Búsqueda

Búsqueda ciega y búsqueda informada

Contenidos

- Formalización de la resolución de problemas
- Ejemplos
- Procedimiento general de búsqueda
- Estrategias de control
- Búsqueda heurística

Resolución de problemas: formalización

CONCEPTO BÁSICO: ESPACIO DE ESTADOS

(Influencia de la teoría de autómatas finitos)

< Q, R, C >

Q: representación de los posibles estados del problema (estructura de datos que describe el estado).

R: reglas u operaciones que describen las transiciones en el espacio de estados:

C: control.

$$R_Q \; X \; Q \rightarrow Q$$

PROCESO DE RESOLUCIÓN:

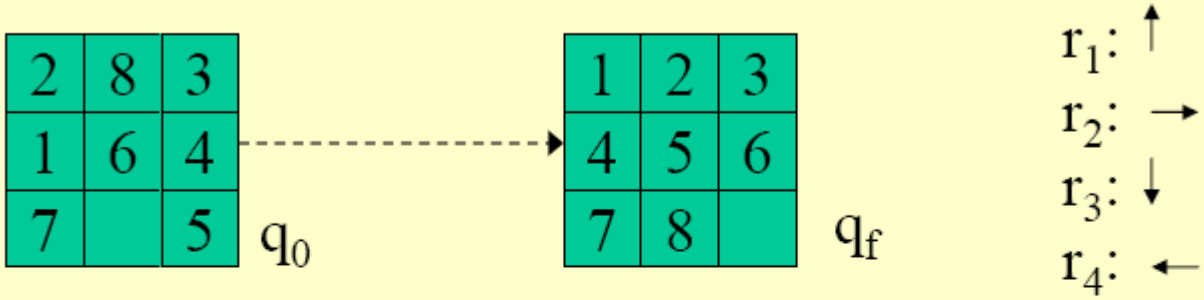
Encontrar r_1, r_2, ... Que conducen de q_0 a q_f

Ejemplos

- Problema del "8 puzzle":
  - Estados: Todas las posibles configuraciones de las fichas
  - Operaciones:
    - Una para mover cada ficha en cada una de las cuatro direcciones (¡muchas!)
    - Mejor: Una para mover el espacio en cada una de las cuatro direcciones: 4 operaciones.
  - Control: ??

Ejemplo 1º: El 8-puzzle

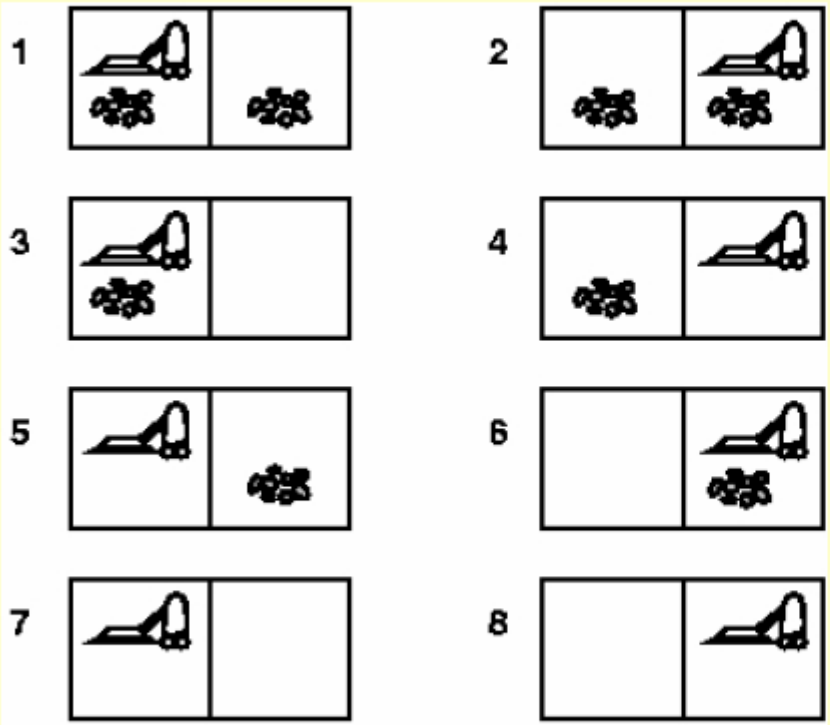
Problema del "8-puzzle":



$$card(Q) = 9! = 362.880$$

Ejemplo 2º: Robot aspirador

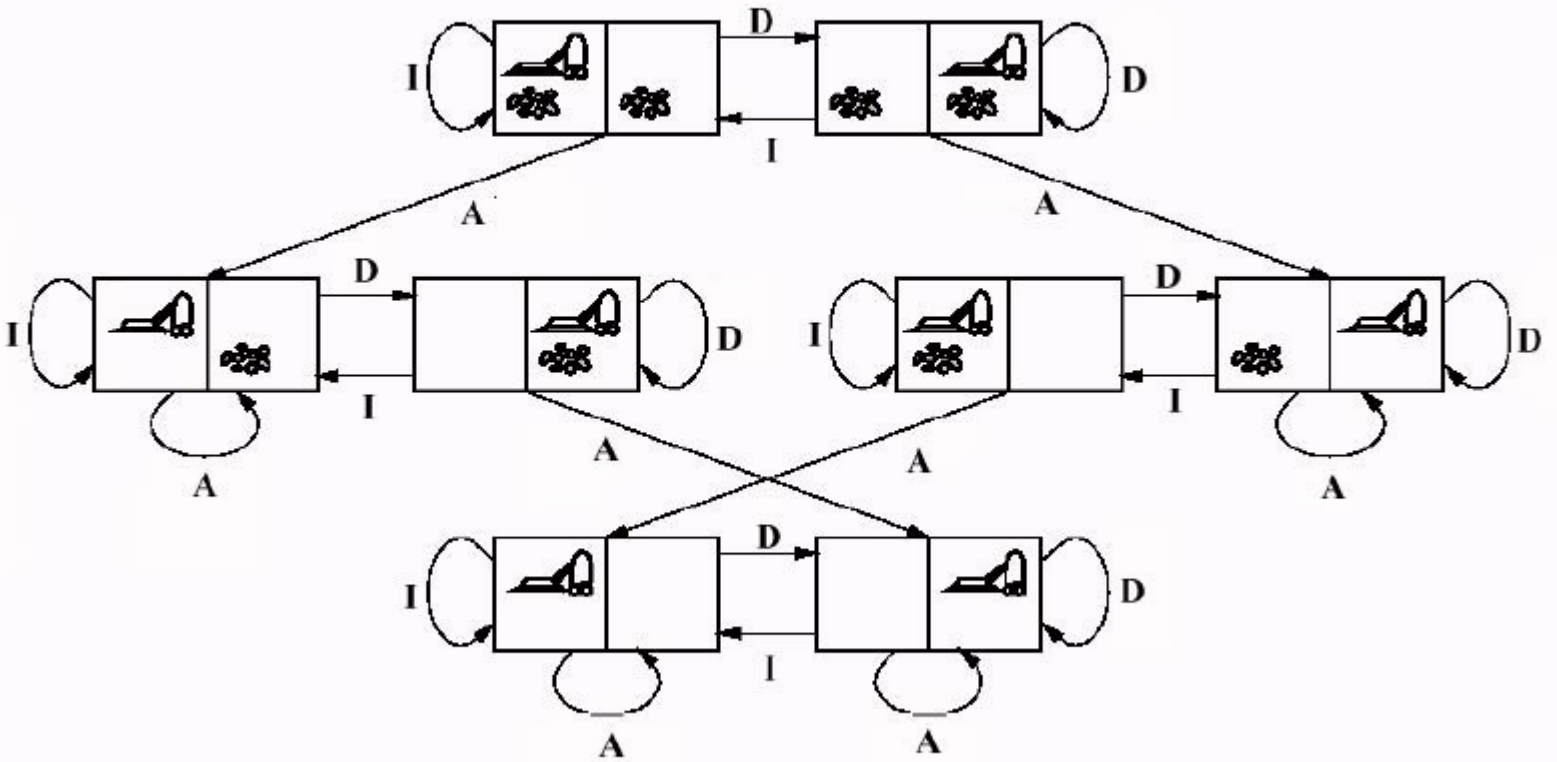
Problema del robot aspirador:



I: Ir a la izquierda
D: Ir a la derecha
A: Aspirar

Robot aspirador funcionando

Problema del robot aspirador:



Ejemplo 3º: Los cubos de agua

Problema de los "cubos de agua":

Descripción: Disponemos de 2 cubos, uno de 4 litros y otro de 3 y un grifo para llenarlos. Debemos consegir que el cubo de 4 litros tenga 2 litros de agua.

Estado: Par ordenado (x, y). $x=0,1,2,3,4$ $y=0,1,2,3$

Estado inicial: (0,0) Estafo final: (2,n)

Reglas:

$(x,y),\ x<4 \rightarrow (4,y)$	$(x,y),\ x>0 \rightarrow (0,y)$
$(x,y),\ y<3 \rightarrow (x,3)$	$(x,y),\ x+y\geq 4,\ y>0 \rightarrow (4,y-(4-x))$
$(x,y),\ x>0 \rightarrow (x-d,y)$	$(x,y),\ x+y\geq 3,\ x>0 \rightarrow (x-(3-y),y)$
$(x,y),\ y>0 \rightarrow (x,y-d)$	$(x,y),\ x+y\leq 4,\ y>0 \rightarrow (x+y,0)$
$(x,y),\ y>0 \rightarrow (x,0)$	$(x,y),\ x+y \leq 3,\ x>0 \rightarrow (0,(x+y))$

Una solución: $(0,0) \xrightarrow{2} (0,3) \xrightarrow{9} (3,0) \xrightarrow{2} (3,3) \xrightarrow{7} (4,2) \xrightarrow{5} (0,2) \xrightarrow{9} (2,0)$

Búsqueda en el espacio de Estados: procedimiento general

```
begin
    estado := est_inic
    while not estado = est_fin do
        begin
            regla := selecc(R, estado)
            estado := aplicación(regla, estado)
        end
end
```

regla := selecc (R, estado)

CONTROL

según sea, la búsqueda será más o menos "inteligente"

Estrategias de control: características fundamentales

- Debe causar que el problema avance.

  Ejemplo: en el problema de los "cubos de agua" no es válido seleccionar siempre la primera regla aplicable.

- Debe ser sistemático.

  Es decir, siempre debe producir la misma solución ante el mismo problema.

  Ejemplo: no es válido como estrategia de control elegir una operación al azar.

Tipos de búsqueda

- **ALGORITMO**

  Disponemos de información segura sobre qué operación aplicar

- **BUSQUEDA EXHAUSTIVA (A CIEGAS)**

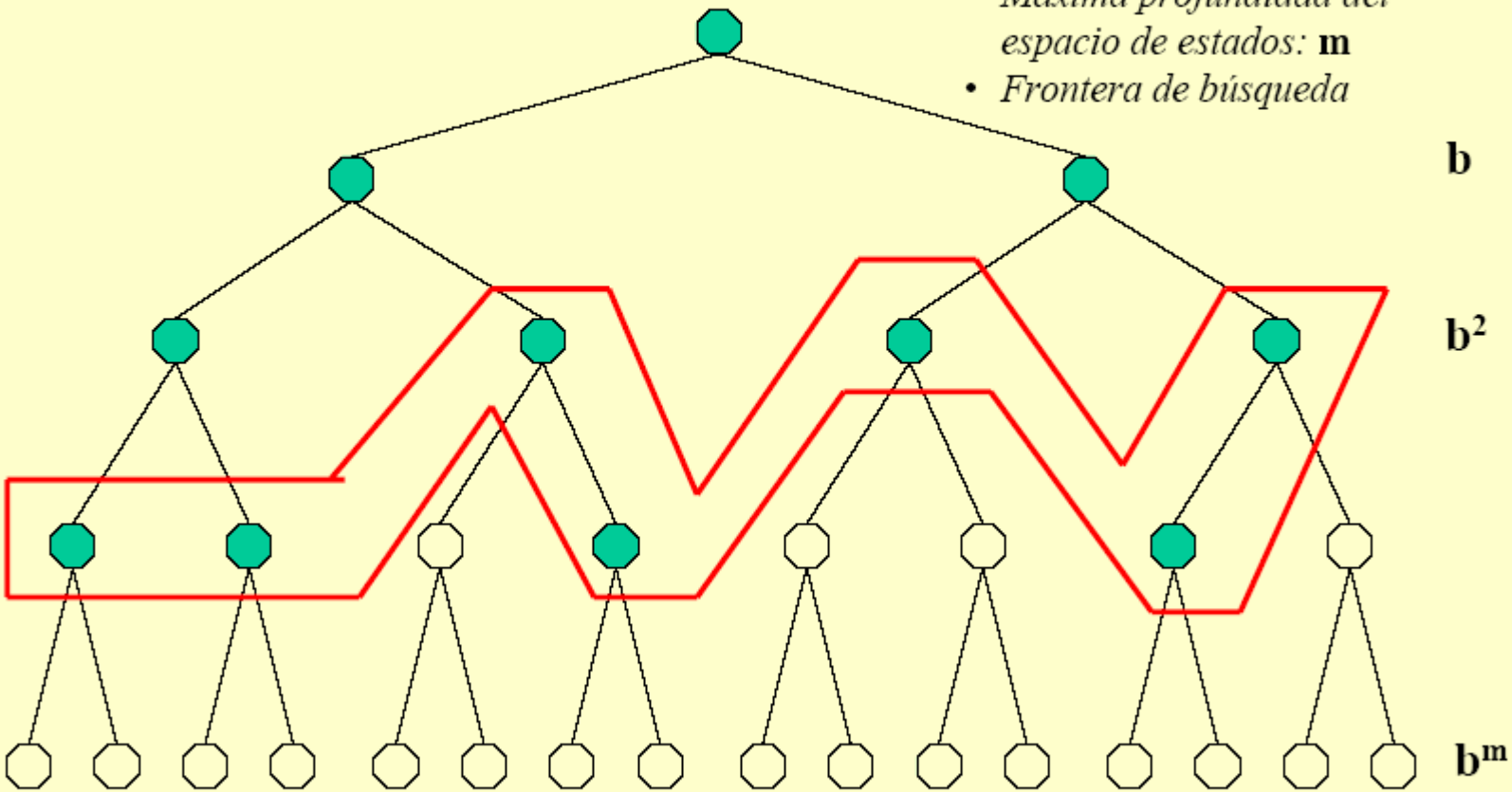
  Exploración del árbol de búsqueda sistemáticamente pero sin información

- **BUSQUEDA HEURÍSTICA (INFORMADA)**

  información sobre el problema (información del dominio) que permite reducir la búsqueda.

Árboles de búsqueda

- *Factor de ramificación:* **b**
- *Profundidad del nodo solución óptimo:* **d**
- *Máxima profundidad del espacio de estados:* **m**
- *Frontera de búsqueda*



Es decir: $O(b^m)$ nodos = Crecimiento exponencial

Búsqueda ciega: tipos

Estrategia de búsqueda: Elección del orden de expansión de los nodos del arbol

1. Búsqueda en profundidad
2. Búsqueda en amplitud
3. Búsqueda de coste uniforme
4. Búsqueda en profundidad limitada
5. Búsqueda en profundidad iterativa
6. Búsqueda bidireccional

Evaluación de estrategias

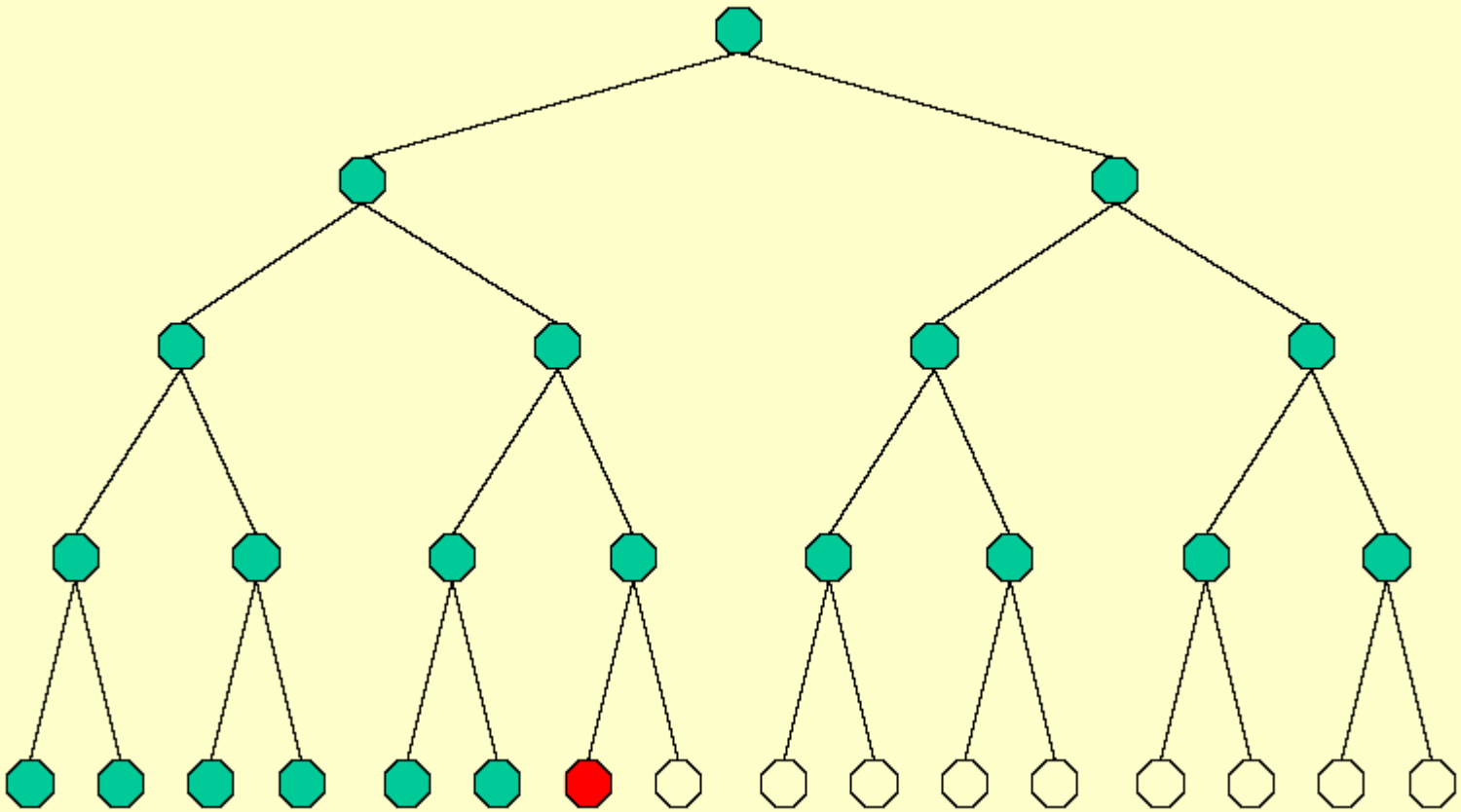
1. **¿Completa?:**
   Encuentra siempre la solución si existe

2. **Complejidad:**
   a) Temporal:
      Número de nodos explorados
   b) Espacial:
      Máximo número de nodos en memoria

3. **¿Óptimo?:**
   Encuentra siempre la mejor solución

Búsqueda en Amplitud

Se basa en desarrollar completamente cada nivel del árbol antes de pasar a desarrollar el siguiente.



Búsqueda en Amplitud: algoritmo

1- *lista* := *estado_inicial*

2- MIENTRAS (*lista* ≠ 0) y no solucion

  a- Eliminar primer elemento de *lista* y asignarlo a *E*

  b- Para cada regla aplicable a *E*

    i- Aplicar la regla

    ii- Si el estado resultante el objetivo, salir devolviéndolo

    iii- Si no, añadir el nuevo estado a *lista*

Evaluación de la búsqueda en amplitud

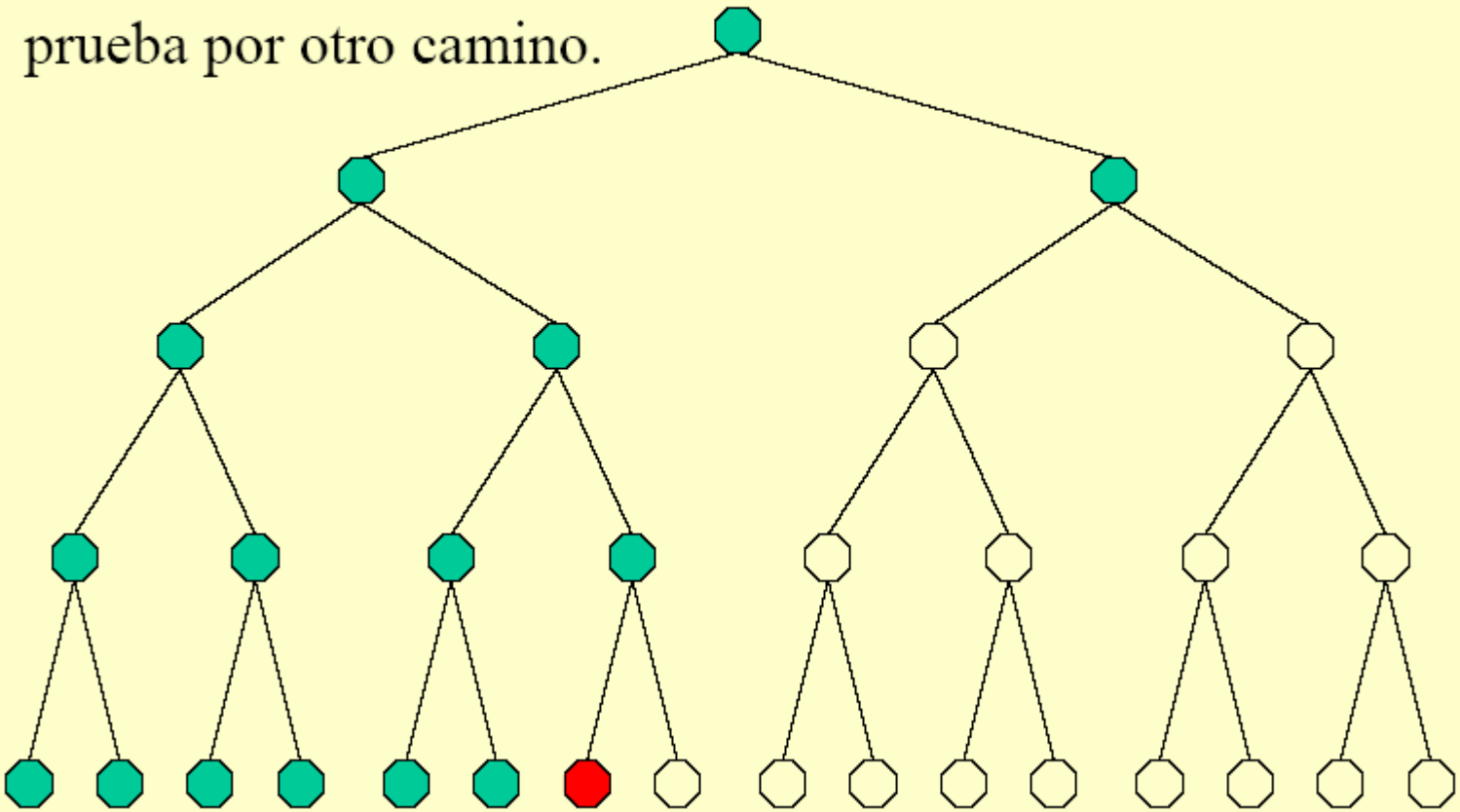
1. **¿Completa?:**
   Sí (si b es finito)

2. **Complejidad:**
   a) Temporal:
   $$1 + b^2 + b^3 + \ldots + b^d \Rightarrow O(b^d)$$
   b) Espacial:
   $O(b^d)$: exponencial, hay que guardar todos los nodos en memoria

3. **¿Óptimo?:**
   Sí (si el coste de cada paso es 1)

Búsqueda en profundidad

Se basa en elegir un camino en el árbol y seguirlo hasta el final. Si no se encuentra la solución se retrocede (*"backtraking"*) y se prueba por otro camino.



Búsqueda en profundidad

Se implementa mediante un algoritmo recursivo.

1- SI *estado_inicial* = *estado_objetivo* ENTONCES salir con éxito
2- MIENTRAS no éxito y no fracaso
    a- Generar los sucesores del estado inicial.
        SI no hay más sucesores ENTONCES fracaso
    b- Llamar al algoritmo para cada uno de los nodos generados como estado inicial
    c- Si se devuelve éxito, devolver éxito, si no continuar el bucle

Evaluación de la búsqueda en profundidad

1. **¿Completa?:**
   No (falla si hay espacios de profundidad infinita o bucles). Sí en espacios finitos

2. **Complejidad:**
   a) Temporal:
      $O(b^m)$ (m puede ser mucho mayor que d)
   b) Espacial:
      $O(bm)$: lineal

3. **¿Óptimo?:**
   No

Ventajas de la búsqueda en extensión

- No se "pierde" explorando caminos infructuosos que consumen mucho tiempo sin llegar a una solución o de los que no se vuelve nunca (bucles en profundidad).
- Si hay una solución la encuentra. Es más, si hay varias encuentra la óptima.

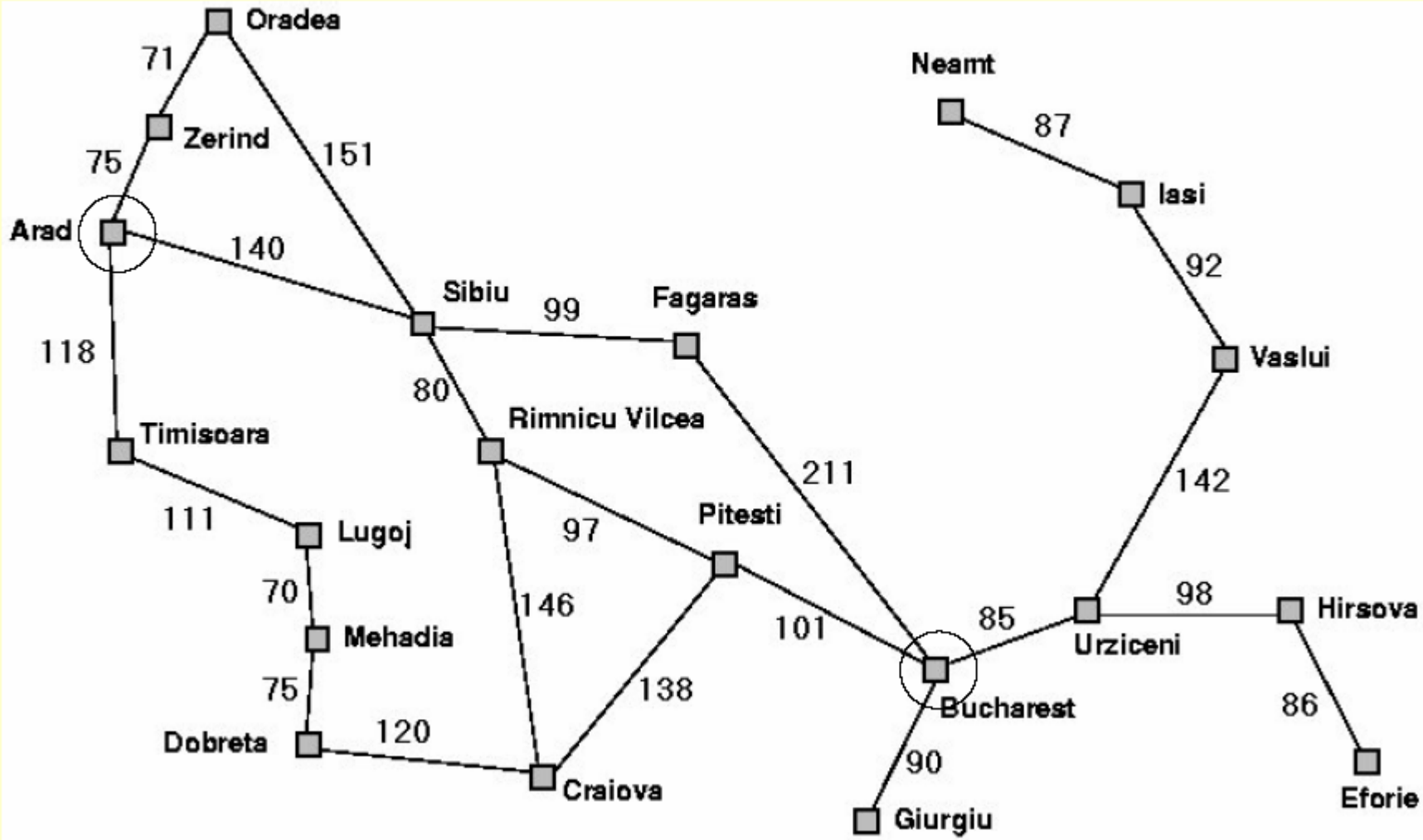
Ventajas de la búsqueda en profundidad

- Requiere mucha menos memoria (sólo hay que guardar el camino actual).
- Puede encontrar una solución sin examinar mucho el árbol, sobre todo si hay varios caminos a la solución.

Evaluación de estrategias

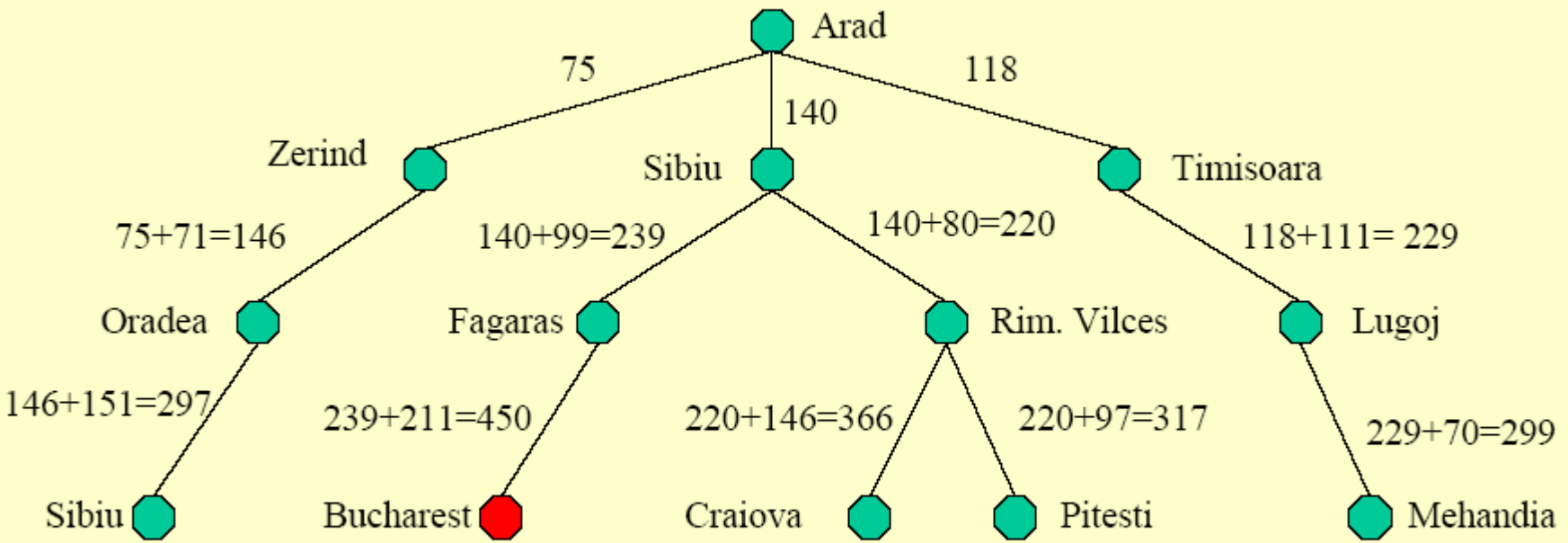
	Amplitud	Profundidad
Completa	Si	No
Complejidad Temporal	$O(b^d)$	$O(b^m)$
Complejidad Espacial	$O(b^d)$	$O(bm)$
Óptima	Si (con coste 1)	No

Búsqueda de coste uniforme



Ejemplo de búsqueda de coste uniforme

Se basa en desarrollar el nodo con menor coste.



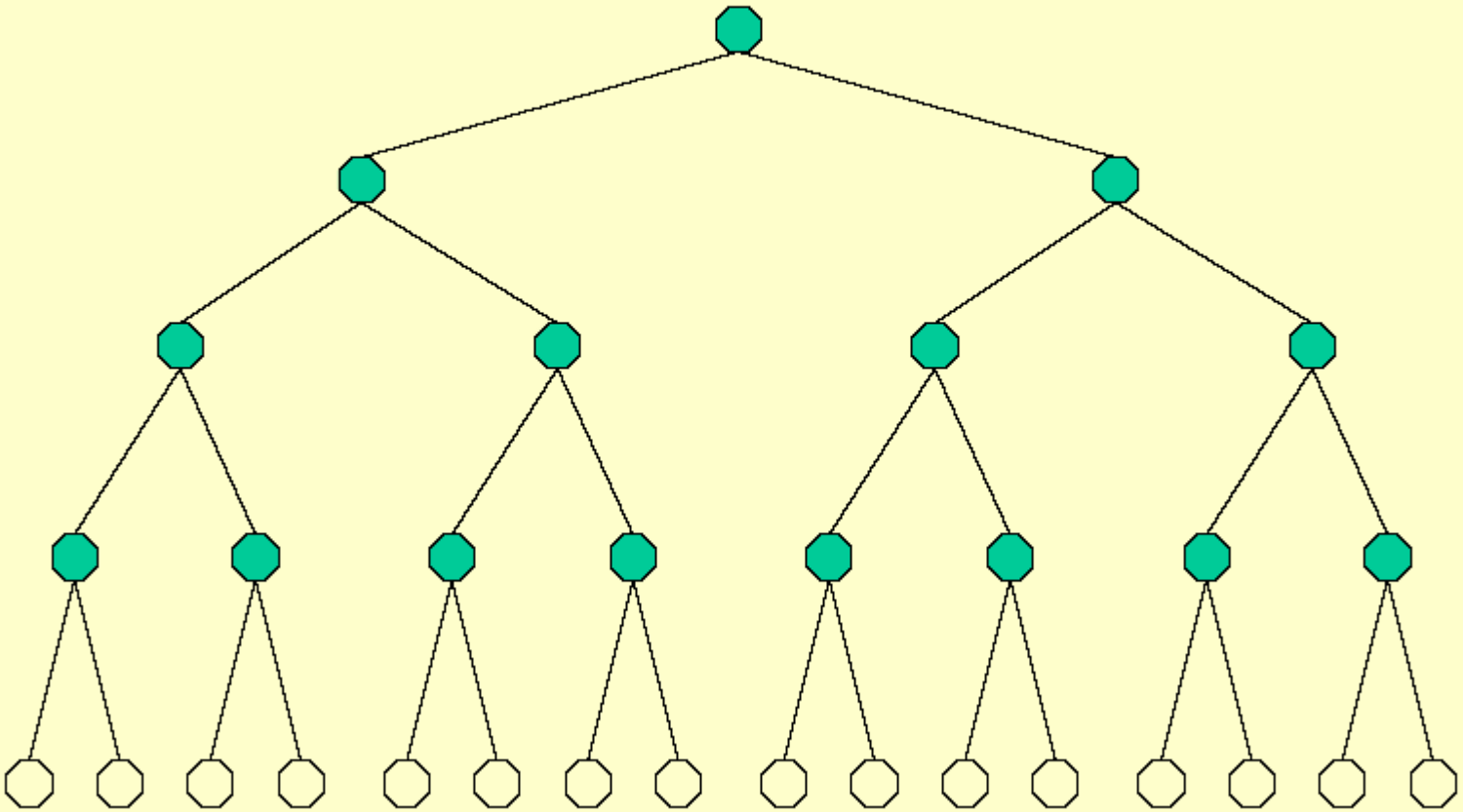
Evaluación de la búsqueda de coste uniforme

1. **¿Completa?:**
   Sí
2. **Complejidad:**
   a) Temporal:
      Número de nodos con $g \leq$ coste de la solución óptima: $O(b^d)$
   b) Espacial:
      Número de nodos con $g \leq$ coste de la solución óptima: $O(b^d)$
3. **¿Óptimo?:**
   Si

Búsqueda en profundidad limitada

Búsqueda en profundidad con un límite de profundidad *l*.

Implementación: los nodos a profundidad *l* no tienen sucesores



Búsqueda en profundidad limitada

Problema: calcular *l*.

- En ciertos problemas es fácil:
  Viaje por Rumanía: 20 ciudades: con $l = 19$ encontramos solución

- En otros imposible

Evaluación de la búsqueda en profundidad limitada

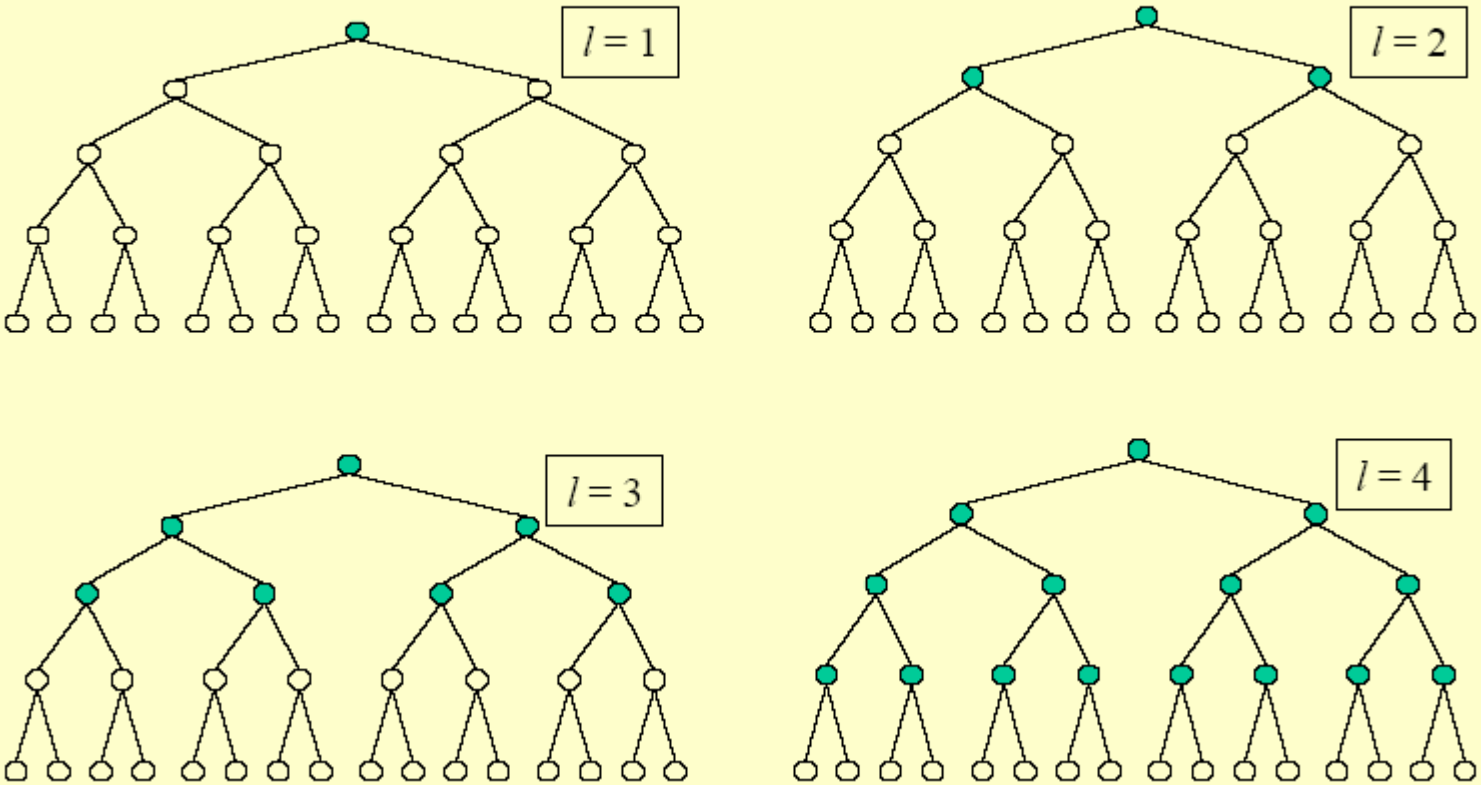
1. **¿Completa?:**
   No (falla si *l* es pequeño)

2. **Complejidad:**
   a) Temporal:
      $O(b^l)$
   b) Espacial:
      $O(bl)$

3. **¿Óptimo?:**
   No

Búsqueda en profundidad iterativa

Se basa en elegir utilizar búsqueda en profundidad limitada aumentando *l* si no se encuentra la solución.



Evaluación de la búsqueda en profundidad iterativa

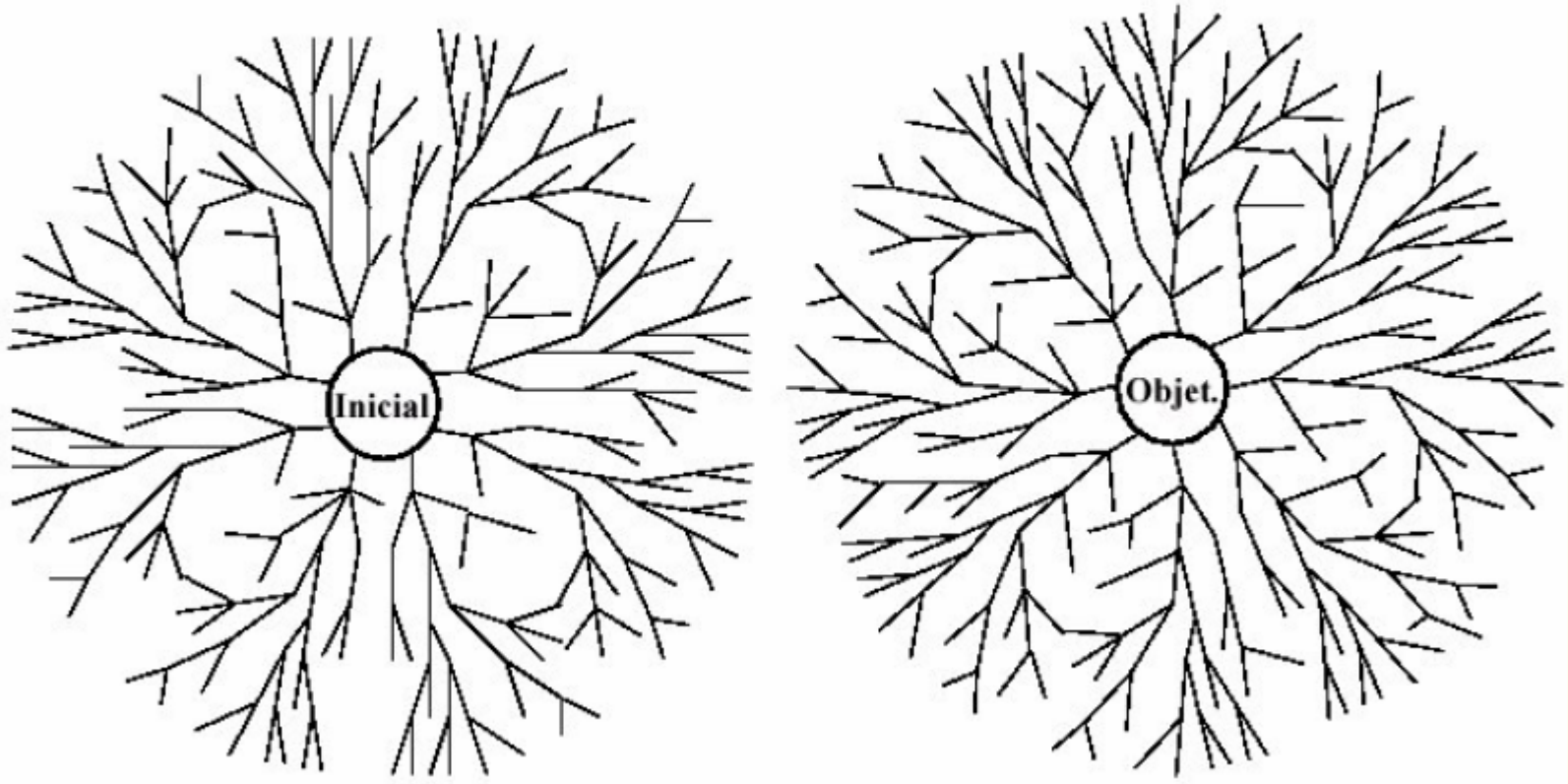
1. **¿Completa?:**
   Si

2. **Complejidad:**
   a) Temporal:
   $$(d+1)1 + db^2 + (d-1)b^3 + \ldots + b^d \Rightarrow O(b^d)$$
   b) Espacial:
   $$O(bd)$$

3. **¿Óptimo?:**
   Si (con coste 1). Se puede modificar para coste uniforme

Búsqueda bidireccional

Buscar simultáneamente desde el estado inicial y el final.



Evaluación de la búsqueda bidireccional

1. **¿Completa?:**
   Si

2. **Complejidad:**
   a) Temporal:
      $O(b^{d/2})$
   b) Espacial:
      $O(b^{d/2})$

3. **¿Óptimo?:**
   Si (con coste 1). Se puede modificar para coste uniforme

Búsqueda bidireccional: condicionantes

No vale para todos los problemas:

- Los operadores deben ser reversibles.
- Problema si hay varias soluciones.
- Debe haber comprobación eficiente de encuentro.

Comparación de estrategias

	Amplitud	Coste Uniforme	Profund.	Profund. Limitada	Profund. Iterativa	Bidirecc. (si se puede)
Complejidad Temporal	$O(b^d)$	$O(b^d)$	$O(b^m)$	$O(b^l)$	$O(b^d)$	$O(b^{d/2})$
Complejidad Espacial	$O(b^d)$	$O(b^d)$	$O(bm)$	$O(bl)$	$O(bd)$	$O(b^{d/2})$
Óptima	Si	Si	No	No	Si	Si
Completa	Si	Si	No	Si (si $l \geq d$)	Si	Si

Búsqueda ciega y explosión combinatoria

COMPLEJIDAD
(EXPLOSIÓN COMBINATORIA)

- **Espacial y temporal ¿Es tan importante?**

Ejemplo: $b = 10$ 1000 nodos/sg 100 bytes/nodo

Profundidad	Nodos	Tiempo	Memoria
0	1	1 ms.	100 bytes
2	111	100 ms.	1 Kb.
4	11.111	11 seg.	1 Mb.
6	10^6	18 min.	111 Mb.
8	10^8	31 horas	11 Gb.
10	10^{10}	128 días	1 Tb.
12	10^{12}	35 años	111 Tb.
14	10^{14}	3500 años	11.111 Tb.

Ejemplo de explosión combinatoria

Problema del viajante:

Descripción: Un viajante debe visitar varias ciudades (existen carreteras que unen todas las ciudades entre sí). El viajante desea encontrar el camino más corto que una todas las ciudades de tal manera que pase una sola vez por cada una de ellas.

Espacio de estados:

- Para 10 ciudades: 10! = 3.628.800 estados
- Para 25 ciudades: 25! = 15.511.210.043.330.985.984.000.000 estados (!!!)