Resolvendo problemas com busca

Inteligência Artificial

Aydano Pamponet Machado

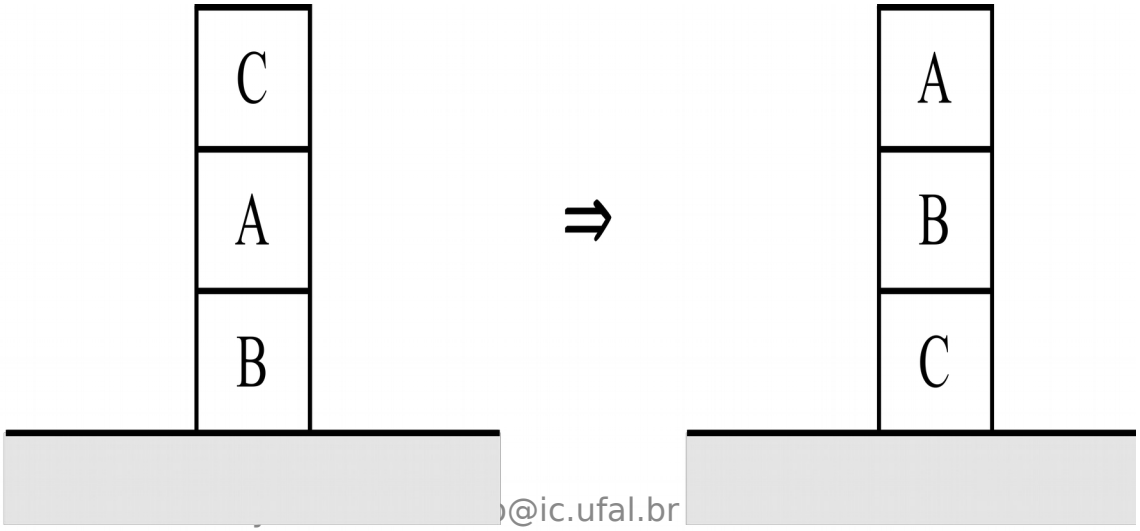
Universidade Federal de Alagoas - UFAL
Instituto de Computação – IC
aydano.machado@ic.ufal.br

Resolução de Problemas

- Resolução de Problemas em IA
  - Agente de resolução de problemas
- Métodos de Descrição do Espaço de Busca
  - Escolha de uma Representação
    - Espaço de Estados x Espaço do Problema
- O que é uma Solução para um Problema?
  - Conjunto, geralmente ordenado, de operadores que devem ser aplicados para transformar uma situação inicial numa situação final.
- Técnicas de Busca em Grafo
  - Busca não informada (busca cega)
  - Busca informada (busca heurística)

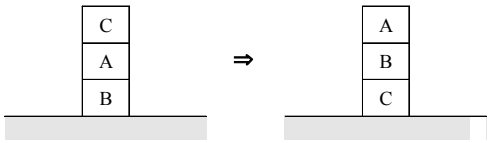
Resolução de Problemas em IA

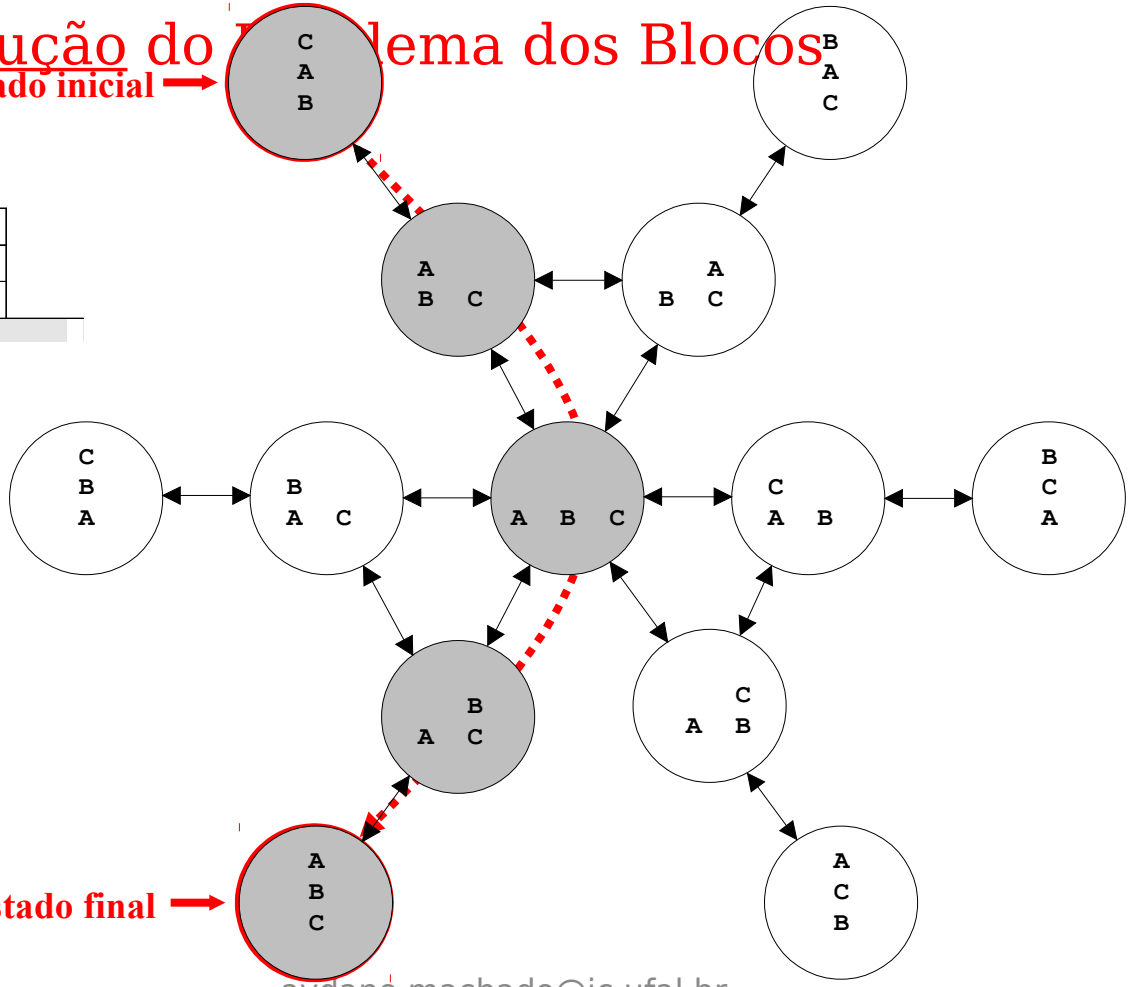
- **O que faz o S*olucionador de problemas?***
  - busca uma *sequência de ações* que leve a estados desejáveis (*objetivos*)
  - Exemplo dos blocos
    - Dada a configuração inicial dos blocos, use as regras e restrições dadas, para alcançar o estado objetivo, tal como indicados abaixo.



Resolução de problemas em IA

Uma Solução do Problema dos Blocos

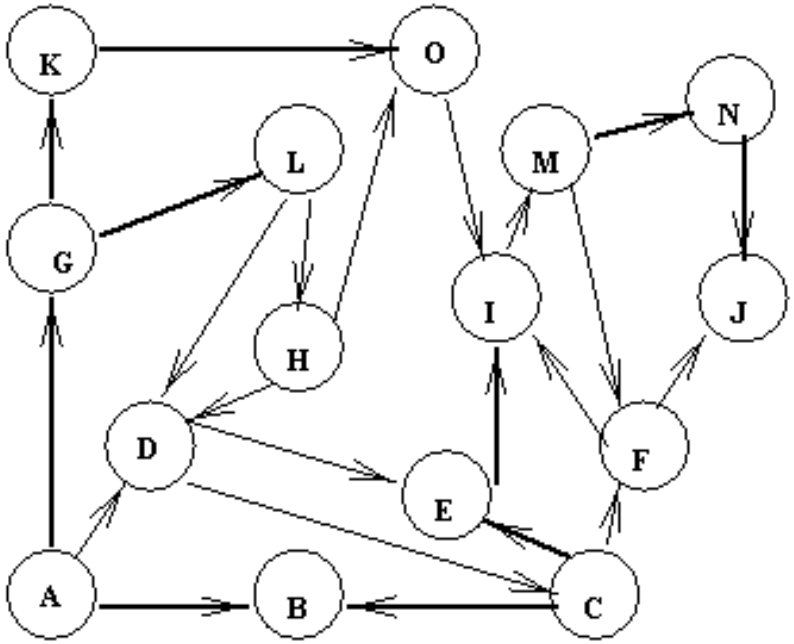




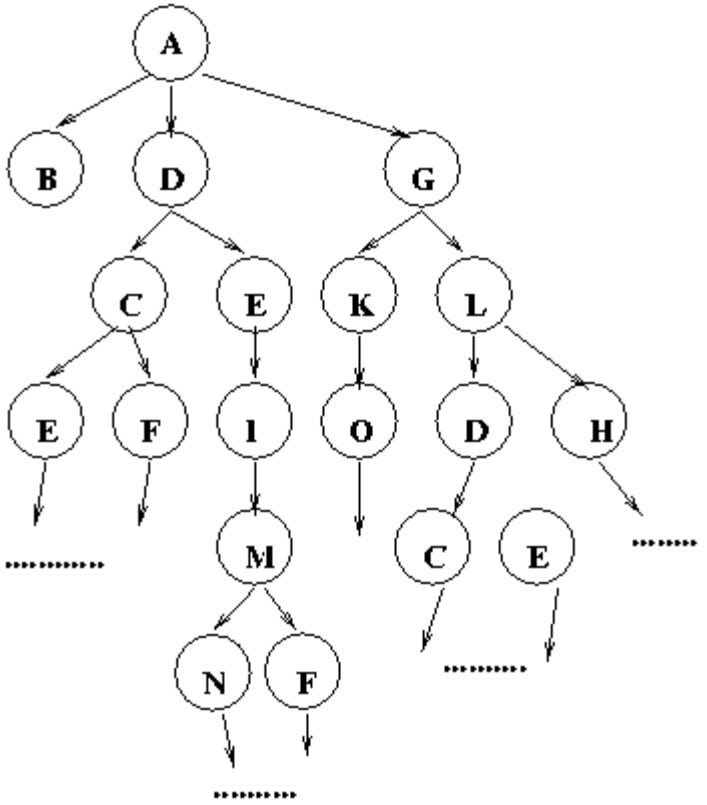
Espaço de Estados do Problema

Onde procurar a solução de um problema?

Problema de busca geral:

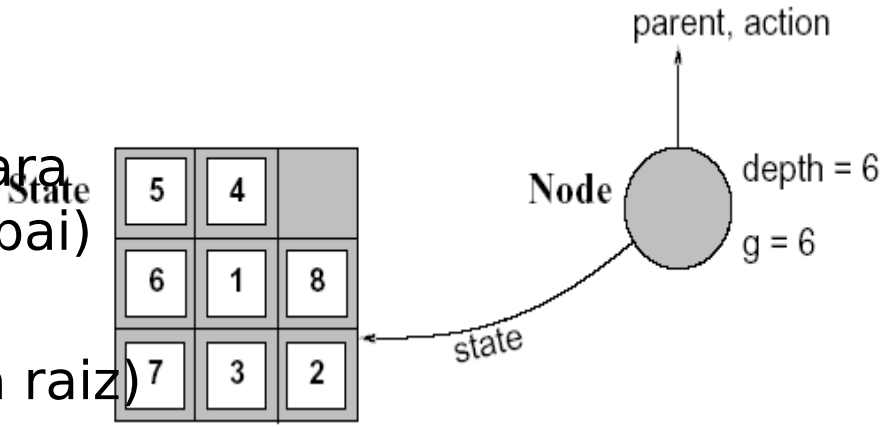


Representação de uma árvore de busca:



Busca em Espaço de Estados: Implementação

- Espaços de Estados
  - podem ser representados como uma árvore onde os estados são nós e as operações são arcos.
  - Um estado = representação de uma configuração física
- Os nós da árvore podem guardar mais informação do que apenas o estado:
  1. o estado correspondente
  2. o seu nó pai
  3. nós-filhos
  4. o operador aplicado para gerar o nó (a partir do pai)
  5. a profundidade do nó
  6. o custo do nó (desde a raiz)



Busca em Espaço de Estados

- Uma vez o problema bem formulado... o estado final deve ser “buscado”
- **Em outras palavras, deve-se usar um** método de busca **para saber a** ordem correta de aplicação dos operadores **que levará do estado inicial ao final**
- **Isto é feito por um processo de** geração **(de estados possíveis)** e teste **(para ver se o objetivo está entre eles)**
- **Uma vez a busca terminada com sucesso, é só** executar a solução **(= conjunto ordenado de operadores a aplicar)**
- Tipos de busca
  - **Busca exaustiva (busca não informada ou busca cega)**

Critérios de Avaliação das Estratégias de Busca

- Completa?
  - a estratégia **sempre** encontra uma solução quando existe alguma?
- Ótima?
  - a estratégia encontra **a melhor solução** quando existem soluções diferentes?
    - menor custo de caminho
- Custo de tempo?
  - quanto **tempo** gasta para encontrar uma solução?
- Custo de memória?
  - quanta **memória** é necessária para realizar a busca?

Resolução de problemas

Como procurar a solução de um problema?

Estratégia de busca

- **Busca exaustiva ou Cega**
  - A busca da solução começa a partir de um nó raiz
    - Se a raiz for um nó solução, o problema está resolvido
    - Senão: a raiz é desenvolvida e um dos seus sucessores é analisado...
    - O processo se repete até que um nó alvo seja alcançado
  - Não sabe qual o melhor nó da fronteira a ser expandido = menor custo de caminho desse nó até um ***nó final*** (objetivo) .
  - **Estratégias de Busca** (ordem de expansão dos nós):
    - caminhamento em largura
    - caminhamento em profundidade

    ... e suas variações.

Resolução de problemas

Como procurar a solução de um problema?

Busca Cega

- **Estratégias**
  1. Busca em largura
  2. Busca com custo uniforme
  3. Busca em profundidade
  4. Busca com profundidade limitada
  5. Busca com aprofundamento iterativo

Resolução de problemas

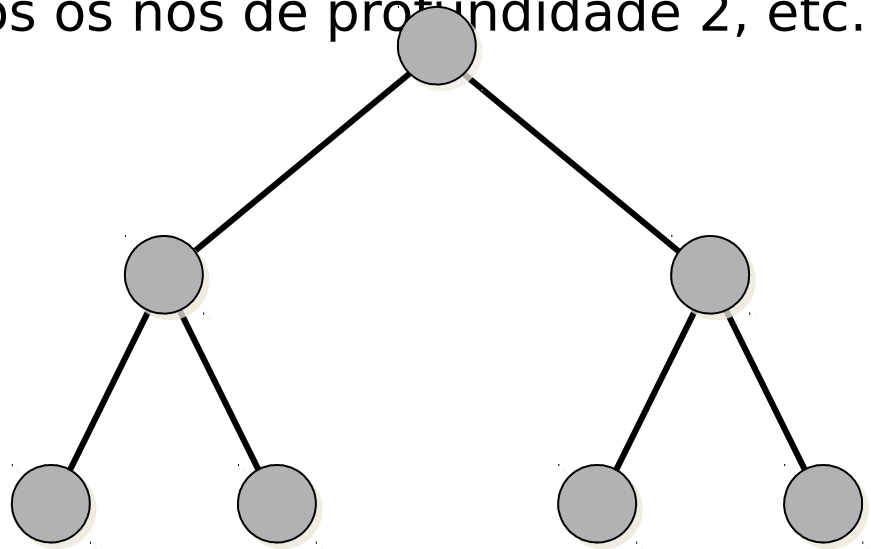
Como procurar a solução de um problema?

Busca Cega – *Busca em Largura*

Consiste em expandir sucessivamente os nós do mesmo nível de profundidade...Método nível por nível

Ordem de expansão dos nós:

1. Nó raiz
2. Todos os nós de profundidade 1
3. Todos os nós de profundidade 2, etc...



Estratégia de Busca Exaustiva

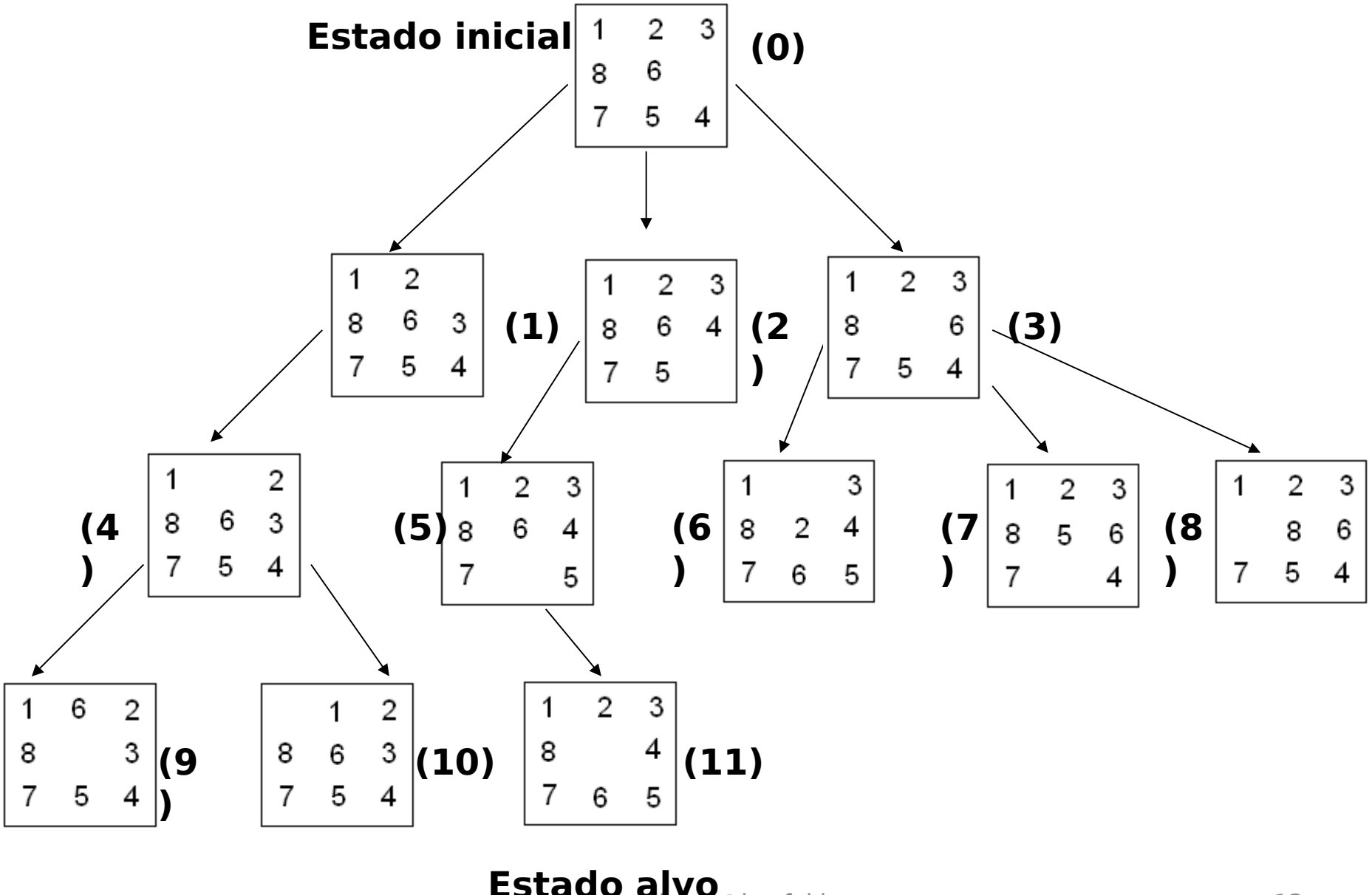
Resolução do Problema do Taquin em LARGURA

Busca em Largura

Jogo de 8 números

1	2	3
8		6
7	5	4

Resolução do problema dos 8 números - Busca em Largura



Busca em Largura

- Esta estratégia é *completa*
- É *ótima* ?
  - Sempre encontra a solução mais “rasa”
  - que nem sempre é a solução de menor **custo de caminho,** caso os operadores tenham valores diferentes
    - ex. ir para uma cidade D passando por B e C pode ser mais perto do que passando só por E
- Em outras palavras, é *ótima* se **custo de caminho** cresce com a profundidade do nó
  - O que ocorre quando todos os operadores têm o mesmo custo (=1)

Resolução de problemas

Como procurar a solução de um problema?

Busca Cega – *Busca em Largura*

Profundidade	Nós	Tempo	Memória
0	1	1 milissegundo	100 bytes
2	111	0.1 segundo	11 quilobytes
4	11111	11 segundos	1 megabytes
6	10^6	18 minutos	111 megabytes
8	10^8	31 horas	11 gigabytes
10	10^{10}	128 dias	1 terabyte
12	10^{12}	35 anos	111 terabytes
14	10^{14}	3500 anos	11111 terabytes

- **Custo de memória**:

  A fronteira do espaço de estados deve permanecer na memória, o que é um problema mais crucial do que o tempo de execução da busca

  Conclusão

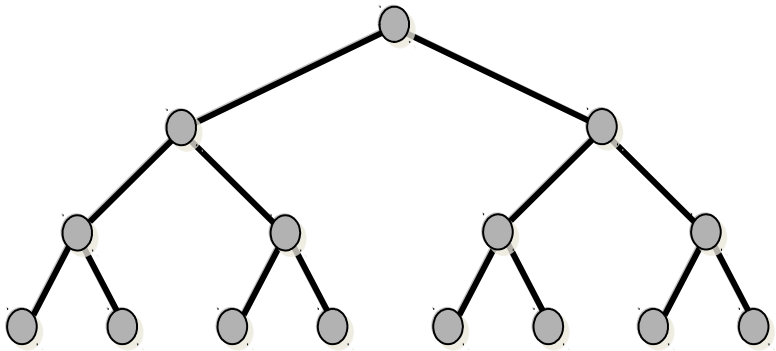
  Esta estratégia só dá bons resultados quando a profundidade da árvore de busca é pequena.

Resolução de problemas

Como procurar a solução de um problema?
Busca Cega – *Busca em Profundidade*

Ordem de expansão dos nós:

- sempre expande o nó no nível mais profundo da árvore:
  1. nó raiz
  2. primeiro nó de profundidade 1
  3. primeiro nó de profundidade 2, etc....
- Quando um nó final não é solução, o algoritmo volta para expandir nós mais superficiais.

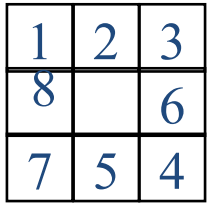


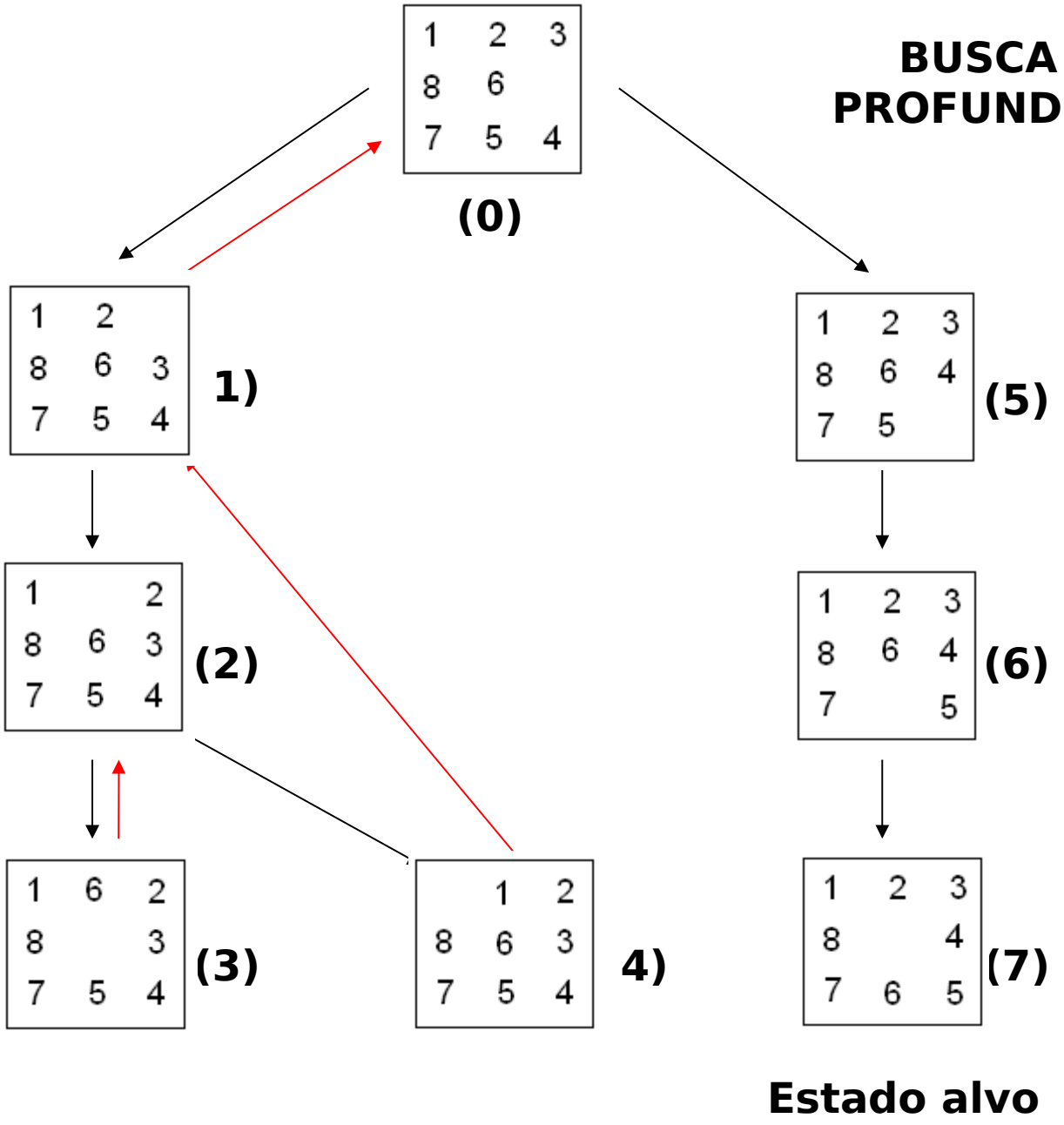
Estratégia de Busca Exaustiva

Resolução do Problema do Taquin em PROFUNDIDADE

Busca em Profundidade

Jogo de 8 números




BUSCA EM PROFUNDIDADE
1 2 3
8 6
7 5 4
(0)
1 2
8 6 3
7 5 4
1)
1 2 3
8 6 4
7 5
(5)
1 2
8 6 3
7 5 4
(2)
1 2 3
8 6 4
7 5
(6)
1 6 2
8 3
7 5 4
(3)
1 2
8 6 3
7 5 4
4)
1 2 3
8 4
7 6 5
(7)
Estado alvo

Resolução de problemas

Como procurar a solução de um problema?
Busca Cega – *Busca em Profundidade*

- Esta estratégia *não é completa nem é ótima*.
  - Esta estratégia deve ser evitada quando as árvores geradas são muito *profundas* ou geram *caminhos infinitos*.
  - Para problemas com várias soluções, esta estratégia pode ser bem mais rápida do que busca em largura.
- **Custo de memória**:
  - mantém na memória o caminho que está sendo expandido no momento, e os nós irmãos dos nós no caminho
  - $O(b.m)$: necessita armazenar apenas $b.m$ nós para um espaço de estados com fator de expansão b e profundidade m
    - m pode ser maior que d (profundidade da 1a. solução).
- **Custo de tempo**: $O(b^m)$, no pior caso.

Busca de Custo Uniforme

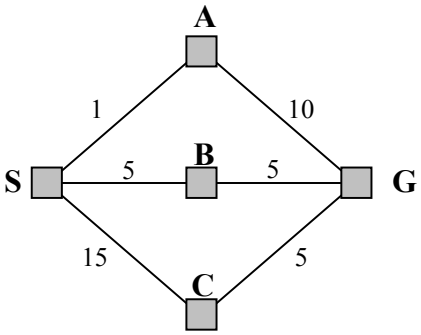
- Estende a busca em largura:
  - expande o nó da fronteira com menor custo de caminho até o momento
  - cada operador pode ter um custo associado diferente, medido pela função *g(n)* que dá o custo do caminho da origem ao nó n
- Na busca em largura: ***g(n) = profundidade (n)***
- Completa sempre e ótima se condição **g(sucessor(n)) ≥ g(n)** é satisfeita.
- Complexidade: teoricamente igual a busca em largura

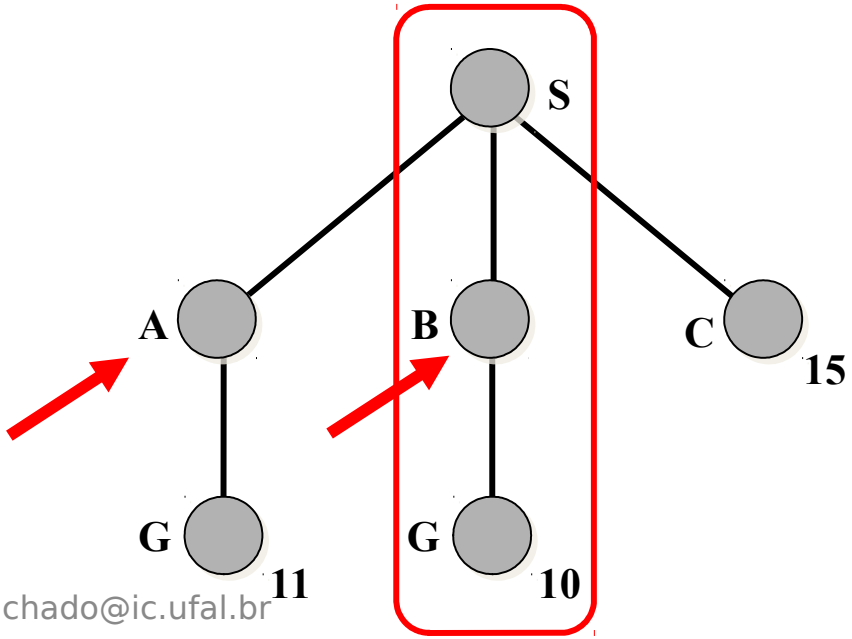
Resolução de problemas

Como procurar a solução de um problema?
Busca Cega – *Busca do Custo Uniforme*

Ordem de expansão dos nós:

1. nó raiz
2. nó de profundidade 1 com menor custo
3. segundo nó de profundidade 1 com menor custo, etc.





Resolução de problemas

Como procurar a solução de um problema?

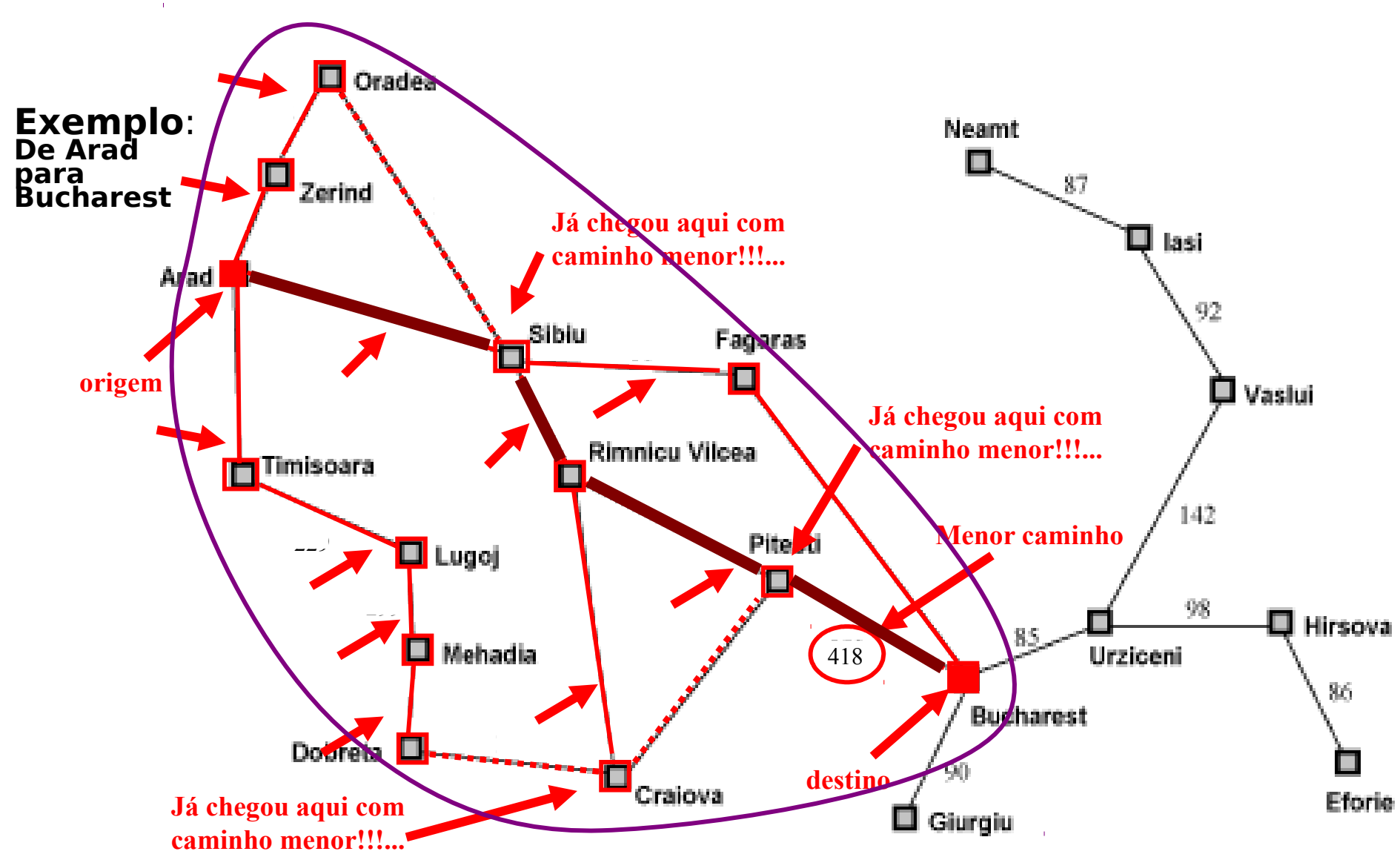
Busca Cega – *Busca do Custo Uniforme*

Exemplo:
De Arad para Bucharest

Resolução de problemas

Como procurar a solução de um problema?

Busca Cega – *Busca do Custo Uniforme*



Resolução de problemas

Como procurar a solução de um problema?
Busca Cega – *Busca com Profundidade Limitada*

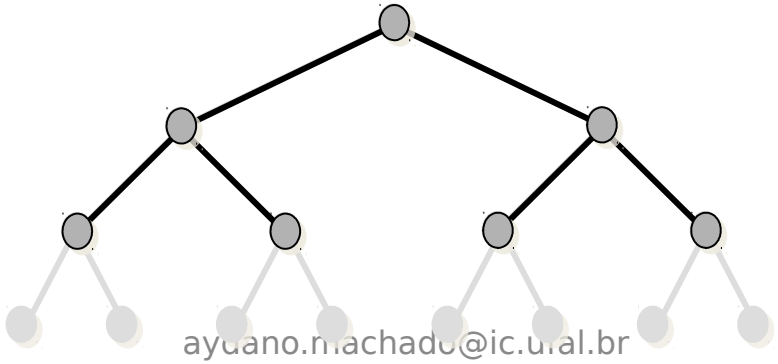
- Evita o problema de caminhos muito longos ou infinitos impondo um limite máximo de profundidade para os caminhos gerados

- Esse limite deve assegurar a completude da busca

  $l \geq d$, onde l é o limite de profundidade e d é a profundidade da primeira solução do problema

- Não é ótima (= busca em profundidade)

No exemplo anterior, se profundidade = 2 ...



Resolução de problemas

Como procurar a solução de um problema?
Busca Cega – *Busca com Profundidade Limitada*

- **Complexidade de tempo e de espaço**:
  - necessita armazenar apenas $b.l$ nós para um espaço de estados com fator expansão b
  - similar a da busca em profundidade: $O(b^l)$, onde l é o limite de profundidade

- **Problema**:
  - em geral, é difícil de prever um bom limite de profundidade antes de se buscar uma solução do problema

Resolução de problemas

Como procurar a solução de um problema?
Busca Cega – *Busca com Aprofundamento Iterativo*

- Tenta limites com valores crescentes, partindo de zero, até encontrar a primeira solução
  - fixa profundidade = *i*, executa busca
  - se não chegou a um objetivo, recomeça busca com profundidade = *i + n* (*n* qualquer)

- se *i* = 1 e *n* = 1, é igual à Busca em Largura
- Combina as vantagens de *busca em largura* com *busca em profundidade*
- É *ótima e completa*.

Resolução de problemas

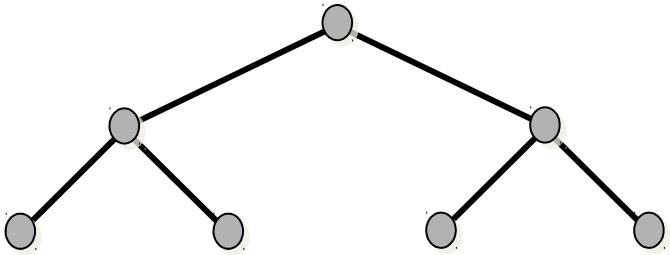
Como procurar a solução de um problema?

Busca Cega – *Busca com Aprofundamento Iterativo*

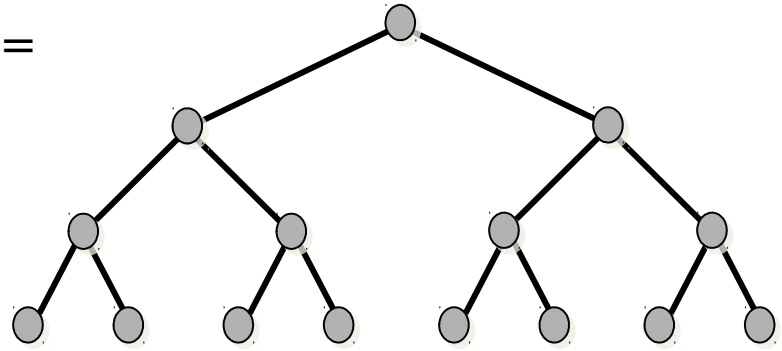
Limite = 0

Limite = 1

Limite = 2



Limite = 3



:

Resolução de problemas

Como procurar a solução de um problema?
Busca Cega – *Busca com Aprofundamento Iterativo*

- **Complexidade**
  - *Fator de expansão*: $1 + b + b^2 + \ldots + b^{d-2} + b^{d-1} + b^d$
  - Nº expansões: $(d+1).1 + (d).b + (d-1).b^2 + \ldots + 3.b^{d-2} + 2.b^{d-1} + 1.b^d$

- **Custo de memória**: $b.d$
  - necessita armazenar apenas $b.d$ nós para um espaço de estados com fator de expansão b e limite de profundidade d
- **Custo de tempo**: $O(b^d)$
  - Apresenta bons resultados quando o espaço de estados é grande

    e a profundidade desconhecida

Resolução de problemas

Como procurar a solução de um problema?

Busca Cega – *Direção da busca*

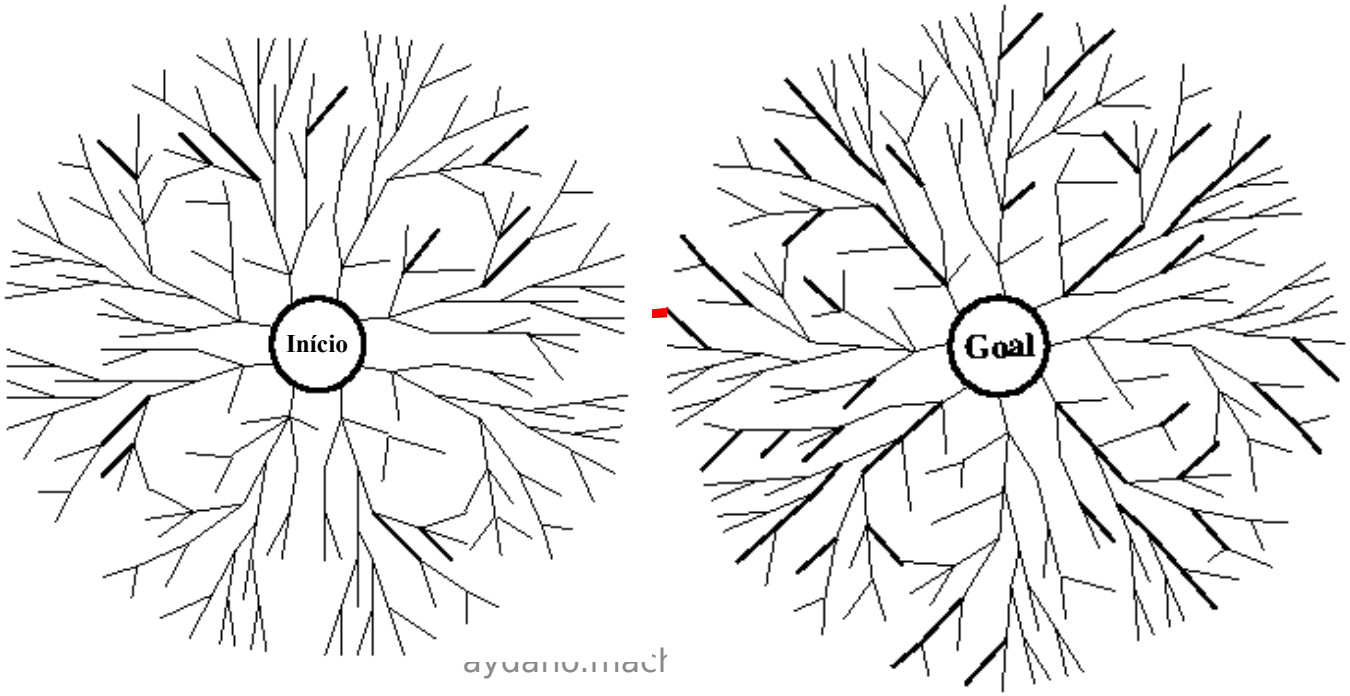
- **Estratégia**:
  - Para frente, a partir do estado inicial
  - A busca pára quando um estado final é encontrado.

- **Estratégia**:
  - Para trás, a partir do estado final desejado
  - A busca pára quando o estado inicial é encontrado.

Resolução de problemas

Como procurar a solução de um problema?
Busca Cega – *Direção da busca* – *Busca bidirecional*

- **Estratégia**:
  - Para frente, a partir do nó inicial, e para trás, a partir do nó final (objetivo)
  - A busca pára quando os dois processos geram um mesmo estado intermediário.



- É possível utilizar estratégias diferentes (largura, profundidade, etc.) em cada direção da busca.

- Busca para trás
  - gera *predecessores* do nó final
  - Se os operadores são reversíveis: conjunto de predecessores do nó = conjunto de sucessores do nó
- **Custo de tempo**:
  - se o fator de expansão dos nós é *b* nas duas direções e a profundidade do último nó gerado é *d*, cada processo busca até metade da profundidade do nó.
  - $O(2b^{d/2}) = O(b^{d/2})$
- **Custo de memória**: $O(b^{d/2})$

Resolução de problemas

Como procurar a solução de um problema?
Busca Cega – *Busca Bidirecional*

- **Problemas**:
  - Existem muitos estados finais (objetivos) no problema
    - conjunto específico de estados finais
      - aplica-se a função predecessor ao conjunto de estados
    - descrição desse conjunto (propriedade abstrata)
      - Ex.: Como determinar exatamente todos os estados que precedem um estado de xeque-mate?
  - Necessita-se de operadores *reversíveis*, porém...
    - esses operadores podem gerar *árvores infinitas*!
      - Ex.: Cálculo de rota, missionários - canibais
  - Pode ser caro verificar se cada novo estado gerado aparece na outra metade do espaço de estados
  - É necessário evitar a geração de estados repetidos

Resolução de problemas

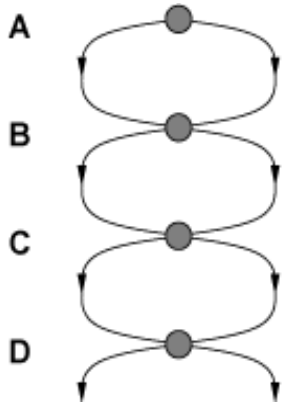
Como procurar a solução de um problema?
Busca Cega – *Evitando estados repetidos*

- **Problema geral em busca:**
  - expandir estados presentes em caminhos já explorados

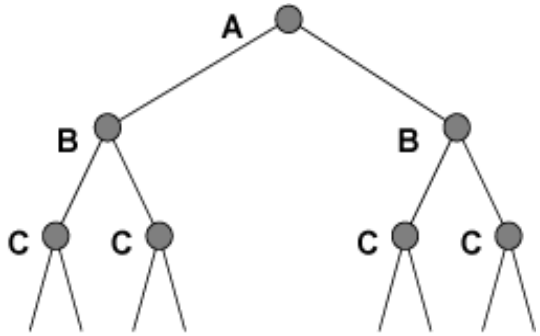
- É inevitável quando existe operadores reversíveis
  - Ex.: Encontrar rotas, canibais e missionários
  - a árvore de busca é potencialmente infinita
- **Idéia**:
  - **podar** ("*prune*") estados repetidos, para gerar apenas a parte da árvore que corresponde ao grafo do espaço de estados (que é finito!)
  - mesmo quando esta árvore é finita ... evitar estados repetidos pode reduzir exponencialmente o custo da busca

Ex.:

Espaço de estados



Árvore de busca



($m + 1$) estados no espaço $\Rightarrow 2^m$ caminhos na árvore

Questão:

- Como evitar expandir estados presentes em caminhos já explorados sem aumentar de forma considerável o custo computacional?

Soluções:

1. Não retornar ao estado “pai”
   - função que rejeita geração de sucessor igual ao pai
2. Não criar caminhos com ciclos
   - não gerar sucessores para qualquer estado que já apareceu no caminho sendo expandido
3. Não gerar qualquer estado que já tenha sido criado antes (em qualquer ramo)
   - requer que todos os estados gerados permaneçam na memória
   - custo de memória: $O(b^d)$
   - pode ser implementado mais eficientemente com tabelas *hash*

Resolução de problemas

Como procurar a solução de um problema?
Busca Cega – *Considerações finais*

Comparação entre os métodos:

Critério	Largura	Custo Uniforme	Profundidade	Profundidade Limitada	Aprofundamento Iterativo	Bidirecional
Tempo	b^d	b^d	b^m	b^l	b^d	$b^{d/2}$
Espaço	b^d	b^d	bm	bl	bd	$b^{d/2}$
Otima?	Sim	Sim	Não	Não	Sim	Sim
Completa?	Sim	Sim	Não	Sim, se $l \geq d$	Sim	Sim

b = fator de expansão do problema
d = nível da primeira solução para o problema
l = limite de profundidade
m = profundidade máxima

Problema:

- Custo de armazenamento e verificação ✕ Custo extra de busca

Solução:

- Depende do problema
- Quanto mais “loops”, mais vantagem em evitá-los!