Inteligência Artificial

Busca Heurística

Sumário

- Estratégias de Busca Heurística
  - Busca pela Melhor Escolha (BFS)
    - Busca Gulosa
    - Busca A*
    - Busca IDA*
    - Busca pela Melhor Escolha Recursiva (RBFS)
- Funções Heurísticas

Busca Heurística

Estratégias Heurísticas de Busca

Busca Heurística

- Na busca exaustiva, nenhuma informação adicional sobre o problema era usada

- As buscas heurísticas (informadas) utilizam informações específicas sobre o problema sendo resolvido

Best First Search

- A Busca pela Melhor Escolha seleciona qual o melhor nó para expandir a partir de uma Função de Avaliação, *f(n)*
  - Geralmente o *menor* valor é o escolhido

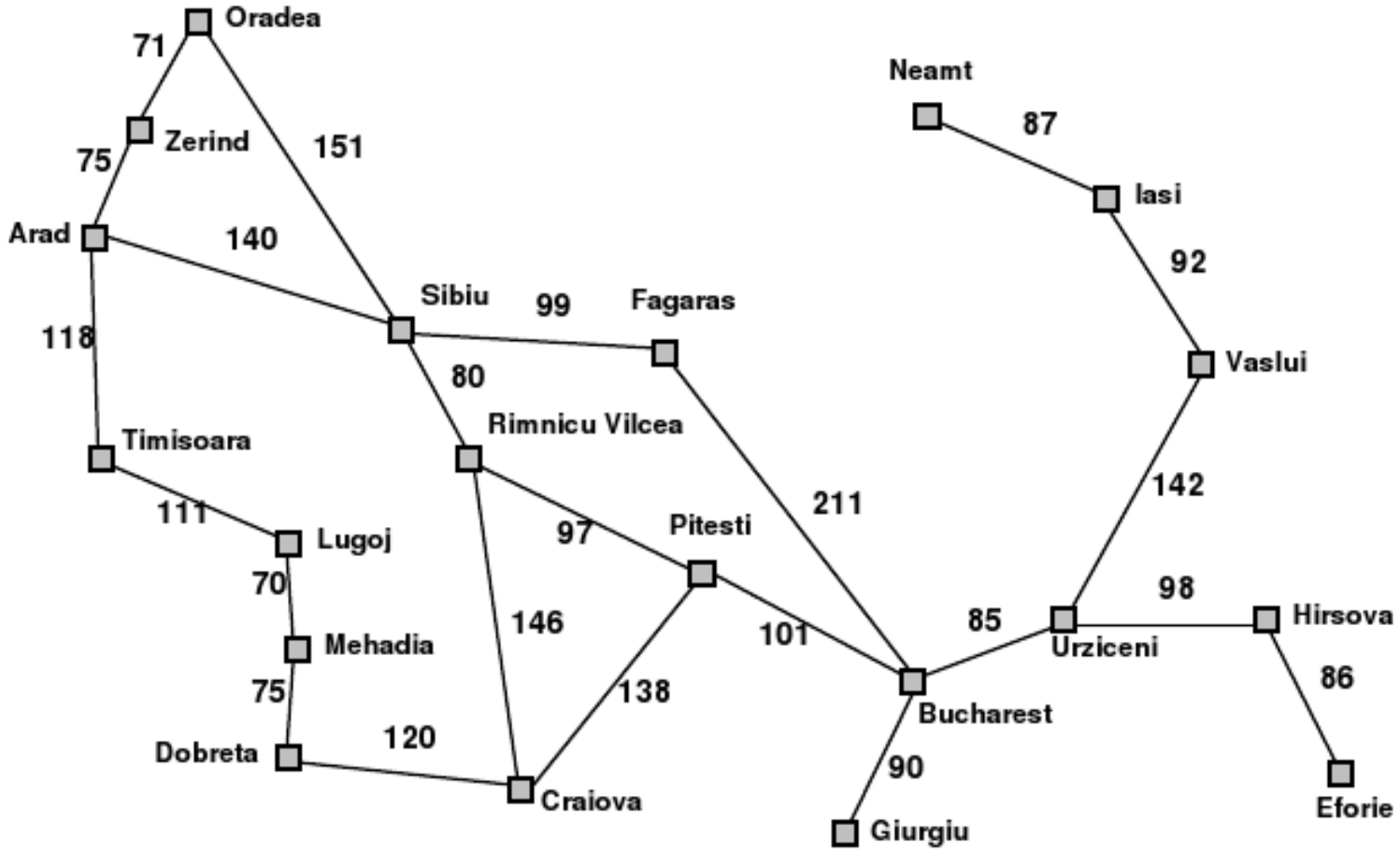
- Busca pela Melhor Escolha utiliza fila de prioridades como *franja*

Função Heurística

- Em geral, a função Heurística *h(n)* representa:
  - *O custo **estimado** do caminho mais curto do nó **n** para um nó **objetivo***
  - se ***n*** for um nó objetivo, ***h(n) = 0***

- Viajante na Romênia
  - $h_{DLR}(n) \rightarrow$ distância em linha reta de ***n*** até Bucareste
  - $h_{DLR}(Em(Arad)) = 366$
  - O valor da distância em linha reta não pode ser calculado a partir da definição do problema

Viajante na Romênia



Distância em Linha Reta até Bucareste

Arad	366
Bucharest	0
Craiova	160
Dobreta	242
Eforie	161
Fagaras	178
Giurgiu	77
Hirsova	151
Iasi	226
Lugoj	244
Mehadia	241
Neamt	234
Oradea	380
Pitesti	98
Rimnicu Vilcea	193
Sibiu	253
Timisoara	329
Urziceni	80
Vaslui	199
Zerind	374

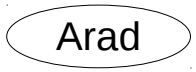
Busca Gulosa

- Expande o nó que está mais próximo do objetivo
  - Espera-se que isso leve a uma solução rapidamente
  - Função de Avaliação

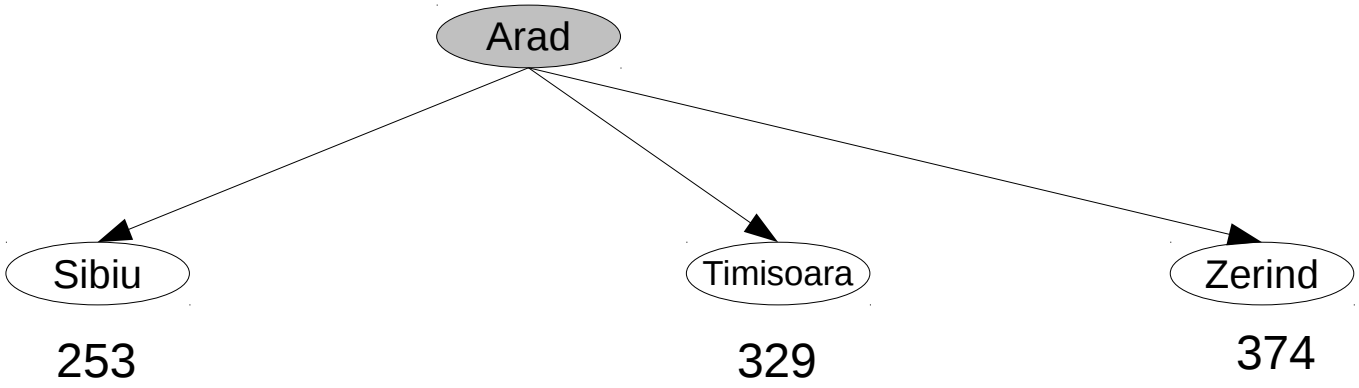
$$f(n) = h(n)$$

    - *h(n)* é a função Heurística

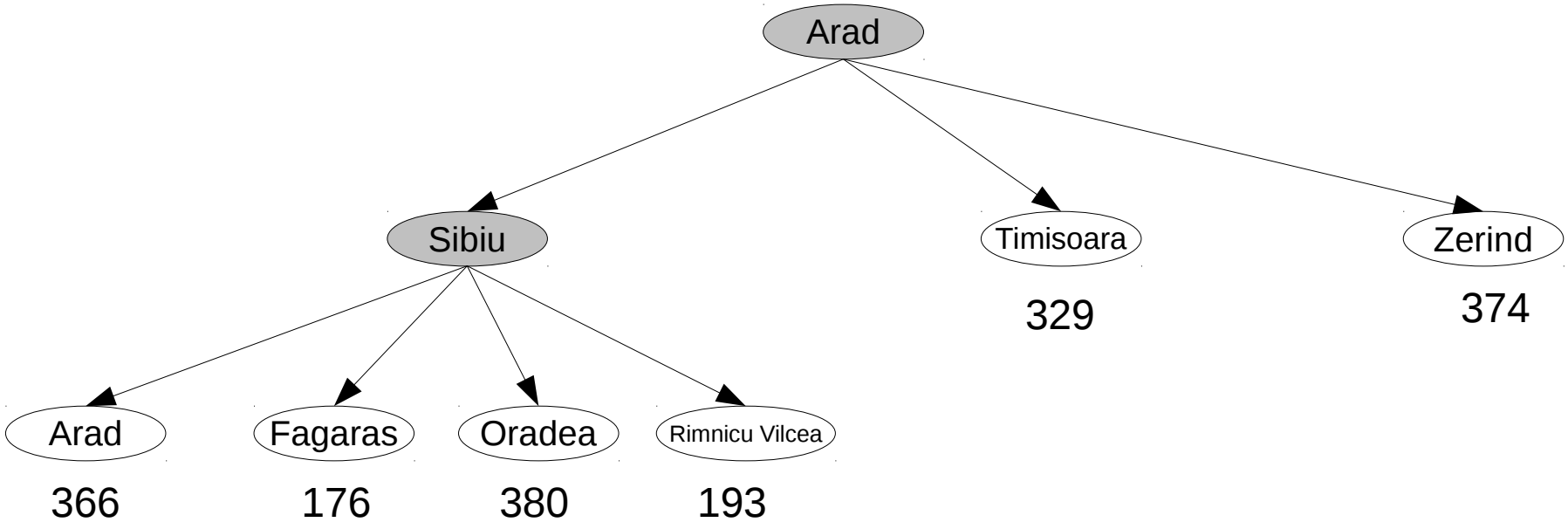
Viajante na Romênia



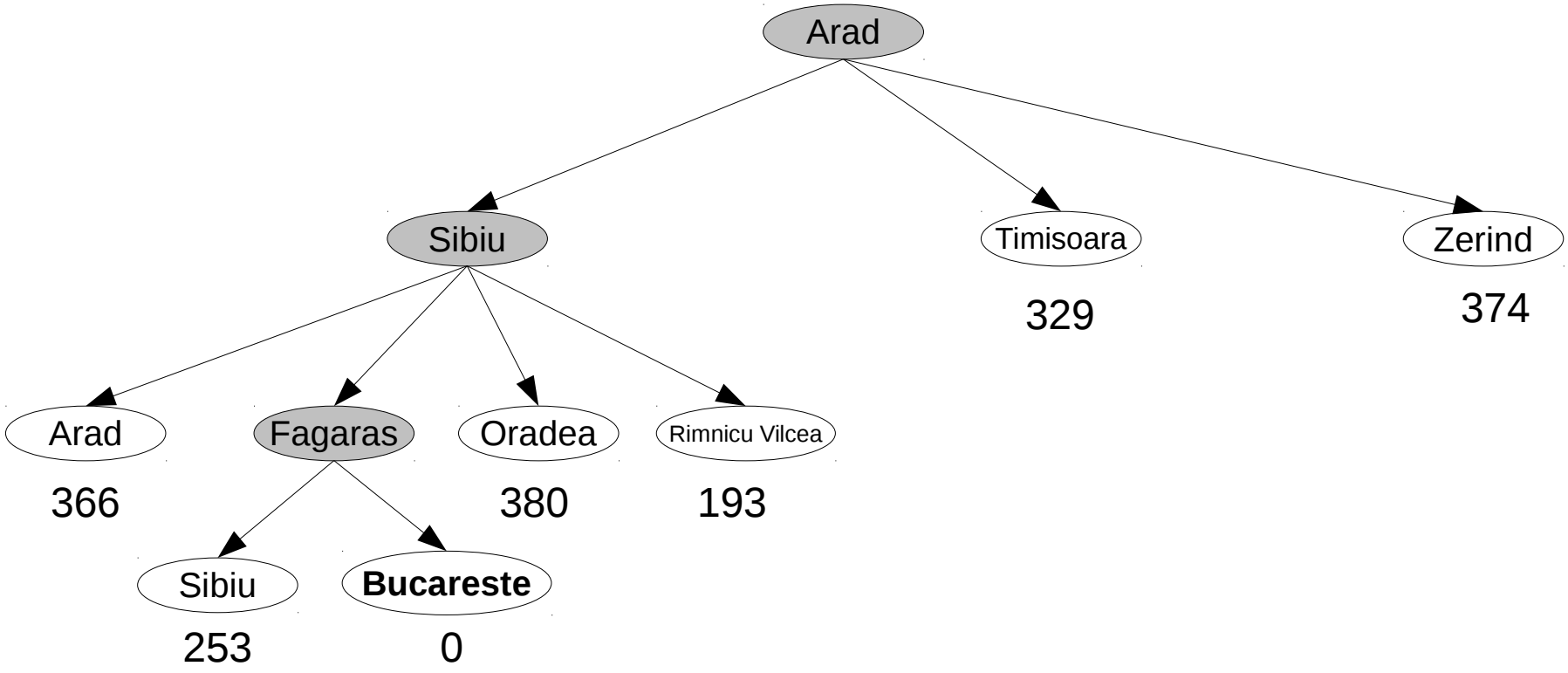
Viajante na Romênia



Viajante na Romênia



Viajante na Romênia



Análise

- Completude
  - Não. Pode ficar preso em *loops*
    - *Iasi* → *Neamt* → *Iasi* → *Neamt* → …
  - Garante com uma *lista fechada*
- Optimalidade
  - Não. Por isso *Guloso*
  - Se passasse por Rimnicu Vilcea e Pitesti, ganharia 32 quilômetros
- Complexidade (tempo e espaço)
  - $O(b^m), \text{Para o pior caso}$
  - Uma boa Heurística reduz esta complexidade significativamente
    - Depende do problema e da qualidade da heurística

Busca A*

- Evita expandir os caminhos que já são muito caros
  - Função de Avaliação

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

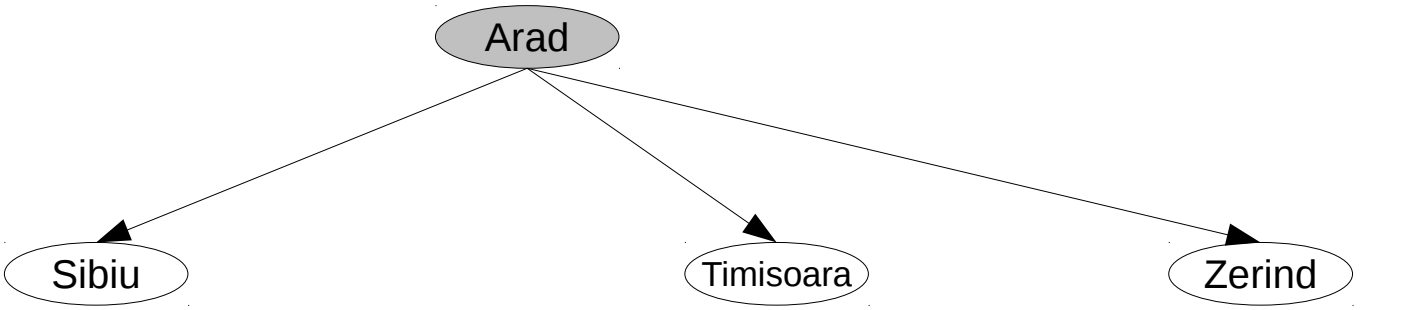
    - *g(n)* é o custo do caminho até o momento
    - *h(n)* é a função Heurística

Viajante na Romênia

Arad

366=0+366

Viajante na Romênia

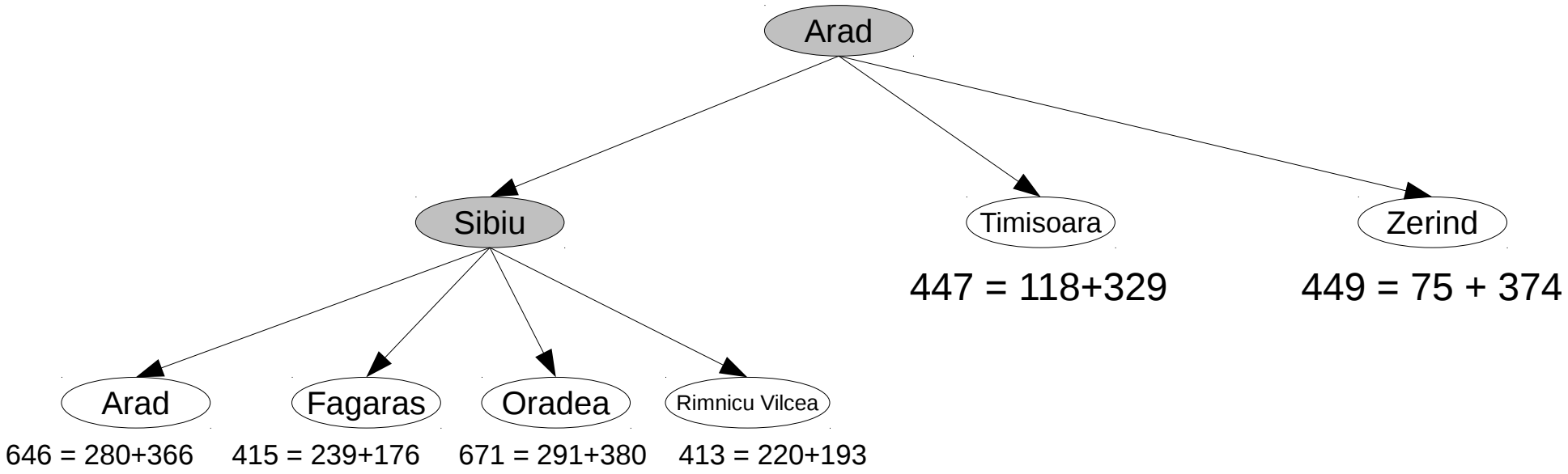


393 = 140 +253

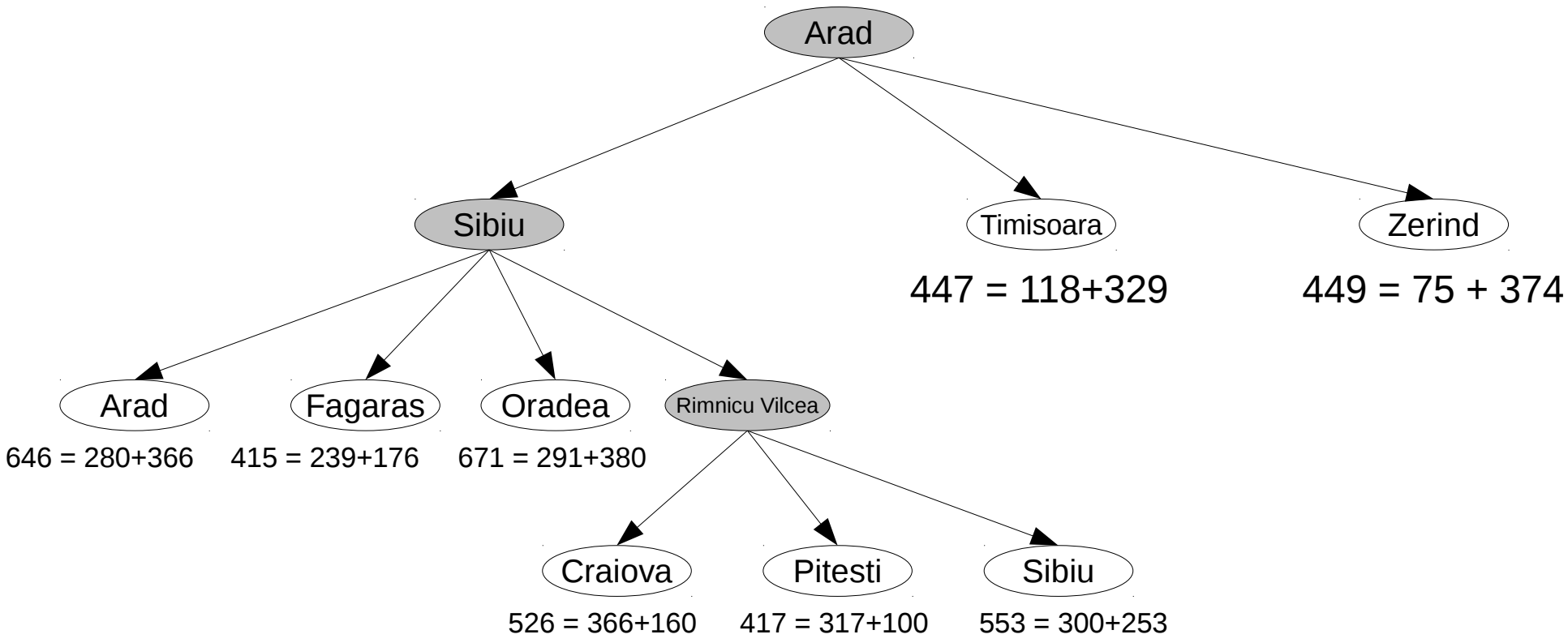
447 = 118+329

449 = 75 + 374

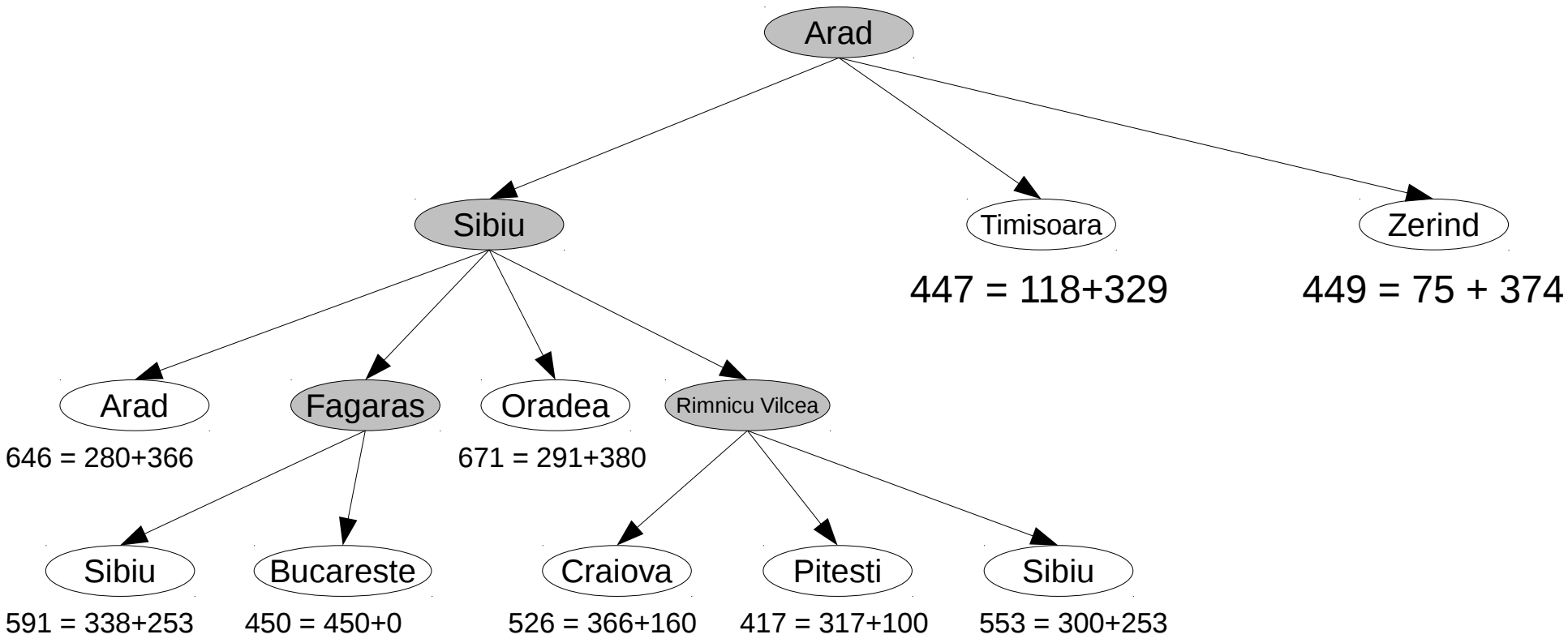
Viajante na Romênia



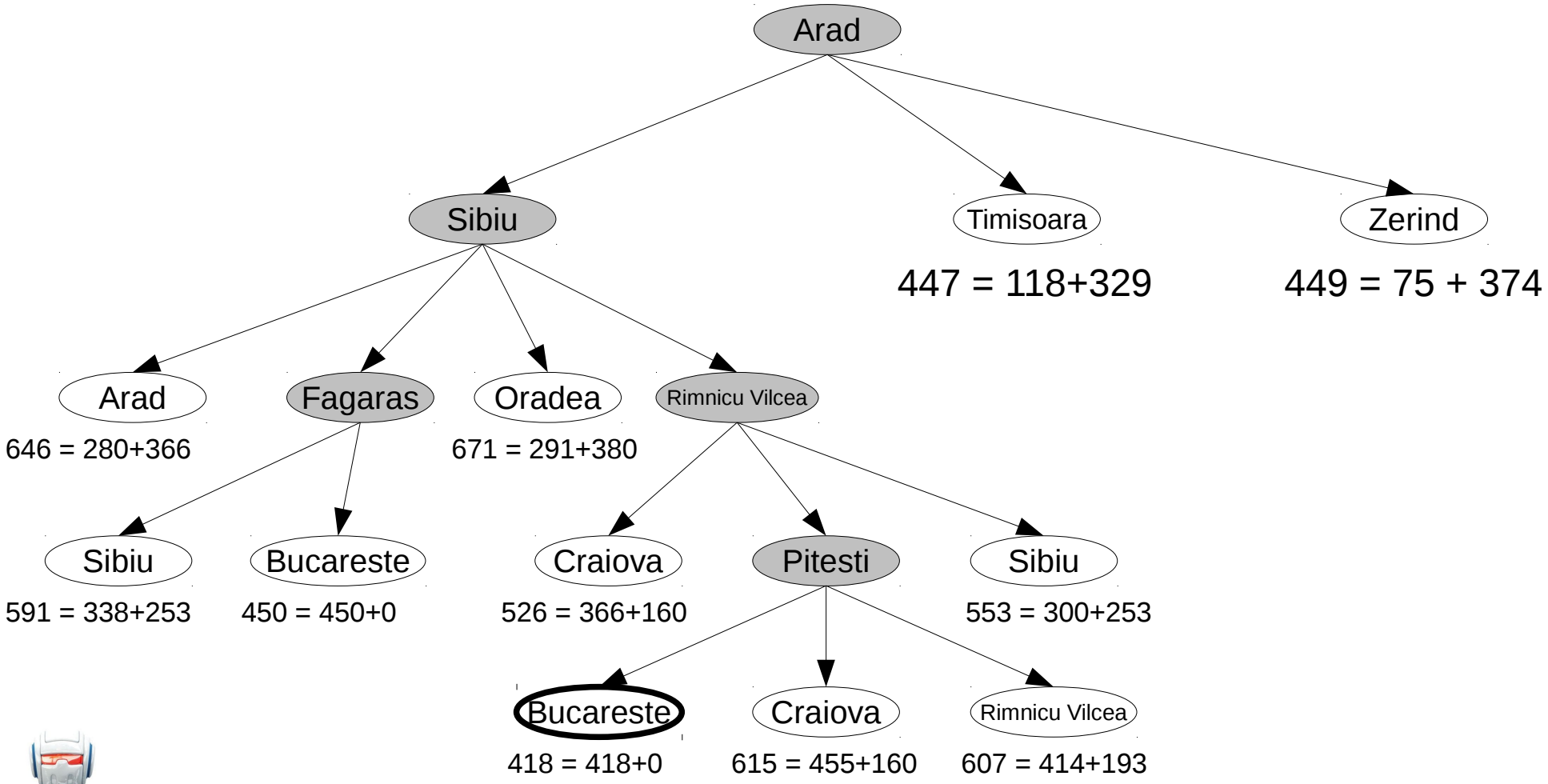
Viajante na Romênia



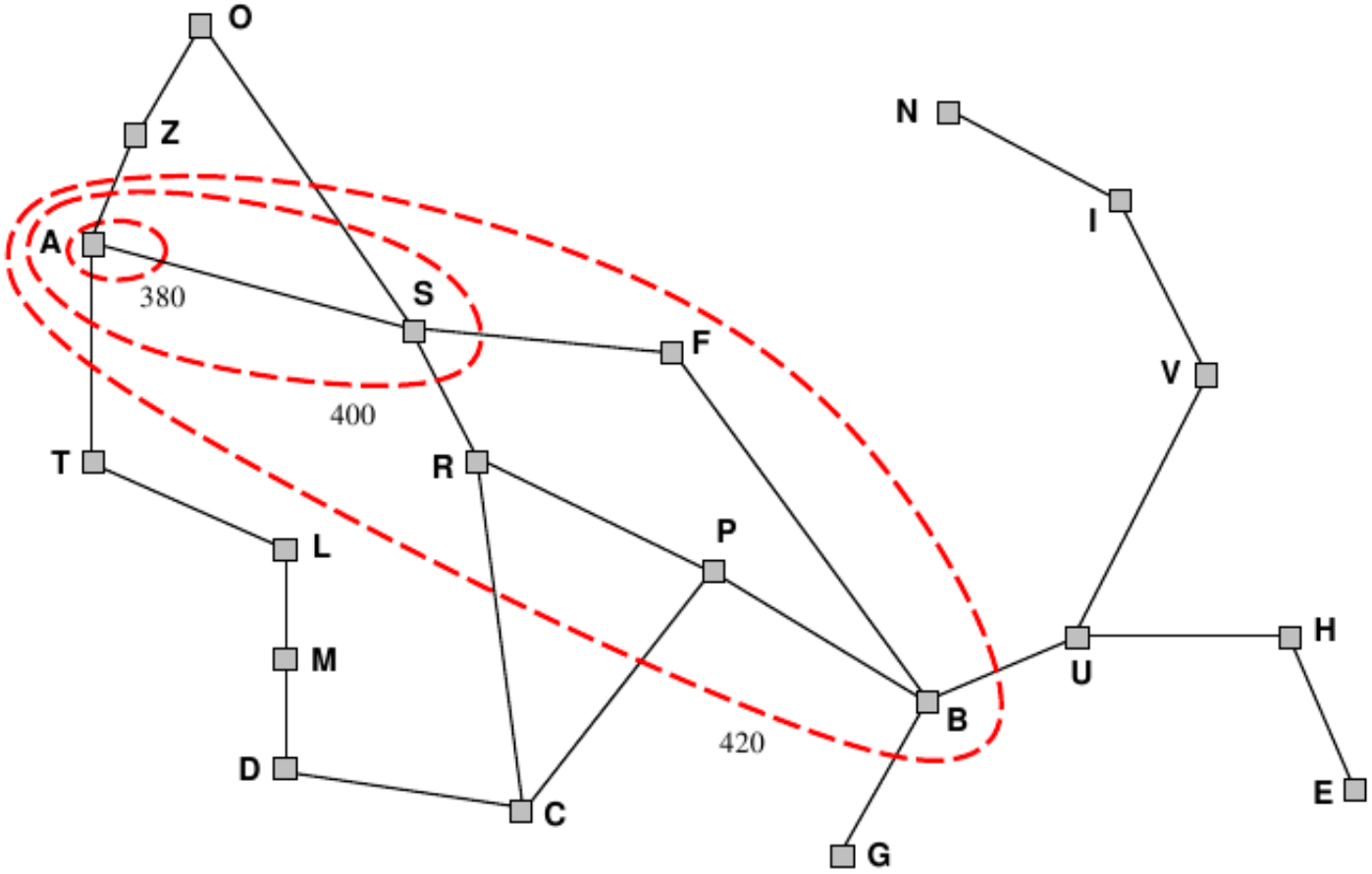
Viajante na Romênia



Viajante na Romênia



Contorno do Espaço de Busca



Análise

- Completude
  - Sim
    - A não ser que existam infinitos nós com $f < f(G)$
- Optimalidade
  - Sim
    - Desde que a função *h(n)* seja uma heurística admissível (nunca sobre-estime o custo real)
- Complexidade
  - Para sub-exponencial

$$|h(n) - h^*(n)| \leq O(\log h^*(n))$$

    - *h*(n)* é o custo ótimo real de *n* até o objetivo

Análise

- Geralmente o erro é linear
  - A complexidade é, então, exponencial
  - O algoritmo exaure a memória bem antes gastar muito tempo
    - Invalida o uso do algoritmo para problemas muito grandes
    - Mesmo assim, é muito melhor que Busca Exaustiva
  - Pode-se sacrificar a optimalidade
    - Heurísticas mais precisas, mas não estritamente admissíveis

Análise

- A* expande
  - Todos os nós com $f(n) < C^*$
    - Nenhum outro algoritmo **ótimo** expande menos nós do que A* (otimamente eficiente)
  - Alguns nós com $f(n) = C^*$
  - Nenhum nó com $f(n) > C^*$
    - Subárvores com custo maior são *podados*
  - Onde C^* é o custo da solução ótima

Prova da Optimalidade

- Suponha *G2* um objetivo não ótimo

  $f(G2) = g(G2) + h(G2) > C^*$

- Suponha um nó *n* qualquer no caminho ótimo

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

  - Como $h(n)$ nunca sobre-estima a distância

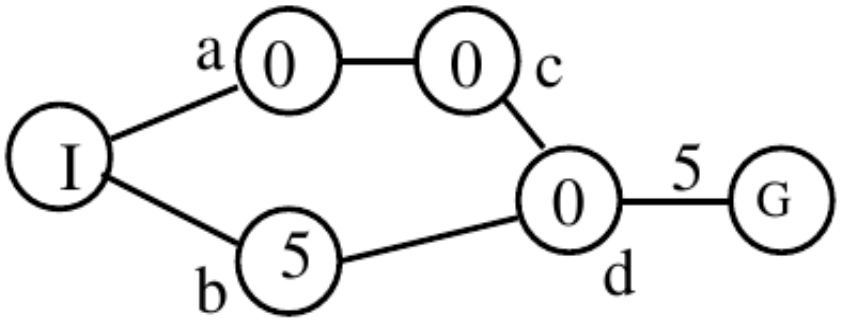
  $f(n) \leq C^*$

- Então G2 nunca é expandido, pois

  $f(n) \leq C^* < f(G2)$

A* em Grafos

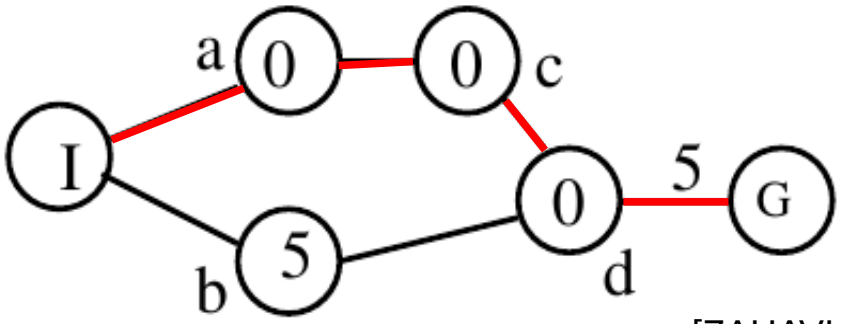
- A prova anterior não serve para Grafos
  - Caminhos ótimos podem ser descartados devido a um estado repetido



  - Todos os passos custam 1, exceto quando explicitado
  - Dentro dos nós está representado o valor de *h*

A* em Grafos

- A prova anterior não serve para Grafos
  - Caminhos ótimos podem ser descartados devido a um estado repetido



[ZAHAVI et al., 2007]

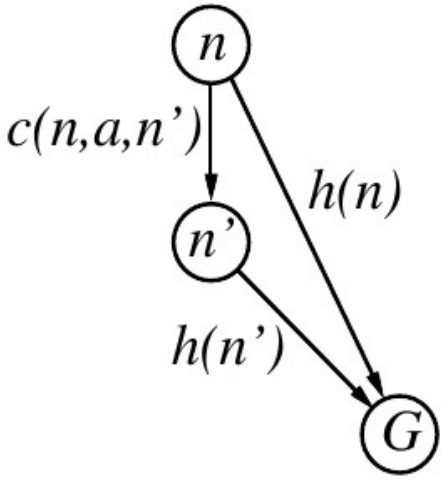
- Todos os passos custam 1, exceto quando explicitado
- Dentro dos nós está representado o valor de *h*
- *f(IacdG)* = 8 e *f(IbdG) = 7*
  - Caminho seguido é o mais caro

A* em Grafos

- Solução 1:
  - Descartar o caminho mais caro quando 2 caminhos são encontrados para o mesmo nó
    - Modificação no algoritmo
      - A *lista fechada* tem que guardar o próprio nó (ou um ponteiro para o nó) além do estado
      - O nó gerado com caminho mais caro tem que ser retirado da *franja*
      - Se o nó já foi expandido, todos os sucessores devem ser removidos da franja ou terem seus custos atualizados
    - Solução Complexa, mas garante optimalidade

A* em Grafos

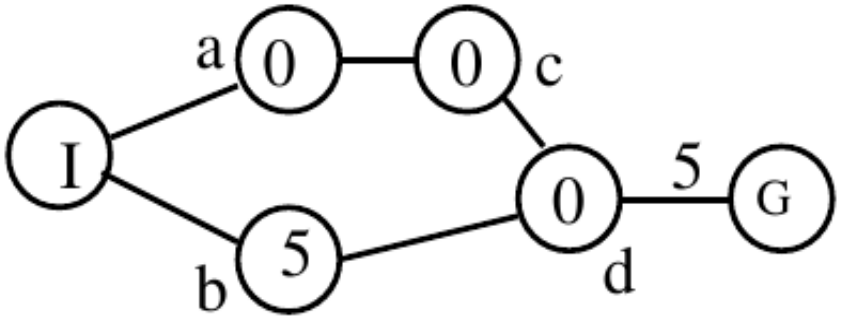
- Solução 2:
  - Garantir que a heurística *h(n)* seja consistente (ou monotônica)



$$h(n) < c(n, a, n') + h(n')$$

Heurística Inconsistente

- A heurística abaixo é inconsistente



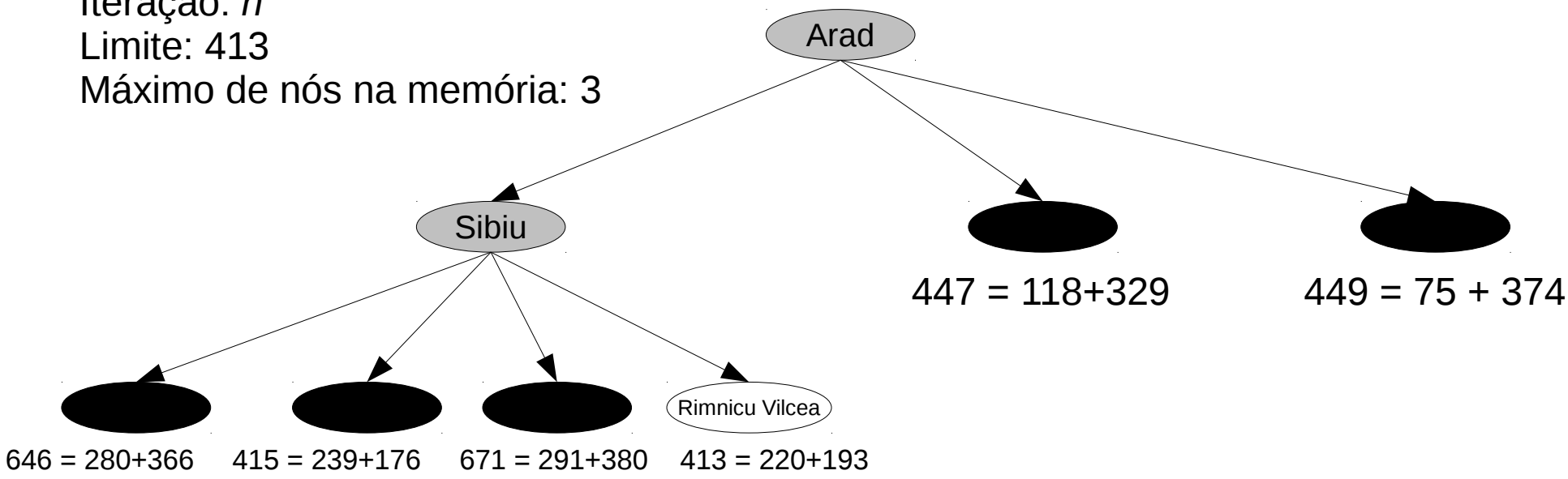
  - *h(b) = 5*
  - *c(b,action,d) = 1*
  - *h(d) = 0*
    - *h(b) > c(b, action, d) + h(d)* → Inconsistente

Buscas Heurísticas com Compromisso de Memória

- IDA*
  - Algoritmo similar ao A*, mas utiliza profundidade iterativa
    - Limita dos nós expandidos em *f*
    - A cada iteração, *f* é o menor *f* que excedeu o *f* anterior
  - Não precisa manter uma fila de prioridades

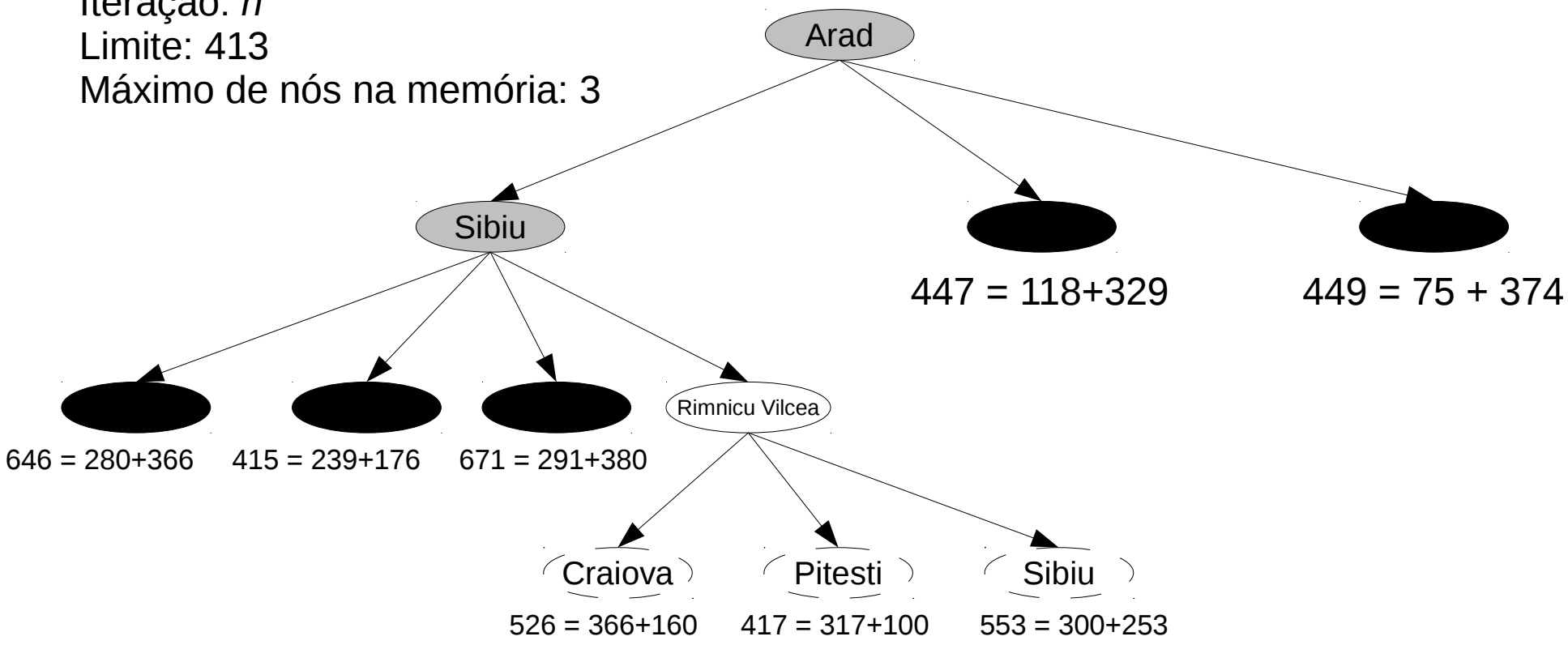
Viajante na Romênia

Iteração: *n*
Limite: 413
Máximo de nós na memória: 3



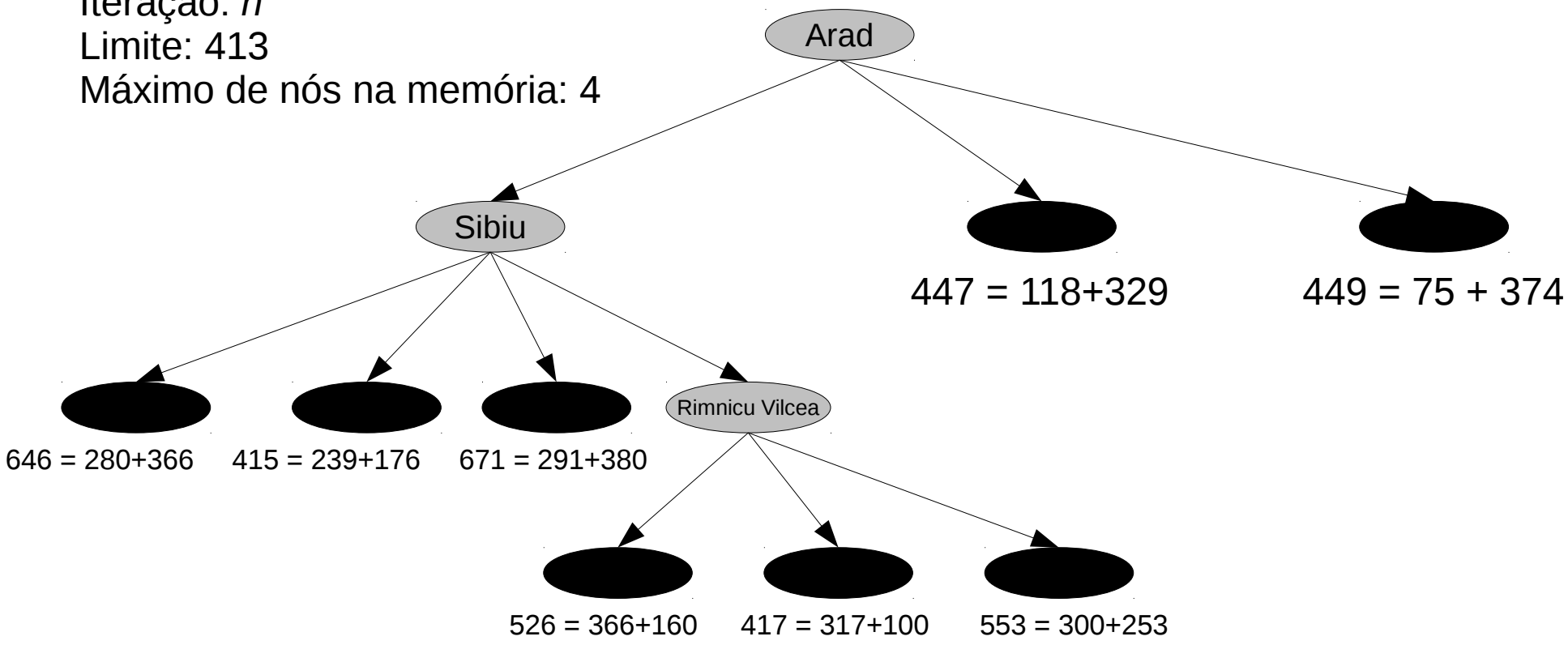
Viajante na Romênia

Iteração: *n*
Limite: 413
Máximo de nós na memória: 3



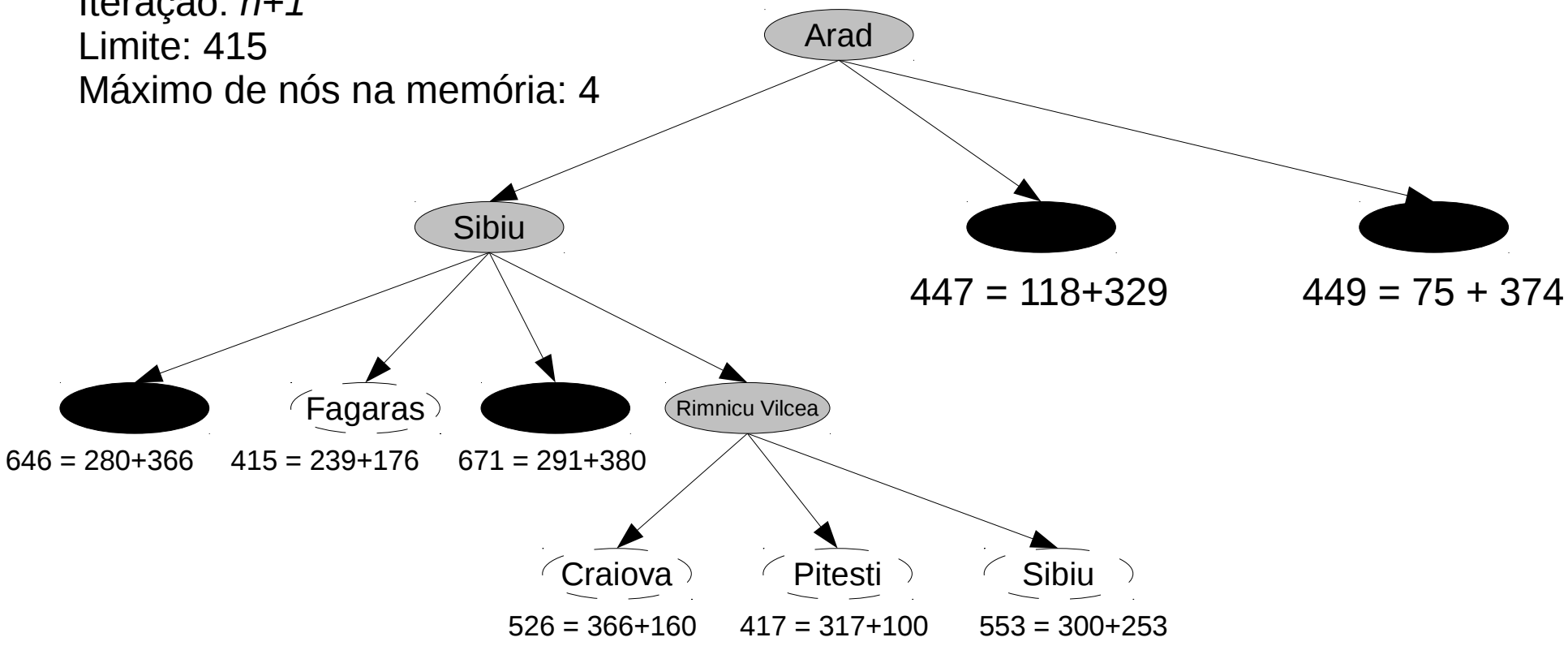
Viajante na Romênia

Iteração: *n*
Limite: 413
Máximo de nós na memória: 4



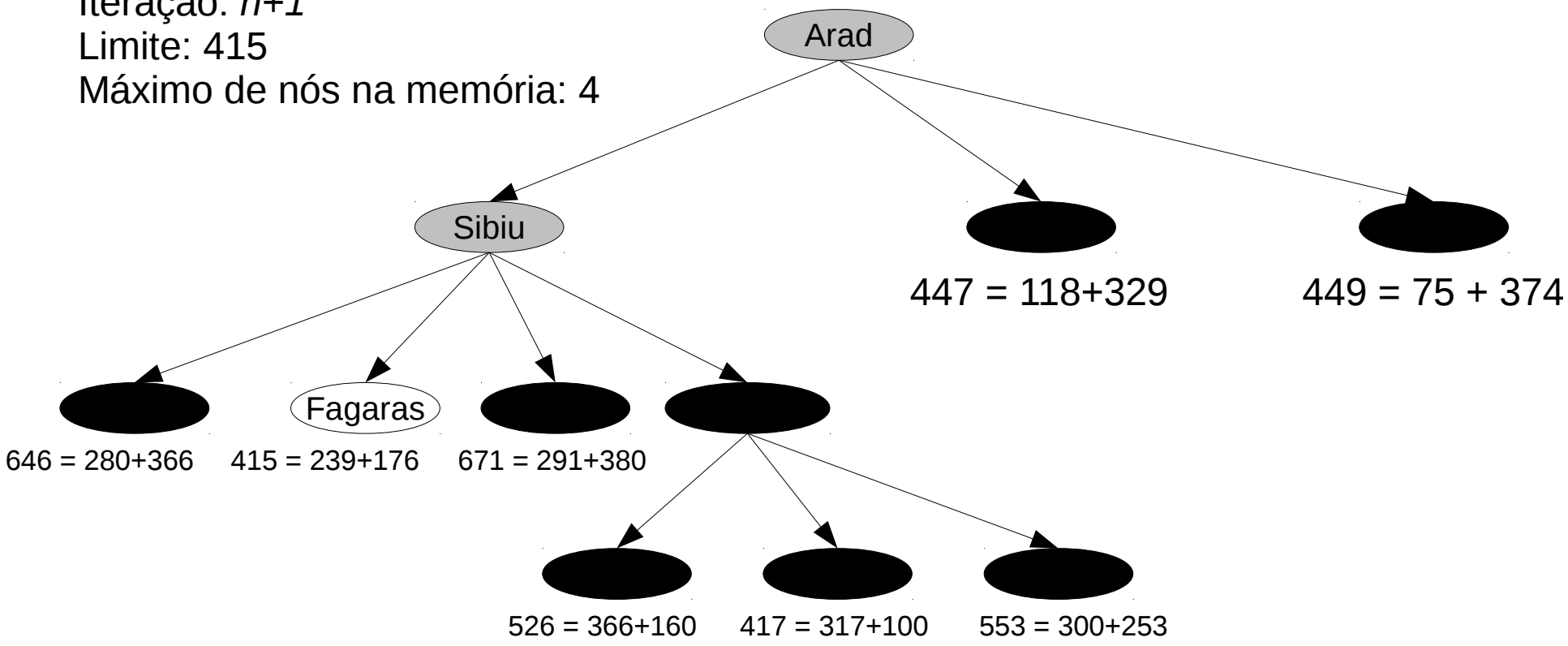
Viajante na Romênia

Iteração: *n+1*
Limite: 415
Máximo de nós na memória: 4



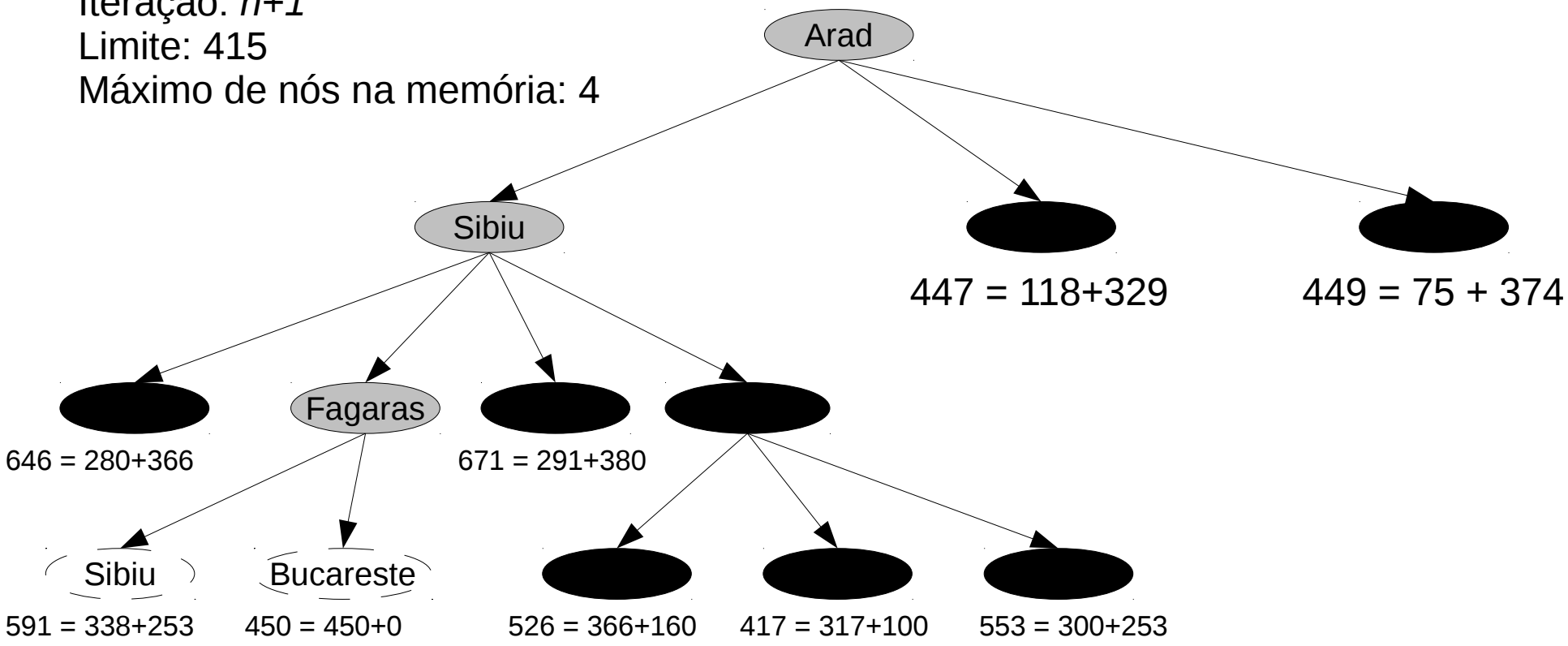
Viajante na Romênia

Iteração: *n+1*
Limite: 415
Máximo de nós na memória: 4



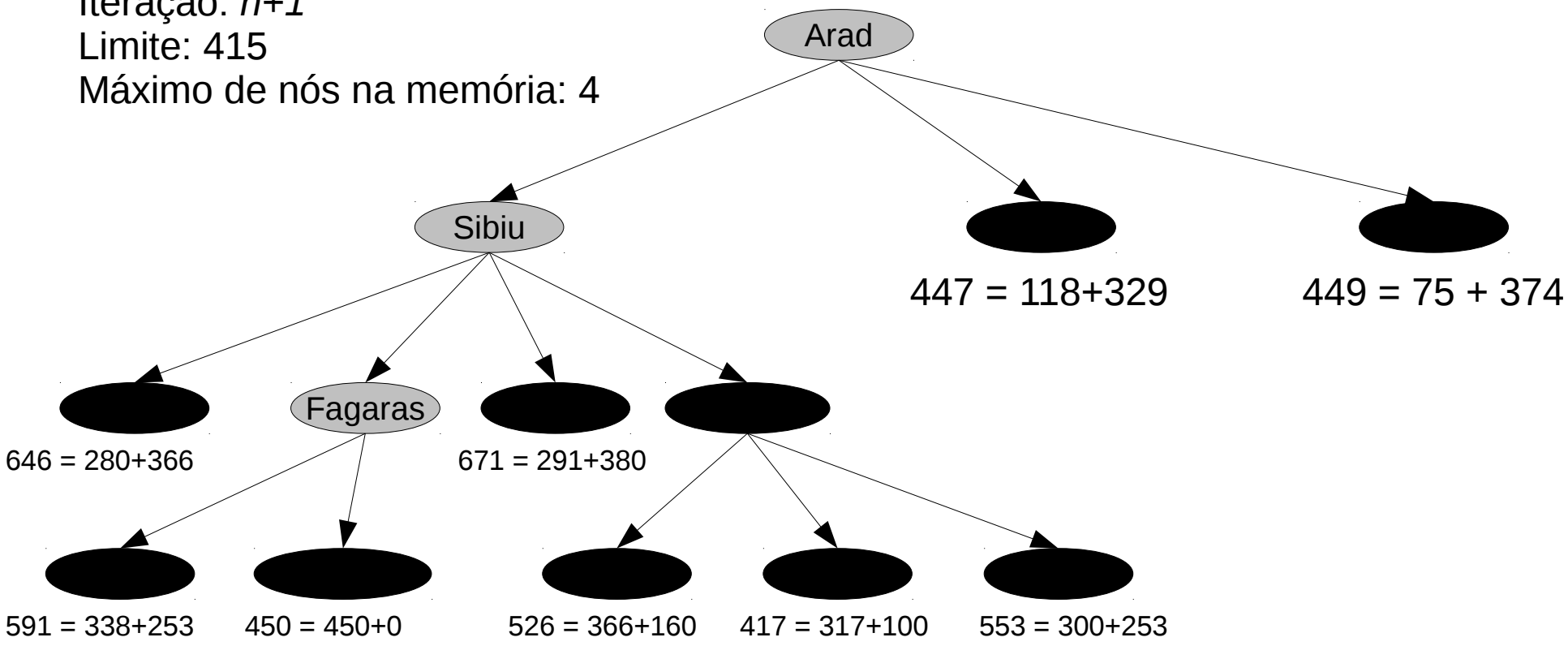
Viajante na Romênia

Iteração: *n+1*
Limite: 415
Máximo de nós na memória: 4



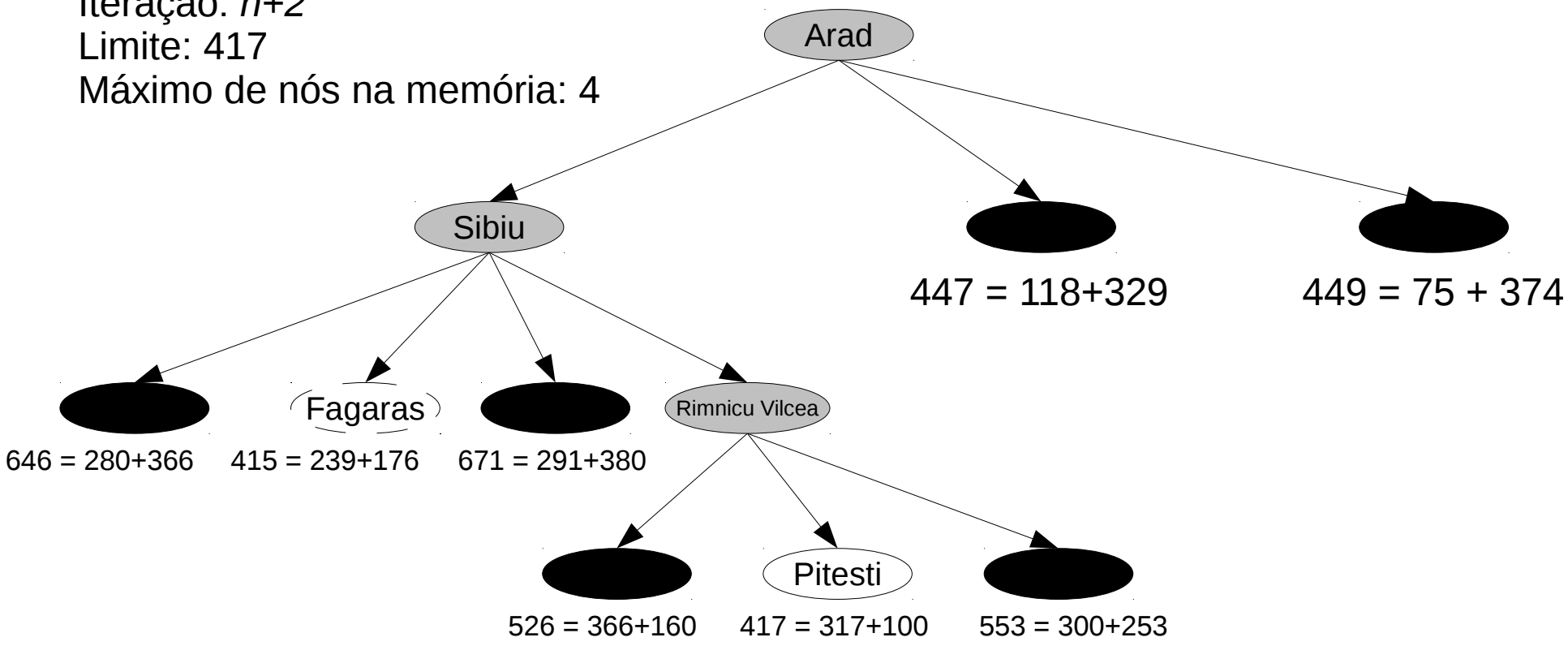
Viajante na Romênia

Iteração: *n+1*
Limite: 415
Máximo de nós na memória: 4



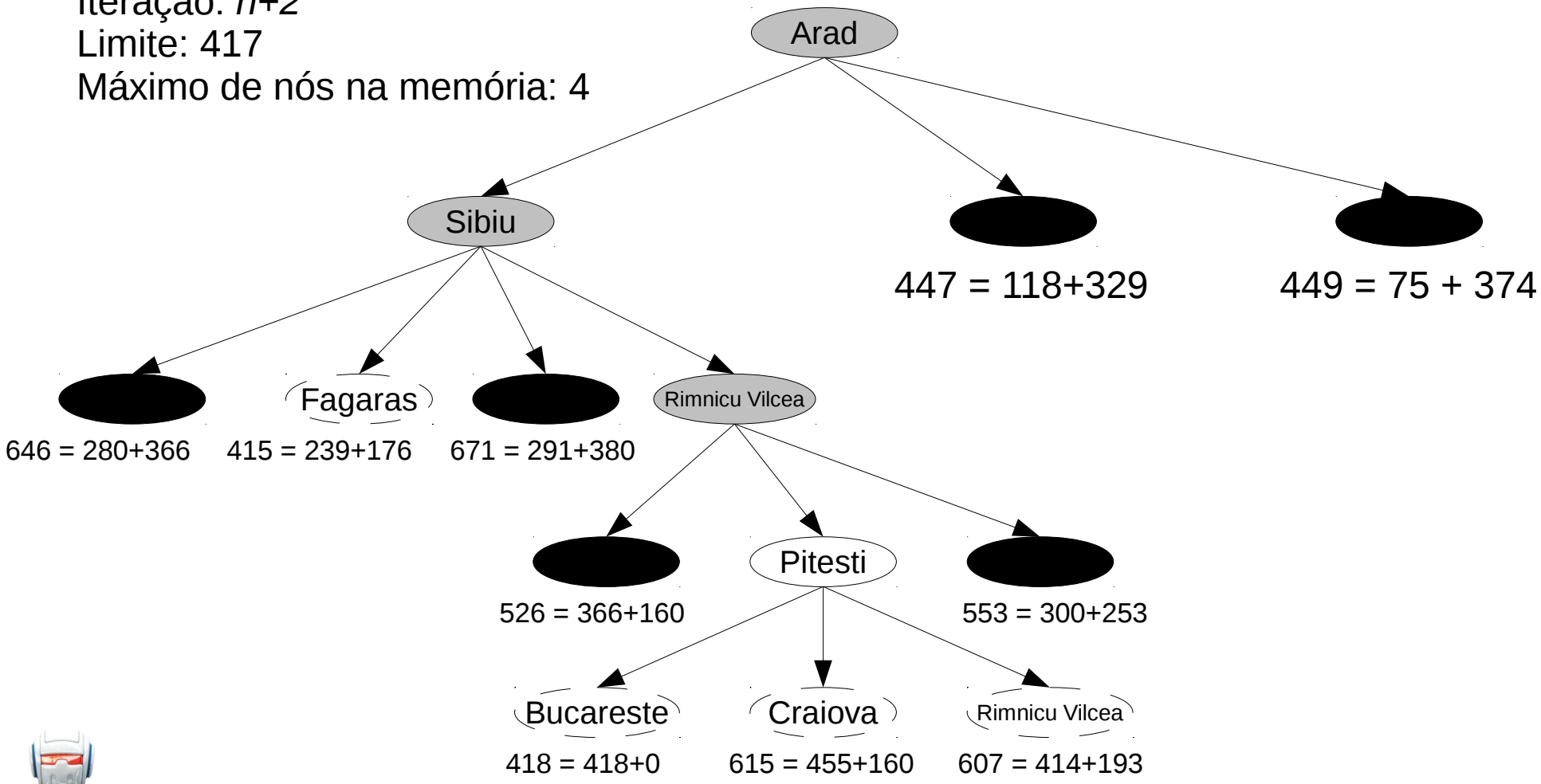
Viajante na Romênia

Iteração: *n+2*
Limite: 417
Máximo de nós na memória: 4



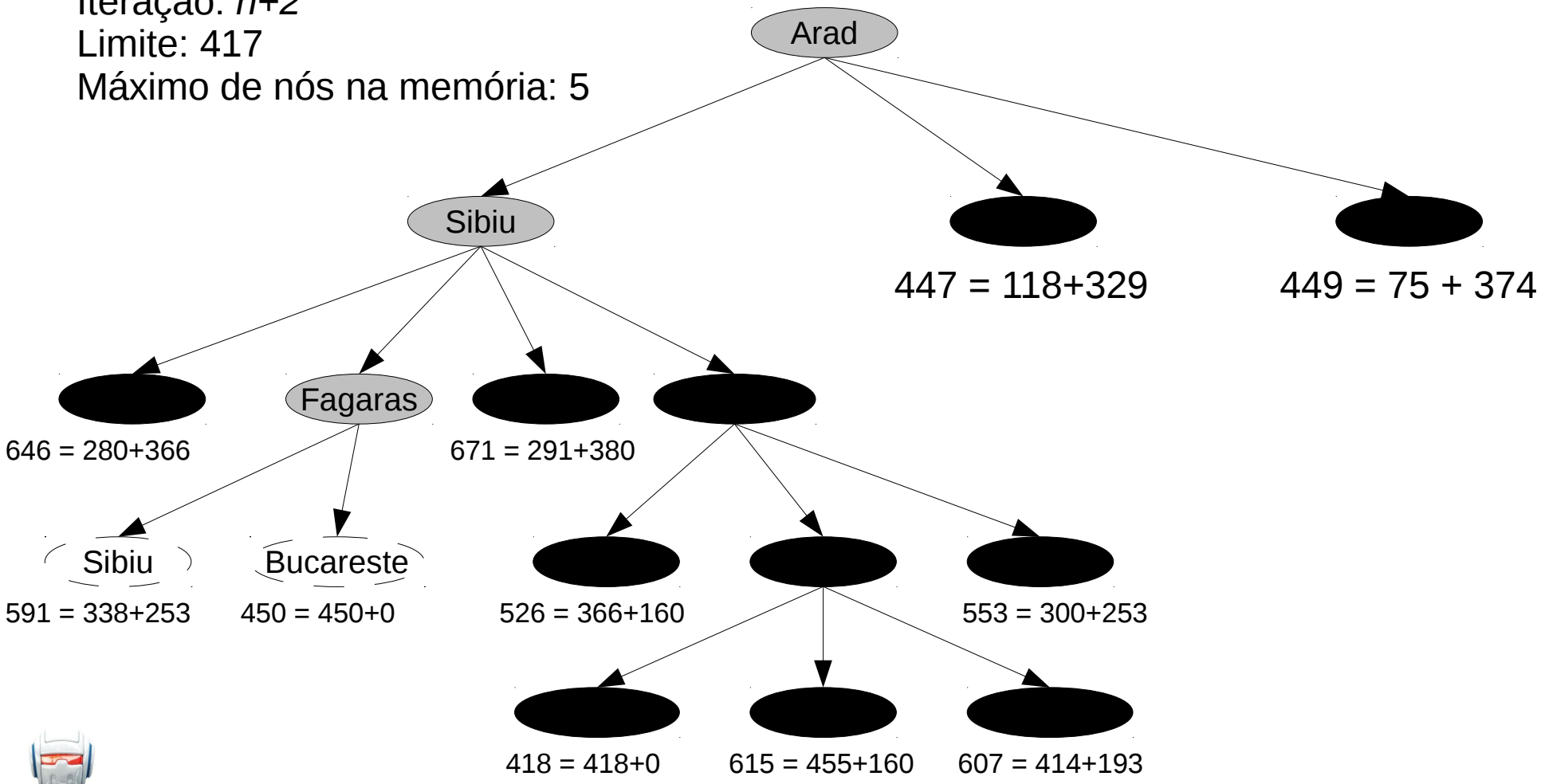
Viajante na Romênia

Iteração: *n+2*
Limite: 417
Máximo de nós na memória: 4



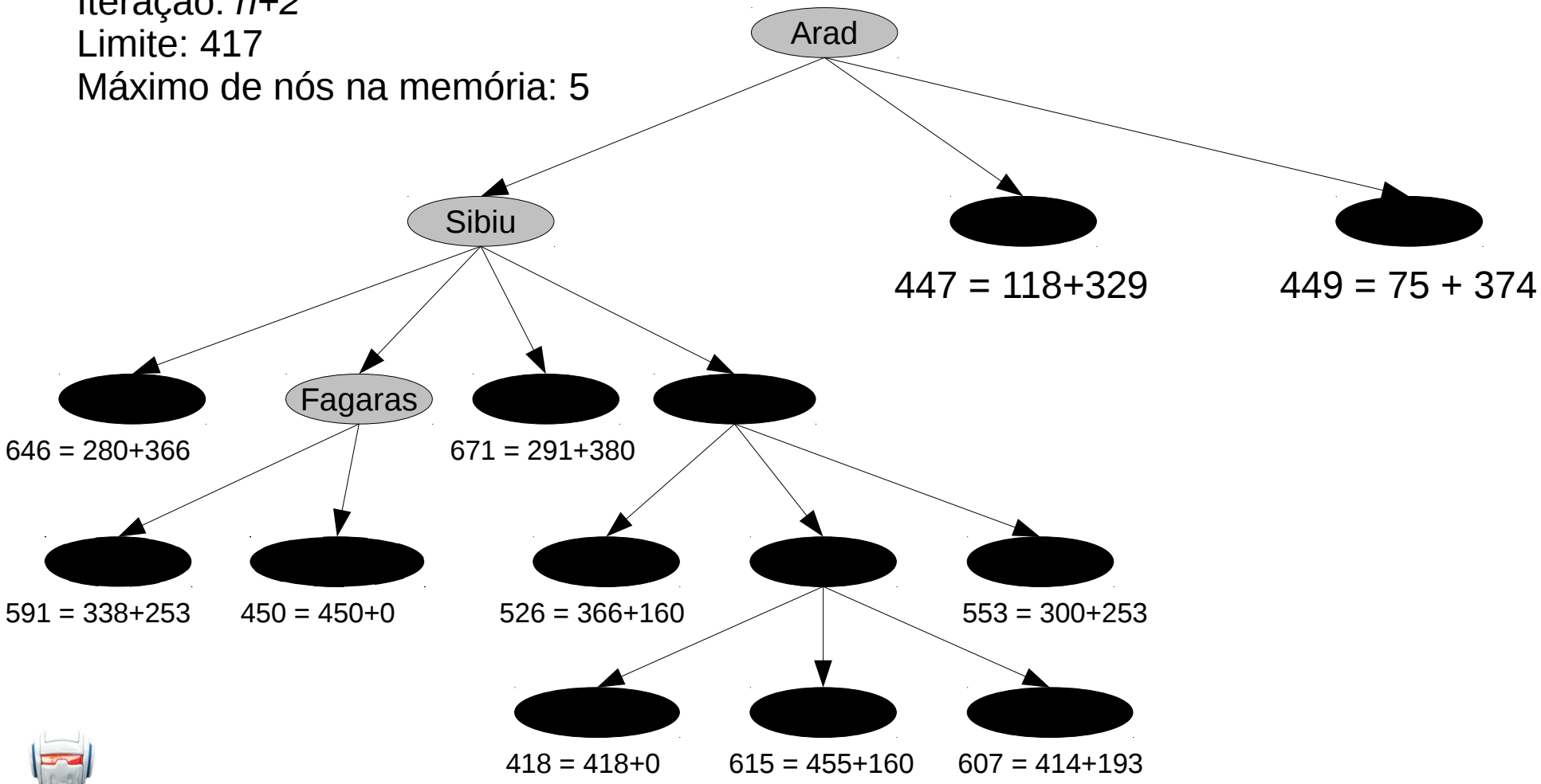
Viajante na Romênia

Iteração: *n+2*
Limite: 417
Máximo de nós na memória: 5



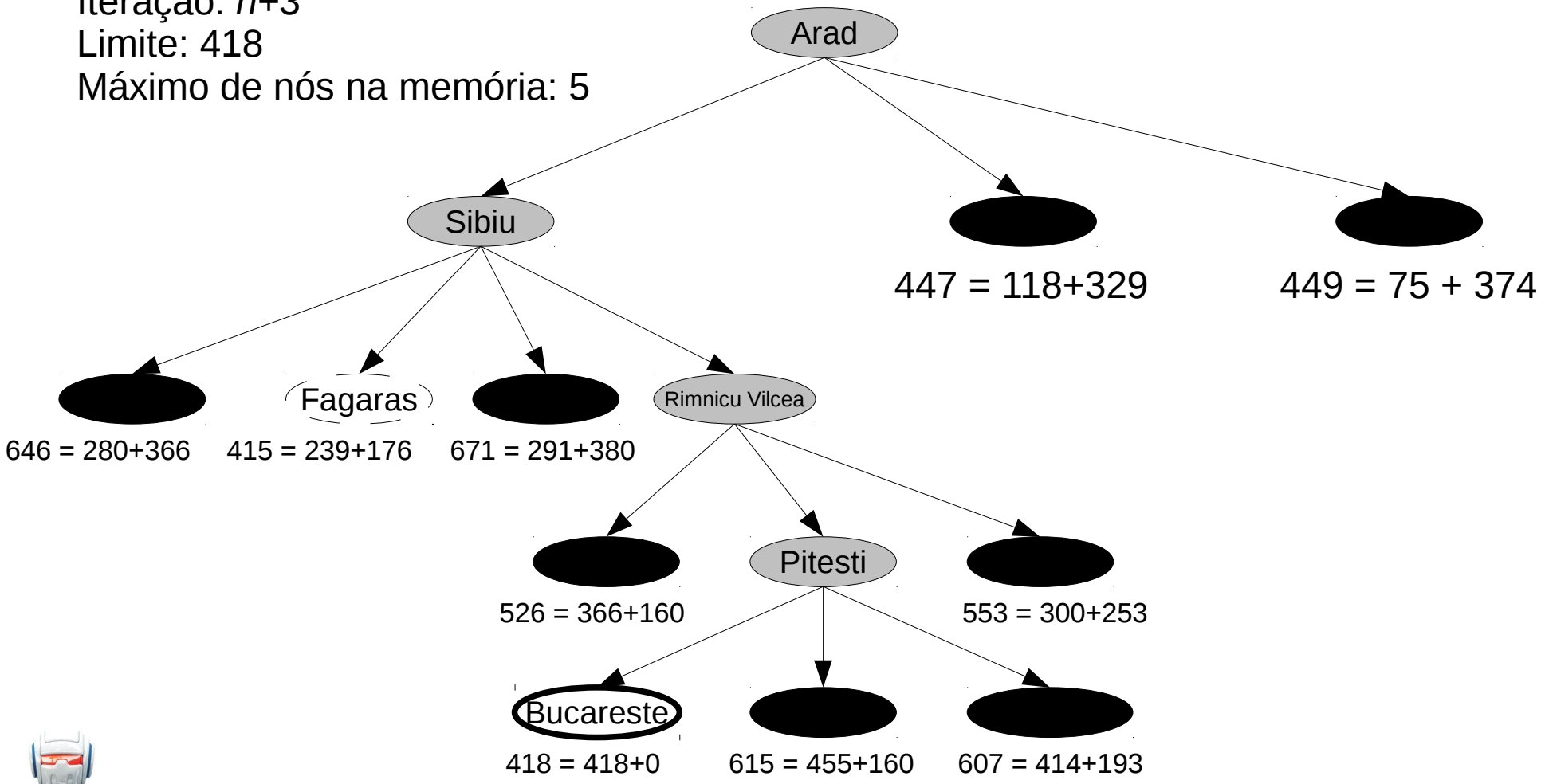
Viajante na Romênia

Iteração: *n+2*
Limite: 417
Máximo de nós na memória: 5



Viajante na Romênia

Iteração: *n+3*
Limite: 418
Máximo de nós na memória: 5



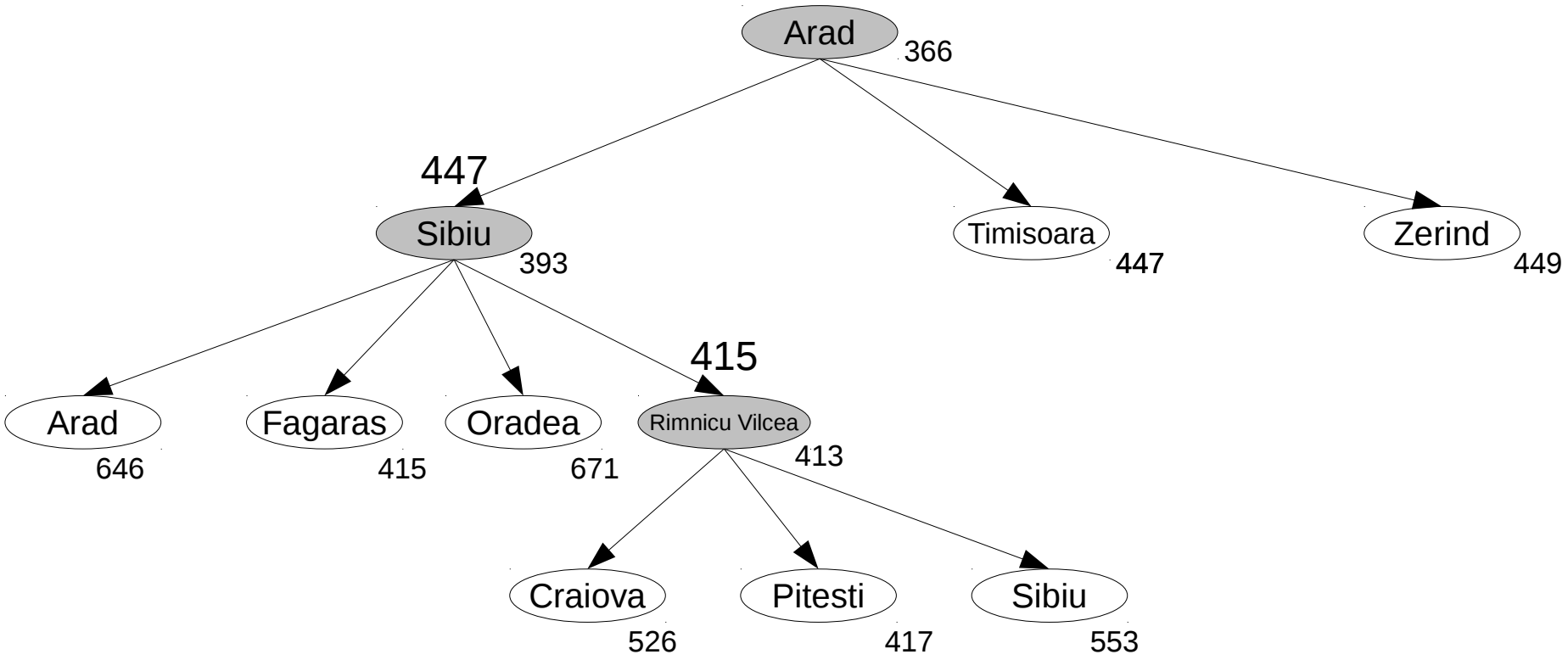
Análise

- Completude
  - Em geral não garante (*Loops* com custo 0)
    - Para resolver, cada caminho pode ter um custo mínimo ε
- Optimalidade
  - Garante se o custo mínimo do passo for ε
- Complexidade no Tempo
  - Não é simples de ser calculada, mas exponencial
- Complexidade no espaço
  - Linear $O(m)$

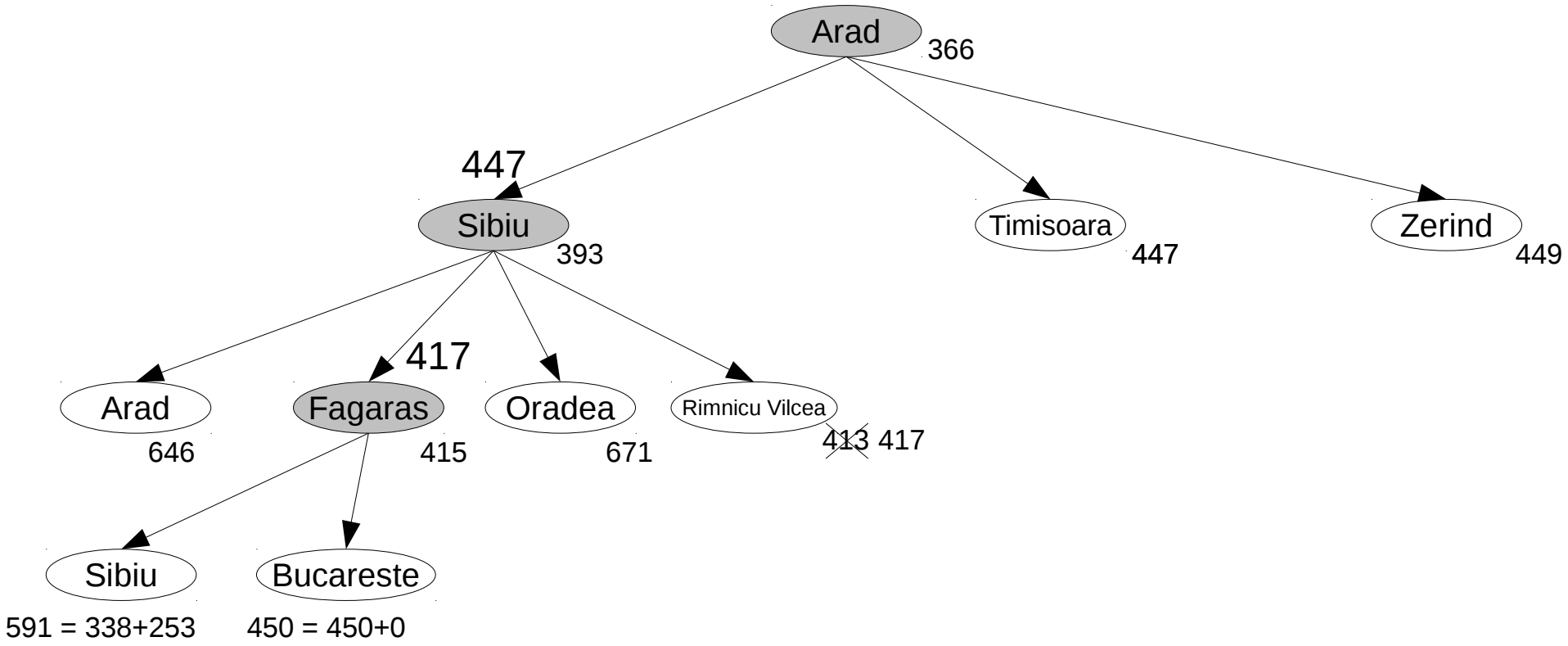
Buscas Heurísticas com Compromisso de Memória

- Busca pela Melhor Escolha Recursiva
  - Similar às buscas pela melhor escolha
    - Linear no espaço
    - Mais eficiente que IDA*
  - Não precisa manter uma fila de prioridades

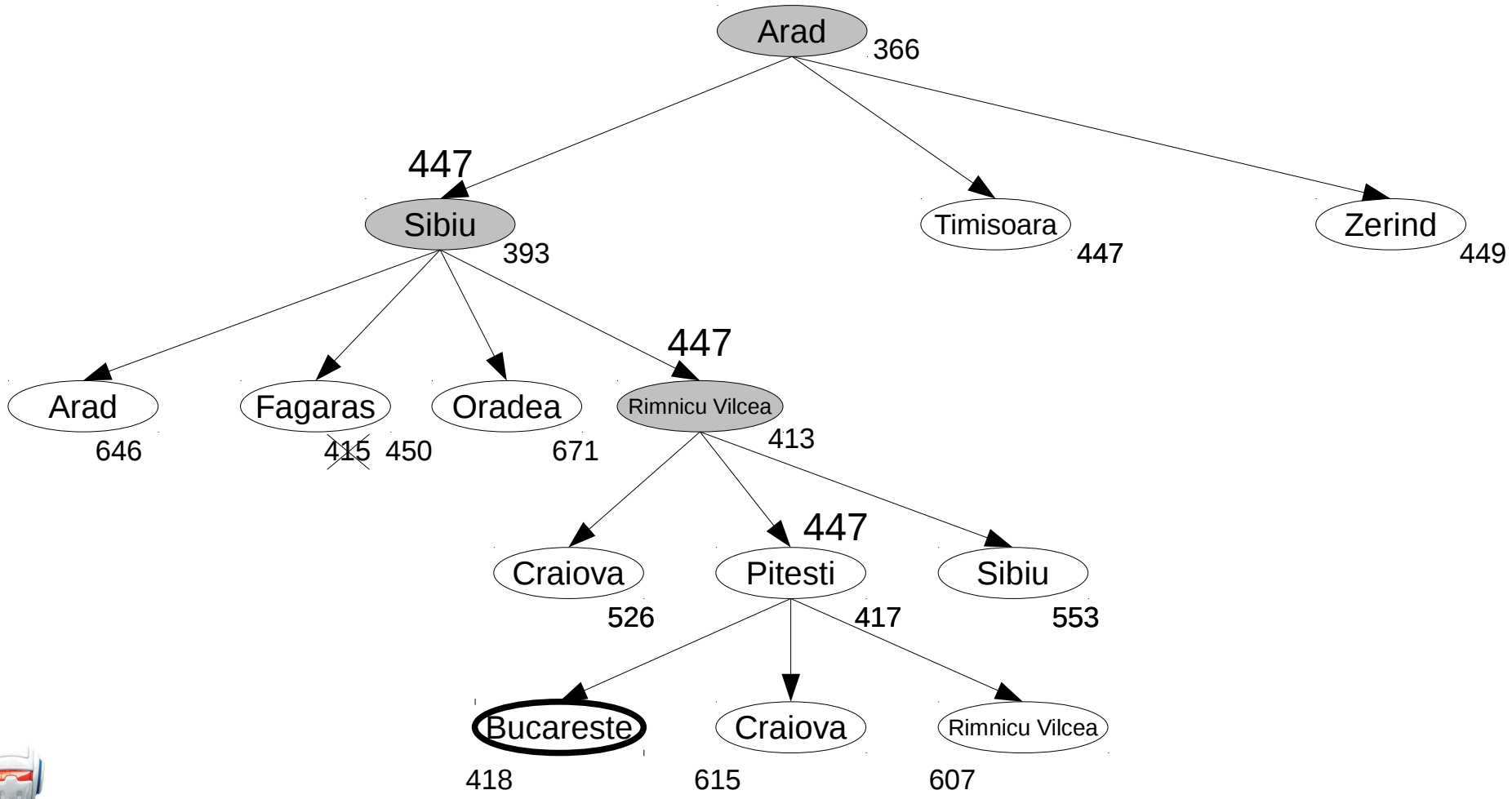
Viajante na Romênia



Viajante na Romênia



Viajante na Romênia



Análise

- Completude
  - Em geral não garante (*Loops* com custo 0)
    - Para resolver, cada caminho pode ter um custo mínimo ε
- Optimalidade
  - Garante se o custo mínimo do passo for ε
- Complexidade no Tempo
  - Não é simples de ser calculada, mas exponencial
  - Melhor do que o IDA*
- Complexidade no espaço
  - $O(bm)$

Outros Algoritmos

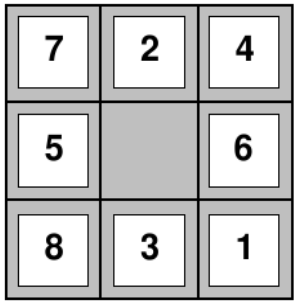
- MA*
  - *Memory Bounded A**
- SMA*
  - *Simplified MA**
  - Funcionamento básico
    - Executa o A* até memória acabar
      - Se acabar, então, apaga a pior folha
      - Guarda o limite como no RBFS para o caso de este ser o caminho ótimo

Busca Heurística

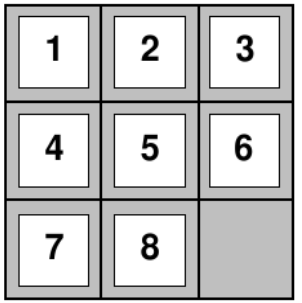
Funções Heurísticas

Revisitando o Quebra-cabeça de blocos Deslizantes

- Jogo com 8 blocos
  - Solução, em média, em 22 passos
  - Fator de Ramificação (b) em torno de 3
    - Se o bloco está no meio é 4; Se está no canto é 2; se está no lado é 3
  - Busca exaustiva: $3^{22} \approx 3{,}1 \times 10^{10}$
    - Evitando-se estados repetidos, chega-se a 181.440
- Jogo com 15 blocos
  - $(n+1)!/2 \approx 10 \times 10^{13}$
- Achar uma Heurística!



Estado Inicial



Estado Objetivo

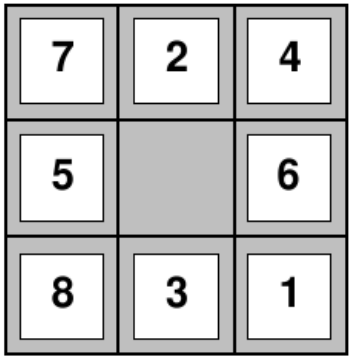
Heurísticas

- Para usar o algoritmo A*, a heurística deve ser admissível
  - Nunca sobre-estimar o valor de *h(n)* para qualquer estado *n*

- Candidatas históricas:
  - h_1 → Número de blocos fora de posição
    - Cada peça deslocada deve ser movida ao menos 1 vez
  - h_2 → Soma das Distâncias *Manhattan* de cada bloco para a sua posição final
    - Distância calculada como se estivesse andando pelas ruas da cidade

Blocos fora da Posição

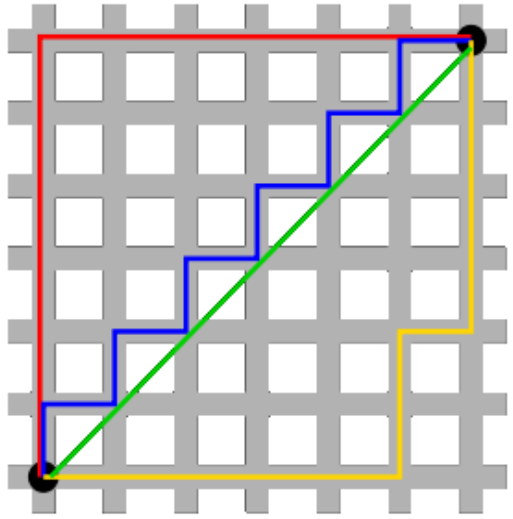
- Heurística h_1
  - Blocos fora de posição
    - $h_1 = 6$
      - O valor máximo de h_1 é 8
    - A Heurística é admissível
      - Bem menor que os 26 passos necessários para resolver esta instância



Distância Manhattan

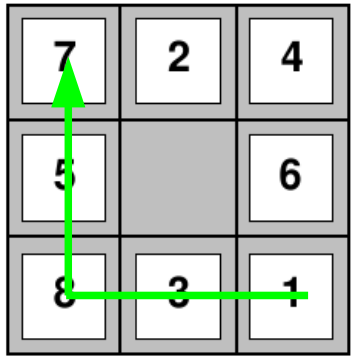
- Heurística h_2
  - Soma das Distâncias *Manhattan*
    - h_2 =
    - A Heurística é admissível?

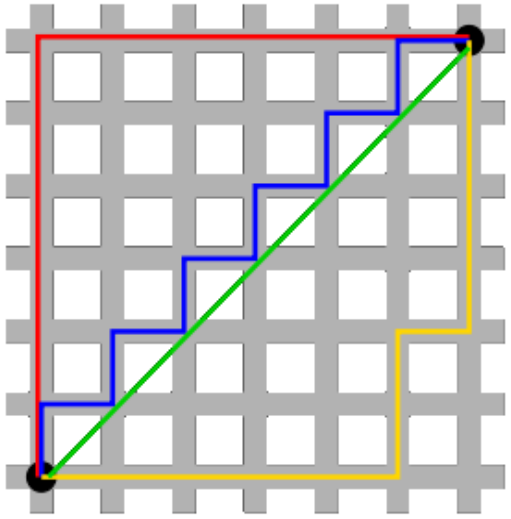
7	2	4
5		6
8	3	1



Distância Manhattan

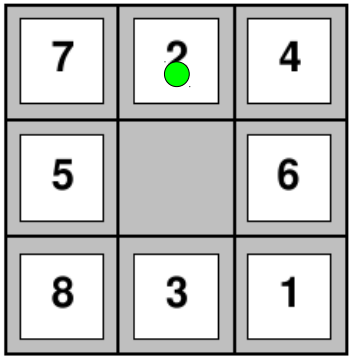
- Heurística h_2
  - Soma das Distâncias *Manhattan*
    - $h_2 = 4 +$
    - A Heurística é admissível?

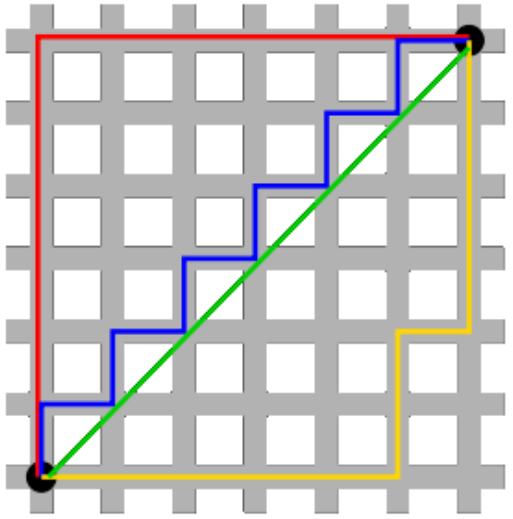




Distância Manhattan

- Heurística h_2
  - Soma das Distâncias *Manhattan*
    - $h_2 = 4 + 0 +$
    - A Heurística é admissível?

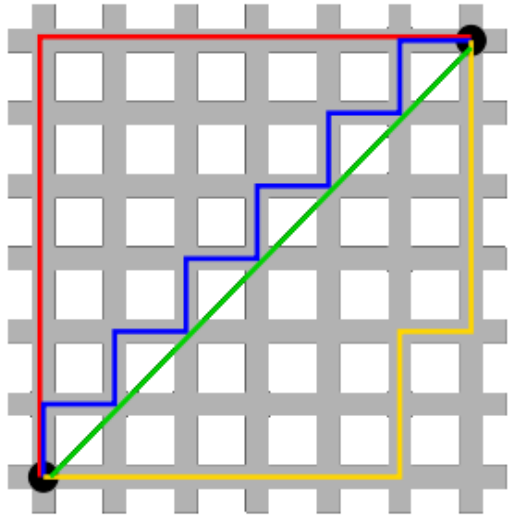




Distância Manhattan

- Heurística h_2
  - Soma das Distâncias *Manhattan*
    - $h_2 = 4 + 0 + 3 + 3 + 1 + 0 + 2 + 1 = 14$
    - A Heurística é admissível
      - Menor que os 26 passos necessários para resolver esta instância

7	2	4
5		6
8	3	1



Fator de Ramificação Efetivo

- Utilizado para Caracterizar a qualidade da heurística

$$N + 1 = 1 + (b^*) + (b^*)^2 + ... + (b^*)^d$$

  - N é o número de nós gerados pelo algoritmo A*
  - *d* é a profundidade da solução menos profunda
  - *b** é o fator de ramificação de uma árvore balanceada de profundidade *d* teria de forma a possuir *N + 1* nós
- Quando $b^* \approx 1$ problemas muito grandes podem ser resolvidos

$$\sum_{k=0}^{d} b^{*k} = \frac{b^{*d+1}}{b^* - 1} = N + 1$$

Fator de Ramificação Efetivo

- Como calcular
  - Executar experimentos simplificados pode dar uma ideia da utilidade da heurística

	Custo da Busca			Fator de Ramificação Efetivo		
d	**IDS**	**A*(h_1)**	**A*(h_2)**	**IDS**	**A*(h_1)**	**A*(h_2)**
2	10	6	6	2,45	1,79	1,79
4	112	13	12	2,87	1,48	1,45
6	680	20	18	2,73	1,34	1,30
8	6384	39	25	2,80	1,33	1,24
10	47127	93	39	2,79	1,38	1,22
12	3644035	227	73	2,78	1,42	1,24
14	-	539	113	-	1,44	1,23
16	-	1301	211	-	1,45	1,25
18	-	3056	363	-	1,46	1,26
20	-	7276	676	-	1,47	1,27
22	-	18094	1219	-	1,48	1,28
24	-	39135	1641	-	1,48	1,26

Dominância

- $h_2(n) > h_1(n)$ para qualquer nó n
  - h_2 domina h_1

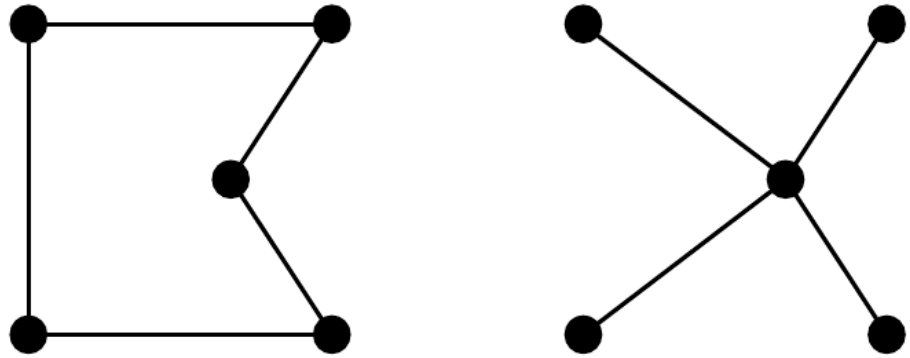
- A dominância é traduzida diretamente em eficiência
  - Todos os nós para o qual $h(n) < C^* - g(n)$ serão expandidos
    - Quanto maior o $h(n)$, menos nós serão expandidos, desde que $h(n)$ seja admissível
      - No limite, $h(n) = h^*(n)$

Problema Simplificado

- Heurísticas admissíveis podem ser derivadas de soluções exatas para uma versão simplificada do problema
  - *h1* → Derivada da simplificação onde não existe restrição de movimento. O bloco pode ser colocado diretamente na posição final.
  - *h2* → Derivada da simplificação onde uma peça pode ser movida para qualquer direção, mesmo se a posição destino estiver ocupada.
- É possível pois o custo do problema simplificado é sempre *menor* do que do problema real

Caixeiro-Viajante

- MSTs (*Minimum Spanning Trees*) podem ser usados como Heurística
  - Estimar o custo de completar a viagem, dado um caminho parcialmente completo
  - MST podem ser calculadas em *O(E log V)*
  - MSTs são o limite inferior para a distância de uma viagem (aberta)



Referências

- [ZAHAVI et al., 2007] Zahavi, Uzi; Felner, Ariel; Schaeffer, Johnathan; e Sturtevant Nathan. *Inconsistent Heuristics*. American Association for Artificial Intelligence, 2007.