

Inteligencia Computacional

Unidad VI

Búsqueda.

Estrategias informadas y no informadas.

Complejidad computacional.

Planificación.

Docente:

Dr. Georgina Stegmayer

gstegmayer@santafe-conicet.gov.ar

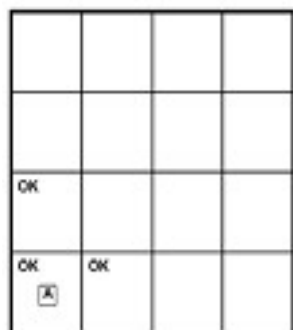
Inteligencia Artificial

- ✓ el trabajo de la IA es diseñar el **programa** del agente

función que implementa la relación que establece
el *agente* entre *percepciones* y *acciones*.

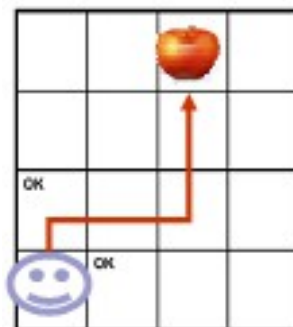
Se asumirá que este *programa* será ejecutado sobre algún
tipo de dispositivo computacional, al cual llamaremos la
arquitectura del agente.

$$\textit{agente} = \textit{arquitectura} + \textit{programa}$$



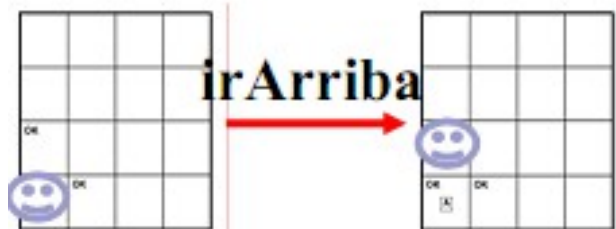
FORMULACIÓN DE OBJETIVOS

Un OBJETIVO es un conjunto de estados del mundo (estados en los cuales el objetivo se encuentra satisfecho).



FORMULACIÓN DEL PROBLEMA

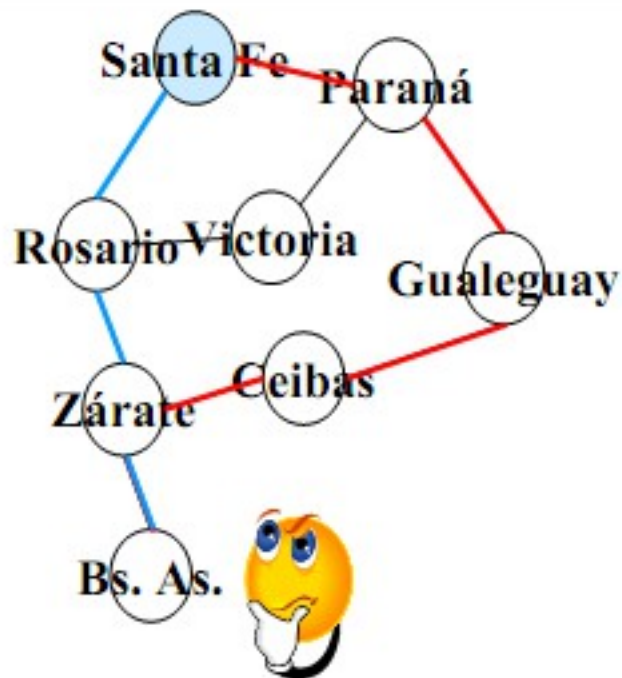
consiste en el proceso de definir qué acciones y estados considerar para alcanzar el **OBJETIVO** (incluye la formulación del **OBJETIVO**)



ACCIONES

Una **ACCIÓN** es una operación que causa una transición entre estados del mundo.

RESOLUCIÓN DE PROBLEMAS



El proceso de identificar las secuencia posibles de
estado-acción-nuevo estado

es denominado **BUSQUEDA.**

Formular el Objetivo

En Santa Fe.

Estados: enSantaFe, enParana...

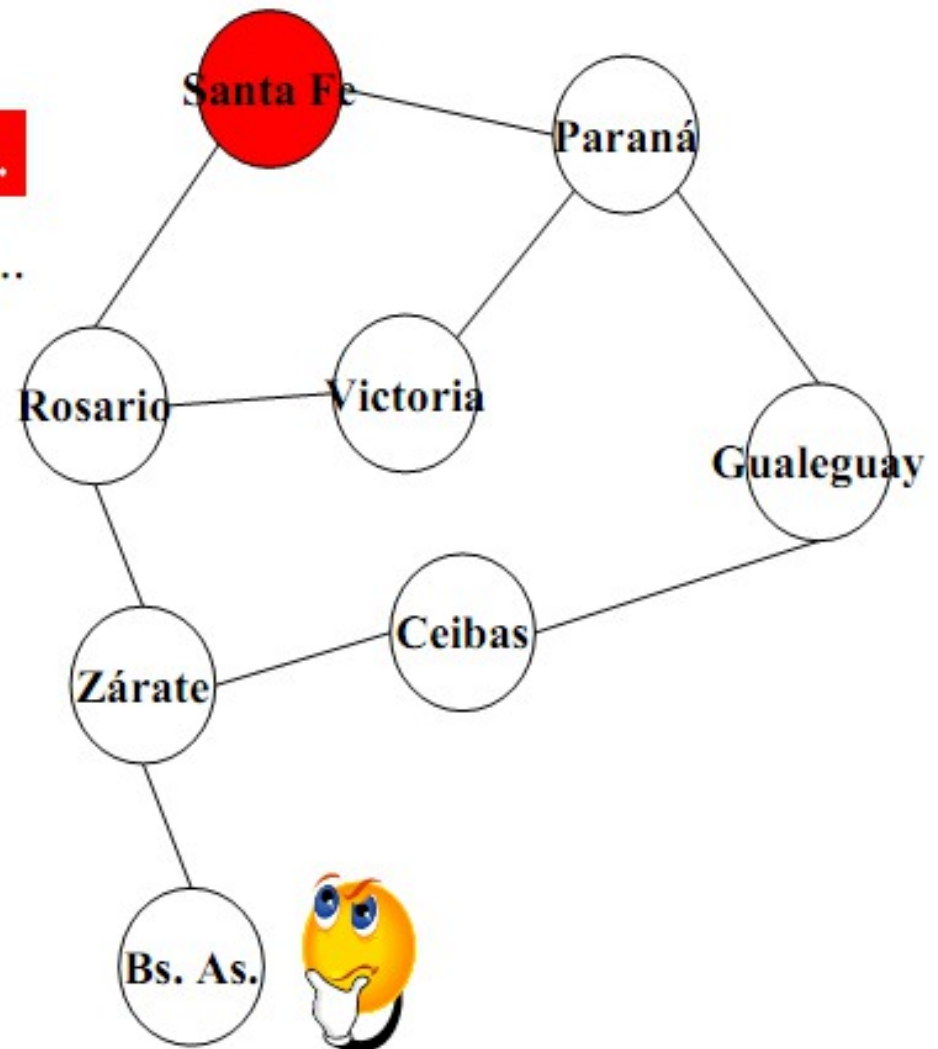
Acciones:

irRosario
irVictoria
irParaná
irZárate....

Solución:

(secuencia de acciones)

irZárate
irRosario
irSanta Fe.



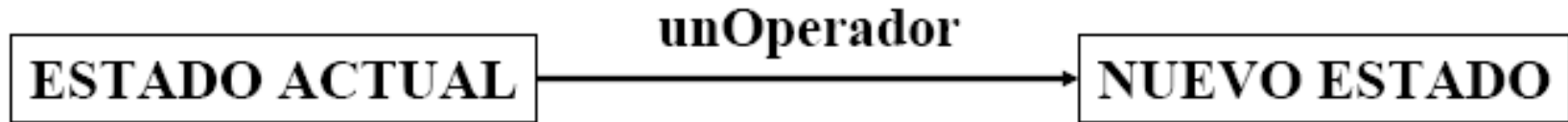
ALGORITMOS DE BÚSQUEDA

Problemas bien definidos

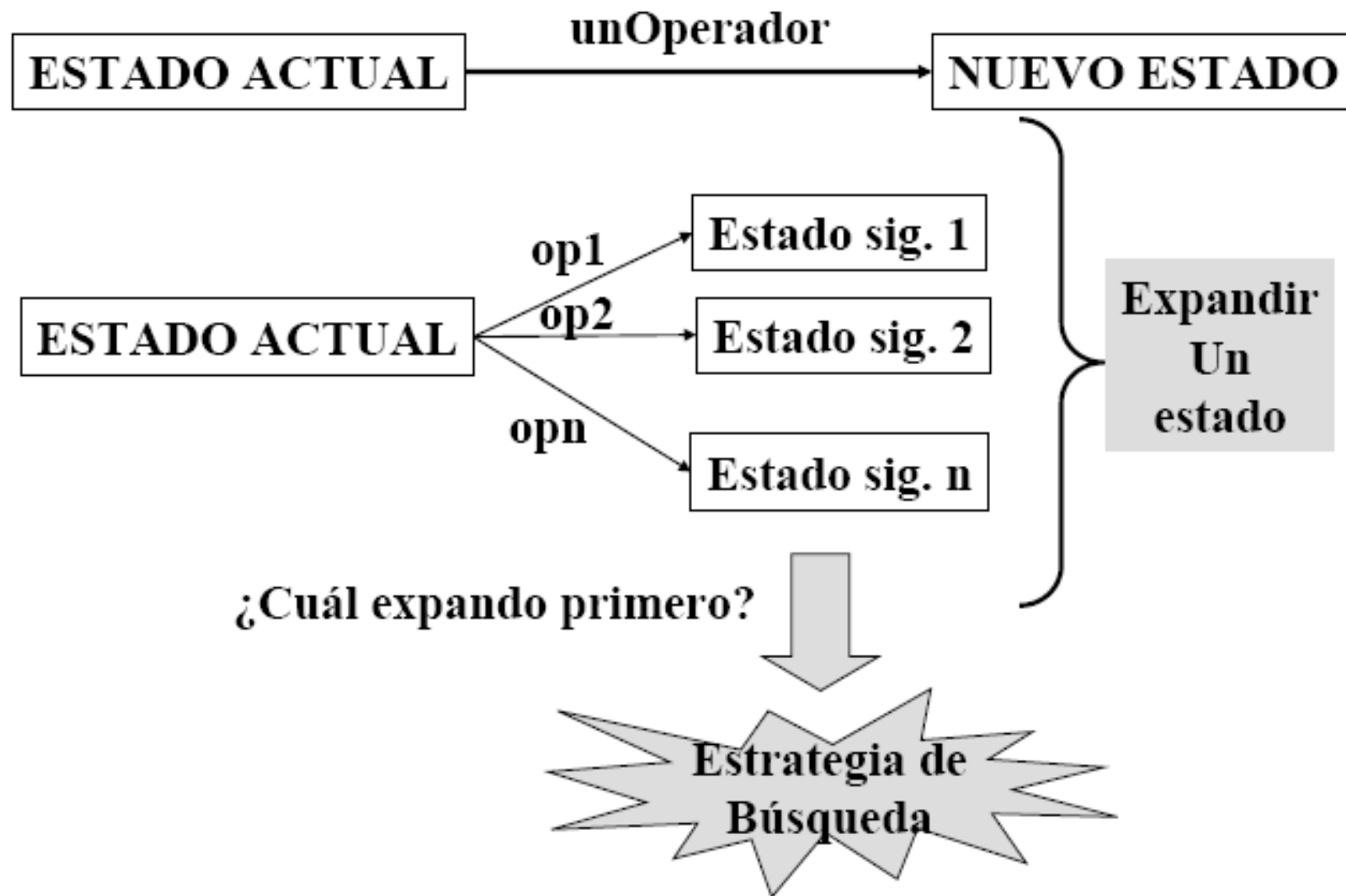
- **Estado inicial**
- Conjunto de **acciones** disponibles
- **Espacio de Estados del problema**: el conjunto de todos los estados alcanzables desde el estado inicial por cualquier secuencia de acciones u operadores
- El **Test de verificación** de objetivo
- **Función de evaluación** de Costo del camino

El **proceso de Búsqueda** puede ser pensado como la construcción de un **Árbol de Búsqueda** el cual está superpuesto al **espacio de estados**, donde la raíz corresponde al **estado inicial**

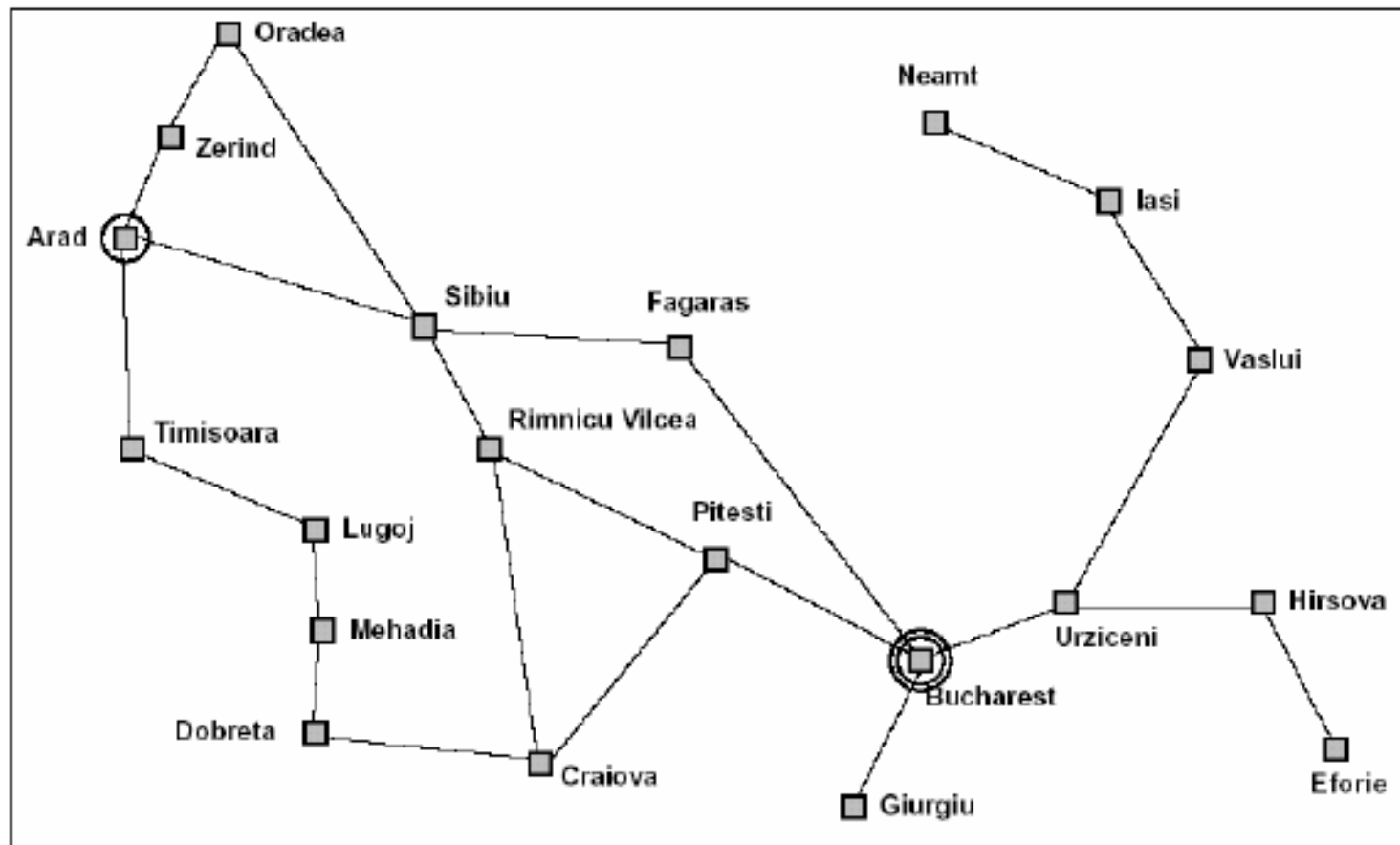
PROCESO DE BÚSQUEDA



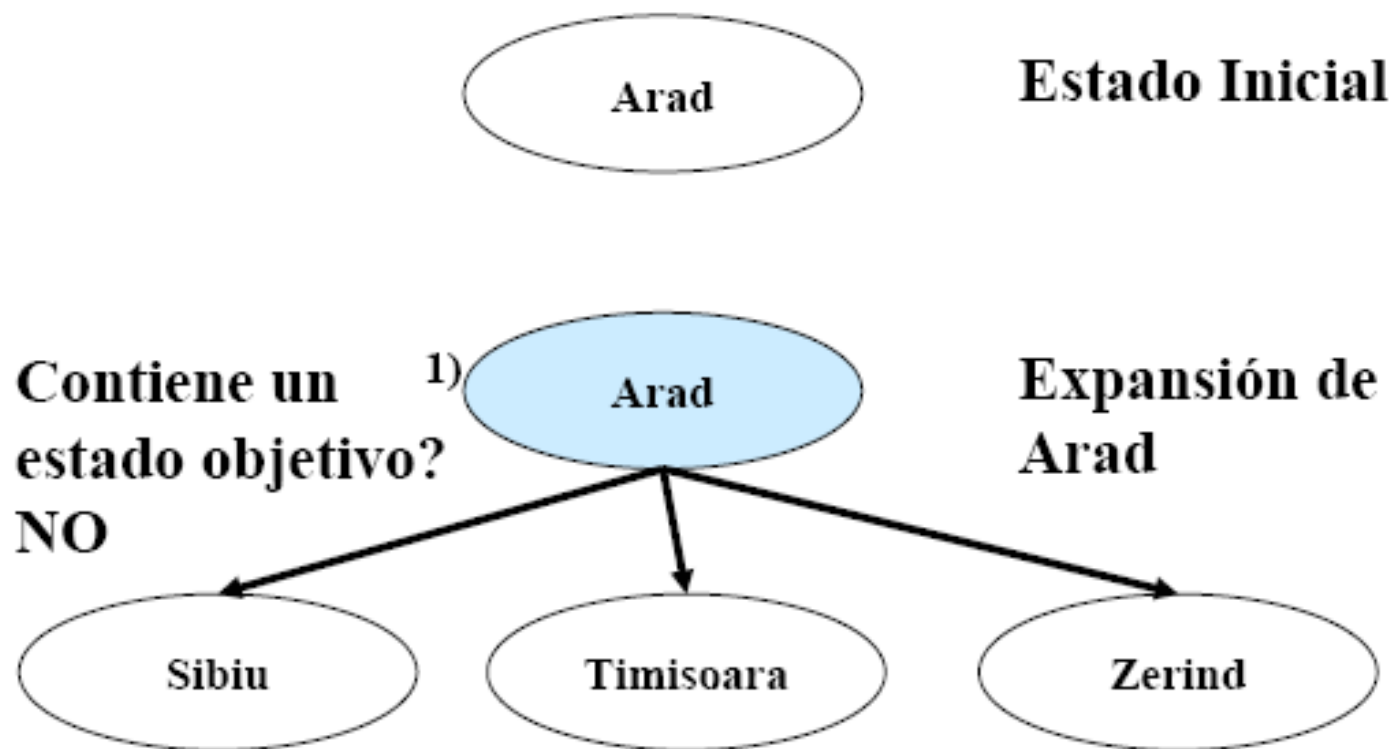
PROCESO DE BÚSQUEDA

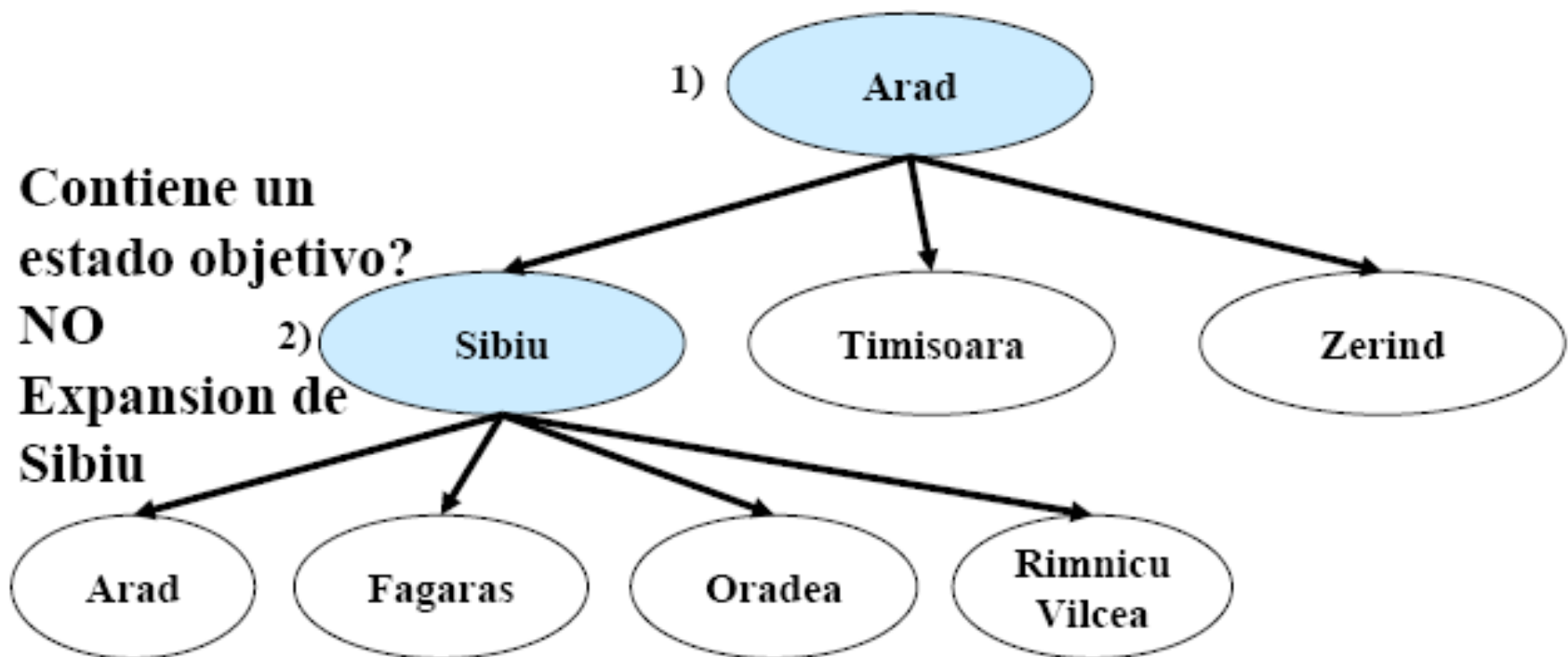


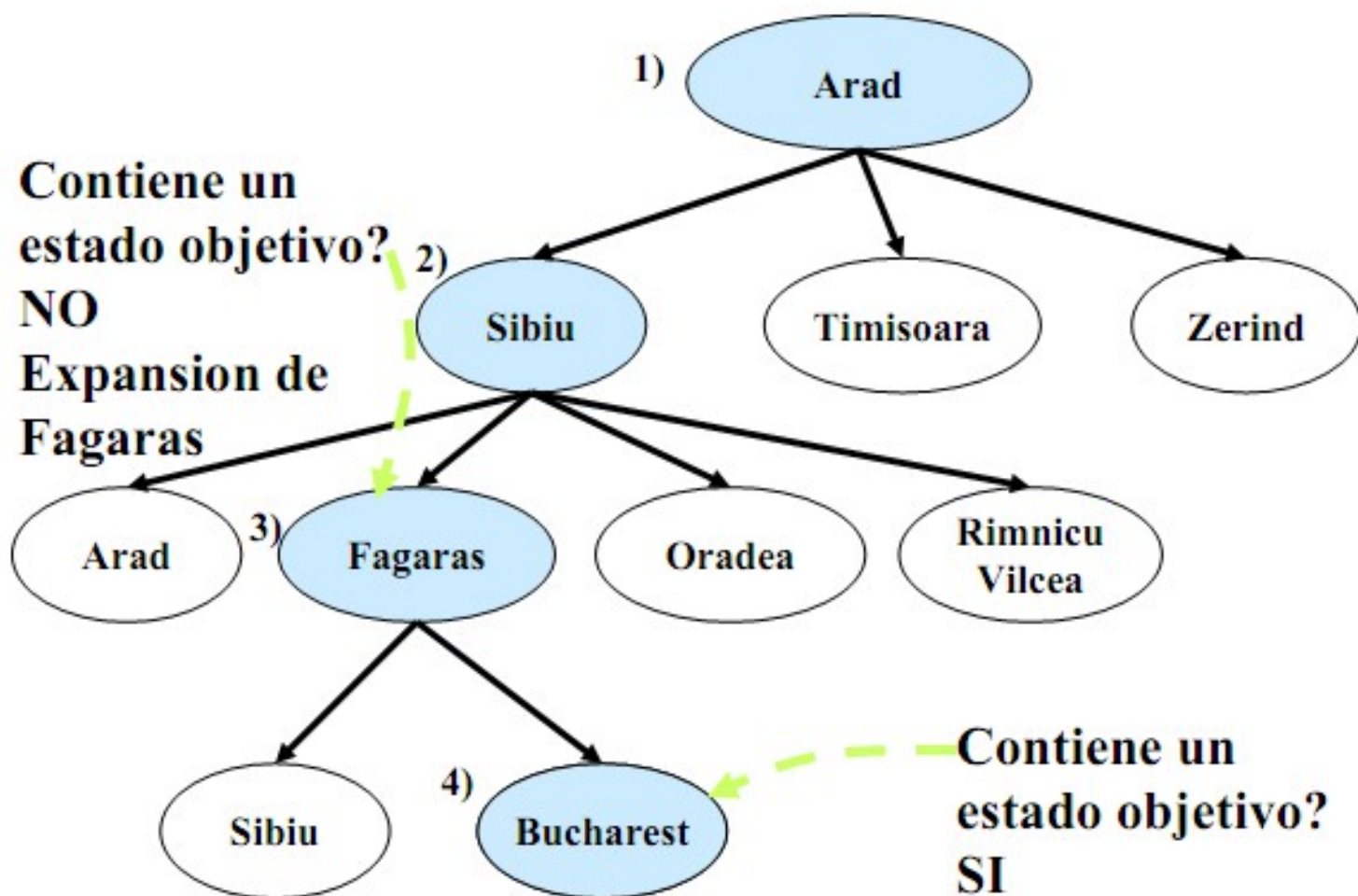
PROBLEMA DEL VIAJERO



El agente inicialmente se encuentra en la ciudad de Arad, y desea arribar a la ciudad de Bucarest.





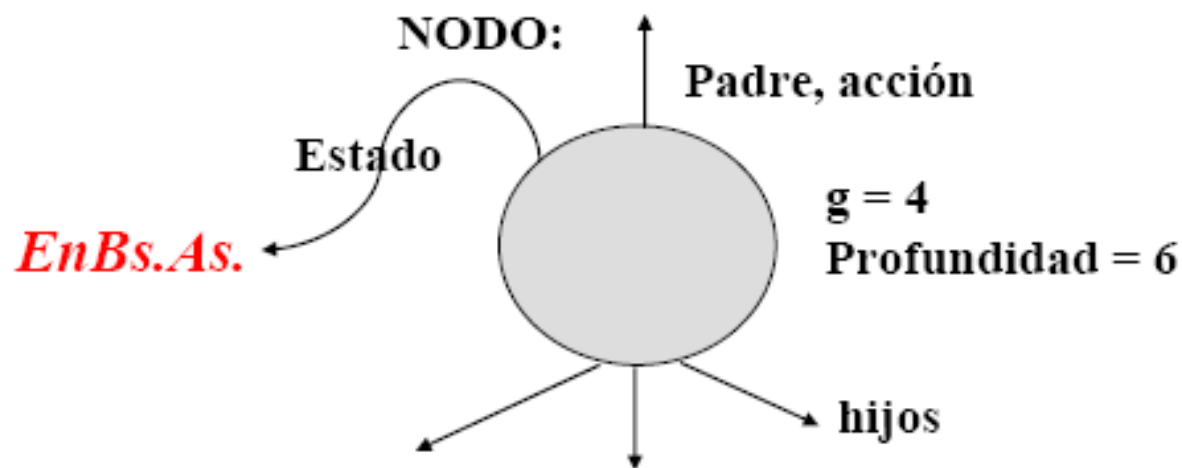


Solución: Arad – Sibiu – Fagaras - Bucharest

ÁRBOL DE BÚSQUEDA: nodos y arcos

Una posible estructura de datos para un **nodo** del árbol de búsqueda sería:

- **estado**
- **el nodo padre**
- **el operador aplicado** para su generación
- **el número de nodos** desde la raíz
- **el costo del camino** desde el nodo inicial hasta él



IMPORTANTE!

Cual es la diferencia entre **ESTADO** y **NODO**?

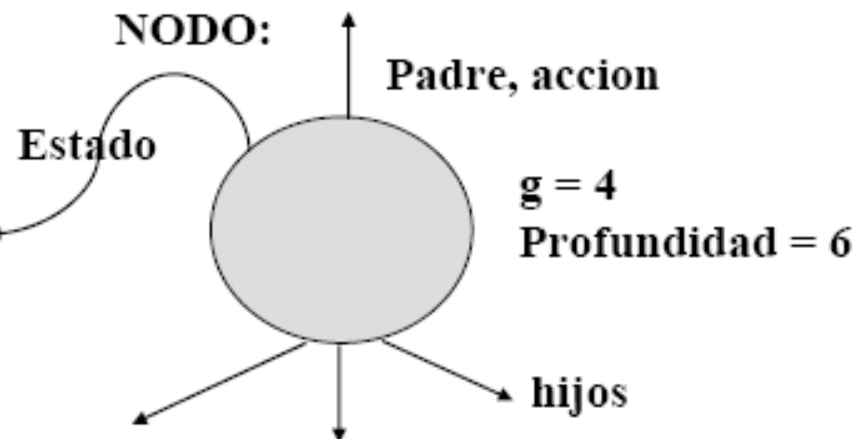
El **ESTADO** es la representación de una configuración física

Un **NODO** es una estructura de datos que forma parte del árbol de búsqueda incluye: estado, padre, accion, costo, hijos.

ESTADO:



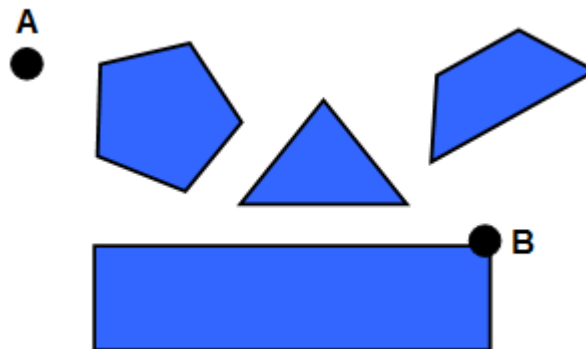
NODO:



Problema de navegación de robots

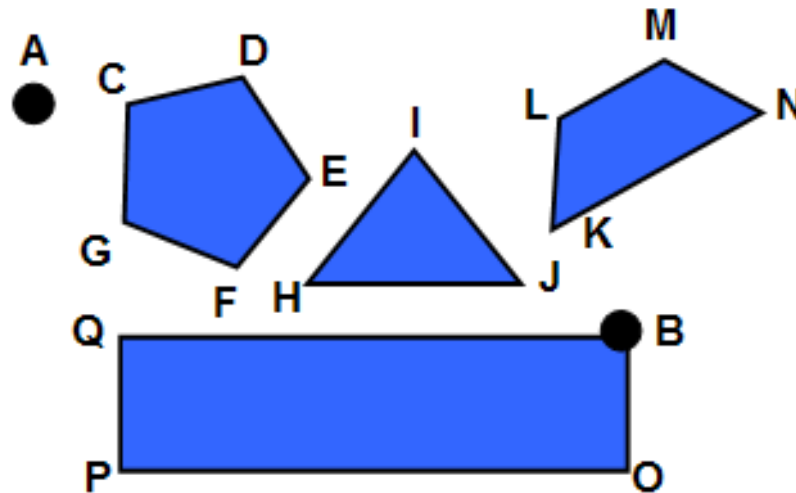
Considere el problema de la determinación del camino más corto entre dos puntos, en el que existe un conjunto de obstáculos. Dicho problema es típicamente estudiado en robótica para poder lograr que un robot pueda moverse en un ambiente, p.e. para transporte de mercancías, exploración de un terreno, vehículos inteligentes, entre otros.

Utilizando búsqueda, encuentre el camino más corto que debe seguir un robot que desee desplazarse desde el punto A hasta el punto B.



Problema de navegación de robots

sucesor(x): función auxiliar que toma un vértice como entrada y retorna el conjunto de vértices que pueden ser alcanzados en línea recta a partir de allí.



Ejemplos de la función sucesor:

$$\textit{sucesor}(A) = \{C, G\}$$

$$\textit{sucesor}(G) = \{A, C, F, Q\}$$

...

Problema de navegación de robots

- ✓ Representación del estado: (*posiciónAgente*)

Problema de navegación de robots

- ✓ Representación del estado: (*posiciónAgente*)
- ✓ Estado inicial: (A)
- ✓ Estado final: (B)

Problema de navegación de robots

- ✓ Representación del estado: (*posiciónAgente*)
- ✓ Estado inicial: (A)
- ✓ Estado final: (B)
- ✓ Prueba de meta:
$$\text{SI } \textit{posiciónAgente} = B \rightarrow \text{ÉXITO}$$

Problema de navegación de robots

- ✓ Representación del estado: (*posiciónAgente*)
- ✓ Estado inicial: (A)
- ✓ Estado final: (B)
- ✓ Prueba de meta:

SI $\text{posiciónAgente} = B \rightarrow \text{ÉXITO}$

- ✓ Operadores:

irA:

SI $A \in \text{sucesor}(\text{posiciónAgente}) \rightarrow (A)$

irB:

SI $B \in \text{sucesor}(\text{posiciónAgente}) \rightarrow (B)$

...

ALGORITMO DE BUSQUEDA GENERAL

función Búsqueda-General (*problema* , *estrategia*)
returns *solución* , o *falla*

1. *inicializar* el árbol de búsqueda empleando el *estado inicial del problema*
2. *loop do*
3. *if* no hay candidatos para expansión *then return* *falla*
4. *elegir* un nodo hoja para expansión de acuerdo a la *estrategia*
5. *if* el nodo contiene un estado objetivo
6. *then return* *la solución*
7. *else expandir* el nodo y *adicionar* los nodos resultantes al árbol de búsqueda
8. *end*

ESTRATEGIAS DE BÚSQUEDA

Básicamente la *estrategia de búsqueda* define como elegir el *próximo nodo a expandir*.

Las estrategias de búsqueda se consideran a partir de los siguientes criterios:

- **Completitud:** la estrategia garantiza encontrar una solución cuando al menos existe una?
- **Complejidad temporal:** cuanto tiempo se tarda en encontrar una solución?
- **Complejidad espacial:** cuánta memoria es necesaria para realizar la búsqueda?
- **Optimalidad:** la estrategia encuentra la solución de mejor calidad cuando existen varias soluciones?

ESTRATEGIAS DE BÚSQUEDA

La complejidad temporal y espacial es medida en función de:

- ☒ máximo factor de ramificación del árbol (b)
- ☒ profundidad de la solución más barata (d)
- ☒ Profundidad máxima del espacio de estado (m)

NO INFORMADAS O CIEGAS

- ☒ Amplitud (u horizontal)
- ☒ Profundidad
- ☒ Costo uniforme

INFORMADAS O HEURISTICAS

- ☒ A*
- ☒ Avara

Inteligencia Computacional

Estrategias de búsqueda ciega o no informada

Docente:

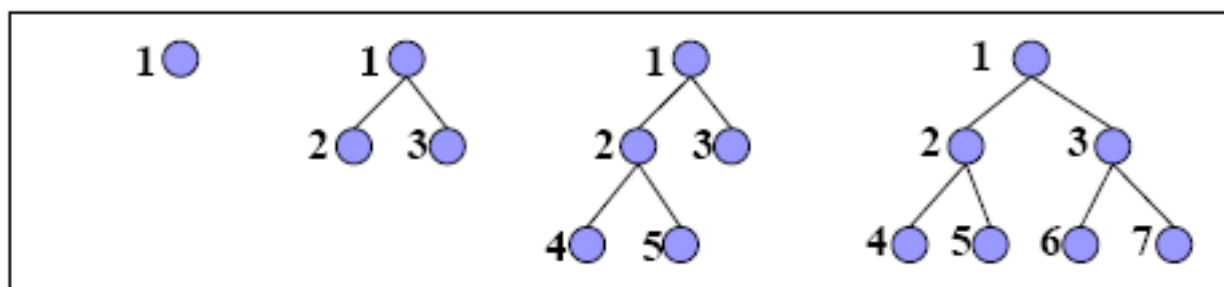
Dr. Georgina Stegmayer

gstegmayer@santafe-conicet.gov.ar

Método de Búsqueda Horizontal (MBH)

Todos los nodos de profundidad d en el árbol de búsqueda se expanden antes que los nodos de profundidad $d+1$.

Implementación: la lista de nodos no expandido es *FIFO*



Método de Búsqueda Horizontal (MBH)

Consideremos el siguiente escenario

- **b: Factor de ramificación**
- **d: profundidad de la solución**

Completa? Si (si b es finito)

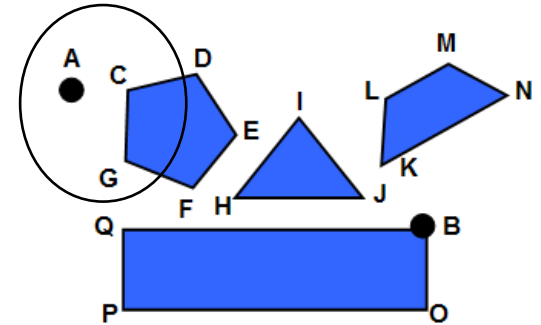
Tiempo? $1+b+b^2+b^3+\dots+b^d$

Espacio? b^d todos los nodos se mantienen en memoria

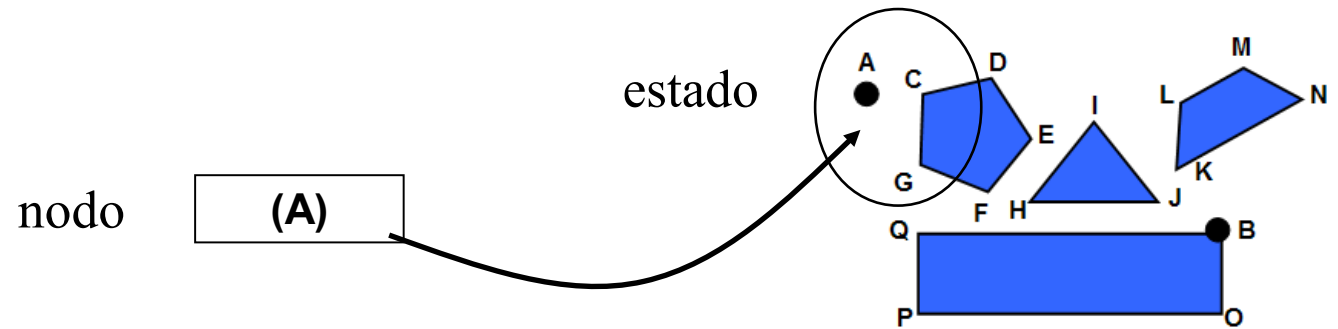
Óptima? Si, si el costo es 1 en cada etapa. En gral no es óptima.

Método de Búsqueda Horizontal (MBH)

Lista de nodos a expandir: cola (los nodos a expandir se van colocando al final de la lista)



Método de Búsqueda Horizontal (MBH)



Método de Búsqueda Horizontal (MBH)

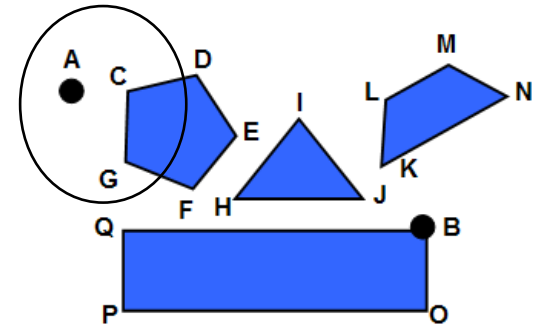
Contiene un
estado objetivo?

NO

→ Entonces expandir nodo

Acción de obtener los nodos hijos.

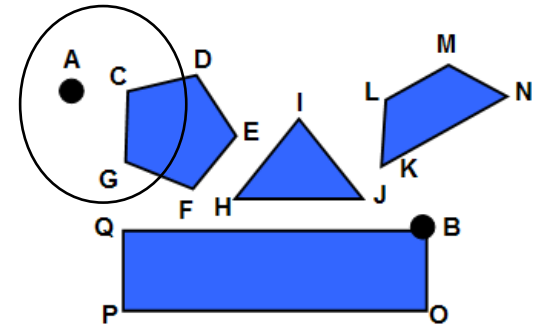
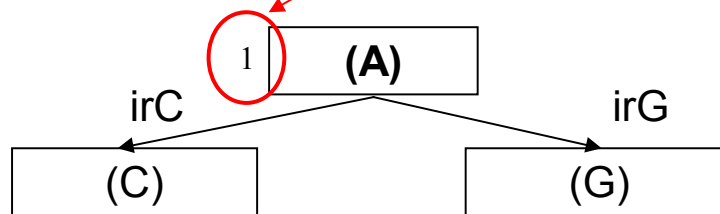
Los nodos hijos se obtienen de aplicar todos los operadores
definidos al estado representado por el nodo padre.



Método de Búsqueda Horizontal (MBH)

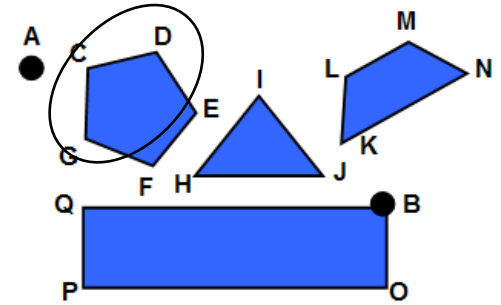
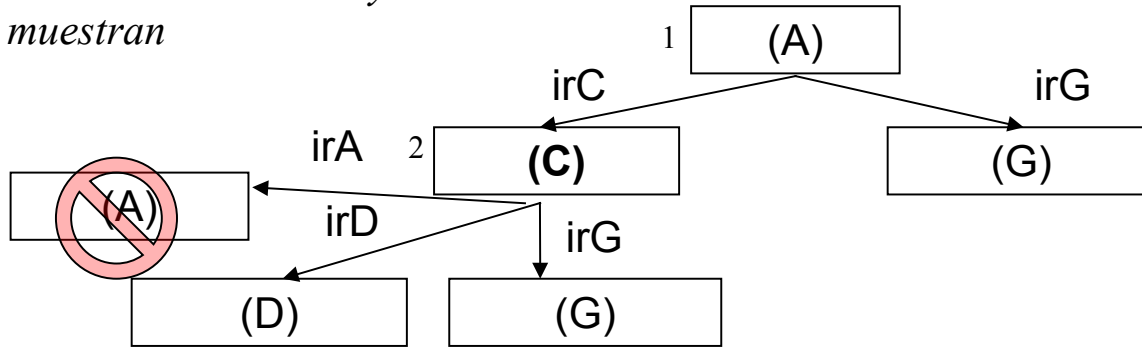
No se muestran en el árbol los operadores que no se pueden aplicar sobre el estado

Posición del nodo en la lista de nodos a expandir

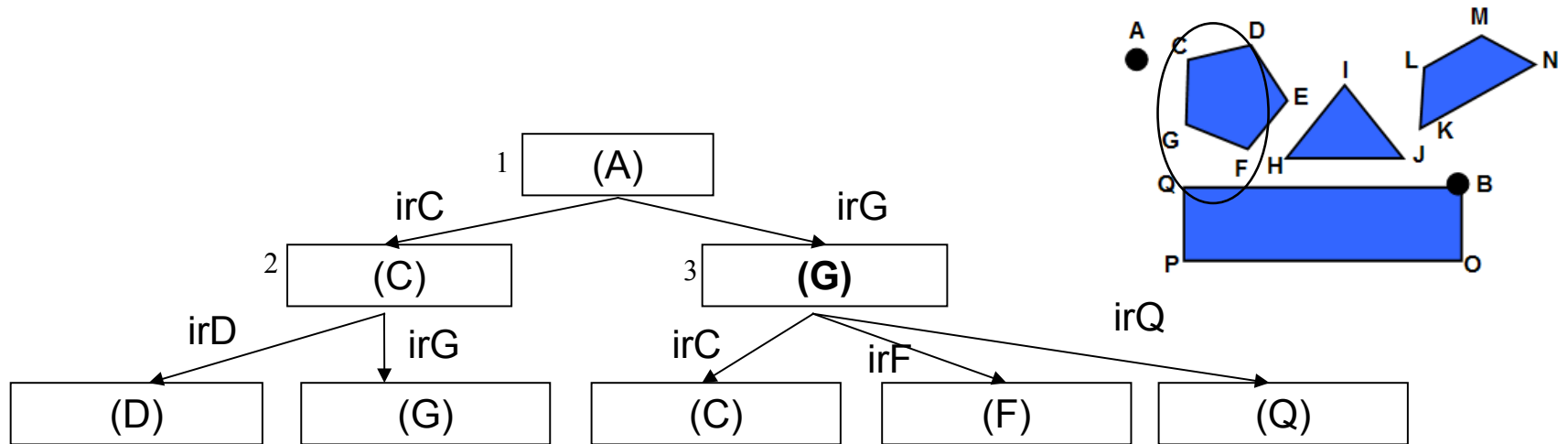


Método de Búsqueda Horizontal (MBH)

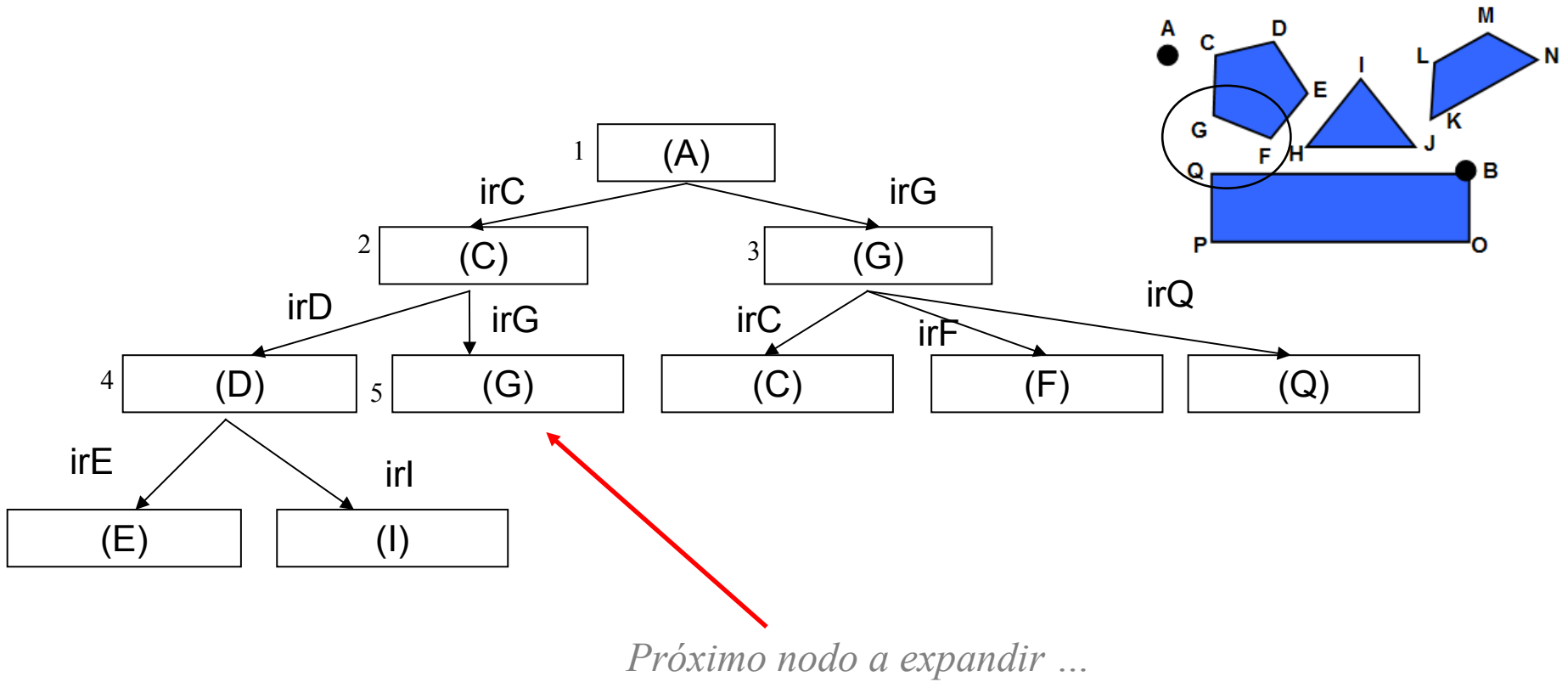
Los estados repetidos son eliminados del árbol y no se muestran



Método de Búsqueda Horizontal (MBH)

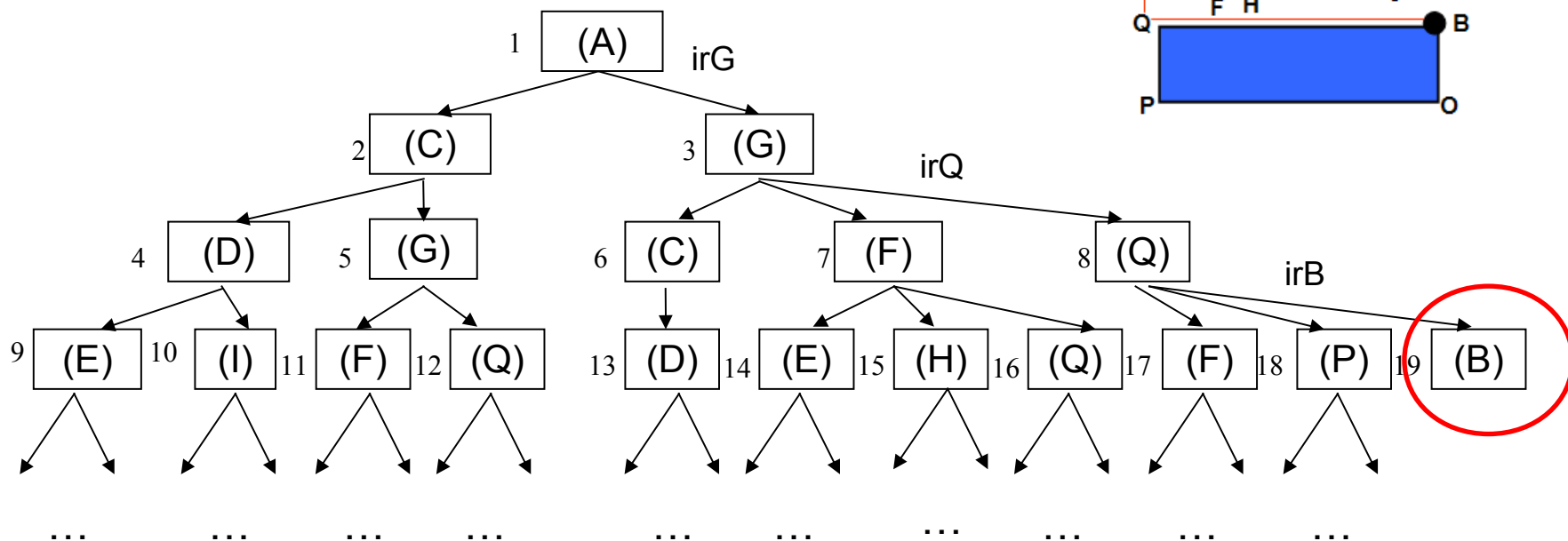
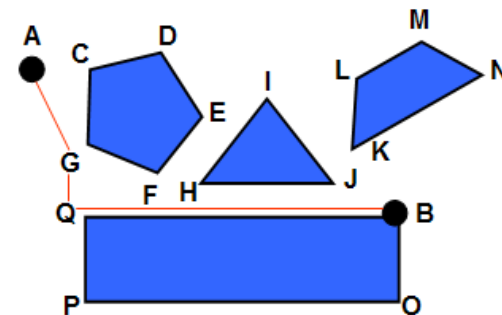


Método de Búsqueda Horizontal (MBH)



Método de Búsqueda Horizontal (MBH)

Solución encontrada con la búsqueda en amplitud u horizontal

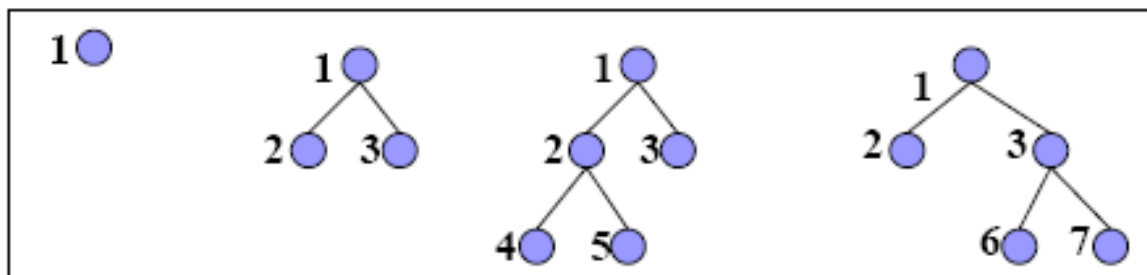


Solución (secuencia de operadores desde el E.I. hasta el E.F.): irG, irQ, irB

Método de Búsqueda Profundidad MBP

En este método los nuevos nodos generados van a la cabeza de la lista de nodos a expandir. (*LIFO*)

El *método* sólo necesita mantener un único camino desde el nodo raíz.



Método de Búsqueda Profundidad MBP

Sean:

- b : Factor de ramificación
- d : profundidad de la solución
- m : La profundidad máxima

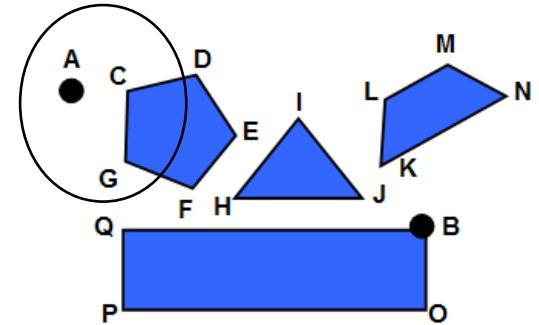
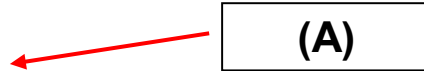
	No. Falla en espacios infinitos. Problemas con loops se controlan estados repetidos. En espacios finitos es completa.
Completa?	
Tiempo?	b^m mala cuando m es mucho mayor a d
Espacio?	bm
Óptima?	no

Método de Búsqueda Profundidad MBP

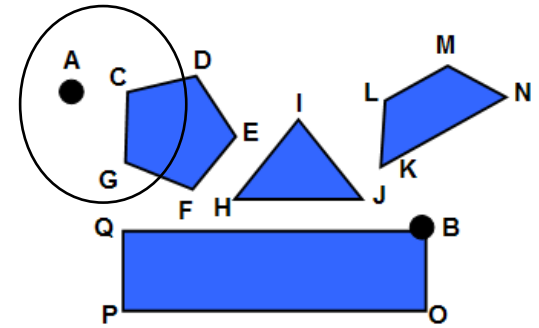
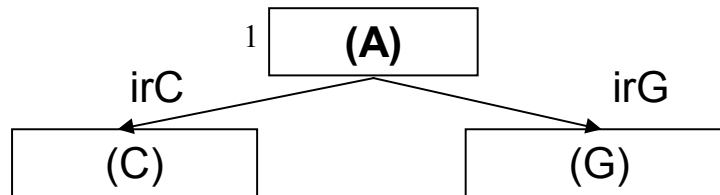
Contiene un
estado objetivo?

NO

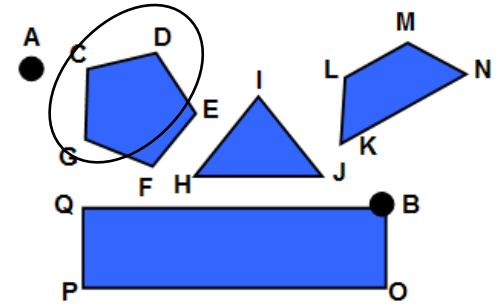
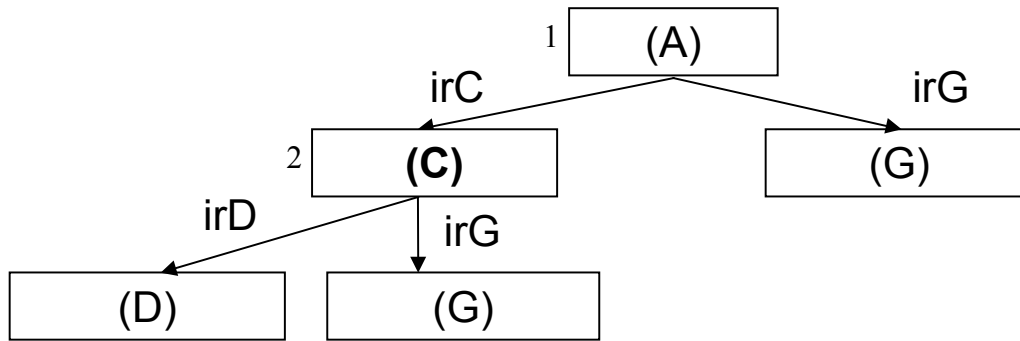
→ Entonces expandir nodo



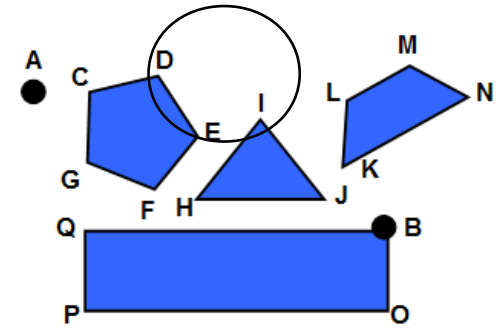
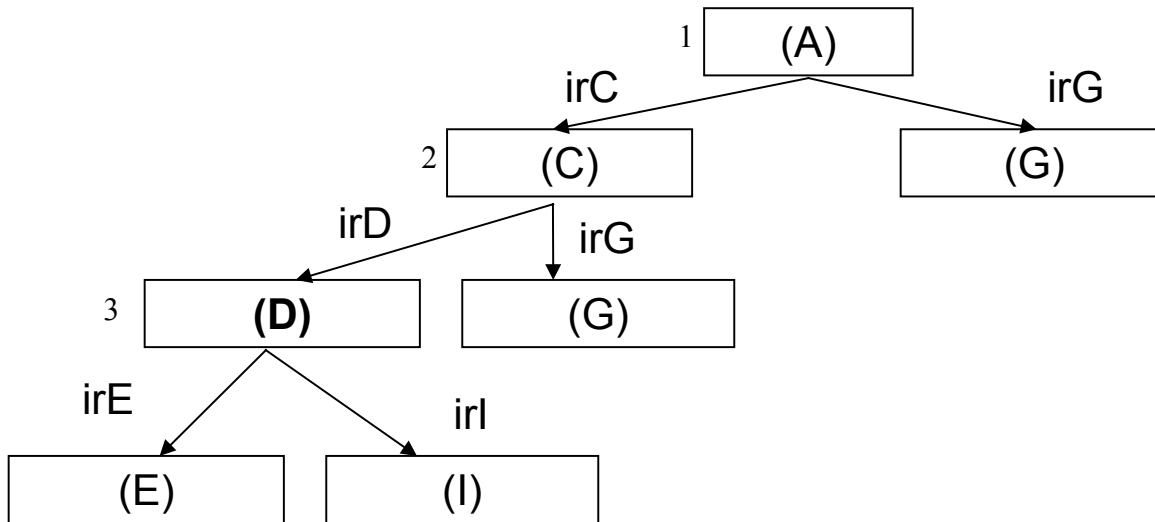
Método de Búsqueda Profundidad MBP



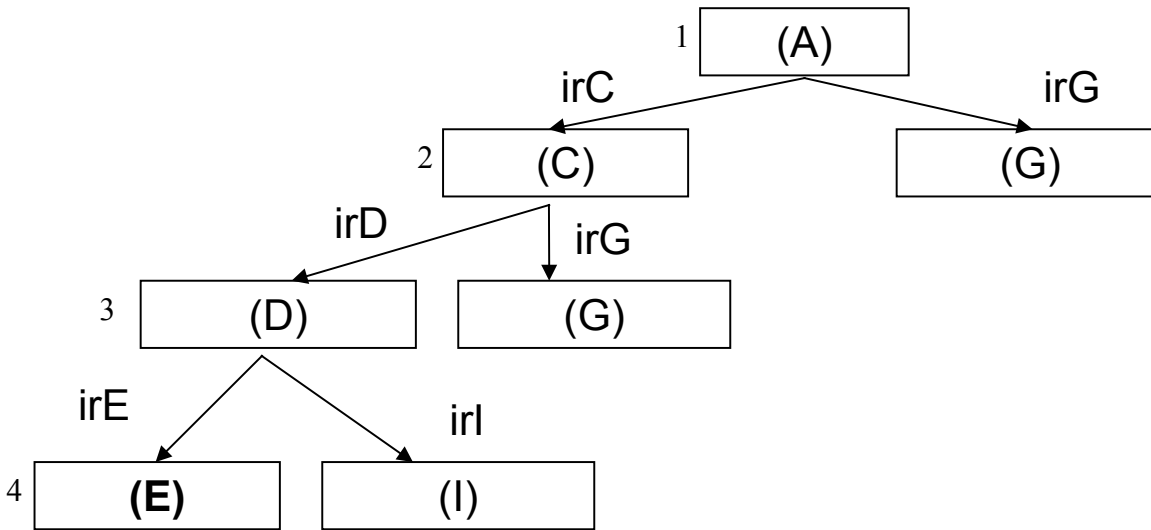
Método de Búsqueda Profundidad MBP



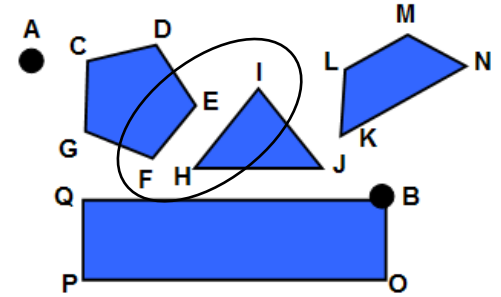
Método de Búsqueda Profundidad MBP



Método de Búsqueda Profundidad MBP

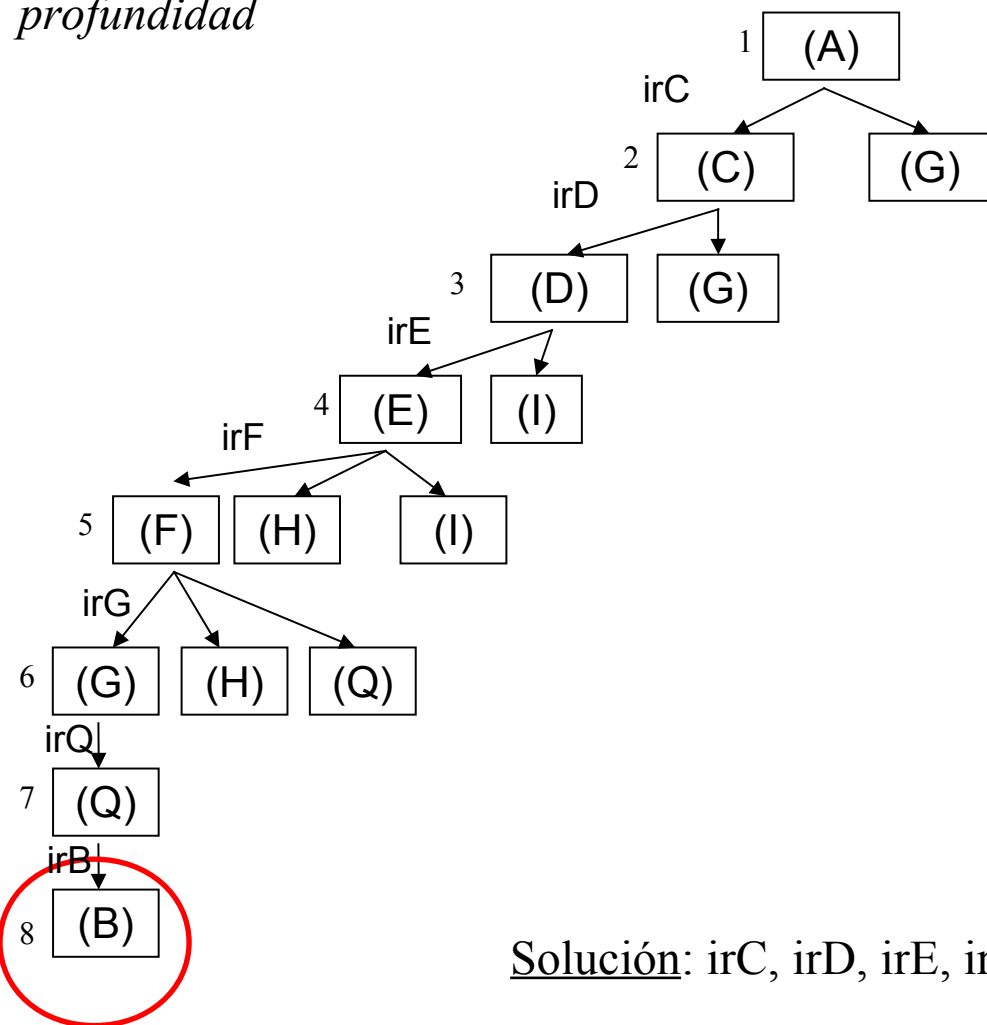


Próximo nodo a expandir ...

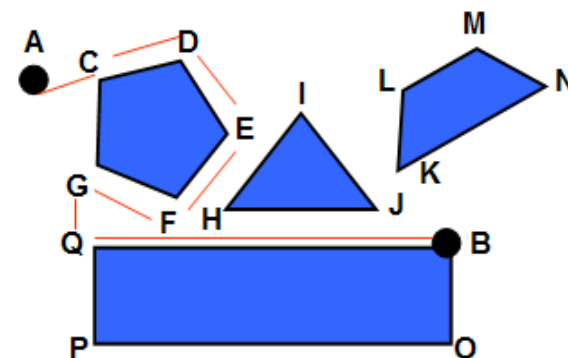


Método de Búsqueda Profundidad MBP

Solución encontrada con la búsqueda en profundidad



Solución: irC, irD, irE, irF, irG, irQ, irB



Método de Búsqueda Costo Uniforme MBCU

El MBCU modifica el MBH expandiendo primero siempre el nodo de menor costo, medido por una función que evalúa el *costo del camino*, $g(n)$. Ambos métodos coinciden cuando

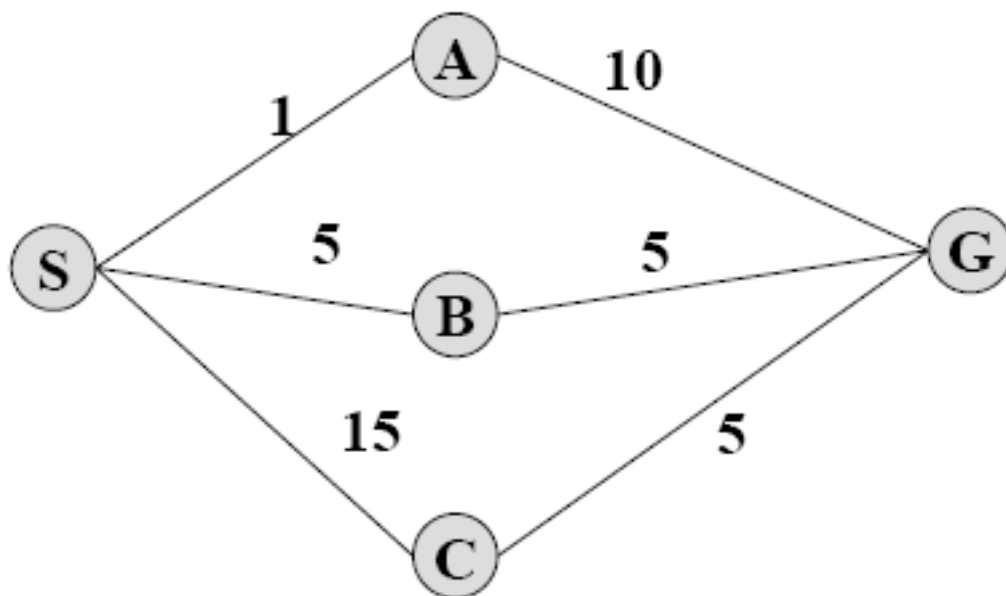
$$g(n) = \text{Profundidad}(n)$$

Implementación: lista ordenada de menor costo a mayor.

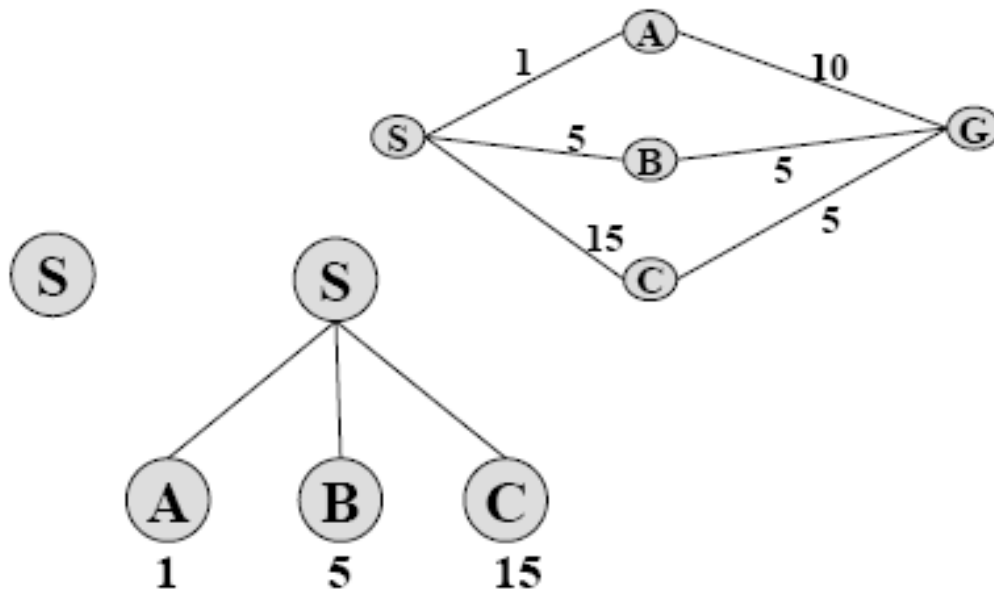
Cuando se verifican ciertas condiciones se garantiza que la primer solución que se encuentra es la de mínimo costo. La restricción es que el costo de un camino nunca debe decrecer al avanzar en su desarrollo.

Método de Búsqueda Costo Uniforme MBCU

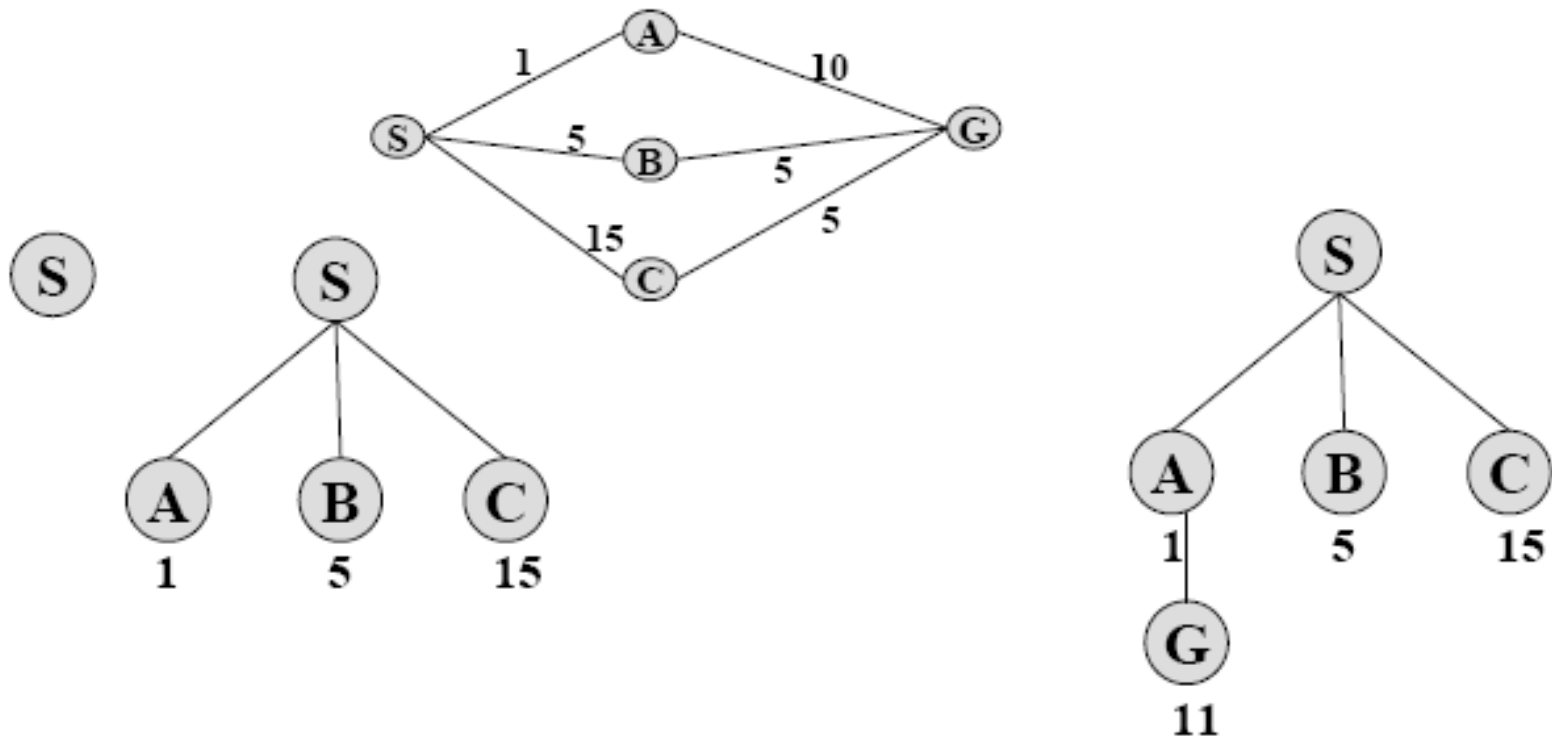
Supongamos el siguiente problema que consiste en conseguir el camino de mínimo costo entre S y G, donde la función de costos es la suma de los costos de cada arco.



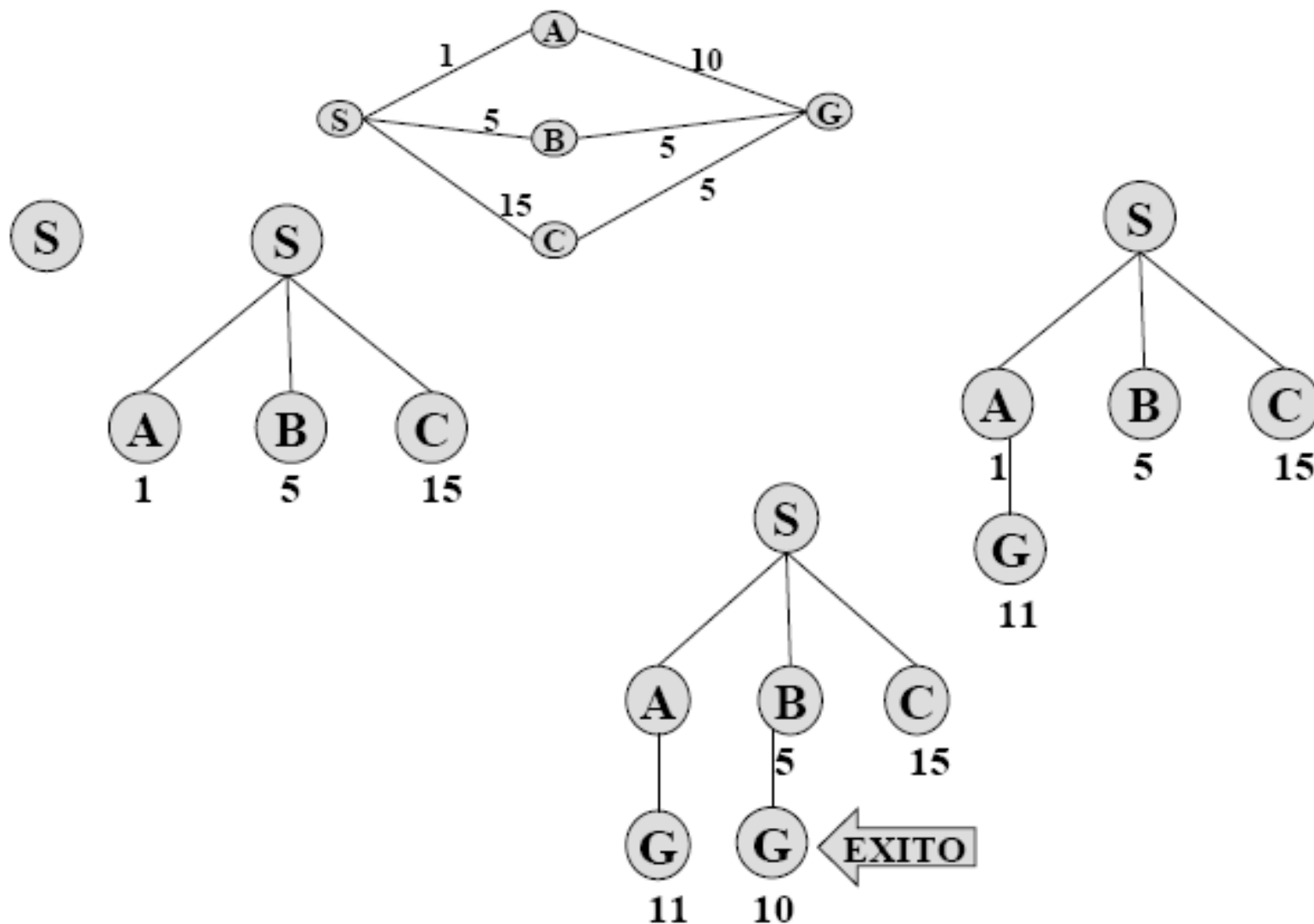
Método de Búsqueda Costo Uniforme MBCU



Método de Búsqueda Costo Uniforme MBCU



Método de Búsqueda Costo Uniforme MBCU



Método de Búsqueda Costo Uniforme MBCU

Completa? Si. Si el costo siempre aumenta o se mantiene igual

Tiempo? Nodos con $g \leq$ costo del nodo óptimo

Espacio? Nodos con $g \leq$ costo del nodo óptimo

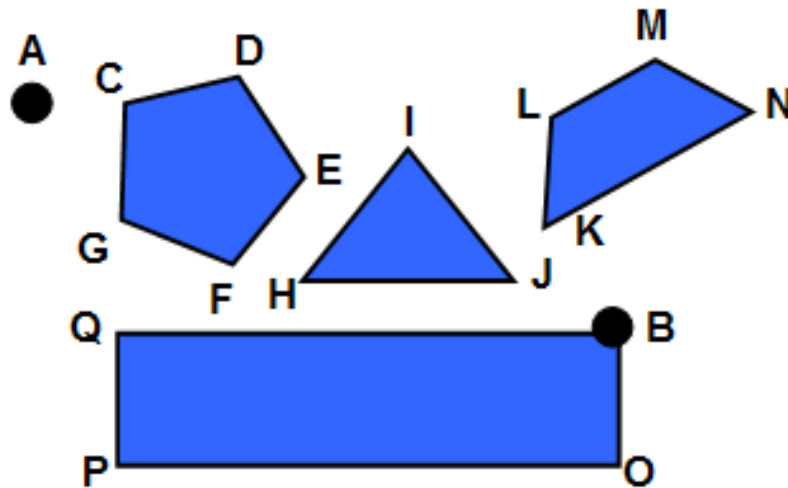
Óptima? Si

Método de Búsqueda Costo Uniforme MBCU

$g(n)$ = costo del camino

?

$g(n)$ = costo de ruta = suma de las distancias recorridas desde el nodo inicial

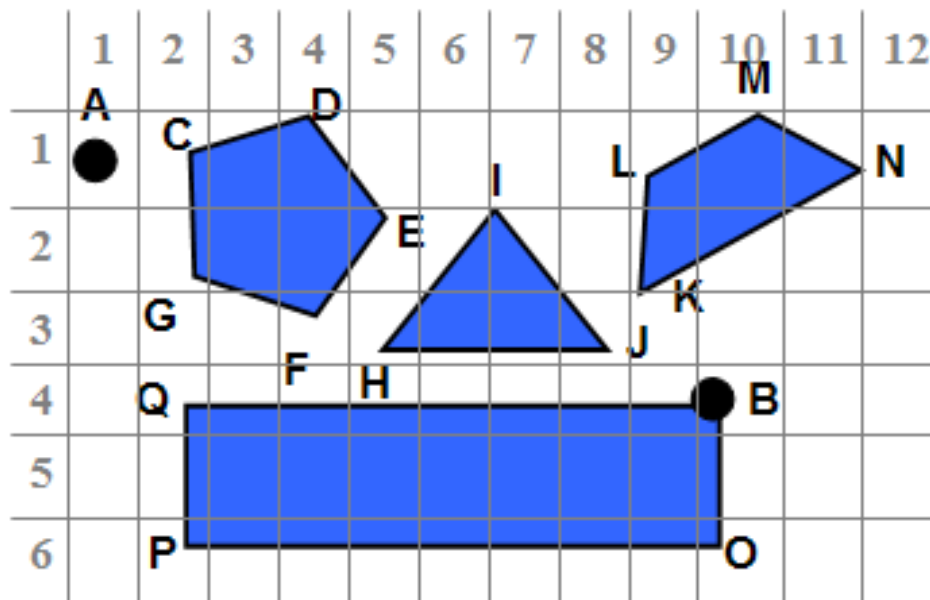


Método de Búsqueda Costo Uniforme MBCU

$g(n)$ = costo del camino

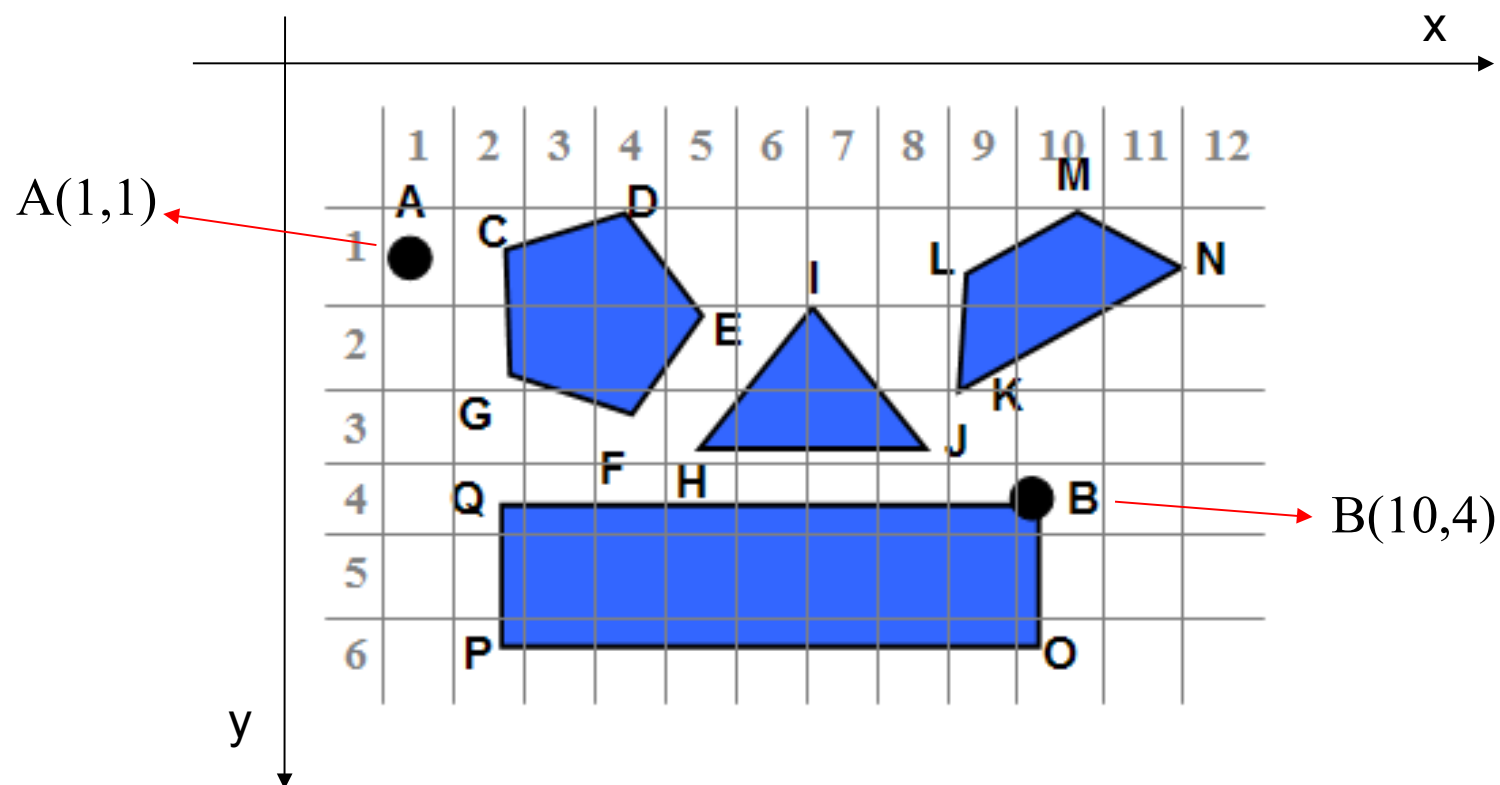
?

$g(n)$ = costo de ruta = suma de las distancias recorridas desde el nodo inicial



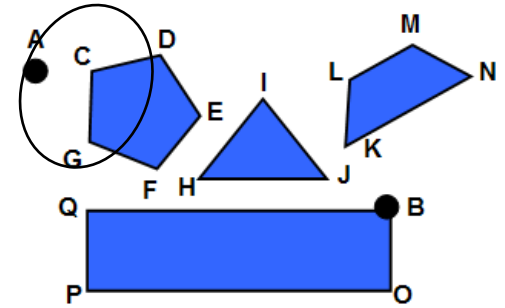
Método de Búsqueda Costo Uniforme MBCU

distancia en línea recta entre dos puntos = $\sqrt{(x_1 - x_0)^2 + (y_1 - y_0)^2}$

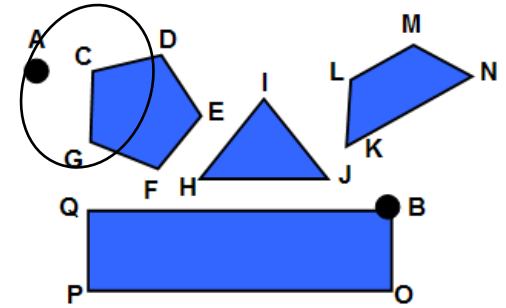
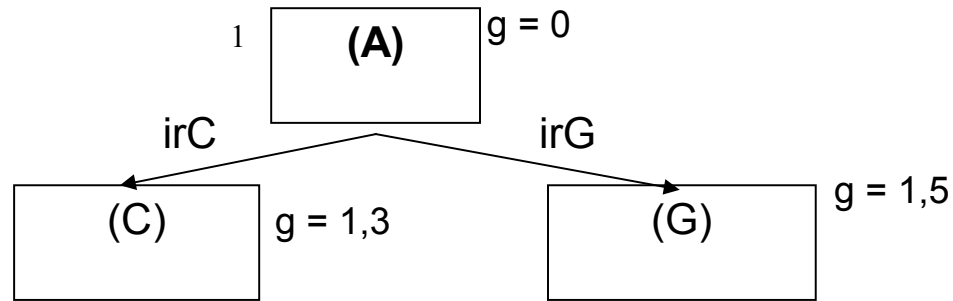


Método de Búsqueda Costo Uniforme MBCU

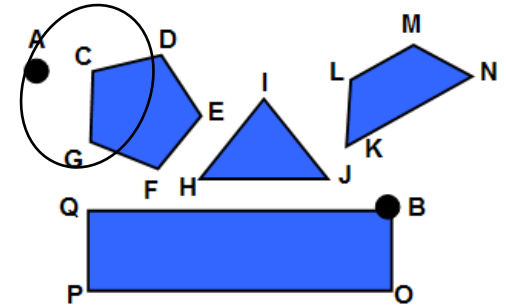
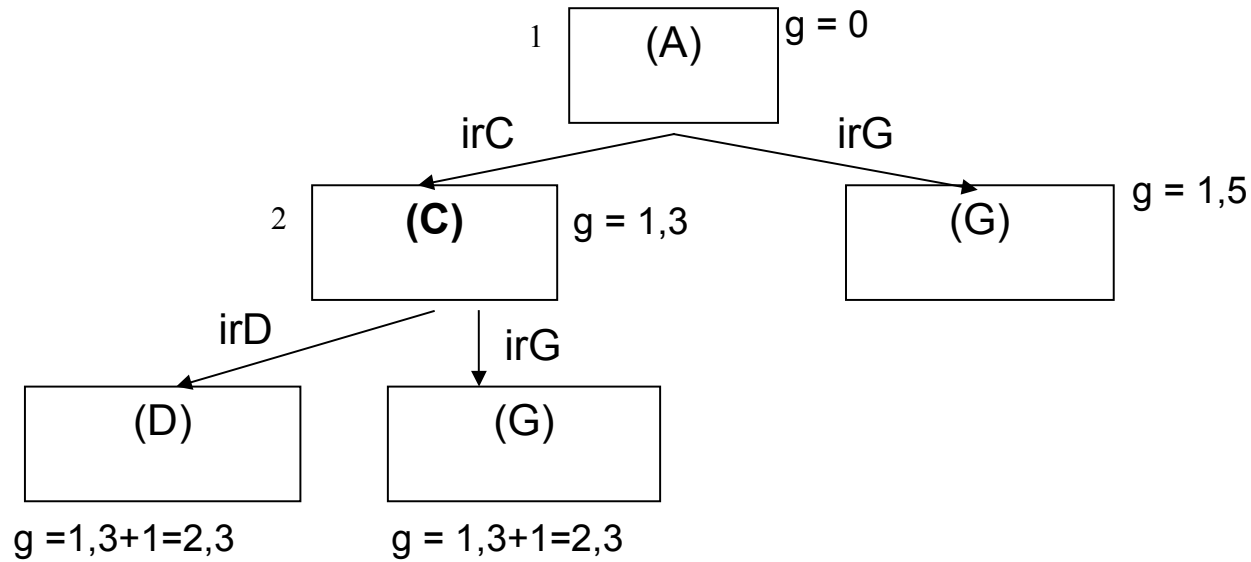
Lista de nodos a expandir: los nodos a expandir se van ordenando en la lista según su **costo**



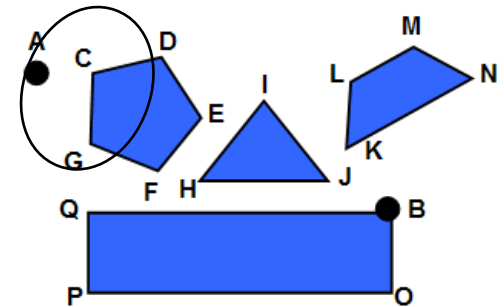
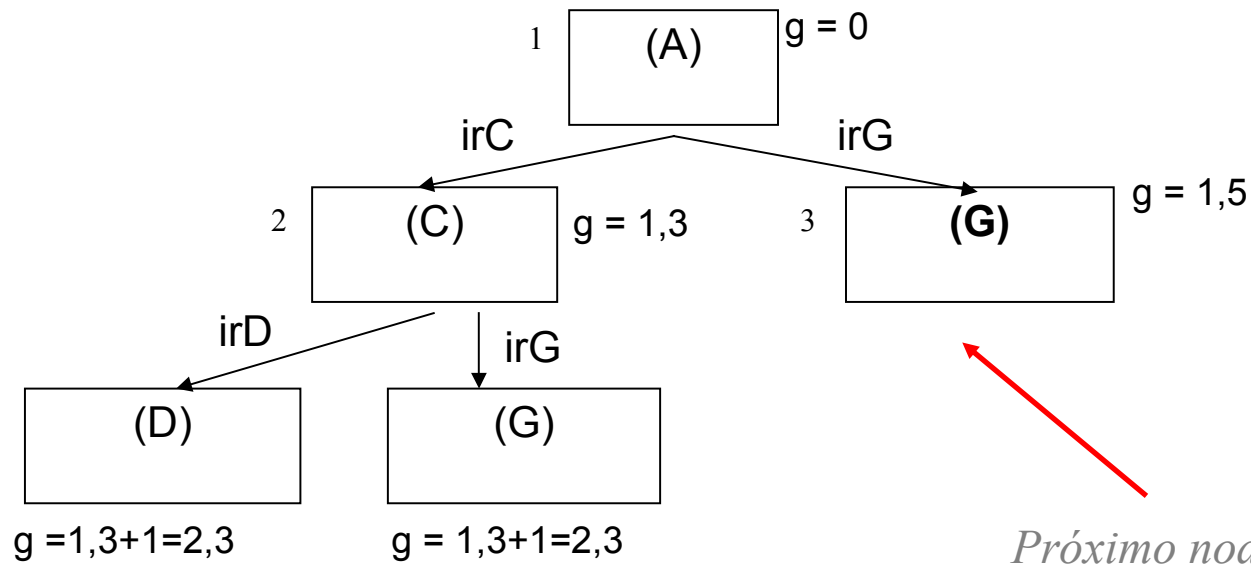
Método de Búsqueda Costo Uniforme MBCU



Método de Búsqueda Costo Uniforme MBCU

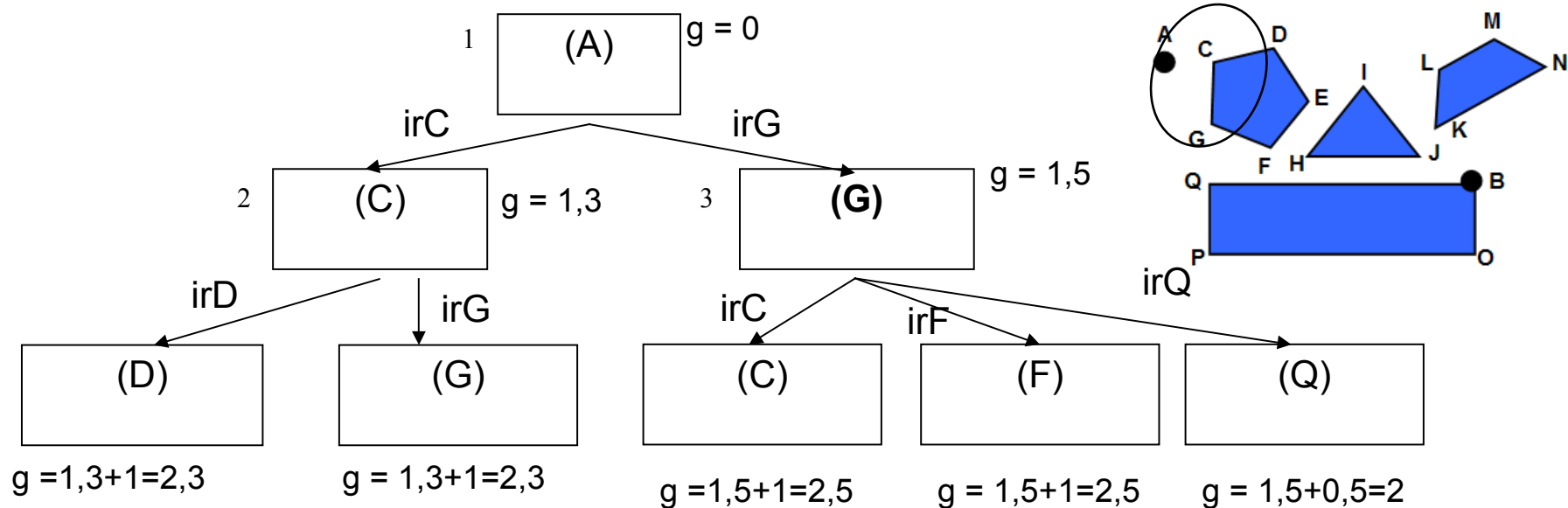


Método de Búsqueda Costo Uniforme MBCU

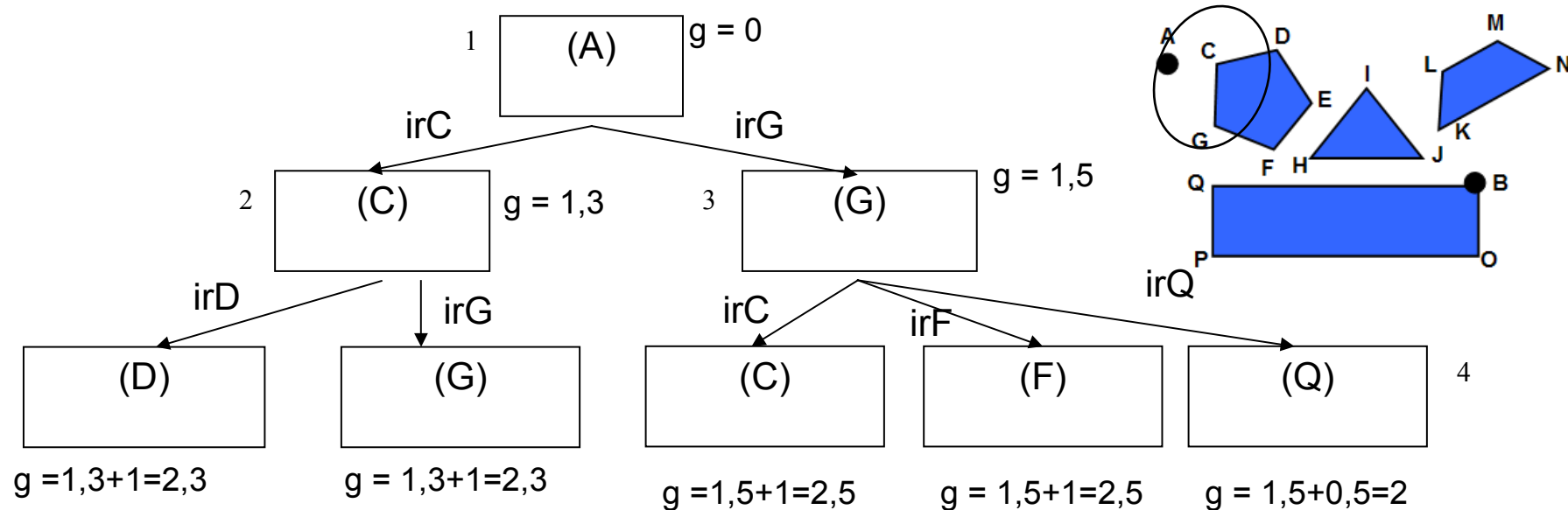


Próximo nodo a expandir ...

Método de Búsqueda Costo Uniforme MBCU



Método de Búsqueda Costo Uniforme MBCU



Método de Búsqueda Bidireccional

El método *busca hacia delante* desde el *estado inicial* y *hacia atrás* desde el *estado final*, y se detiene cuando ambas búsquedas se encuentran.

Puntos a considerar:

- Qué significa la búsqueda hacia atrás?
- Cuando los operadores son reversibles no hay problemas, sin embargo calcular el predecesor no es trivial (por ejemplo ajedrez)
- Que hacer cuando existen varios estados meta.
- Contar con verificación de nodos repetidos.
- Qué tipo de búsqueda se empleará en cada sentido?

Método de Búsqueda Bidireccional

Sean:

- ***b***: Factor de ramificación
- ***d***: profundidad de la solución

Completa? SI

Tiempo? $b^{d/2}$

Espacio? $b^{d/2}$

Óptima? Si. (si en ambas direcciones se utiliza MBH)

Inteligencia Computacional

Estrategias de búsqueda con información

Docente:

Dr. Georgina Stegmayer

gstegmayer@santafe-conicet.gov.ar

MÉTODOS INFORMADOS O HEURÍSTICOS

Si se utiliza un método de búsqueda cuya estructura general sea como la vista, el único lugar en donde colocar el *conocimiento del problema* es en la función que agrega los nuevos estados, y de esta forma determinar cual es el próximo nodo a expandir.

El conocimiento para realizar esta determinación es representado mediante una función de evaluación, que retorna una evaluación del grado de *prioridad* de expandir un nodo.

MÉTODOS INFORMADOS O HEURÍSTICOS

Si se utiliza un método de búsqueda cuya estructura general sea como la vista, el único lugar en donde colocar el *conocimiento del problema* es en la función que agrega los nuevos estados, y de esta forma determinar cual es el próximo nodo a expandir.

El conocimiento para realizar esta determinación es representado mediante una función de evaluación, que retorna una evaluación del grado de *prioridad* de expandir un nodo.

Cuando los nodos son ordenados de tal forma que el nodo con mejor evaluación es expandido primero, estamos frente al Método de Búsqueda de Primero el Mejor (MBPM).

MÉTODOS INFORMADOS O HEURÍSTICOS

- ❑ El MBPM en realidad expande el nodo que aparentemente es el mejor, ya que usualmente la función de evaluación es una estimación.
- ❑ Para orientar la búsqueda, la *función de evaluación* además del costo que demandó llegar hasta el estado debe incorporar alguna *estimación del costo del camino desde el estado hasta el estado objetivo más cercano*

MÉTODOS INFORMADOS O HEURÍSTICOS

- ❑ El MBPM en realidad expande el nodo que aparentemente es el mejor, ya que usualmente la función de evaluación es una estimación.
- ❑ Para orientar la búsqueda, la *función de evaluación* además del costo que demandó llegar hasta el estado debe incorporar alguna estimación del costo del camino desde el estado hasta el estado objetivo más cercano

Para esto hay dos enfoques:

- tratar de desarrollar primero el nodo más cercano al objetivo
- desarrollar primero el nodo sobre el camino de menor costo.

Minimizar el Costo Estimado para Alcanzar el Objetivo: BUSQUEDA AVARA

- ❑ Uno de los enfoques más simples de la estrategia MBPM es minimizar el costo estimado en alcanzar el objetivo.**
- ❑ Esto implica, desarrollar primero aquel nodo cuyo estado se supone más cercano al objetivo.**
- ❑ Este costo se estima por medio de una función heurística, h , aunque no puede determinarse exactamente**

$h(n)$ = costo estimado del camino más barato entre n y el estado objetivo.

Restricción: $h(n) = 0$ cuando n es una meta

BUSQUEDA AVARA

Sean:

- ***b***: Factor de ramificación
- ***d***: profundidad de la solución
- ***m***: profundidad máxima del espacio de búsqueda.

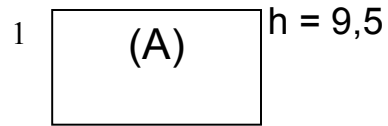
Completa? No. Pueden existir bucles (por ejemplo si la meta es Oradea: Lasi → Neamt → Lasi)

Tiempo? b^m

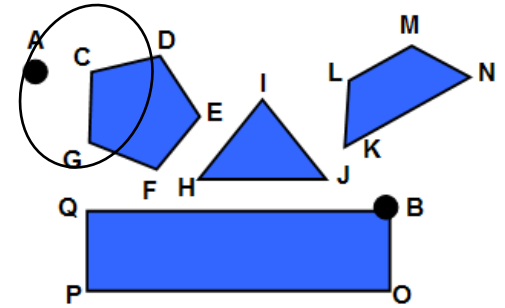
Espacio? b^m (mantiene todos los nodos en memoria)

Óptima? No.

BUSQUEDA AVARA



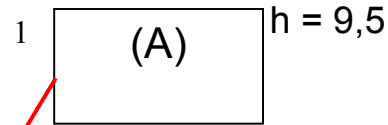
Lista de nodos a expandir: los nodos a expandir se van ordenando en la lista según el valor de $h(n)$



BUSQUEDA AVARA

Estrategia de búsqueda:

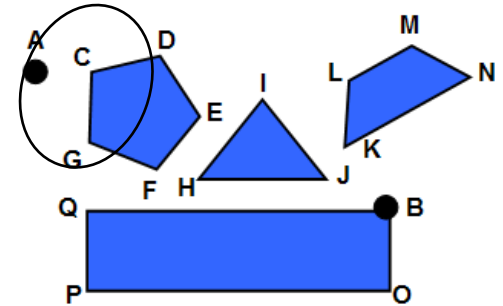
búsqueda avara



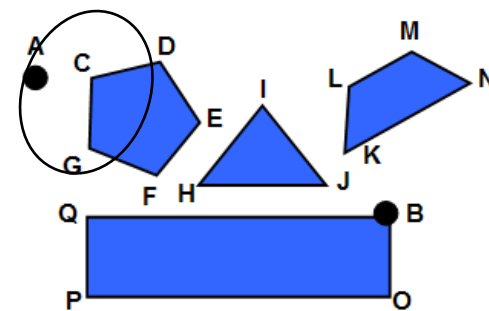
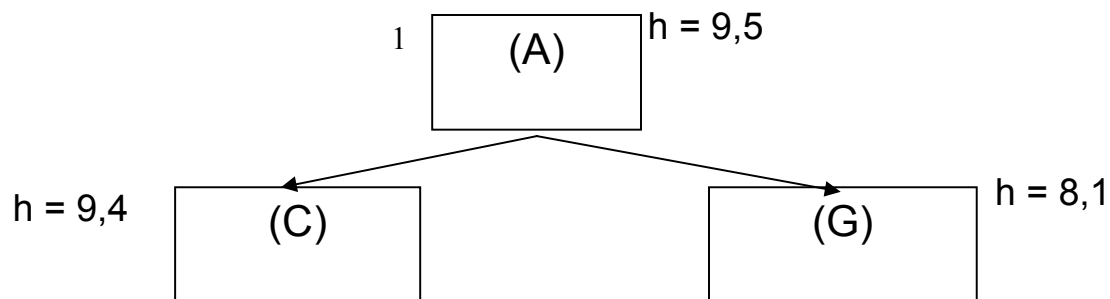
Contiene un
estado objetivo?

NO

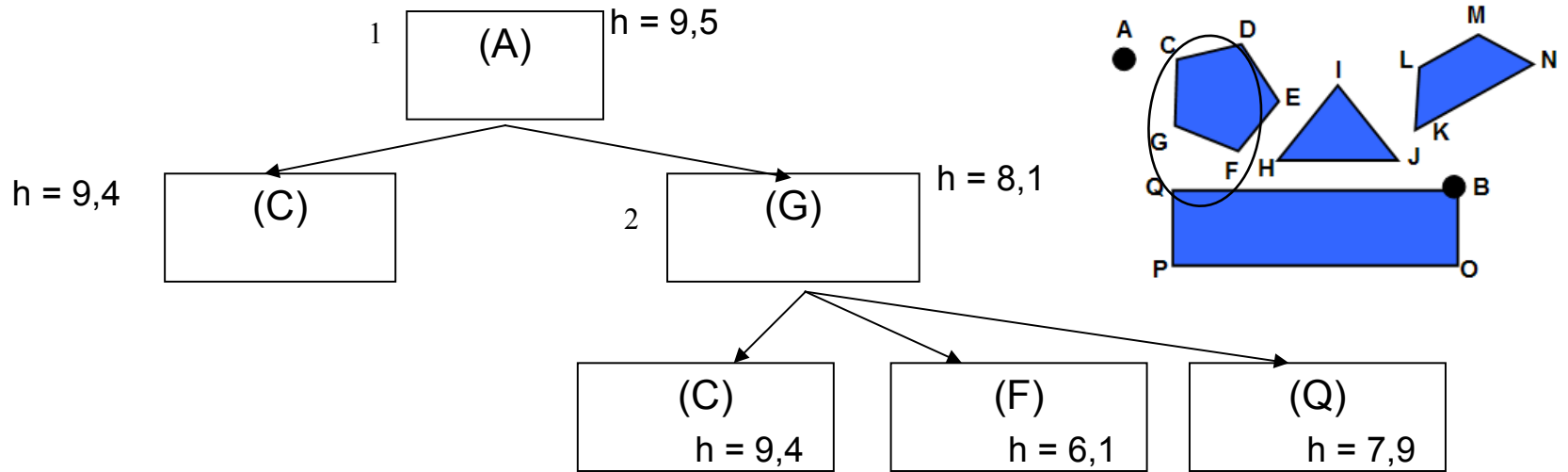
→ Entonces expandir nodo



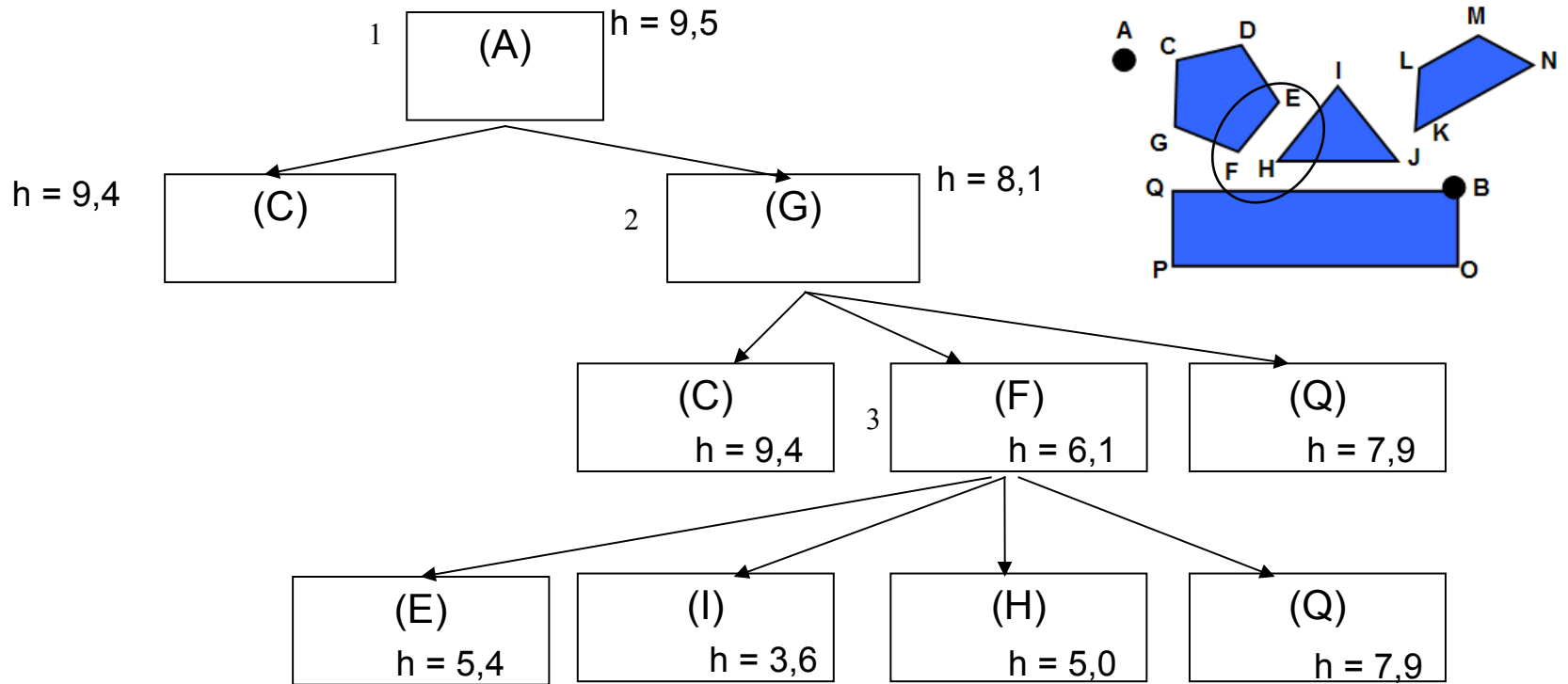
BUSQUEDA AVARA



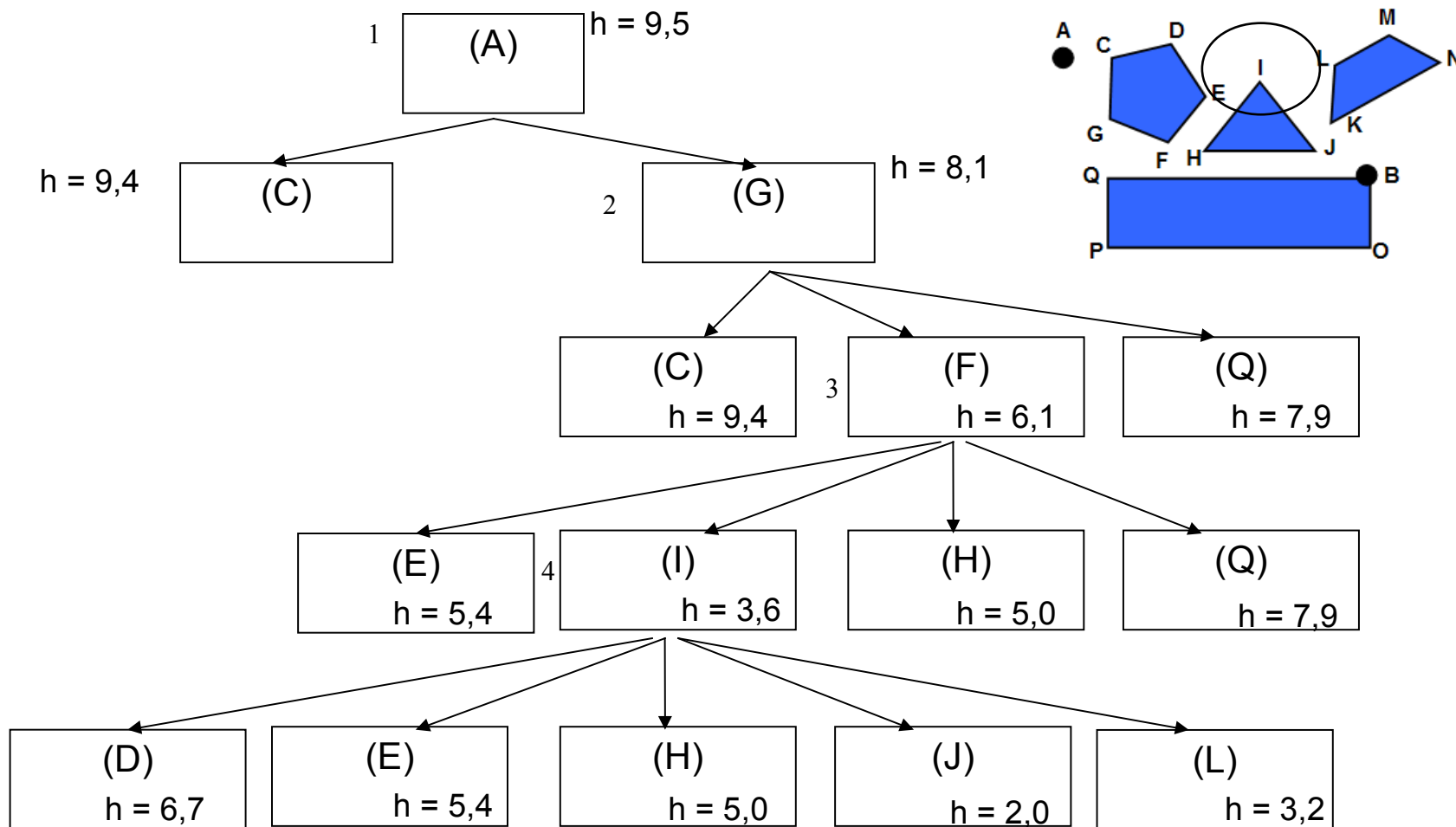
BUSQUEDA AVARA



BUSQUEDA AVARA

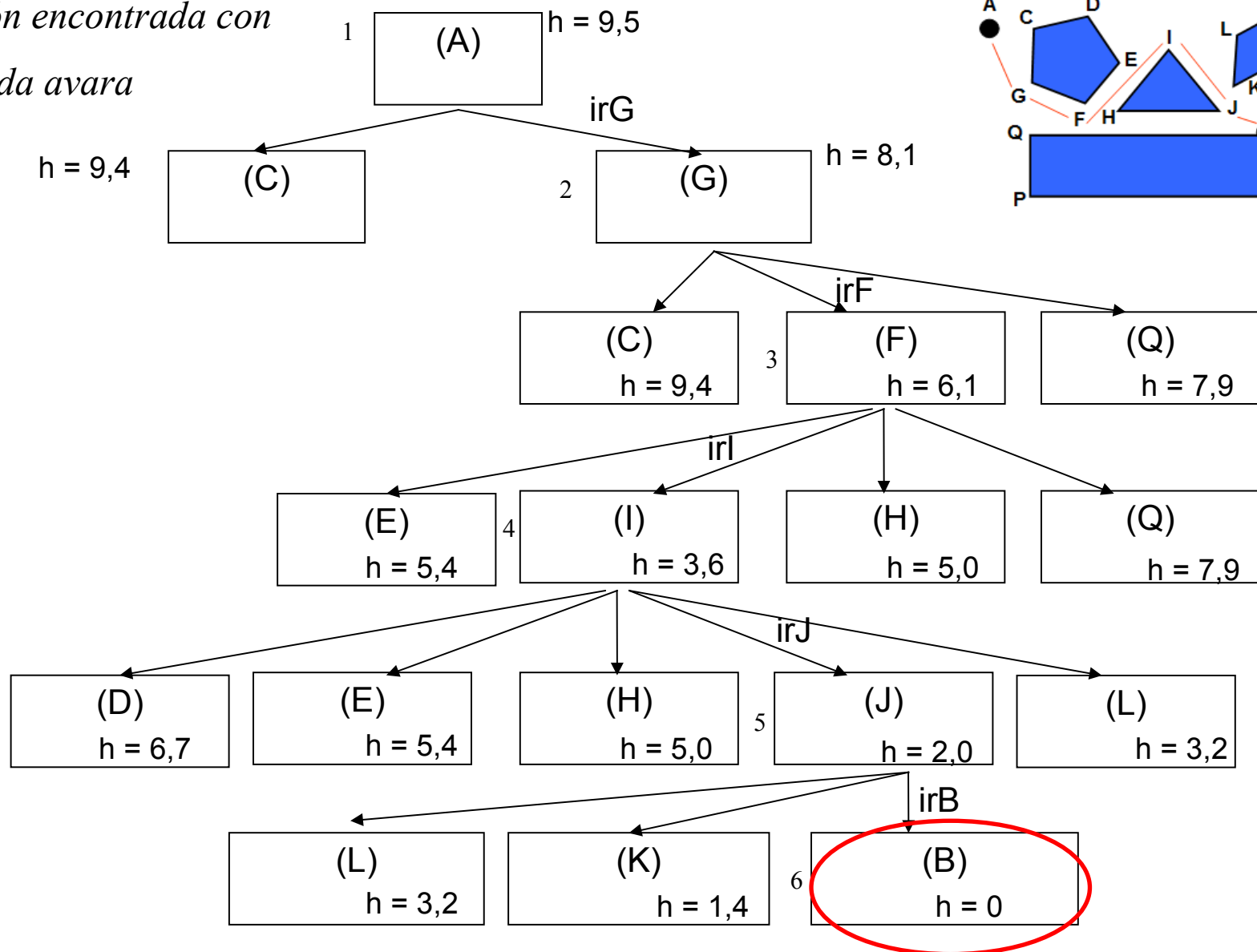


BUSQUEDA AVARA



BUSQUEDA AVARA

*Solución encontrada con
búsqueda avara*



BUSQUEDA A*

En este caso la función heurística es:

$$\underline{f(n) = g(n) + h(n)}$$

en donde

$g(n)$: da el costo desde el nodo inicial

$h(n)$: estima el mínimo costo hasta el nodo objetivo

$f(n)$: estima el costo del mínimo camino que pasa por n

La ventaja de esta función es que se puede demostrar que es *óptima y completa*, si se aplica la restricción que h nunca puede sobreestimar el costo para llegar hasta el objetivo

BUSQUEDA A*

costo de n hasta el objetivo $\geq h(n)$

Si $h(n)$ satisface esta condición se dice que es un *heurístico admisible*, o que la función satisface la *condición de admisibilidad*.

Cómo se plantea este método en el problema del viajante entre las ciudades?

$g?$ Km recorridos

$h?$ Distancia en línea recta

BUSQUEDA A*

$$f(n) = \text{función heurística} = g(n) + h(n)$$

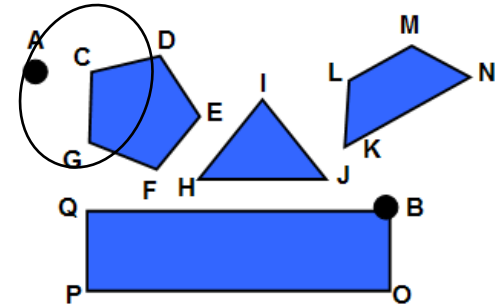
$g(n)$ = costo de ruta = suma de las distancias recorridas desde el nodo inicial (A)

$h(n)$ = heurística = distancia en línea recta hasta el nodo objetivo (B)

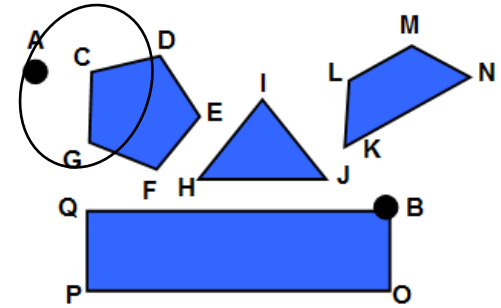
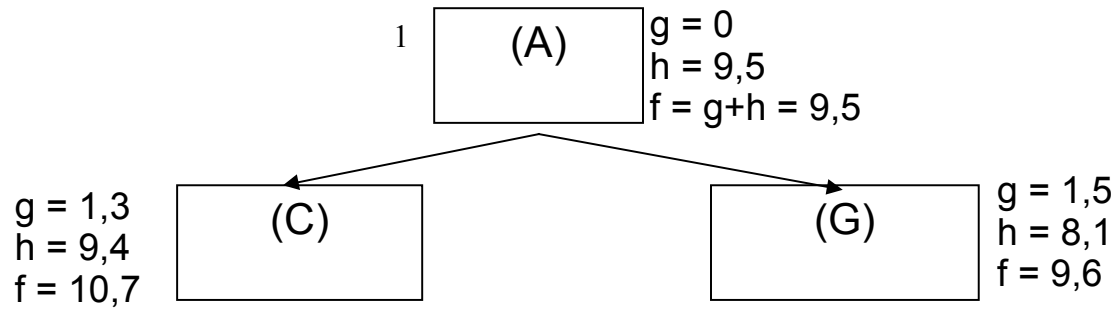
BUSQUEDA A*

1

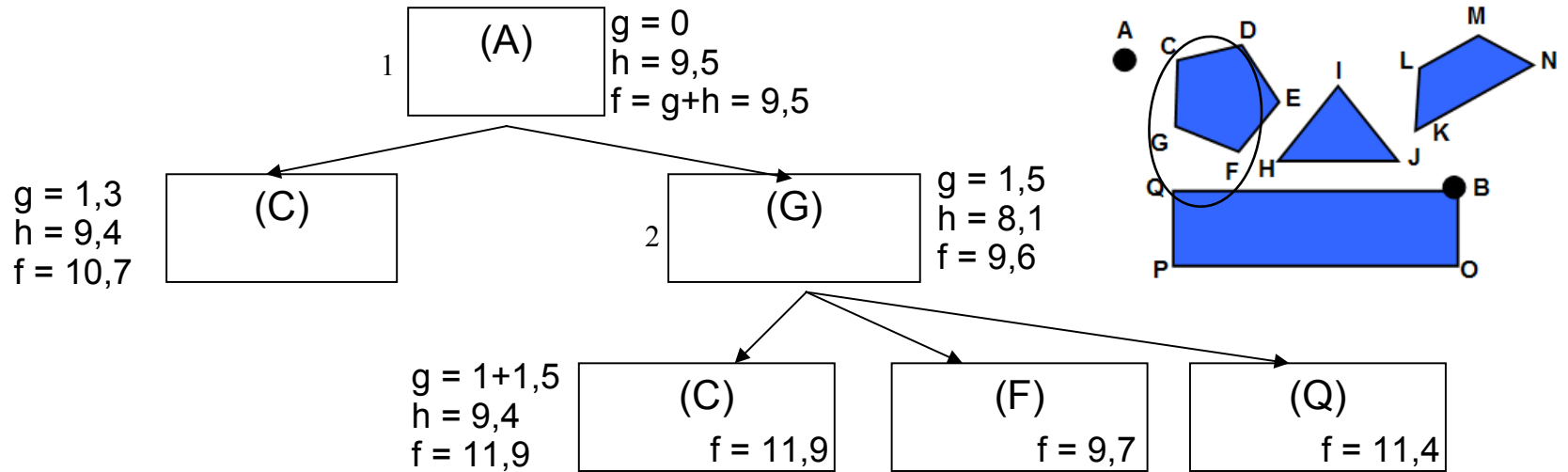
(A)	$g = 0$ $h = 9,5$ $f = g+h = 9,5$
-----	---



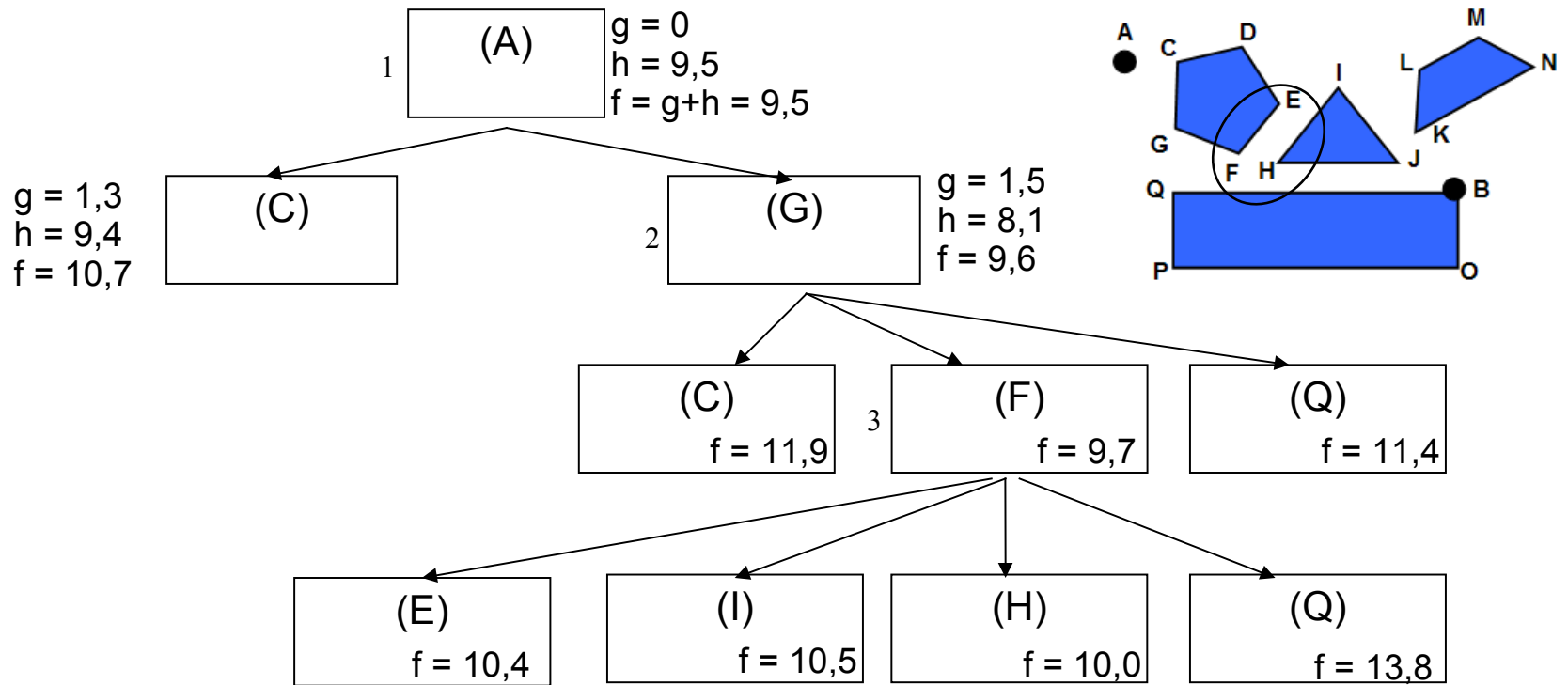
BUSQUEDA A*



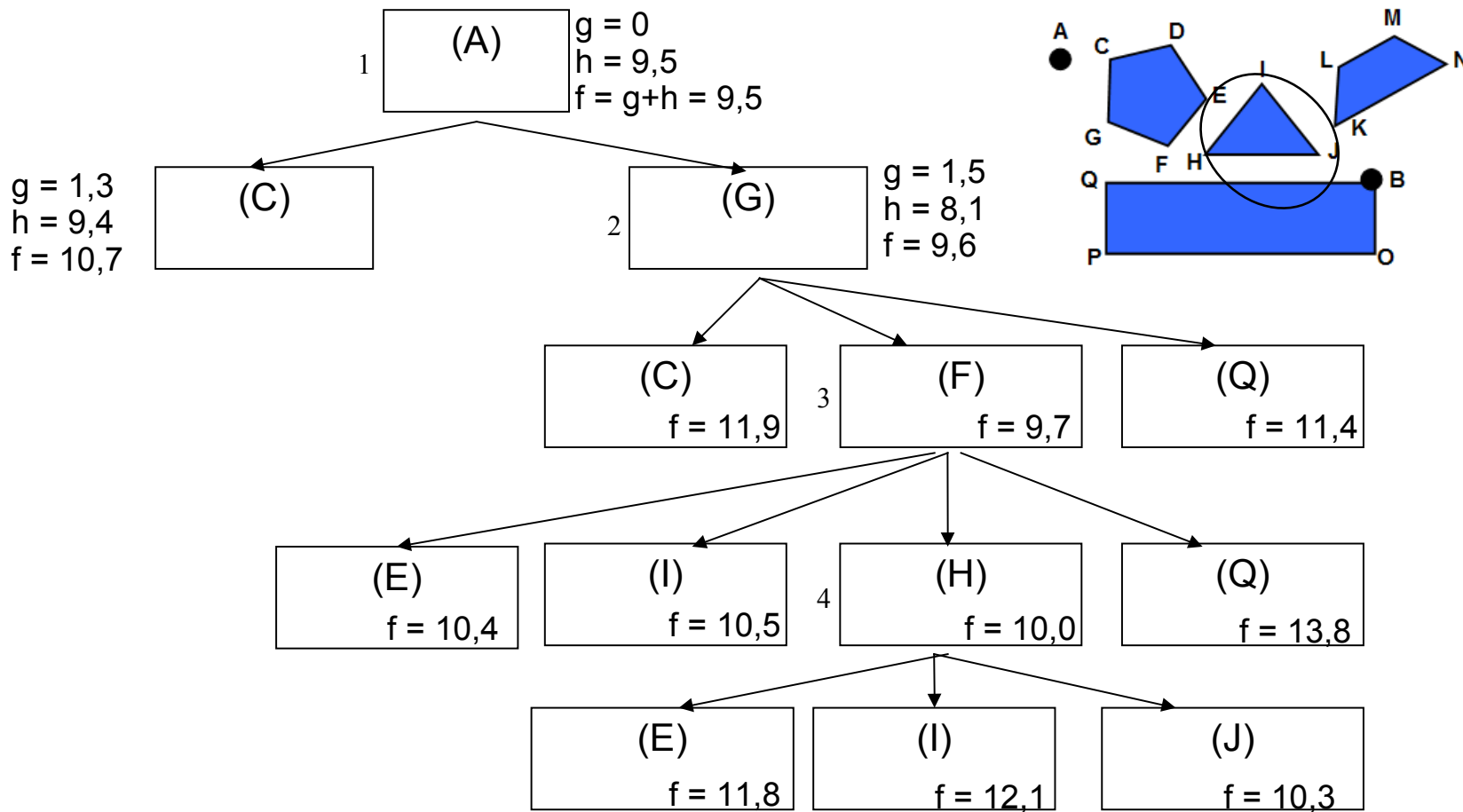
BUSQUEDA A*



BUSQUEDA A*

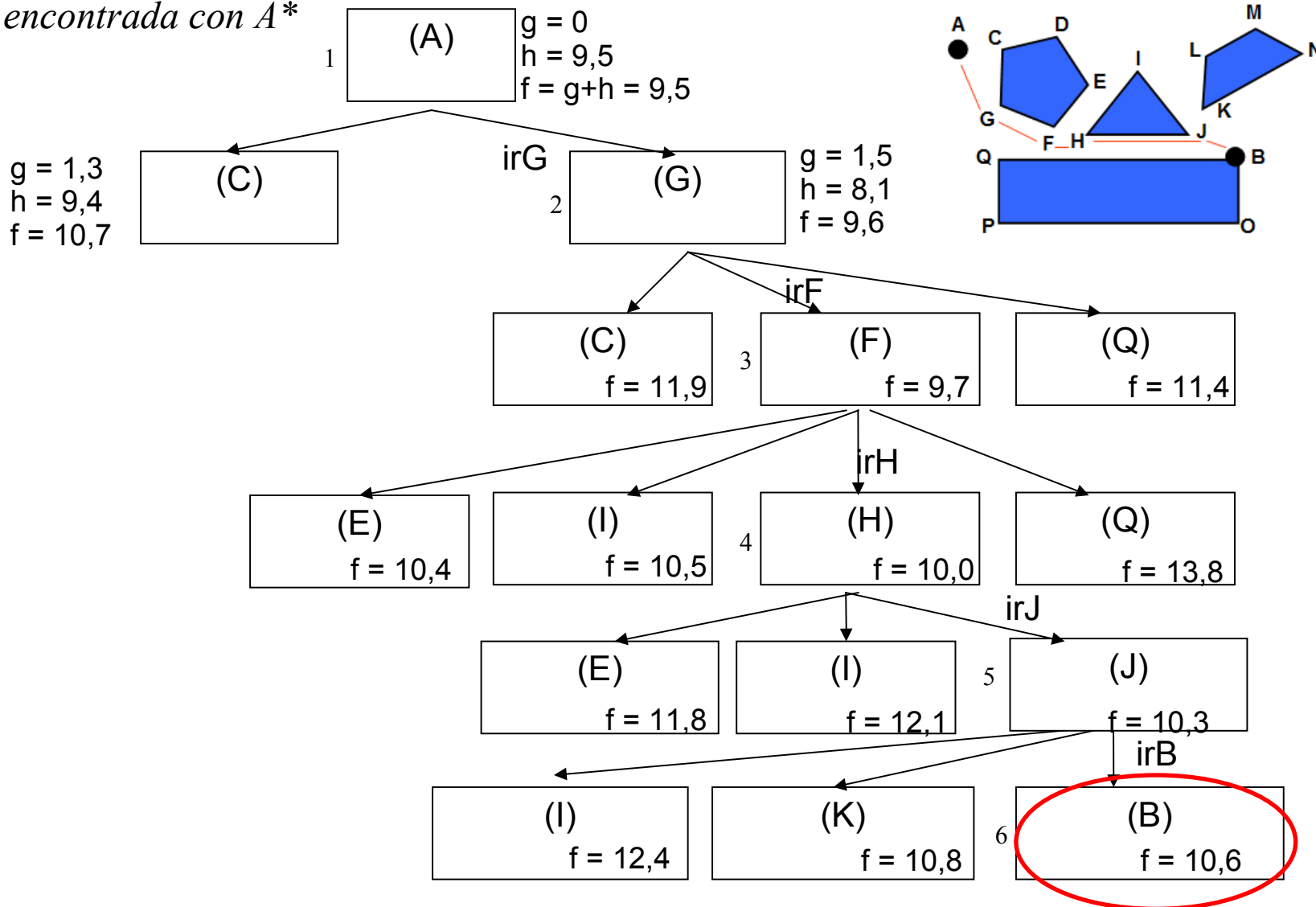


BUSQUEDA A*



BUSQUEDA A*

*Solución encontrada con A**



Problema de navegación de robots

PROBLEMA: encontrar el camino más corto que debe seguir un robot para ir desde A hasta B

	Método	Óptimo?	Completo?	solución	distancia recorrida
Búsqueda no informada	Amplitud	no	sí	A,irG,irQ,irB	11,2
	Profundidad	no	no	A,irC,irD,irE,irF,irG,irQ,irB	16,8
Búsqueda informada	Búsqueda avara	no	no	A,irG,irF,irI,irJ,irB	11,6
	A*	sí	sí	A,irG,irF,irH,irJ,irB	10,6

SOLUCIÓN: el camino más corto entre A y B es A,irG,irF,irH,irJ,irB y vale 10,6.

Inteligencia Computacional

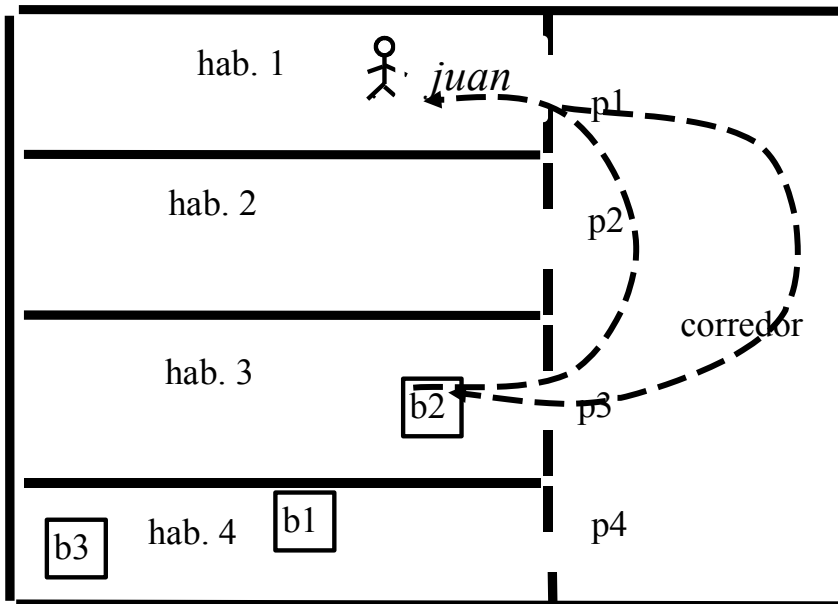
Planificación

Docente:

Dr. Georgina Stegmayer

gstegmayer@santafe-conicet.gov.ar

Planificación: proceso de búsqueda y articulación de una **secuencia de acciones** que permiten alcanzar un objetivo



Problema: llevar la caja b2 desde la hab.3 a la hab. 1

Plan solución:

- Ir corredor
- Ir hab3
- Tomar caja b2
- Ir con la caja al corredor
- Ir con la caja a hab 1

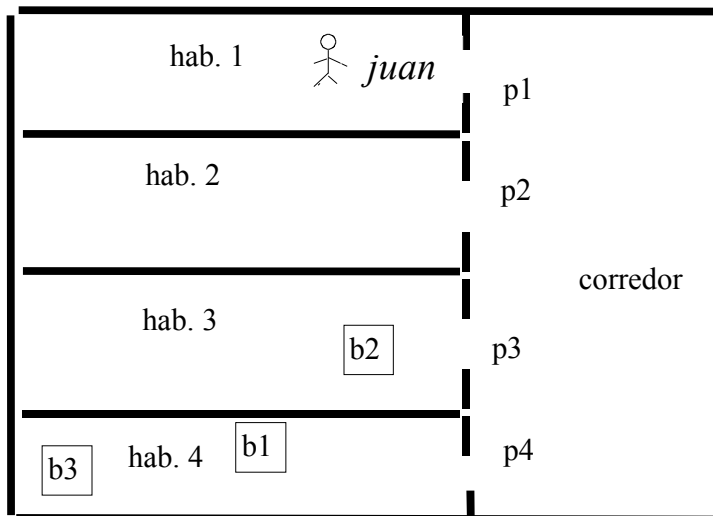
Planificación

Recordando conceptos de *BÚSQUEDA*:

- ✓ **Representación de acciones:** consisten en funciones que generan descripciones de estados sucesores
- ✓ **Representación de estados:** utiliza representaciones completas de los estados.
- ✓ **Representación de objetivos:** solo se tiene un test para verificar el objetivo y una función heurística

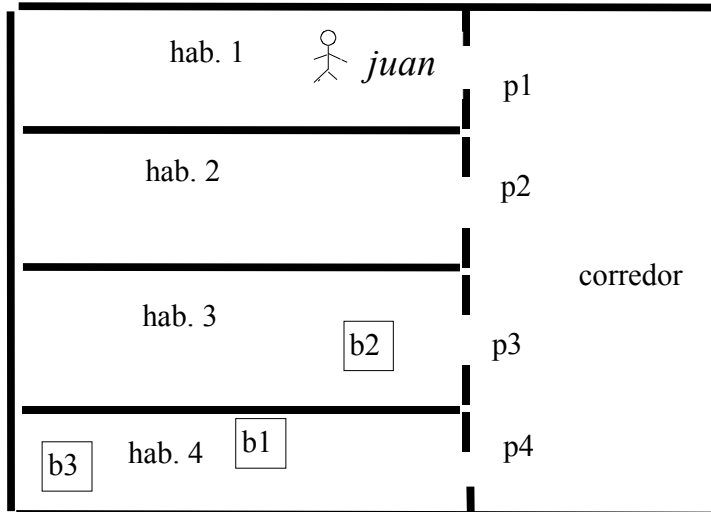
Plantear el problema de búsqueda usualmente exige demasiadas *acciones* y demasiados *estados* para analizar.

Planificación

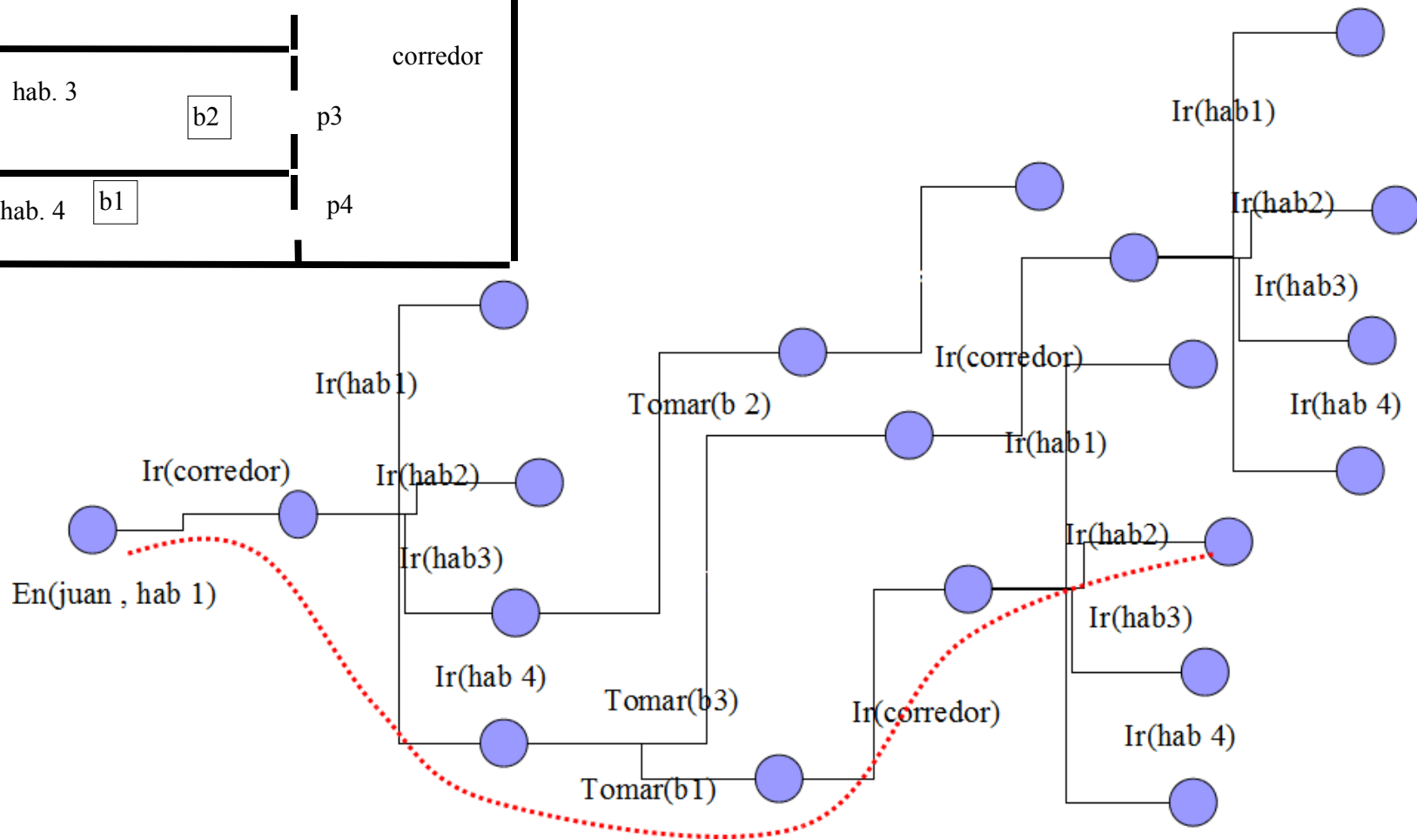


El robot *Juan* debe llevar la caja *b1* a la hab. 2

Planificación



El robot *Juan* debe llevar la caja *b1* a la hab. 2



Planificación

Características:

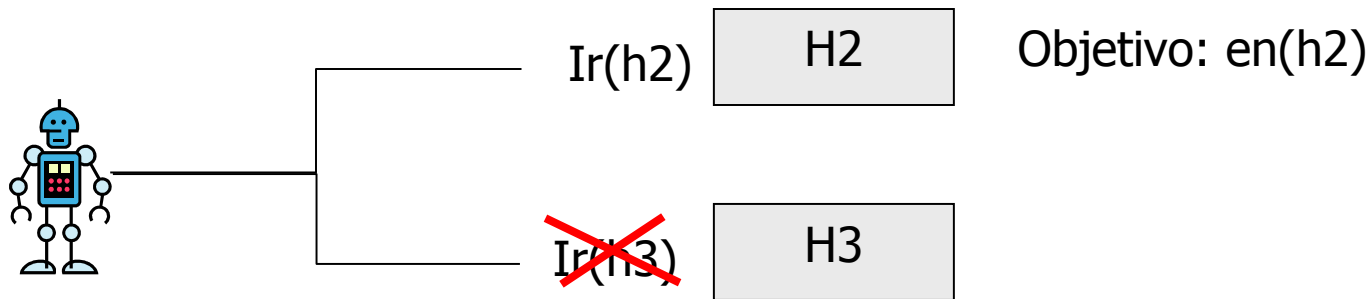
- ✓ Representación explícita del objetivo
- ✓ Descomposición de problemas en sub-problemas.
- ✓ Las acciones trabajan con expresiones de objetivos explícitos.
- ✓ Lenguaje expresivo y suficientemente restrictivo que permita ser tratado por algoritmos operativos y eficientes: **STRIPS**
- ✓ Suposición de independencia de objetivos.
- ✓ Entornos: completamente observables, determinísticos, estáticos y discretos

Planificación

- ✓ **Abrir la representación de:** estados, objetivos y acciones.
- ✓ Usar un lenguaje formal para describirlos
- ✓ Permitir realizar conexiones entre *estados* y *acciones*.

Por ejemplo,

“Ir(x) se reduce a estar en x”



El algoritmo de **búsqueda** debería generar un número de nodos demasiado grande, ya que no posee acceso a la estructura del objetivo.

Planificación

- ✓ **Explotar la independencia entre los objetivos a alcanzar.**
- ✓ El planificador puede trabajar sobre las sub-metas independientemente, aunque después necesita trabajo adicional para combinar los sub-planes resultantes
- ✓ En algunos problemas avanzar sobre una sub-meta puede estar afectando a otra sub-meta.

Por ejemplo, si tuviéramos como objetivo:

$$\text{En}(\mathbf{b1}, \text{hab } 2) \wedge \text{En}(\mathbf{b2}, \text{hab } 2)$$

Se podría obtener un **sub-plan** que obtenga el primer objetivo y otro **sub-plan** para el segundo.

Planificación

Lenguaje Formal:

- ✓ **Representación de estados:** conjunto de literales lógicas *positivas*. Ej: en(caja1, hab1), estado(heladera, sucia)

Deben ser: **ground** y **libre de funciones**

Planificación

Lenguaje Formal:

- ✓ **Representación de estados:** conjunto de literales lógicas *positivas*. Ej: en(caja1, hab1), estado(heladera, sucia)

Deben ser: **ground** y **libre de funciones**

sobre(A,B), sobre(bloque(a), mesa)

Planificación

Lenguaje Formal:

Representación de estados: conjunto de literales lógicas *positivas*. Ej: en(caja1, hab1), estado(heladera, sucia)

Deben ser: **ground** y **libre de funciones**

sobre(A,B), sobre(bloque(a), mesa)

No válida

- ✓ Presunción de *Mundo Cerrado*
- ✓ Espacio de estado finito

Planificación

Lenguaje Formal:

Representación de estados: conjunto de literales lógicas *positivas*. Ej: en(caja1, hab1), estado(heladera, sucia)

Deben ser: **ground** y **libre de funciones**

sobre(A,B), sobre(bloque(a), mesa)

No válida

✓ Presunción de *Mundo Cerrado*

✓ Espacio de estado finito

Todo lo que no figura explícitamente como un hecho y tampoco se puede deducir, es falso

Planificación

Representación de acciones:

Una **acción** se representa a través de:

- precondiciones
- efectos

Un **operador** STRIPS consta de tres partes:

- ✓ Un conjunto **PC** de literales denominado *precondiciones* del operador.
- ✓ Un conjunto **B** de literales base denominado *lista borrar*
- ✓ Un conjunto **A** de literales base denominado *lista añadir*.

Planificación

Los *operadores STRIPS* se definen mediante *esquemas de representación* o *Reglas STRIPS*

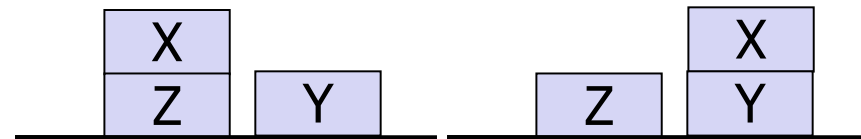
Ejemplo:

PutOn(X,Y,Z)

PC: $\text{clear}(Y) \wedge \text{clear}(X) \wedge \text{on}(X,Z)$

B: $\text{clear}(Y), \text{on}(X,Z)$

A: $\text{clear}(Z), \text{on}(X,Y)$



Planificación

Los *operadores STRIPS* se definen mediante *esquemas de representación* o *Reglas STRIPS*

Ejemplo:

PutOn(X,Y,Z)

nombre

variables libres

precondición: conjunción de literales.

PC: $\text{clear}(Y) \wedge \text{clear}(X) \wedge \text{on}(X,Z)$

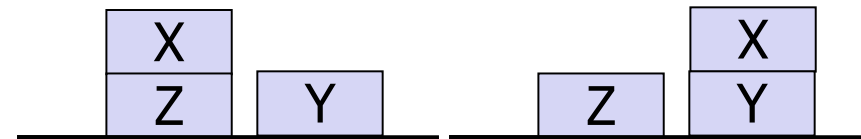
B: $\text{clear}(Y), \text{on}(X,Z)$

A: $\text{clear}(Z), \text{on}(X,Y)$

*otra forma de
representarlos*

$\sim \text{clear}(Y) \sim \text{on}(X,Z)$
 $\text{clear}(Z), \text{on}(X,Y)$

*efecto: Lista borradores + lista
adiciones.*



Planificación

Operador = instancia de una regla.

*Una instancia de una regla STRIPS se puede **aplicar** a la descripción de un estado s si existe una instancia base de PC (considerada como un objetivo) que es satisfecha por la descripción del estado.*

PutOn(X,Y,Z)

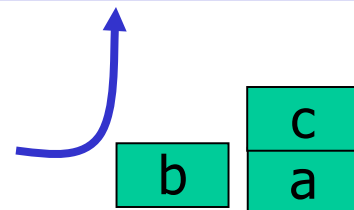
PC: $\text{clear}(Y) \wedge \text{clear}(X) \wedge \text{on}(X,Z)$

B: $\text{clear}(Y), \text{on}(X,Z)$

A: $\text{clear}(Z), \text{on}(X,Y)$

$\text{clear}(b) \wedge \text{clear}(c) \wedge$
 $\text{on}(c,a) \wedge \text{on}(b,\text{table}) \wedge$
 $\text{on}(a,\text{table}).$

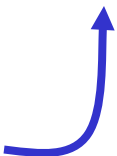
ESTADO



REGLA

$\theta = \{Y/b, X/c, Z/a\}$

UNIFICADOR



Planificación

La instancia se obtiene al aplicar θ a los conjuntos: PC , A y B .

PutOn(X,Y,Z)

PC: $\text{clear}(Y) \wedge \text{clear}(X) \wedge \text{on}(X,Z)$

B: $\text{clear}(Y), \text{on}(X,Z)$

A: $\text{clear}(Z), \text{on}(X,Y)$

$\theta = \{Y/b, X/c, Z/a\}$

PutOn(c, b, a)

PC: $\text{clear}(b) \wedge \text{clear}(c) \wedge \text{on}(c,a)$

B: $\text{clear}(b), \text{on}(c,a)$

A: $\text{clear}(a), \text{on}(c,b).$

Planificación

La instancia se obtiene al aplicar θ a los conjuntos: PC , A y B .

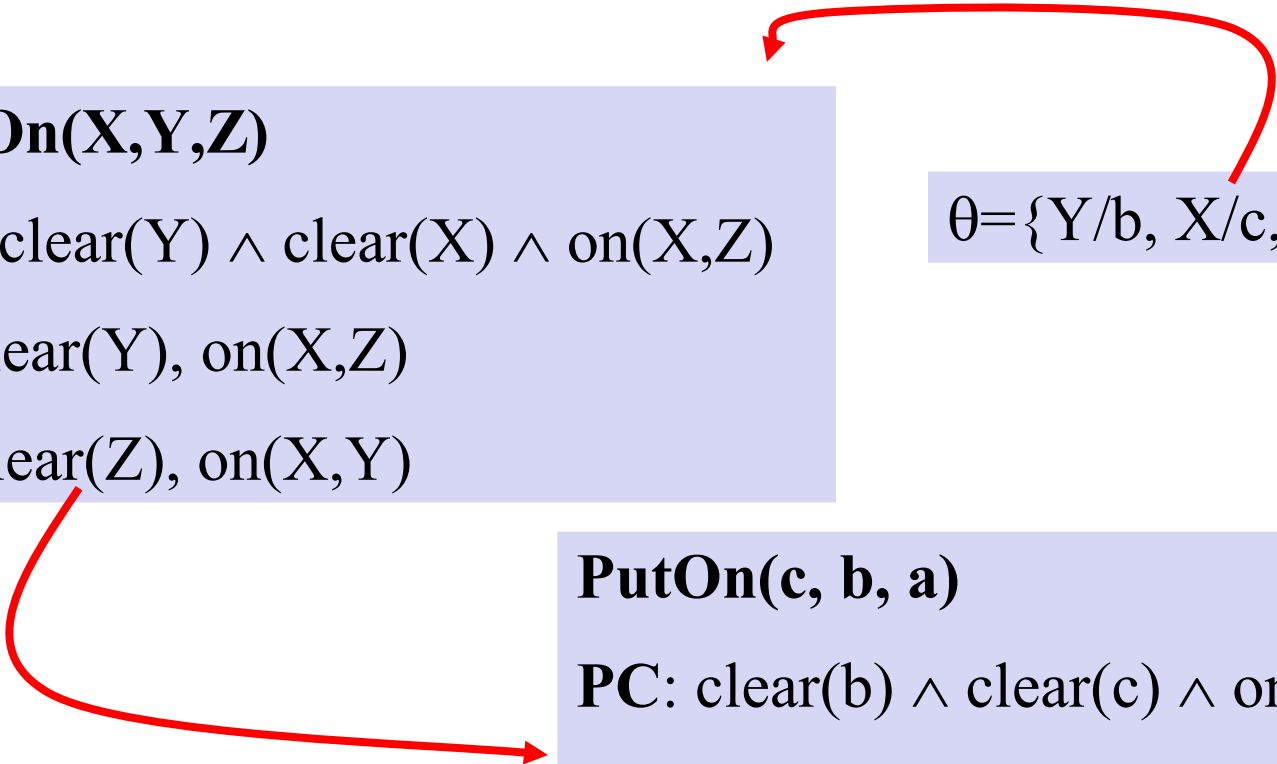
PutOn(X,Y,Z)

PC: $\text{clear}(Y) \wedge \text{clear}(X) \wedge \text{on}(X,Z)$

B: $\text{clear}(Y), \text{on}(X,Z)$

A: $\text{clear}(Z), \text{on}(X,Y)$

$\theta = \{Y/b, X/c, Z/a\}$



PutOn(c, b, a)

PC: $\text{clear}(b) \wedge \text{clear}(c) \wedge \text{on}(c,a)$

B: $\text{clear}(b), \text{on}(c,a)$

A: $\text{clear}(a), \text{on}(c,b).$

No pueden aparecer en PC, ni en las listas A y B variables libres que no sean argumentos de la regla.

Planificación

Planificador de **orden parcial (POP)**:

- ✓ El planificador trabaja sobre sub-metas en forma independiente una de otra.
- ✓ No se preocupa del orden de los pasos durante la búsqueda.
- ✓ Estrategia: *mínimo compromiso* demorar las decisiones durante la búsqueda.
- ✓ Pueden aparecer dos acciones en un plan sin identificar orden entre ellas.

Planificación

Definimos dos acciones: *inicio* y *fin*

Inicio

PC: true.

Estado inicial

B:

A: on(B,Table), on(A,Table), on(C,A) cl(A), cl(B).

Fin

PC: on(A,B), on(B,C), on(C,table), cl(A).

B:

A:

meta

Planificación

Representación de un plan:

Plan(pasos: {S1:op(acción:comenzar); S2: op(acción: ir_a(casa)).....})

orden {S1 < S2, comenzar < S1, ... }

asociaciones {X/casa; ...}

c

links { S1 \rightarrow S2;})

Asociaciones: relación variables a constantes

Links: relaciones causales

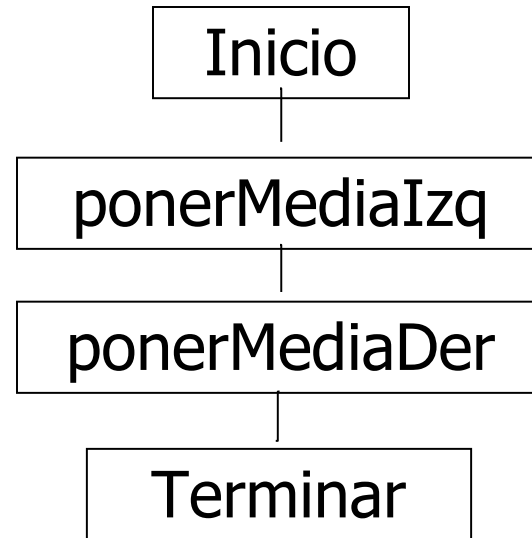
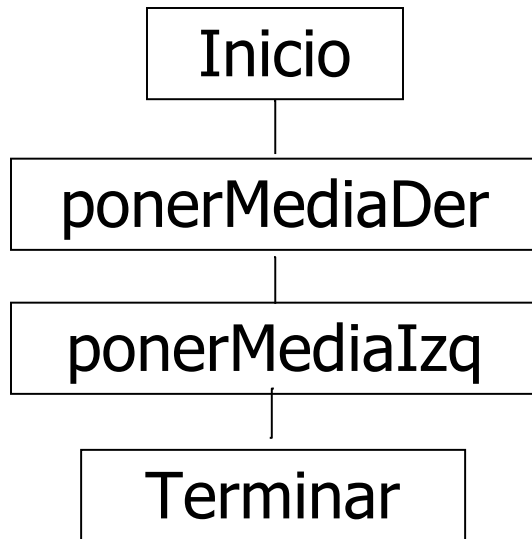
$S_i \xrightarrow{c} S_j$

La acción **S_i** produce c para **S_j** (c precondition de **S_j**)

Planificación

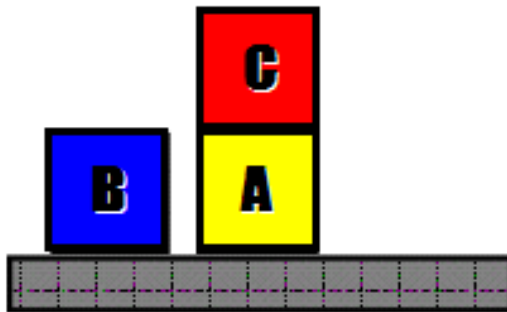
- La solución parcial representa dos planes de orden total cada uno de ellos es una *linealización* del plan de orden parcial.

Planes de orden total:

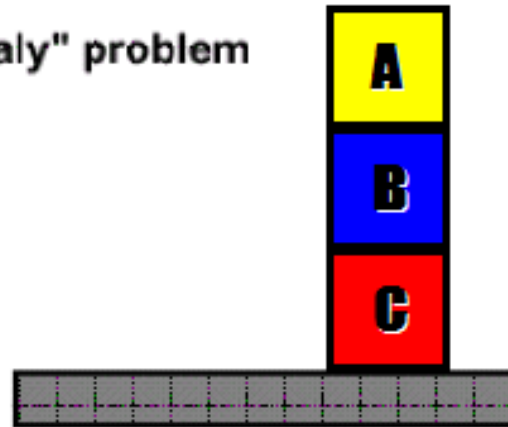


Planificación

"Sussman anomaly" problem



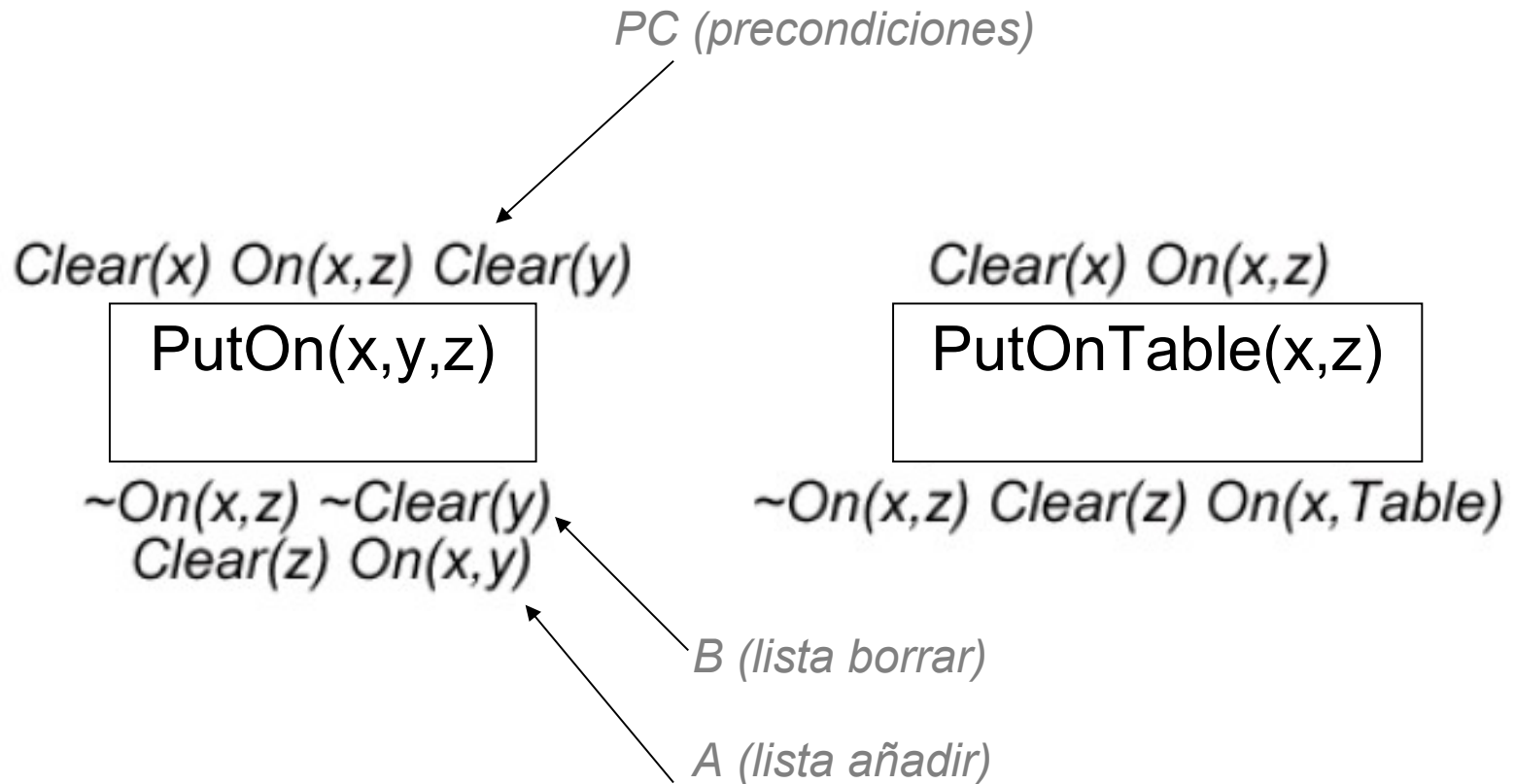
Estado inicial



Estado final

Planificación

Reglas u operadores “STRIPS”

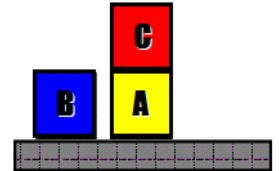


Planificación

S_0

START

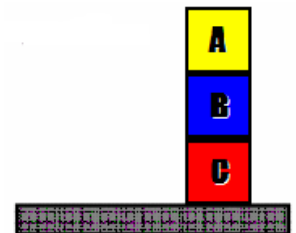
On(C,A) On(A,Table) Cl(B) On(B,Table) Cl(C)



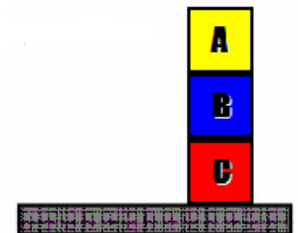
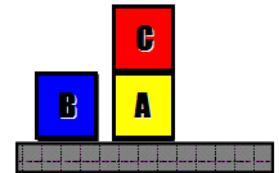
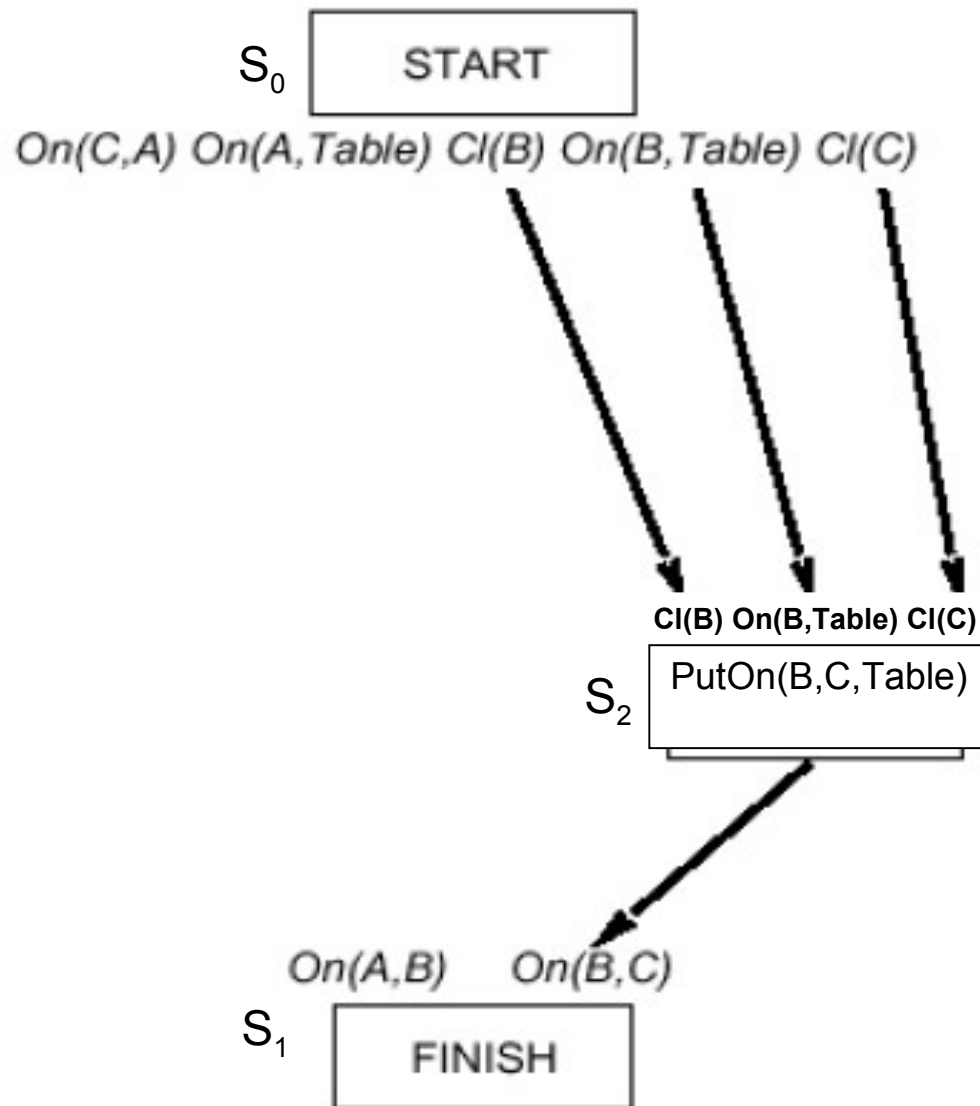
S_1

On(A,B) On(B,C)

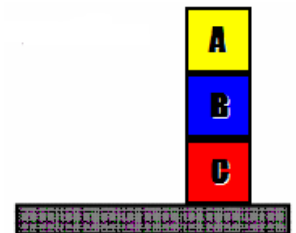
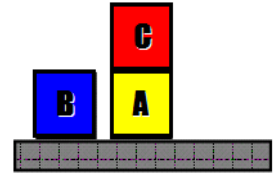
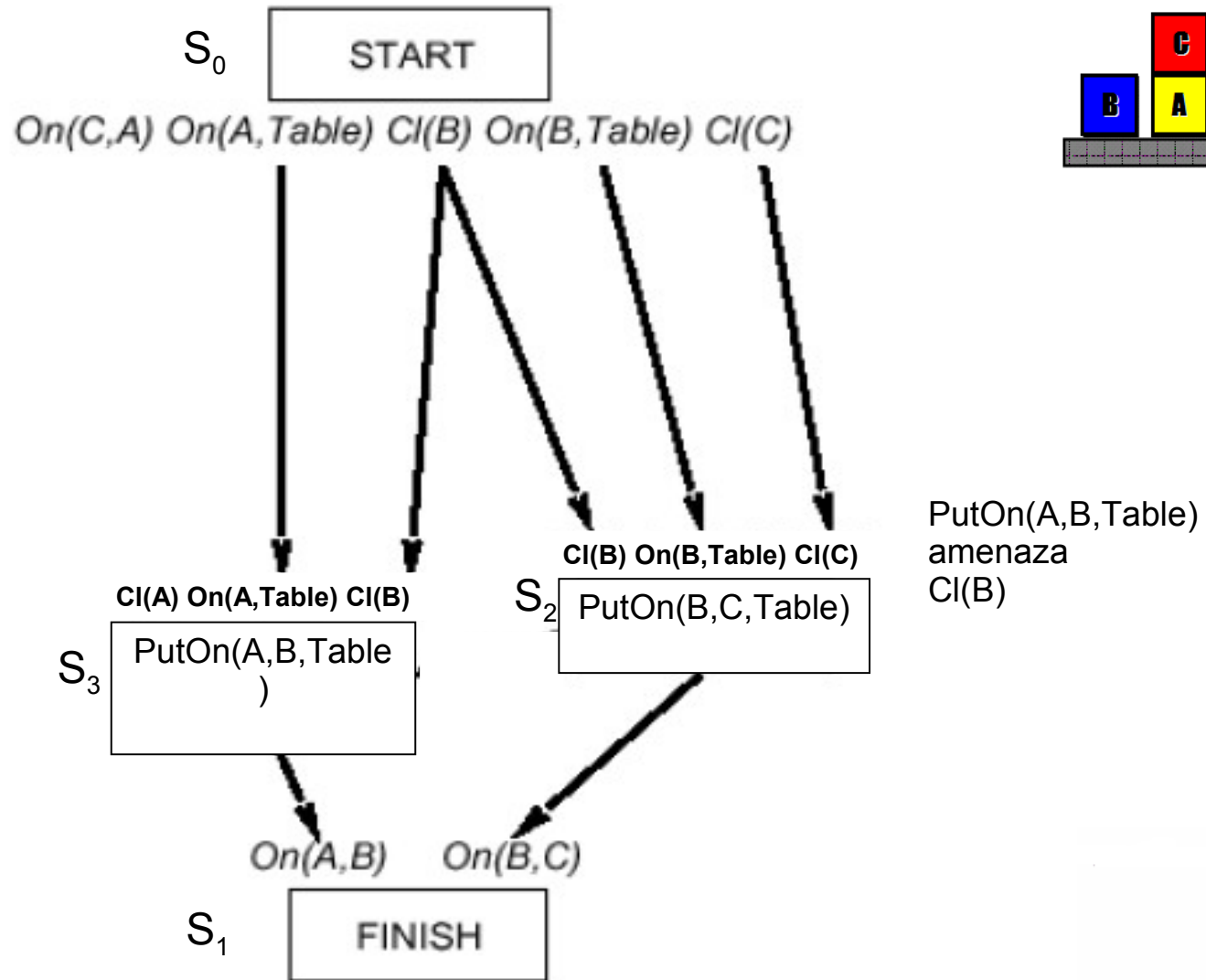
FINISH



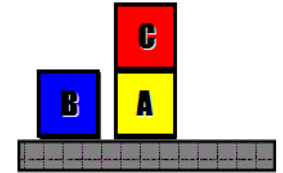
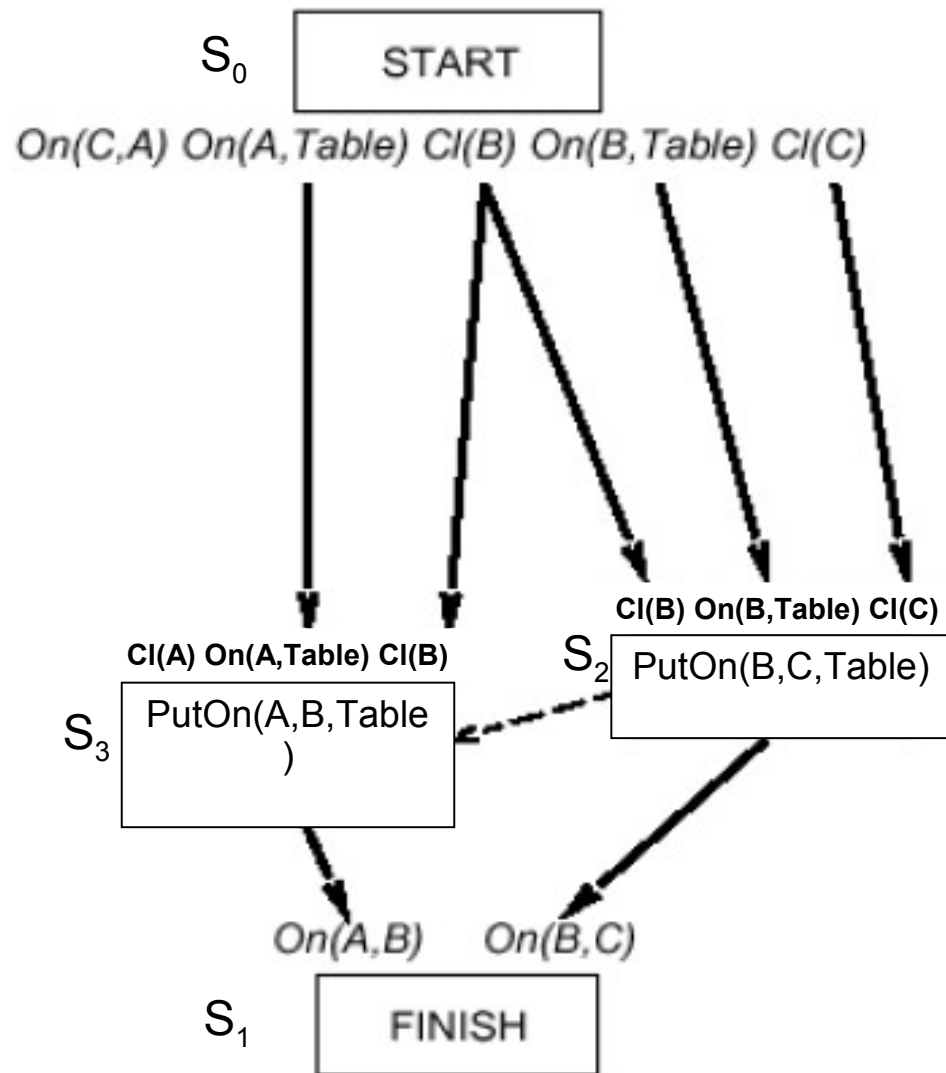
Planificación



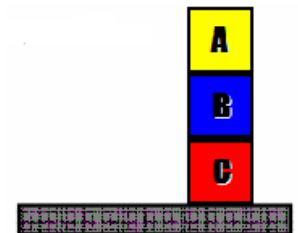
Planificación



Planificación

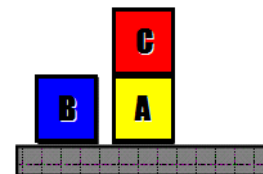
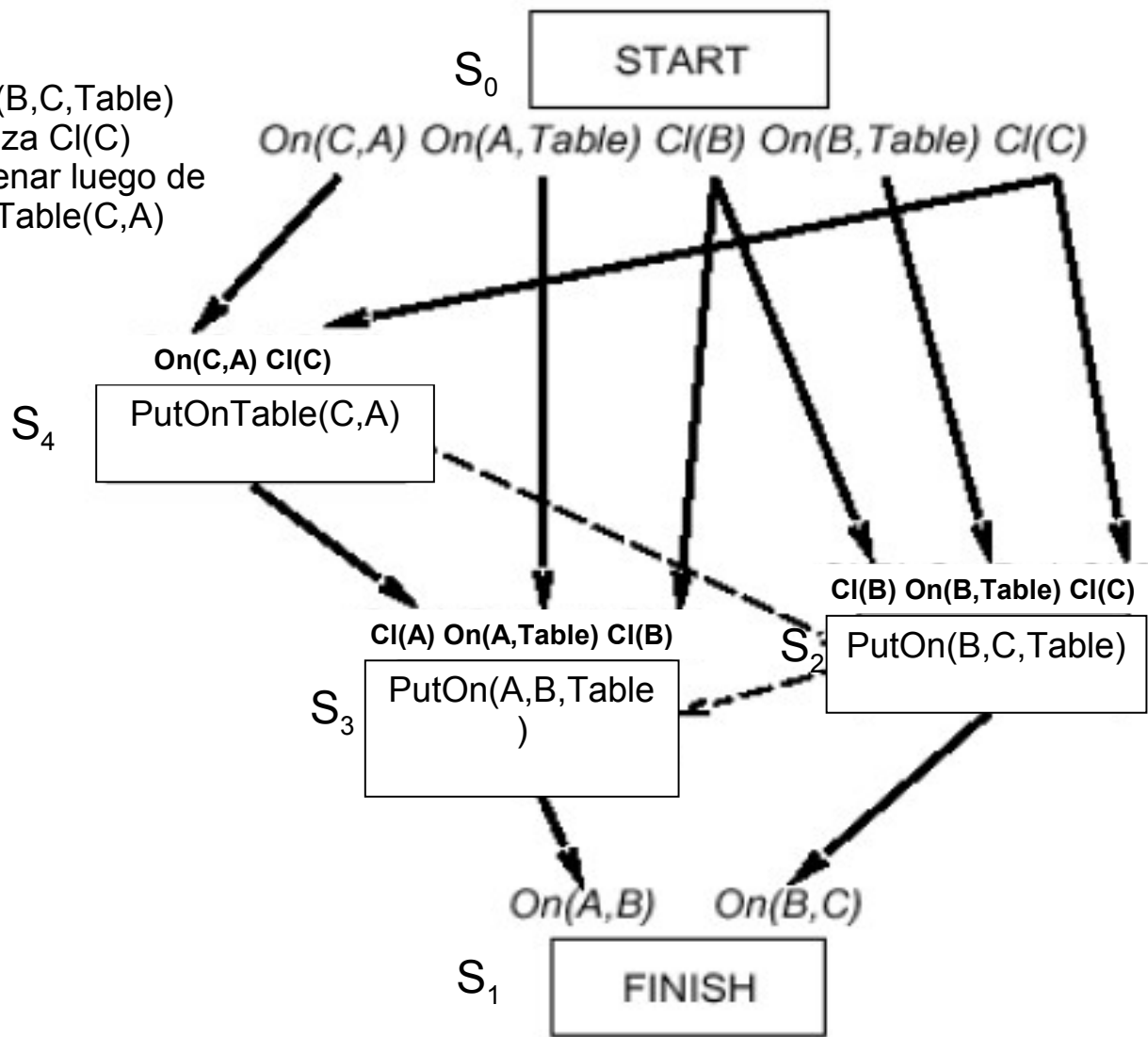


$PutOn(A,B,Table)$
amenaza $Cl(B)$
→ ordenar luego de
 $PutOn(B,C,Table)$
con un link de ordenamiento

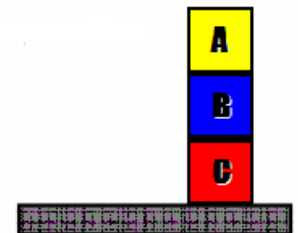
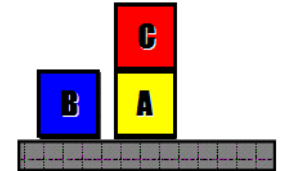
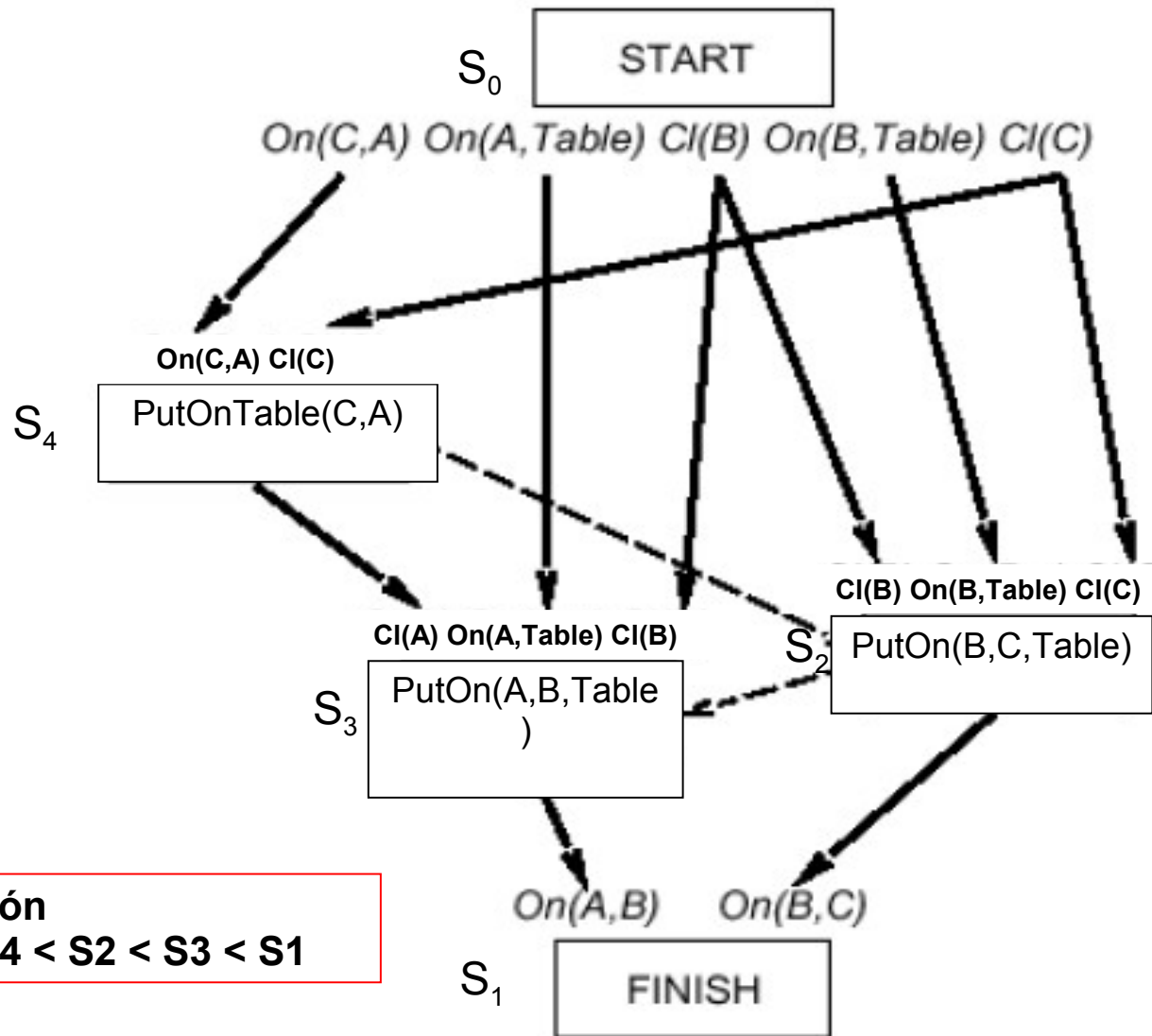


Planificación

PutOn(B,C,Table)
amenaza Cl(C)
→ Ordenar luego de
PutOnTable(C,A)



Planificación



Solución

$S_0 < S_4 < S_2 < S_3 < S_1$