DECSAI
Departamento de Ciencias de la Computación e I.A.
Universidad de Granada

Búsqueda heurística

© Fernando Berzal, berzal@acm.org

Búsqueda heurística

- Búsqueda primero el mejor
  p.ej. búsqueda de coste uniforme [UCS]
- Heurísticas
- Búsqueda greedy
- El algoritmo A*
  - Heurísticas admisibles
  - Heurísticas consistentes

Búsqueda primero el mejor

Uso de información en la búsqueda

Exploración primero el mejor

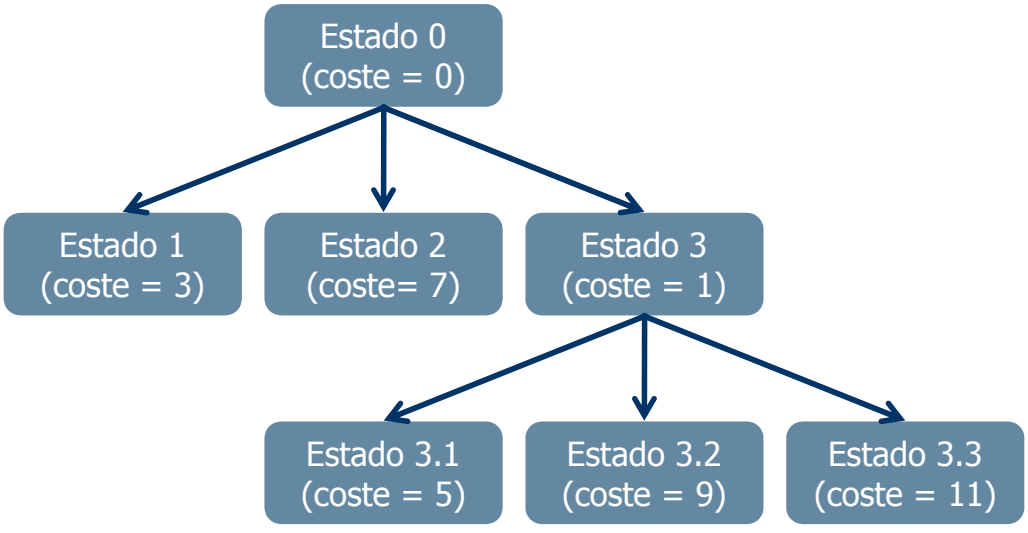
Se explora exhaustivamente el grafo utilizando una función heurística para determinar el orden en que se exploran los nodos:

1. Seleccionar el nodo E del grafo que tenga mayor valor para la función heurística.
2. Seleccionar un operador R aplicable sobre E.
3. Aplicar R, para obtener un nuevo nodo R(E).
4. Añadir el arco E $\rightarrow$ R(E) al grafo
5. Repetir el proceso.

Búsqueda primero el mejor

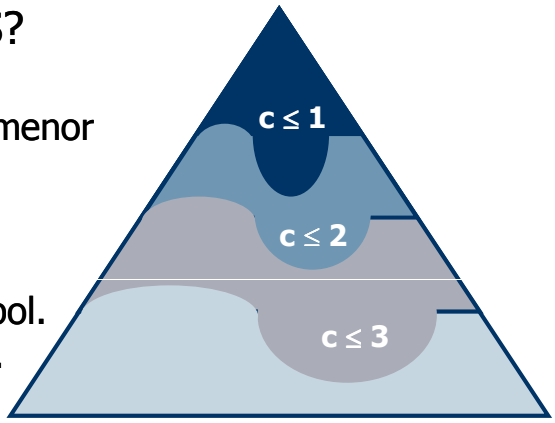
Búsqueda de coste uniforme

Uniform Cost Search (UCS)



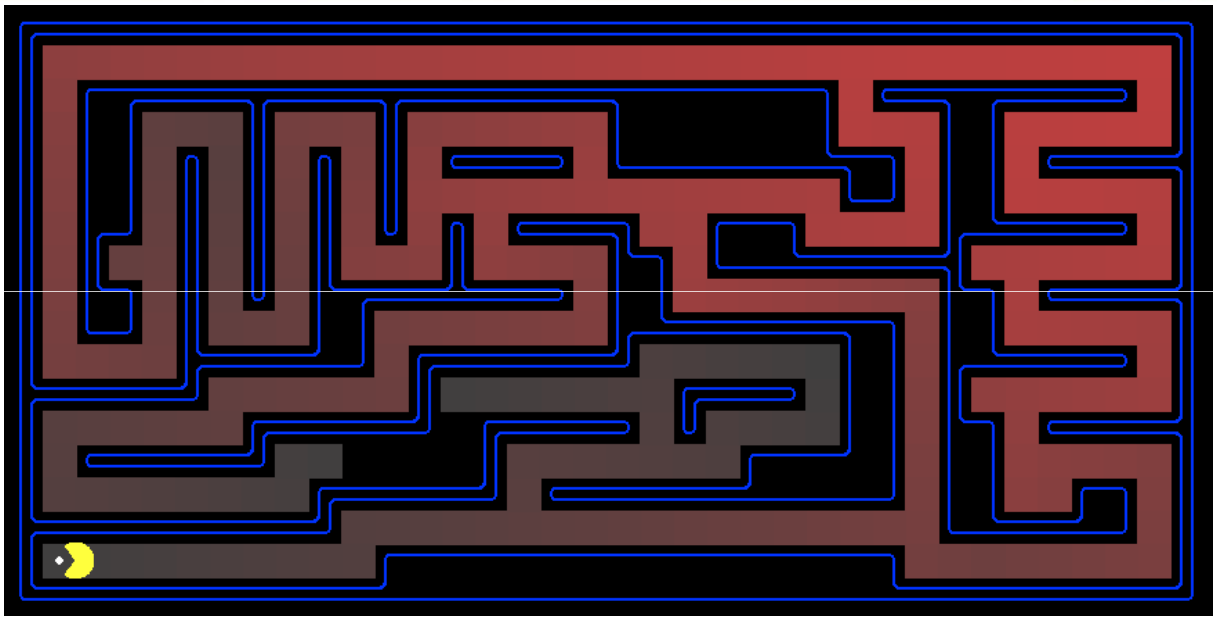
Búsqueda primero el mejor

- ¿Cuántos nodos expande UCS?
  - Todos los que tengan un coste menor que la solución de menor coste.
  - Tiempo $\mathbf{O(b^{C^*/\varepsilon})}$
    b: Factor de ramificación del árbol.
    C*: Coste de la solución óptima.
    ε: Coste mínimo de cada arco.



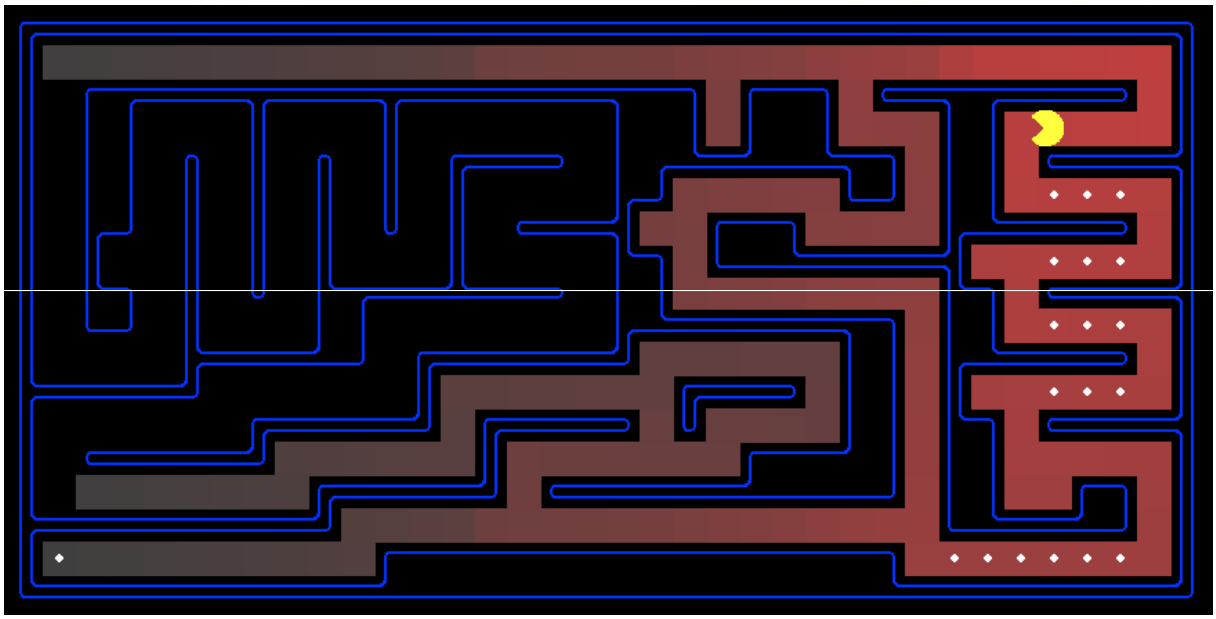
- ¿Cuántos nodos puede haber en la frontera de búsqueda?
  - Los del nivel correspondiente al coste de la solución, $\mathbf{O(b^{C^*/\varepsilon})}$
- Si existe una solución de coste finito y ε es positivo, el algoritmo encuentra la solución.

Búsqueda primero el mejor



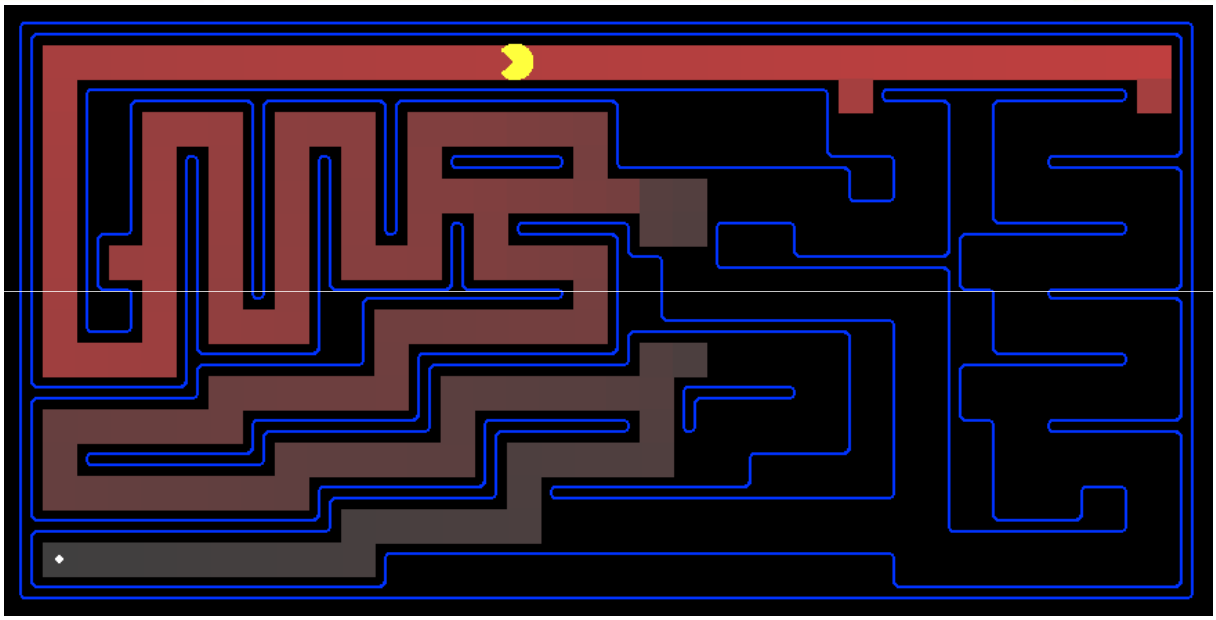
Demo: UCS con costes uniformes = BFS

Búsqueda primero el mejor



Demo: UCS

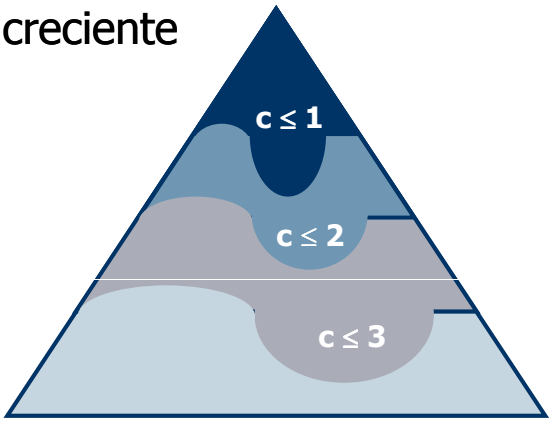
Búsqueda primero el mejor



Demo UCS

Búsqueda primero el mejor

UCS explora contornos de coste creciente



Ventajas

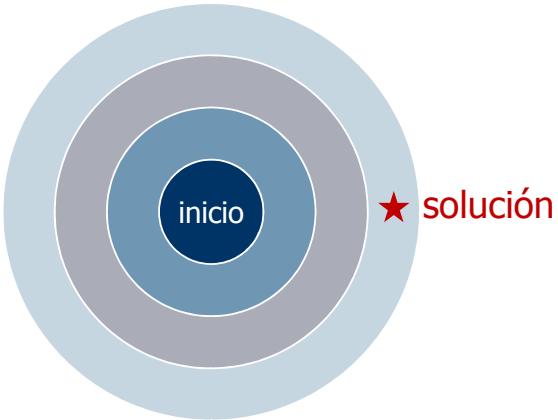
- Algoritmo completo
  (encuentra la solución)
- Algoritmo óptimo
  (encuentra la mejor solución)

Desventajas

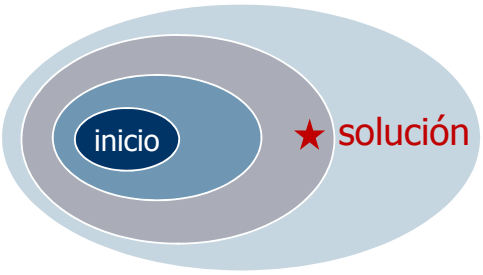
- Explora en todas direcciones
  (no tiene en cuenta cuál es nuestro objetivo)

Búsqueda primero el mejor

UCS



Lo deseable…



Heurísticas

Las heurísticas son criterios, métodos o principios para decidir cuál de entre varias acciones promete ser la mejor para alcanzar una determinada meta.

- En la generación del árbol de búsqueda, nos podemos guiar con heurísticas que nos den una indicación acerca de cómo de bueno o prometedor es un determinado estado u operador.

  p.ej.
  ¿Qué pieza deberíamos mover en una partida de ajedrez?
  ¿Qué regla aplicamos en primer lugar al hacer un diagnóstico?

Heurísticas

El uso de heurísticas nos permite guiar nuestra búsqueda de una solución, lo que nos permitirá obtener una solución más rápidamente que si utilizásemos estrategias de búsqueda a ciegas.

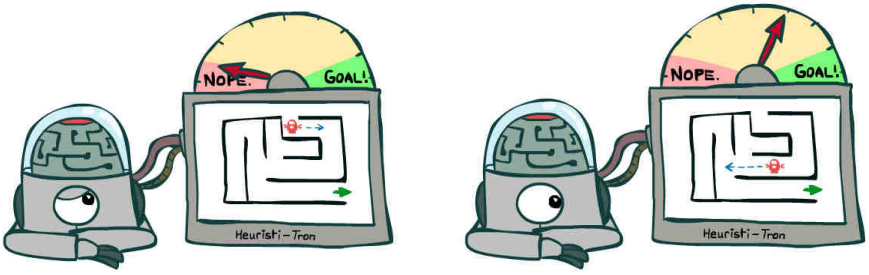
NOTA: Compárese el uso heurístico de información para guiar el proceso de búsqueda en Inteligencia Artificial con formalismos como la teoría de la decisión o la teoría de juegos, que en ocasiones nos permitirán determinar la decisión óptima si somos capaces de identificar todos los factores relevantes para el problema en cuestión (y las incertidumbres asociadas a ellos !!).

http://en.wikipedia.org/wiki/Decision_theory
http://en.wikipedia.org/wiki/Game_theory

Heurísticas

En problemas de búsqueda:

- Una heurística será una función que utilizaremos para estimar cómo de cerca estamos de la solución.
- Cada heurística estará diseñada para un problema de búsqueda particular.



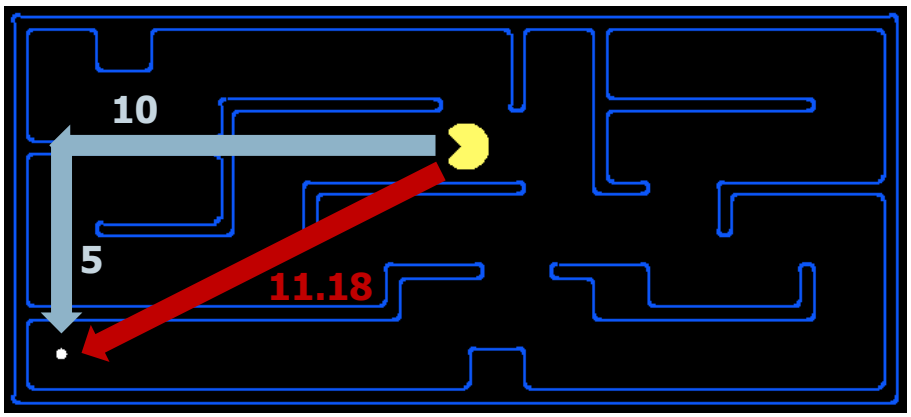
Berkeley CS188

Heurísticas

Ejemplo

En problemas de búsqueda de caminos/trayectorias:

- Distancia euclídea
- Distancia de Manhattan



Heurísticas

¿Qué sucede si utilizamos directamente los valores proporcionados por nuestra función heurística?

Búsqueda greedy ("voraz")

Saltamos al nodo de mejor valor heurístico…

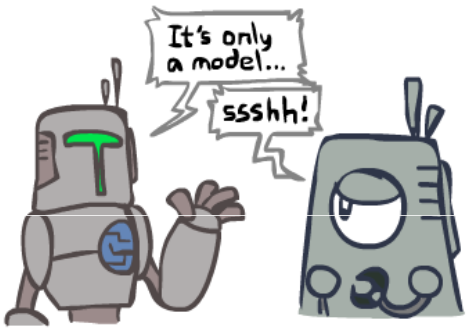
… lo que nos lleva "directamente" a una solución.

Sin embargo, puede que no sea la mejor solución :-(

Funcionamiento en el peor caso: Como DFS
(puede que tengamos que explorar el espacio de búsqueda completo, siempre y cuando evitemos ciclos).

Heurísticas

No siempre funcionan bien…



Berkeley CS188

El algoritmo A*

IDEA

Combinar UCS y greedy

- La búsqueda por coste uniforme sólo tiene en cuenta el coste de llegar al nodo actual (hacia atrás): **g(n)**
- La búsqueda greedy sólo tiene en cuenta el coste estimado para llegar a la solución (hacia adelante): **h(n)**

El algoritmo A* guía la búsqueda utilizando la suma:

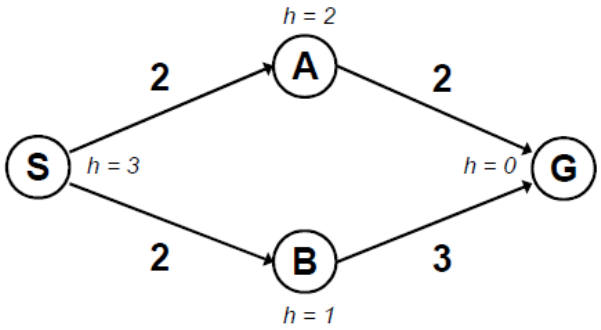
f(n) = g(n) + h(n)

El algoritmo A*

Cuestiones de implementación

¿Cuándo debemos terminar la búsqueda?

No cuando se encuentre una solución, sino cuando se expanda el nodo correspondiente a la solución.

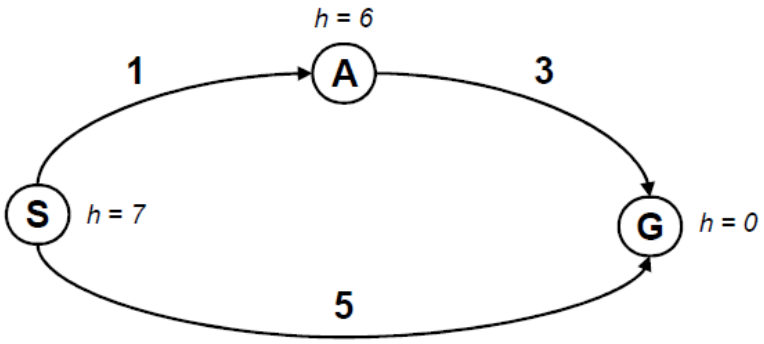


El algoritmo A*

Cuestiones de implementación

¿Se encuentra siempre la solución óptima?

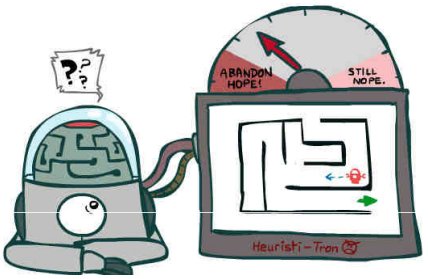
Depende de la función heurística que utilicemos…



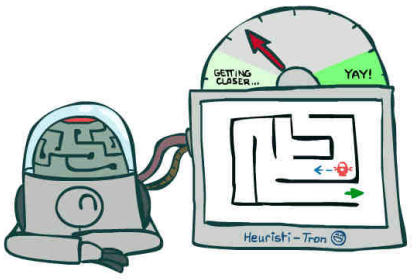
El algoritmo A*

Heurísticas admisibles

- Las heurísticas pesimistas (inadmisibles) impiden la optimalidad del algoritmo A* al descartar buenos planes.



- Las heurísticas optimistas (admisibles) no subestiman la calidad de un buen plan.

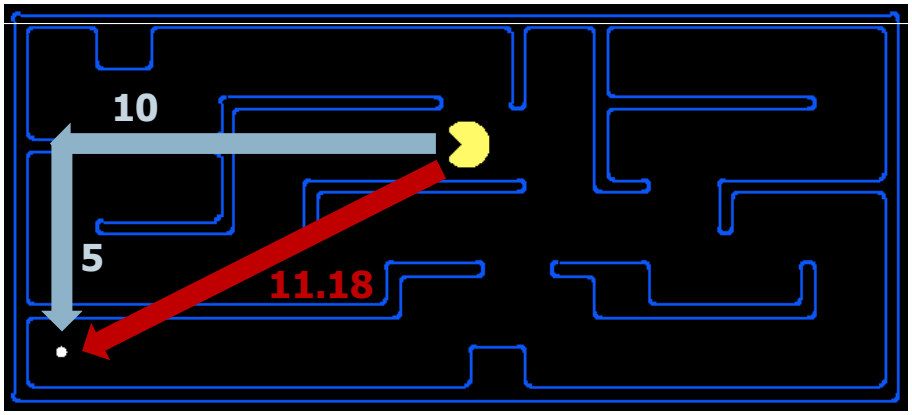


Berkeley CS188

El algoritmo A*

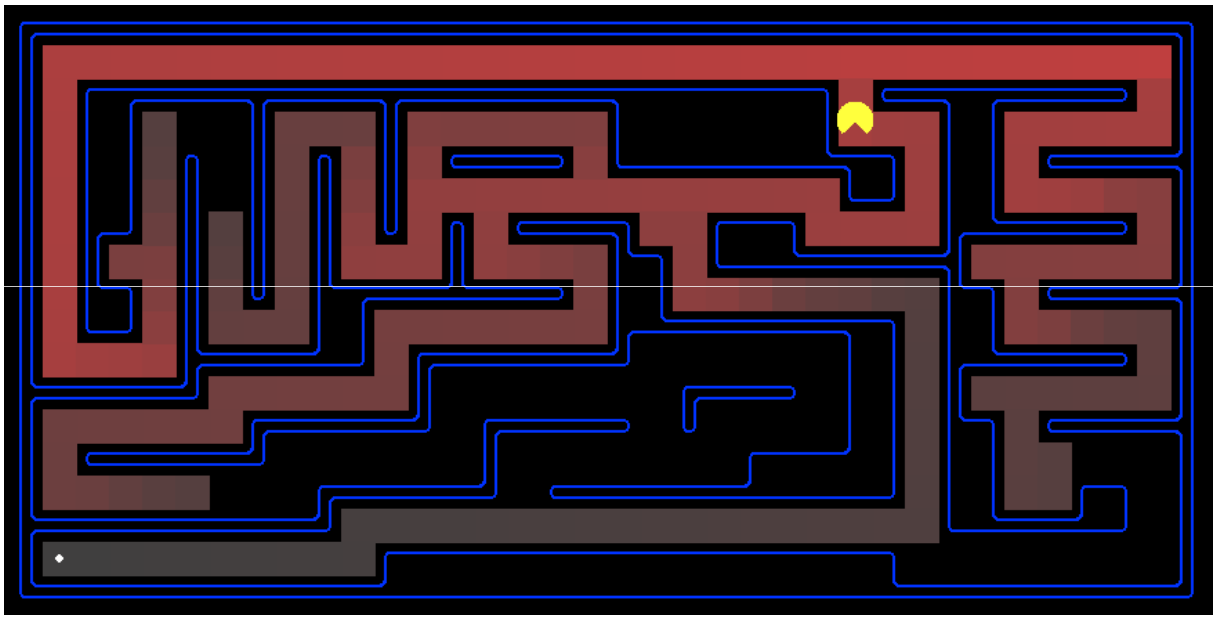
Heurísticas admisibles

Una heurística es admisible si **h(n) ≤ h*(n)** donde **h*(n)** es el coste verdadero de la solución desde **n**.



NOTA: También se requiere h(n) ≥ 0,
de modo que h(G) = 0 para cualquier solución G.

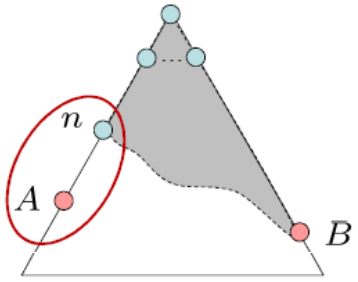
El algoritmo A*



Demo

El algoritmo A*

Optimalidad del algoritmo A*



Supongamos que

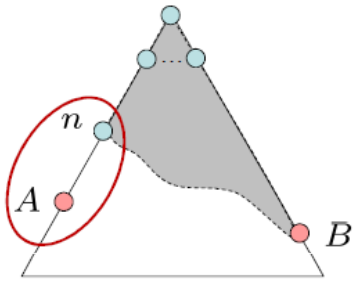
- h(n) es una heurística admisible
- A es una solución óptima
- B es una solución subóptima

El algoritmo A* expandirá el nodo A antes que el B.

El algoritmo A*

Optimalidad del algoritmo A*

El algoritmo A* expandirá A antes que B.



Si B está en la frontera de búsqueda,

- Algún ancestro n de A también estará en la frontera de búsqueda (puede que el propio A).
- n se expandirá antes que B, ya que $f(n) \leq f(A) < f(B)$
  - $f(n) \leq f(A)$ $\quad f(n) = g(n)+h(n) \leq g(A) = f(A)$
  - $f(A) < f(B)$ $\quad g(A) < g(B) \rightarrow f(A) < f(B)$
- Todos los ancestros de A se expanen antes que B.
- A se expande antes que B.

El algoritmo A*

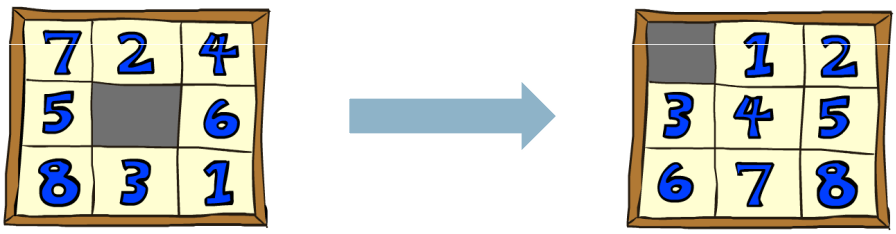
Diseño de heurísticas admisibles

- El secreto de resolver problemas difíciles de búsqueda está en encontrar heurísticas admisibles.
- A menudo, una forma de obtener heurísticas admisibles consiste en resolver problemas relajados (problemas similares al que queremos resolver, pero con menos restricciones sobre posibles acciones).

El algoritmo A*

Diseño de heurísticas admisibles

8-puzzle



Estado inicial

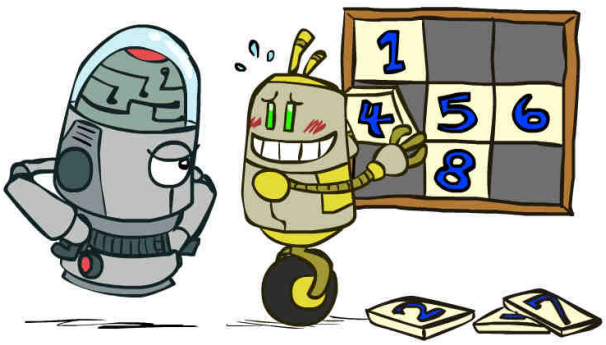
Objetivo

El algoritmo A*

Diseño de heurísticas admisibles

8-puzzle

Heurística 1: Número de piezas mal colocadas



Berkeley CS188

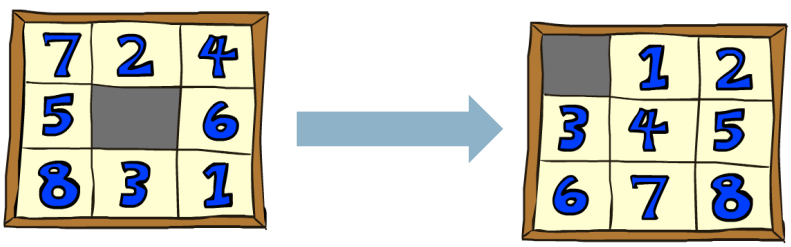
El algoritmo A*

Diseño de heurísticas admisibles

8-puzzle

Heurística 2: Distancia de Manhattan total

$$h(n) = 3+1+2+\ldots = 18$$

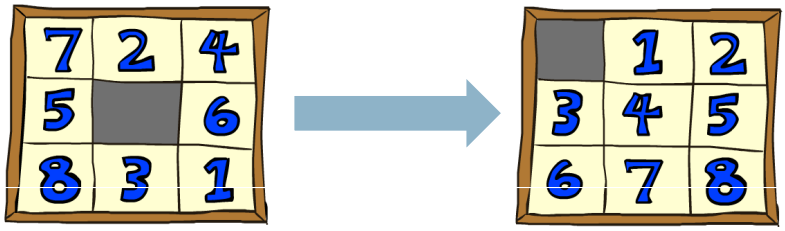


El algoritmo A*

Diseño de heurísticas admisibles

8-puzzle

Resultados



Número de nodos expandidos cuando la solución óptima tiene…

Heurística	4 pasos	8 pasos	12 pasos
h=0 (UCS)	112	6300	3.6M
#piezas	13	39	227
Manhattan	12	25	73

El algoritmo A*

Diseño de heurísticas admisibles

- Mejor heurística admisible posible:
  Coste real de la solución.

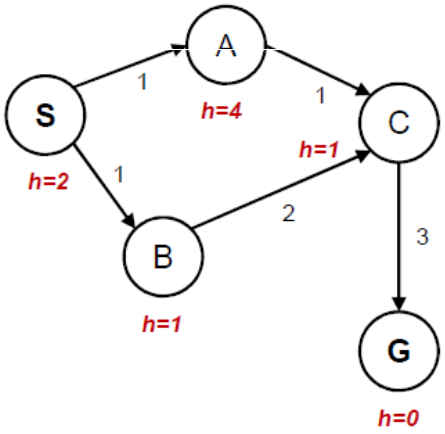
- Tiempo de ejecución:
  Cuanto más nos acerquemos al coste real, tendremos que expandir menos nodos pero… también realizaremos más trabajo en cada nodo.

- Combinación de heurísticas admisibles:
  $h(n) = \max \{ h_a(n), h_b(n) \}$
  El máximo de heurísticas admisibles también es admisible.

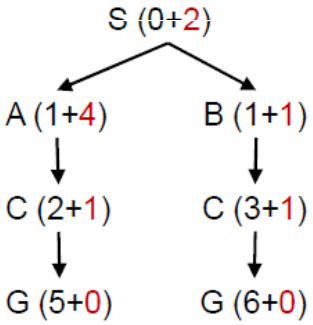
El algoritmo A*

Búsqueda sobre grafos…

Espacio de estados

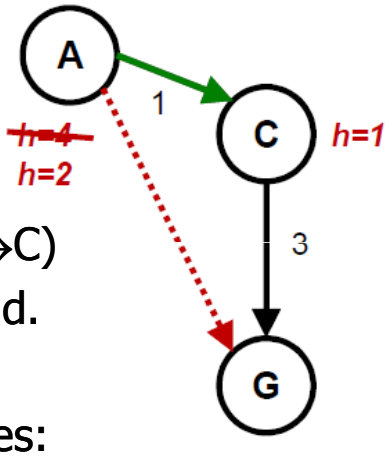


Árbol de búsqueda



El algoritmo A*

Heurísticas consistentes



Coste heurístico estimado ≤ coste real

- Admisibilidad: $h(A) \leq h^*(A)$
- Consistencia: $h(A) - h(C) \leq \text{coste}(A \rightarrow C)$

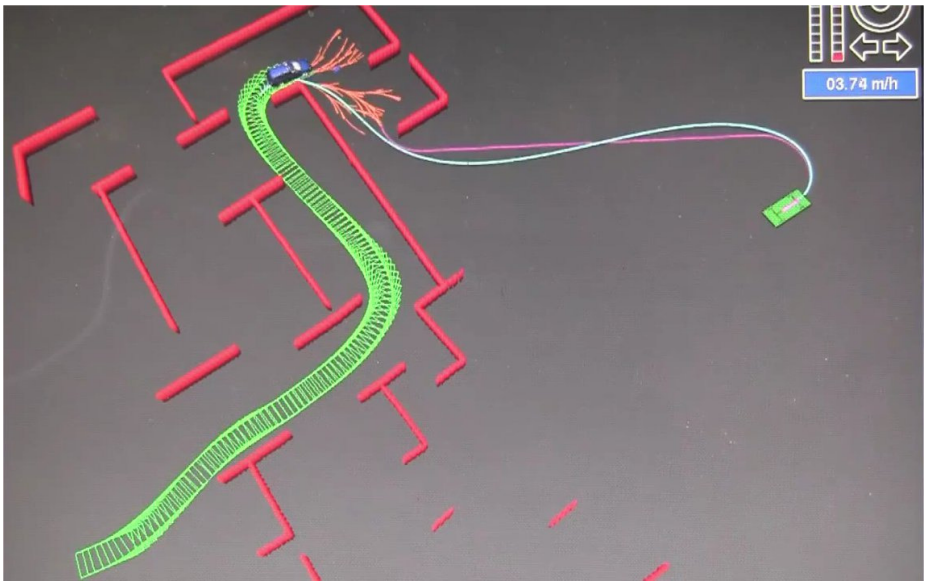
  NOTA: Consistencia implica admisibilidad.

Propiedades de las heurísticas consistentes:

- El valor de f a lo largo de un camino nunca decrece.

  $h(A) \leq \text{coste}(A \rightarrow C) + h(C)$
- El algoritmo A* sobre grafos es óptimo (siempre que la heurística utilizada sea consistente).

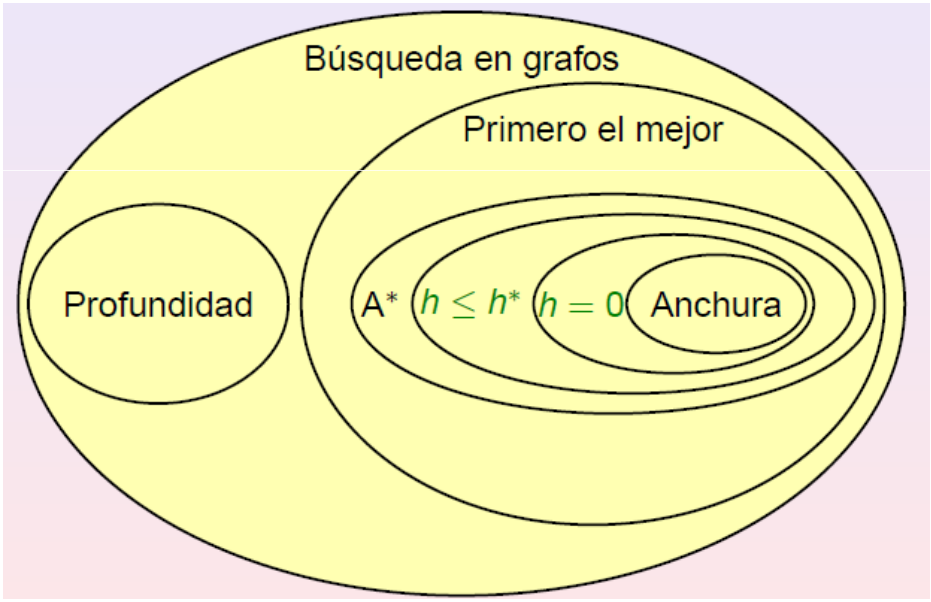
El algoritmo A*



Demo
Path planning @ Google self-driving car

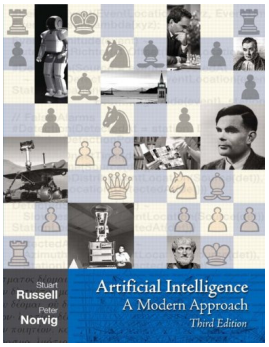
Resumen

Algoritmos de búsqueda

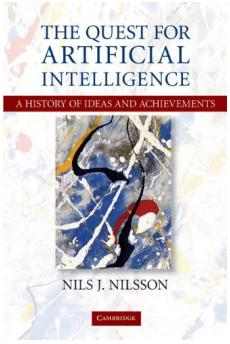


Bibliografía

- Stuart Russell & Peter Norvig:
  Artificial Intelligence:
  A Modern Approach
  Prentice-Hall, 3rd edition, 2009
  ISBN 0136042597



- Nils J. Nilsson
  The Quest for Artificial Intelligence
  Cambridge University Press, 2009
  ISBN 0521122937



Bibliografía

Bibliografía complementaria

- Elaine Rich & Kevin Knight:
  Artificial Intelligence.
  McGraw-Hill, 1991.

- Patrick Henry Winston:
  Artificial Intelligence.
  Addison-Wesley, 1992.

- Nils J. Nilsson:
  Principles of Artificial Intelligence.
  Morgan Kaufmann, 1986.