

Busca Cega e Busca Informada

Profa. Mercedes Gonzales
Márquez

Tópicos

- Busca cega ou busca não informada
 - Busca em largura
 - Busca em profundidade
 - Busca em profundidade limitada
 - Busca com aprofundamento iterativo
 - Busca direcional
- Busca informada
 - Busca gulosa
 - Busca A*

Estratégias de Busca Sem Informação (ou Busca Cega)

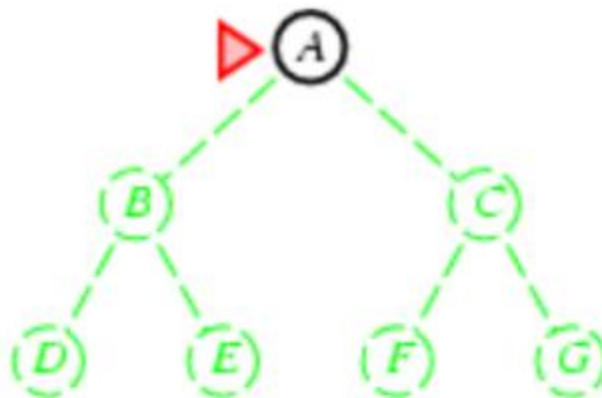
- Estratégias de busca sem informação não recebe qualquer indicação sobre a proximidade que um estado se encontra de sua meta. Veja o exemplo de Zarad no mapa.
- As estratégias de busca sem informação se distinguem pela ordem em que os nós são expandidos.

Estratégias de Busca Sem Informação (ou Busca Cega)

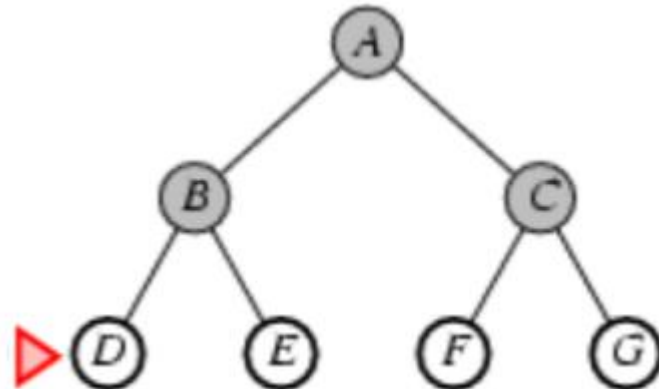
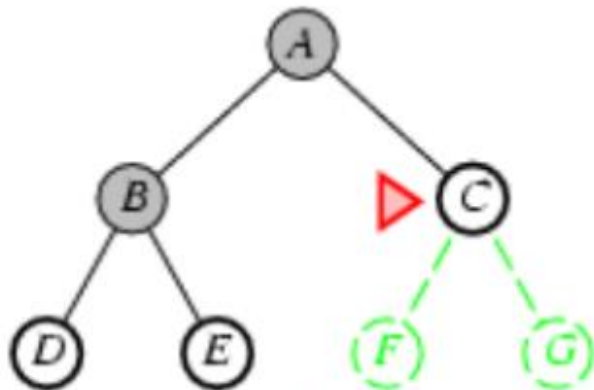
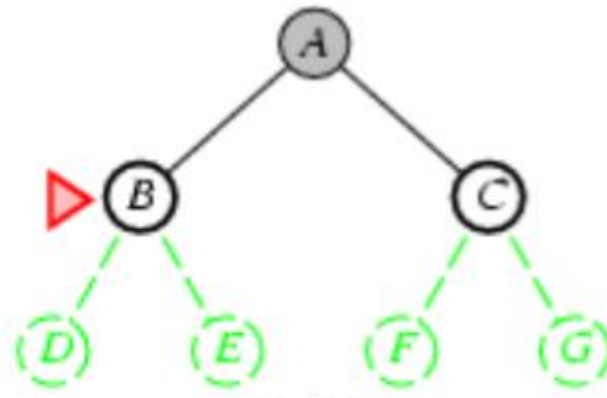
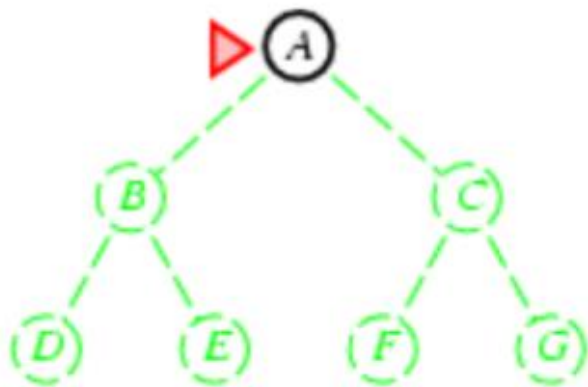
- Busca em largura
- Busca em profundidade
- Busca em profundidade limitada
- Busca de aprofundamento iterativo.
- Busca de custo uniforme

Busca em largura

- O nó raiz é expandido primeiro e, em seguida, todos os sucessores dele, depois todos os sucessores desses nós.
- Todos os nós em uma dada profundidade são expandidos antes de todos os nós do nível seguinte.



Busca em largura



Busca em largura

- Uma fila (FIFO) é usada : novos nós (que são sempre mais profundos do que seus pais) vão para o fim da fila, enquanto os antigos, que são mais rasos que os novos, são expandidos primeiro.
- A busca em largura sempre acha uma solução com um número mínimo de ações porque quando está gerando nós na profundidade d , ela já gerou todos os nós na profundidade $d-1$.

Busca em largura

- Propriedades da busca em largura
- Completa? Sim (se b é finito)
- Tempo? $1+b+b^2+b^3+\dots +b^d = O(b^d)$
- Espaço? $O(b^d)$ (mantém todos os nós na memória)
- Ótima? Sim (se todas as ações tiverem o mesmo custo).

Outro exemplo

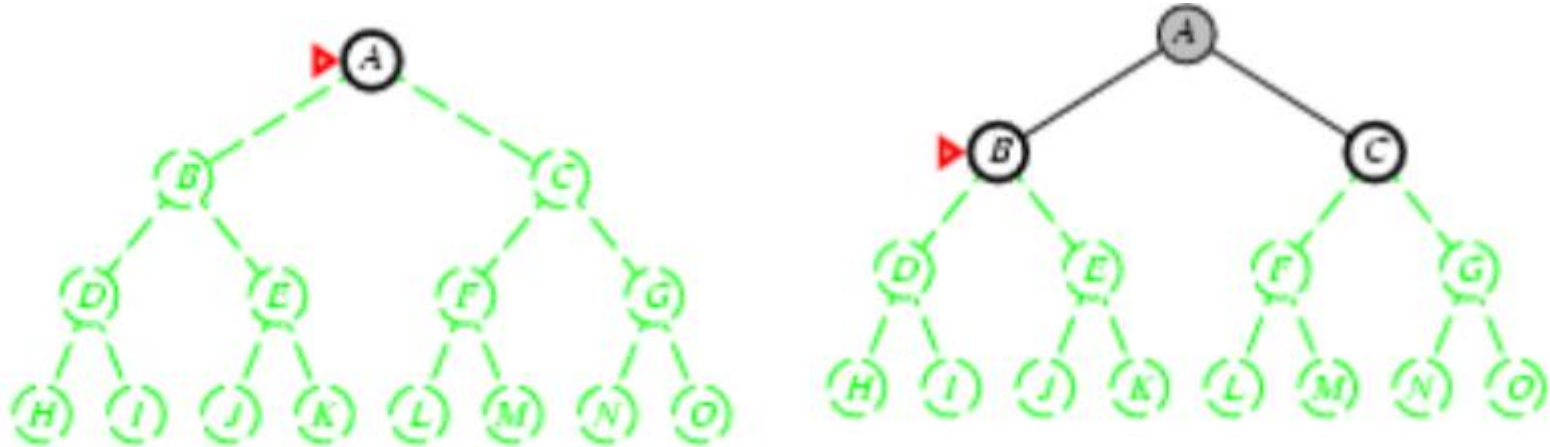


Busca em profundidade

- Expande o nó não-expandido mais profundo.
- A busca prossegue até o nível mais profundo da árvore, onde os nós não tem sucessores.
- Então a busca devolve ao nó seguinte mais profundo acima que ainda tem sucessores inexplorados.
- Uma fila (LIFO) é usada : novos nós (que são sempre mais profundos do que seus pais) são expandidos primeiro, enquanto os antigos, que são mais rasos que os novos, vão para o final da fila.

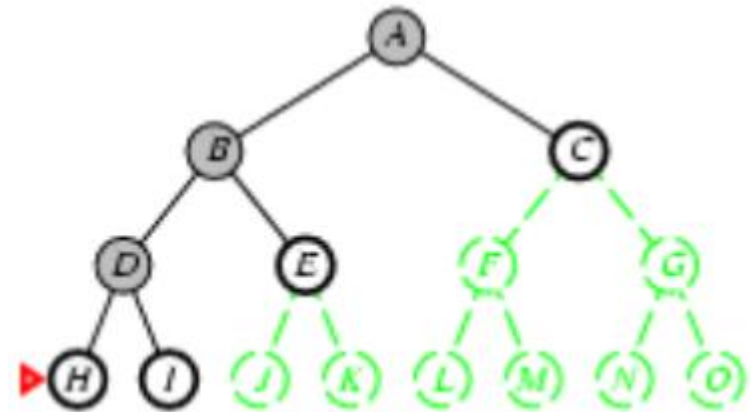
Busca em profundidade

- Expande o nó não-expandido mais profundo.



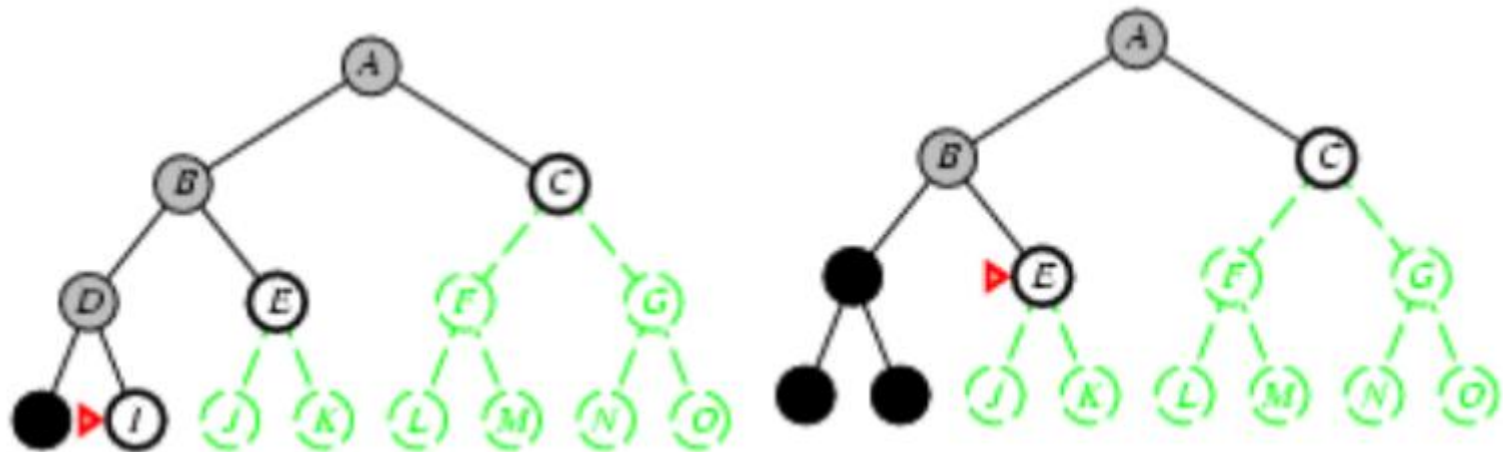
Busca em profundidade

- Expande o nó não-expandido mais profundo.



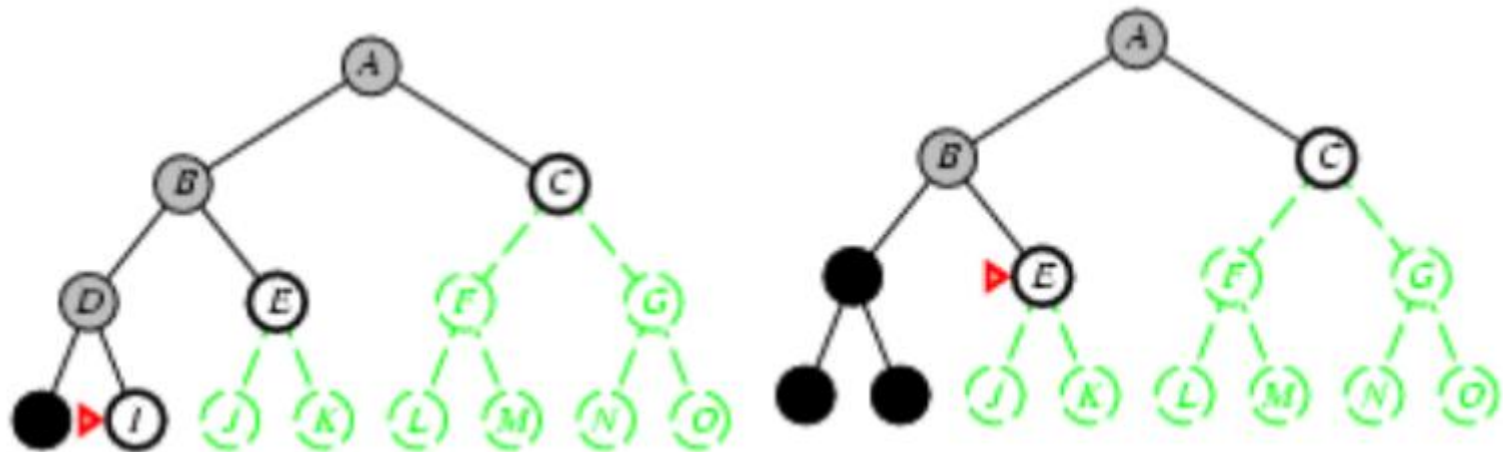
Busca em profundidade

- Expande o nó não-expandido mais profundo.



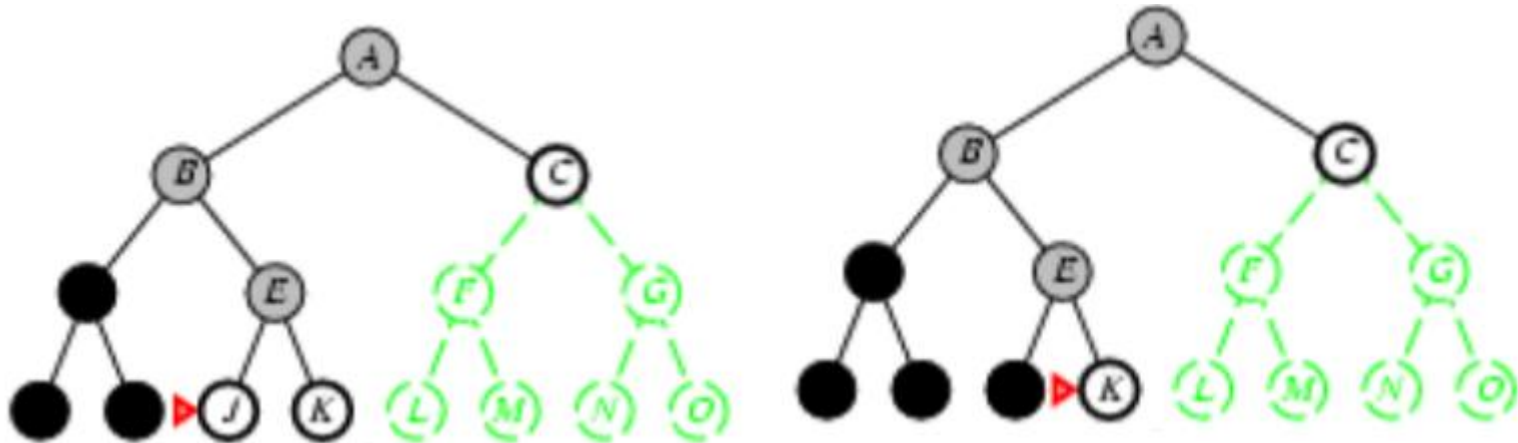
Busca em profundidade

- Expande o nó não-expandido mais profundo.



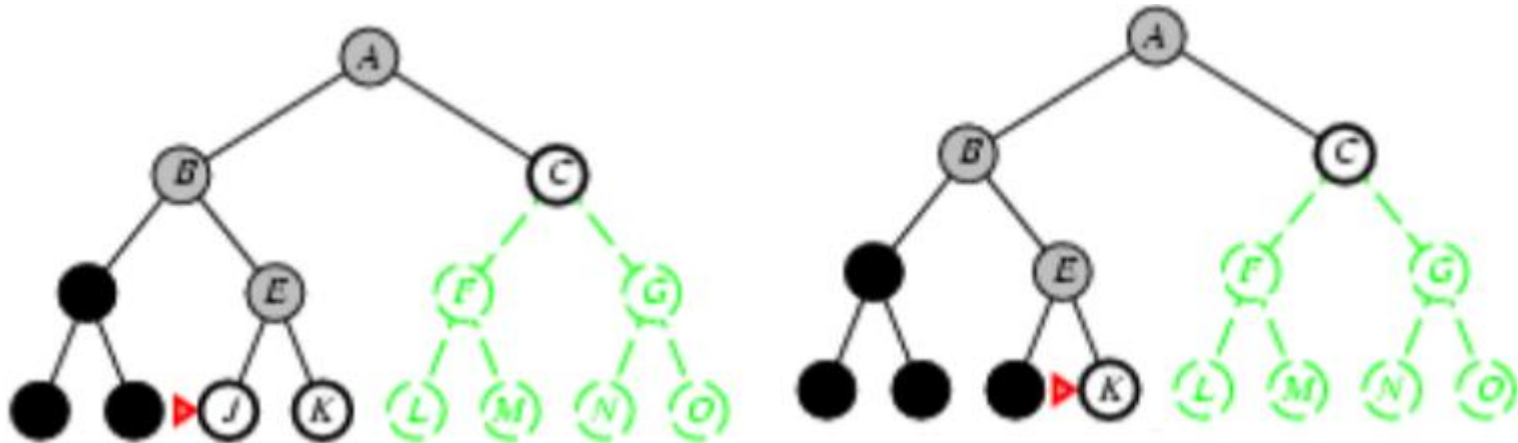
Busca em profundidade

- Expande o nó não-expandido mais profundo.



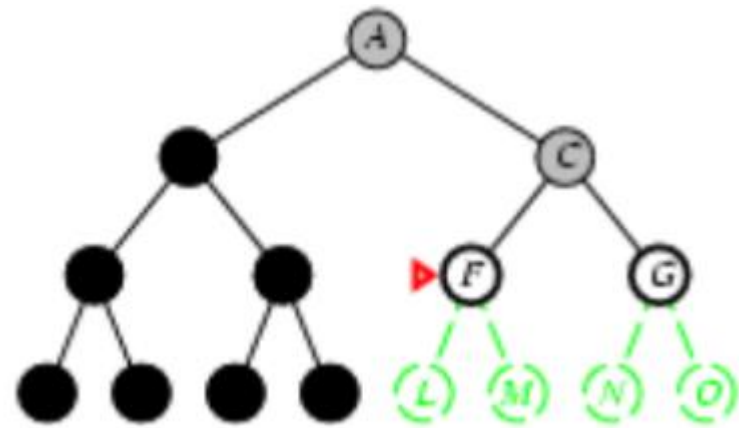
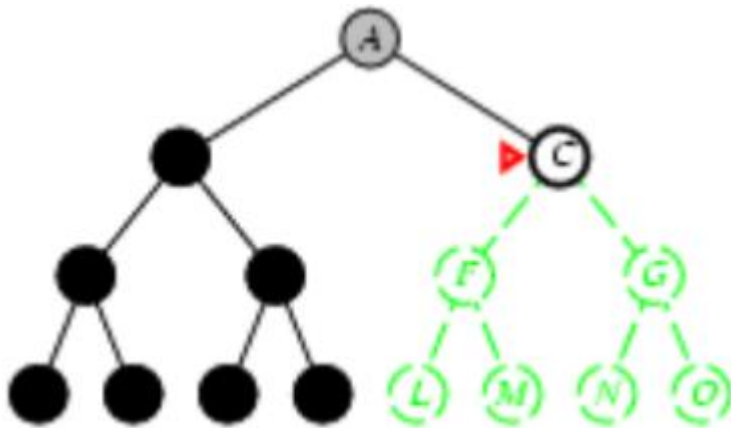
Busca em profundidade

- Expande o nó não-expandido mais profundo.



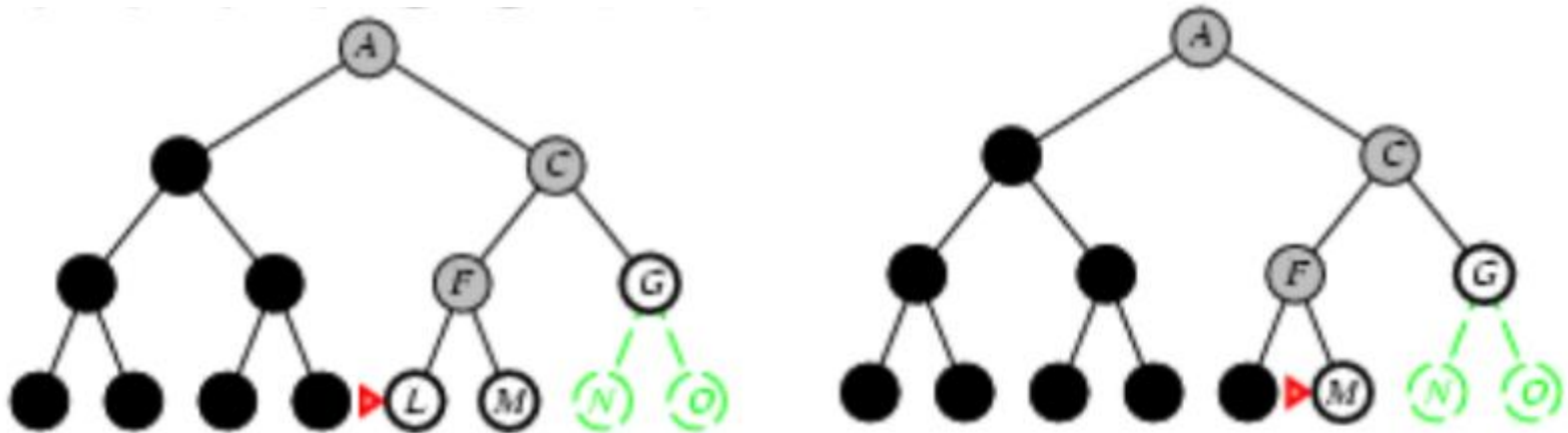
Busca em profundidade

- Expande o nó não-expandido mais profundo.



Busca em profundidade

- Expande o nó não-expandido mais profundo.



Busca em profundidade

- Só precisa armazenar um único caminho da raiz até um nó folha, e os nós irmãos não expandidos;
- Nós cujos descendentes já foram completamente explorados podem ser retirados da memória;
- Faz muito menos uso de memória que a busca em largura. A fronteira da busca em largura pode ser vista como a superfície de uma esfera sempre em expansão, enquanto que a fronteira da busca em profundidade é apenas um raio dessa esfera.

Busca em profundidade

- Logo para ramificação b e profundidade máxima m , a complexidade espacial é: $O(bm)$
- Propriedades da busca em profundidade.
- Completa? Sim em espaços com profundidade finita. Não completa em espaços com loops.
- Tempo? $O(b^m)$
- Espaço? $O(bm)$
- Ótima? Não.

Busca em profundidade limitada

- Para resolver o problema de busca em profundidade em árvores infinitas, um limite L restringe a busca, ou seja, nós na profundidade L são tratados como se não tivessem sucessores.
- Resolve caminhos infinitos, porém adiciona mais incompletude;.

Busca em profundidade limitada

- Completa? Não; a solução pode estar além do limite.
- Tempo? $O(b^l)$
- Espaço? $O(b^l)$
- Ótima? Não

Busca em Aprofundamento Iterativo

- Resolve o problema da escolha de um bom valor de l testando todos os valores: primeiro 0, depois 1, depois 2, e assim por diante.
- Combina muitos dos benefícios da busca em profundidade e da busca em largura.

Busca em Aprofundamento Iterativo

Limit = 0

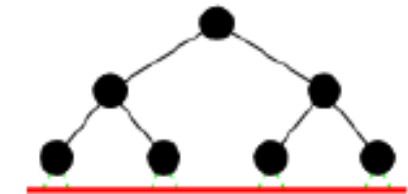
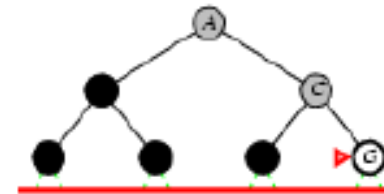
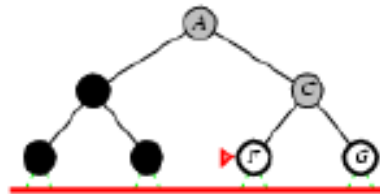
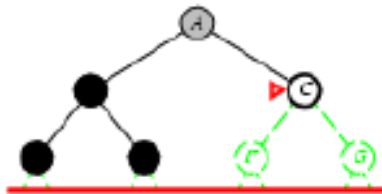
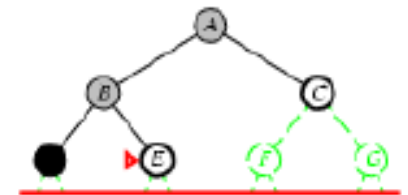
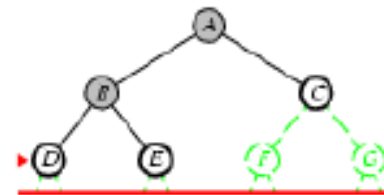
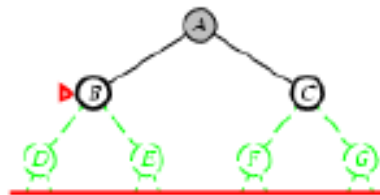
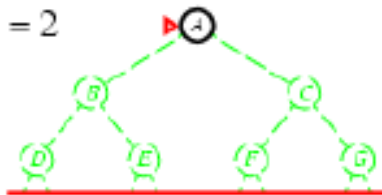


Limit = 1



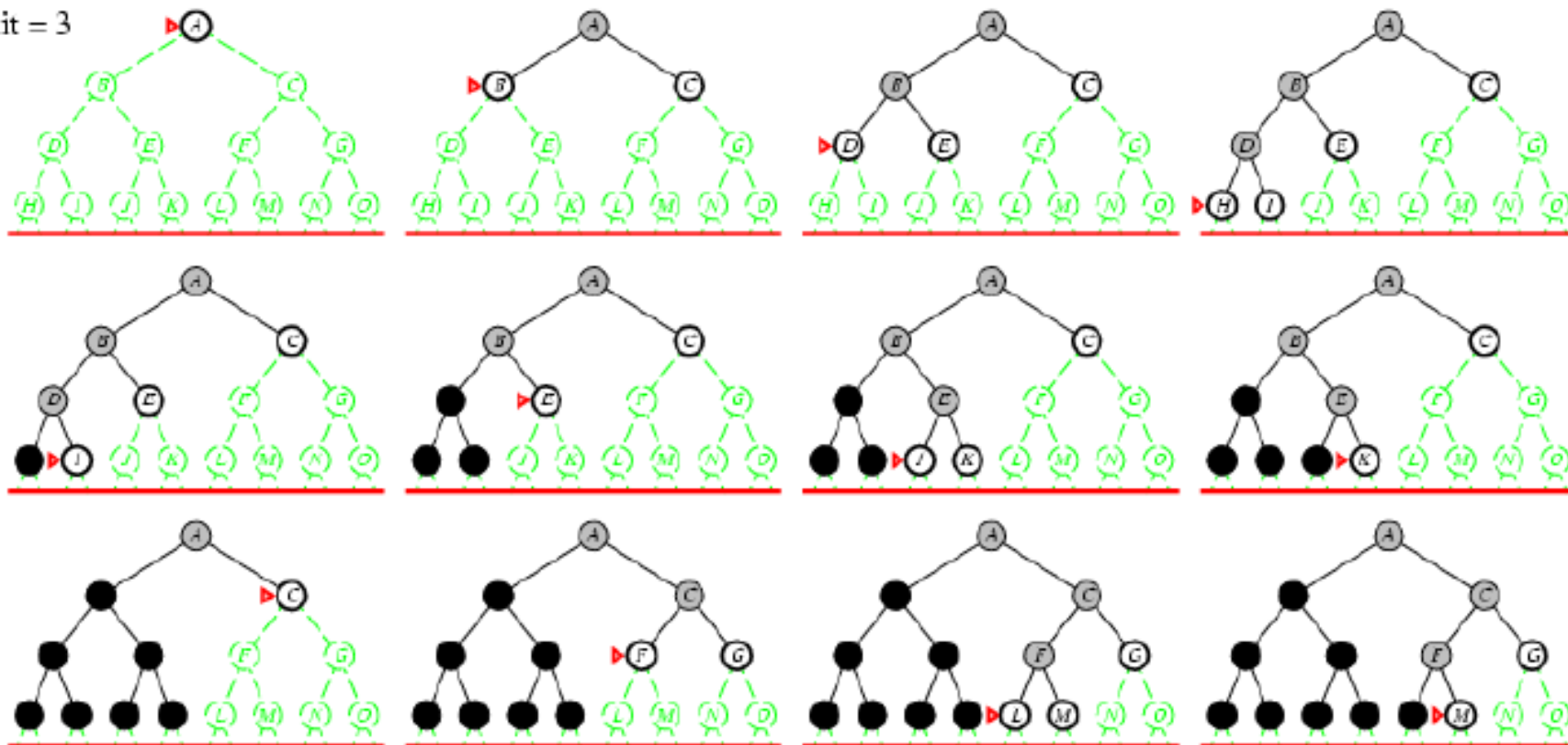
Busca em Aprofundamento Iterativo

Limit = 2



Busca em Aprofundamento Iterativo

Limit = 3



Busca em Aprofundamento Iterativo

- Ótima?
- Completa?
- Tempo?
- Espaço?

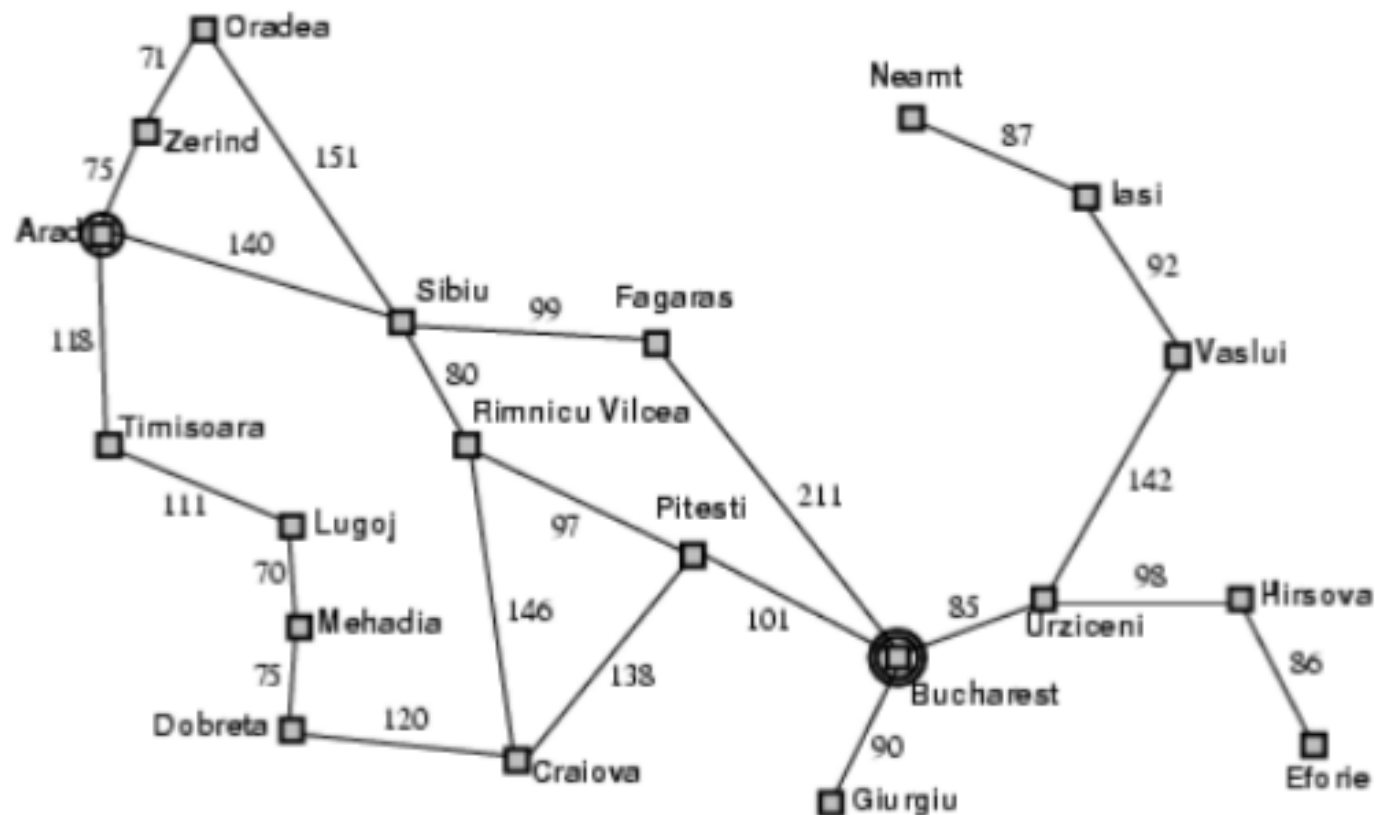
Busca direcional

- Realiza duas buscas simultâneas – uma direta, a partir do estado inicial, e a outra inversa, a partir do estado meta, esperando que as duas buscas se encontrem em um ponto intermediário.

Busca de custo uniforme

- Variante de Busca em Extensão
- Expande o nó n com custo de caminho $g(n)$ mais baixo.
- Os nós da borda são armazenados em uma fila de prioridades (Heap).
- O teste de objetivo é aplicado quando o nó é selecionado para a expansão.
- Aplicar busca de custo uniforme para achar o caminho mais curto entre Sibiu e Bucareste.

Busca de custo uniforme



Busca de custo uniforme

- Ótima?
- Completa?
- Tempo?
- Espaço?

Outro exemplo



Busca Informada ou heurística

- Usa conhecimento sobre o domínio para encontrar soluções mais eficientes do que busca cega.
- Define uma ***função de avaliação*** para cada nó da árvore (mede o quanto o nó é desejável!).
- Expande o nó que tem a função de avaliação mais baixa.
- A implementação utiliza uma fila de nós ordenada segundo a função de avaliação.

Busca Informada ou heurística

- Função de avaliação ($f(n)$): mede o custo de um nó até o objetivo.
- Função heurística ($h(n)$): custo estimado do caminho mais econômico do nó n até o nó objetivo.

Busca Gulosa

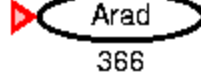
- Expande o nó mais próximo da meta, supondo que isso levará rapidamente a uma solução;
- Portanto, $f(n) = h(n)$
- Exemplo: encontrar uma rota na Romênia usando da ***heurística da distância em linha reta (hDLR)***

Busca Gulosa

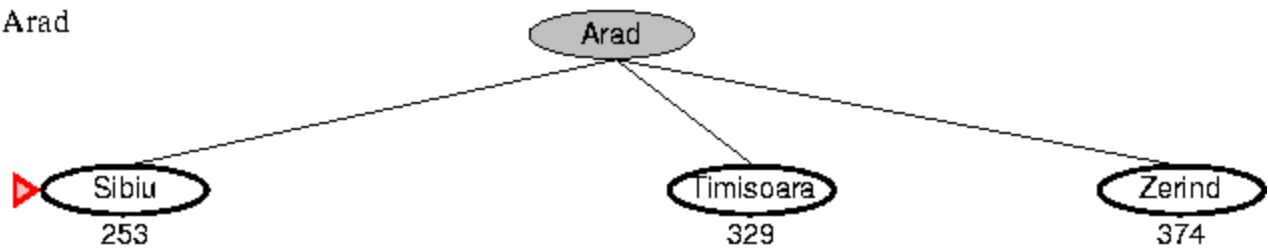
Arad	366	Mehadia	241
Bucharest	0	Neamt	234
Craiova	160	Oradea	380
Dobreta	242	Pitesti	100
Eforie	161	Rimnicu Vilcea	193
Fagaras	176	Sibiu	253
Giurgiu	77	Timisoara	329
Hirsova	151	Urziceni	80
Iasi	226	Vaslui	199
Lugoj	244	Zerind	374

Distâncias em linha reta de cada cidade romena à Bucareste.

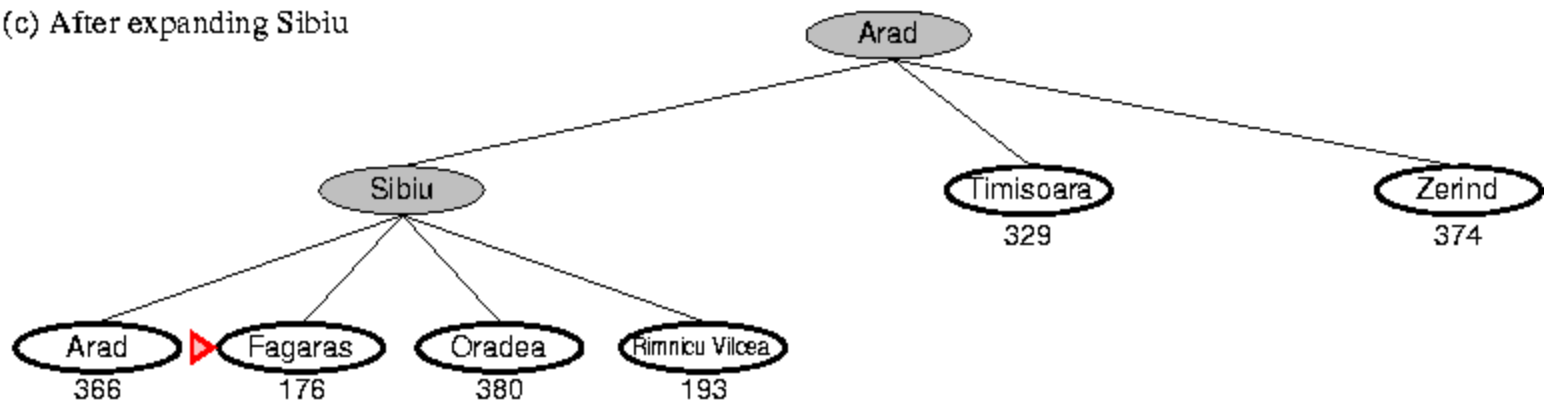
(a) The initial state



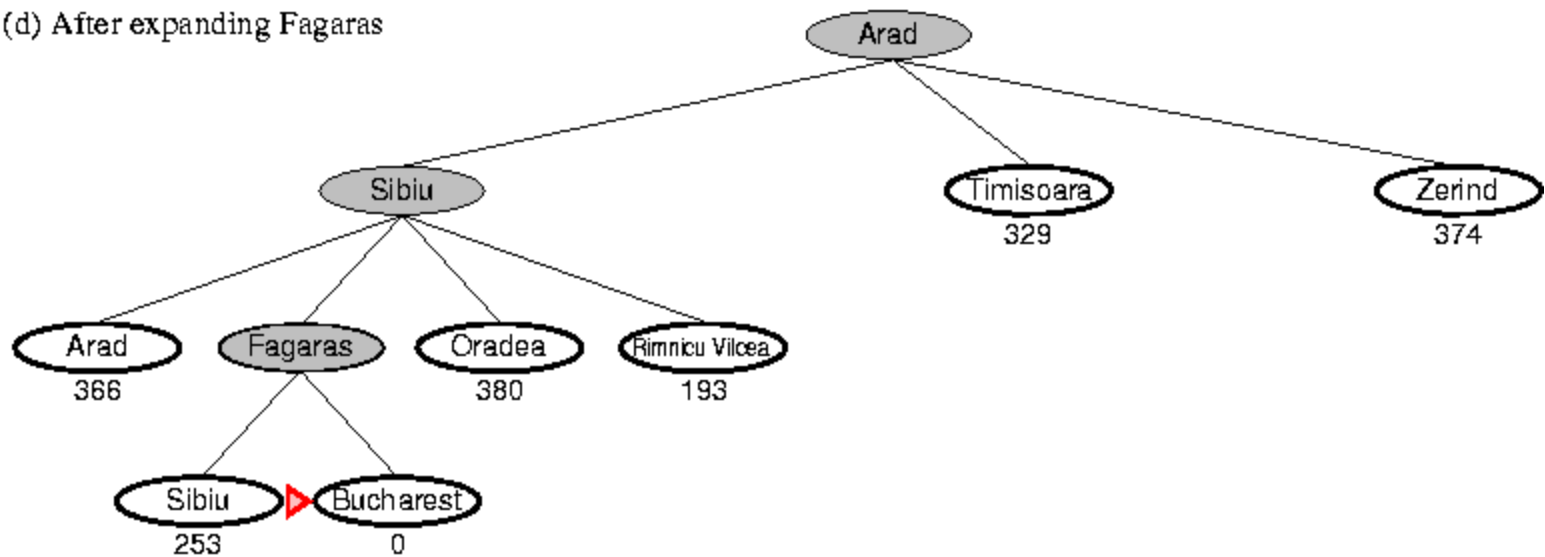
(b) After expanding Arad



(c) After expanding Sibiu



(d) After expanding Fagaras



Busca Gulosa

- Ótima?? **Não!!** pois segue o melhor passo **considerando somente o momento atual.**
- Esta busca assim concebida é suscetível a cair em mínimos locais.
- Pode haver um caminho melhor seguindo algumas opções piores em alguns pontos da árvore de busca.

Busca Gulosa

- Completa?
- Tempo?
- Espaço?

Outro exemplo

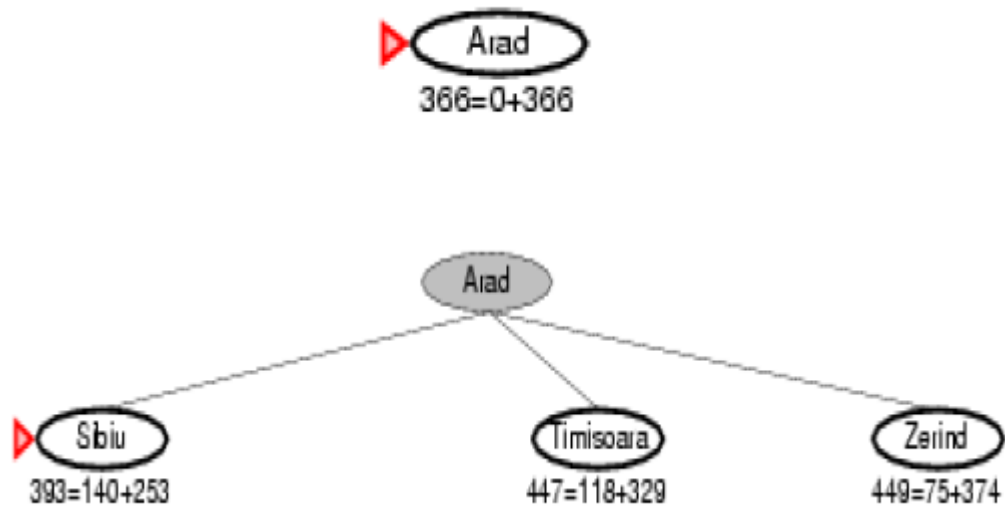


	POA	FLN	CWB	SÃO	RIO	BHZ	BSB	CGB	SSA	FOR	MAO
POA	0	376	547	852	1124	1341	1621	1679	2303	3241	3132
FLN	376	0	251	489	748	973	1315	1544	1931	2858	2982
CWB	547	251	0	339	676	821	1081	1302	1784	2671	2734
SÃO	852	489	339	0	357	490	873	1326	1454	2369	2589
RIO	1124	748	676	357	0	340	933	1576	1210	2190	2849
BHZ	1341	973	821	490	340	0	621	1373	964	1893	2446
BSB	1621	1315	1081	873	933	621	0	880	1059	1688	1939
CGB	1679	1544	1302	1326	1576	1373	880	0	1915	2329	1453
SSA	2303	1931	1784	1454	1210	964	1059	1925	0	1028	2606
FOR	3241	2858	2671	2369	2190	1893	1688	2329	1028	0	2384
MAO	3132	2982	2734	2589	2849	2446	1939	1453	2606	2384	0

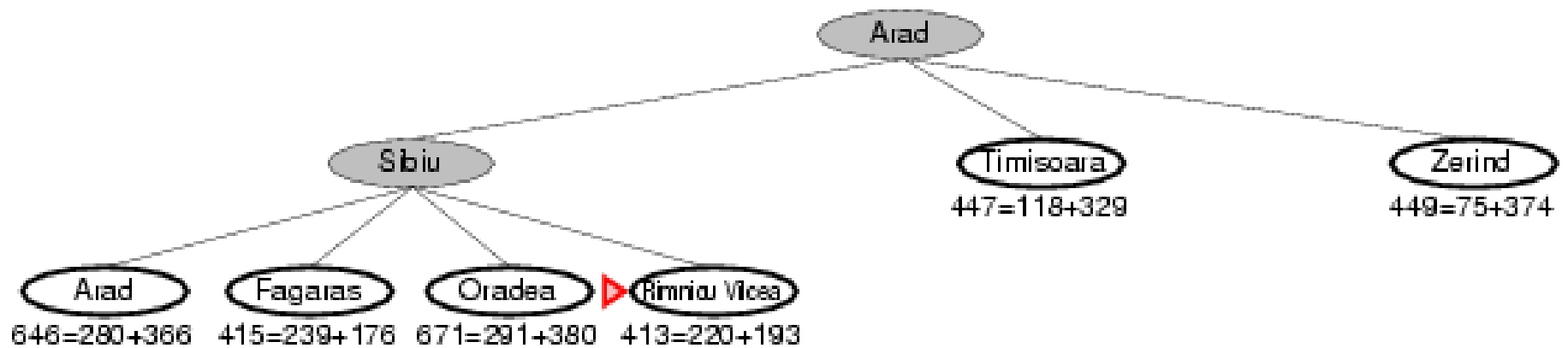
Busca A*

- Avalia nós combinando o custo para alcançar cada nó ($g(n)$) e o custo estimado para ir deste nó até o objetivo ($h(n)$): $f(n) = g(n) + h(n)$.
 - $g(n)$ = custo até o momento para alcançar n
 - $h(n)$ = custo estimado de n até o objetivo
 - $f(n)$ = custo total estimado do caminho através de n até o objetivo.
- Para a solução de custo mais baixo, seguir os estados de menor valor de $g(n) + h(n)$.

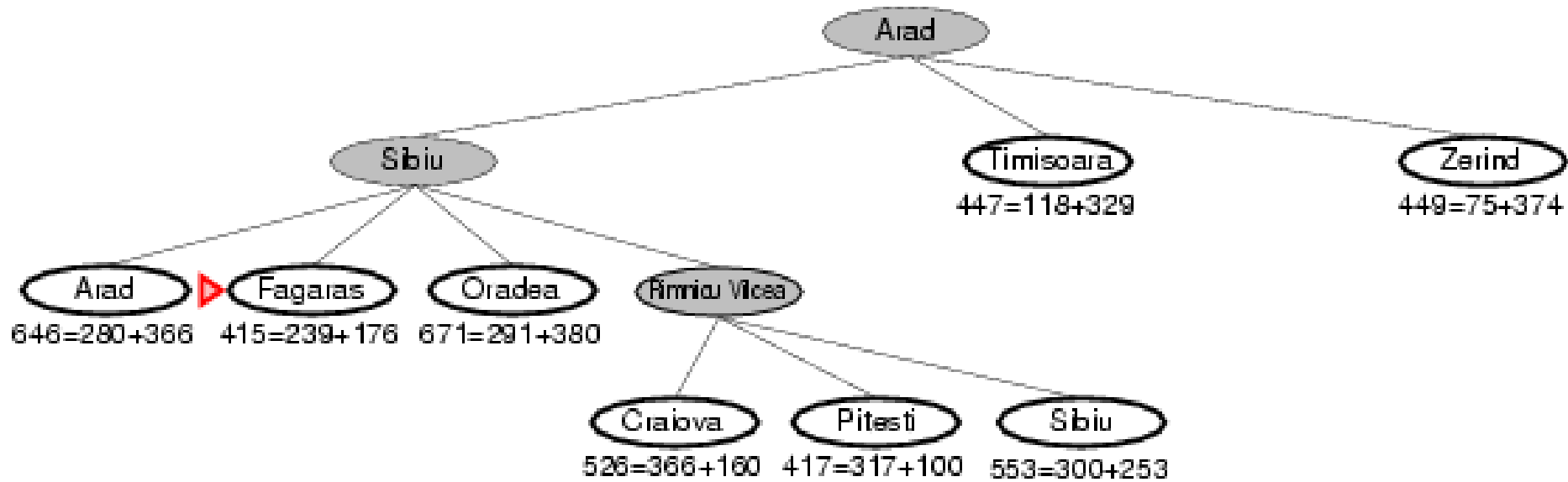
Busca A*



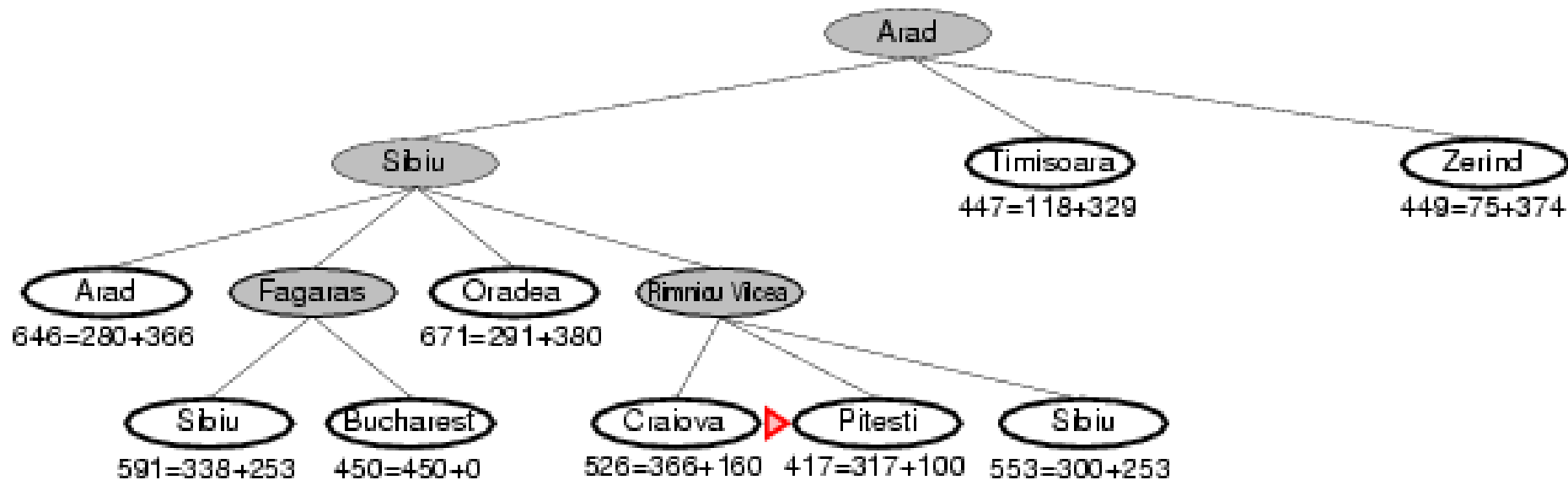
Busca A*



Busca A*



Busca A*



Busca A*

