

Algoritmos-de-búsqueda-con-ejemp...



misimisi



Inteligencia Artificial



2º Grado en Ingeniería Informática



Escuela Técnica Superior de Ingenierías Informática y de
Telecomunicación
Universidad de Granada

CÓMO SERÍAS DE
SUPERHÉROE
O SUPERHÉROÍNA
SI FUERES MS MARVEL.

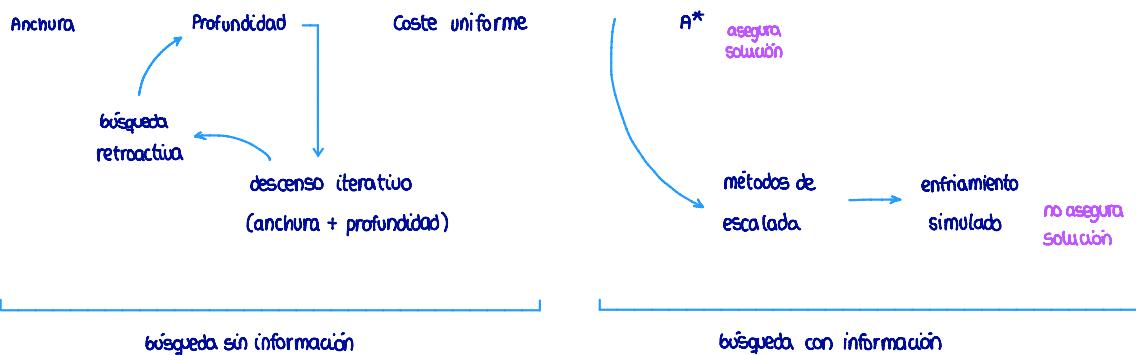
dibújate aquí

WWE STUDIOS
MS MARVEL

TOMARSE UN CAFELITO YA HECHO FEELS LIKE...

HACERSE CHULETAS EN LA CALCULADORA.

Resumen de los algoritmos de búsqueda en clase de prácticas



En Russell: Un enfoque moderno se llama "frontier" a ABIERTOS y "exploded" a CERRADOS.

CÓMO REPRESENTAR Y RESOLVER UN PROBLEMA DE BÚSQUEDA.

- ① Representación del concepto de estado (fila, columna, orientación)
- ②
 - 1 Estado inicial (fila, columna, orientación) del agente
 - 2 Estado meta (fila, columna, ^{irrelevante} x) del objetivo
 - 3 operadores para cambiar de un estado a otro
 - ① Avanzar
 - ② Ida 90°
 - ③ Dcha 90°

Con esto ya podemos abordarlo como un problema de búsqueda.

DECISIONES QUE TOMAR AL DISEÑARLO

- ④ Queremos el óptimo:
 - en acciones → anchura
 - en coste → coste uniforme o A* (con heurística admisible).
- ② No queremos óptimo: cualquier otro algoritmo (solo me interesa si hay solución).

```
struct Nodo{
    estado st;
    int coste;
    list<Action> plan;
}
```

Un nodo es un estado Enriquecido. Es el estado más información adicional necesaria para encontrar el camino.

100% NATURALES



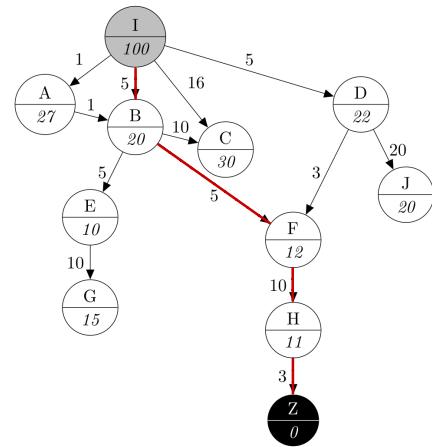
Búsqueda en anchura

Dilevela el camino con el mínimo número de operaciones.

* ABIERTOS es una cola.

* No tiene en cuenta el coste de los arcos (cada arco vale 1).

La estructura de un nodo es (nombre, padre).



ALGORITMO

1. Colocar el estado inicial en ABIERTOS

2. Coger el primer nodo de ABIERTOS

2.1 ¿Es solución?

→ Sí: He terminado y construyo el camino hacia atrás, encadenando padres e hijos en la lista de cerrados.

→ No: Meto el nodo en cerrados y lo expando (meto todos sus hijos en abiertos)

2.2 Vuelvo a 2

Notas importantes:

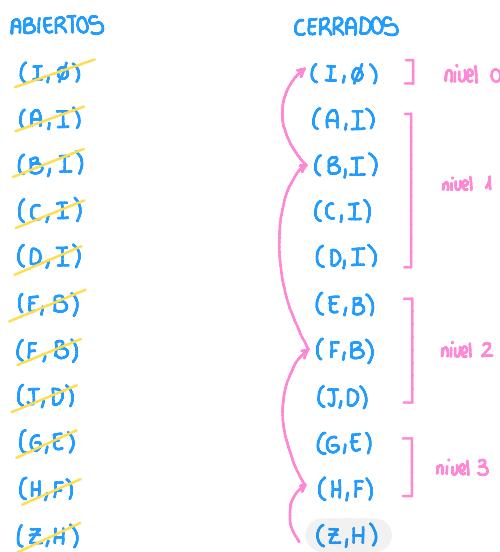
1. Si un nodo ya está en las listas, no se vuelve a meter (se omite).

2. En caso de empate, prioritar por el orden alfabético.

Acuerdo:

Cuando encontramos un nodo terminal (aparece en abiertos) vamos a terminar la búsqueda porque es algo trivial.

Proceso:



Solución:

I → B → F → H → Z.



yo elijo cerveza **SIN**

**Sea cual sea
el vehículo que
conduces, elige
cerveza SIN.**



WWW.CONDUCCIONRESPONSABLECERVEZASIN.COM



**UNA GRAN CERVEZA.
UNA GRAN RESPONSABILIDAD.**

© CONDUCCIÓN RESPONSABLE, CERVEZA SIN es una iniciativa de la Asociación de Cerveceros de España con el apoyo de la Dirección General de Tráfico.



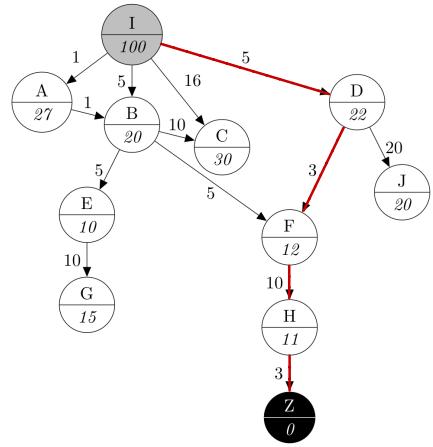
Búsqueda en profundidad

Asegura llegar a la solución (si la hay).

* ABIERTOS es una pila.*

* No tiene en cuenta el coste de los arcos (cada arco vale 1).

La estructura de un nodo es (nombre, padre).



(*) El profe suele invertir la lista (\downarrow), e invertir el orden al meter los nodos (\leftarrow) para que salga igual que en anchura, pero da igual.

ALGORITMO

1. Colocar el estado inicial en ABIERTOS

2. Coger el último nodo de ABIERTOS

2.1 ¿ES SOLUCIÓN?

→ Sí: He terminado y construyo el camino hacia atrás, encadenando padres e hijos en la lista de cerrados.

→ No: Meto el nodo en cerrados y lo expando (meto todos sus hijos en abiertos)

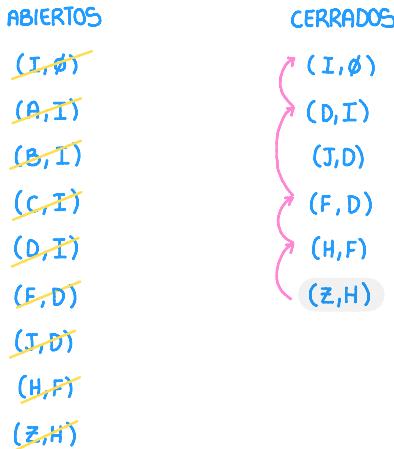
2.2 Vuelvo a 2

Notas importantes:

1. Si un nodo ya está en las listas, no se vuelve a meter (se omite).

2. En caso de empate, priorizar por el orden alfabético.

Proceso:



Solución:

I \rightarrow D \rightarrow F \rightarrow H \rightarrow Z.

**TOMARSE UN
CAFELITO YA HECHO
FEELS LIKE...**

**HACERSE CHULETAS
EN LA CALCULADORA.**

Búsqueda con coste uniforme

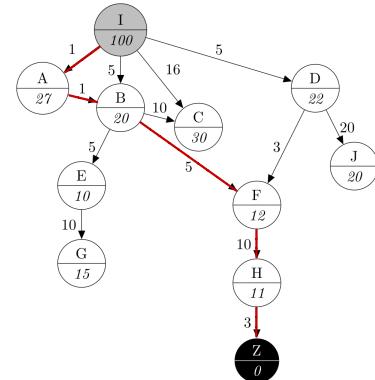
DIJKSTRA

Devuelve el camino con menor coste.

- * ABIERTOS es una cola con prioridad.
- * Si tiene en cuenta el coste de los arcos.

La estructura de un nodo es (nombre, coste, parente).

El coste se calcula como el coste del parente + coste del arco (operación), y representa lo que cuesta llegar hasta ese nodo desde el nodo inicial.



ALGORITMO

1. Colocar el estado inicial en ABIERTOS
2. Coger el nodo con menos coste de la lista de ABIERTOS.
 - 2.1 ¿Es solución?
 - Sí: He terminado y construyo el camino hacia atrás, encadenando padres e hijos en la lista de cerrados.
 - No: Meto el nodo en cerrados y lo expando (meto todos sus hijos en abiertos con sus respectivos costes).
 - 2.2 Vuelvo a 2

Notas importantes:

1. Si un nodo ya está en las listas, nos quedamos con el que menos coste tiene y omitimos el otro.
2. En caso de empate, prioritar por el orden alfabético.
3. Si en la lista de cerrados los costes no ascienden en valor, no lo estamos haciendo bien).
4. Si un nodo ya está en cerrados ya no lo puedo repetir.

Proceso:

ABIERTOS	CERRADOS
(I, 0, \emptyset)	
(A, 1, I)	(I, 0, \emptyset)
(B, 5, I)	(A, 1, I)
(C, 16, I)	(B, 2, A)
(D, 5, I)	(D, 5, J)
(B, 2, A)	(E, 7, B)
(C, 12, B)	(F, 7, B)
(E, 7, B)	(C, 12, B)
(F, 7, B)	(G, 17, E)
(F, 8, D)	(H, 17, F)
(J, 25, D)	(Z, 20, H)
(G, 17, E)	
(H, 17, F)	
(Z, 20, H)	

Solución:

I → A → B → F → H → Z con coste 20.

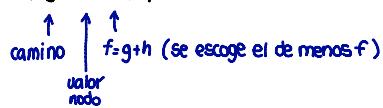
100% INGREDIENTES NATURALES



A*

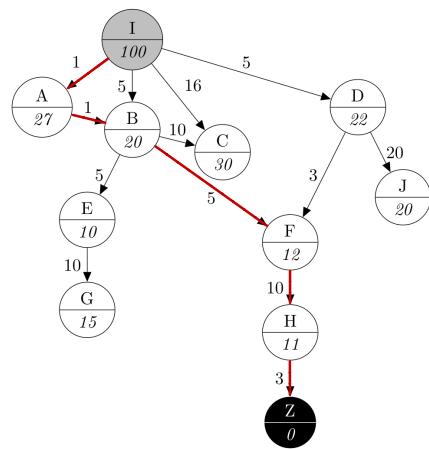
ABIERTOS

Notación: (Nodo, g, h, f, padre)



CERRADOS

(Nodo, g, padre)



Notas importantes:

- Si $h=0$ (no hay heurísticas, solo caminos), es coste uniforme.
- Es bueno ir poniendo las actualizaciones al lado y no abajo.

Proceso:
(no corregido)

ABIERTOS

(I, 0, 100, 100, \emptyset)

(A, 1, 27, 28, I)

(B, 5, 20, 25, I)

(C, 16, 30, 46, I)

(D, 5, 22, 27, I)

~~(E, 15, 30, 45, B)~~

(E, 10, 10, 20, B)

(F, 10, 12, 22, B)

~~(G, 20, 15, 35, E)~~

~~(H, 20, 14, 34, F)~~

(F, 8, 12, 20, D)

(J, 25, 20, 45, D)

~~(H, 18, 14, 29, F)~~

(B, 2, 20, 22, A)

(C, 12, 30, 42, B)

(E, 7, 10, 17, B)

(F, 7, 12, 19, B)

(G, 17, 15, 32, E)

(H, 17, 14, 28, F)

(Z, 20, 0, 20, H)

CERRADOS

(I, 0, \emptyset)

~~(B, 5, I)~~

~~(E, 10, B)~~

~~(F, 10, B)~~

(D, 5, I)

~~(C, 15, 30, 45, B)~~

~~(E, 10, 10, 20, B)~~

~~(F, 10, 12, 22, B)~~

~~(G, 20, 15, 35, E)~~

~~(H, 20, 14, 34, F)~~

~~(F, 8, 12, 20, D)~~

~~(J, 25, 20, 45, D)~~

~~(H, 18, 14, 29, F)~~

(B, 2, A)

(A, 1, I)

~~(E, 7, B)~~

~~(F, 7, B)~~

~~(H, 17, F)~~

~~(Z, 20, H)~~

Solución:

I → A → B → F → H → Z con coste 20.

¿Cuándo es admisible la heurística del A*?

Cuando se cumple $\forall n \quad h(n) \leq h^*(n)$ (lo de dentro del nodo \leq coste óptimo desde nodo n a nodo solución).

$h(H) = 11 \leq h^*(H) = 3$, luego la heurística no es admisible.

Con encontrar uno que no valga no nos sirve. Si uno si vale, seguimos buscando.

Si primero analizamos la raíz es más fácil:

Nuestra solución es 20 ($h^*(I) = 20$) y $h(I) = 100$, luego la heurística no es admisible.

Que una heurística sea admisible nos garantiza que la solución que nos da es óptima. Si no es admisible, no garantiza la mejor solución posible.

Descenso iterativo

Búsqueda en profundidad por niveles

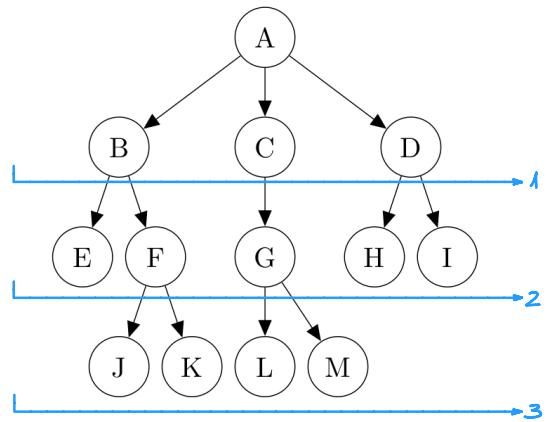
Se tiene que saber el grado de poda. Si no, asumimos que es 1.

Meto los hijos al revés porque es profundidad (consultar búsqueda en profundidad).

! Todo lo que está a más distancia que el grado de poda no lo tiene en cuenta.

Proceso:

	ABIERTOS	CERRADOS	
	(A, \emptyset)	(A, \emptyset)	nivel = 1
	(D, A)	(B, A)	
	(C, A)	(C, A)	
Vuelvo a hacer búsqueda en profundidad	(B, A)	(D, A)	iteración 1
<u> </u> →	(A, \emptyset)	(A, \emptyset)	nivel = 2
	(D, A)	(B, A)	
	(C, A)	(E, B)	
	(B, A)	(F, B)	
	(F, B)	(C, A)	
	(E, B)	(G, C)	
	(G, C)	(D, A)	
	(I, D)	(H, D)	
	(H, D)	(I, D)	iteración 2
	(A, \emptyset)	(A, \emptyset)	nivel = 3
	(D, A)	(B, A)	
	(C, A)	(E, B)	
	(B, A)	(F, B)	
	(F, B)	(J, F)	
	(E, B)	(K, F)	
	(K, F)	(C, A)	
	(J, F)	(G, C)	
	(G, C)	(L, G)	
	(M, G)	(M, G)	
	(L, G)	(D, A)	
	(D, A)	(H, D)	
	(H, D)	(I, D)	iteración 3



**TOMARSE UN
CAFELITO YA HECHO
FEELS LIKE...**

**HACERSE CHULETAS
EN LA CALCULADORA.**

Búsqueda retroactiva

Voy haciendo una búsqueda normal hacia abajo hasta que encuentra el nodo final.

Notas importantes:

- Si miro todas las secuencias, los nodos con " \circ " definen el orden de búsqueda.

Proceso:

A^0 ← los que he explorado hijos de este nodo

$A^1 B^0$

$A^1 B^1 E^0$

$A^1 B^1$ ↳ no tiene hijos, lo elimino

$A^1 B^2 F^0$

$A^1 B^2 F^1 J^0$

$A^1 B^2 F^1$

$A^1 B^2 F^2 K^0$

$A^1 B^2 F^2$

$A^1 B^2$

A^1

$A^2 C^0$

$A^2 C^1 G^0$

$A^2 C^1 G^1 L^0$

$A^2 C^1 G^1$

$A^2 C^1 G^2 H^0$

$A^2 C^1 G^2$

$A^2 C^1$

A^2

$A^3 D^0$

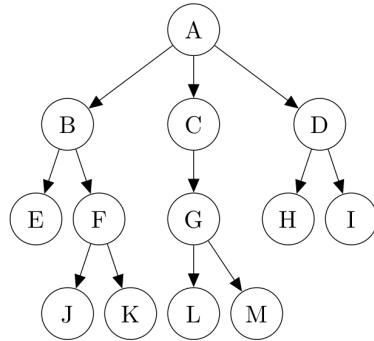
$A^3 D^1 H^0$

$A^3 D^1$

$A^3 D^2 I^0$

$A^3 D^2$

A^3

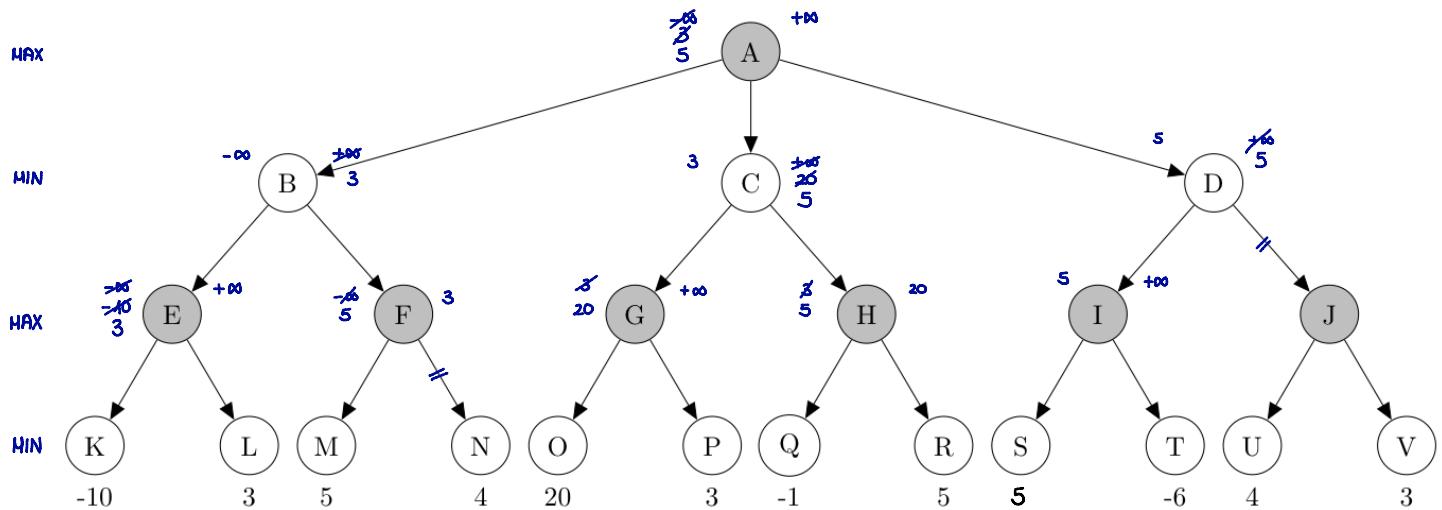


**100% INGREDIENTES
NATURALES**

Poda alfa-beta

Algoritmo:

1. Poner MAX y MIN en cada nivel.
2. Escribir en la raíz $\alpha = -\infty$ y $\beta = +\infty$
3. Cada vez que bajo a un nodo bajo la α y la β . Bajo hasta llegar al padre de un nodo hoja.
4. Para evaluar un nodo:
 - x Si el nodo es MAX, pongo en α (izda) el valor de su descendiente.
 - x Si el nodo es MIN, pongo en β (dcha) el valor de su descendiente.
5. Poda: si después de evaluar un nodo $\alpha \geq \beta$, no miro los demás descendientes del nodo (podo las ramas con //).
6. Para subir las evaluaciones hacia los nodos de arriba:
 - x Si el nodo de arriba es MAX, pongo el valor máximo entre α y β en la α del padre si el nuevo α es mayor que el α actual.
 - x si el nodo de arriba es MIN, pongo el valor mínimo entre α y β en la β del padre si el nuevo β es menor que el β actual.



7. El coste minimax:

- x Si el nodo raíz es MAX, es el α de la raíz.
- x Si el nodo raíz es MIN, es el β de la raíz.

Coste minimax = 5.