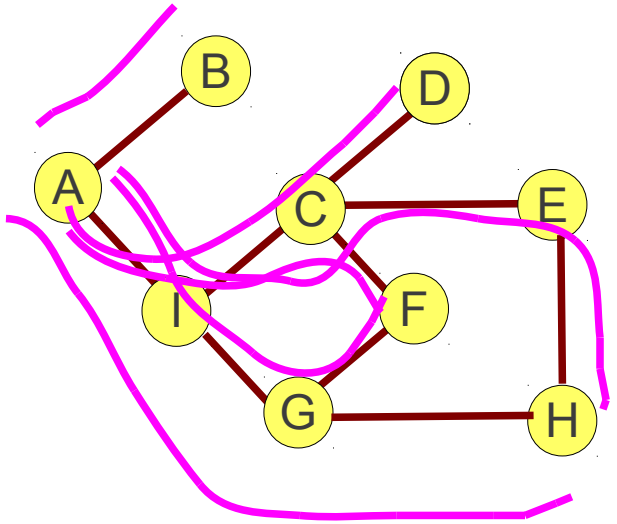
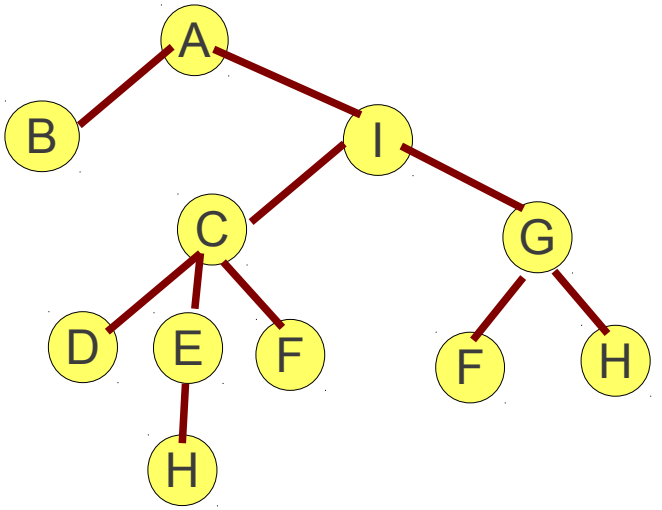
Inteligencia Computacional

Búsqueda: sin información - otros algoritmos, con información

http://blancavg.com/tc3023/

Blanca A. Vargas Govea * blanca.vg@gmail.com * Agosto 17, 2012





A lo ancho

{A}
{B,I}
{I}
{C,G}
{G,D,E,F}
{D,E,F,F,H}
{E,F,F,H}
{F,F,H,H}
{H}

En profundidad

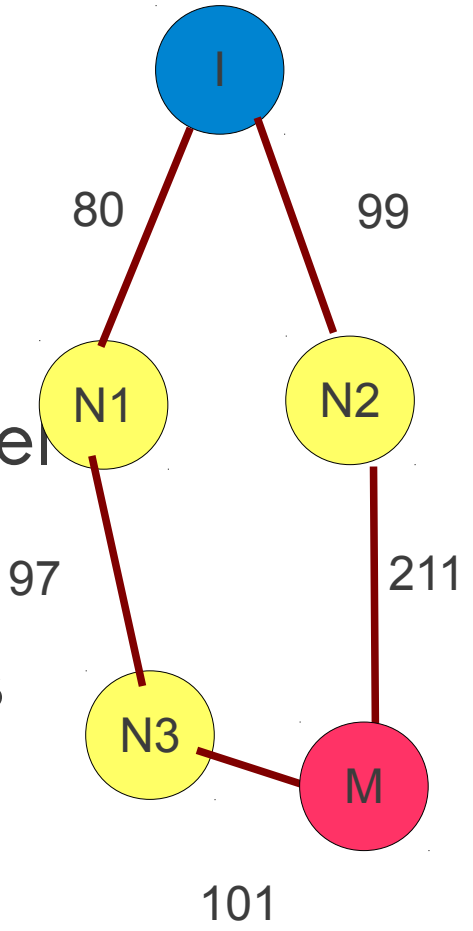
{A}
{B,I}
{I}
{C,G}
{D,E,F,G}
{H,F,G}
{F,G}
{G}
{F,H}
{H}

Solución ejercicio

Otros algoritmos de búsqueda

En vez de expandir el nodo más superficial, se expande el nodo n con el menor costo del camino g(n).

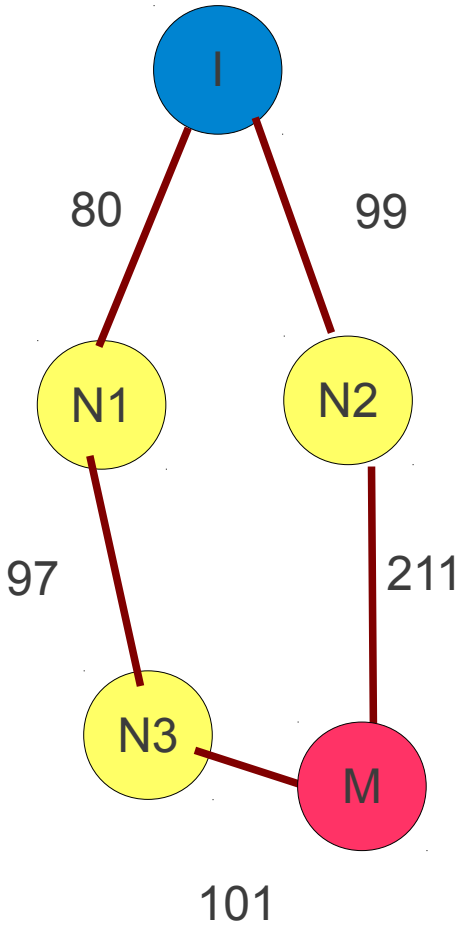
Se ordenan los nodos expandidos.



{I}
{N1(80),N2(99)} – sort -
{N2(99),N3(177)}
{N3(177),M(310)}
{M(310),M(278)}

Costo uniforme

La prueba de la meta se hace cuando el nodo se selecciona para expansión y no cuando es generado (puede estar en un camino sub-óptimo).

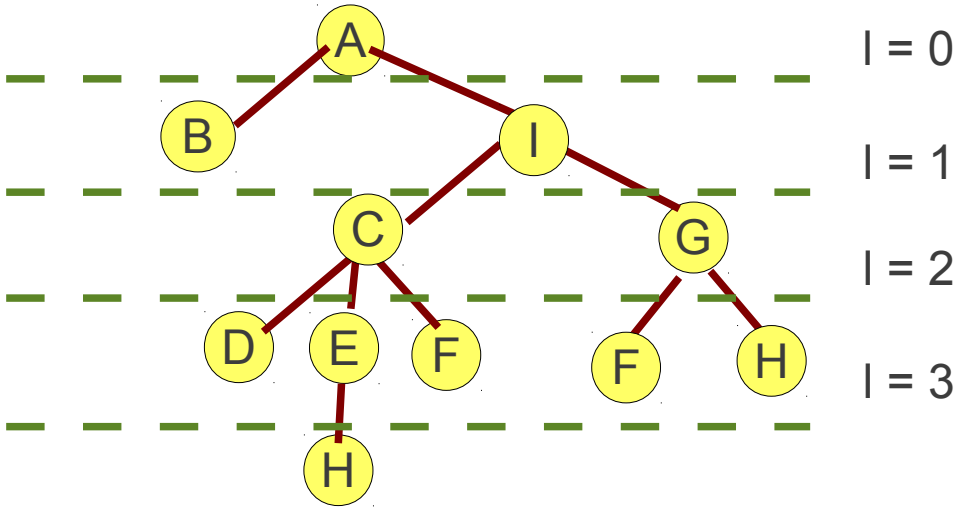


{I}
{N1(80),N2(99)} – sort -
{N2(99),N3(177)}
{N3(177),M(310)}
{M(310),M(278)}

No toma en cuenta número de pasos, solamente el costo total.

Costo uniforme - Diferencia con b. anchura

Para reducir la búsqueda en espacios de estados grandes en búsqueda en profundidad

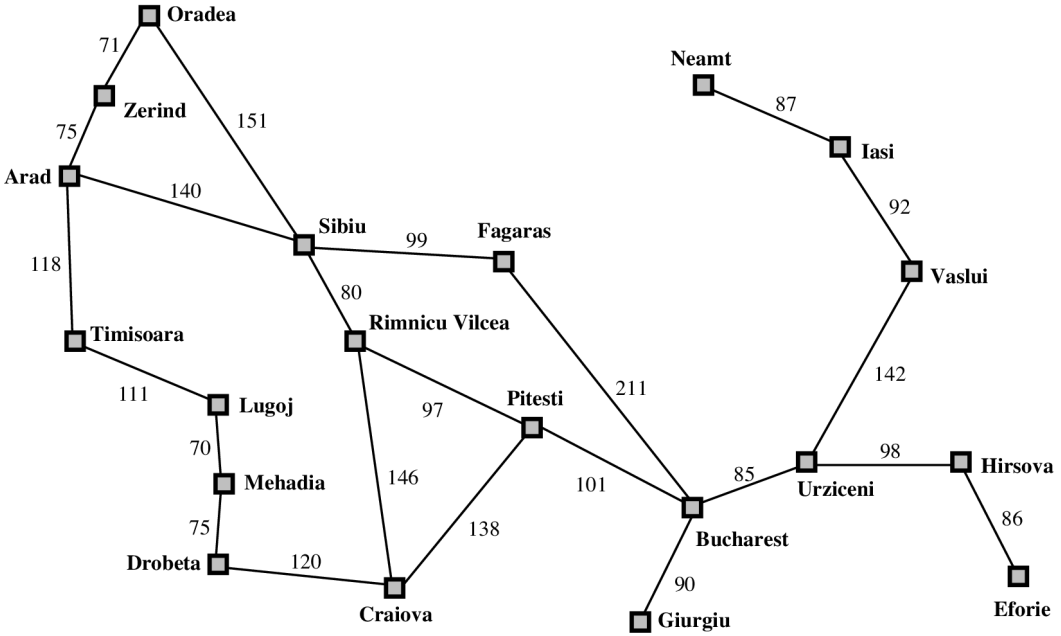


Límite de profundidad l

Desventaja: si l < profundidad (d) de la meta

Depth limited

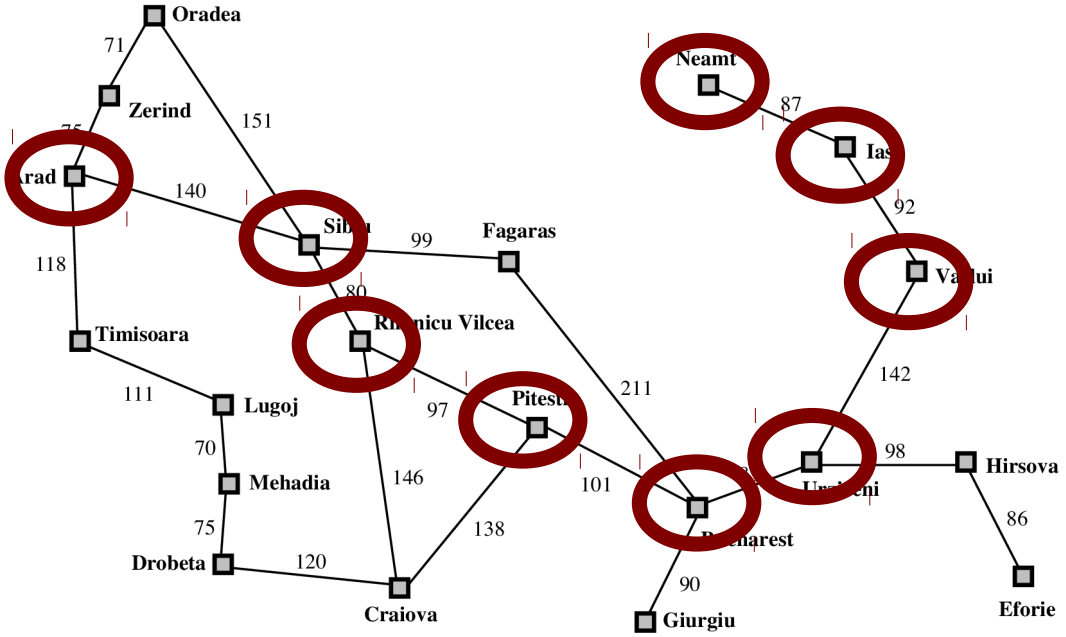
Se puede usar conocimiento sobre el problema



Ejemplo: 20 ciudades, entonces cuando mucho la solución (si existe,) estará a 19 ciudades.

Depth limited

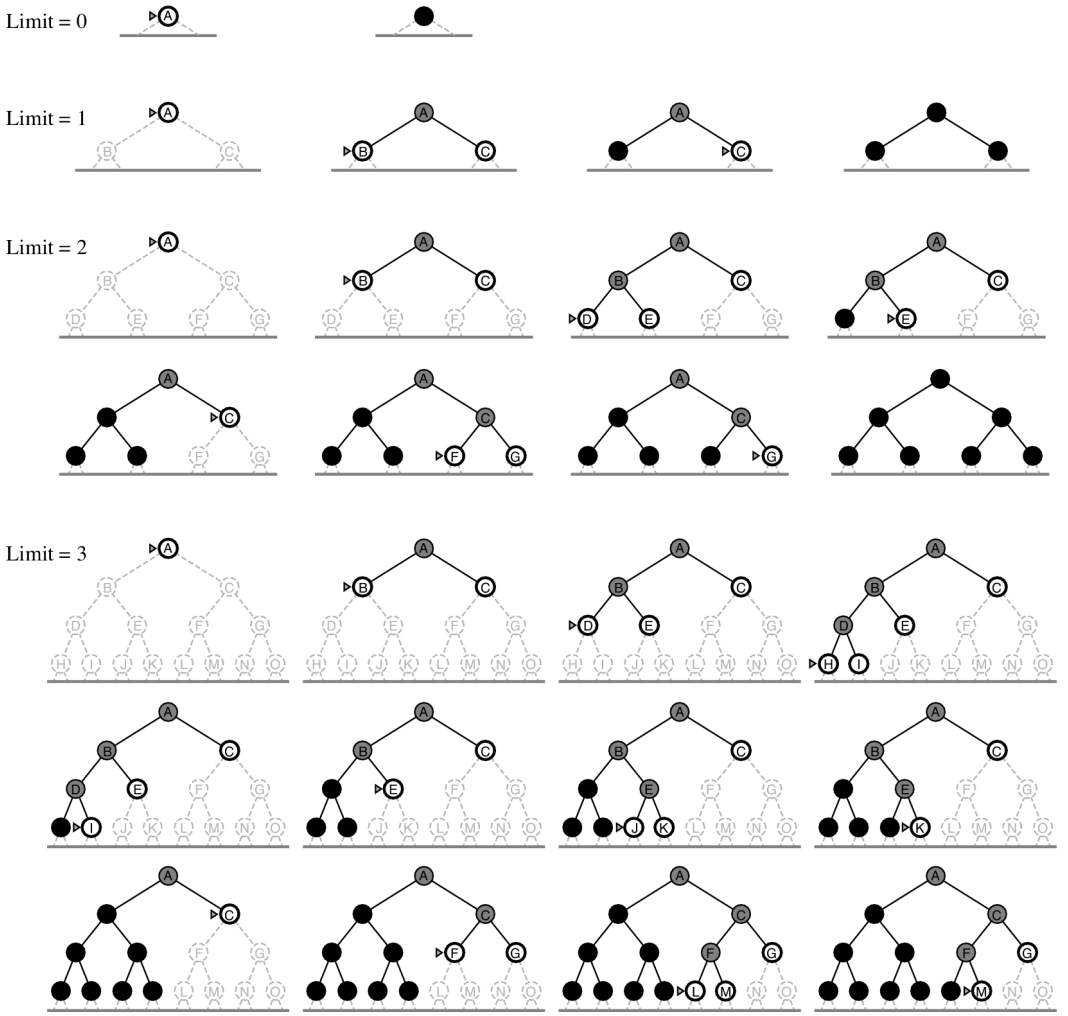
De cualquier ciudad se puede llegar a otra cuando mucho en 9 pasos → diámetro del espacio de estados



La mayoría de las veces no es posible conocer un valor para l

Depth limited - diámetro

Se usa en combinación con depth-first (en profundidad) y encuentra el mejor límite



Incrementa gradualmente el límite hasta encontrar una meta

Iterative deepening

Combina los beneficios de búsqueda a lo ancho y en profundidad

a lo ancho

Es completo si el factor de arborescencia (no. de hijos de cada nodo) es finito

Es óptimo si el costo del camino es una función no decreciente

en profundidad

Menos recursos de memoria

Iterative deepening- beneficios

¿Desperdicio?

Para un árbol con el mismo o parecido factor de arborescencia en cada nivel, la mayoría de los nodos están en el nivel más profundo (d)

Los nodos en d se generan una vez

El método es el preferido cuando el espacio de estados es grande y no se conoce la profundidad

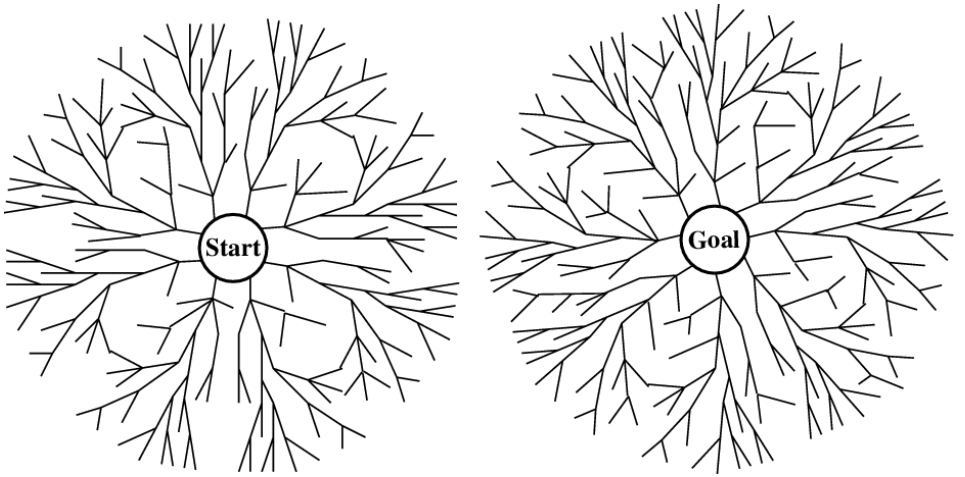
Iterative deepening- ¿desperdicio?

Correr dos búsquedas simultáneas:

1) desde el estado inicial

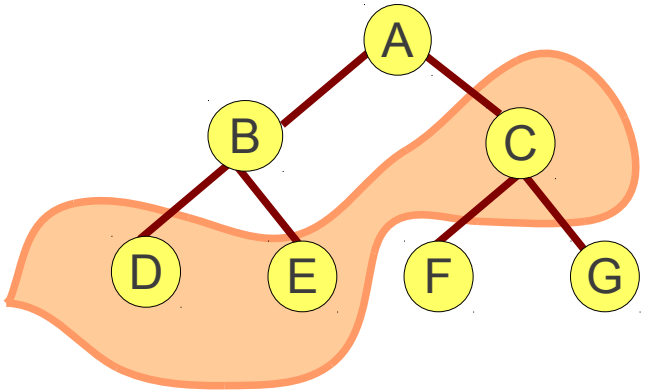
2) desde la meta

esperando que se encuentren



Bidireccional

En vez de probar si el estado es meta, se verifica si existe intersección en la frontera de las búsquedas



La frontera es el conjunto de nodos que no han sido expandidos (e.g., los del queue)

Bidireccional

Ventajas

Velocidad, memoria

Desventajas

Se necesita conocer el estado meta

No siempre es posible la búsqueda hacia atrás

La implementación requiere otras consideraciones

Ventajas/Desventajas

Búsqueda con información

Utiliza información más allá de la definición del problema.

Encuentra soluciones más eficientes que la estrategia sin información

Características

Un nodo se selecciona para expansión con base en una función de evaluación f(n)

La función se construye como un estimado del costo del nodo hacia la meta

Enfoque general: best first

h(n) = costo estimado del mejor camino del nodo n al estado meta

Diferente de las distancias en el mapa

Ejemplo: calcular el camino menos costoso de Arad a Bucharest mediante la distancia de las líneas rectas

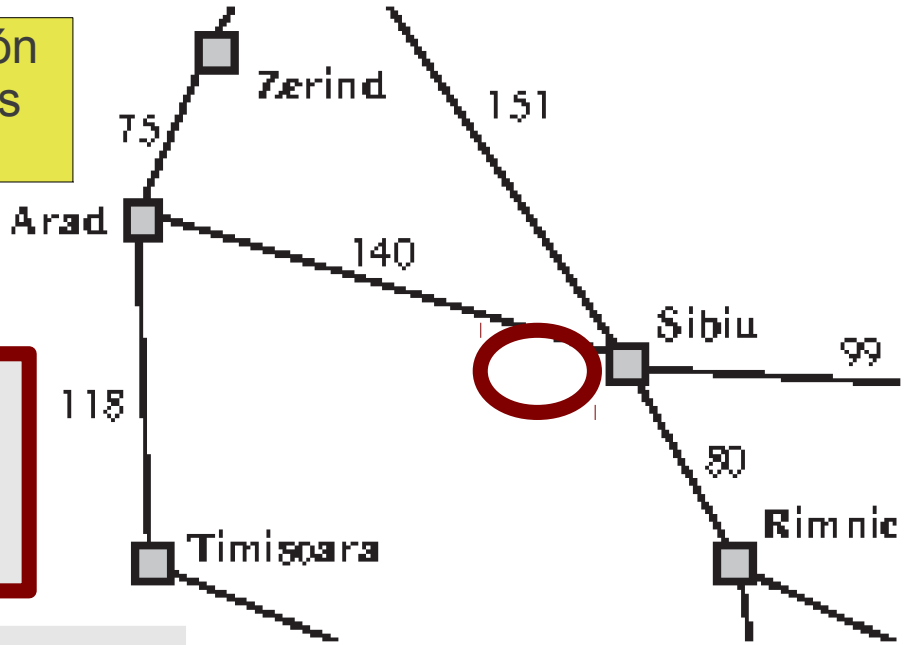
si n es un nodo meta, h(n) = 0

Heurística general

Trata de expandir el nodo que está más cerca de la meta suponiendo que llevará a la meta rápidamente.

Arad	366	Mehadia	241
Bucharest	0	Neamt	234
Craiova	160	Oradea	380
Drobeta	242	Pitesti	100
Eforie	161	Rimnicu Vilcea	193
Fagaras	176	Sibiu	253
Giurgiu	77	Timisoara	329
Hirsova	151	Urziceni	80
Iasi	226	Vaslui	199
Lugoj	244	Zerind	374

La solución 32km más larga



Arad-Sibiu-Fagaras-Bucarest

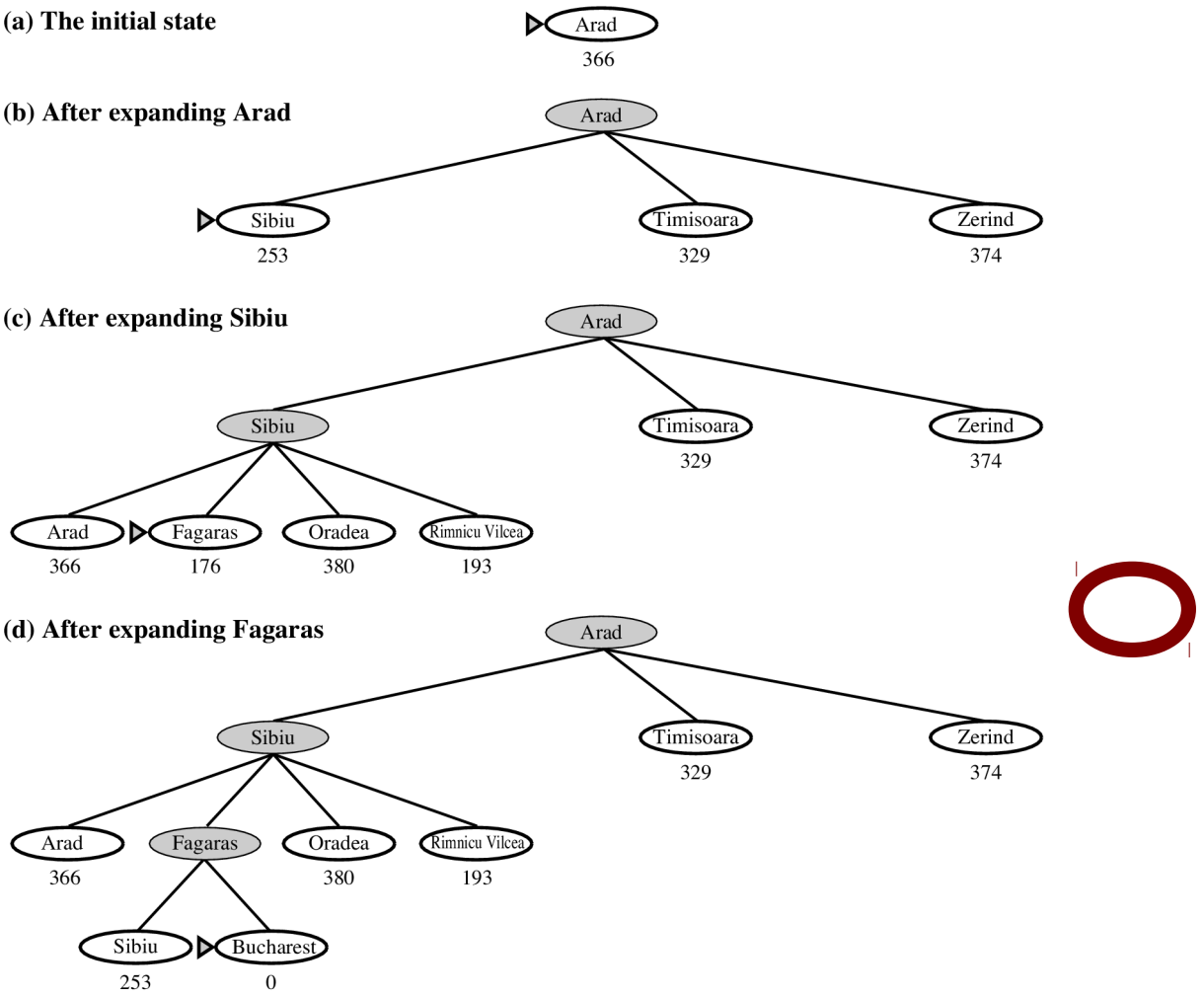
140+199+211 = 450 km

Arad-Sibiu-Rimnicu-Pitesti-Bucarest

140+80+97+101 = 418 km

Greedy best-first

Glotón, voraz, codicioso



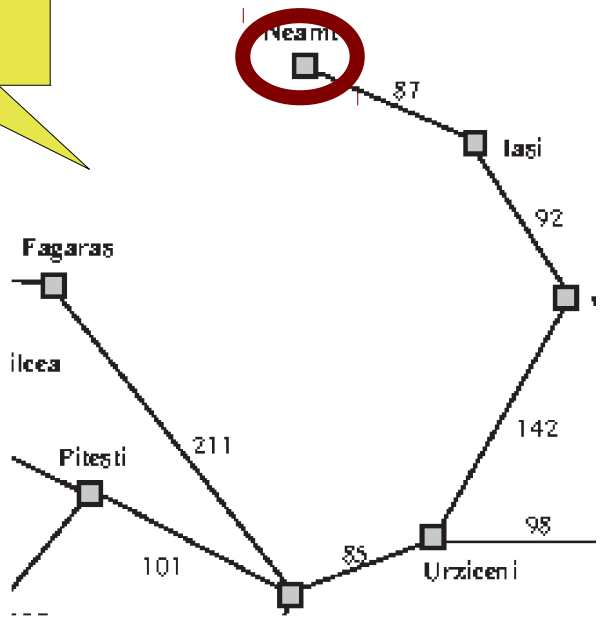
Greedy best-first - mapa

Glotón, voraz, codicioso

Si se quiere ir de Iasi a Fagaras, se expandiría Neamt porque es más cercano a Fagaras

Arad	366	**Mehadia**	241
Bucharest	0	**Neamt**	234
Craiova	160	**Oradea**	380
Drobeta	242	**Pitesti**	100
Eforie	161	**Rimnicu Vilcea**	193
Fagaras	176	**Sibiu**	253
Giurgiu	77	**Timisoara**	329
Hirsova	151	**Urziceni**	80
Iasi	226	**Vaslui**	199
Lugoj	244	**Zerind**	374

Punto muerto



A partir de Neamt, como Iasi es el más cercano a Fagaras, entraría en un ciclo infinito

Greedy best-first - Puntos muertos - ciclos

Glotón, voraz, codicioso

Evalúa los nodos combinando g(n) el costo de alcanzar el nodo y h(n), el costo del nodo de llegar a la meta

$f(n) = g(n) + h(n)$

g(n) = costo estimado de la solución menos costosa pasando por n

algoritmo igual que el de costo uniforme pero en vez de g(n) es g(n) + h(n)

A^* (la más conocida de los best first)

Si la heurística h(n) satisface ciertas condiciones, A* es completa y óptima

Admisibilidad

h(n) no sobreestima el costo de llegar a la meta

La heurística de la distancia de la línea recta es admisible porque siempre será la más corta

A* condiciones de h(n)

Si la heurística h(n) satisface ciertas condiciones, A* es completa y óptima

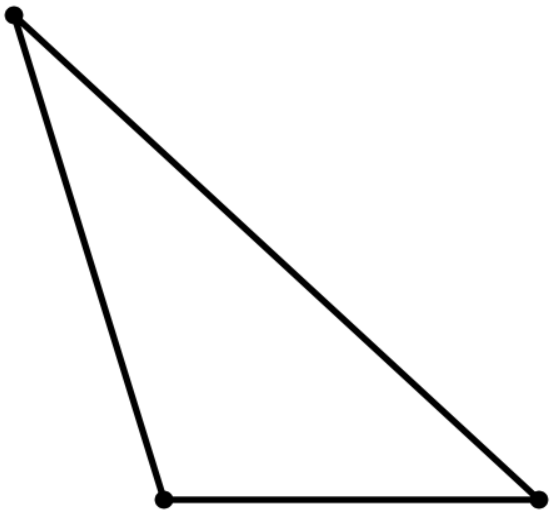
Consistencia

h(n) es consistente si para cada nodo *n* y cada sucesor *n'* generado por una acción *a* el costo estimado de alcanzar la meta desde *n* no es mayor que el costo de llegar al sucesor *n'* más el costo estimado de llegar a la meta desde *n'*

$$h(n) \leq c(n,a,n') + h(n')$$

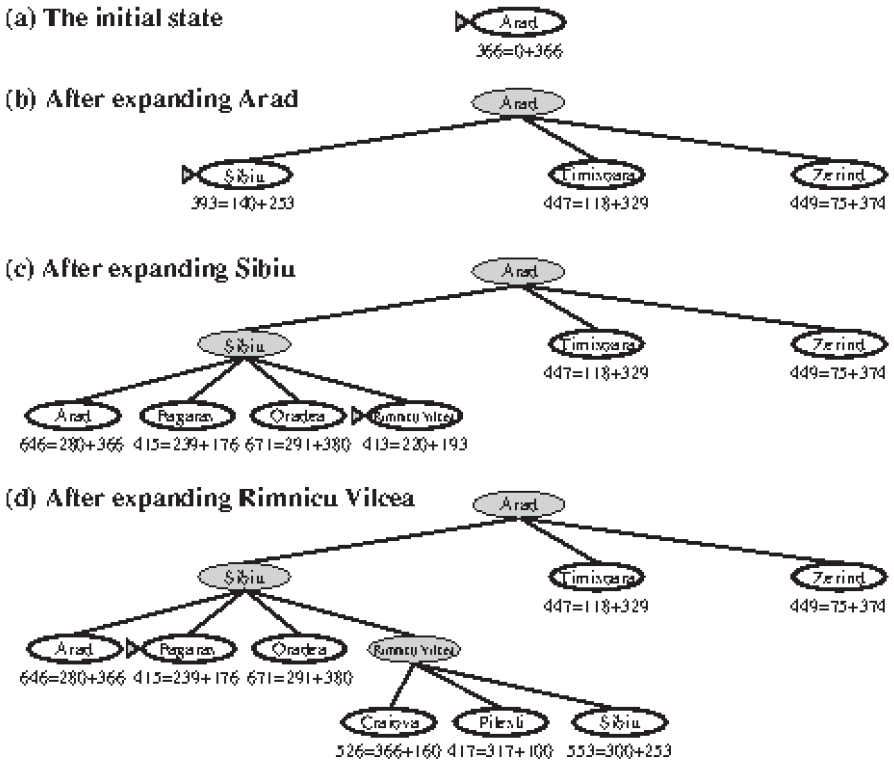
A* condiciones de h(n)

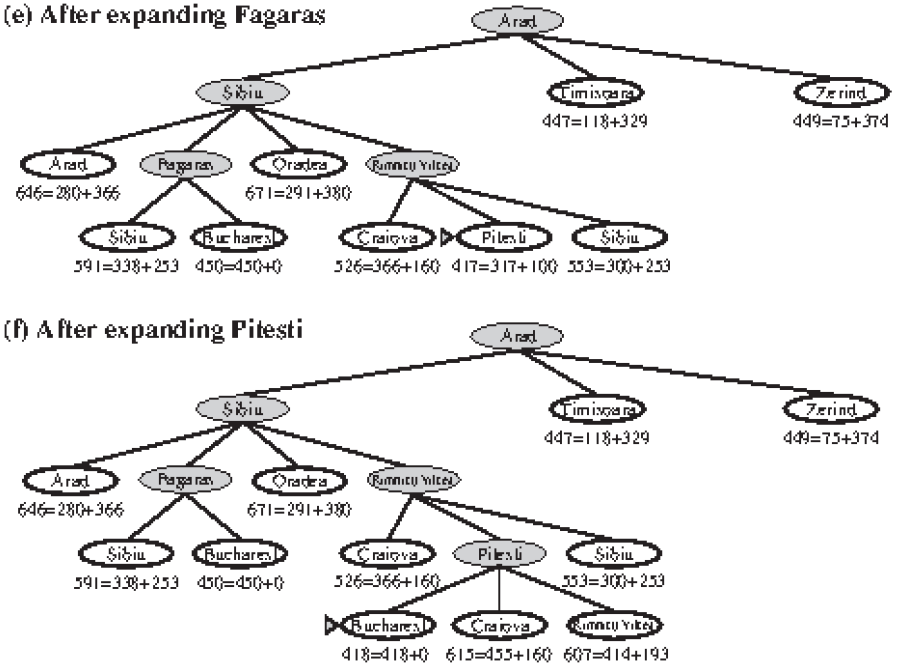
Si la heurística h(n) satisface ciertas condiciones, A* es completa y óptima



Cada lado de un triángulo no puede ser mayor que la suma de los dos lados

A* condiciones de h(n)


(a) The initial state
Arad
366=0+366
(b) After expanding Arad
Arad
Sibiu
Timisoara
Zerind
393=140+253
447=118+329
449=75+374
(c) After expanding Sibiu
Arad
Sibiu
Timisoara
Zerind
447=118+329
449=75+374
Arad
Fagaras
Oradea
Rimnicu Vilcea
646=280+366
415=239+176
671=291+380
413=220+193
(d) After expanding Rimnicu Vilcea
Arad
Sibiu
Timisoara
Zerind
447=118+329
449=75+374
Arad
Fagaras
Oradea
Rimnicu Vilcea
646=280+366
415=239+176
671=291+380
Craiova
Pitesti
Sibiu
526=366+160
417=317+100
553=300+253


(e) After expanding Fagaras
Arad
Sibiu
Timisoara
Zerind
447=118+329
449=75+374
Arad
Fagaras
Oradea
Rimnicu Vilcea
646=280+366
671=291+380
Sibiu
Bucharest
Craiova
Pitesti
Sibiu
591=338+253
450=450+0
526=366+160
417=317+100
553=300+253
(f) After expanding Pitesti
Arad
Sibiu
Timisoara
Zerind
447=118+329
449=75+374
Arad
Fagaras
Oradea
Rimnicu Vilcea
646=280+366
671=291+380
Sibiu
Bucharest
Craiova
Pitesti
Sibiu
591=338+253
450=450+0
526=366+160
553=300+253
Bucharest
Craiova
Rimnicu Vilcea
418=418+0
615=455+160
607=414+193

A* árbol

Ventajas

Es completa y óptima si la heurística es admisible

Desventajas

La complejidad espacial es grande

A*

Búsqueda local

Si el camino no importa, solamente la solución, entonces se utilizan algoritmos de búsqueda local

Búsqueda local

Referencias

Russell, S., y Norvig, P. (2003). Artificial intelligence: A modern approach (2nd edition ed.). Prentice-Hall, Englewood Cliffs, NJ.

Notas de cursos: Eduardo Morales, L. Enrique Sucar