

BÚSQUEDA INFORMADA

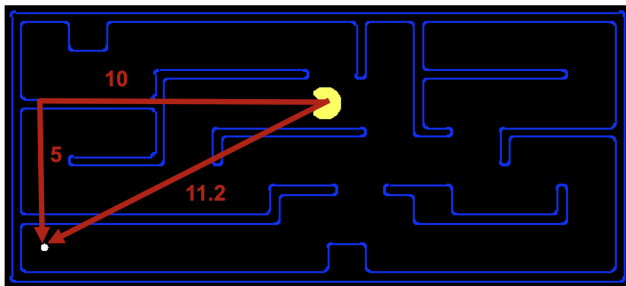
ALAN REYES-FIGUEROA
INTELIGENCIA ARTIFICIAL

(AULA 07) 29.ENERO.2024

Búsqueda Informada

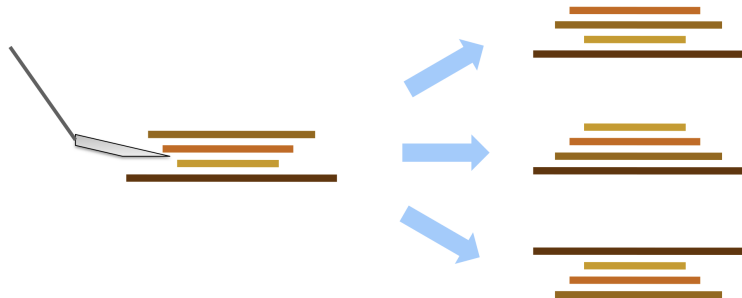
Heurística: Dado $G = (V, E)$ un grafo de estados. Una **heurística de búsqueda** es una función $h : V \rightarrow \mathbb{R}$ que estima qué tan cerca (o lejos) estamos del estado o configuración objetivo.

- Es una aproximación del costo real de cuánto falta para llegar al objetivo.
- Son diseñadas específicamente para cada problema de búsqueda.



Heurísticas

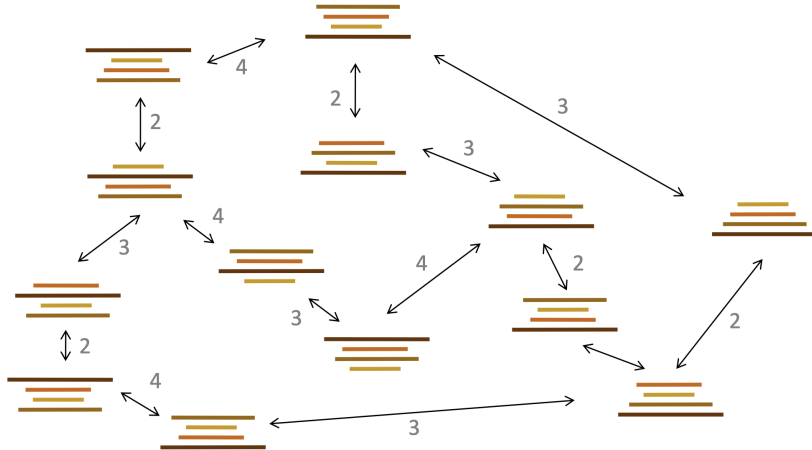
Ejemplo: El problema de los panqueques.



En este ejemplo, el costo de un movimiento es el número o tamaño del *stack* de panqueques a los cuales se les hace *flip* (1, 2, 3 ó 4).

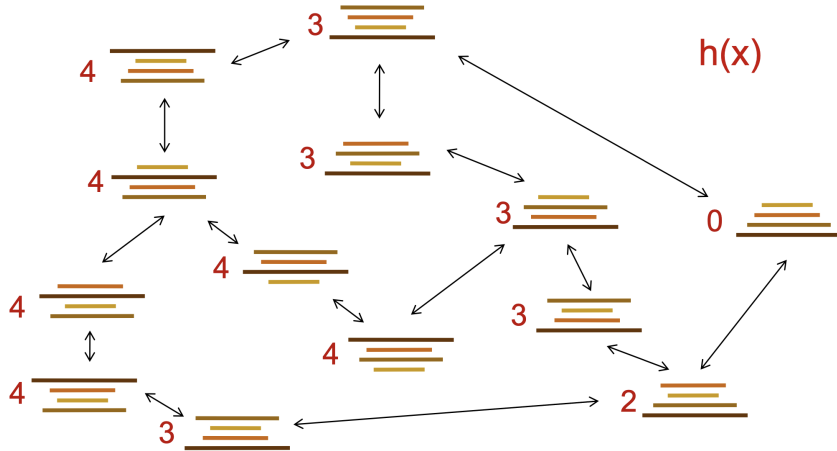
Heurísticas

Ejemplo: El problema de los panqueques.



Heurísticas

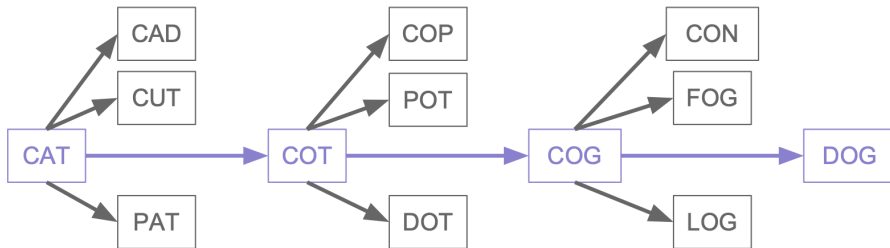
Ejemplo: El problema de los panqueques.



Heurísticas

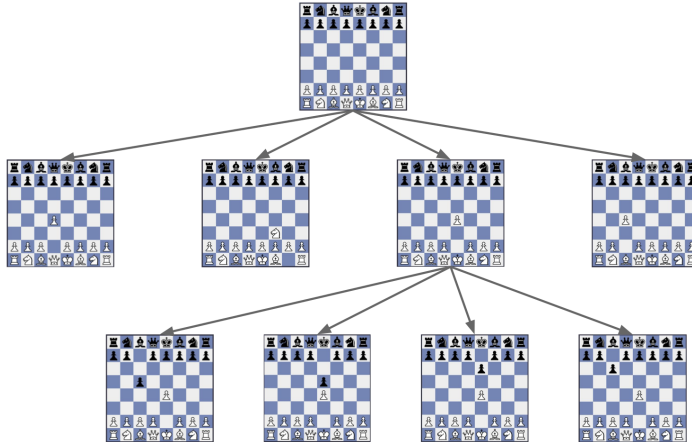
Ejemplo: Juego de mutar palabras

Convertir la palabra **cat** en **dog**.



Heurísticas

Ejemplo: Ajedrez.

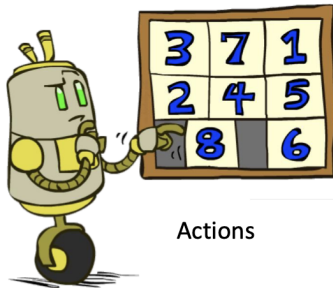


Heurísticas

Ejemplo: 8-puzzle.

7	2	4
5		6
8	3	1

Start State



Actions

	1	2
3	4	5
6	7	8

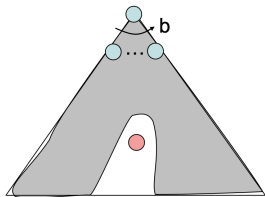
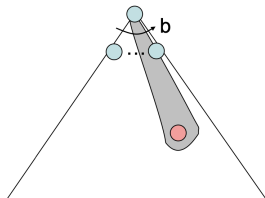
Goal State

Greedy Search

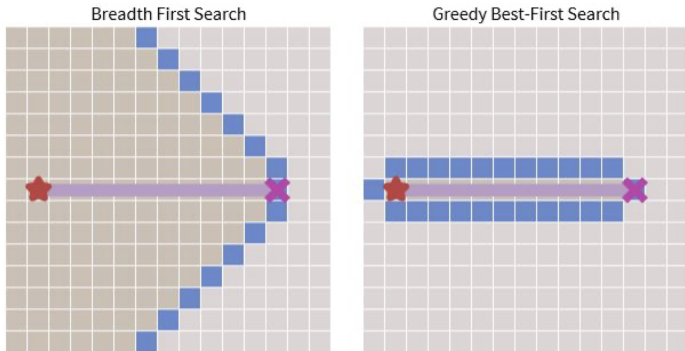
- Estrategia: expandir el nodo que uno cree está más próximo del estado objetivo. Cuidado! Esto se mide usando la función heurística h .
- Recordemos que h es apenas una aproximación de la distancia real al objetivo.

Casos comunes:

- Best-first: Greedy te lleva directo al estado objetivo (incorrecto).
- Peor-caso: Se comporta como un DFS mal guiado.



Greedy Search



Cuidado! *Greedy Search* busca y toma como primera opción la que se vea más prometedora (*best-looking*).



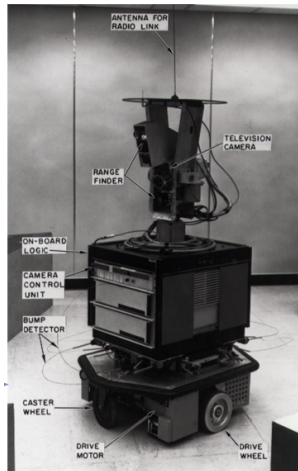
UCS



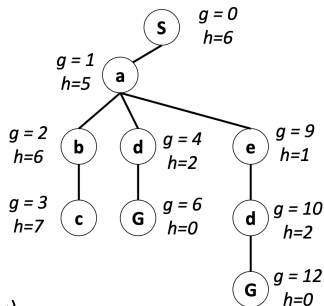
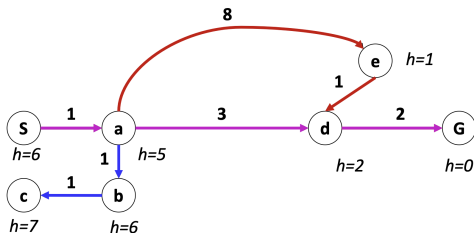
Greedy

 A^*

- Desarrollado por Peter Hart, Nils Nilsson, Bertram Raphael en Stanford (1968).
- Originalmente se desarrolló para ayudar a navegar a *Shakey the Robot* a través de una habitación con obstáculos.
- A* combina las fortalezas de BFS y de *Greedy Search*.
- Al igual que el BFS: este encuentra la ruta más corta.
Al igual que el *Greedy Search*: este método es rápido.



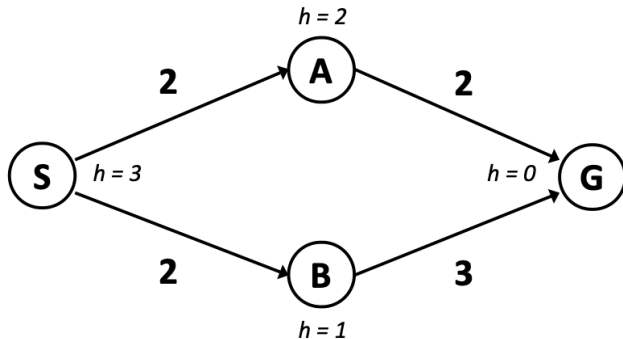
- **UCS (Costo uniforme)** ordena por costo pasado, o *backward cost* $g(V)$.
- **Greedy Search** ordena por proximidad al objetivo, o *forward cost* $h(V)$.



A* ordena con base en la suma $f(n) = g(n) + h(n)$.

¿Debemos detener el algoritmo A^* al encolar (*enqueue*) un objetivo?

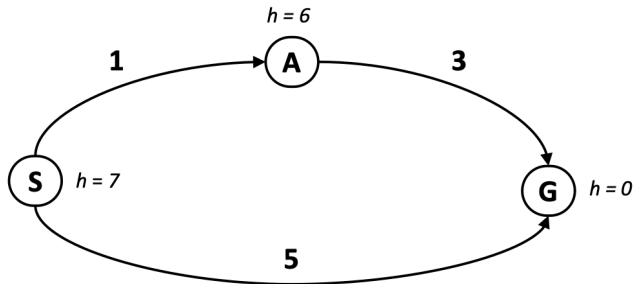
Respuesta: No.



La idea sería detener el algoritmos hasta que el objetivo se saque de la cola (*dequeue*).

¿Es A* optimal?

Respuesta: No siempre.



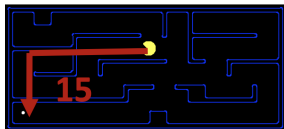
- ¿Qué puede salir mal?
- Costo real al objetivo estimado (worst) < Costo al objetivo (good)
- Requerimos que nuestros estimados sean menores que el costo verdadero!

Heurísticas Admisibles

Una heurística $h : V \rightarrow \mathbb{R}$ es **admissible** (u **optimista**) si

$$0 \leq h(v) < h^*(v), \text{ para todo } v \in V,$$

donde $h^*(v)$ es el costo real de llegar a un objetivo.



4



Comentario: Construir una heurística admisible es lo más importante (y a la vez lo más complicado) a la hora de implementar A^* en la práctica.

Teorema

Si la función heurística $h : V \rightarrow \mathbb{R}$ es un límite inferior para el costo del camino más corto hacia el objetivo, es decir

$$h(v, obj) \leq \text{ruta_mas_corta}(v, obj), \text{ para todo } v \in V,$$

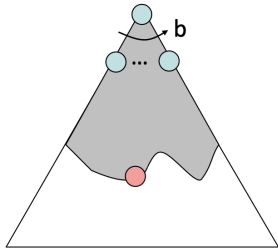
entonces el algoritmo A^ es óptima (siempre encuentra el camino más corto).*

Idea de prueba: la heurística h es optimista, por lo que nunca ignora un buen camino. A medida que se exploran todos los buenos caminos, descubrimos el camino óptimo.

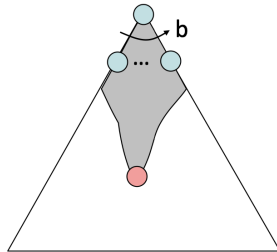
(Voy a agregar la prueba más detallada después).

Propiedades de A^*

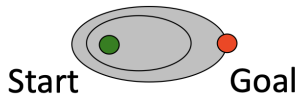
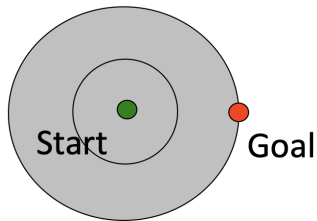
Uniform-Cost



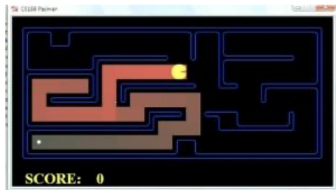
A^*



Propiedades de A^*



Propiedades de A^*



Greedy

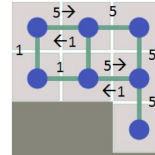
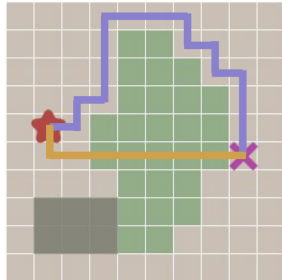


Uniform Cost



A^*

Weighted A^*



- Minimum number of steps
- Minimum cost path

Weighted A^*

Contraste entre A^* y *Weighted A^** :

- En el A^* regular, tenemos una función que prioriza como

$$f(v) = \underbrace{\text{pasos a partir de la fuente}}_{g(v)} + \underbrace{\text{costo aproximado al objetivo}}_{h(v)}.$$

- En el *weighted A^** , tenemos una función que prioriza como

$$f(v) = \underbrace{\text{costo a partir de la fuente}}_{c(v)} + \underbrace{\text{costo aproximado al objetivo}}_{h(v)}.$$

Aplicaciones

- Videojuegos
- Problemas de ruteo / planificación de rutas
- Problemas de distribución de recursos
- *Robot motion planning*
- Análisis de lenguaje
- Traducción de máquina
- *Speech recognition*
- ...

