



INTELIGENCIA ARTIFICIAL

Universidad Nacional Mayor de San Marcos

Facultad de Ingeniería de Sistemas e Informática

Ing. Mg. Rolando A. Maguiña Pérez

❖ Métodos de búsqueda

- Conceptos sobre estrategias de búsqueda
- Métodos ciegos
 - ❖ Búsqueda en Amplitud
 - ❖ Búsqueda en Profundidad
 - ❖ Búsqueda No determinista



Métodos de Búsqueda

A Ciegas
(No informada)

Informada
(Heurística)



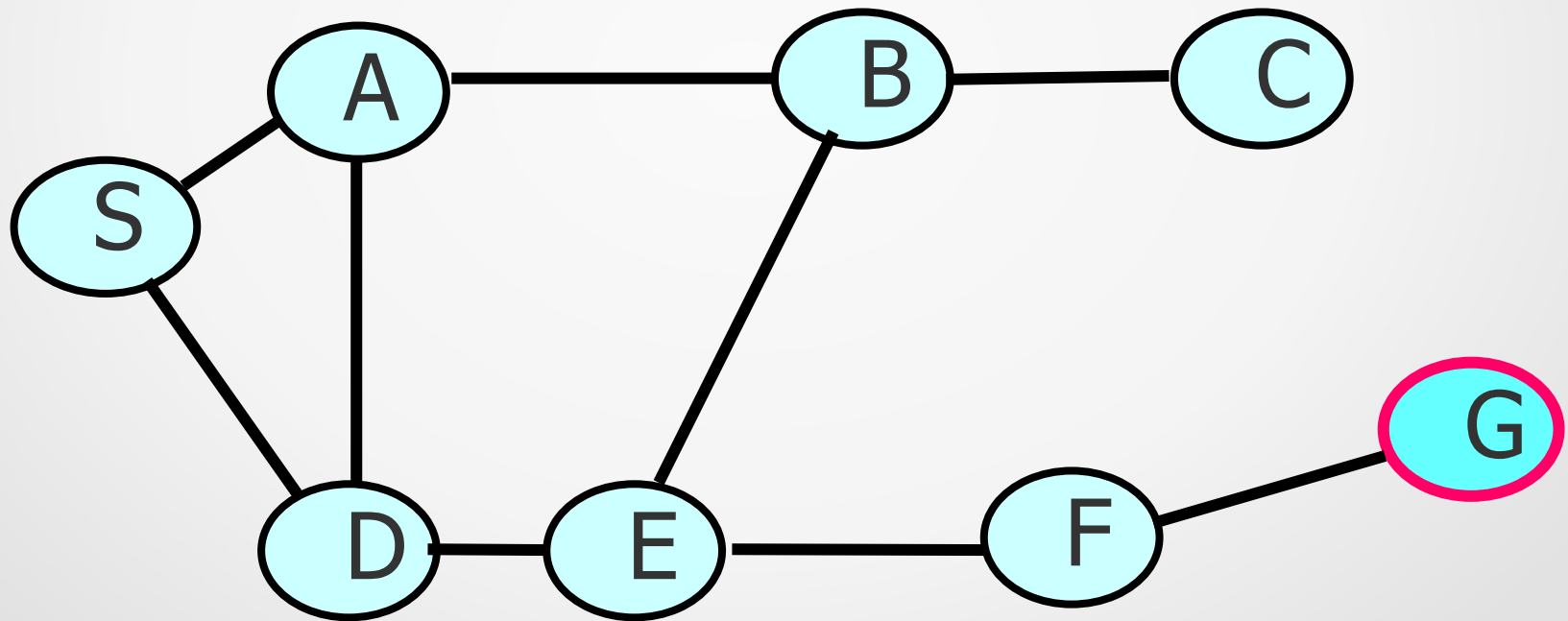
Métodos de Búsqueda a Ciegas

Ing. Mg. Rolando A. Maguiña Pérez

- ❖ Encontrar la forma apropiada de decidir las reglas a aplicar desde el estado inicial para llegar al estado final y el orden en que estas se aplican

Supongamos nos encontramos en una región de un país ficticio. En esa región queremos ir desde una ciudad que denominaremos S hasta la ciudad llamada G

no disponemos de un mapa de carreteras!

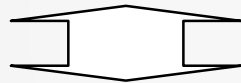
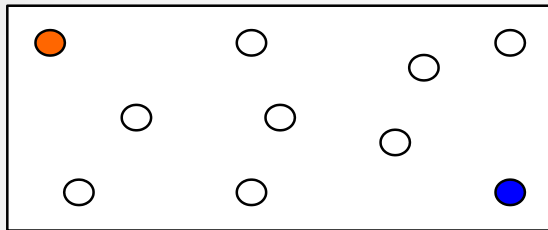




- ❖ Una ruta es una secuencia conectada de nodos
- ❖ Un ciclo es una secuencia conectada de nodos que se inicia y termina en el mismo nodo. La condición para la existencia de un ciclo es que exista dos caminos diferentes para conectar dos nodos.
- ❖ Un árbol es un grafo que no tiene ciclos
- ❖ Un árbol de estados es un árbol donde los nodos representan estados y las aristas o arcos muestran la relación de precedencia entre dos nodos

Los Métodos se basan en el árbol de estados

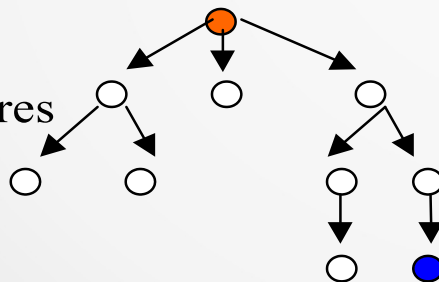
Espacio de estados



El espacio de estados se transforma en un árbol de estados

Raíz

Sucesores



- Estado Meta
- Estado Inicial

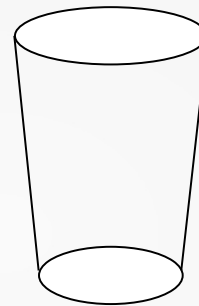
Árbol de Estado

Ing. Mg. Rolando A. Maguiña Pérez

Cómo se construye un árbol de estado?

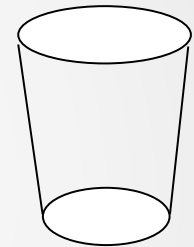
EJM. Vasijas de Agua

Llenar exactamente 1 litro de agua en vasija de 4. Considere que hay un surtidor de agua...



4 litros

vacías y
sin marcas



3 litros

Solución

Estado Inicial: $(0,0)$

Estado Meta: $(1,0), (1,1), (1,2), (1,3)$

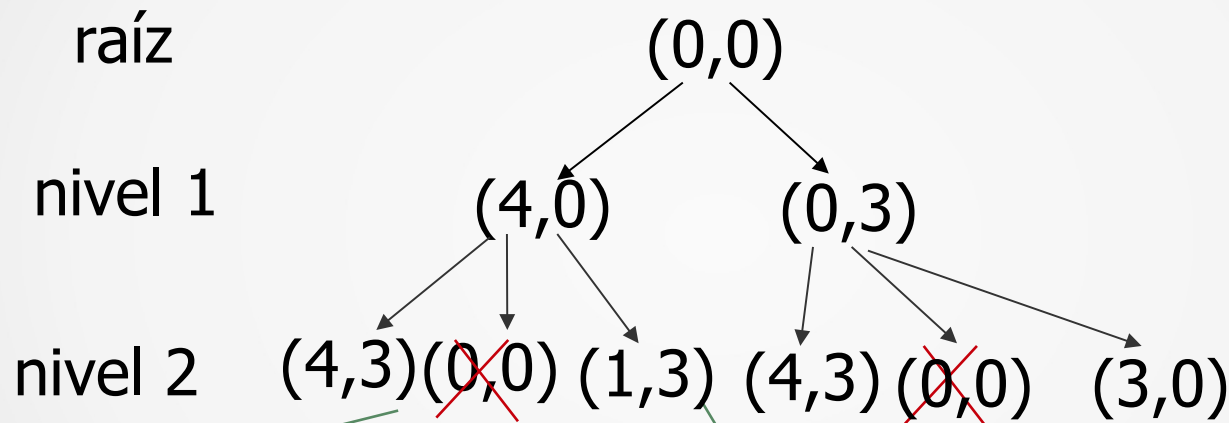
Como se construye un árbol de estado

Regla	Restricción	Nuevo estado
Llenar_4	$x < 4$	$(4, y)$
Llenar_3	$y < 3$	$(x, 3)$
Vaciar_4	$x > 0$	$(0, y)$
Vaciar_3	$y > 0$	$(x, 0)$
Pasar_4_a_3	$x > 0$ $y < 3$	$(x - m, y + m)$
Pasar_3_a_4	$x < 3$ $y > 0$	$(x + n, y - m)$

donde: $m = \text{mínimo}\{x, 3-y\}$, $n = \text{mínimo}\{y, 4-x\}$

Reglas para el Problema de las Vasijas de Agua

Cómo se construye un árbol de estado?



No se debe considerar porque es redundante

Estado Meta

Árbol de Estado – Problema de las Vasijas

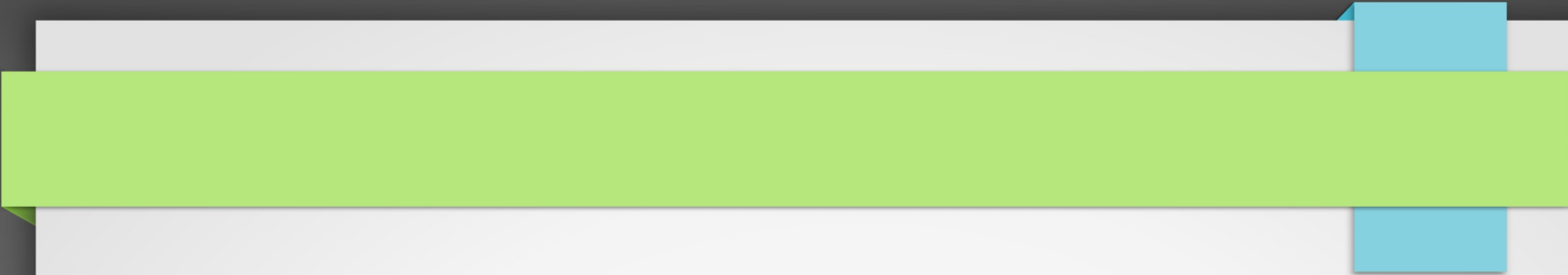
❖ Definición

- Son procedimientos sistemáticos de búsqueda del estado meta sobre el árbol de estado
- Se llaman así porque usan estrategias de búsqueda que sólo consideran la relación de precedencia entre estados

La información sobre el beneficio, utilidad, de pasar de un estado para otro estado no es considerada

❖ Métodos ciegos más conocidos:

- Búsqueda en amplitud (anchura)
- Búsqueda en profundidad
- Búsqueda no determinista



Métodos de Búsqueda a Ciegas

Búsqueda en Amplitud

Ing. Mg. Rolando A. Maguiña Pérez

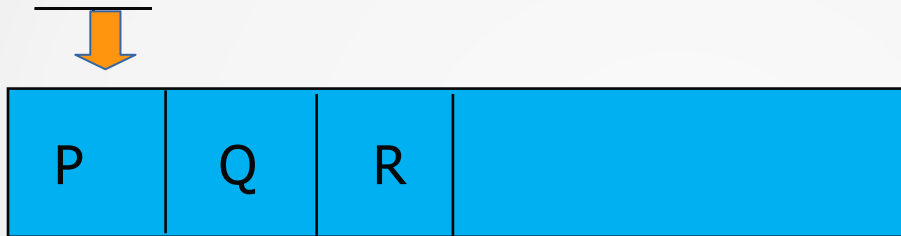
Procedimiento

Inicie en el nodo raíz del árbol de estado. Si el nodo corresponde al estado meta termine, caso contrario pase a generar los nodos sucesores no redundantes a este (nodos del primer nivel). Si alguno de los nodos del primer nivel corresponde al estado meta termine, caso contrario pase a generar los nodos sucesores no redundantes de los nodos del primer nivel (nodos del segundo nivel). El proceso se repite hasta encontrar el estado meta o cuando no sea posible generar nuevos sucesores.

Implementación – Listas

LE:

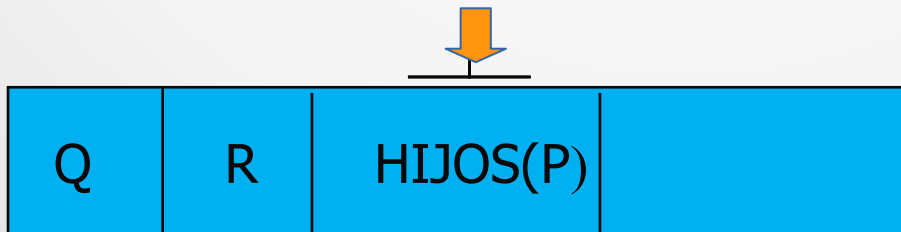
PROCESAR



Se procesa siempre el primer elemento de la lista LE

LE:

REGISTRO



Los sucesores son registrados en LE al final

Algoritmo - Listas

Inicio

1. $LE := (\text{Estado_Inicial});$
2. $LV := ();$

Test de Parada

3. Si $(LE = ())$ entonces
 Escribir("no hay solución"), PARE;
4. $P := \text{Primer}(LE);$
5. Si $(P \text{ es Meta})$ entonces
 Escribir("solución =", P), PARE;

Genera Sucesores:

6. $\text{Adiciona_ultimo}(P, LV);$
7. $\text{Elimina_primer}(LE);$
8. $\text{Adicionar_ultimo}(\text{Hijos}(P) - LV, LE)$
9. Ir Para 3

Ejemplo: Determine un camino c-e.
Considere la lectura en sentido horario

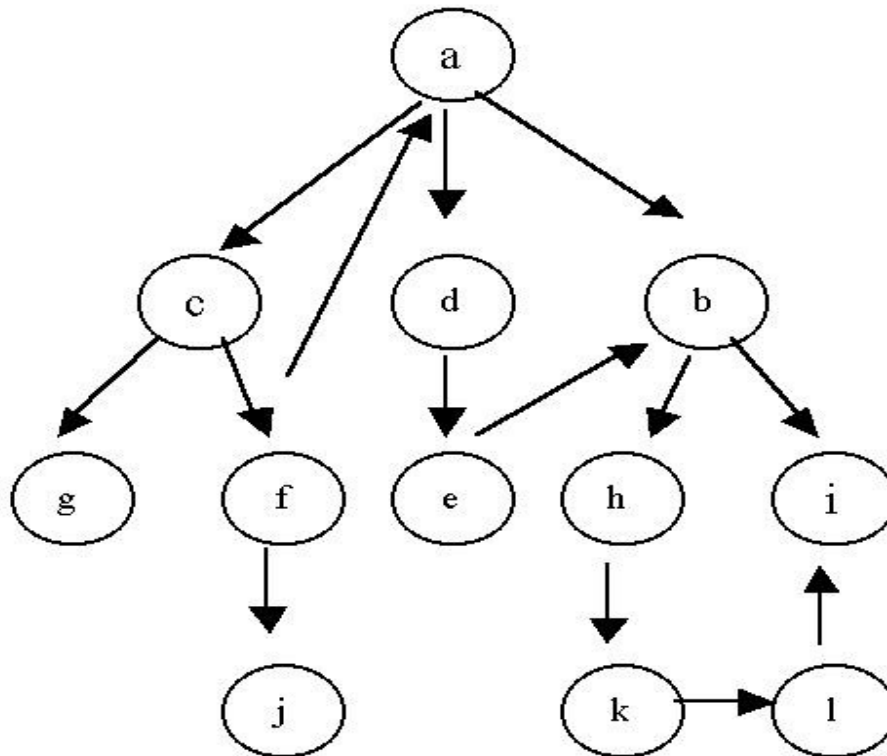


Tabla con resultados de aplicación del algoritmo

It	LE	P	LV
1	(c)	c	()
2	(f g)	f	(c)
3	(g a j)	g	(c f)
4	(a j)	a	(c f g)
5	(j b d)	j	(c f g a)
6	(b d)	b	(c f g a j)
7	(d i h)	d	(c f g a j b)
8	(i h e)	i	(c f g a j b d)
9	(h e)	h	(c f g a j b d i)
10	(e k)	e	(c f g a j b d i h)

La ruta solución es: ??

Ing. Mg. Rolando A. Maguiña Pérez

Determinación de la Secuencia de Estados Solución

La versión del algoritmo de búsqueda en amplitud presentada, detecta el estado meta o indica que el problema no tiene solución. Sin embargo, caso el problema tenga solución, no muestra la secuencia de estados solución.

Recordemos que una *secuencia de estados solución* o *ruta solución* es una secuencia de estados que comienza en el estado inicial y termina en el estado meta, y donde cada nodo (menos el inicial) de la secuencia se obtiene al aplicar una regla válida al nodo anterior a este.

Determinación de la Secuencia de Estados Solución

- ❖ Alternativa para determinar la ruta solución registrar para cada nodo de LE :
(nodo ruta)

LE estará constituida por una lista de listas de estados

(P ruta(P)) (Q ruta(Q)) (R ruta(R))

Enseguida se presenta una versión del algoritmo de búsqueda en amplitud que incorpora este criterio. Esta versión determina directamente la secuencia de estados solución

Algoritmo – Listas (da secuencia de estados sol)

Inicio

1. LE := ((Estado_Inicial));

2. LV:=();

Test de Parada

3. Si (LE = ()) entonces
 Escribir("no hay solución"), PARE;

4. LISTA := Primer(LE);

5. P := Ultimo(LISTA);

6. Si (P es Meta) entonces
 Escribir("solución =", LISTA), PARE;

Genera Sucesores:

7. Adiciona_ultimo(P, LV);

8. Elimina_primer(LE);

9. Para (Nodos \in (Hijos(P) – LV))

 W_LISTA := LISTA;

 Adicionar_ultimo(Nodo, W_LISTA);

 Adicionar_ultimo(W_LISTA, LE);

Fin_Para

10. Ir Para 3

Búsqueda en Amplitud

Algoritmo - Listas

Its	LE	LISTA	P	LV
1	((c))	(c)	c	()
2	((c) f) (c g))	(c f)	f	(c)
3	((c g) (c f a) (c f j))	(c g)	g	(c f)
4	((c f a) (c f j))	(c f a)	a	(c f g)
5	((c f j) (c f a b) (c f a d))	(c f j)	j	(c f g a)
6	((c f a b) (c f a d))	(c f a b)	b	(c f g a j)
7	((c f a d) (c f a b i) (c f a b h))	(c f a d)	d	(c f g a j b)
8	((c f a b i) (c f a b h) (c f a d e))	(c f a b i)	i	(c f g a j b d)
9	((c f a b h) (c f a d e))	(c f a b h)	h	(c f g a j b d i)
10	((c f a d e) (c f a b h k))	(c f a d e)	e	(c f g a j b d i h)

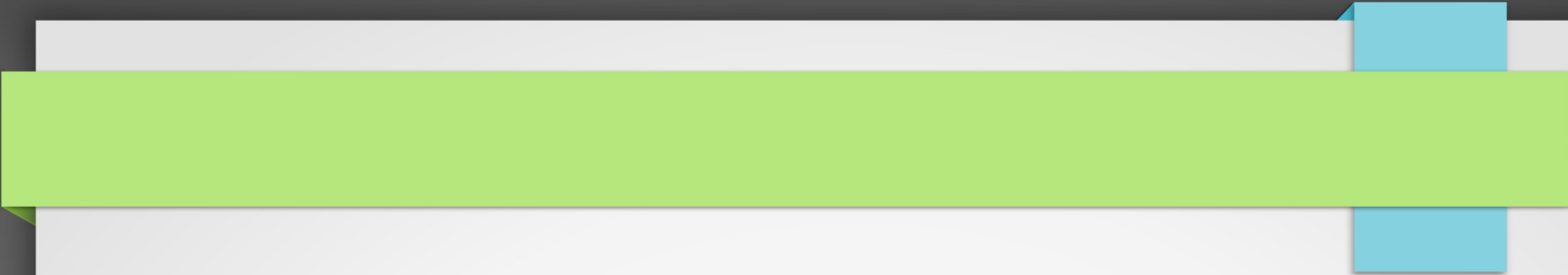
La ruta solución es c-f-a-d-e

Ing. Mg. Rolando A. Maguiña Pérez

Estado Meta

Propiedades

- ❖ Completitud: **completo** (encuentra solución si existe y el factor de ramificación es finito en cada nodo)
- ❖ Eficiencia: **buena si las metas están cercanas**
- ❖ Complejidad en tiempo: $O(b^d)$,
b: factor de ramificación.
d: profundidad de la solución.
- ❖ Complejidad en espacio: $O(b^d)$,
Todos los nodos en memoria



Métodos de Búsqueda a Ciegas

Búsqueda en Profundidad

Ing. Mg. Rolando A. Maguiña Pérez

Conceptos

- ❖ Una hoja de un árbol es un nodo del árbol que no tiene sucesores.
- ❖ Una rama de un árbol es un camino que inicia en el nodo raíz y termina en un nodo hoja.

Procedimiento

El método consiste en una búsqueda por las ramas del árbol de estados

Si en una de las ramas se encuentra el estado meta entonces el procedimiento termina, de lo contrario se pasa a investigar sobre otra rama no redundante. El procedimiento se repite hasta encontrar el estado meta o hasta que no existan más ramas a investigar.

Propiedades

- ❖ Completitud: **no es completa**
- ❖ Eficiencia: bueno cuando metas están alejadas de EI o problemas de memoria
- ❖ Complejidad en tiempo: $O(b^d)$
 - b: factor de ramificación
 - d: máx profundidad de la búsqueda
- ❖ Complejidad en espacio: $O(bd)$ [en implementación sin la ruta], $O(b^d)$ [con la ruta]

Implementación – Listas

LE:

Lista de nodos en espera de proceso
(comparados con el estado meta)

LV:

Lista de nodos ya procesados (comparados
con el estado meta)

Implementación – Listas

LE:

PROCESAR



Se procesa siempre el primer elemento de la lista LE

LE:

REGISTRO



Los sucesores son registrados en LE al inicio

Algoritmo - Listas

Inicio

1. $LE := ((Estado_Inicial));$
2. $LV := ();$

Test de Parada

3. Si $(LE = ())$ entonces
 Escribir("no hay solución"), PARE;
4. $P := Primer(LE);$
5. Si $(P \text{ es Meta})$ entonces
 Escribir("solución =", P), PARE;

Genera Sucesores:

6. $Adiciona_ultimo(P, LV);$
7. $Elimina_primer(LE);$
8. $Adicionar_inicio((Hijos(P) - LV), LE);$
9. Ir Para 3

Algoritmo – Listas (da secuencia de estados sol)

Inicio

1. $LE := ((Estado_Inicial));$

2. $LV := ();$

Test de Parada

3. Si $(LE = ())$ entonces
 Escribir("no hay solución"), PARE;

4. $LISTA := Primer(LE);$

5. $P := Ultimo(LISTA);$

6. Si $(P \text{ es Meta})$ entonces
 Escribir("solución =", LISTA), PARE;

Genera Sucesores:

7. $Adiciona_ultimo(P, LV);$

8. $Elimina_primer(LE);$

9. Para $(Nodos \in (Hijos(P) - LV))$

$W_LISTA := LISTA;$

$Adicionar_ultimo(Nodo, W_LISTA);$

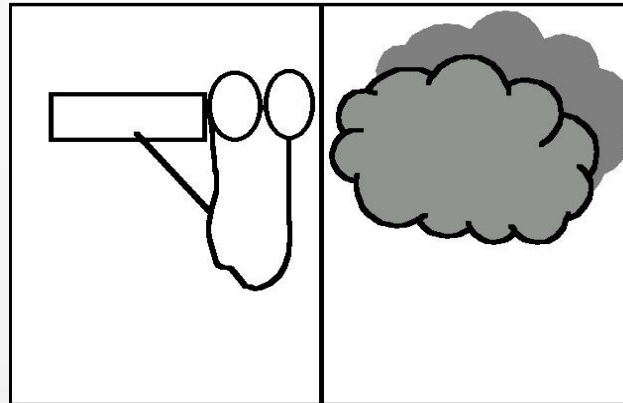
$Adicionar_primero(W_LISTA, LE);$

Fin_Para

10. Ir Para 3

El Mundo de la Aspiradora

En este mundo hay dos posibles ubicaciones. En ellas puede o no haber suciedad y el agente (aspiradora) se encuentra en una de las dos ubicaciones. El mundo puede asumir 8 posibles estados.



El Mundo de la Aspiradora-continuación

Son tres las acciones que el agente puede emprender: desplazarse a la izquierda, a la derecha y aspirar.

Suponga que al inicio hay mugre en las dos ubicaciones; y que la aspiradora está en el lado izquierdo. Suponga también que la eficiencia del aspirado es de 100%.

Problema

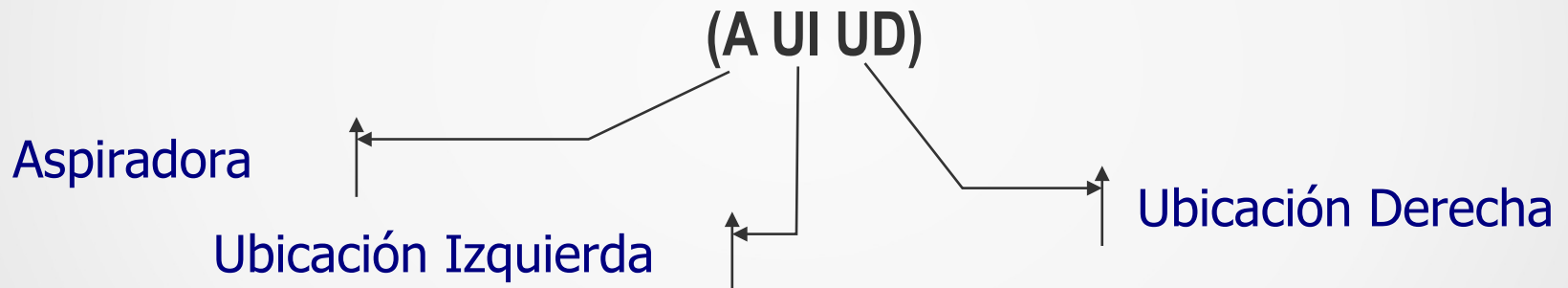
Consiste en eliminar toda la suciedad

Representación de problemas como EE

Objetos

aspiradora, ubicación izquierda, ubicación derecha, mundo

Estado



$$A \in \{I, D\}$$

$$UI, UD \in \{M, L\}$$

Reglas para el Mundo de la Aspiradora

Regla	Condición	Nuevo estado
Desp_izquierda	$A=D$	(I UI UD)
Desp_derecha	$A=I$	(D UI UD)
Aspirar_izquierdo	$A=I \wedge UI=M$	(I L UD)
Aspirar_derecho	$A=D \wedge UD=M$	(D UI L)

Búsqueda en Profundidad

Algoritmo – Listas con rutas

It	LE	Lista	P	LV
1	(((I M M)))	((I M M))	(I M M)	()
2	(((I M M) (D M M))) ((I M M) (I L M)))	((I M M) (D M M))	(D M M)	((I M M))
3	((I M M) (D M M) (D M L))) ((I M M) (I L M)))	((I M M) (D M M) (D M L))	(D M L)	((I M M) (D M M))
4	((I M M) (D M M) (D M L) (I M L))) ((I M M) (I L M j)))	((I M M) (D M M) (D M L) (I M L))	(I M L)	((I M M) (D M M) (D M L))
5	((I M M) (D M M) (D M L) (I M L) (I L L))) ((I M M) (I L M)))	((I M M) (D M M) (D M L) (I M L) (I L L))	(I L L)	((I M M) (D M M) (D M L) (I M L))
7				

La ruta solución es ((I M M)-(D M M)-(D M L)-(I M L)-(I L L))

Comparativa Amplitud vs Profundidad

LE:

PROCESAR



Se procesa el primer elemento de LE

LE:

REGISTRAR



¿Es el primer nodo el mejor para ser procesado?

HIJOS(P)

Profundidad

HIJOS(P)

Amplitud

Observaciones: Amplitud - Profundidad

- ❖ El estado a ser procesado en ambos métodos es el primero de la lista LE.
- ❖ Las listas generadas en los métodos de búsqueda en amplitud y en profundidad son registradas respectivamente al final e inicio de LE.

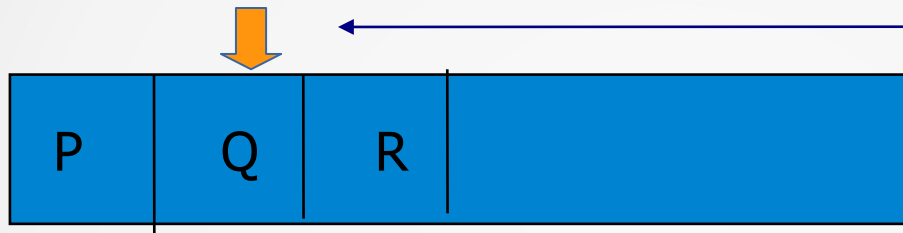
Procedimiento

En este método el nodo a procesar es seleccionado aleatoriamente de la lista LE. Los nodos sucesores son colocados al inicio o al final de LE.

Implementación - Listas

LE:

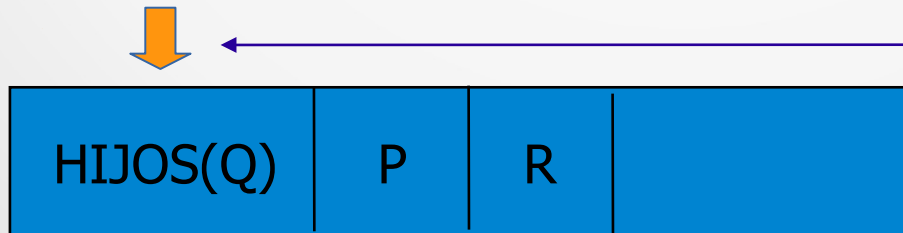
PROCESAR



Selección
aleatoria

LE:

REGISTRO



Conveniencia

Algoritmo - Listas

Inicio

1. LE := ((Estado_Inicial)); LV:=();

Test de Parada

2. Si (LE = ()) entonces
 Escribir("no hay solución"), PARE;

3. LISTA := **Aleatorio**(LE)

4. P := Ultimo(LISTA)

5. Si (P es Meta) entonces
 Escribir("solución =", LISTA), PARE;

Genera Sucesores:

6. Adiciona_ultimo(P, LV);

7. **Elimina_Aleatorio**(LE);

8. Para (Nodo \in (Hijos(P) - LV))

 W_LISTA := LISTA;

 Adicionar_ultimo(Nodo, W_LISTA);

 Adicionar_primero(W_LISTA, LE);

 Fin_Para

9. Ir Para 2