Sistemas de Inteligencia Artificial

# Métodos de Búsqueda Informados

# Métodos de Búsqueda Informados

# Heurística

*¿Por qué buscamos "a ciegas"?*
*¿Tiene sentido podar caminos?*
*¿Se puede estimar cuanto falta para llegar al objetivo?*

# Heurística

Función **Heurística** h(e):

- Costo estimado de la ruta más barata desde el estado **e** hacia un estado meta.
- Si **e** es estado objetivo, **h(e) = 0**
- Si **e** no es un estado objetivo
  - Si **g(a) > 0** ; ∀ a: Acción
    - **h(e) > 0**
  - Si **g(a) ≥ 0** ; ∀ a: Acción
    - **h(e) ≥ 0**

“Abuso de notación”

**h(n) = h(e)** ; donde **e** es el estado representado en el nodo **n**.

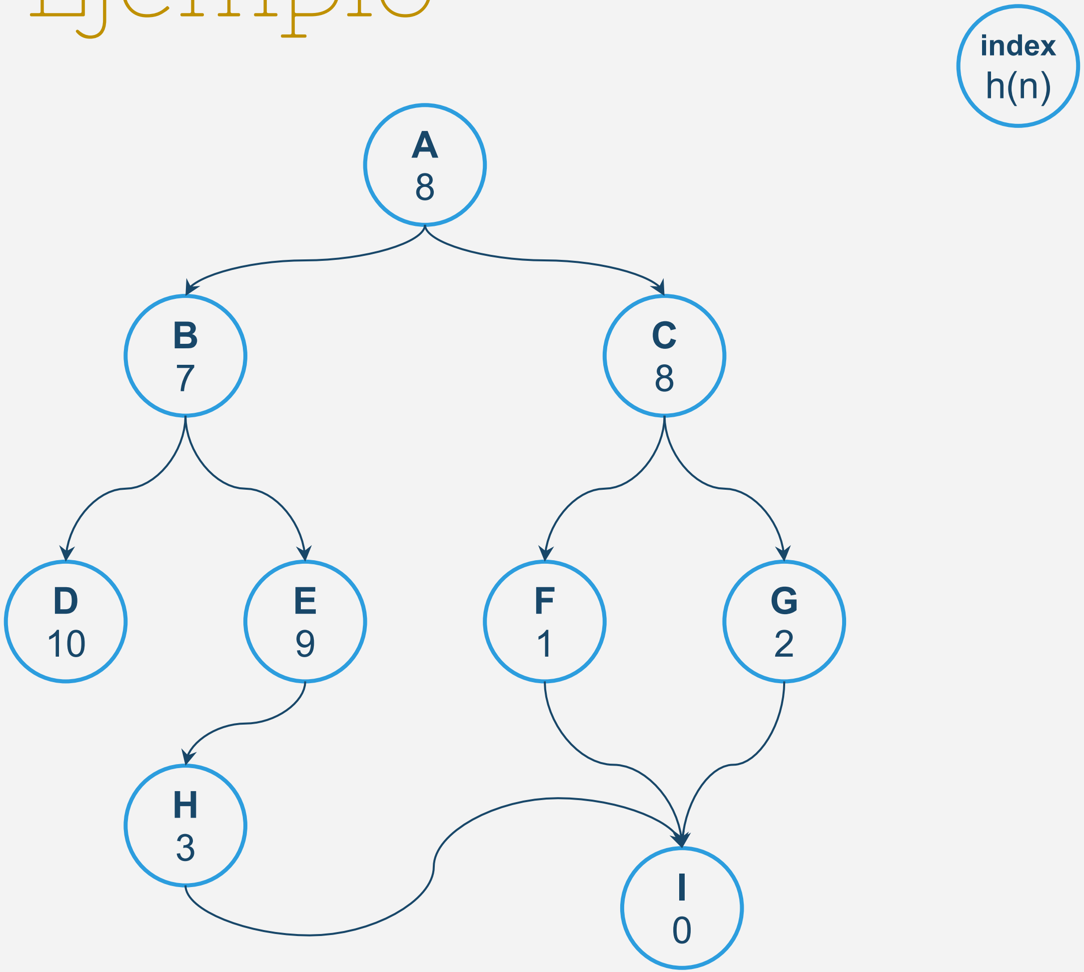
# Local Greedy Search

- Se comienza con el nodo raíz.
- Se expande, y de los recién expandidos se calcula **h**.
- Se toma el nodo con menor **h** y se expande.
- Si el conjunto recientemente expandido es vacío, se realiza backtracking.
- No es **óptima**.
- No es **completa** si no evalúa repetición de estados.
- Una buena función heurística reduce significativamente su complejidad temporal y espacial.
- Opera siempre sobre un conjunto acotado, lo que la hace muy veloz.

# Global Greedy Search

- Se utiliza el Algoritmo de búsqueda genérico, utilizando **h(n)** como ordenamiento de **Fr**.
- No es **óptima**.
- No es **completa** si no evalúa repetición de estados.
- Una buena función heurística reduce significativamente su complejidad temporal y espacial.
- Es un poco más costosa que LGS pero logra un backtracking más inteligente.

# Ejemplo



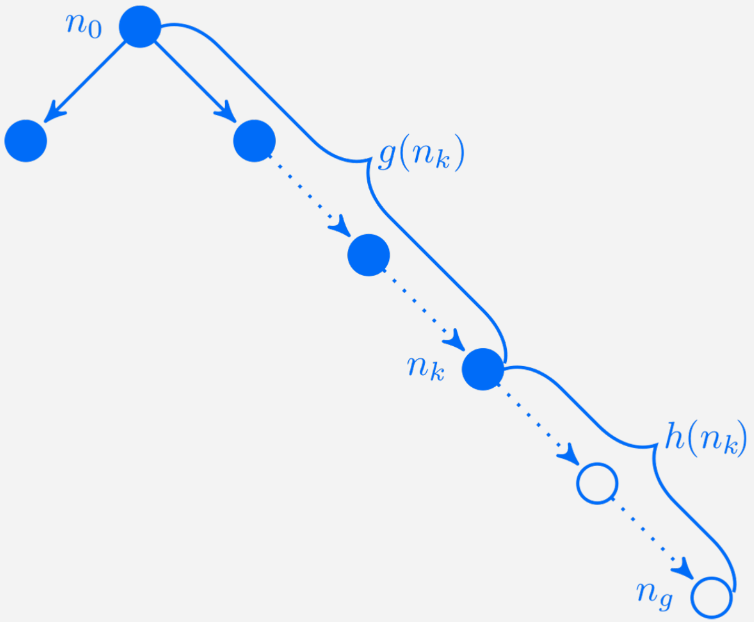
# Resumen

**Greedy Search**

- Busca expandir los nodos con mínimo **h(n)**, es eficiente pero no es ***óptima***.
- Minimiza **h(n)**

**Uniform Cost Search**

- Es ***óptima*** en un gran espectro de problemas, pero es ineficiente (similar a BFS).
- Minimiza **g(n)**



# Heurísticas

- Una heurística es **admisible** si nunca <u>sobreestima</u> el costo real.
- La heurística perfecta (que estima el costo real) se denomina **heurística estrella** : **$h^*(n)$**

# A* Search

- Se basa en la función
  **$f(n) = g(n) + h(n)$**
- Ordena a los nodos en frontera según esta función.
  - Si dos nodos tienen mismo **$f(n)$**, elegir el nodo con menor **$h(n)$**.
- Si **$h(n)$** es **admisible** $\Rightarrow$ **$f(n)$** nunca sobreestima el costo real de la mejor solución que pasa por el nodo **$n$**.
- Es **completa** si hay una ramificación finita y el costo es mayor que un $\varepsilon > 0$.
- Es **óptima** bajo ciertas circunstancias (más adelante…)
- Requiere memoria y procesamiento de heurísticas.

A* con una heurística $h_1$ se denomina **$A^*_1$** .

# Dominancia

## de métodos de búsqueda

- $\mathbf{M_1}$ domina a $\mathbf{M_2}$ si cada nodo expandido por $\mathbf{M_1}$ también es expandido por $\mathbf{M_2}$ .
- $\mathbf{M_1}$ domina **estrictamente** a $\mathbf{M_2}$ si:
  - $\mathbf{M_1}$ domina a $\mathbf{M_2}$
  - $\mathbf{M_2}$ no domina a $\mathbf{M_1}$

# ¿Cuándo es A* óptima?

- Cada nodo del grafo debe tener un número finito de sucesores.
- El costo de cada arco debe ser mayor que un $\varepsilon > 0$.
- La heurística debe ser **admisible**.
    - **$h(n) \leq h^*(n)$**

Estas tres condiciones <u>garantizan</u> que A* encuentre el camino óptimo a la solución, si existe.

Se define **$f^*(n) = g(n) + h^*(n)$**

# Lema 1

En cualquier momento dado en la búsqueda de A* (bajo las condiciones anteriores), existe un nodo **n** en **Frontera** que cumple que:

1. **n** está en el camino óptimo al objetivo.
2. **$f(n) \leq f^*(n)$**

En un camino óptimo no pueden existir “loops”, ya que de así serlo, deberían existir acciones cuyo costo sea nulo.

# Lema 1

## Caso Base: $n_0$

Para el nodo inicial $n_0$

1) Todos los caminos comienzan con $n_0$, y por existir una solución $\Rightarrow$ $n_0$ está en el camino óptimo al objetivo.

1) $\mathbf{f(n_0)} = g(n_0) + h(n_0) = h(n_0) \leq h^*(n_0) = \mathbf{f^*(n_0)}$

Definición    Definición

$g(n_0) = 0$    $g(n_0) = 0$

# Lema 1

## Paso Inductivo

1) La frontera en el paso K contenía a un nodo que estaba en el camino óptimo. Elijo nodo $\mathbf{n^e}$ para expandir.
   a) Si $\mathbf{n^e}$ no es tal nodo, la frontera va a seguir teniendo a dicho nodo ⇒ se cumple (1) y **n’** = dicho nodo
   b) Si $\mathbf{n^e}$ es tal nodo, A* lo expande, y alguno de sus hijos debería estar en el camino óptimo ⇒ se cumple (1) y $\mathbf{n' = n^e}$
2) $\mathbf{f(n')} = g(n') + h(n') \leq g(n') + h^*(n') = \mathbf{f^*(n')}$

Definición

Definición

$h(n') \leq h^*(n')$

# A* debe terminar

Partiendo de que la ramificación es finita, y el costo de las reglas/acciones son siempre mayor o igual a un número mayor a 0.

Si A* puede no terminar, entonces eventualmente llega a un punto donde **$f(n) > f^*(n)$ ∀ n** en Frontera.

# Eficiencia
## de heurísticas

Suponiendo $h_1$ y $h_2$ heurísticas admisibles, si $h_2$ **domina** a $h_1$
$\Rightarrow A^*_2$ expandirá menos nodos que $A^*_1$.

**¿Por qué?**

Existirán nodos que $A^*_1$ expandió (y no llevan al camino óptimo) y $A^*_2$ no, por lo que, para esos nodos:

$$g(n) + h_1(n) \leq f^*(n) \leq g(n) + h_2(n)$$

De esta forma, como regla general conviene tomar heurísticas **admisibles** <u>que den el mayor valor posible</u>.

# Combinación
## de heurísticas

Dado un conjunto de heurísticas admisibles $h_1, \ldots, h_m$ ; se puede definir como una nueva heurística **h’** a la combinación de ellas, definida de la siguiente forma:

$$\mathbf{h'} = \mathrm{Max}(h_1, \ldots, h_m)$$

Esta nueva heurística tiene como propiedades:

- Es **admisible**.
- Domina a todas las heurísticas que la conforman.

# Grafo de búsqueda

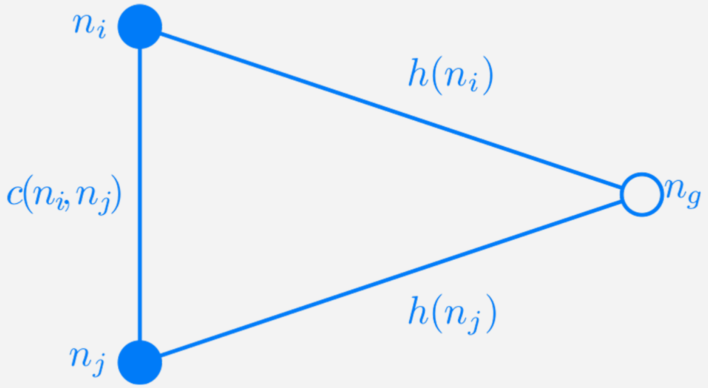
Se define **grafo de búsqueda** a un grafo que contiene como nodos los estados del problema, como aristas los costos de las acciones, donde un nodo será vecino de otro cuando haya una acción que conecte dichos estados.

# Consistencia
## de heurísticas

Si una heurística es **consistente**, cada vez que A* expanda un nodo, habrá encontrado un camino óptimo al mismo.

Si una heurística es **admisible**, no necesariamente.

# Consistencia
## de heurísticas




Sean $n_i$ y $n_j$ dos nodos en un grafo de búsqueda tal que $n_j$ es vecino de $n_i$ ; y sea **c(a, b)** el COSTO de la acción desde **a** hasta **b**.

- $h(n_i) \leq c(n_i , n_j) + h(n_j)$
- $h(n_g) = 0$

Si se puede demostrar esta propiedad para todo par de nodos vecinos para una heurística, esta será **consistente**.

- h **consistente** ⇒ h **admisible**
  - El sentido opuesto no siempre es cierto.
- También llamadas **monótonas** por que su demostración indica que la solución parcial es de costo creciente (no estrictamente).

# Completitud de A*

A* podría no terminar nunca.

Algunas de las situaciones donde puede suceder:

- La ramificación es infinita para al menos 1 estado.
- Existen arcos con costos ≤ 0
- Los arcos tienen costos > 0, pero son asintóticamente decrecientes (Paradoja de Zenon)

# Iterative Deepening A* (IDA*)

Concepto inspirado en IDDFS, se realiza un corte iterativo con un límite. También aquí se utiliza DFS.

Esta búsqueda es **completa** y **óptima** bajo las mísmas condiciones que A*.

Requiere menos memoria que A*, pero puede expandir más nodos e incluso expandir muchas veces el mismo nodo (mismo problema que IDDFS).

# Iterative Deepening A* (IDA*)

**Algoritmo**

El límite **Lím.** es un threshold para **f(n) = g(n) + h(n)**.

- Inicialmente, se toma **Lím. =** $\mathbf{f(n_0)}$.
- Mientras que no se encuentre solución, se realiza DFS hasta **Lím.**
- Si no se encontró una solución, se toma **n'** el nodo de frontera con menor valor de **f**, y se toma este valor como el nuevo límite.

# Heuristic Path Algorithm (HPA)

Best-First con **f(n) = (1-$w$)***g(n) **+ $w$***h(n)

- **$w$ = 0** ⇒ Uniform Cost Search
- **$w$ = ½** ⇒ A* Search
- **$w$ = 1** ⇒ Global Greedy Search

# Búsqueda de heurísticas

Algunas de las técnicas utilizadas para encontrar heurísticas son:

- Dividir al problema en sub-problemas.
- Relajar las limitaciones/reglas del problema.
- Tomar una heurística como el máximo de otras.
- Mezclar heurísticas, contemplando la valuación de cada una de ellas, sobre todo cuando atacan diferentes sub-problemas.

# Posibles heurísticas para...

- 8 reinas
- 8-puzzle
- Problema del vendedor ambulante
- Laberinto
- Misioneros y Caníbales

# Reparación Heurística

Se comienza con un estado, generalmente aleatorio, donde todas las variables fueron inicializadas.

El estado inicial se encuentra en violación de una o más restricciones (de no ser así, se tiene la solución).

Se realizan operaciones sobre el estado, de modificación, para tratar de solventar dichas restricciones.

Esta técnica es útil para cuando el problema es conocer el estado solución.