Resolução de Problemas por meio de busca – Busca Cega ou Exaustiva

Aula 02 – Busca Cega ou Exaustiva

Prof. Msc. Luiz Mário Lustosa Pascoal

O QUE É UM PROBLEMA?

- Qualquer **situação** a ser "resolvida" por uma **seqüência ações** a ser executada, com vistas a atingir um **objetivo**;
  - *Na descrição acima:*

  Situação = **estado inicial;**

  Seqüência de Ações = **operações** que irão gerar uma **sucessão de estados;**

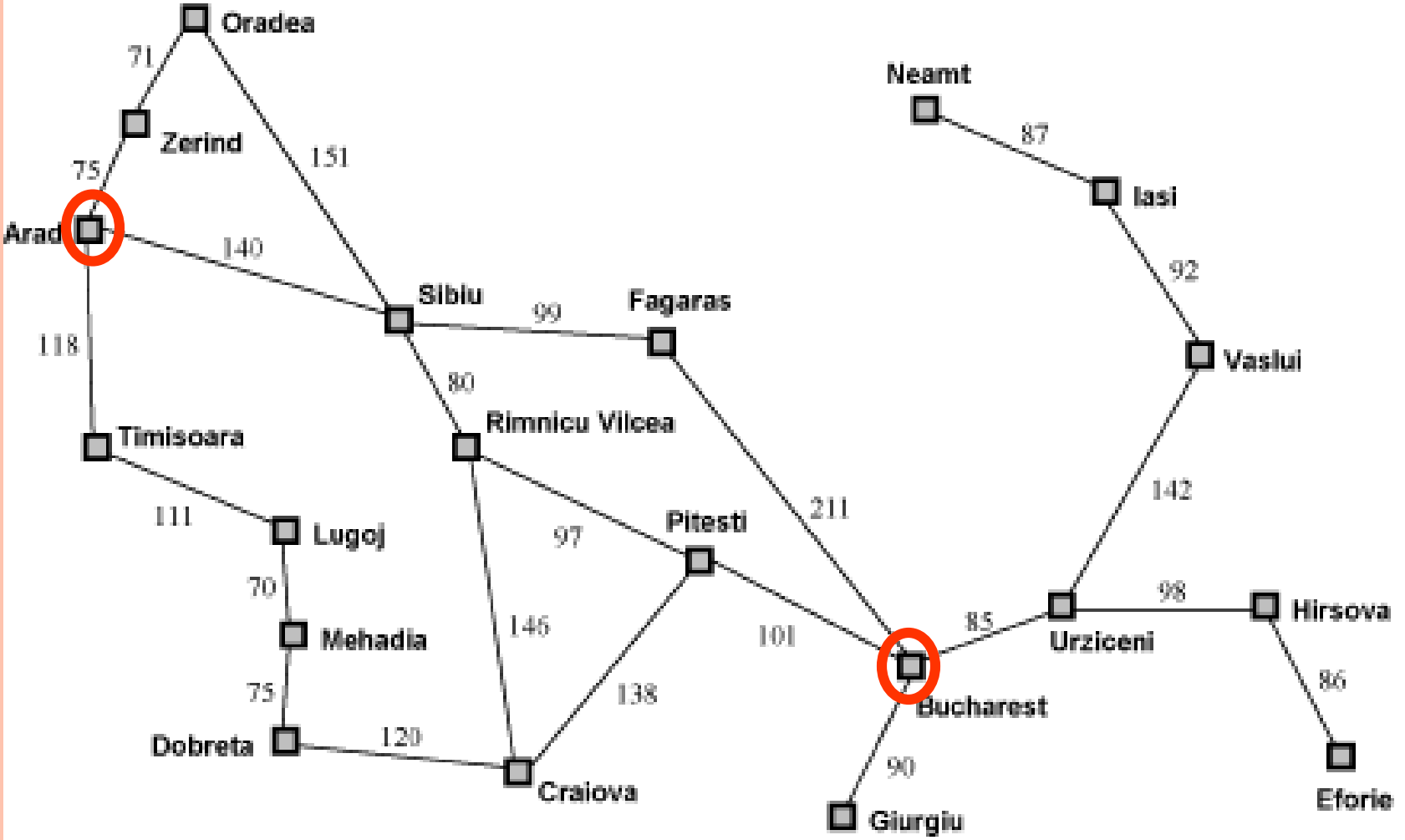
  Objetivo = **estado final** (ou conjunto de) desejado;

- Metáfora da resolução de problemas por meio de busca:
  - Transformar um problema de raciocínio de um agente *diretamente* em um **problema de navegação** num **espaço de estados**, no qual, a partir de um **estado inicial,** um agente pode **buscar** uma **seqüência de ações** que conduzem ao **estados final** desejado.

  - A ***solução*** de um problema é conjunto de seqüências de ações que levam de um estado inicial ao estado objetivado

  - Uma ***solução ótima*** é uma seqüência de ações que leva de um estado inicial ao estado objetivado com o menor **custo***.

Romênia: ir de Arad a Bucharest



Straight line distance to Bucharest	
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 da Solução

- Algoritmo Genérico de busca

Função AGENTE-DE –RESOLUÇÃO-DE PROBLEMAS-SIMPLES(percepção) retorna uma ação

entradas: percepção, uma percepção

variáveis státicas: seq, uma sequencia de ações inicialmente vazia

estado, alguma descrição do estado atual do mundo

Objetivo, um objetivo, inicialmente nulo

Problema, uma formulação de problema

Estado <- ATUALIZAR-ESTADO(estado,percepção)

se seq está vazia **então faça**

objetivo <- FORMULAR-OBJETIVO(estado)

problema <- FORMULAR-PROBLEMA(estado, objetivo)

seq <- BUSCA(problema)

ação <- PRIMEIRO(seq)

seq <- RESTO(seq)

retornar ação

Antes de qualquer coisa… **Formular o problema**

- Formular o **objetivo**: definir um estado final que satisfaz o agente

  ex: "Estar em Bucareste"

- Especificar o ***problema*** em termos de:
  - **Estado Inicial** ex: "Estou em Arad"
  - **Função Sucessor S(x)** = conjunto de pares ação-estado
    - ex: S(Arad) = { (Arad→Zerind, Zerind), (Arad→Sibiu, Sibiu), ...}
  - **Teste do Objetivo:** condição que é capaz de determinar se o estado final foi alcançado
    - ex: "Estou em Bucareste?"
  - **Custo (path cost)** ex: soma de distancias, nº de ações executadas, etc.

A Formulação do Problema define o Espaço de Estados possíveis

- **O Espaço de Estados**
  - É definido implicitamente pelo **estado inicial** juntamente com todos os **estados alcançáveis** através da **função sucessora S(x)**.
  - Forma um **grafo** onde os **nós** representam **estados** e os **arcos** entre os nós representam **ações.**

Entretanto:

Estado
Corresponde a uma configuração do mundo.
Ex: “estou em Bucareste”

≠

Nó
Estrutura de Informação que compõe uma seqüência percorrida. Ex:

Nó(“Bucareste”) ***(1)***
Antecessor = Nó(“Pitesti”)
Path-cost = 418 Km

Nó(“Bucareste”) ***(2)***
Antecessor = Nó(“Fagarás”)
Path-cost = 450 Km

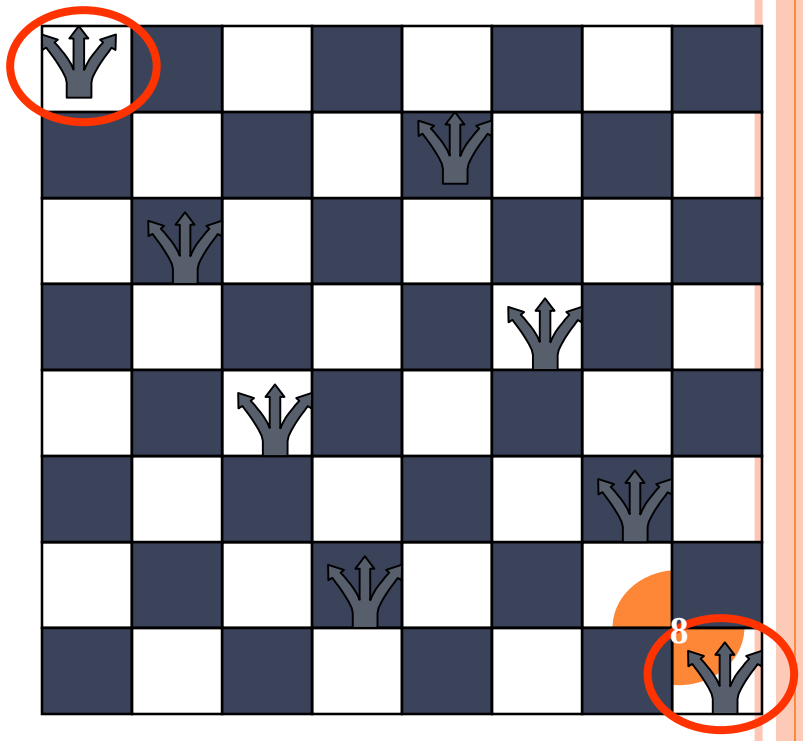
A Formulação do Problema exige Abstração do Mundo Real

- O mundo real é absurdamente complexo: espaço de estados tem que ser abstraído para a resolução de problemas.
- Estado (abstrato) = conjunto de estados reais
- Ação (abstrata) = combinação complexa de ações reais, onde cada ação abstrata deveria ser "mais fácil" que o problema original.
  - Ex: "Arad→Zerind" representa um complexo de rotas possíveis, paradas para descanso, etc.
- Solução (abstrata) = conjunto de caminhos reais que são soluções no mundo real para o problema.

Importância de uma Formulação adequada do Problema

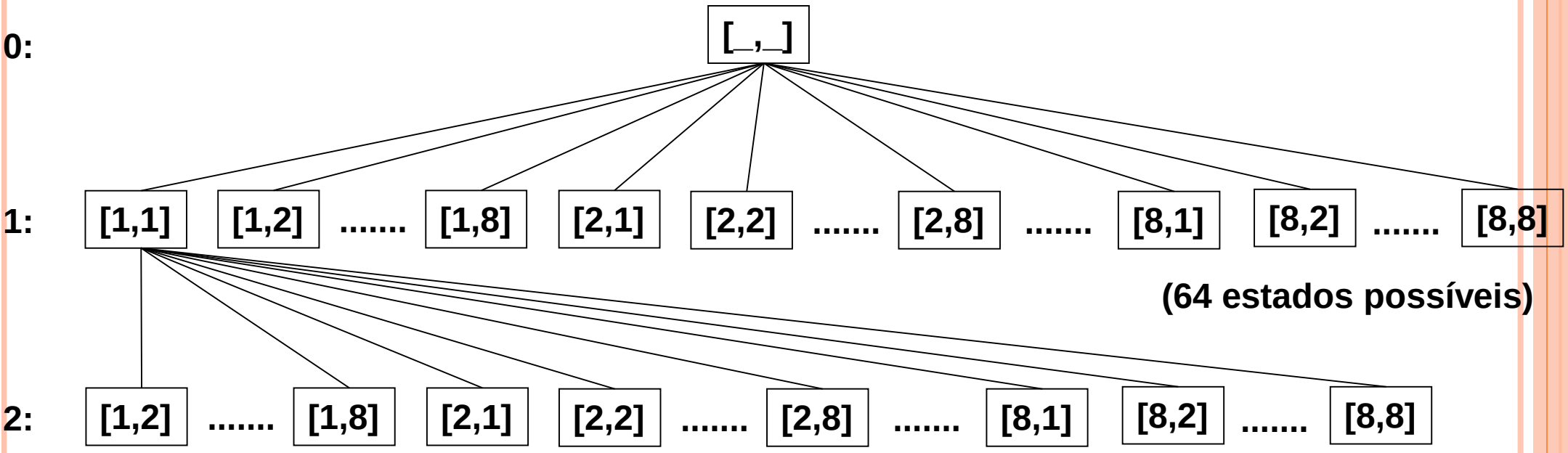
- O Espaço de Estados derivado da formulação do problema impacta na eficiência de busca da solução
- Ex: Problema das 8 Rainhas

  Dispor 8 rainhas num tabuleiro de xadrez, sem que qualquer uma delas esteja "sob ataque" das demais:
  - não pode haver mais de uma rainha em uma mesma linha, coluna ou diagonal
  - somente o custo da busca conta (nº de passos para a solução)



Importância de uma Formulação adequada do Problema

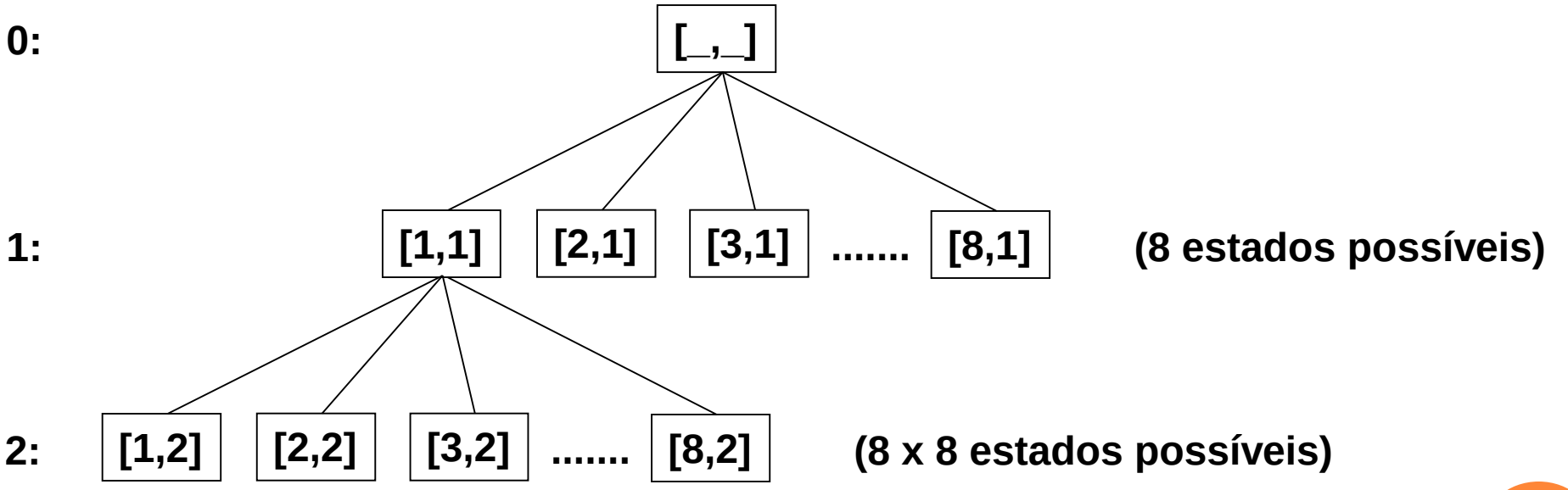
- **Problema das 8 Rainhas - Formulação 1:**
  - estado inicial: nenhuma rainha no tabuleiro
  - operadores: adicionar uma rainha a qualquer quadrado vazio
  - teste do objetivo: 8 rainhas no tabuleiro sem ataque mútuo?



.... E assim por diante, até 64 x 63 x 62 x ..... x 57 ≈ 1,8 x 10^{14} seqüências possíveis a serem testadas

Importância de uma Formulação adequada do Problema

- **Problema das 8 Rainhas - Formulação 2:**
  - estado inicial: nenhuma rainha no tabuleiro
  - operadores: adicionar uma rainha a qualquer quadrado da coluna mais a esquerda que não contém nenhuma rainha
  - teste do objetivo: 8 rainhas no tabuleiro sem ataque mútuo?



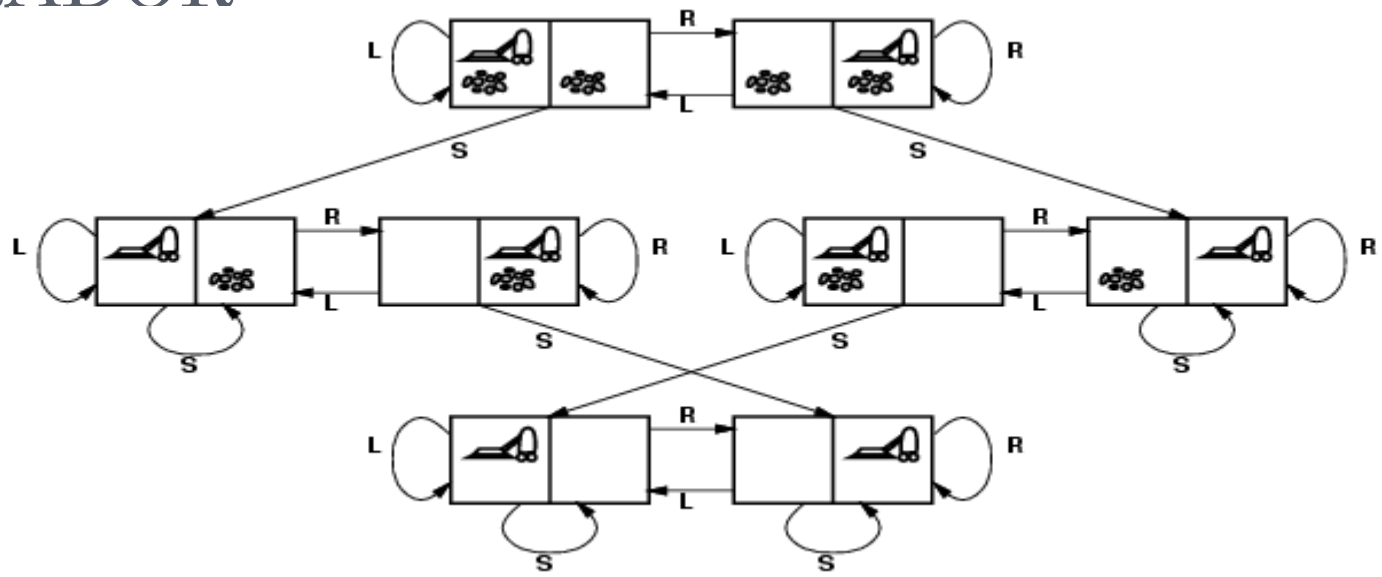
.... E assim por diante, até $8^8 \approx 1{,}7 \times 10^7$ seqüências possíveis a serem testadas **<<** **$1{,}8 \times 10^{14}$ possibilidades da formulação 1**

Busca em um Espaço de Estados

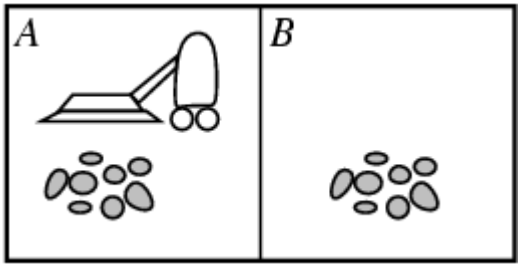
- Uma vez o problema bem formulado... usa-se um **método de busca** para encontrar o estado final desejado.

- Métodos genéricos de busca:
  - **Busca exaustiva ou cega**
    - Não sabe qual o **melhor** nó da fronteira a ser expandido = menor custo de caminho desse nó até um ***nó final*** *(objetivo).*
  - **Busca heurística - informada**
    - Estima qual o melhor nó da fronteira a ser expandido com base em ***funções heurísticas*** => conhecimento

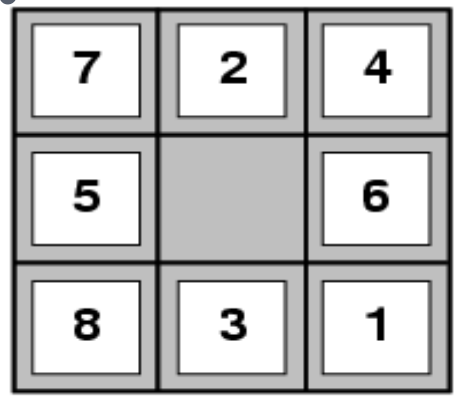
Gráfico do Espaço de Estados do Aspirador



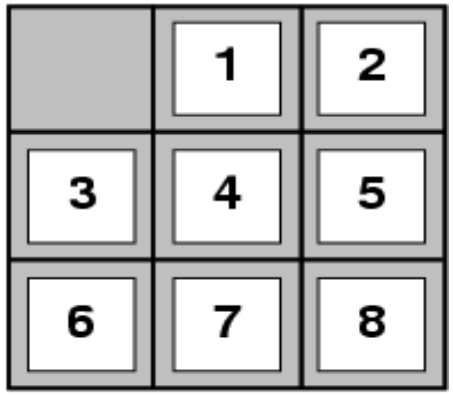
- Estados: O agente está entre duas posições que podem estar limpas ou sujas. Ou seja, há 2 X 2^2=8 estados possíveis
- Ações: *Esquerda*, *Direita*, *Aspirar*
- Teste de Objetivo: Todos os locais limpos
- Custo de Caminho: 1 por ação



Exemplo: Quebra-cabeça 8 peças



Start State



Goal State

- Estados: Cada peça pode estar em qualquer lugar
- Ações: Mover para o espaço vazio em qualquer direção
- Teste de Objetivo = Verifica se as peças estão devidamente sequenciadas
- Custo de Caminho: 1 por movimento

Árvore de busca

- Nosso nó será uma estrutura de dados com cinco componentes:
  - **ESTADO**: o estado no espaço de estados a que o nó corresponde;
  - **NÓ-PAI**: O nó da árvore de busca que gerou esse nó;
  - **AÇÃO**: A ação que foi aplicada ao pai para gerar o nó;
  - **CUSTO-DO-CAMINHO**: O custo, tradicionalmente denotado por g(n), do caminho desde o estado inicial até o nó, indicado pelos ponteiros do pai.

Busca em Árvore

- Descrição informal do algoritmo

  função BUSCA-EM-ÁRVORE(problema, estratégia)
  retorna uma solução ou falha

  Inicializar a árvore de busca usando o estado inicial de problema

  repita

  se não existe nenhum candidato para expansão **então retornar** falha

  escolher um nó para expansão de acordo com a estratégia

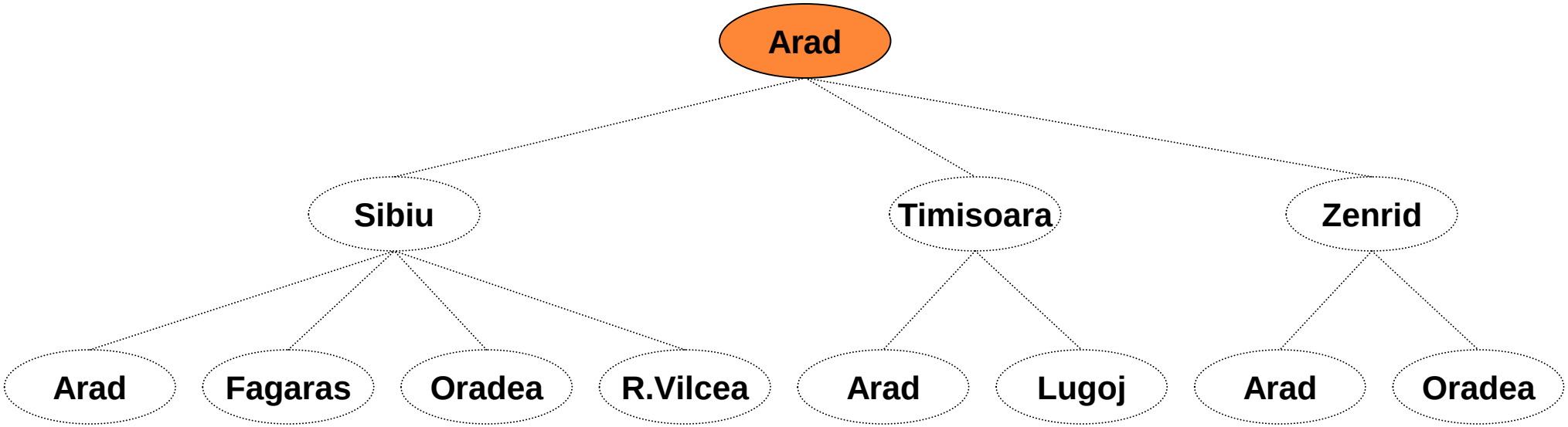
  se o nó contém um estado objetivo **então retornar** solução correspondente

  Senão expandir o nó e adicionar os nós restantes à arvore de busca

Busca em um Espaço de Estados

Grafo do espaço de estados do problema

Árvore de Busca

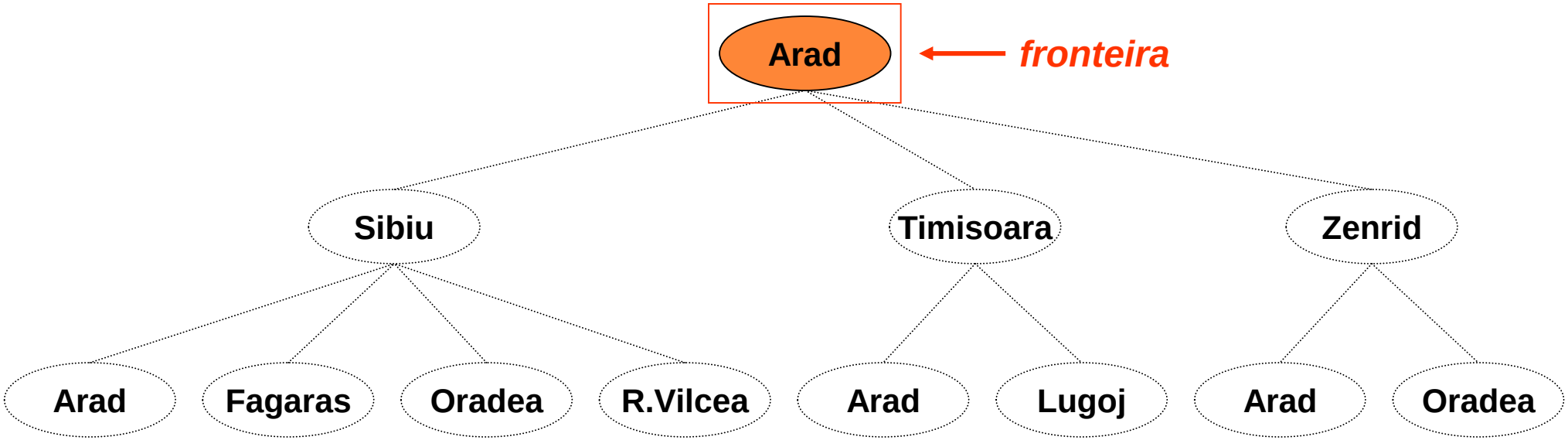


A cada passo, a **árvore de busca** é **expandida** a partir de sua ***fronteira,*** pelos **operadores** definidos na sua **função sucessora**

Busca em um Espaço de Estados

Grafo do espaço de estados do problema

Árvore de Busca

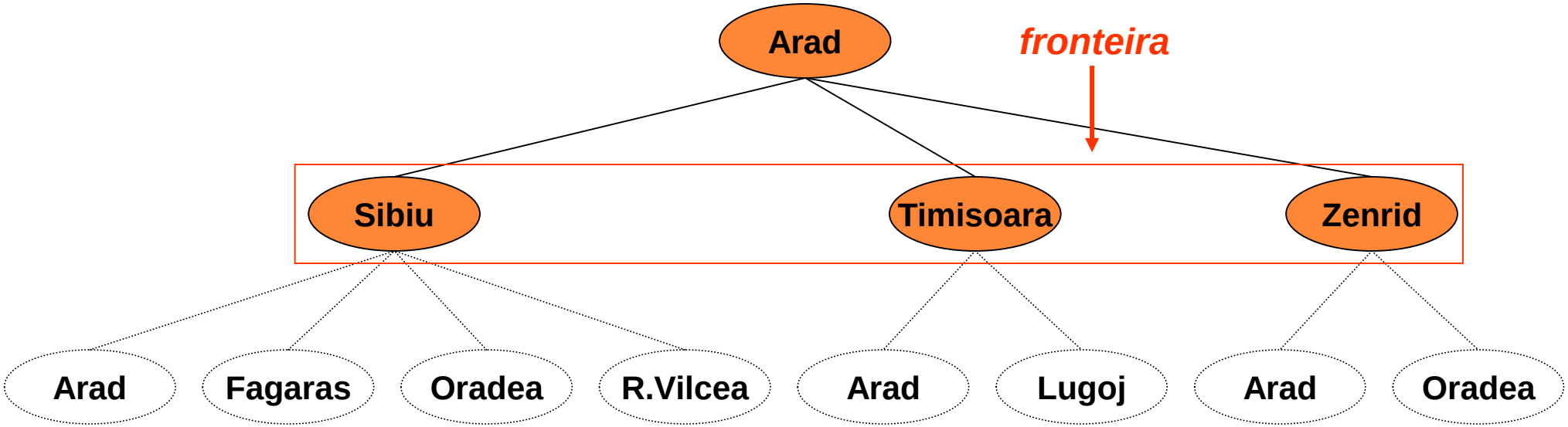


A cada passo, a **árvore de busca** é **expandida** a partir de sua ***fronteira,*** pelos **operadores** definidos na sua **função sucessora**

Busca em um Espaço de Estados

Grafo do espaço de estados do problema

Árvore de Busca

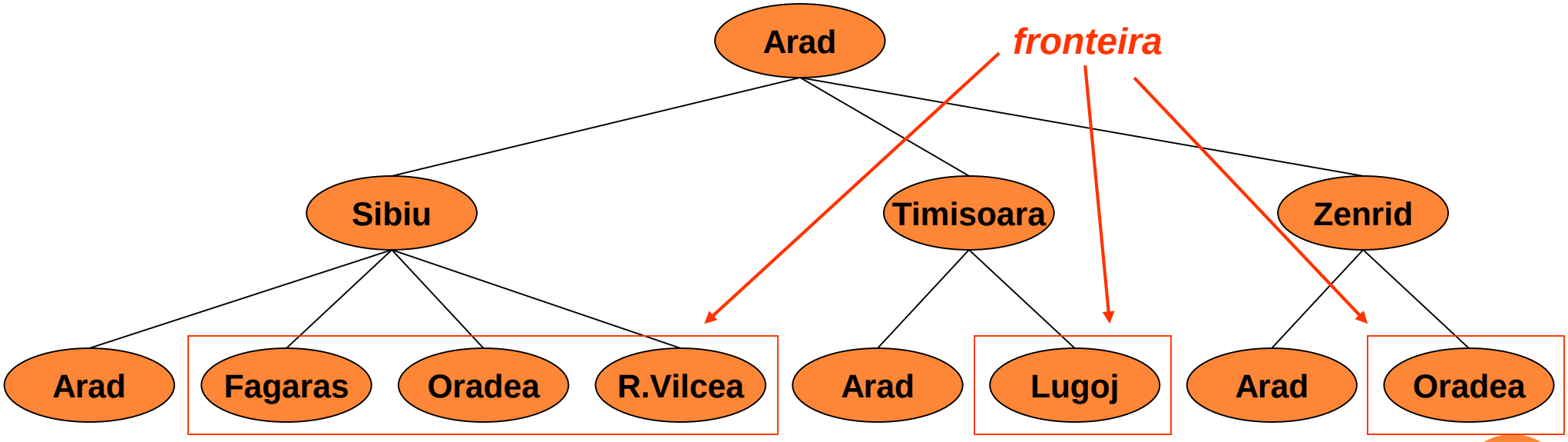


A cada passo, a **árvore de busca** é **expandida** a partir de sua ***fronteira,*** pelos **operadores** definidos na sua **função sucessora**

Busca em um Espaço de Estados

Grafo do espaço de estados do problema

Árvore de Busca



A cada passo, a **árvore de busca** é **expandida** a partir de sua ***fronteira,*** pelos **operadores** definidos na sua **função sucessora**

Desempenho de uma Estratégia de Busca

- **Completude**: se sempre encontra uma solução, se ela existe;
- **Complexidade de Tempo**: em função do nº de nós gerados/expandidos;
- **Complexidade de Espaço**: em função do nº máximo de nós armazenados em memória;
- **Otimização**: se sempre encontra a solução de menor custo

  - Complexidade de tempo e espaço é mensurada em termos de:
    b (*branching*) - fator máximo de ramificação da árvore de busca;
    d (*depth*) – profundidade da solução de menor custo;
    m – profundidade máxima do espaço de estados (que pode ser infinita)

Estratégias de Busca

Busca Cega ou Exaustiva

Métodos de Busca

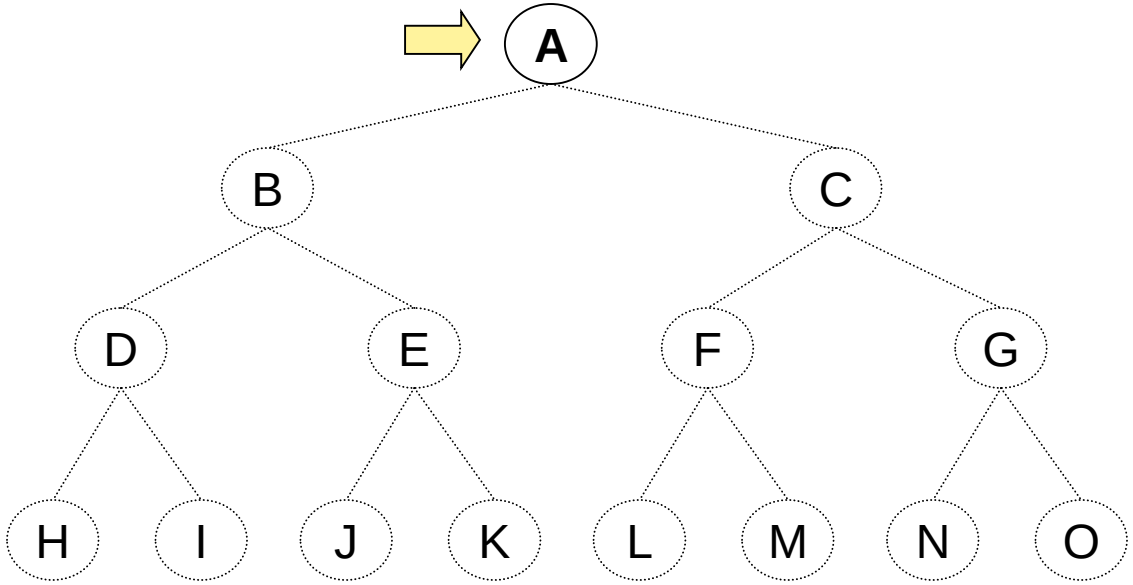
- Exaustiva: Busca em todo espaço de estados, com sequência sistemática de ações que atingem todos os estados. Também chamada de Busca Cega, Sistemática ou Uniforme
- Parcial: Busca num conjunto restrito de espaço de estados a partir do conhecimento a cerca de características específicas do problema ou classe de problema
- Heurística: Estimativa ou conhecimento aproximado sobre a classe ou família em que o algoritmo de busca possa "confiar" para aprimorar o tempo de execução e/ou exigências de espaço;
- Buscas Não-heuristicas não são escalonáveis a grandes problemas. (pior caso é exponencial em tempo e/ou espaço)
- Busca Heurística, parcial não oferece garantia de encontrar uma solução se ela existir ou a melhor solução entre muitas existentes.

Busca Cega

- **Busca em Extensão/Largura (Breadth-first);**
- Busca de Custo Uniforme;
- **Busca em Profundidade (Depth-first);**
- Busca em Profundidade Limitada (Depth-limited)
- Busca de Aprofundamento Iterativo (Iterative deepening)

Busca em Largura (Breadth-first)

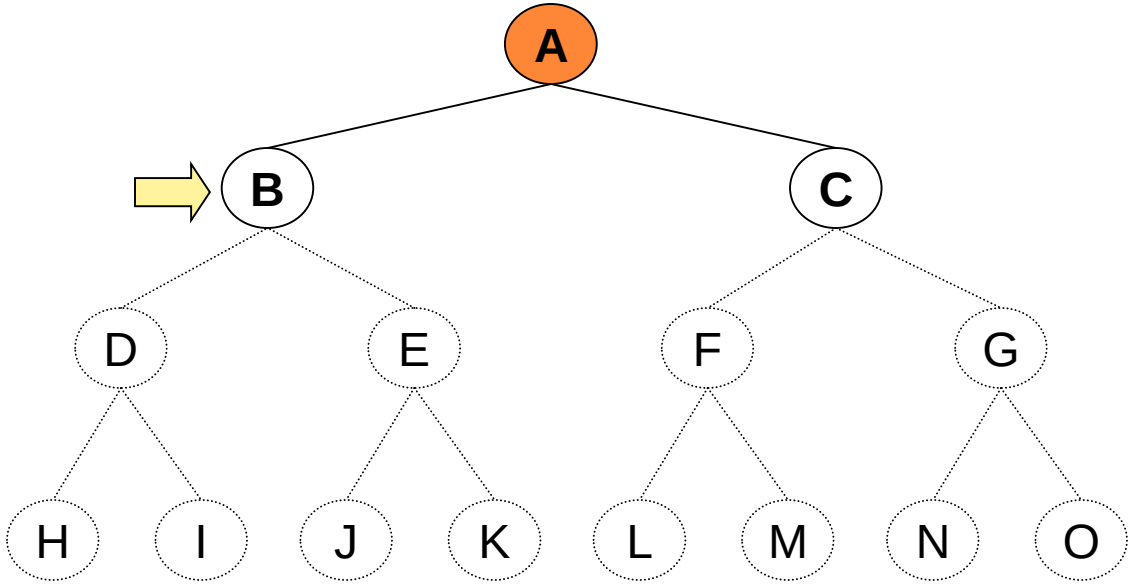
- Expande sempre o nó menos profundo ainda não expandido;
  - ***Fronteira*** é uma **fila do tipo FIFO**, i.e., novos sucessores são postos no fim



Fronteira = (A)

Busca em Largura (Breadth-first)

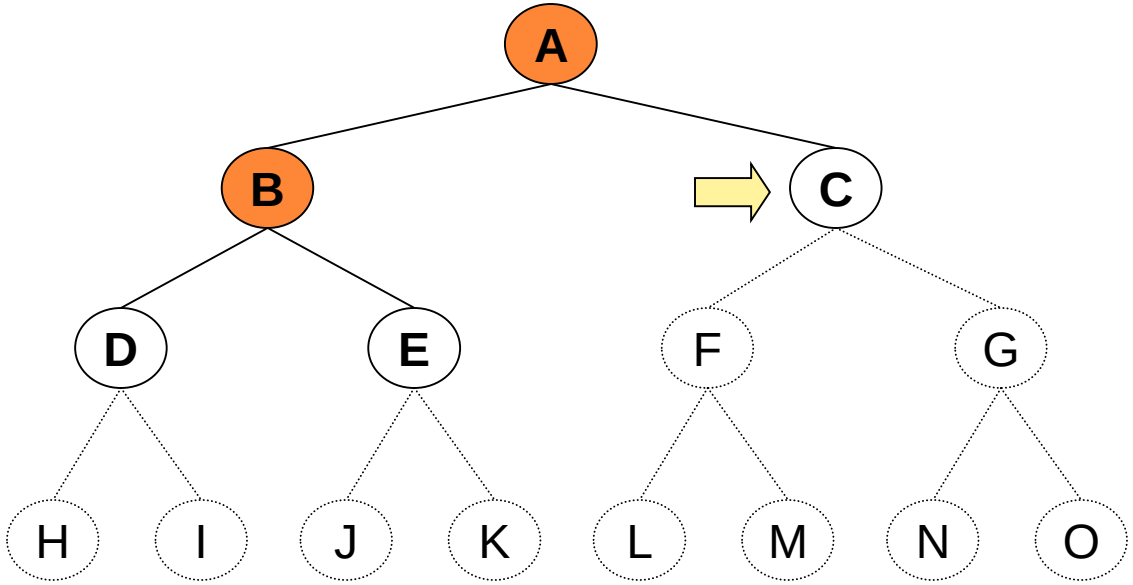
- Expande sempre o nó menos profundo ainda não expandido;
  - ***Fronteira*** é uma **fila do tipo FIFO**, i.e., novos sucessores são postos no fim



Fronteira = (B,C)

Busca em Largura (Breadth-first)

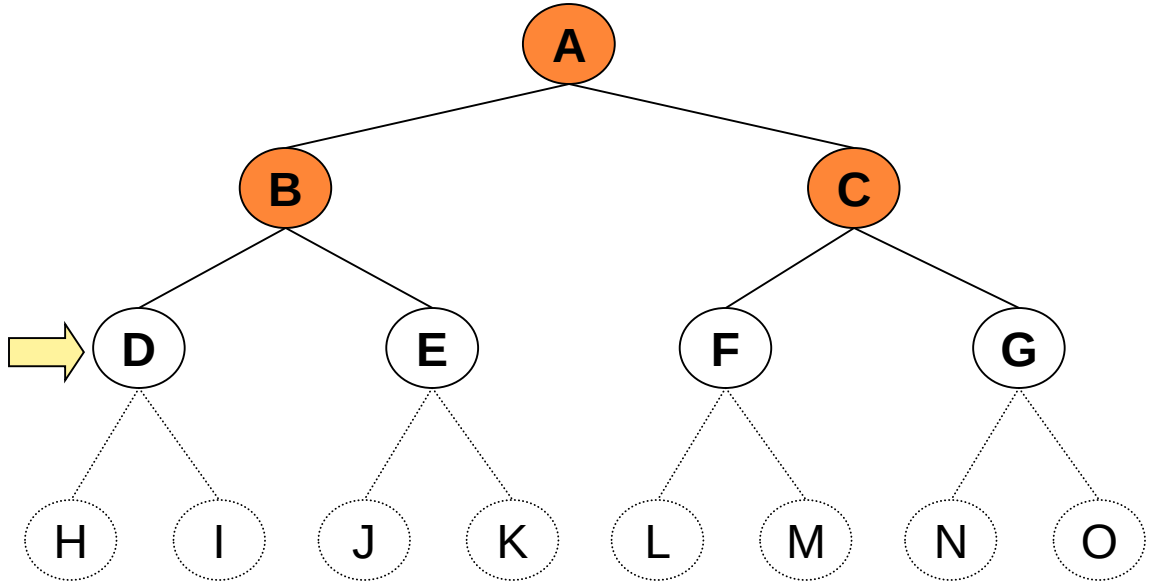
- Expande sempre o nó menos profundo ainda não expandido;
  - ***Fronteira*** é uma **fila do tipo FIFO**, i.e., novos sucessores são postos no fim



Fronteira = (C,D,E)

Busca em Largura (Breadth-first)

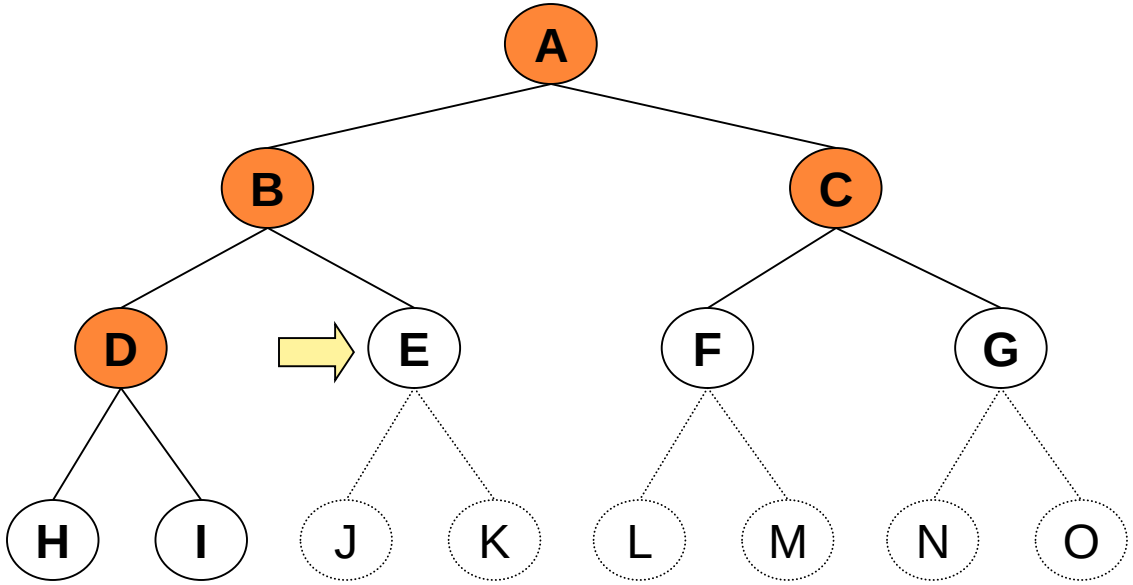
- Expande sempre o nó menos profundo ainda não expandido;
  - ***Fronteira*** é uma **fila do tipo FIFO**, i.e., novos sucessores são postos no fim



Fronteira = (D,E,F,G)

Busca em Largura (Breadth-first)

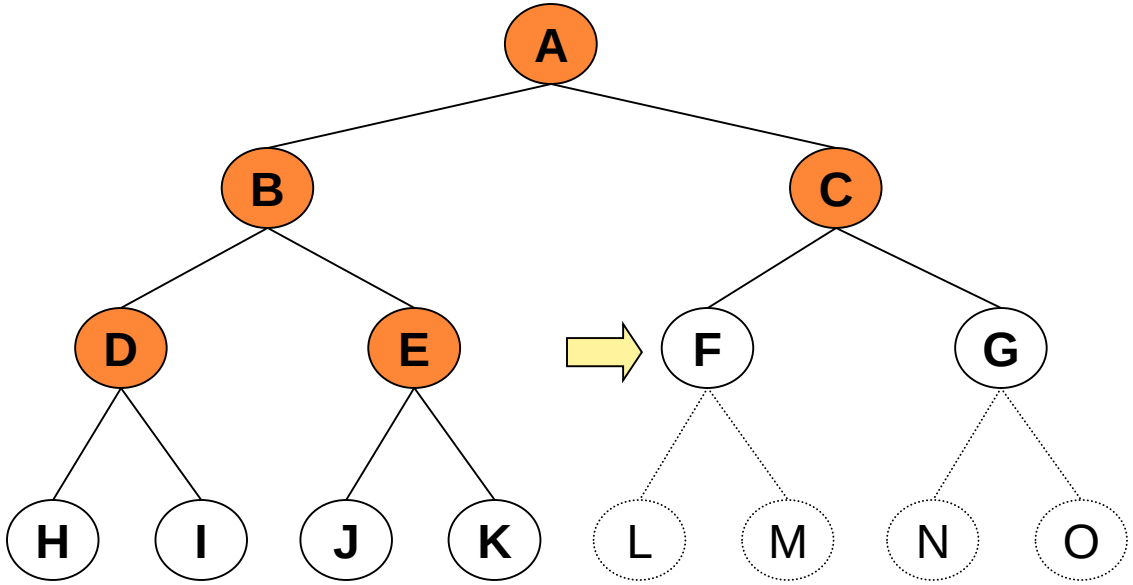
- Expande sempre o nó menos profundo ainda não expandido;
  - ***Fronteira*** é uma **fila do tipo FIFO**, i.e., novos sucessores são postos no fim



Fronteira = (E,F,G,H,I)

Busca em Largura (Breadth-first)

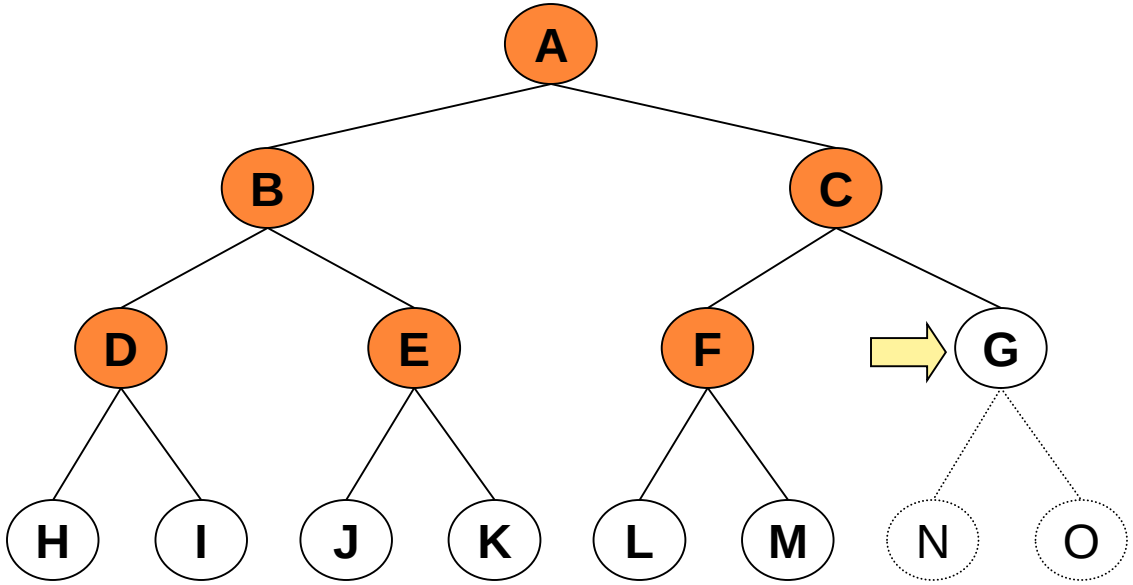
- Expande sempre o nó menos profundo ainda não expandido;
  - ***Fronteira*** é uma **fila do tipo FIFO**, i.e., novos sucessores são postos no fim



Fronteira = (F,G,H,I,J,K)

Busca em Largura (Breadth-first)

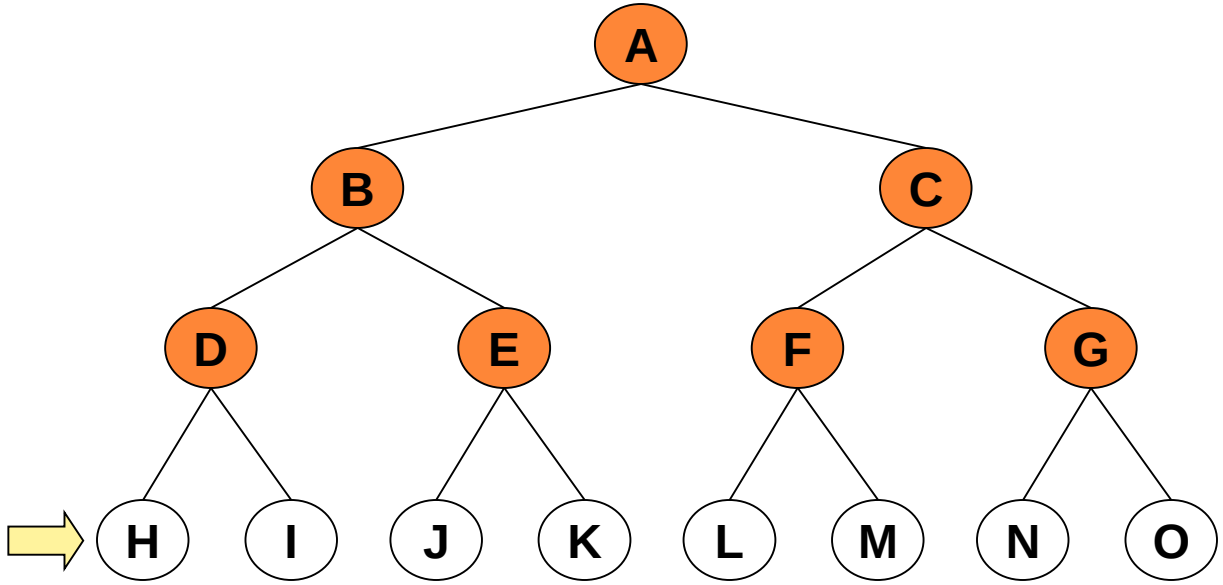
- Expande sempre o nó menos profundo ainda não expandido;
  - ***Fronteira*** é uma **fila do tipo FIFO**, i.e., novos sucessores são postos no fim



Fronteira = (G,H,I,J,K,L,M)

Busca em Largura (Breadth-first)

- Expande sempre o nó menos profundo ainda não expandido;
  - ***Fronteira*** é uma **fila do tipo FIFO**, i.e., novos sucessores são postos no fim



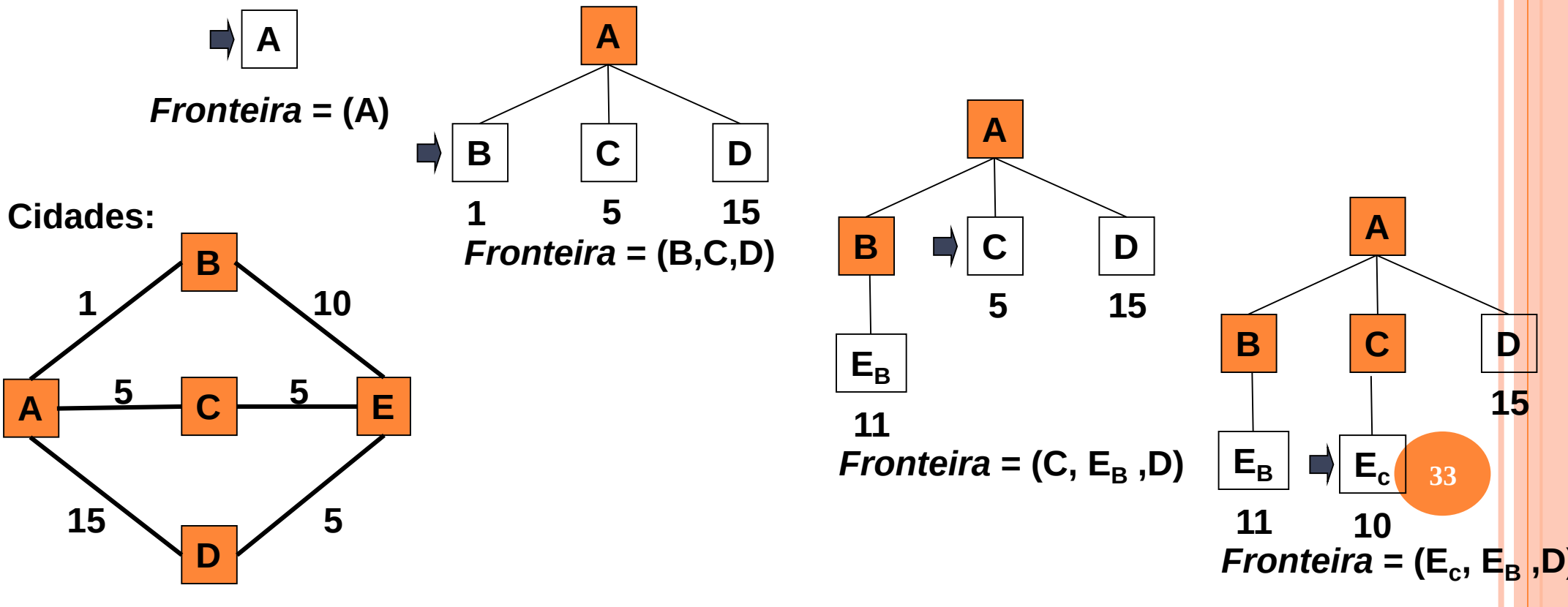
Fronteira = (H,I,J,K,L,M,N,O)

Desempenho da Busca em Largura (Breadth-first)

- Completa?
  - **Sim, desde que *b* (fator de ramificação) seja finito**
- Tempo?
  - **$1 + b + b^2 + b^3 + \ldots + b^d + b(b^d - 1) = O(b^{d+1})$, i.e., exponencial em *d* *(fator de profundidade)***
- Espaço?
  - **$O(b^{d+1})$ (armazena cada nó na memória)**
- Ótima?
  - **Em geral, não. Sim quando o custo é constante a cada passo**
- **Espaço** é o grande problema; pode facilmente gerar nódulos a 10MB/sec, o que em 24h chegaria a 860GB !!!!!

Busca de Custo Uniforme

- Expande sempre o próximo nó ainda não expandido que possui caminho **de menor custo**
  - ***Fronteira*** = fila de nós ordenada pelo custo do caminho até cada nó



Desempenho da Busca de Custo Uniforme

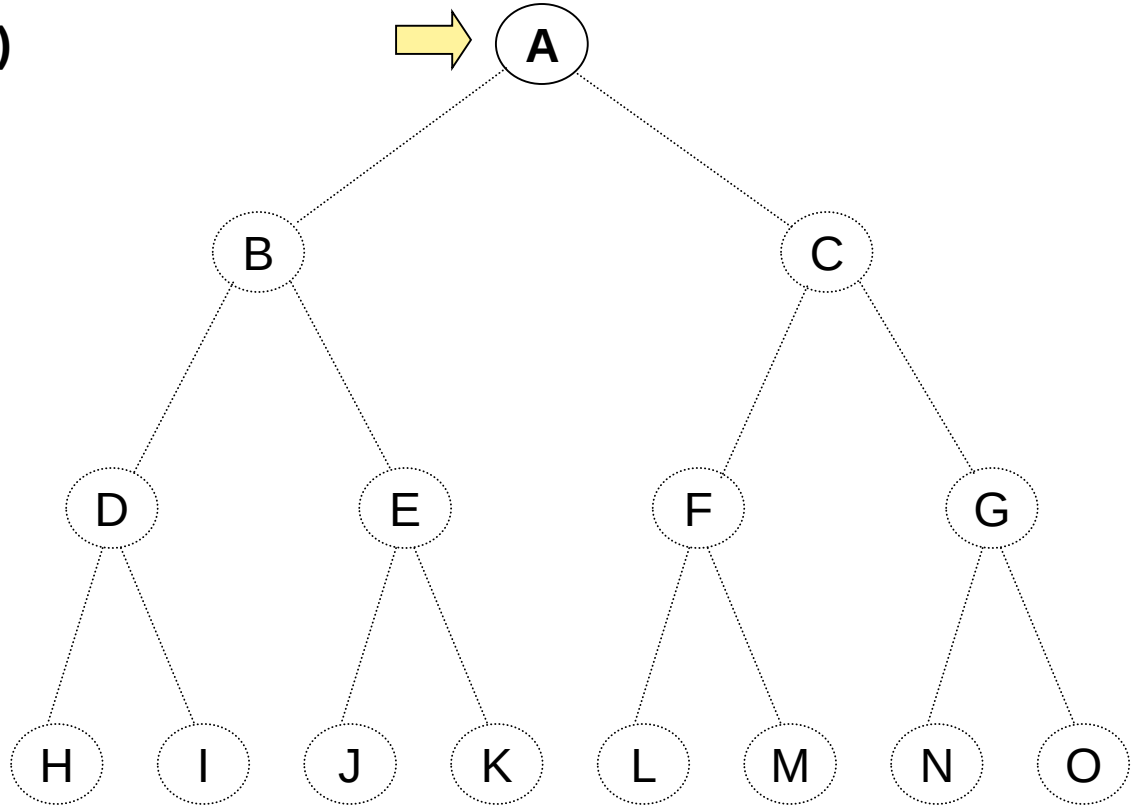
- Completa? **Sim, desde que o custo de cada nó ≠ *0***
- Tempo? **Nº de nós com *custo(nó)* < *custo(solução ótima)***
- Espaço? **Nº de nós com *custo(nó)* < *custo(solução ótima)***
- Ótima? **Sim, já q os nódulos expandem em ordem crescente de *custo(nó)***

Se o custo dos nós de um mesmo nível for igual,
o desempenho é equivalente ao da Busca em Largura

Busca em Profundidade (Depth-first)

- Expande sempre o nó mais profundo ainda não expandido;
  - ***Fronteira*** é uma **fila do tipo LIFO**, i.e., novos sucessores são postos no início

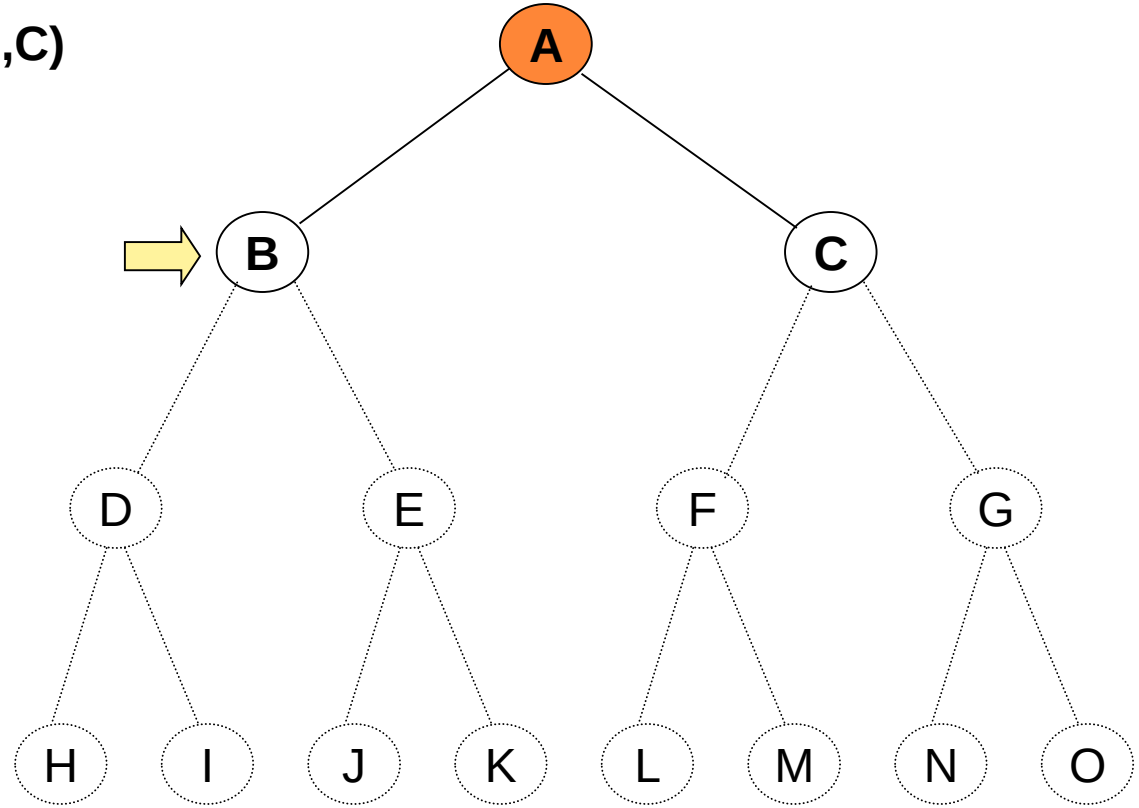
***Fronteira* = (A)**



Busca em Profundidade (Depth-first)

- Expande sempre o nó mais profundo ainda não expandido;
  - ***Fronteira*** é uma **fila do tipo LIFO**, i.e., novos sucessores são postos no início

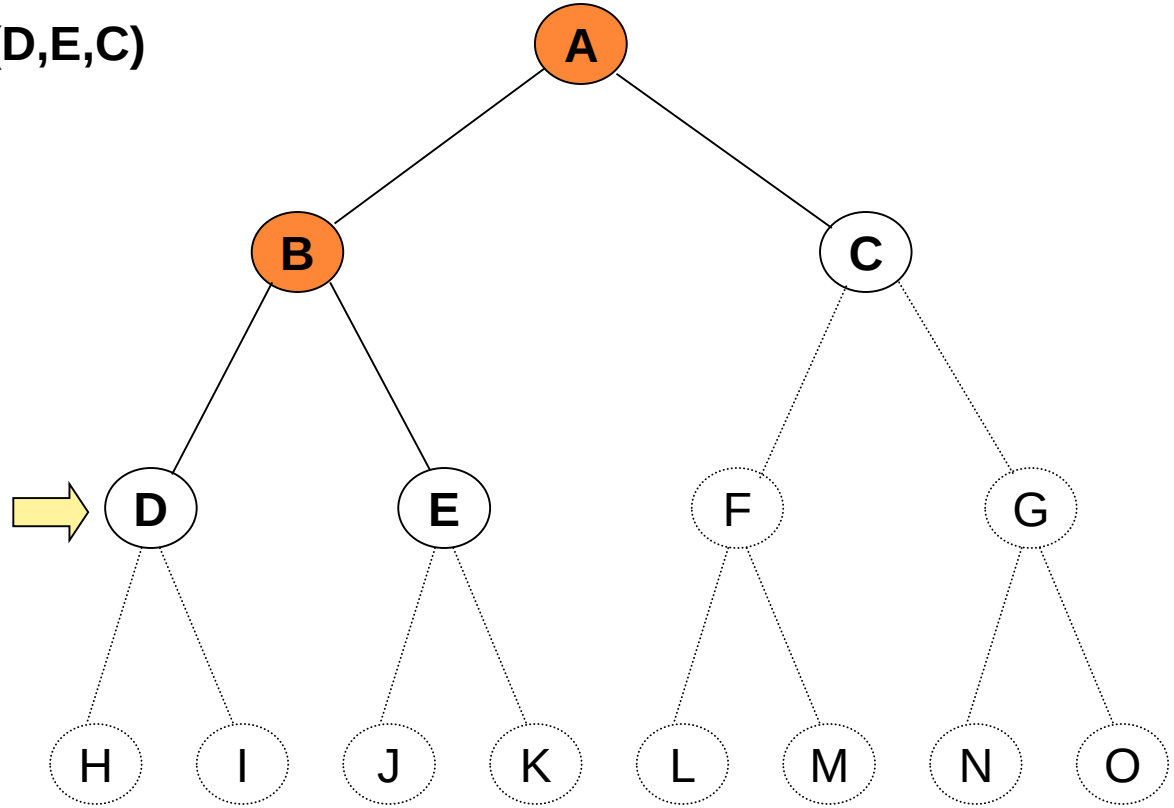
***Fronteira* = (B,C)**



Busca em Profundidade (Depth-first)

- Expande sempre o nó mais profundo ainda não expandido;
  - ***Fronteira*** é uma **fila do tipo LIFO**, i.e., novos sucessores são postos no início

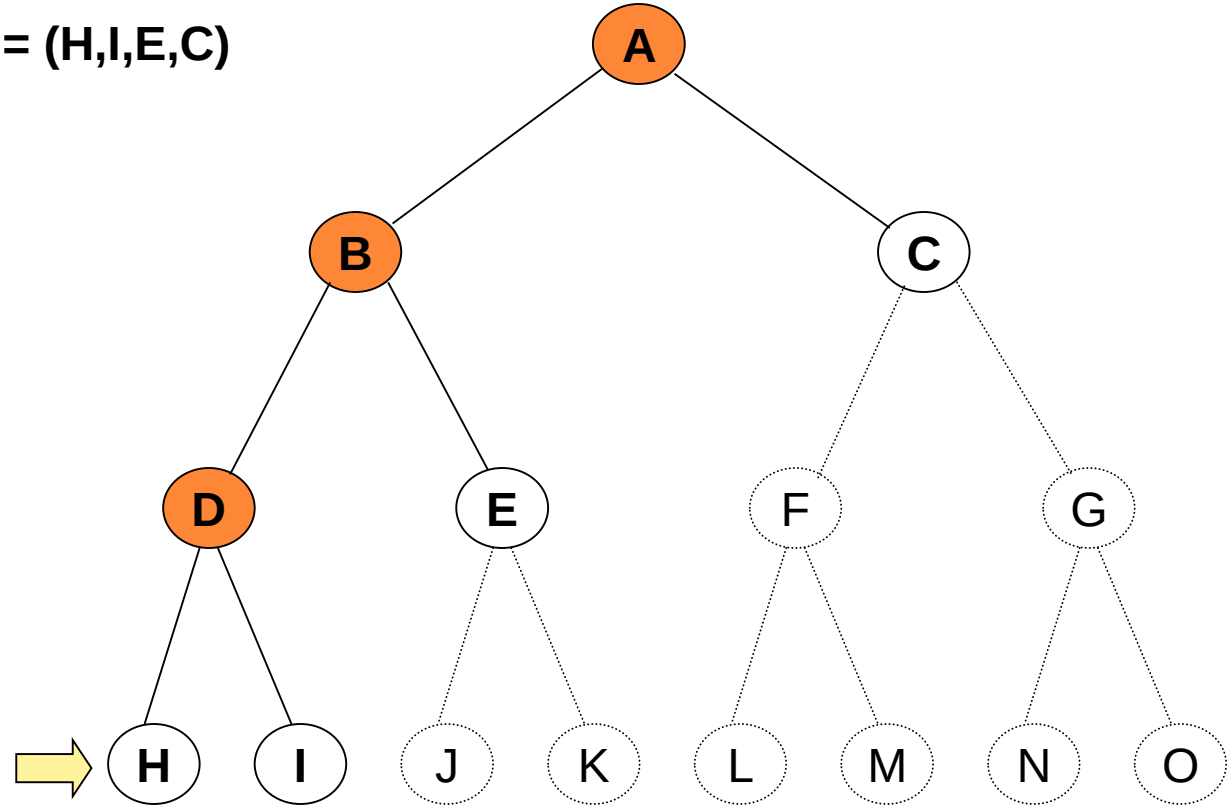
***Fronteira* = (D,E,C)**



Busca em Profundidade (Depth-first)

- Expande sempre o nó mais profundo ainda não expandido;
  - ***Fronteira*** é uma **fila do tipo LIFO**, i.e., novos sucessores são postos no início

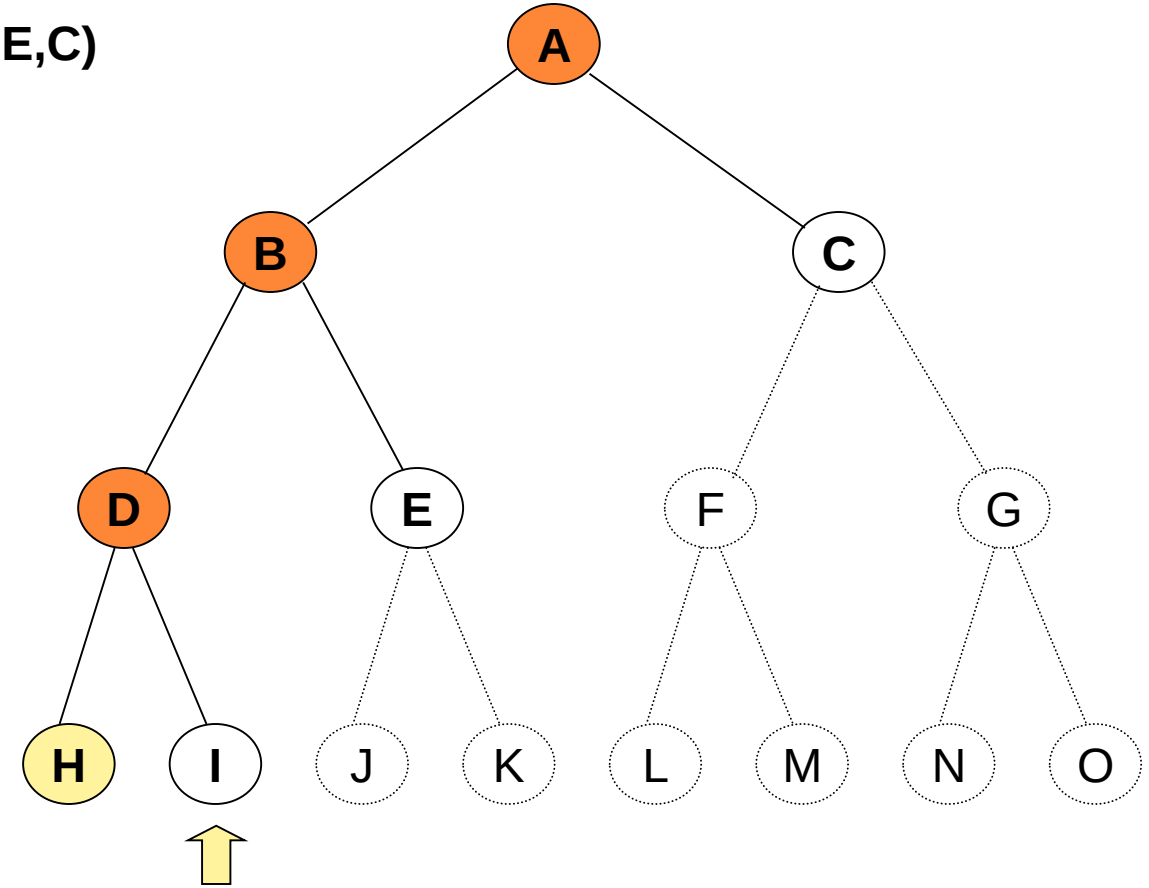
***Fronteira* = (H,I,E,C)**



Busca em Profundidade (Depth-first)

- Expande sempre o nó mais profundo ainda não expandido;
  - ***Fronteira*** é uma **fila do tipo LIFO**, i.e., novos sucessores são postos no início

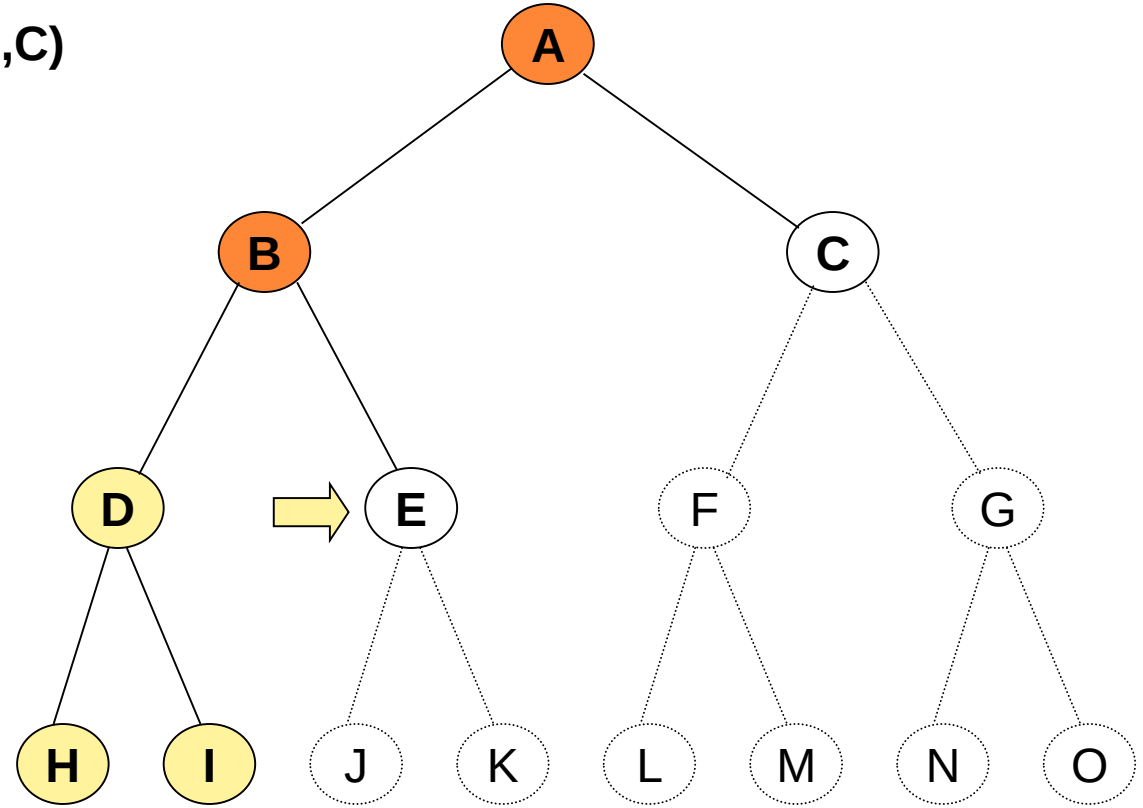
***Fronteira* = (I,E,C)**



Busca em Profundidade (Depth-first)

- Expande sempre o nó mais profundo ainda não expandido;
  - ***Fronteira*** é uma **fila do tipo LIFO**, i.e., novos sucessores são postos no início

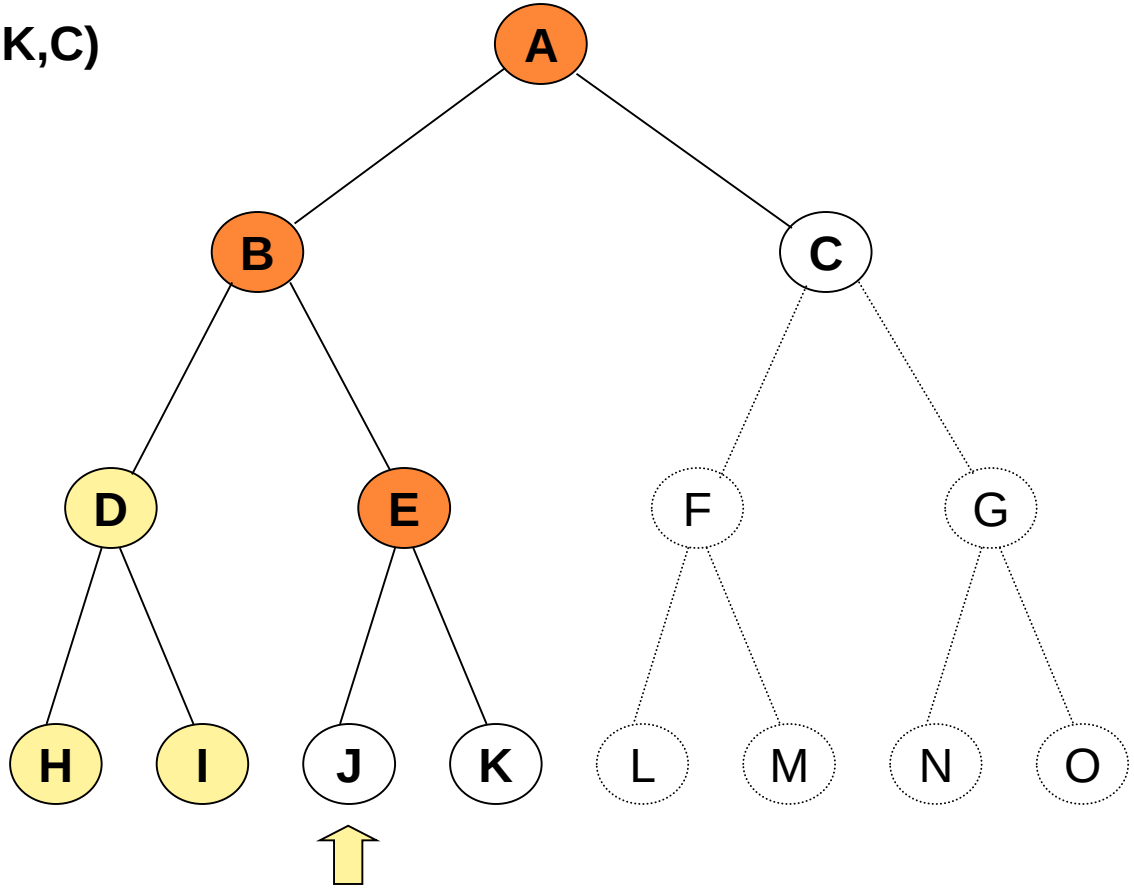
***Fronteira* = (E,C)**



Busca em Profundidade (Depth-first)

- Expande sempre o nó mais profundo ainda não expandido;
  - ***Fronteira*** é uma **fila do tipo LIFO**, i.e., novos sucessores são postos no início

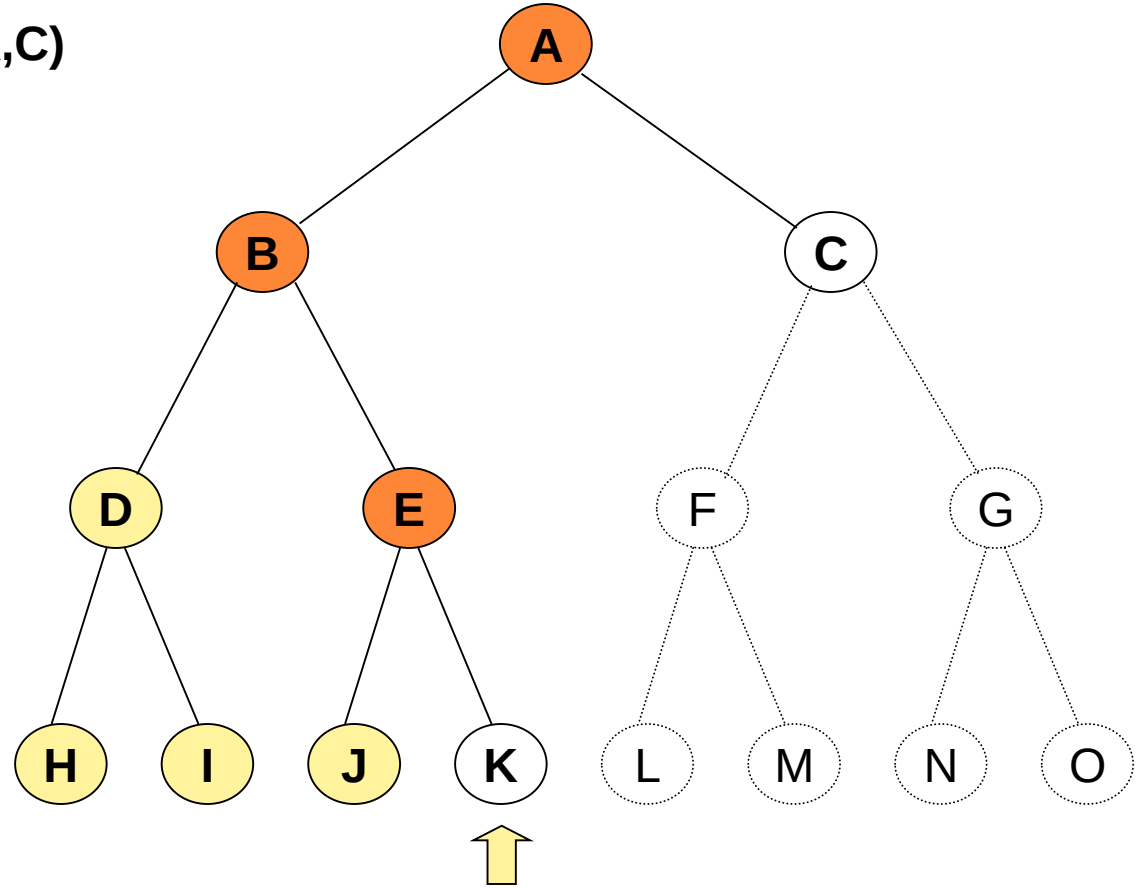
***Fronteira* = (J,K,C)**



Busca em Profundidade (Depth-first)

- Expande sempre o nó mais profundo ainda não expandido;
  - ***Fronteira*** é uma **fila do tipo LIFO**, i.e., novos sucessores são postos no início

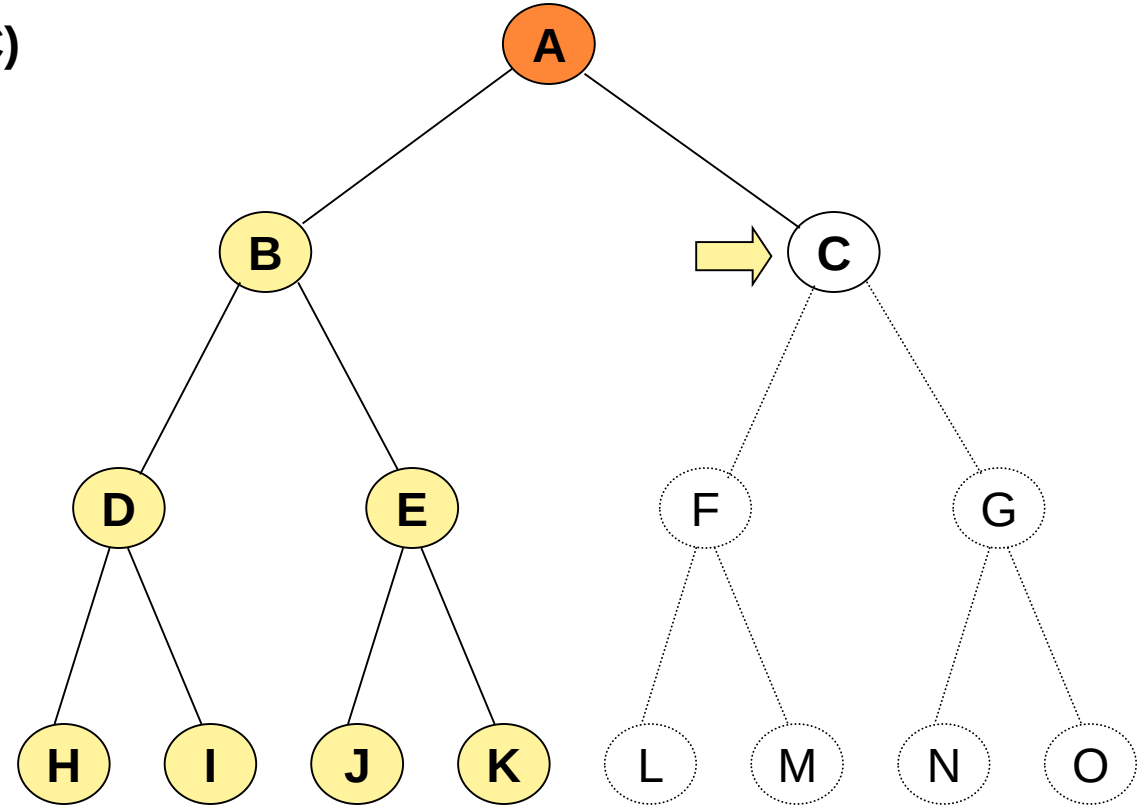
***Fronteira* = (K,C)**



Busca em Profundidade (Depth-first)

- Expande sempre o nó mais profundo ainda não expandido;
  - ***Fronteira*** é uma **fila do tipo LIFO**, i.e., novos sucessores são postos no início

***Fronteira* = (C)**



Desempenho da Busca em Profundidade (Depth-first)

- Completa?

  Não: falha em árvores de profundidade infinita. Nesse caso, pode-se arbitrar um limite de profundidade *L:* *Depth-limited search*

- Tempo?

  O*(*b*^{m}): muito ruim se *m* >> *d *(m profundidade máxima, d profundidade da solução)*

- Espaço?

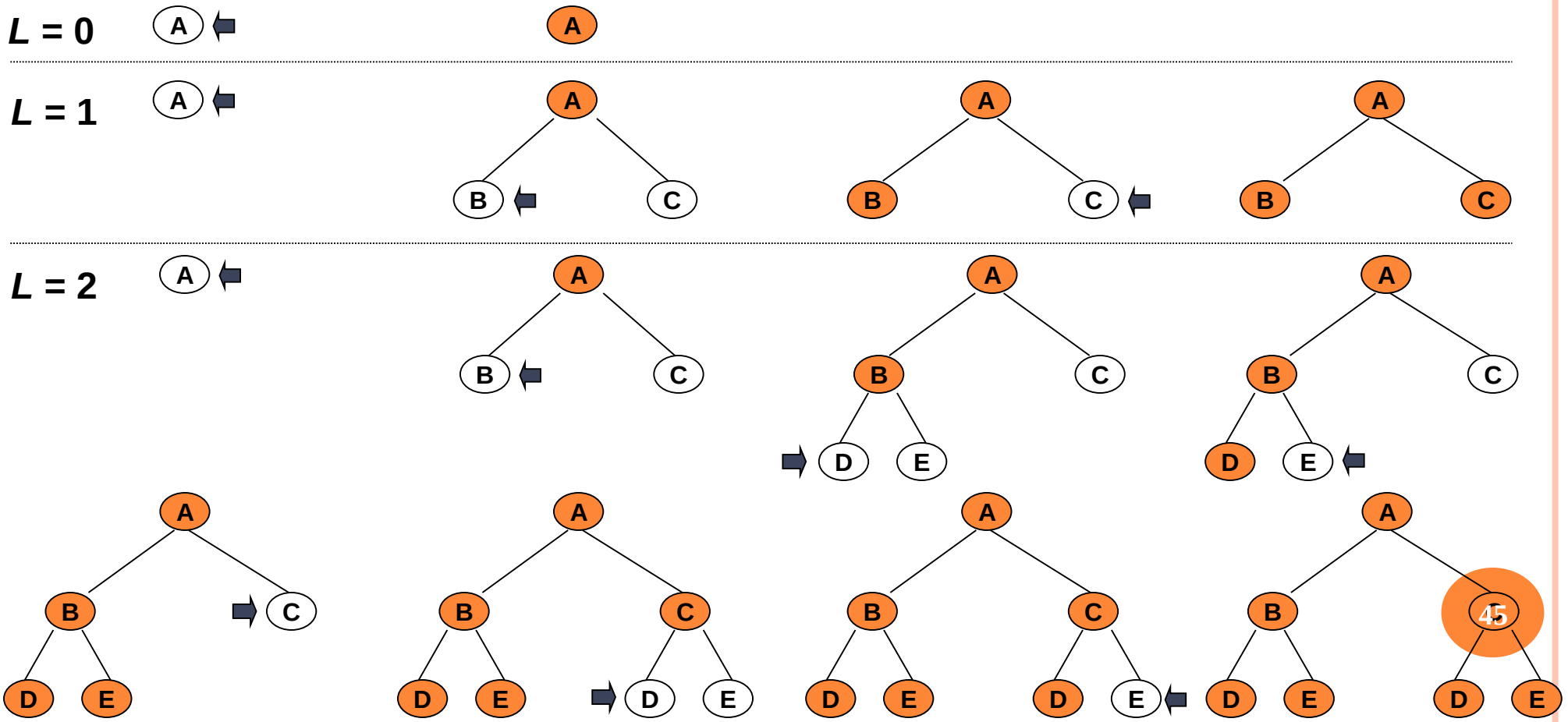
  ***O*(*b.m*), i.e., função de crescimento linear**

- Ótima?

  Não: deve ser evitada em árvores muito profundas ou profundidade infinita

Busca de Aprofundamento Iterativo

- Variação da Busca em Profundidade, que utiliza um limite de profundidade ***L***, que inicia em 0 e vai sendo incrementado de 1, a cada iteração.

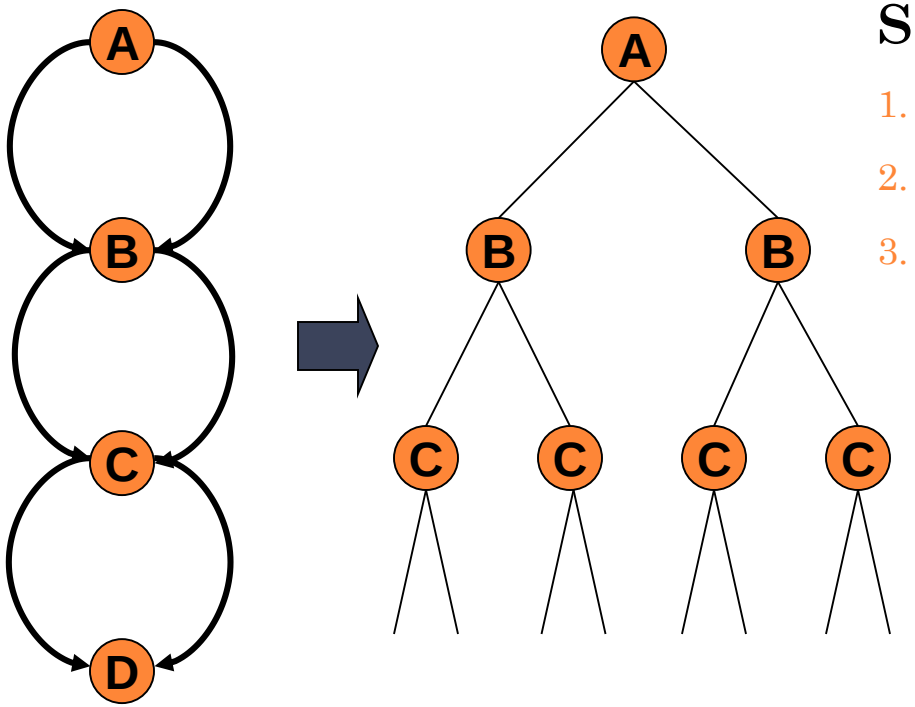


Desempenho da Busca por Aprofundamento Iterativo

- Completa? **Sim, desde que b *(fator de ramificação)* seja finito**
- Tempo? ***O(b^d)*** *(b fator de ramificação, d profundidade do nó objetivado)*
- Espaço? ***O(b.d)*** *(b fator de ramificação, d profundidade do nó objetivado)*
- Ótima? **Em geral, não. Sim quando o custo é constante a cada passo**

Em geral, é o método de *Busca Cega* preferido para grandes espaços de busca e quando a profundidade da solução é desconhecida

Repetição de Estados na Busca



Soluções possíveis (Custo x Eficácia)

1. Não retornar ao estado "pai"
2. Não retornar a um ancestral
3. Não gerar qualquer estado que já tenha sido criado antes (em qualquer ramo)
   - requer que todos os estados gerados permaneçam na memória: custo $O(b^d)$
   - pode ser implementado mais eficientemente com *hash tables*
   - quando encontra nó igual tem de escolher o melhor (menor custo de caminho até então)