

# Métodos de Busca



# Métodos de Busca



- Um dos obstáculos mais difíceis de superar quando são aplicadas técnicas de IA nos problemas do mundo real é a magnitude e complexidade da maioria das situações.
- Nos primeiros anos da pesquisa em IA, desenvolver bons métodos de busca era o principal objetivo.
- Os pesquisadores acreditavam que a busca é a base da resolução de problemas, que é um ingrediente crucial da inteligência.

# Métodos de Busca



## Terminologia

**Estado** — situação relevante para o problema.

**Estado Inicial** — Estado onde o agente se encontra no início.

**Operadores ou ações** — conjunto de ações disponíveis ao agente que permitem ir de um estado para outro.

**Espaço de estados** — conjunto de todos os estados alcançáveis a partir do estado inicial a partir da aplicação de uma sequência de ações.

**Caminho** — uma sequência de ações levando de um estado a outro.

# Métodos de Busca



## Terminologia

- **Teste de objetivo** — teste aplicado pelo agente para verificar se chegou a um estado objetivo.
- **Custo do caminho** — a soma total dos custos da ações individuais ao longo de um caminho, denotada pela função  $g$ .
- **Solução** — um caminho que parte do estado inicial e leva a um estado objetivo.

# Métodos de Busca

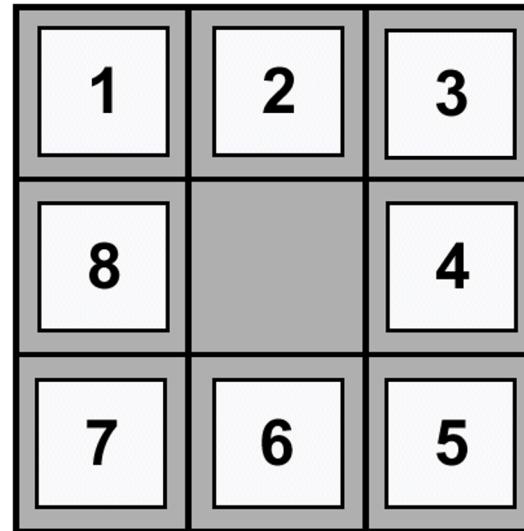
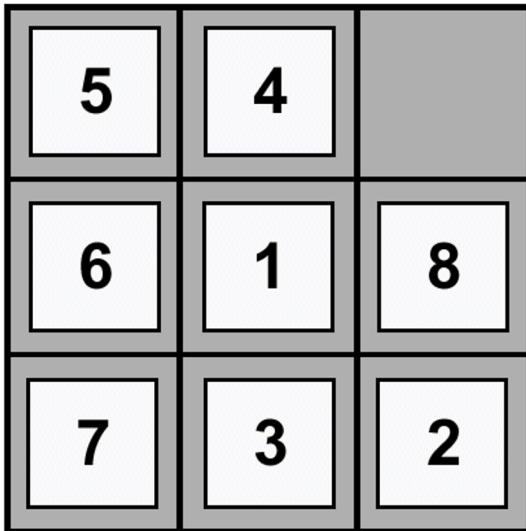


## Medida de desempenho

- **Completude** — a estratégia garante achar uma solução se existir?
- **Complexidade no tempo** — quanto demorou a achar a solução ?
- **Complexidade no espaço** — quanta memória foi necessária para achar a solução ?
- **Optimalidade** — a solução encontrada foi a melhor?

# Métodos de Busca

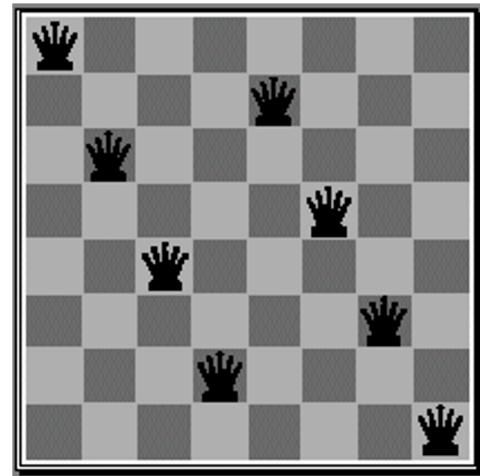
## Exemplo: Quebra-cabeça



- **Estados:** descrições das localizações das 8 peças.
- **Operadores:** mover o espaço para cima, baixo, e lados.
- **Teste de objetivo:** verificar se chegou na configuração da direita.
- **Custo do caminho:** número de movimentos.

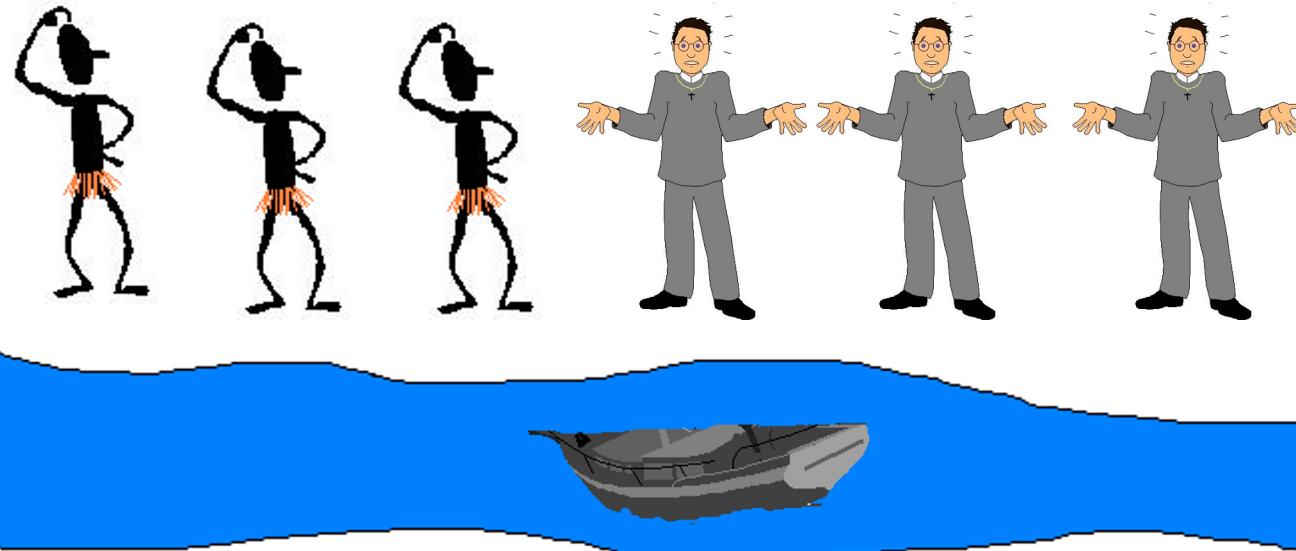
# Métodos de Busca

## Exemplo: 8 rainhas



- **Estados:** Tabuleiro 8 rainhas, uma em cada coluna.
- **Operadores:** mover uma rainha atacada para outro quadrado na mesma coluna.
- **Teste de objetivo:** 8 rainhas no tabuleiro sem se atacarem.
- **Custo do caminho:** número de movimentos.

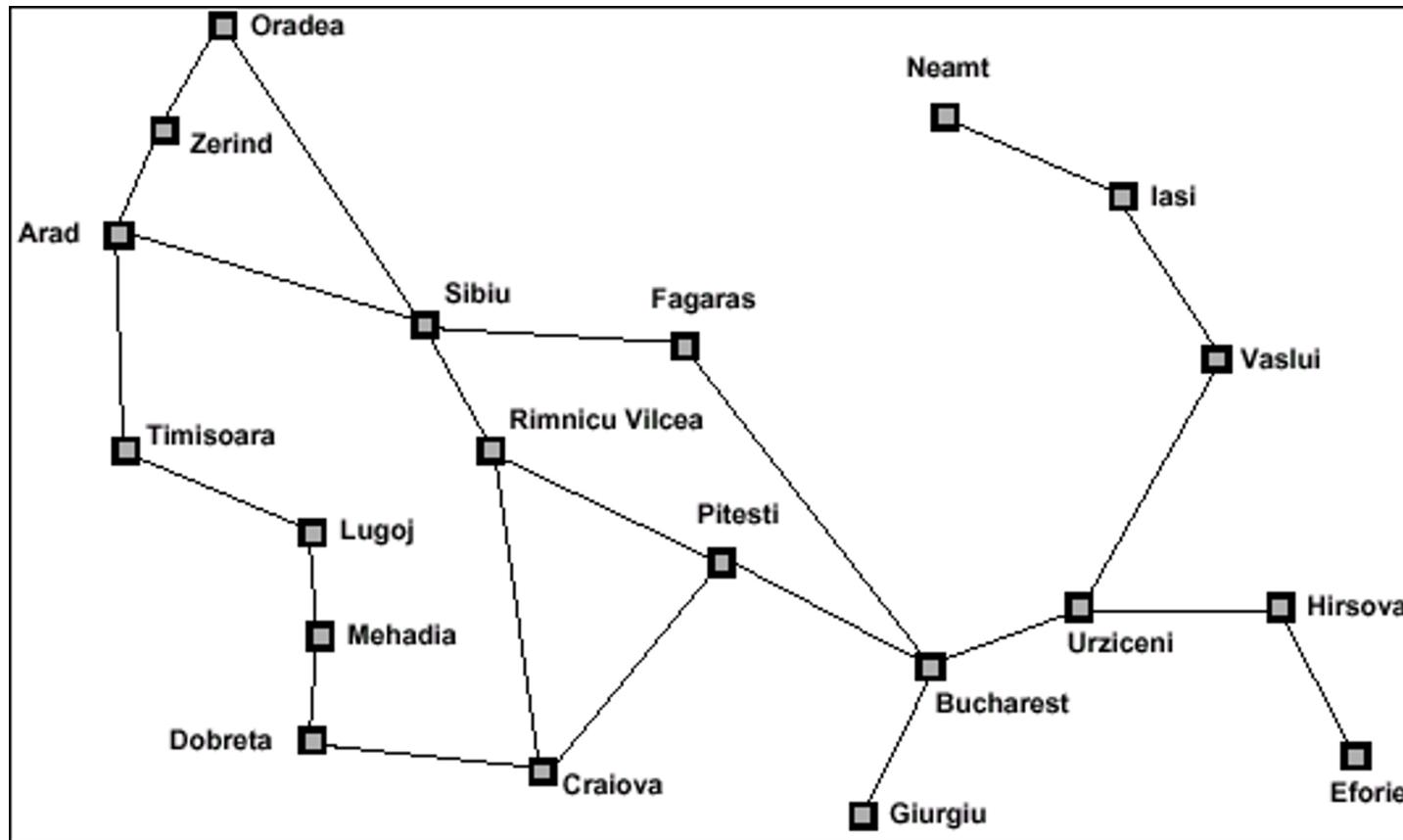
# Métodos de Busca Exemplo: missionários e canibais



- **Estados:** Uma tripla com o número de canibais, missionários e barcos na margem inicial do rio. O estado inicial é (3,3,1).
- **Operadores:** retirar ou adicionar um missionário ou um canibal ou dois missionários ou dois canibais ou um missionário e um canibal.
- **Teste de objetivo:** verificar se é o estado (0,0,0).
- **Custo do caminho:** número de travessias.

# Métodos de Busca

## Estratégias: problema exemplo



- Sair de Arad e chegar a Bucareste.

# Métodos de Busca

## Estratégias: Tipos

- **Busca cega ou não informada**

Busca em amplitude

Busca de custo uniforme

Busca em profundidade

Busca em profundidade limitada

Busca em profundidade iterativa  
bidirecional

- **Busca informada ou heurística**

- **Busca Local**

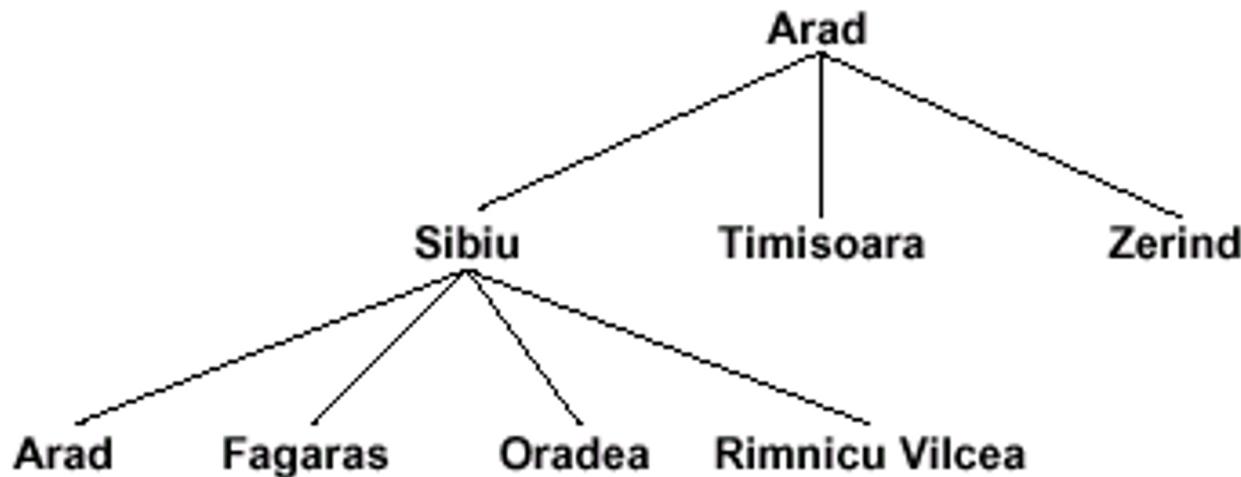
Subida da encosta  
Tempera simulada  
Algoritmo Genético

Gulosa  
 $A^*$

# Métodos de Busca

## Estratégias

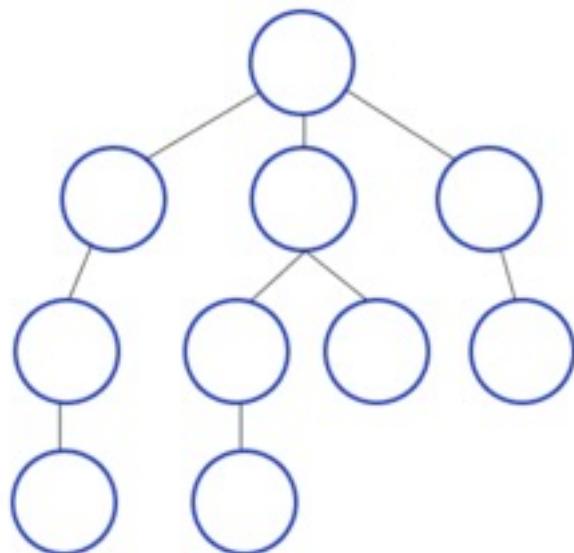
- O funcionamento de um método de busca pode ser visto como um processo de expansão de uma árvore.



# Métodos de Busca

## Estratégias: Busca em Amplitude

- Nesta estratégia o nó raiz é expandido. O nós gerados são expandidos em seguida, e assim por diante.
- A busca em amplitude examina todos os estados no mesmo nível antes de prosseguir no próximo nível.



# Métodos de Busca

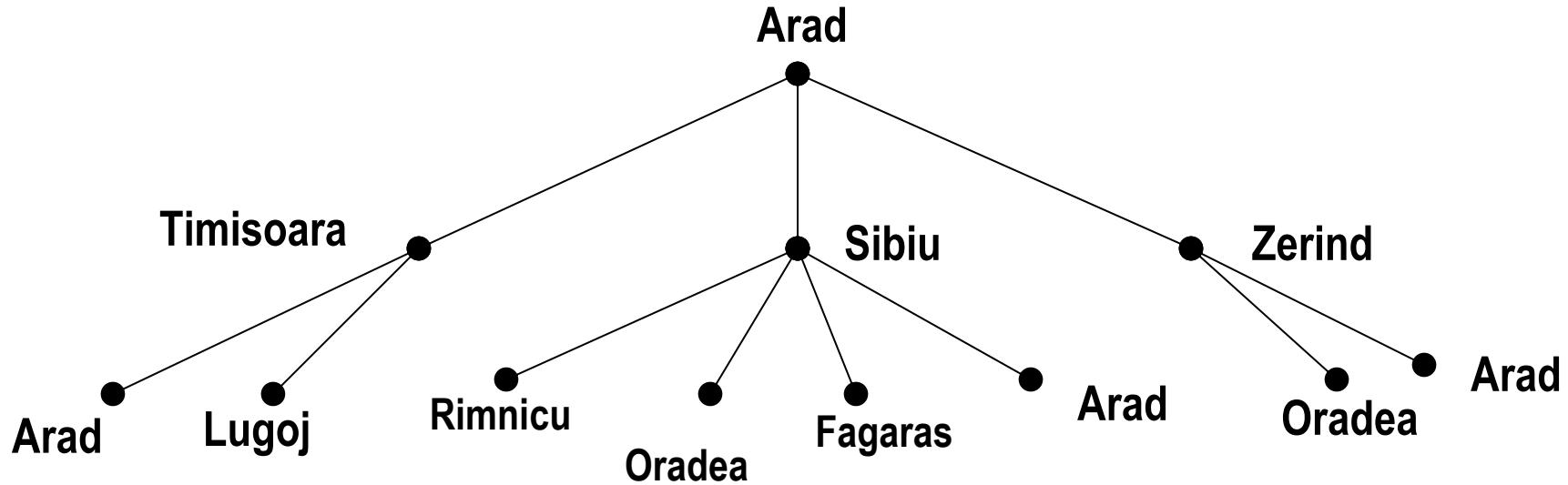
## Estratégias: Busca em Amplitude

- Nesta estratégia o nó raiz é expandido. O nós gerados são expandidos em seguida, e assim por diante.
- A busca em amplitude examina todos os estados no mesmo nível antes de prosseguir no próximo nível.

```
Function busca-Amplitude(problema) returns solução ou falha
    nós := cria_fila(estado_inicial(problema))
    loop
        if nós vazio then return falha
        nó := obtem_primeiro(nós)
        if (estado_objetivo(problema,nó)) then return nó
        nós := adiciona_no_fim(nós, expande(nó,operadores(problema)))
    end
```

# Métodos de Busca

## Estratégias: Busca em Amplitude



- Para analisar a busca em amplitude é preciso conhecer o fator médio de ramificação:  $b$  (branching factor).
- Se a solução de um problema possuir profundidade  $d$ , então o número de nós expandidos antes de achar a solução é:

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

# Métodos de Busca

## Estratégias: Busca em Amplitude

Depth	Nodes	Time	Memory
0	1	1 millisecond	100 bytes
2	111	.1 seconds	11 kilobytes
4	11,111	11 seconds	1 megabyte
6	$10^6$	18 minutes	111 megabytes
8	$10^8$	31 hours	11 gigabytes
10	$10^{10}$	128 days	1 terabyte
12	$10^{12}$	35 years	111 terabytes
14	$10^{14}$	3500 years	11,111 terabytes

$$b = 10; 1000/s; 100 \text{ bytes por nó}$$

- Vantagem: acha a solução com menor número de conexões.

# Métodos de Busca

**Estratégias: Custo uniforme (Dijkstra)**

# Métodos de Busca

## Estratégias: Custo uniforme

- A busca em amplitude acha a solução mais rasa mas não necessariamente a de menor custo.
- A busca de menor custo é uma modificação da busca em amplitude de a expandir sempre o nó de menor custo, medido pela função  $g$  desde que a condição abaixo seja satisfeita:

$$g(\text{sucessor}(n)) \geq g(n)$$

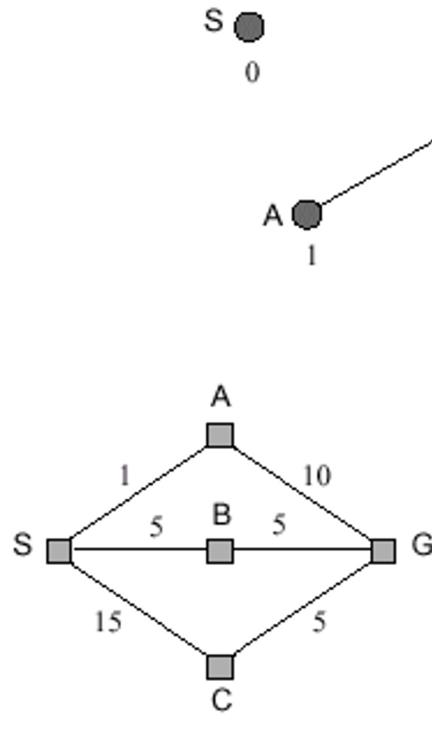
# Métodos de Busca

## Estratégias: Custo uniforme

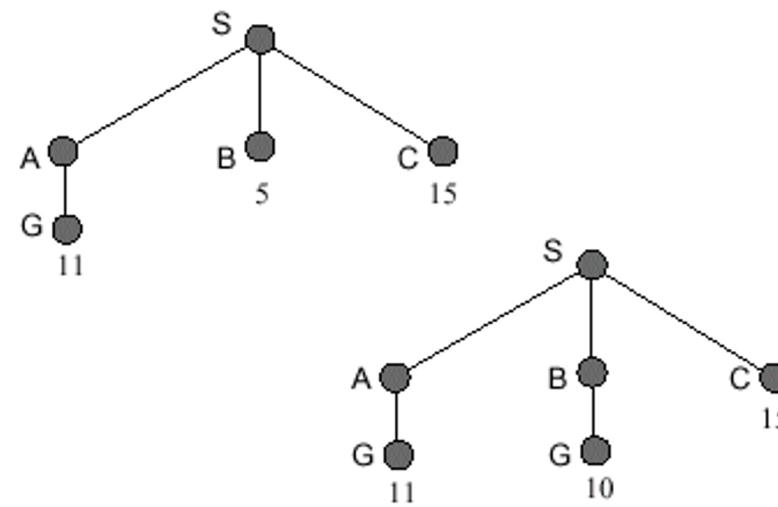
```
function UNIFORM-COST-SEARCH(problem) returns a solution, or failure
    node ← a node with STATE = problem.INITIAL-STATE, PATH-COST = 0
    frontier ← a priority queue ordered by PATH-COST, with node as the only element
    explored ← an empty set
    loop do
        if EMPTY?(frontier ) then return failure
        node ← POP(frontier ) /* chooses the lowest-cost node in frontier */
        if problem.GOAL-TEST(node.STATE) then return SOLUTION(node)
        add node.STATE to explored
        for each action in problem.ACTIONS(node.STATE) do
            child ← CHILD-NODE(problem, node, action)
            if child.STATE is not in explored or frontier then
                frontier ← INSERT(child,frontier )
            else if child.STATE is in frontier with higher PATH-COST then
                replace that frontier node with child
```

# Métodos de Busca

## Estratégias: Custo uniforme



(a)



(b)

(a) espaço de estado

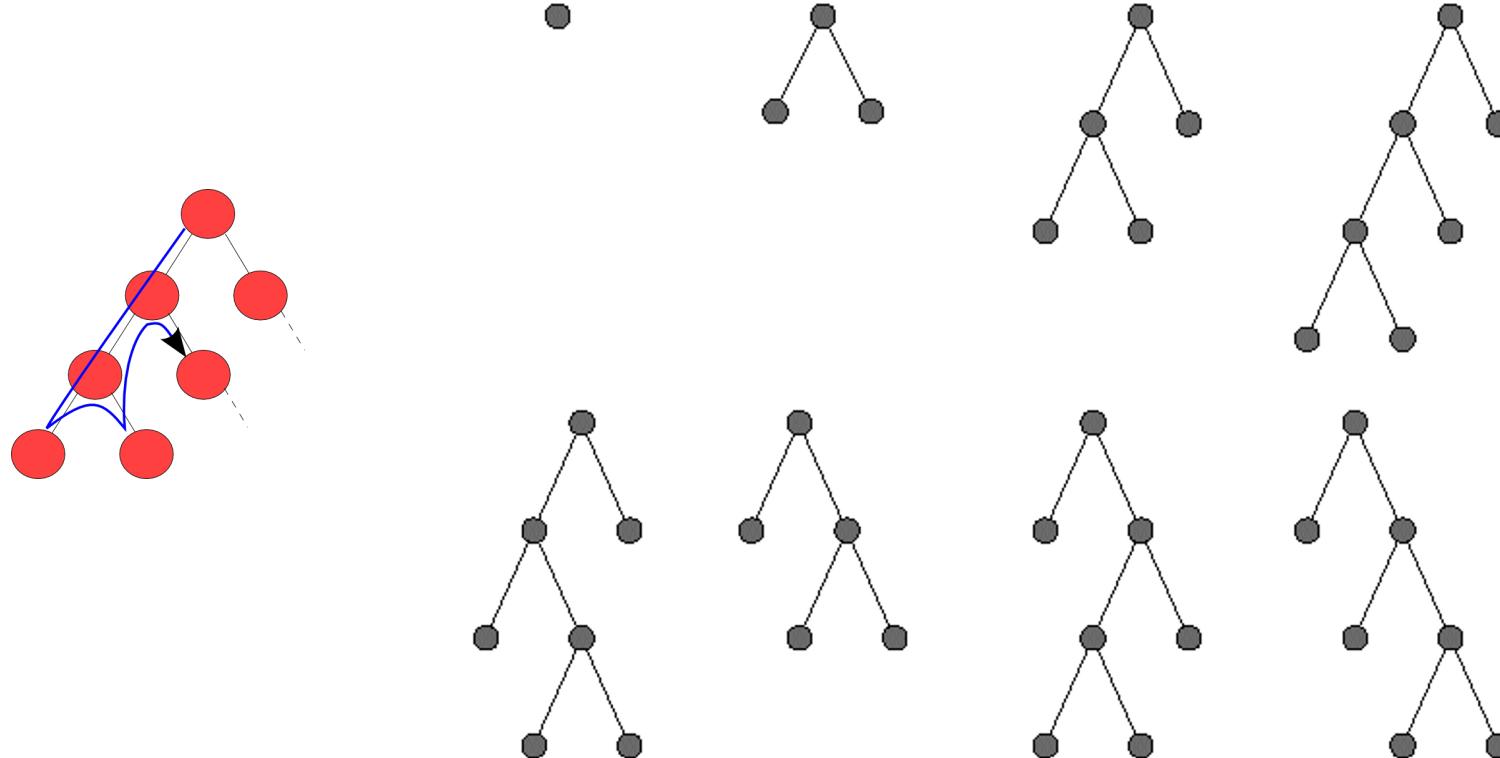
(b) progresso da busca

# Métodos de Busca

**Estratégias: Busca em profundidade**

# Métodos de Busca

## Estratégias: Busca em profundidade



- A busca em profundidade expande o nó no nível mais profundo da árvore. Somente quando atinge um nó terminal a busca retorna para expandir nós mais rasos.

# Métodos de Busca

## Estratégias: Busca em profundidade

- A escolha do nó a ser expandido é puramente estrutural. Geralmente é escolhido o nó mais a esquerda ou mais a direita.
- O método possui requerimentos de memória modestos.
- A busca em profundidade é inadequada para o uso em problemas modelados por árvores muito profundas, já que muito recurso computacional pode ser desperdiçado na exploração de um ramo profundo que não leva ao objetivo.
- Esta estratégia pode não retornar (entra em loop) se existirem ramos com profundidade infinita (laço).
- Esse tipo de técnica é muito utilizada em sistemas especialistas na realização de deduções. Isso é feito baseado na conjectura de que o conhecimento especializado é pouco profundo, ou seja, são necessários poucos passos para chegar a uma conclusão.

# Métodos de Busca

## Estratégias: Busca em profundidade limitada

- Evita as desvantagens da busca em profundidade pela imposição de um limite à profundidade da busca.
- É útil quando se sabe a profundidade máxima da solução.
- Por exemplo, no caso da busca de um caminho de Arad até Bucareste em um mapa com 20 cidades sabe-se que o maior caminho terá no máximo 19 etapas.

# Métodos de Busca

## Estratégias: Busca em profundidade iterativa

- Esta estratégia é útil quando não se sabe qual o limite que deve ser imposto
- Ela executa a busca em profundidade limitada várias vezes, passando a cada chamada um limite maior, até que seja retornada a solução
- Ela combina os benefícios da busca em amplitude e da busca em profundidade

# Métodos de Busca

## Estratégias: Busca em profundidade iterativa

```
function ITERATIVE-DEEPENING-SEARCH(problem) returns a solution sequence
  inputs: problem, a problem

  for depth  $\leftarrow 0$  to  $\infty$  do
    if DEPTH-LIMITED-SEARCH(problem, depth) succeeds then return its result
  end
  return failure
```

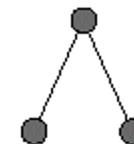
# Métodos de Busca

## Estratégias: Busca em profundidade iterativa

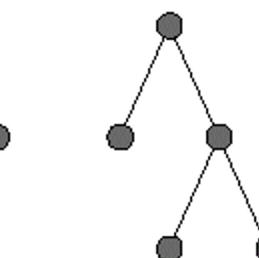
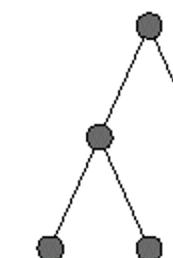
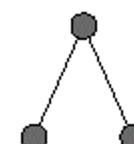
Limit = 0



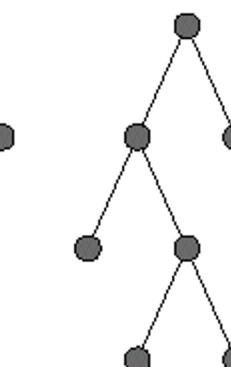
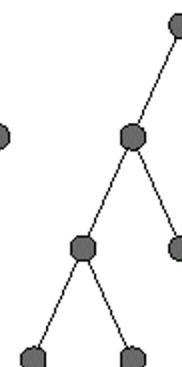
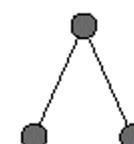
Limit = 1



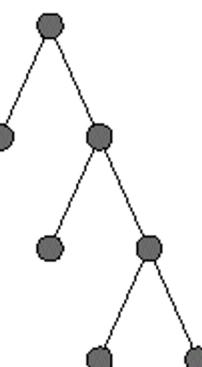
Limit = 2



Limit = 3



.....



# Métodos de Busca

## Estratégias: Busca em profundidade iterativa

- Apesar de parecer ineficiente, na realidade a sobrecarga das expansões múltiplas é comparativamente pequeno.
- A razão é que em uma árvore de busca a maioria dos nós estão no nível mais baixo.
- Por exemplo seja uma árvore com  $b=10$  e  $d=5$

$1+10+100+1000+10000+100000 = 111111$       Busca em amplitude

$$(d+1)1 + (d)b + (d-1)b^2 + \dots + 3b^{d-2} + 2b^{d-1} + b^d$$

$6+50+400+3000+20000+100000 = 123456$       Busca iterativa

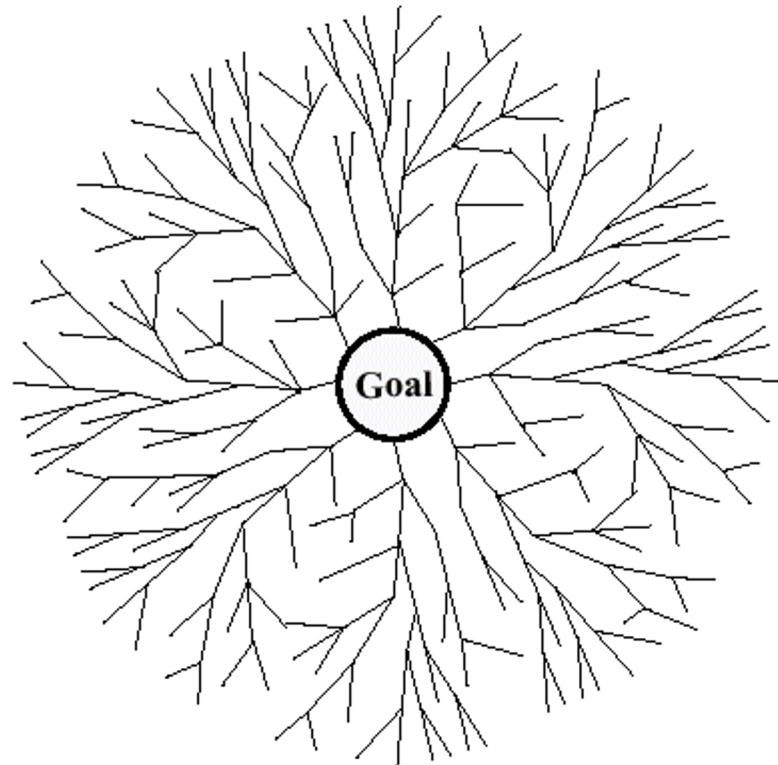
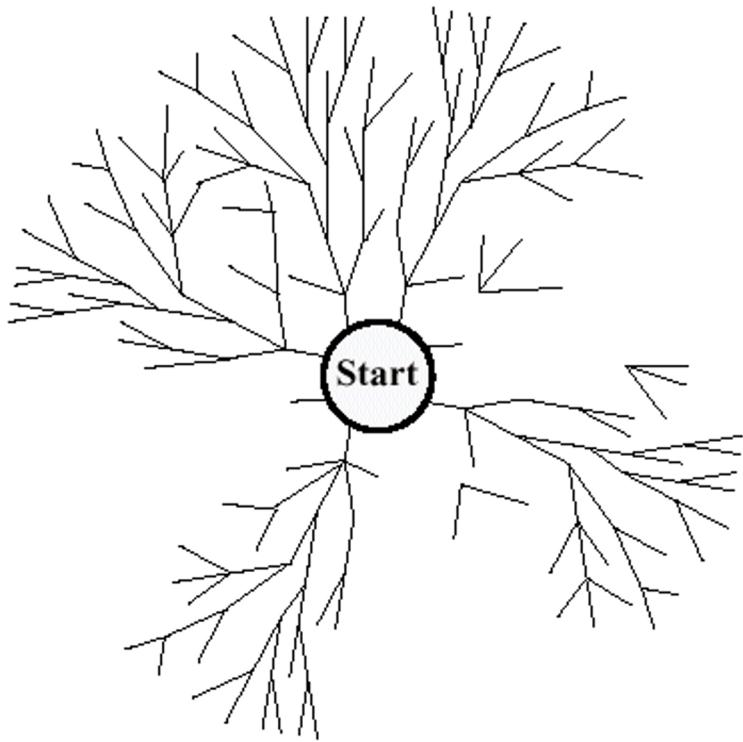
# Métodos de Busca

## Estratégias: Busca bi-direcional

- Na busca b-direcional a busca é realizada simultaneamente a partir do estado inicial e a partir do estado objetivo.
- A busca termina quando as duas buscas se encontram.
- Em problemas onde o fator de ramificação é b em ambas as direções esta estratégia é vantajosa.
- No entanto só pode ser aplicada se sabe como expandir a partir do objetivo. Ou seja, as ações são reversíveis e o objetivo (ou objetivos) são conhecidos.

# Métodos de Busca

## Estratégias: Busca bi-direcional



# Métodos de Busca

## Estratégias: Avaliação

Criterion	Breadth-First	Uniform-Cost	Depth-First	Depth-Limited	Iterative Deepening	Bidirectional (if applicable)
Time	$b^d$	$b^d$	$b^m$	$b^l$	$b^d$	$b^{d/2}$
Space	$b^d$	$b^d$	$bm$	$bl$	$bd$	$b^{d/2}$
Optimal?	Yes	Yes	No	No	Yes	Yes
Complete?	Yes	Yes	No	Yes, if $l \geq d$	Yes	Yes

d - profundidade da solução

m - profundidade da árvore

# Métodos de Busca



**FIM**