Inteligência Artificial

Busca sem informação

Profa. Debora Medeiros

Busca

- Abordagem para resolução de problemas em IA
- Mecanismo de resolução de problemas universal que:
  - Sistematicamente explora as alternativas
  - Encontra a sequência de passos para uma solução

Exemplo de problema

- Viajar de SJCampos a Ribeirão Preto
  - Adotando a menor rota possível

Resolução de problemas

- Objetivos
  - O que deve ser alcançado
    - Ir a Ribeirão Preto
  - Situação atual
    - Estamos em São José dos Campos
  - E medida de desempenho
    - Menor rota rodoviária possível
- Objetivo pode ser visto como um estado
  - Entre possíveis estados do problema
  - Ex. estado: estar em alguma cidade

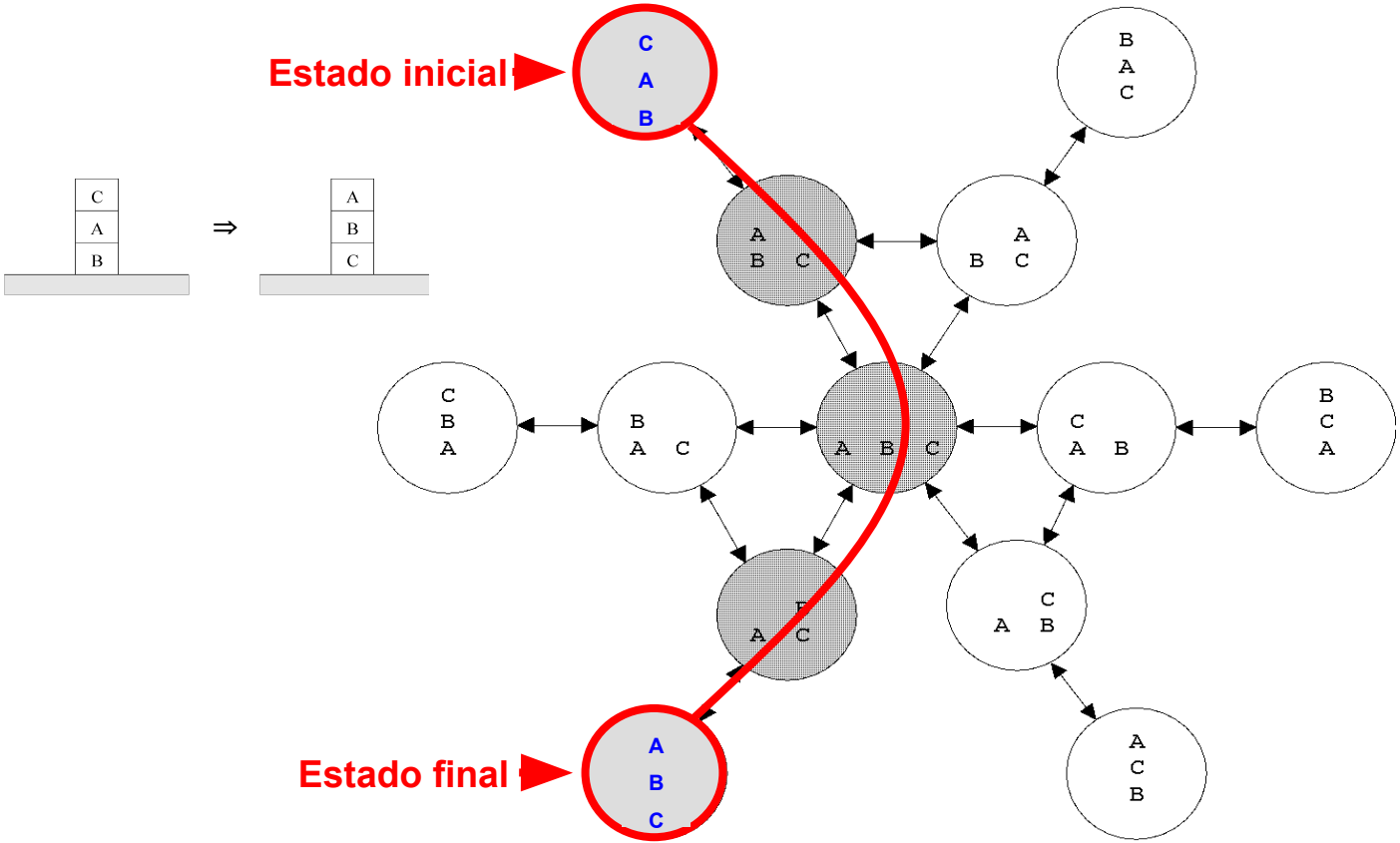
Resolução de problemas

- Podem haver estados intermediários no “caminho” da solução
  - Ex.: há três estradas saindo de São José dos Campos
- Descobrir que sequência de ações levará ao objetivo
  - Que cidades percorrer para chegar a Ribeirão Preto?

Resolução de problemas

- Espaço de estados
- Representação do problema
  - Estados e operadores de mudanças de estados
- Busca-se uma sequência de ações/operadores que leve a estados desejáveis (objetivos)

Resolução de problemas



Busca

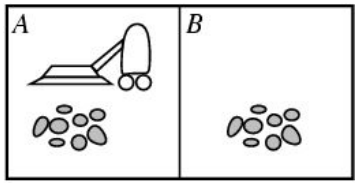
- Buscar solução
  - Escolha de sequência de ações a ser seguida entre diversas possibilidades
- Algoritmo de busca
  - Entrada = problema
  - Saída = sequência de ações para chegar à solução

Solução

- Solução:
  - Caminho desde o estado inicial até o estado objetivo
- Solução ótima
  - Menor custo de caminho entre todas as possíveis soluções

Exemplo

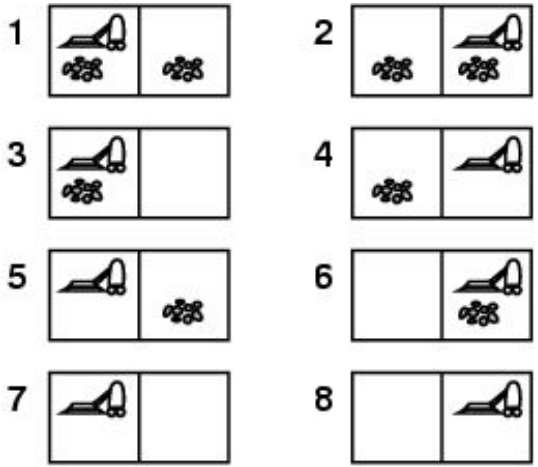
- Aspirador de pó
  - Percorre quadrados e vê se tem sujeira a limpar



  - Operações possíveis:
    - Mover para a direita
    - Mover para a esquerda
    - Aspirar pó
    - Não fazer nada

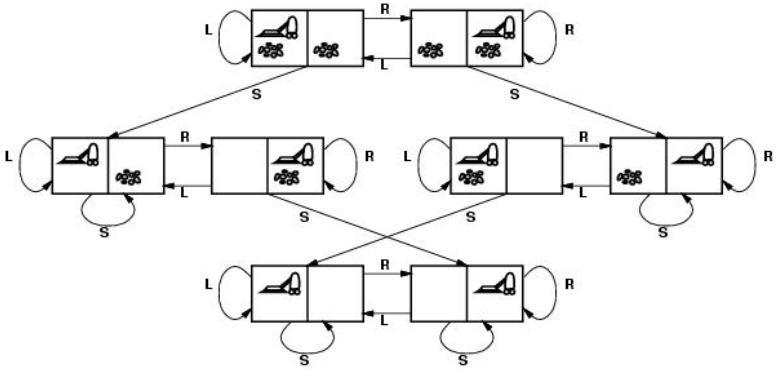
Exemplo

- Formulação do problema:
  - *Estados*:
    - 2 quadrados, contendo ou não sujeira
    - 8 estados possíveis



Exemplo

- Formulação do problema:
  - *Estado inicial*:
    - Qualquer um dos estados possíveis
    - Ex.: estado 5
  - *Função para gerar novos estados (operadores)*:
    - Execução de: esquerda (L), direita (R) ou aspirar (S)

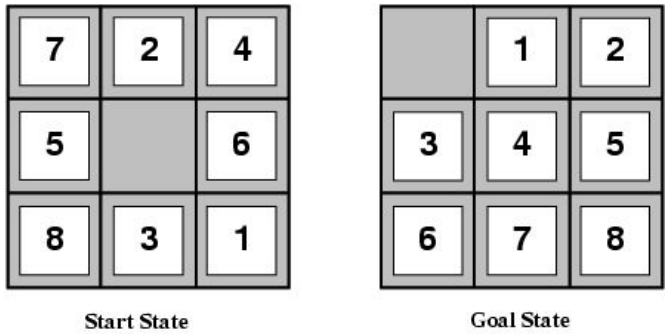


Exemplo

- Formulação do problema:
  - *Teste de objetivo (término)*:
    - Verificar se quadrados estão limpos
  - *Custo de caminho*:
    - Cada passo custa 1
    - Custo do caminho é o número de passos do caminho

Exemplo

- Quebra-cabeça de 8 peças
  - Tabuleiro 3x3 com 8 peças numeradas e um espaço vazio
  - Uma peça pode deslizar para o espaço
  - Exemplo:



Exemplo

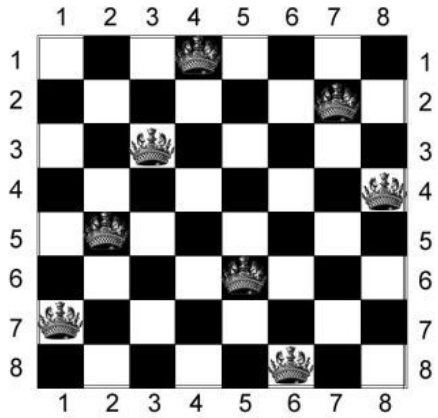
- Formulação do problema:
  - *Estados*:
    - Um estado especifica a posição de cada peça e do espaço vazio nos 9 quadrados
  - *Estado inicial*:
    - Qualquer estado
  - *Operadores*:
    - Espaço vazio se desloca para esquerda, direita, acima ou abaixo
  - *Teste de término*:
    - O estado é o final?
  - *Custo de caminho*:
    - Cada passo custa 1
    - Custo do caminho = número de passos no caminho

Exemplo

- Problema dos quebra-cabeças deslizantes
  - NP-completo
    - Ainda não existem algoritmos determinísticos polinomiais para sua solução
      - 8 peças: 181440 estados possíveis
      - 15 peças: 1,3 trilhão de estados
      - 24 peças: 10^{25} estados

Exemplo

- Problema das 8 rainhas
  - Posicionar 8 rainhas em um tabuleiro de xadrez de forma que nenhuma rainha ataque outra



Exemplo

- Formulação incremental
  - Adiciona rainhas ao tabuleiro
  - *Estados*:
    - Qualquer disposição de 0 a 8 rainhas no tabuleiro
    - Há $3x10^{14}$ possíveis estados
  - *Estado inicial*:
    - Nenhuma rainha no tabuleiro
  - *Operadores*:
    - Colocar uma rainha qualquer em um espaço vazio
  - *Teste de término*:
    - 8 rainhas no tabuleiro e nenhuma é atacada
  - *Custo*:
    - Não interessa, apenas estado final é importante

Exemplo

- Outra possível formulação
  - Proibir colocação de rainha em quadrados sob ataque
  - *Estados*:
    - Disposições de n rainhas (n ≤ 8) no tabuleiro, uma por coluna nas n colunas mais à esquerda, sem que uma rainha ataque a outra
    - Há 2057 possíveis estados
  - *Operadores*:
    - Colocar uma rainha em um espaço em coluna vazia mais à esquerda de modo que ela não seja atacada

Exemplo

- Contudo, para 100 rainhas
  - Primeira formulação: 10^{400} estados
  - Segunda formulação: 10^{52} estados
    - Espaço é menor, mas ainda grande

Importância de formulação apropriada!

Outros exemplos

- Problema do caixeiro-viajante
  - Visitar um conjunto de locais uma única vez usando o percurso mais curto
- Problemas de alocação (*Scheduling*)
  - Salas de aula

Outros exemplos

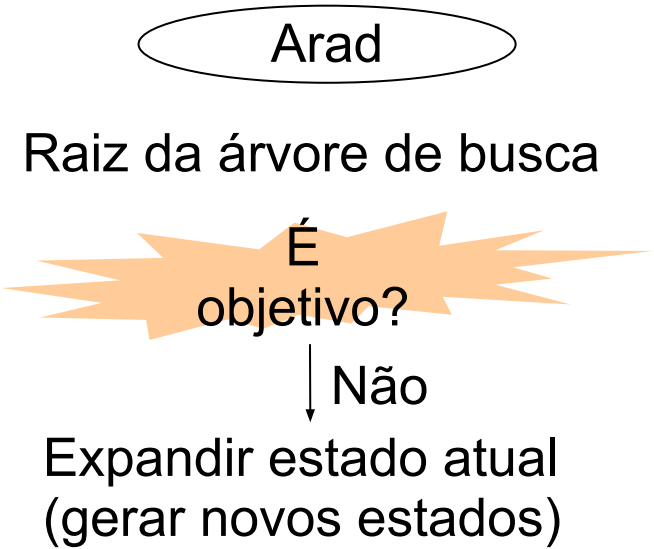
- Layout de VLSI
  - Posicionamento de componentes e conexões em chip minimizando
    - Área
    - Retardos de circuitos
  - E maximizando
    - Rendimento industrial

Busca

- Resolução dos problemas após formulação
  - Por meio de busca no espaço de estados
    - Árvore de busca
      - Gerada pela aplicação das **operações sobre os estados**
      - Iniciando pelo **estado inicial**
      - Até atingir um **estado objetivo**
- Exemplo
  - Viajar de Arad a Bucareste (Romênia)
  - Adotando menor rota possível

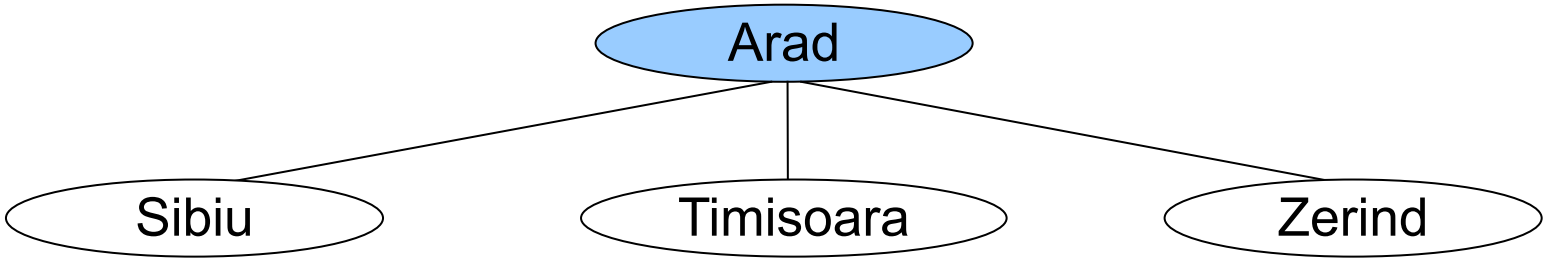
Exemplo

- (a) Estado inicial



Exemplo

- (b) Expandindo Arad

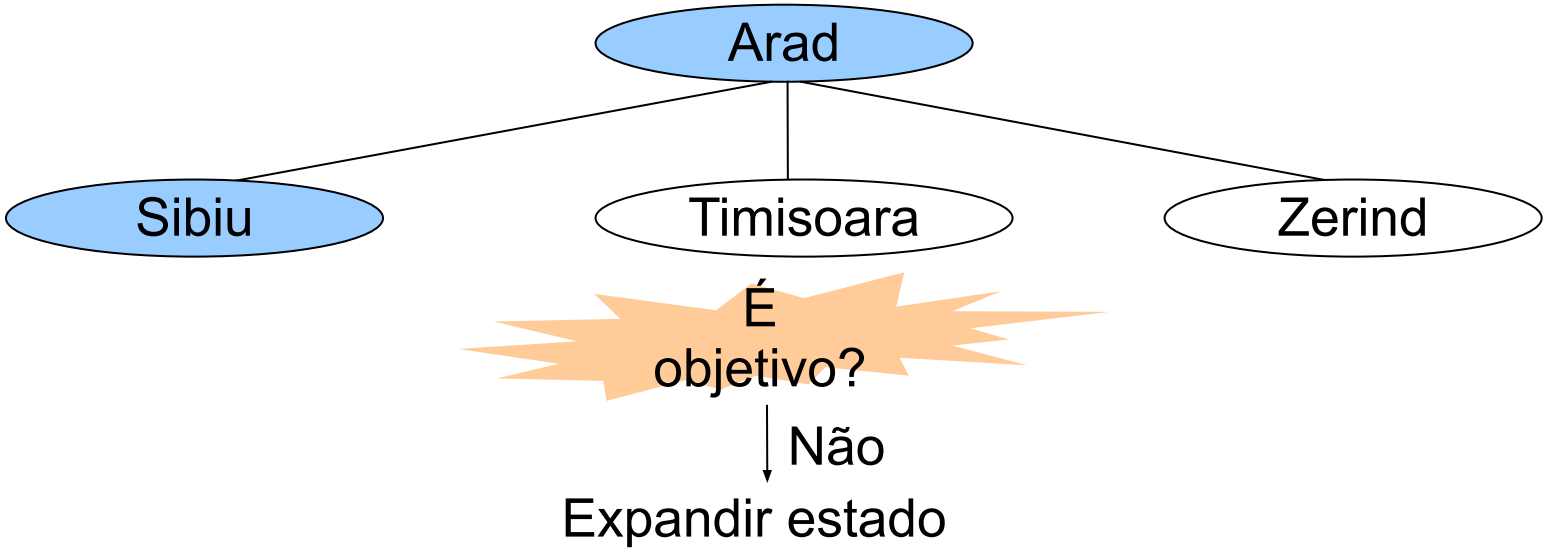


Qual escolher para futuras considerações?

Essência da busca: **seguir uma opção** e **deixar as outras reservadas** para mais tarde, no caso da primeira não levar a uma solução

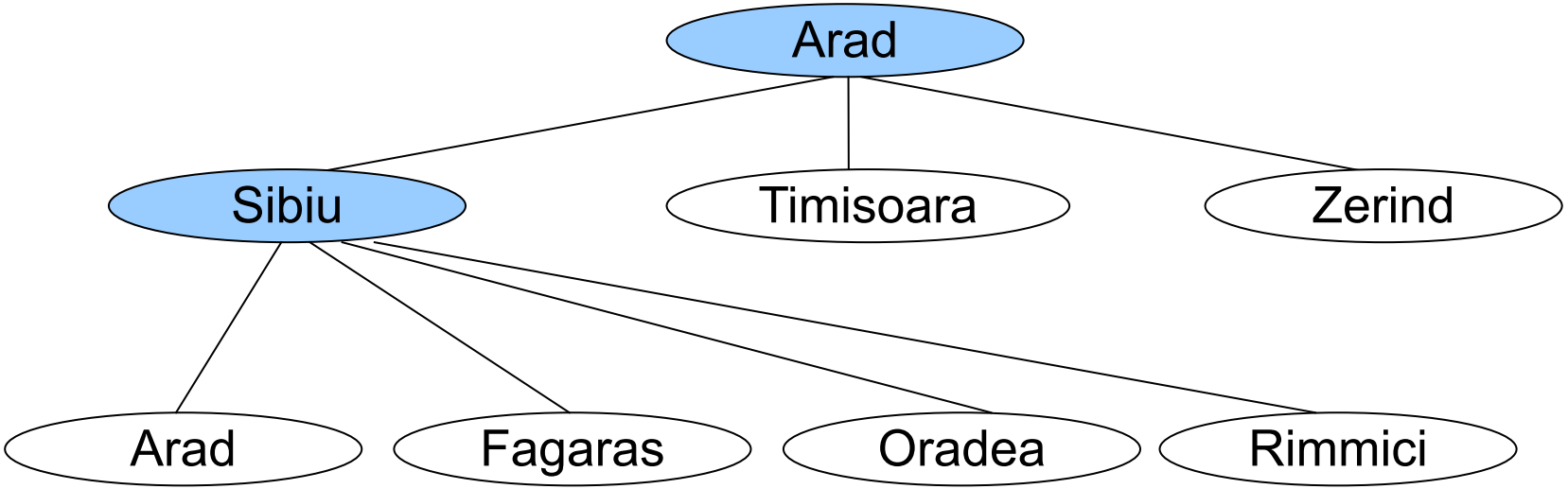
Exemplo

- (c) Supor escolha de Sibiu



Exemplo

- (c) Expandindo Sibiu



Pode escolher entre quaisquer estados ainda não visitados

Exemplo

- Continua-se a escolher, testar e expandir até
  - Encontrar solução ou
  - Não existirem mais estados a serem visitados
- **Escolha de estado a visitar** e expandir é determinada pela **estratégia de busca**

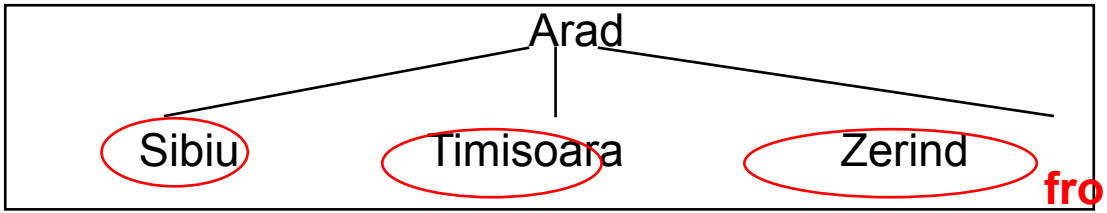
Fronteira

Fronteira **do espaço de estados**

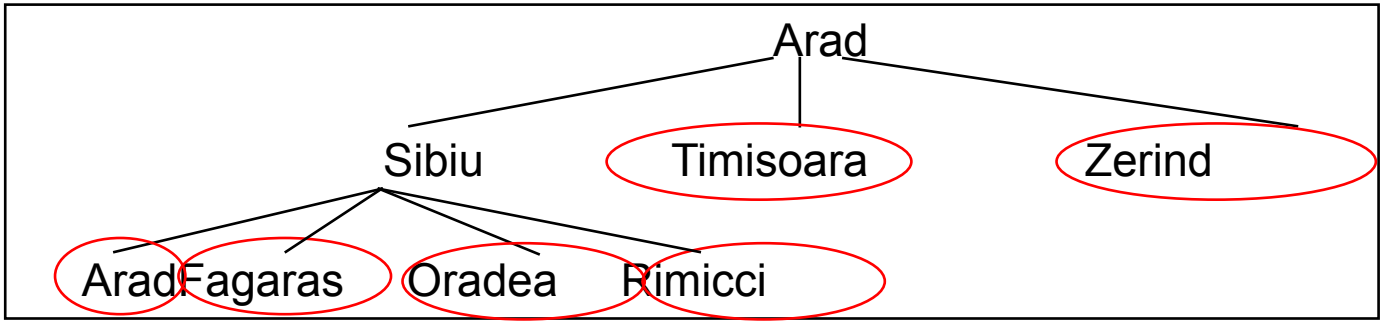
nós (estados) disponíveis para serem expandidos no momento

estado inicial =>

fronteira



fronteira



fronteira

Algoritmo

Algoritmo mais detalhado:

Função Inserir: controla a ordem de inserção de nós na fronteira do espaço de estados (de acordo com estratégia de busca)

função Busca-Genérica (*problema*, Função-Insere)

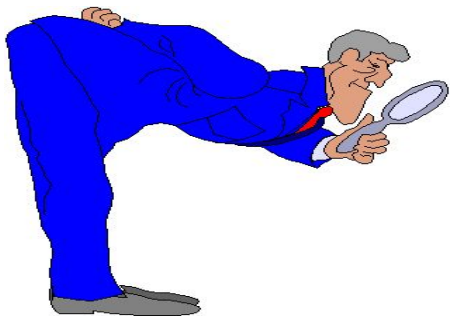
(retorna uma solução ou falha)
fronteira ← Inserir (Nó (Estado-Inicial [*problema*]))
loop do
se *fronteira* está vazia **então retorna** falha
nó ← Remove-Primeiro (*fronteira*)
se Teste-Término [*problema*] aplicado a Estado [*nó*] tiver sucesso
então retorna *nó*
fronteira ← Inserir(*fronteira*, Expandir[*problem*a, *nó*])
end

Estratégias de busca sem informação

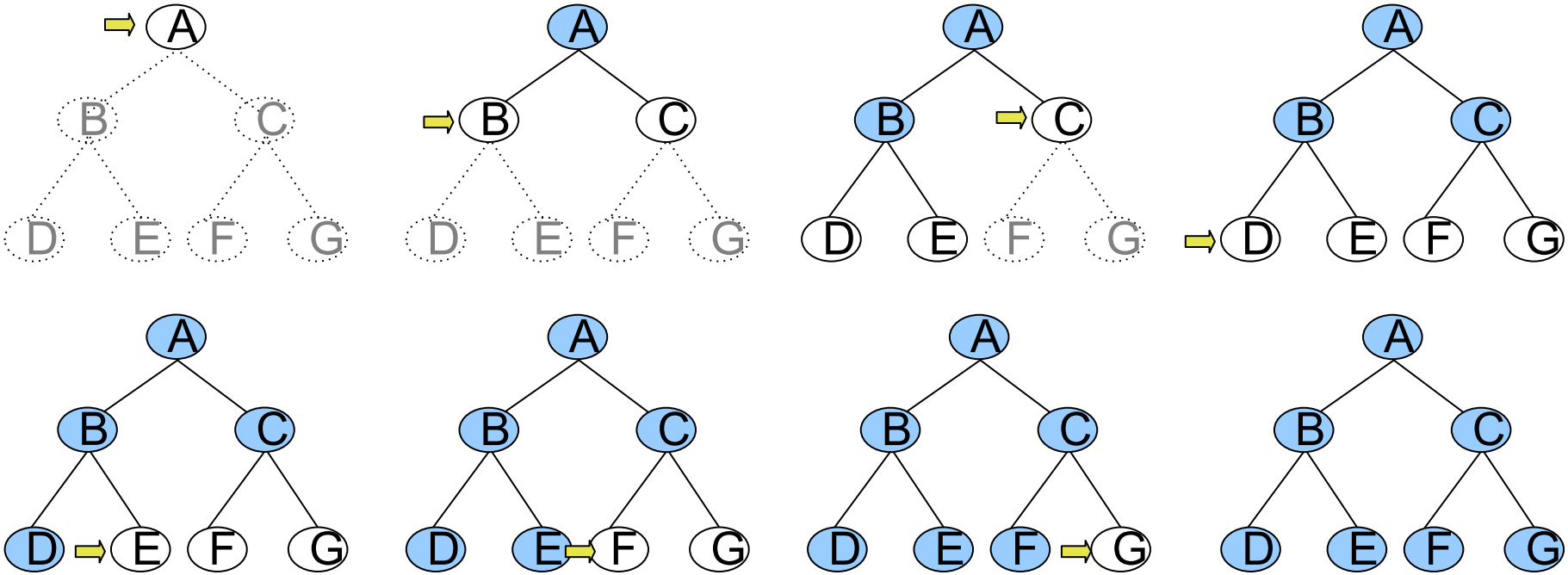
- **Estratégias para determinar a ordem de ramificação dos nós:**
  1. Busca em largura ou extensão
  3. Busca em profundidade
  4. Busca com aprofundamento iterativo
- **Direção da ramificação:**
  1. Do estado inicial para um estado final
  2. De um estado final para o estado inicial
  3. Busca bi-direcional

Busca em largura

- Nó raiz é expandido primeiro, depois todos os seus sucessores, depois os sucessores deles e assim por diante
  - Busca em extensão
  - Todos os nós em um nível da árvore de busca são expandidos antes dos nós do nível seguinte



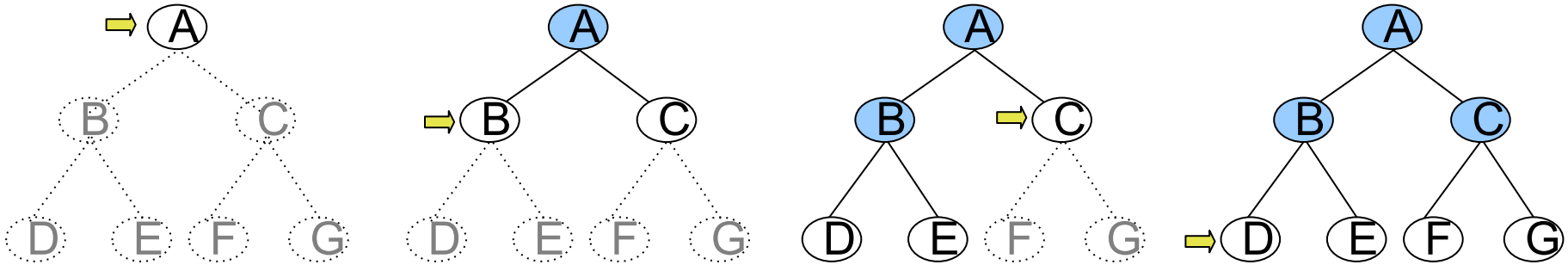
Busca em largura



Busca em largura

- Fronteira pode ser vista como fila
  - Novos sucessores são colocados no final
  - O primeiro da fila é selecionado a cada passo

Árvore de busca:



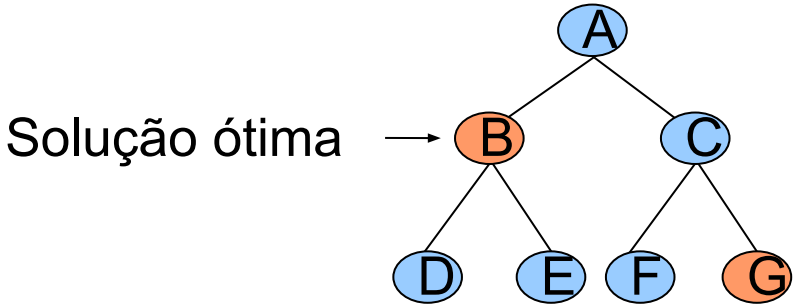
Fila:

A B C C D E D E F G

Considerações sobre desempenho

- **Completeza**
  - O algoritmo garante encontrar solução quando ela existe?

- **Otimização**
  - A estratégia encontra a solução ótima?
    - Para passos com igual custo, é aquela em menor profundidade na árvore de busca



Busca em largura

- É ***completa***
  - Quando fator de ramificação é finito

- É ***ótima*** se custo de caminho cresce com a profundidade do nó
  - Ou seja, quando:

$$\forall n', n \quad \text{profundidade}(n') \geq \text{profundidade}(n) \Rightarrow \text{custo de caminho}(n') \geq \text{custo de caminho}(n)$$

Ex.: quando todas operações tiverem o mesmo custo
Pois sempre explora profundidades mais rasas primeiro

- Pode não ser solução de menor **custo de caminho,** caso 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

Considerações sobre desempenho

- **Complexidade de tempo**
  - Quanto tempo o algoritmo leva para encontrar uma solução?

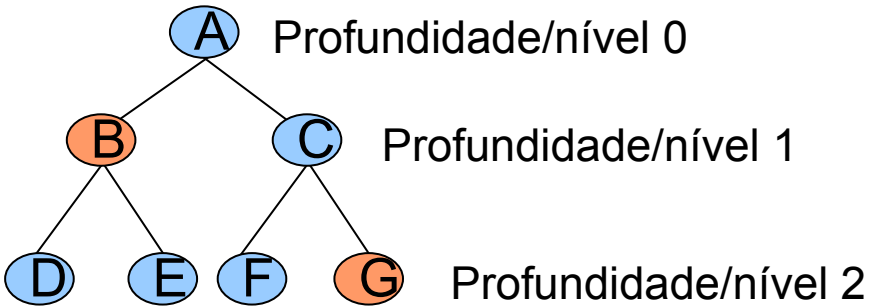
- **Complexidade de espaço**
  - Quanta memória é necessária para executar a busca?

Análises em função de:

b: fator de ramificação da árvore
d: profundidade da solução mais rasa
m: profundidade máxima da árvore de busca

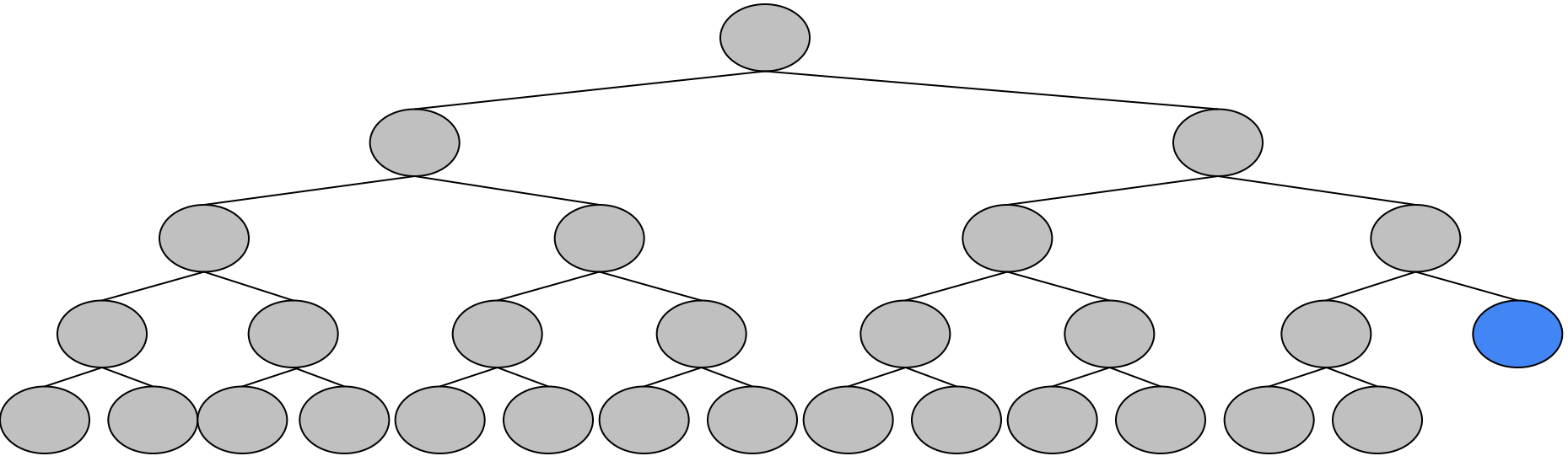
Desempenho

- Exemplo:
  - b = fator de ramificação = 2
  - d = profundidade da solução mais rasa = 1
  - m = profundidade máxima da árvore de busca = 2



Busca em largura – pior caso

Último nó no nível d

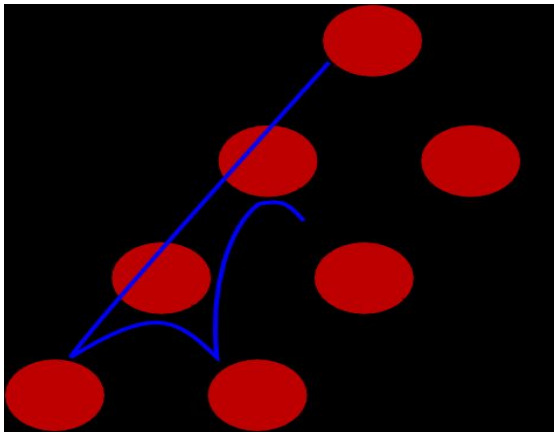


Busca em largura

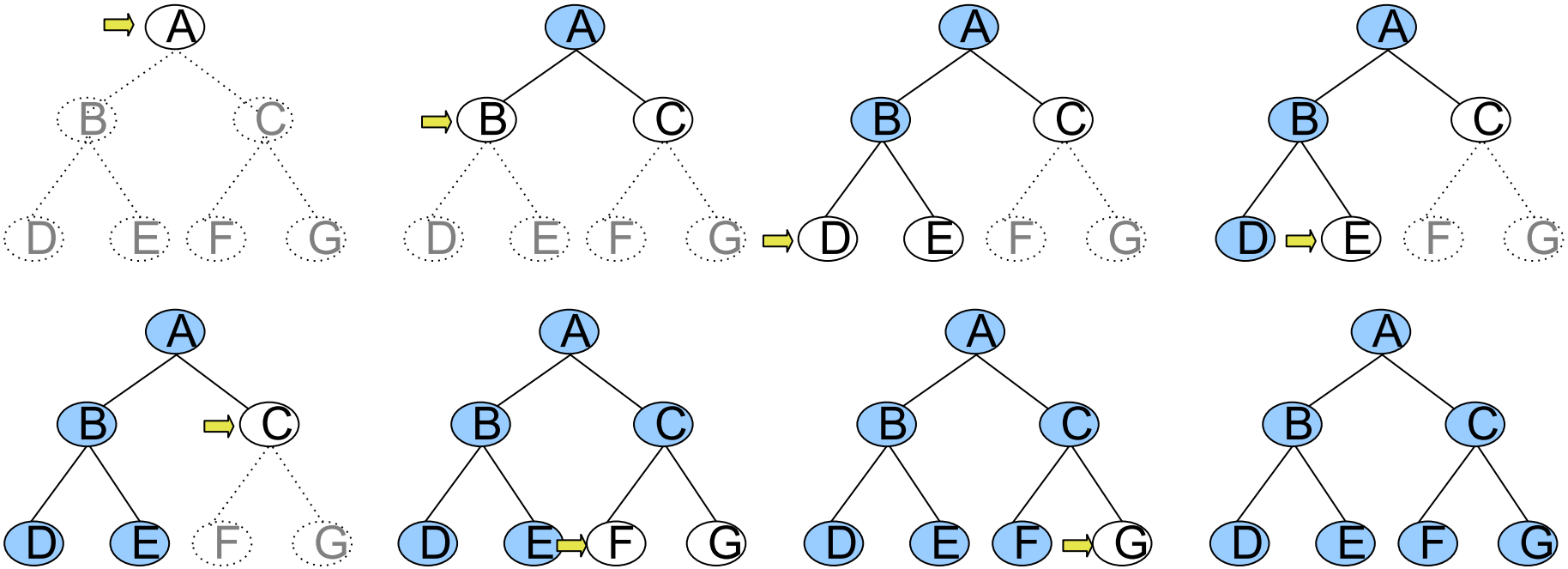
- ***Complexidade de tempo e memória***
  - Considere que cada estado tem b sucessores
    - Número de nós em nível i = b^i
  - Suponha que primeira solução está no nível d
    - No pior caso, expande todos os nós exceto o último no nível d, gerando $b^{d+1} – b$ nós no nível d+1
    - Número total de nós gerados = $1+ b + b^2 +...+ (b^{d+1} – b)$
      - $O(b^{d+1})$: complexidade exponencial
      - Impraticável para problemas grandes
    - Conjunto fronteira (**$O(b^{d+1})$**)

Busca em profundidade

- Sempre expande o nó atual mais profundo
  - Até chegar em objetivo ou em nó folha
    - Neste caso, retorna então ao nó seguinte mais raso



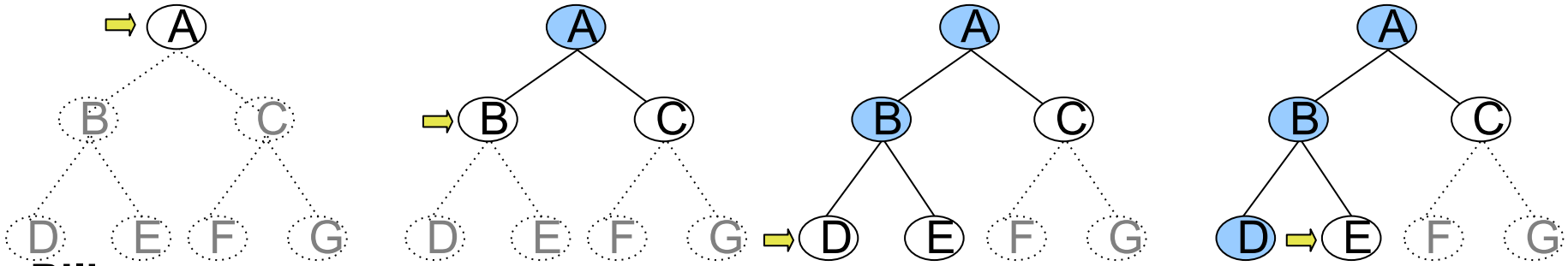
Busca em profundidade



Busca em profundidade

- Fronteira pode ser vista como pilha
  - Novos sucessores são colocados no final
  - O último ou topo da pilha é selecionado a cada passo

Árvore de busca:



Pilha:

A

B
C

D
E
C

E
C

Busca em profundidade

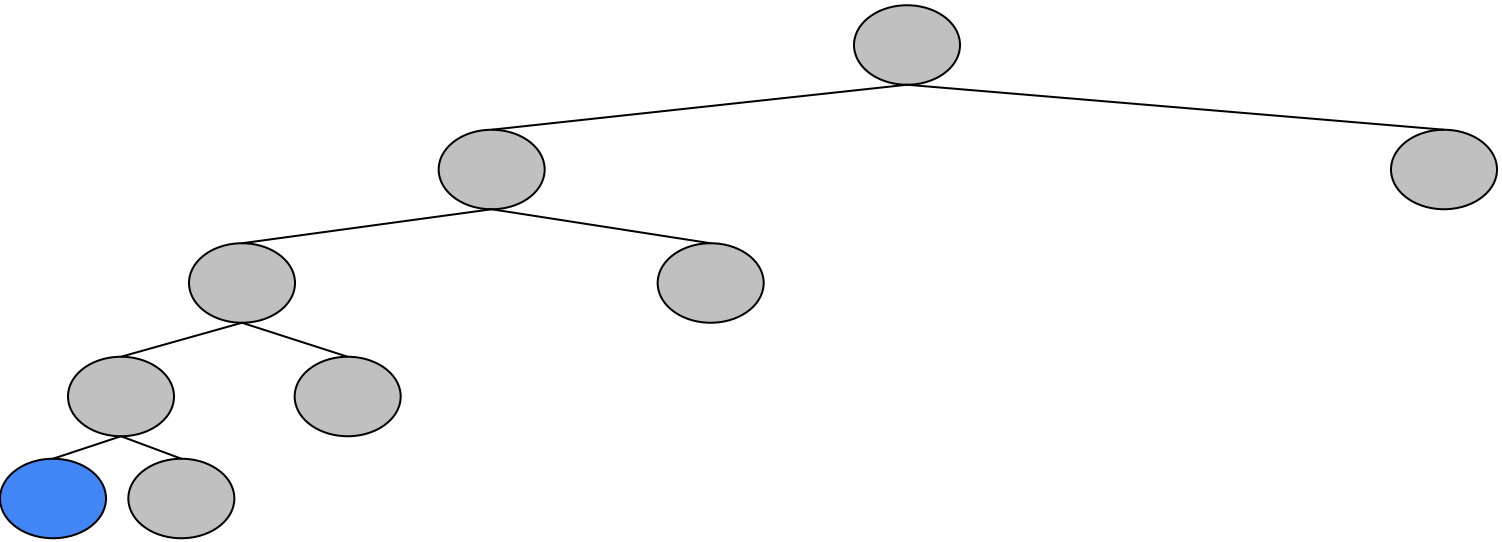
- **Requisitos de memória modestos**
  - Precisa armazenar:
    - um único caminho de raiz a uma folha,
  - Em espaço de busca com:
    - Fator de ramificação b
    - Profundidade máxima m
    - Exige armazenamento de bm+1 nós: O(bm) (linear!)

Busca em profundidade – pior caso para espaço

Objetivo é o último nó do primeiro ramo

Depois disso começamos a deletar nós

Número de nós gerados: b’nós em cada um dos m níveis



Busca profundidade x largura

- **Tempo**
  - *m* = *d*: BP tipicamente ganha
  - *m* > *d*: BL pode ganhar
  - *m* é ***infinito***: BL provavelmente irá melhor
- **Espaço**
  - BP geralmente é melhor que BL

Busca em profundidade

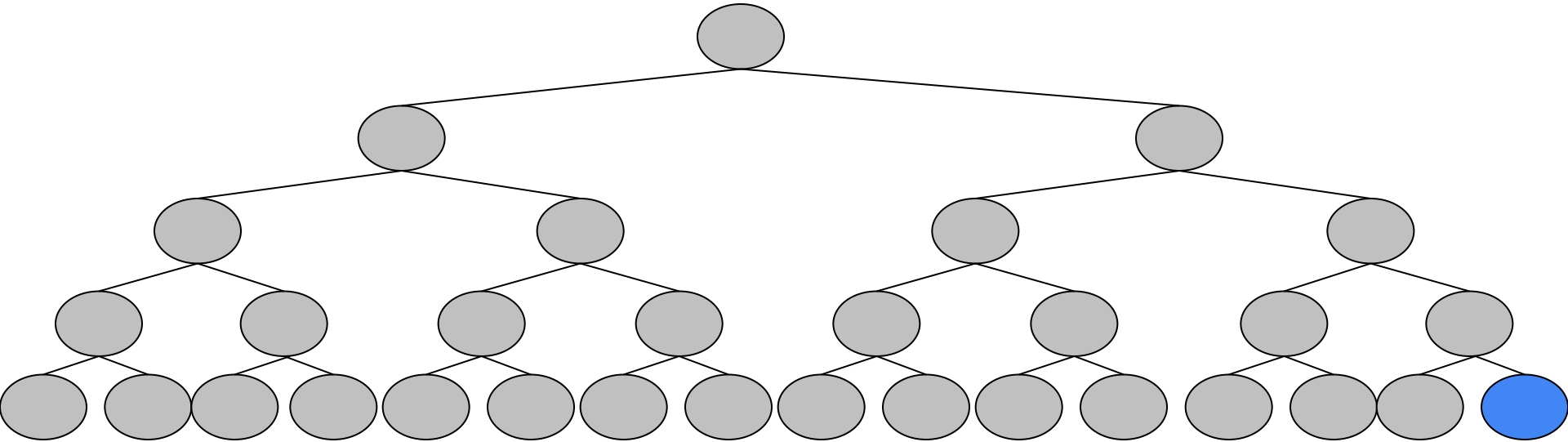
- Caminhos muito longos (ou infinitos)
  - Não é completa

- Também ***não é ótima***
  - Pode retornar solução em profundidade maior na árvore de busca do que a mais rasa

Busca em profundidade – pior caso para tempo

Objetivo é o último nó do último ramo

Número de nós gerados: toda a árvore



Busca em profundidade

- ***Complexidade* de pior caso**
  - Irá gerar todos os $O(b^m)$ nós na árvore de busca
    - m é a profundidade máxima de qualquer nó

Busca em profundidade limitada

- Limite máximo de profundidade a ser explorado
- Pode ser *incompleto*
  - Se solução mais rasa estiver abaixo de limite l
- Se d < l, *não é ótima*
  - *Complexidade de tempo*: $O(b^l)$
  - *Complexidade de espaço*: O(bl)

Busca em profundidade por aprofundamento iterativo

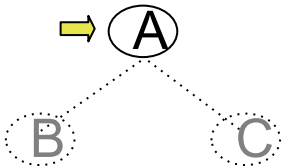
- **Busca por aprofundamento iterativo**
  - Usada em conjunto com busca em profundidade
  - Encontra o melhor limite de profundidade
    - Aumentando gradualmente o limite
      - 0, 1, 2, etc
      - Até encontrar um objetivo
        - Acontece quando alcançar d, profundidade do nó objetivo mais raso

Busca por aprofundamento iterativo

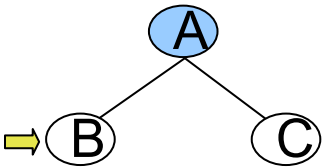
Limite = 0

Busca por aprofundamento iterativo

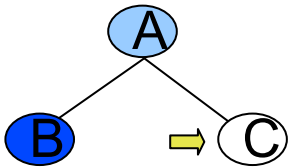
Limite = 1



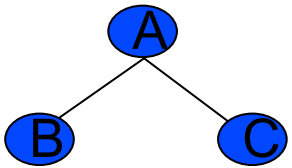
A



A B C

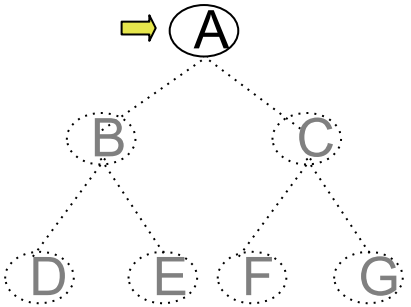


A C

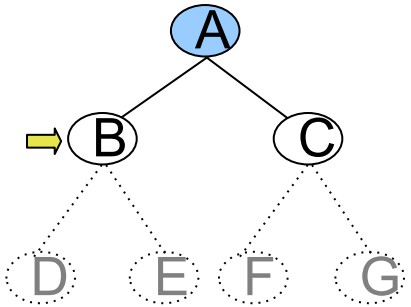


Busca por aprofundamento iterativo

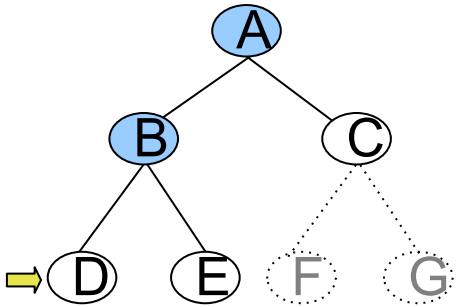
Limite = 2



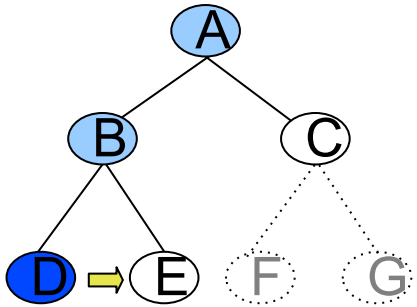
A



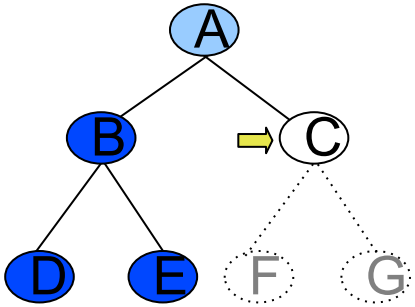
A B C



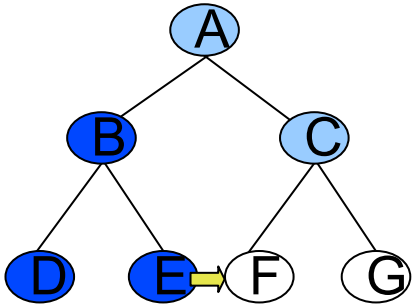
A B C D E



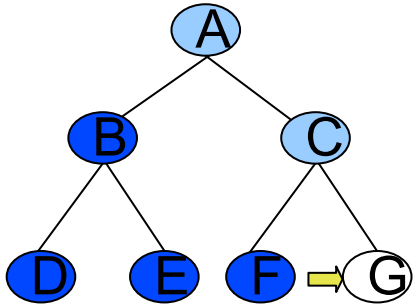
A B C E



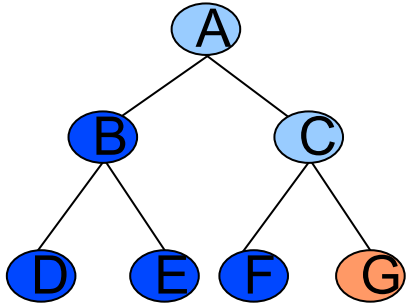
A C



A C F G



A C G



A C G

Busca por aprofundamento iterativo

- **Pode parecer desperdício**
  - Estados são gerados várias vezes
    - Custo não é muito alto, pois a **maior parte dos nós** estará em **níveis inferiores**
      - Nós do nível inferior (profundidade d) são gerados uma vez...
      - Os do penúltimo são gerados duas vezes...
      - E assim por diante...
      - Até filhos da raiz, gerados d vezes

Busca por aprofundamento iterativo

- **Número total de nós gerados é:**
  - nós(BAI) = $db + (d-1)b^2 + ... + (1)b^d$
    - *Complexidade de tempo* $O(b^d)$
    - É mais rápido que busca em largura
    - nós(BL) = $b + b^2 + ... + b^d + (b^{d+1} – b)$
    - nós(BP) = $b + b^2 + ... + b^d$
      - Limitando profundidade em d
  - Ex.: b = 10, d = 5;
    - nós(BAI) = 123450
    - nós(BL) = 1111100
    - nós(BP) = 111111
      - Overhead = (123,456 - 111,111)/111,111 = 11%

Busca em profundidade por aprofundamento iterativo

- **Busca por aprofundamento iterativo**
  - *Requisitos de memória modestos*: O(bd)
  - *Complexidade de tempo*: $O(b^d)$
  - *Completa* quando fator de ramificação é finito
  - *Ótima* quando custo de caminho cresce com a profundidade do nó

Busca por aprofundamento iterativo

Em geral, é método de busca sem informação preferido quando existe um espaço de busca grande e a profundidade da solução não é conhecida

É **similar à busca em largura**, pois explora um nível de nós em cada iteração, porém mais eficiente em tempo e espaço

Comparação entre estratégias de busca sem informação

Critério	BL	BP	BPL	BAI
Completa	Sim*	Não	Não	Sim*
Tempo	$O(b^{d+1})$	$O(b^m)$	$O(b^l)$	$O(b^d)$
Espaço	$O(b^{d+1})$	O(bm)	O(bl)	O(bd)
Ótima	Sim***	Não	Não	Sim***

b é fator de ramificação

d é profundidade da solução mais rasa

m é profundidade máxima da árvore de busca

l é limite de profundidade

* Se b é finito
** Se b é finito e ambos sentidos usam busca em extensão
*** Se custos dos passos são iguais

Referências

- Livro Russel e Norvig, capítulo 3
- Slides de:
  - Ana Carolina Lorena, UNIFESP
  - Ronaldo Prati, UFABC
  - UFPE
  - Prof Marcilio Souto, UFRN
  - Richard Khoury, Universisty of Waterloo
- Apostilas:
  - Grafos, Prof José Guimarães, UFSCar
  - Estruturas de dados, Prof Gustavo Nonato, ICMC-USP
  - Estrutura de dados, Prof Cândido Egypto, CEFET Paraíba