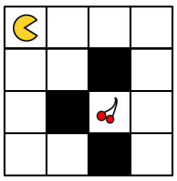
Búsqueda no informada / Búsqueda ciega

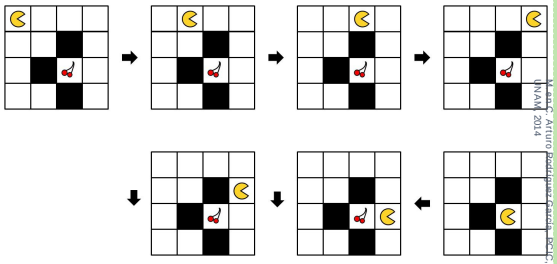
M. en C. Arturo Rodríguez García

Introducción

Búsqueda

- Proceso en el cual un agente construye una secuencia de acciones para alcanzar un objetivo.





Suposiciones del entorno

- Se tienen las siguientes suposiciones acerca del entorno:
  - Es estático
  - Es observable
  - Es discreto
  - Es determinista

Formular-Buscar-Ejecutar

- El primer paso es formular el problema (considerando el estado inicial, las acciones que puede realizar el agente, el objetivo y la medida de rendimiento).
- Un algoritmo de búsqueda recibe como entrada un problema y devuelve una secuencia de acciones para resolverlo.
- Una vez obtenida la solución se ejecutan las acciones.

Formular objetivo → Buscar solución → Ejecutar

- Las soluciones son simples secuencias de acción que no pueden manejar acontecimientos inesperados.
- Las soluciones se ejecutan sin prestar atención a las percepciones.

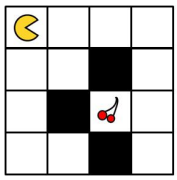
Definición de un problema

- Un problema se define formalmente por cuatro componentes:
  - Estado inicial
  - Función sucesor
  - Función objetivo
  - Función de costo

Estados y acciones

Estados = { a, b, c, d, e, f, h, i, k, l, m, n, p }

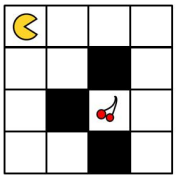
Acciones = {←, ↑, →, ↓}



a	b	c	d
e	f	g	h
i	j	k	l
m	n	o	p

Estado inicial

Estado inicial = a



a	b	c	d
e	f	g	h
i	j	k	l
m	n	o	p

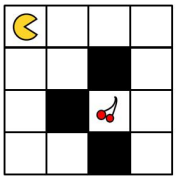
Función sucesor

- Dado un estado particular *x* devuelve un conjunto de pares ordenados <*a*,*s*>, donde cada *a* es una acción legal en el estado *x* y cada *s* es un estado que puede alcanzarse desde *x* aplicando la acción *a*.

Función sucesor

sucesor(a) = { <→, b>, <↓, e> }

sucesor(l) = { <←, k>, <↑, h>, <↓, p> }

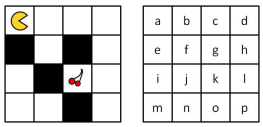


a	b	c	d
e	f	g	h
i	j	k	l
m	n	o	p

Espacio de estados

- Conjunto de todos los estados alcanzables desde el estado inicial.

  EspacioEstados = { a, b, c, d, f, h, k, l, p }

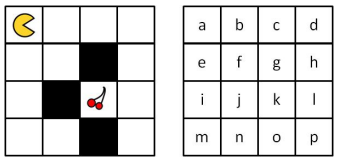


- El espacio de estados forma un grafo en el que los nodos son estados y los arcos son acciones.

M. en C. Arturo Rodríguez García, PCIC, UNAM, 2014

Caminos

- Secuencia de estados conectados por una secuencia de acciones.
  - $a \rightarrow b \rightarrow c \rightarrow d \downarrow h \downarrow l \downarrow p \uparrow l$ es un camino
  - $a \rightarrow b \downarrow m \rightarrow n \uparrow c \rightarrow d$ no es un camino



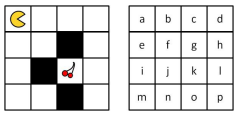
M. en C. Arturo Rodríguez García, PCIC, UNAM, 2014

Función objetivo

- Determina si un estado es un estado objetivo.

  test_objetivo(k) = true

  test_objetivo(f) = false



- En algunos problemas el objetivo se especifica como una propiedad más abstracta, por ejemplo, al jugar gato, el objetivo es tener tres de nuestros símbolos (X / O) alineados.

M. en C. Arturo Rodríguez García, PCIC, UNAM, 2014

Función de costo

- Asigna un costo numérico a cada camino, sumando los costos de las acciones individuales a lo largo del camino.
- Costos individuales:

  costo (x , a , y)

  Es el costo de la acción *a* para ir del estado *x* al estado *y*
- Supondremos que los costos son no negativos.

M. en C. Arturo Rodríguez García, PCIC, UNAM, 2014

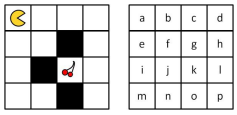
Función de costo

- Si suponemos que los costos son uniformes para todas las acciones, por ejemplo:

  costo(x , acción , y) = 1
- Entonces:

  costo_camino($a \rightarrow b \rightarrow c \rightarrow d \downarrow h \downarrow l \leftarrow k$) =

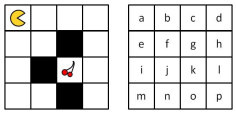
  costo($a, \rightarrow, b$) + costo($b, \rightarrow, c$) + costo($c, \rightarrow, d$) + costo($d, \downarrow, h$) + costo($h, \downarrow, l$) + costo($l, \leftarrow, k$) = 1 + 1 + 1 + 1 + 1 + 1 = 6



M. en C. Arturo Rodríguez García, PCIC, UNAM, 2014

Solución

- Una solución es cualquier camino desde el estado inicial al estado objetivo.
- Una solución óptima tiene el costo más pequeño de entre todas las soluciones.
  - $a \rightarrow b \rightarrow c \rightarrow d \downarrow h \downarrow l \downarrow p \uparrow l \downarrow p \uparrow l \leftarrow k$ es una solución, pero no es óptima
  - $a \rightarrow b \rightarrow c \rightarrow d \downarrow h \downarrow l \leftarrow k$ es una solución y es óptima



M. en C. Arturo Rodríguez García, PCIC, UNAM, 2014

Ejemplos de problemas

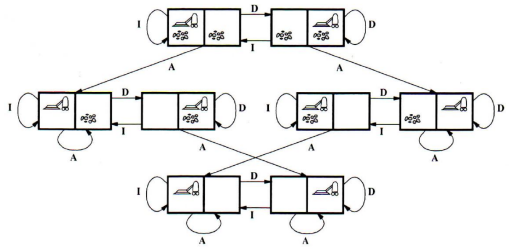
M. en C. Arturo Rodríguez García, PCIC, UNAM, 2014

Problemas de juguete

- Se utilizan para ilustrar o ejercitar los métodos de resolución de problemas.
- Tienen una descripción exacta y concisa, por lo que diferentes investigadores pueden usarlos para comparar el funcionamiento de los algoritmos.

M. en C. Arturo Rodríguez García, PCIC, UNAM, 2014

Mundo de la aspiradora



M. en C. Arturo Rodríguez García, PCIC, UNAM, 2014

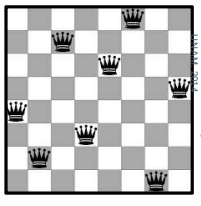
Mundo de la aspiradora

- Estados: el agente puede estar en una de dos localizaciones. Cada locación puede contener o no suciedad. Existen por lo tanto $2 \cdot 2^2 = 8$ posibles estados.
- Estado inicial: cualquier estado puede ser el inicial.
- Función sucesor: genera los estados legales al aplicar las tres acciones posibles (Izquierda, Derecha, Aspirar).
- Test objetivo: comprueba si todos los cuadros están limpios.
- Costo del camino: cada costo individual es 1, así que el costo del camino es el número de pasos que lo compone.

M. en C. Arturo Rodríguez García, PCIC, UNAM, 2014

Problema de las 8 reinas

- Objetivo.- Colocar las 8 reinas de manera que no se puedan atacar entre ellas.
- Formulación incremental.- empezar con el tablero vacío y en cada acción añadir una reina.
- Formulación completa de estados.- comenzar con las 8 reinas en el tablero y moverlas.



M. en C. Arturo Rodríguez García, PCIC, UNAM, 2014

Problema de las 8 reinas (formulación incremental)

- Estados: cualquier combinación de cero a ocho reinas en el tablero.
- Estado inicial: ninguna reina sobre el tablero.
- Función sucesor: añadir una reina a cualquier cuadrado vacío.
- Test objetivo: ocho reinas sobre el tablero, ninguna es atacada.
- Costo del camino: sin importancia (pues únicamente nos interesa el estado final).

Hay un total de 64! / 56! combinaciones por investigar.

M. en C. Arturo Rodríguez García, PCIC, UNAM, 2014

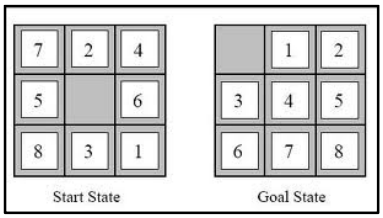
Problema de las 8 reinas (formulación incremental alternativa)

- Estados: la combinación de *n* reinas ($0 \leq n \leq 8$), una por columna desde la columna más izquierda, sin que ninguna ataque a otra.
- Función sucesor: añadir una reina en cualquier cuadrado en la columna más a la izquierda vacía tal que no sea atacada por cualquier otra reina.

Esta formulación reduce el espacio de estados a 2057.

M. en C. Arturo Rodríguez García, PCIC, UNAM, 2014

8-puzzle



M. en C. Arturo Rodríguez García, PCIC, UNAM, 2014

Problemas del mundo real

- Por lo general no tienen una única descripción.
  - Algoritmos de búsqueda de ruta aplicados a planificación de viajes en línea aéreas.
  - Distribución VLSI (colocar las celdas en el chip de manera que no se superpongan y que quede espacio para las conexiones).
  - Encontrar la secuencia correcta para el ensamblado automático.
  - Etc.

M. en C. Arturo Rodríguez García, PCIC, UNAM, 2014

Búsqueda de soluciones

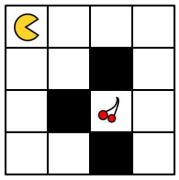
M. en C. Arturo Rodríguez García, PCIC, UNAM, 2014

Árbol de búsqueda

- El árbol de búsqueda es generado por el estado inicial y la función sucesor, definiendo el espacio de estados.
- Si un mismo estado puede alcanzarse desde varios caminos, entonces en realidad tenemos un grafo de búsqueda.
- El estado a expandir es determinado por la estrategia de búsqueda que se esté utilizando.

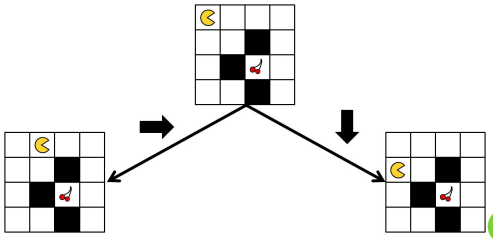
M. en C. Arturo Rodríguez García, PCIC, UNAM, 2014

- Partimos del estado inicial. Si es una solución, termino. De lo contrario se expande.

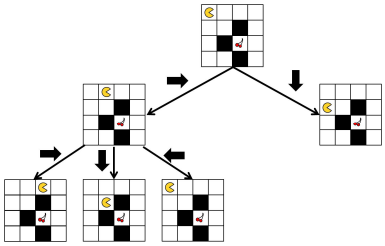


M. en C. Arturo Rodríguez García, PCIC, UNAM, 2014

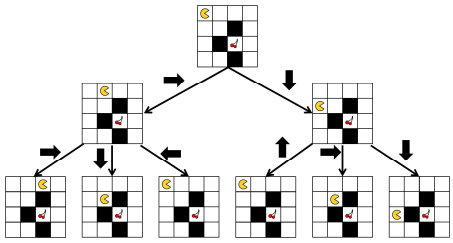
- Expandimos el estado inicial aplicando la función sucesor.



- Elegimos un nuevo nodo. Si es una solución, terminamos. De lo contrario lo expandimos. (Por ejemplo, el hijo izquierdo del estado inicial).



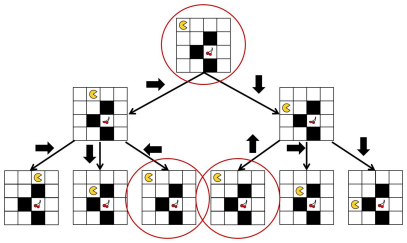
- Elegimos otro nodo. Si es una solución, terminamos. De lo contrario lo expandimos. (Por ejemplo, el hijo derecho del estado inicial).



Nodos vs estado

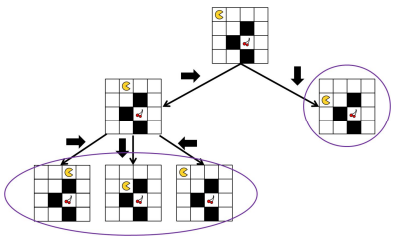
- Un nodo es la estructura de datos usada para representar el árbol de búsqueda. Están en caminos particulares del árbol de búsqueda.
- Los estados son configuraciones del mundo. Son elementos del espacio de estados.

- No confundir entre espacio de estados y árbol de búsqueda. Un mismo estado puede aparecer en varios nodos del árbol.



Frontera

- Conjunto de nodos generados que todavía no se han expandido.



Rendimiento de un algoritmo

- Completitud.- ¿está garantizado que el algoritmo encuentre una solución cuando esta exista?
- Optimización.- ¿encuentra la solución óptima?
- Complejidad en tiempo.- ¿cuánto tarda en encontrar la solución?
- Complejidad en memoria.- ¿cuánta memoria utiliza?

Complejidad

- Se expresa en término de tres cantidades:
  - *b* Factor de ramificación (es el máximo número de sucesores de cualquier nodo).
  - *d* Profundidad del nodo objetivo más superficial.
  - *m* Longitud máxima de cualquier camino en el espacio de estados.
- El tiempo se mide en términos del número de nodos generados.
- El espacio se mide en términos del máximo número de nodos que se almacenan en memoria.

Estrategias de búsqueda no informada

Búsqueda no informada (ciega)

- Únicamente tiene como información la formulación del problema.
- Lo único que puede hacer es generar los sucesores y distinguir si un estado corresponde a un objetivo.
- Existen distintas estrategias que se distinguen por el orden de expansión de los nodos.

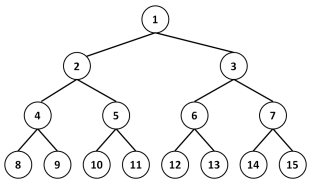
Búsqueda no informada (ciega)

- Búsqueda primero en anchura
- Búsqueda de costo uniforme
- Búsqueda primero en profundidad
- Búsqueda de profundidad limitada
- Búsqueda con profundidad iterativa
- Búsqueda bidireccional

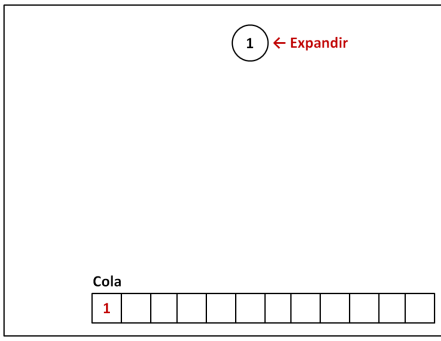
Búsqueda primero en anchura

Búsqueda primero en anchura

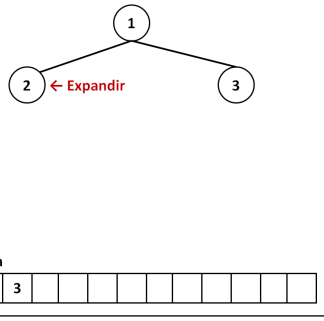
- Expande el nodo más superficial. Por lo tanto, expande todos los nodos a una profundidad en el árbol de búsqueda antes de expandir cualquier nodo del siguiente nivel.
- Utiliza una cola (FIFO).



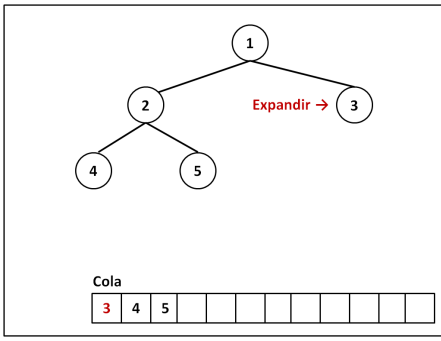
M. en C. Arturo Rodríguez García. PCIC. UNAM. 2014



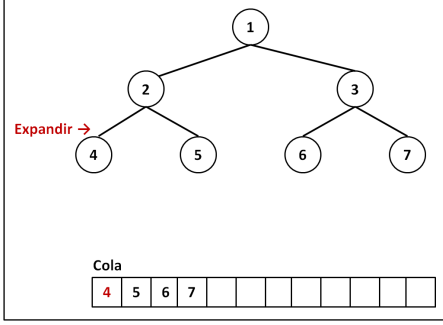
M. en C. Arturo Rodríguez García. PCIC. UNAM. 2014



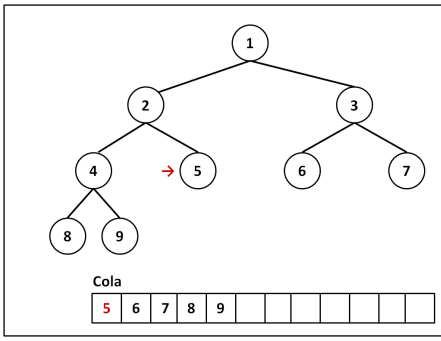
M. en C. Arturo Rodríguez García. PCIC. UNAM. 2014



M. en C. Arturo Rodríguez García. PCIC. UNAM. 2014

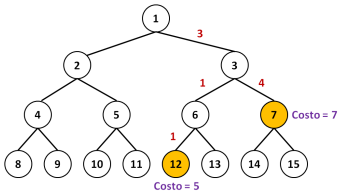


M. en C. Arturo Rodríguez García. PCIC. UNAM. 2014



M. en C. Arturo Rodríguez García. PCIC. UNAM. 2014

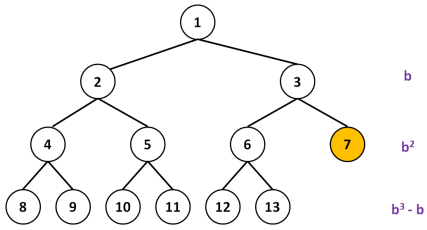
- ¿Es completo? Sólo si *b* es finito.
- ¿Es óptimo? Sólo si el costo del camino es una función no decreciente de la profundidad del nodo (por ejemplo, cuando todas las acciones tienen el mismo costo).



M. en C. Arturo Rodríguez García, PCIC, UNAM, 2014

- Complejidad en tiempo:

$b + b^2 + b^3 + \ldots + b^d + (b^{d+1} - b) = O(b^{d+1})$



M. en C. Arturo Rodríguez García, PCIC, UNAM, 2014

- Complejidad en espacio: la misma que la complejidad en tiempo (más un nodo para la raíz).

$1 + b + b^2 + b^3 + \ldots + b^d + (b^{d+1} - b) = O(b^{d+1})$

- El algoritmo es poco práctico debido a su complejidad exponencial.

M. en C. Arturo Rodríguez García, PCIC, UNAM, 2014

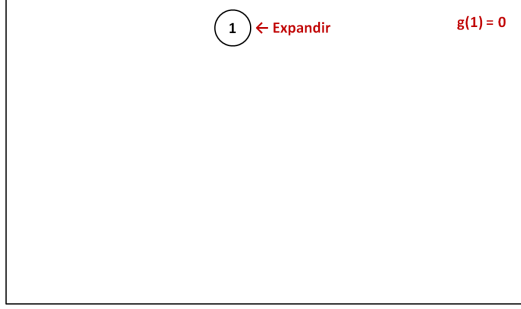
Búsqueda de costo uniforme

M. en C. Arturo Rodríguez García, PCIC, UNAM, 2014

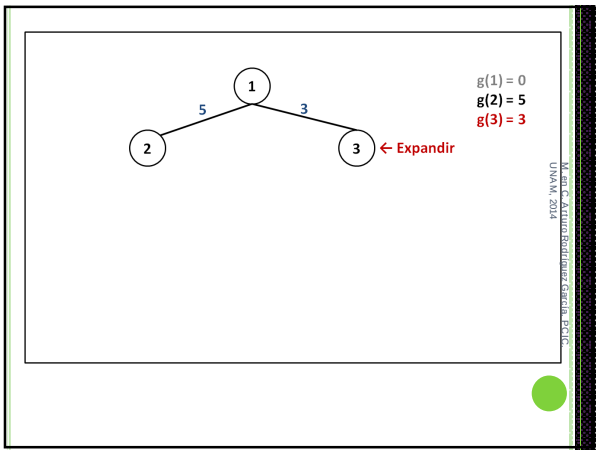
Búsqueda de costo uniforme

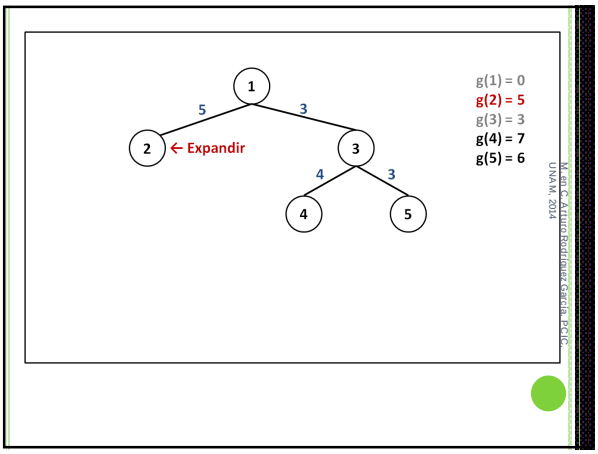
- Expande el nodo con el camino de costo más pequeño.
- Si todos los costos son iguales, es idéntico a la búsqueda primero en anchura.

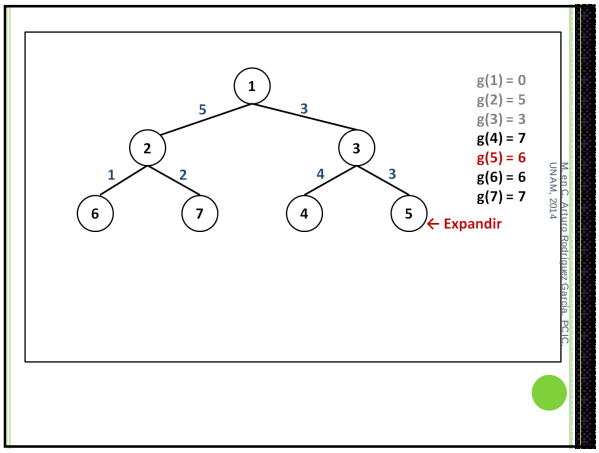
M. en C. Arturo Rodríguez García, PCIC, UNAM, 2014

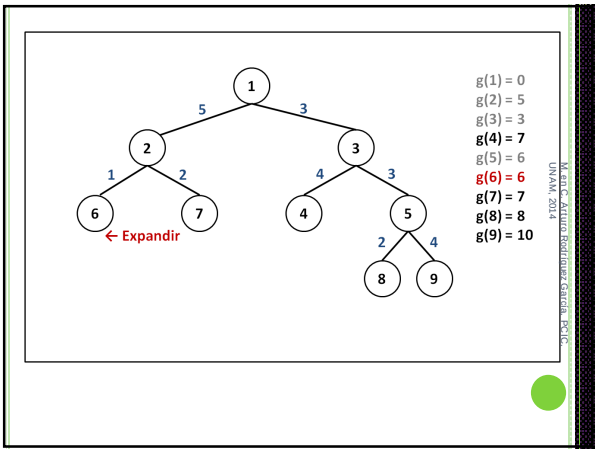


M. en C. Arturo Rodríguez García, PCIC, UNAM, 2014









- Este algoritmo se puede meter en un bucle infinito si expande un nodo con costo cero que lleva al mismo estado.
- Es completo y óptimo si el costo de cada paso es mayor o igual a alguna constante positiva ε. (Esto último significa que el costo de un camino siempre aumenta mientras vamos sobre él).

M. en C. Arturo Rodríguez García. PCIC. UNAM. 2014

- La complejidad no puede ser caracterizada fácilmente en términos de *b* y *d*. En su lugar, C* es el costo de la solución óptima y se supone que cada acción cuesta al menos ε.
- Entonces, la complejidad en tiempo y en espacio es $O(b^{[C^*/\varepsilon]})$, lo que puede ser mucho más grande que $O(b^d)$.
- La gran desventaja del algoritmo es que explora árboles grandes en pequeños pasos antes de explorar caminos que implican pasos grandes pero quizá útiles.

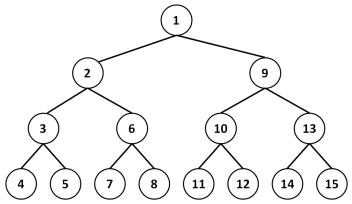
M. en C. Arturo Rodríguez García. PCIC. UNAM. 2014

BÚSQUEDA PRIMERO EN PROFUNDIDAD

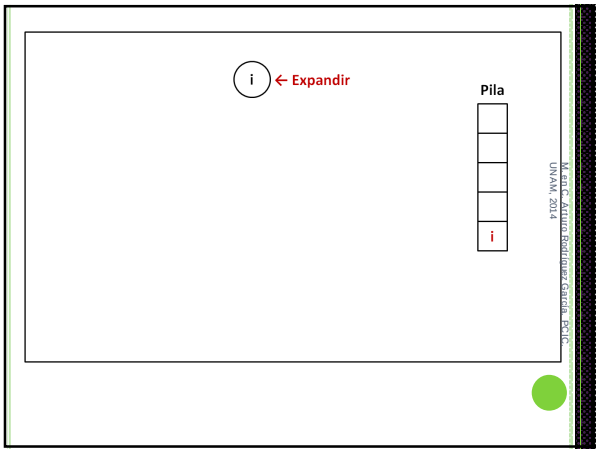
M. en C. Arturo Rodríguez García, PCIC, UNAM, 2014

BÚSQUEDA PRIMERO EN PROFUNDIDAD

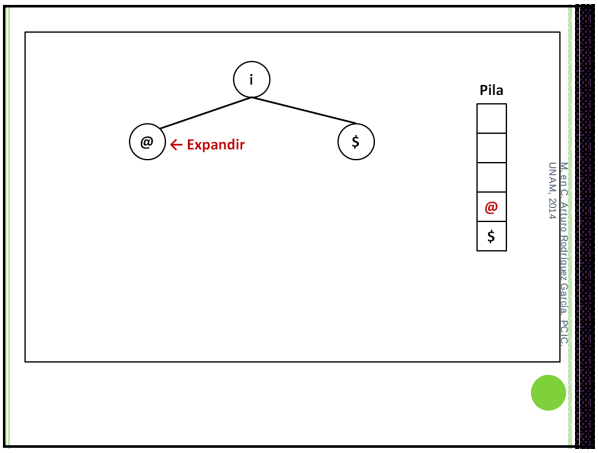
- Expande el nodo más profundo en la frontera del árbol de búsqueda.
- Utiliza una pila (LIFO).



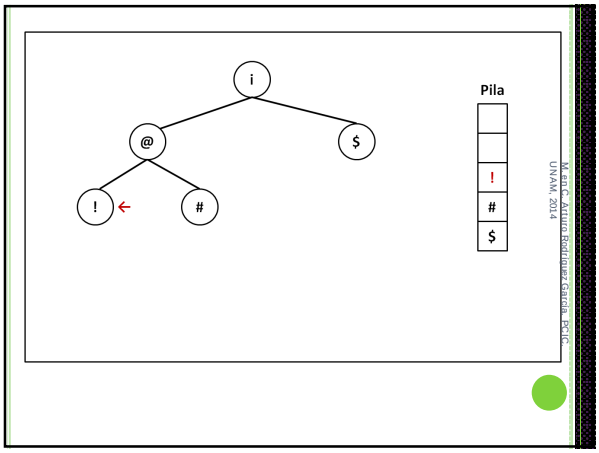
M. en C. Arturo Rodríguez García, PCIC, UNAM, 2014



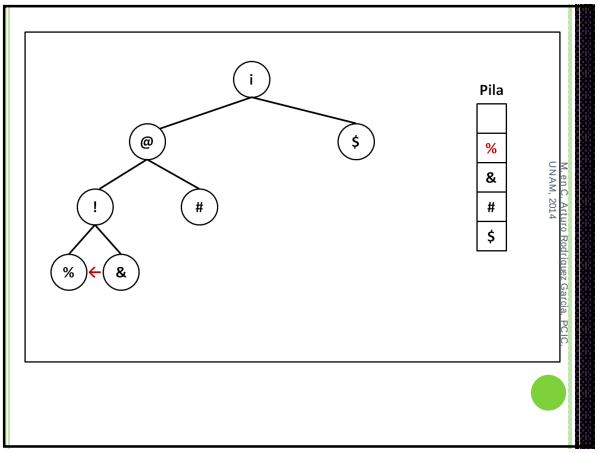
M. en C. Arturo Rodríguez García, PCIC, UNAM, 2014



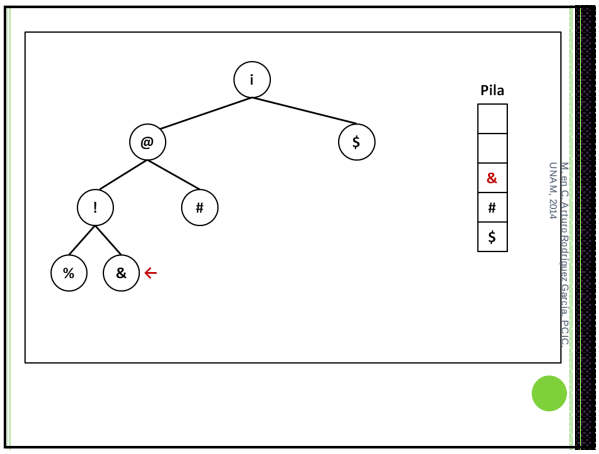
M. en C. Arturo Rodríguez García, PCIC, UNAM, 2014

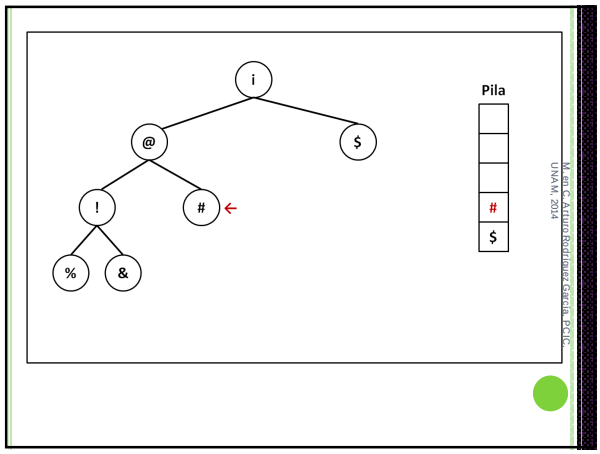


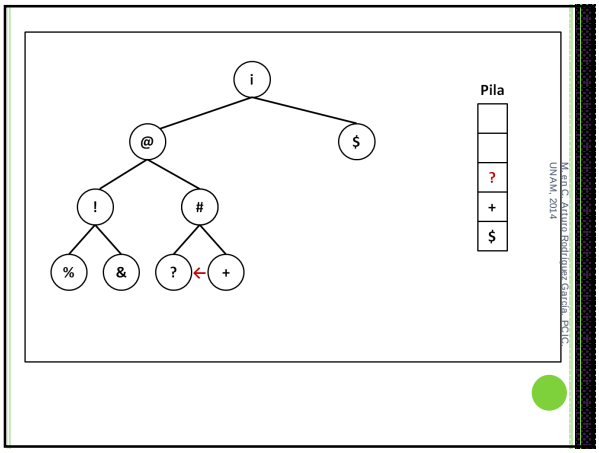
M. en C. Arturo Rodríguez García, PCIC, UNAM, 2014

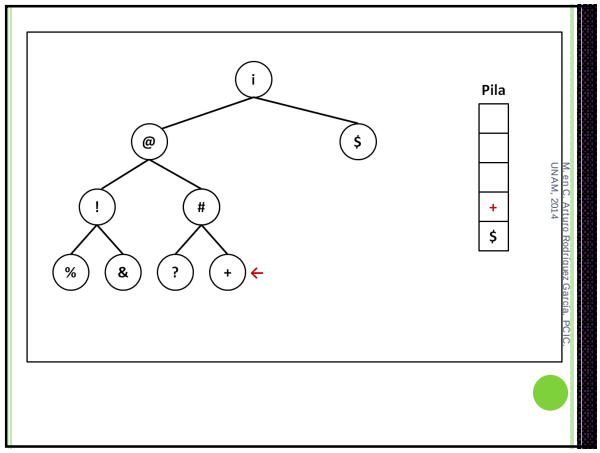


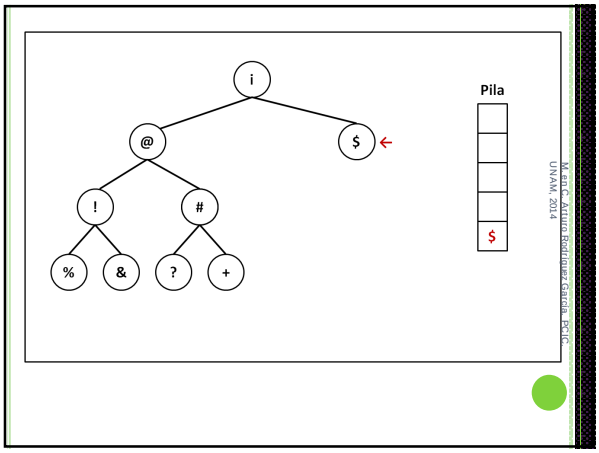
M. en C. Arturo Rodríguez García, PCIC, UNAM, 2014

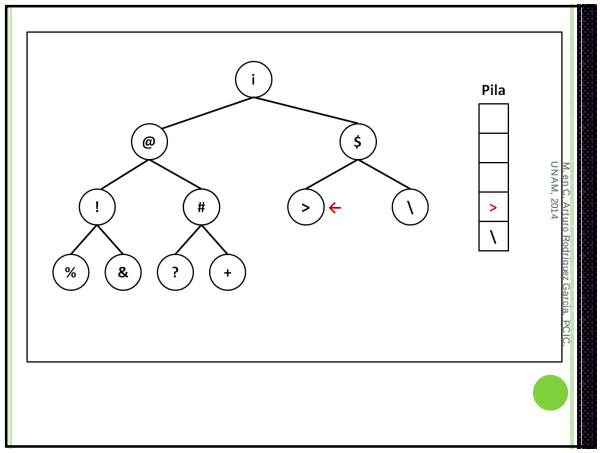

i
@
$
!
#
%
&
Pila
&
#
$
M. en C. Arturo Rodríguez García. FCiC. UNAM. 2014


i
@
$
!
#
%
&
Pila
#
$
M. en C. Arturo Rodríguez García. FCiC. UNAM. 2014

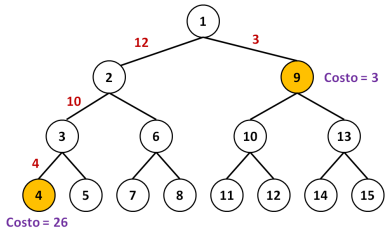

i
@
$
!
#
%
&
?
+
Pila
?
+
$
M. en C. Arturo Rodríguez García. FCiC. UNAM. 2014


i
@
$
!
#
%
&
?
+
Pila
+
$
M. en C. Arturo Rodríguez García. FCiC. UNAM. 2014


i
@
$
!
#
%
&
?
+
Pila
$
M. en C. Arturo Rodríguez García. FCiC. UNAM. 2014


i
@
$
!
#
>
\
%
&
?
+
Pila
>
\
M. en C. Arturo Rodríguez García. FCiC. UNAM. 2014

- ¿Es completo? No. El algoritmo podría tomar un camino infinito sin solución.
- ¿Es óptimo? No.

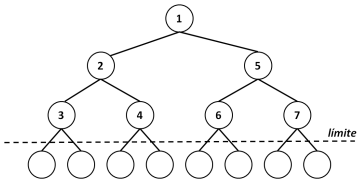


- Requisitos modestos de memoria: almacena sólo un camino desde la raíz hasta un nodo hoja, junto con los nodos hermanos restantes no expandidos de cada nodo de dicho camino.
- Si la profundidad máxima es *m*, un camino tiene a lo mucho *m+1* nodos. Cada nodo (excepto la raíz) tiene a sus hermanos almacenados. Entonces hay a lo mucho *bm + 1* nodos.
- La complejidad en espacio es por lo tanto *O(bm)*.
- En el caso peor, se generan todos los nodos del árbol de búsqueda, por lo tanto la complejidad en tiempo es $O(b^m)$.

Búsqueda de profundidad limitada

Búsqueda de profundidad limitada

- Consiste en aplicar la búsqueda primero en profundidad con un límite de profundidad *l* predeterminado, impidiendo caminos infinitos.
- Los nodos a profundidad *l* se tratan como si no tuvieran ningún sucesor.

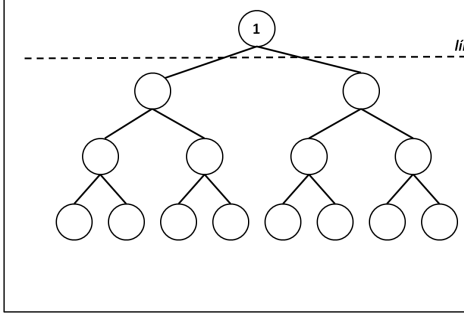


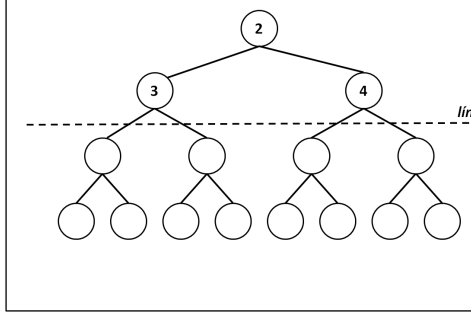
- ¿Es completo? No. Todas las soluciones podrían estar fuera del límite de profundidad.
- ¿Es óptimo? No. La solución óptima podría estar fuera del límite.
- Siguiendo el mismo razonamiento que en búsqueda primero en profundidad, pero cambiando *l* por *m*, la complejidad en tiempo es $O(b^l)$ y la complejidad en espacio es *O(bl)*.
- La búsqueda primero en profundidad es un caso especial de búsqueda de profundidad limitada con $l=\infty$.

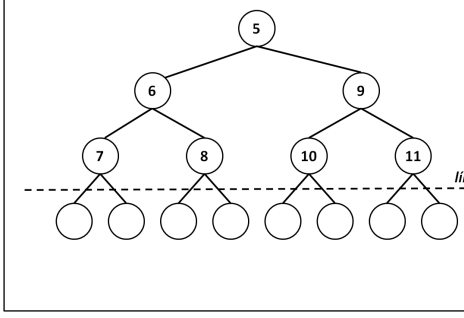
Búsqueda con profundidad iterativa

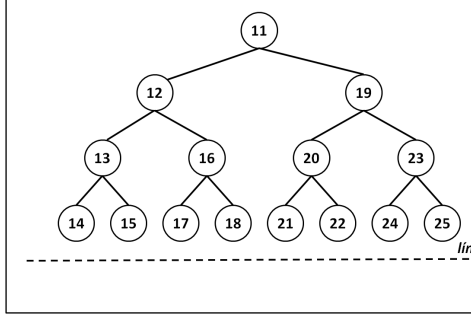
Búsqueda con profundidad iterativa

- Consiste en iterar la búsqueda de profundidad limitada incrementando gradualmente el límite, hasta encontrar un objetivo.









- A primera vista, este tipo de búsqueda parece que derrocha muchos recursos, pero en realidad no lo hace. El número de nodos generados es:

$(d+1)(1) + (d)(b) + (d-1)(b^2) + \ldots + 2(b^{d-1}) + 1(b^d)$

Por lo tanto, la complejidad en tiempo es $O(b^d)$

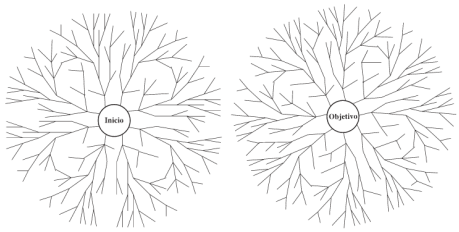
Nivel de profundidad	Número de nodos	Número de iteraciones
0	1	d+1
1	b	d
2	b^2	d – 1
3	b^3	d – 2
…	…	…
d – 2	b^{d-2}	3
d – 1	b^{d-1}	2
d	b^d	1

- La complejidad en tiempo $O(b^d)$ de este algoritmo es incluso menor que la complejidad $O(b^{d+1})$ de la búsqueda primero en anchura (debido a que esta última genera nodos en el nivel $d+1$).
- La complejidad en espacio es $O(bd)$ como resultado de ser una búsqueda en profundidad.
- Es completo si el factor de ramificación es finito.
- Es óptimo si el costo del camino es una función que no disminuye con la profundidad del nodo.

BÚSQUEDA BIDIRECCIONAL

BÚSQUEDA BIDIRECCIONAL

- Ejecuta dos búsquedas simultáneas:
  - Hacia adelante desde el estado inicial.
  - Hacia atrás desde el objetivo.
- Se detiene cuando las dos búsquedas se encuentren.
- Antes de expandir un nodo, se verifica si está en el otro árbol de búsqueda.
- La motivación es que $b^{d/2} + b^{d/2} < b^d$



- La complejidad en tiempo es $O(b^{d/2})$.
- La complejidad en espacio es $O(b^{d/2})$ debido a que se tiene que mantener por lo menos uno de los árboles en memoria para hacer la comprobación de pertenencia.
- Es completa si b es finita.
- Es óptima si los costos iguales y si en ambas direcciones se utiliza búsqueda primero en anchura.

- Se necesita una función extra para determinar los predecesores de un estado.
- Si hay varios estados objetivo explícitamente catalogados se pueden unir como predecesores inmediatos a un estado objetivo ficticio.
- El caso más difícil es cuando el test objetivo no enumera los estados objetivo, sino que da la descripción de ellos. No hay manera general de resolver este problema en forma eficiente.

Evitar estados repetidos

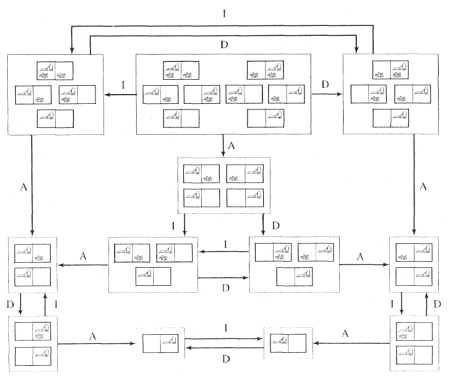
Evitar estados repetidos

- Los árboles de búsqueda para problemas con repetición de estados son infinitos.
- Para evitar estados repetidos es necesario almacenar en una lista todos los estados que ya han sido expandidos.
- Si un algoritmo recuerda cada estado que ha visitado, entonces se está explorando directamente el grafo de estados.

Búsqueda con información parcial

Problemas sin sensores

- Si el agente no tiene ningún sensor, entonces no puede saber en qué estado se encuentra.
- Sin embargo, el agente conoce el efecto de sus acciones.
- El agente debe aplicar algoritmos de búsqueda en el espacio de estados de creencia.



Problemas sin sensores en entornos no deterministas

- Si además de no conocer el estado actual, el agente no conoce el resultado de sus acciones el problema se vuelve no resoluble.
- En el mundo de la aspiradora, por ejemplo, aplicar la acción Aspirar no garantizaría que la casilla quede limpia.

Problemas de contingencia

- Si el entorno es parcialmente observable o si las acciones son inciertas, entonces las percepciones del agente proporcionan nueva información después de cada acción.
- El agente necesita de un plan de contingencia para manejar las circunstancias desconocidas que se puedan presentar.
- Se emplean algoritmos que intercalan la búsqueda y la ejecución.

Problemas de exploración

- Cuando se desconocen los estados y las acciones del entorno, el agente debe actuar para descubrirlos.
- Son un caso extremo de problemas de contingencia.

Referencias

- *Inteligencia Artificial: Un enfoque moderno.* Stuart Russell y Peter Norvig. 2ª Edición. Cápítulo 3. Páginas 67-105.