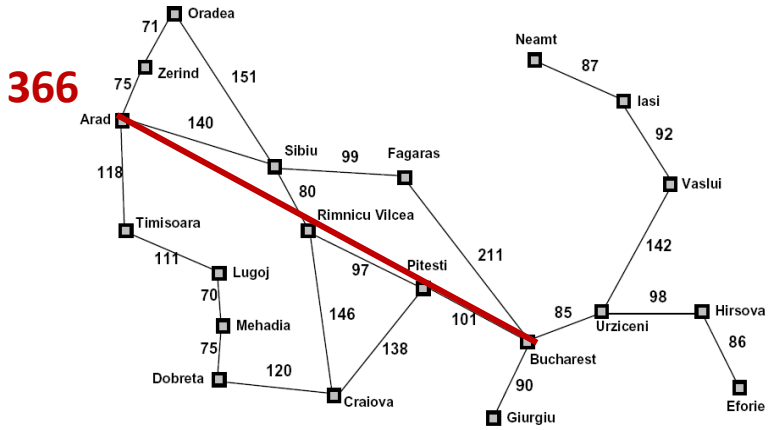
Ejercicio 3

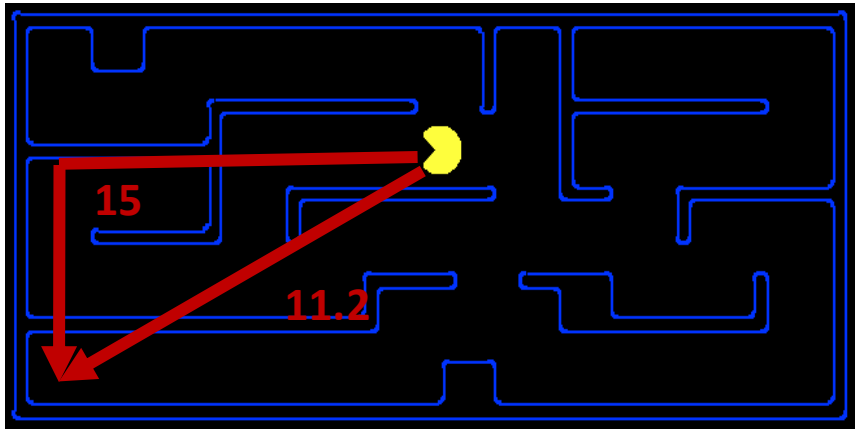
- Encuentre la solución para el siguiente problema utilizando A*
- ¿La heurística es admisible?
- Suponga que los empates se rompen alfabéticamente
  - $S \rightarrow X \rightarrow A$ será expandido antes $S \rightarrow X \rightarrow B$
  - $S \rightarrow A \rightarrow Z$ será expandido antes $S \rightarrow B \rightarrow A$
- Para cada caso encuentre la solución y la secuencia de nodos expandidos



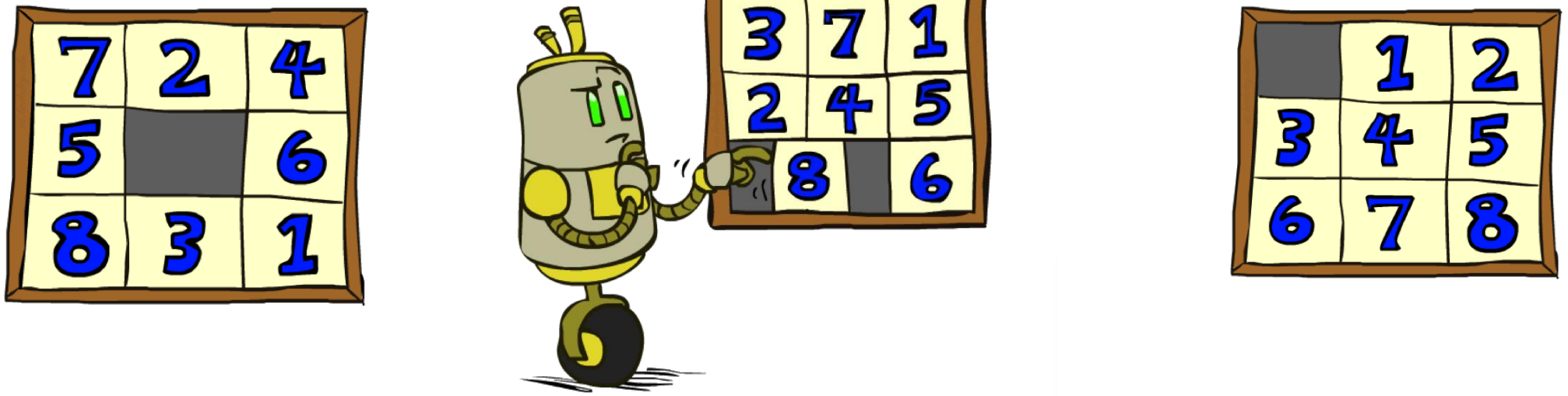
Creando heurísticas admisibles

- Una alternativa para proponer una heurística admisible es relajar un poco las restricciones del problema
- Las heurísticas admisibles pueden verse como una solución al problema donde se permiten nuevas acciones
- Inclusive, heurísticas no admisibles pueden llegar a ser útiles





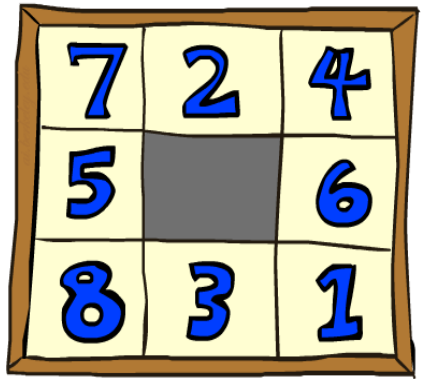
Ejemplo: 8-puzzle



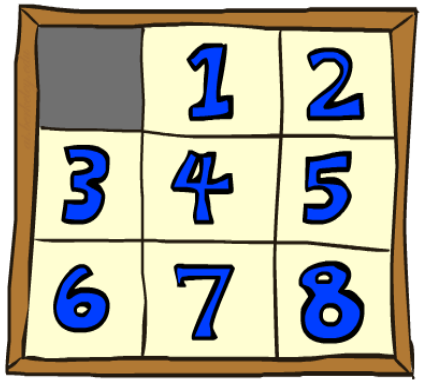
- ¿Cuáles son los estados del problema?
- ¿Cuántos estados alcanzables existen?
- ¿Cuáles son las acciones disponibles?
- ¿Cuál debería ser el costo en cada acción?

Ejemplo: 8-puzzle

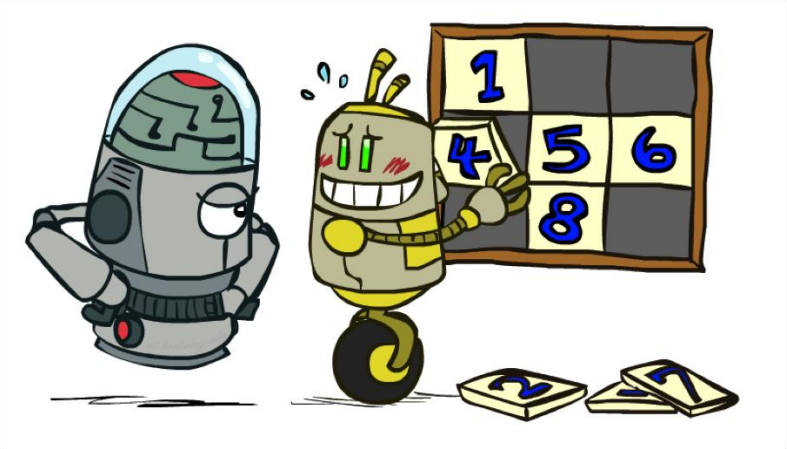
- Heurística: Número de piezas mal ubicadas
- ¿La heurística es admisible?
- h(start) = 8
- ¡Esta heurística corresponde a una versión simplificada del problema!



Start State



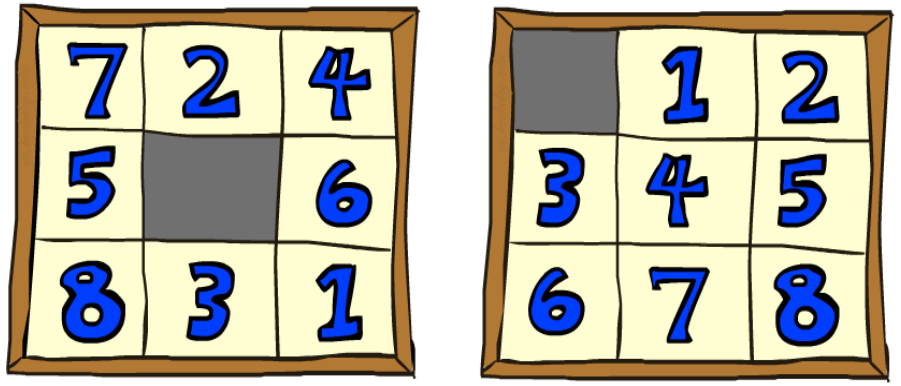
Goal State



	Promedio de nodos expandidos cuando la ruta óptima tiene:		
	…4 pasos	…8 pasos	…12 pasos
UCS	112	6,300	3.6×10^6
h	13	39	227

Ejemplo: 8-puzzle

- Supongamos que cada pieza se puede mover en cualquier dirección ignorando otras piezas
- Sería equivalente a calcular la distancia de Manhattan para cada pieza con respecto a su posición esperada
- ¿La heurística es admisible?
- h(start) = 3 + 1 + 2 + … = 18

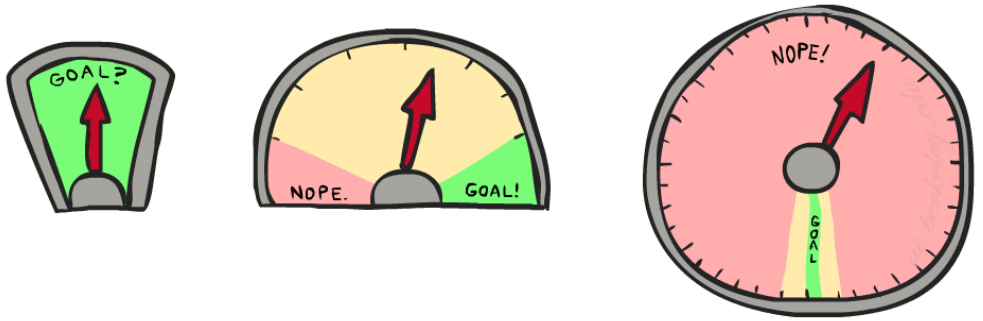


Start State Goal State

	Promedio de nodos expandidos cuando la ruta óptima tiene:		
	…4 pasos	…8 pasos	…12 pasos
h	13	39	227
Manhattan	12	25	73

Ejemplo: 8-puzzle

- ¿Qué ocurriría si utilizáramos el costo real como función heurística?
  - ¿Sería admisible?
  - ¿Expandiríamos menos nodos?
  - ¿Es una buena idea?



- Existe una dependencia entre la precisión de la estimación y la cantidad de trabajo que se requiere realizar por nodo para calcular la función heurística
- Una función heurística que está demasiado cerca del valor real requiere un mayor esfuerzo computacional para calcularse

Combinando heurísticas

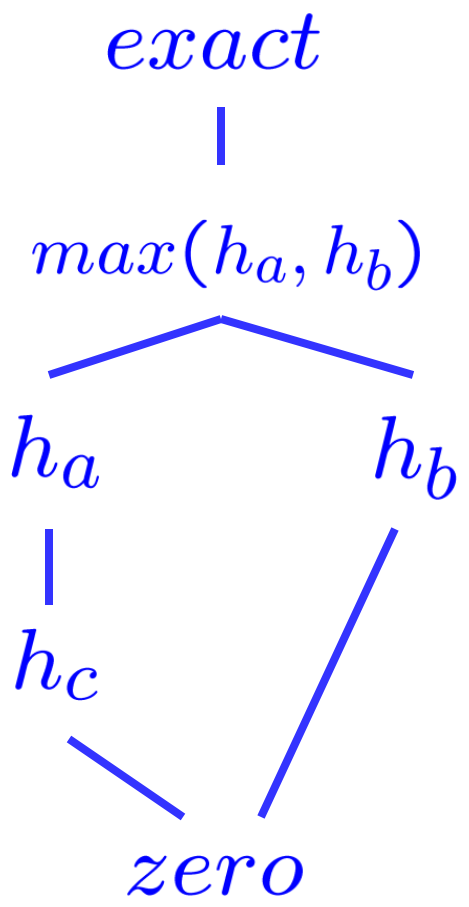
- **Dominancia**: h_a domina a h_c si:

$$\forall n : h_a(n) \geq h_c(n)$$

- **Combinando heurísticas**
  - Podemos combinar dos heurísticas, seleccionando siempre el valor máximo

$$h(n) = max(h_a(n), h_b(n))$$

  - Si ambas heurísticas son admisibles, el resultado de esta combinación también será admisible
- **Heurísticas triviales:** la heurística *cero y el costo exacto*



Heurísticas consistentes

Idea: costo heurístico estimado $\leq$ costo real

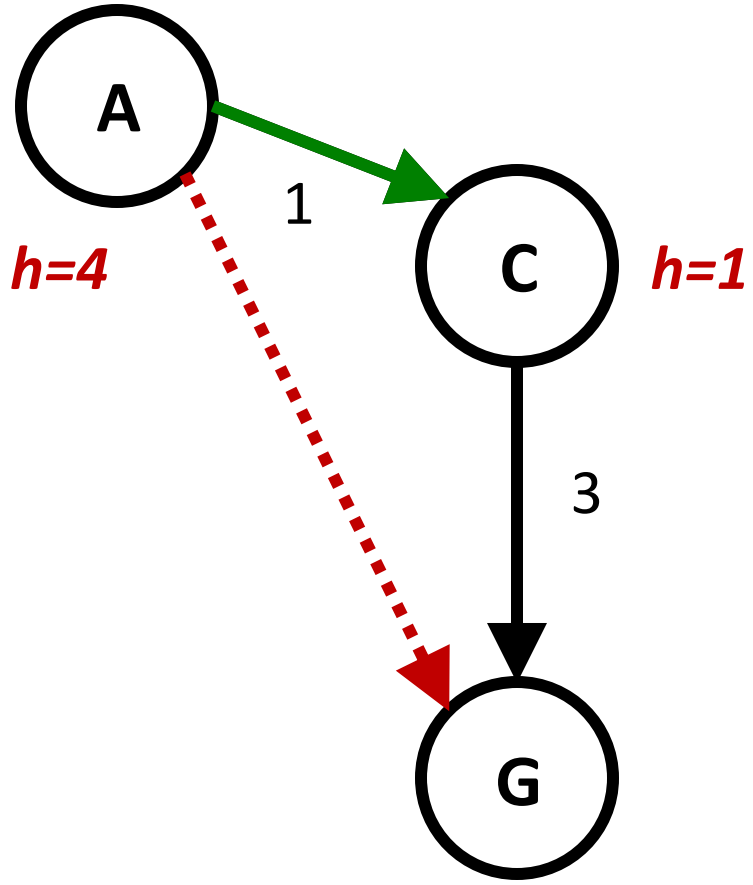
- **Admisibilidad**: $h(A) \leq h^*(A)$
- **Consistencia:** $h(A) - h(C) \leq c(A, C)$

Propiedades heurísticas consistentes:

- El valor de $f(\cdot)$ a lo largo de un camino nunca decrece

$$h(A) \leq c(A, C) + h(C)$$

- Bajo esta condición la búsqueda en grafos es óptima
- La consistencia es una propiedad más robusta que la admisibilidad



Heurísticas consistentes

Idea: costo heurístico estimado $\leq$ costo real

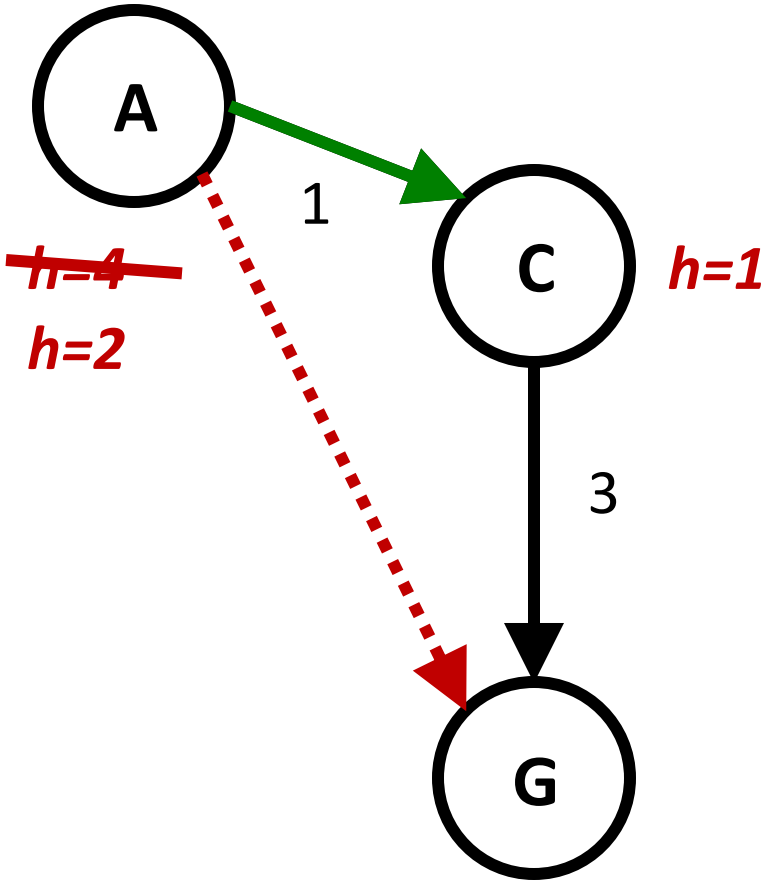
- **Admisibilidad**: $h(A) \leq h^*(A)$
- **Consistencia:** $h(A) - h(C) \leq c(A, C)$

Propiedades heurísticas consistentes:

- El valor de $f(\cdot)$ a lo largo de un camino nunca decrece

$$h(A) \leq c(A, C) + h(C)$$

- Bajo esta condición la búsqueda en grafos es óptima
- La consistencia es una propiedad más robusta que la admisibilidad

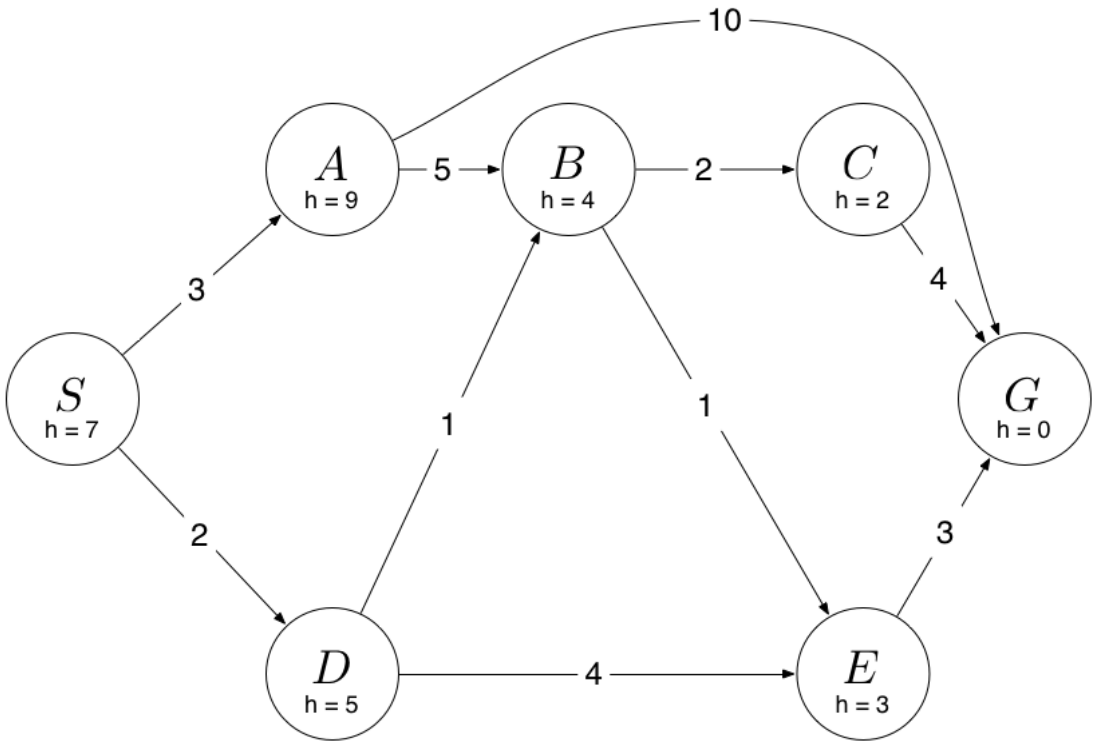


Optimalidad del algoritmo A*

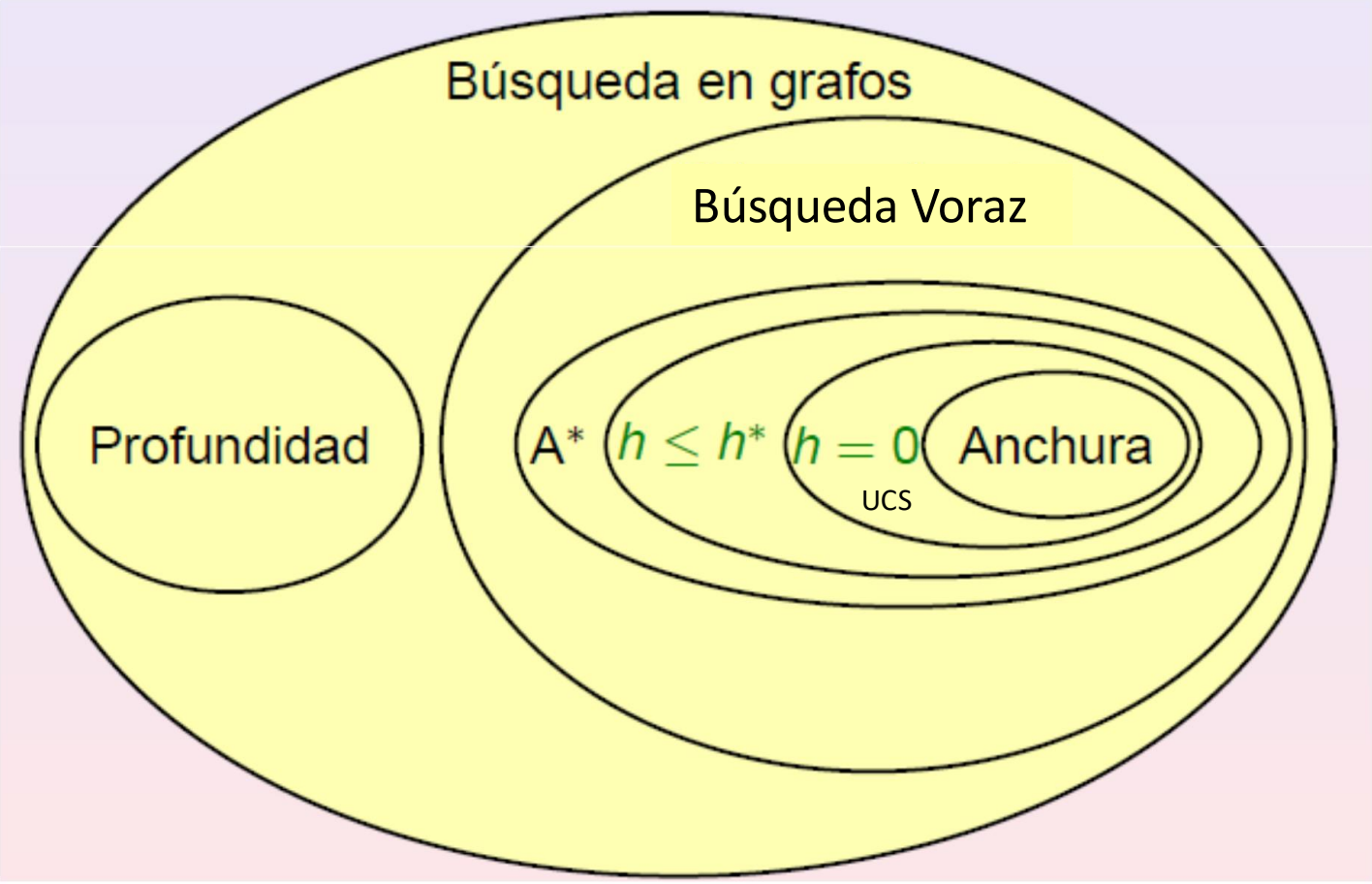
- Búsqueda en grafos:
  - A* es óptimo si la heurística es consistente
- ¡La consistencia implica admisibilidad!
- En la práctica la mayoría de las heurísticas admisibles tienden a ser consistentes, en particular aquellas que surgen de relajar las restricciones de un problema

Ejercicio 4

- Encuentre la solución para el siguiente problema utilizando búsqueda en grafos
- ¿La heurística es consistente?
- Suponga que los empates se rompen alfabéticamente
  - $S \rightarrow X \rightarrow A$ será expandido antes $S \rightarrow X \rightarrow B$
  - $S \rightarrow A \rightarrow Z$ será expandido antes $S \rightarrow B \rightarrow A$
- Para cada caso encuentre el camino que representa la solución



Resumen


Búsqueda en grafos
Búsqueda Voraz
Profundidad
A*
$h \leq h^*$
$h = 0$
UCS
Anchura

Referencias

- RUSSELL, Stuart; NORVIG, Peter. Artificial intelligence: a modern approach.
- MITCHELL, Melanie. *Artificial intelligence: A guide for thinking humans*. Penguin UK.
- Course: Introduction for artificial Intelligence (Spring 2021) (Berkeley)