Inteligência Artificial

Aula #2: Resolução de Problemas Via Busca

Prof. Eduardo R. Hruschka

Agenda

- Tipos de Problemas
  - Estados únicos (totalmente observável)
  - Informação parcial
- Formulação do Problema
- Algoritmos de Busca Básicos
  - Não informados

  - Leitura Recomendada:
    - Russell, S., Norvig, P., Artificial Intelligence – A Modern Approach, Second Edition, Prentice Hall - Capítulo 3 **– Solving Problems by Searching**.

Créditos e Agradecimentos

Adaptado das notas de aula de Tom Lenaerts,

Vlaams Interuniversitair Instituut voor Biotechnologie

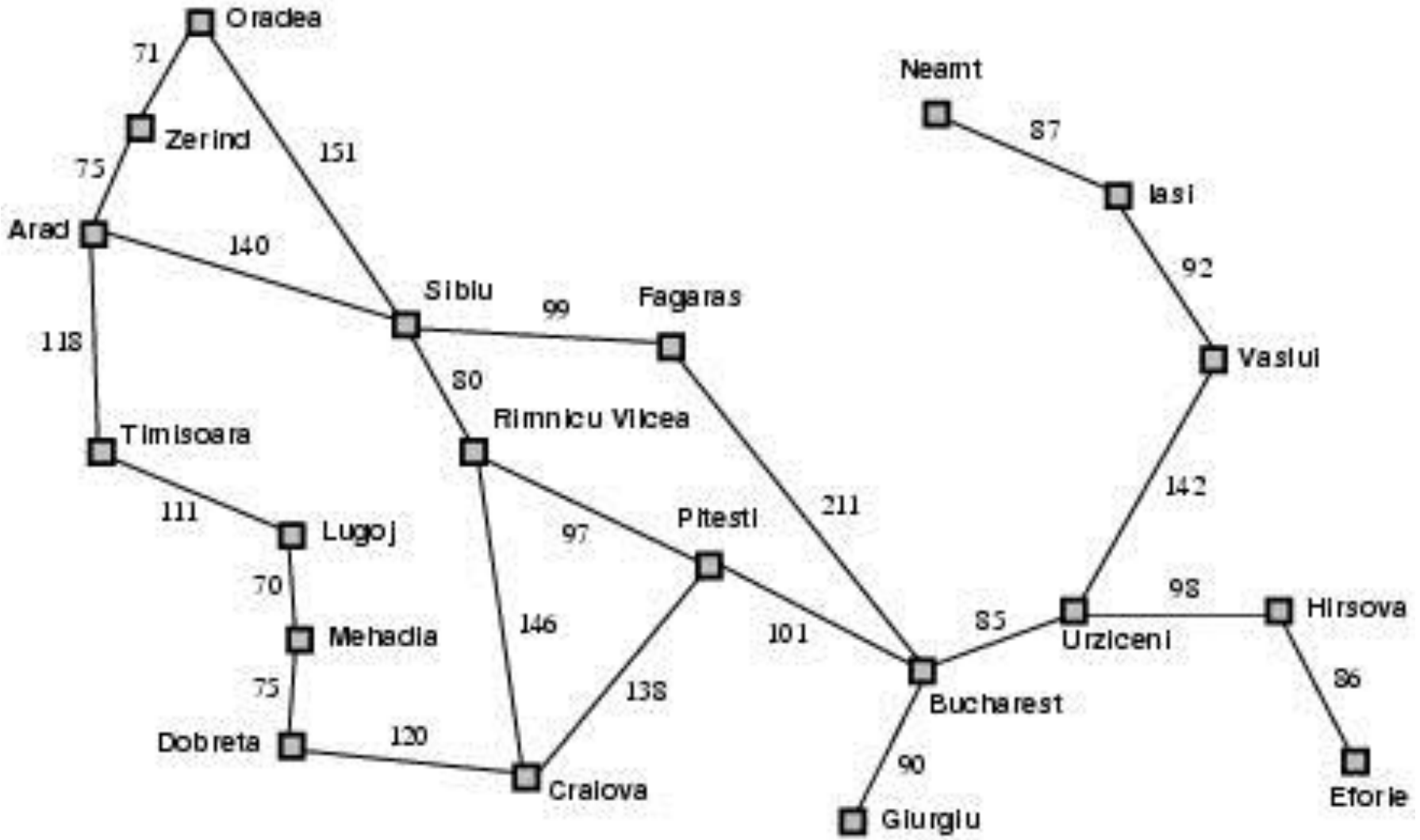
- **http://aima.cs.berkeley.edu**

- Os slides traduzidos foram gentilmente cedidos pelo Prof. Ricardo J. G. B. Campello.

Introdução

- Estudaremos um paradigma para resolução de problemas;
- Encontrar soluções por meio da geração sistemática de novos estados, os quais são testados a fim de se verificar se correspondem à solução do problema;
- Assume-se que o *raciocínio* se reduz à busca;
- Abordagem eficiente para uma série de problemas práticos (principalmente nas versões *informadas* – próxima aula).

Exemplo: Mapa da Romênia



Exemplo: Mapa da Romênia

- Férias na Romênia, atualmente em Arad
  - Vôo sai de Bucareste
- Formulação da Meta
  - Estar em Bucareste
- Formulação do Problema
  - Estados: (estar em) cada cidade representa um estado.
  - Ações: dirigir de uma cidade para outra.
- Solução do Problema
  - Seqüência de cidades; e.g. Arad, Sibiu, Fagaras, Bucareste.

Solução de Problemas por Busca

- Quatro passos gerais:
  - Formulação da meta
    - Qual ou quais estados correspondem à solução do problema?
  - Formulação do problema
    - Quais ações e estados considerar dada a meta?
  - Busca pela solução
    - Encontre uma seqüência (ou a melhor das seqüências) de ações que leve à meta.
  - Execução
    - Implemente as ações.

Tipos de Problemas

- Determinista e Totalmente Observável:
  - Sabe-se exatamente em que estado está e em qual irá estar após uma ação ⇒ *estados únicos*.
- Conhecimento Parcial dos Estados e Ações:
  - Não observável ⇒ *sem sensoriamento*
    - Pode-se não ter idéia de onde se está; busca em "belief states".
  - Não determinista e/ou não totalmente observável ⇒ *problemas de contingências*
    - Resultados das ações podem ser incertos; e/ou
    - Dispõe-se de *nova* informação ao longo do procedimento de solução;
    - Solução usualmente intercala busca e execução (*planning*).
  - Espaço de estados desconhecido ⇒ *problemas de exploração*
    - Estados devem ser explorados "on line" (e.g. robótica móvel).

Formulação do Problema

- Um problema é definido por:
  - Um Estado Inicial, e.g. *Arad.*
  - Função Sucessora *S(x)* = conjunto de pares ação-estado
    - e.g. *S(Arad)={<Arad → Zerind, Zerind>,…}*

    Estado inicial + função sucessora = espaço de estados
  - Teste de Meta, que pode ser
    - Explícito, e.g. *x == 'Bucareste'*
    - Implícito, e.g. *chequemate(x)*
  - Custo de Caminho (aditivo)
    - E.g., soma de distâncias, número de ações executadas, …
    - *c(x,a,y)* é o custo do passo (a partir do estado *x* para o estado *y* através da ação *a*), por premissa não negativo.

  Uma Solução é uma seqüência de ações do estado inicial para o estado meta.

  Uma Solução Ótima é tal que possui o menor custo de caminho.

Espaço de Estados

- Mundo real é muito complexo.

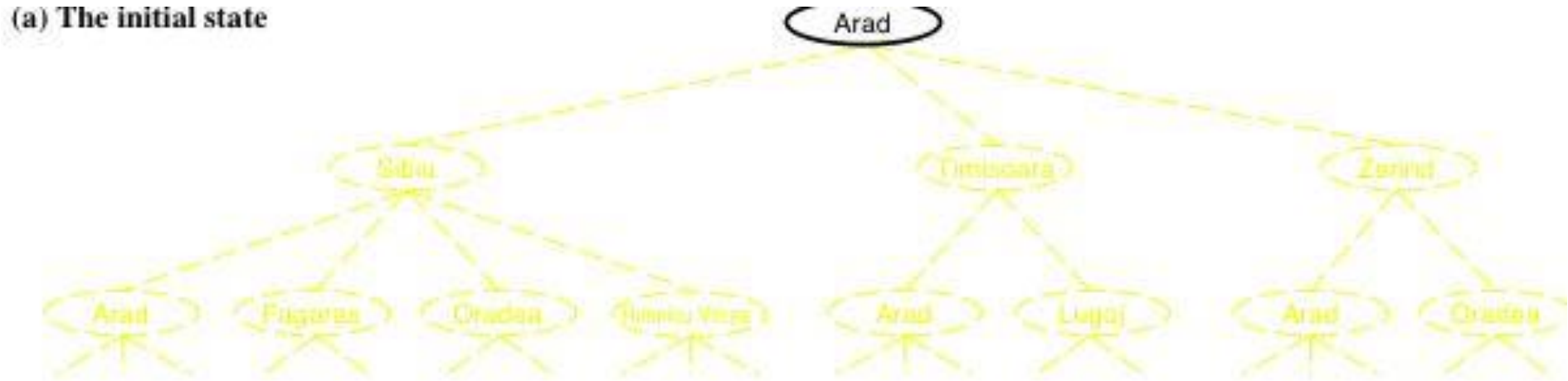
  Espaços de estados e de ações devem ser *abstraídos*.

  - e.g. Arad → Zerind representa um conjunto complexo de possíveis rotas, desvios, paradas de abastecimento, etc.
  - A abstração é válida se o caminho entre dois estados é refletida no mundo real.
- Solução Abstrata = Conjunto de caminhos reais que são soluções no mundo real.
- Cada ação abstrata deve ser "mais fácil" do que no problema real.

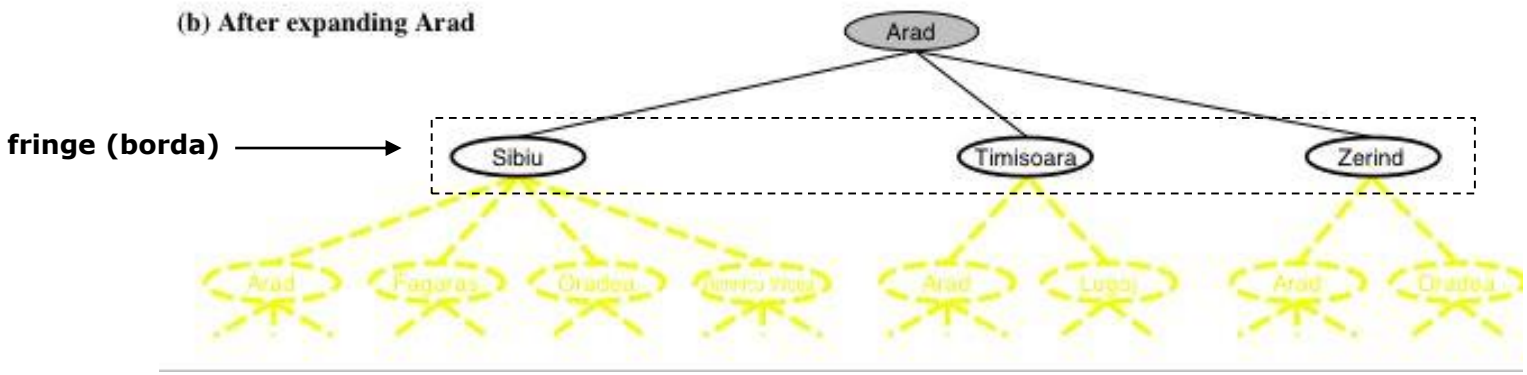
Algoritmos de Busca Básicos

- Soluções para os problemas anteriores?
  - Buscar no espaço de estados;
  - Nos concentraremos na busca através de *geração explícita de árvore*:
    - Raiz = estado inicial.
    - Demais nodos gerados através da função sucessora.
  - Em geral a busca se dá, na verdade, sobre um **grafo** (mesmo estado alcançado por múltiplos caminhos)

Exemplo de Busca em Árvore

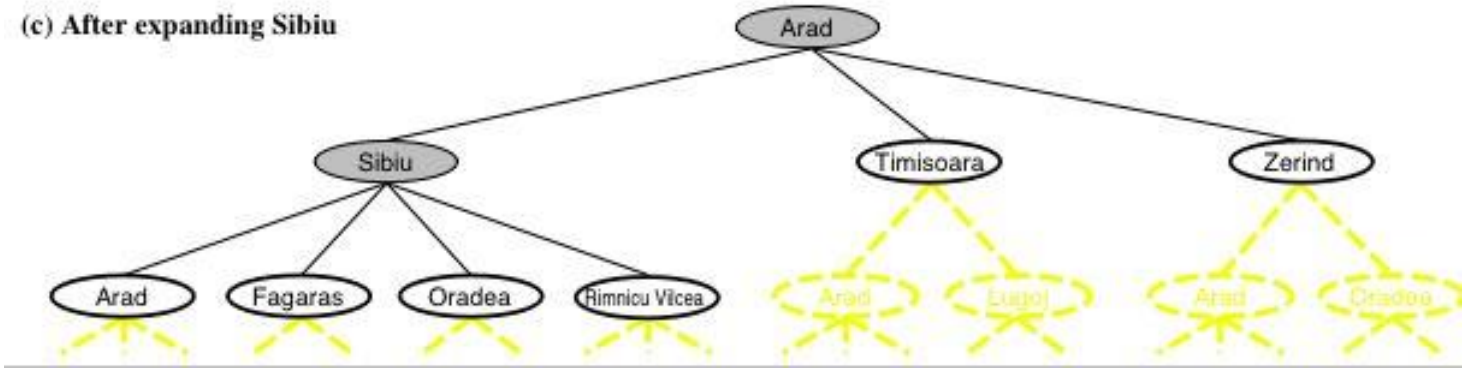


Exemplo de Busca em Árvore



FRINGE = conjunto de nodos que ainda não foram expandidos.

Exemplo de Busca em Árvore



Estratégias de Busca

- Uma estratégia de busca é basicamente uma ordem de expansão dos nodos.
- Medidas de desempenho para diferentes estratégias:
  - Completude: *Sempre encontra uma solução (se existir)?*
  - Otimalidade: *Sempre encontra a solução de custo mais baixo?*
  - Complexidade (tempo): *Número de nodos explorados?*
  - Complexidade (espaço): *Número de nodos armazenados?*

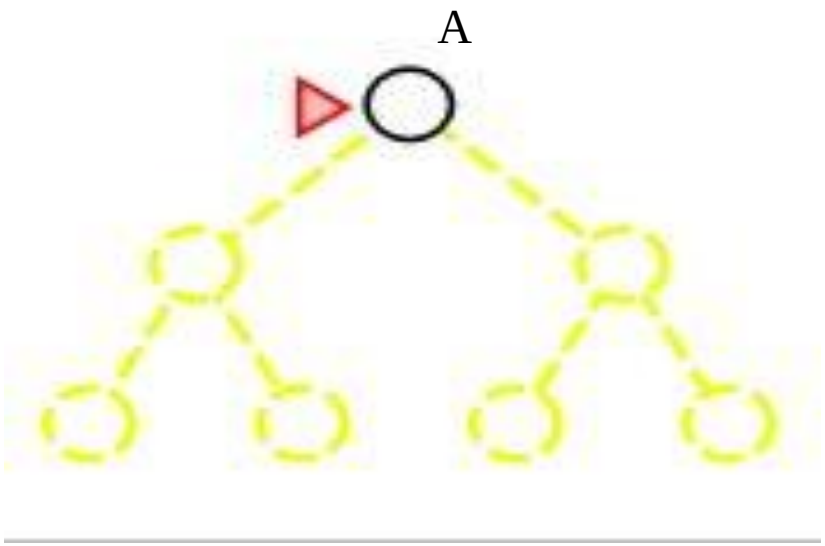
- Complexidade usualmente medida em função da dificuldade do problema:
  - *b : fator de ramificação máxima da árvore de busca.*
  - *d : profundidade da solução de menor custo.*
  - *m : máxima profundidade do espaço de estados (pode ser ∞).*

Estratégias Não-Informadas

- Também denominadas de busca cega: usam estritamente a informação disponível na formulação do problema.
  - Quando é possível utilizar informação adicional para determinar se um nodo não-meta é mais promissor do que outro → busca informada.
- Diferenciam-se pela abordagem de expansão:
  - Busca em largura (*Breadth-first search*).
  - Busca uniforme (*Uniform-cost search*).
  - Busca em profundidade (*Depth-first search*)
  - Busca em profundidade limitada (*Depth-limited search*)
  - Busca em profundidade iterativa (*Iterative deepening search*).

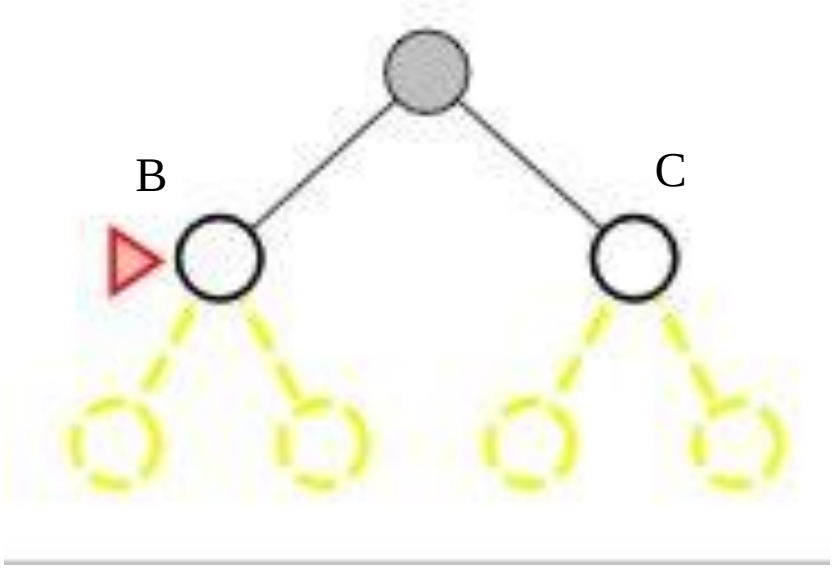
Busca em Largura

- Expande o nodo não expandido mais raso.
- Implementação: *fringe* como uma fila FIFO.
- Exemplo:



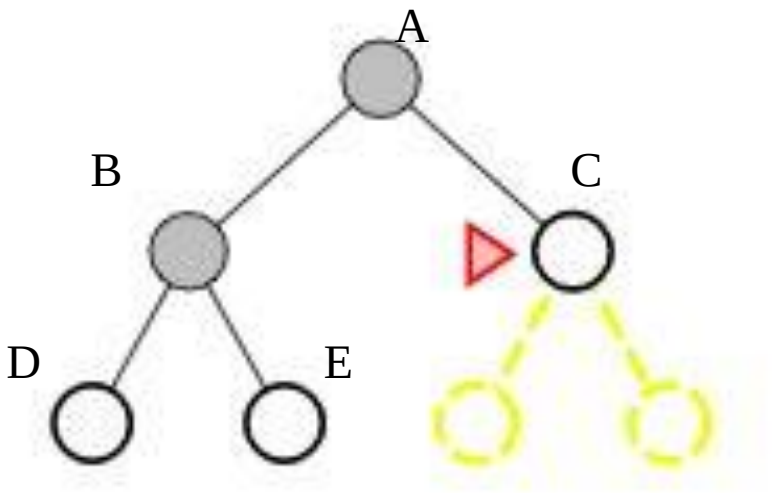
Busca em Largura

- Exemplo:



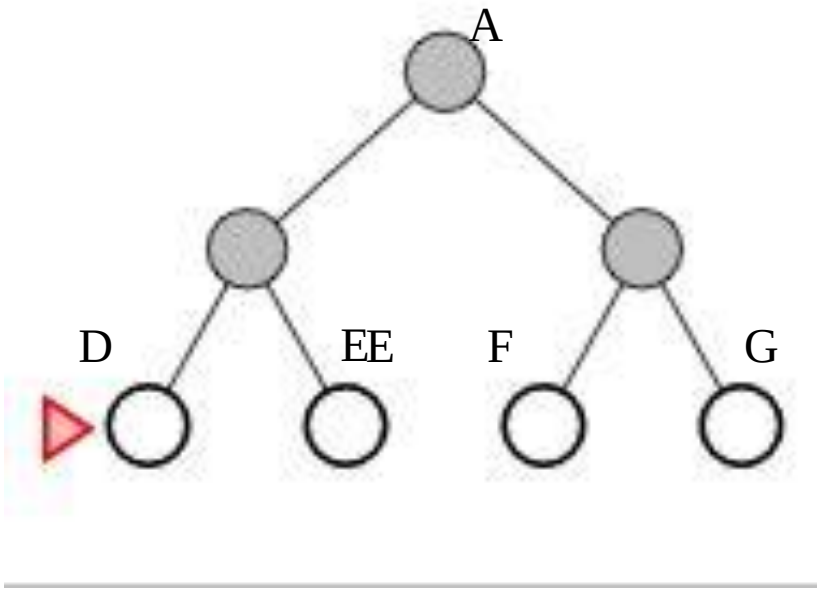
Busca em Largura

- Exemplo:



Busca em Largura

- Exemplo:



Busca em Largura

- Completude:
  - *Sempre encontra uma solução?*
  - SIM (se existir)
    - Se o nó meta mais raso estiver em profundidade finita *d*.
    - Condição: *b* finito (máx. no. nodos sucessores finito).
- Otimalidade:
  - *Sempre encontra a solução de menor custo ?*
  - Apenas se os custos dos caminhos até uma dada profundidade forem iguais ou menores do que aqueles para profundidades maiores (e.g. se todas as ações possuem o mesmo custo).

Busca de Custo Uniforme

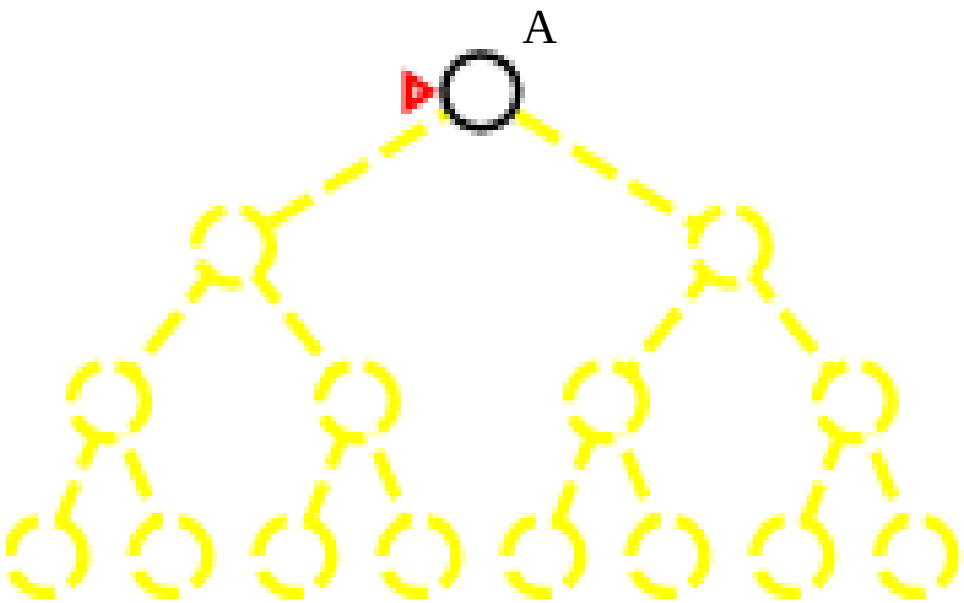
- Extensão da Busca em Largura:
  - Expande o nodo com o *menor custo de caminho*.
- Implementação:
  - *fringe* = fila de prioridade com chave dada pelo custo.

- Busca de Custo Uniforme recai na Busca em Largura quando todos os custos de passo (custo das ações) são iguais.

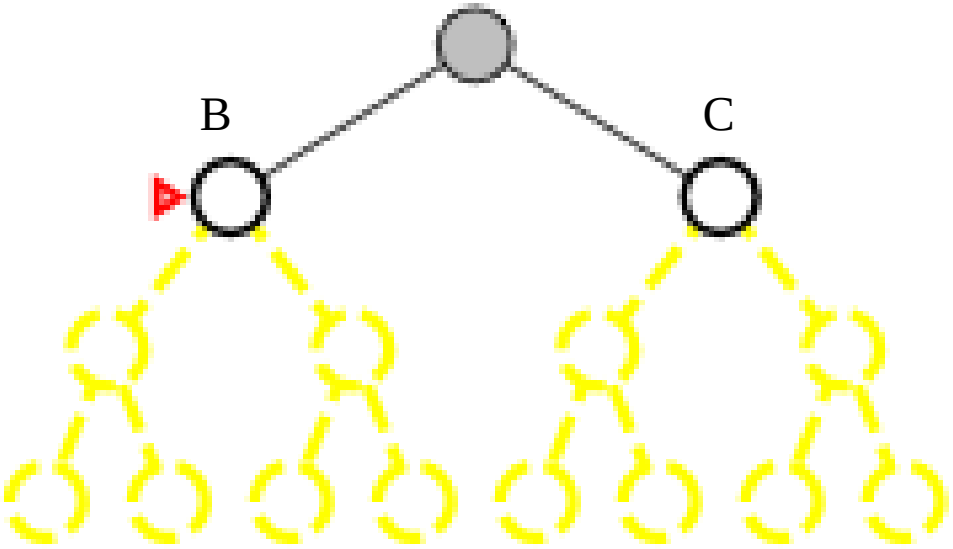
- Completude e Otimalidade:
  - SIM, caso o custo de passo for positivo (o que implica que nodos serão expandidos em ordem crescente de custo de caminho).

Busca em Profundidade

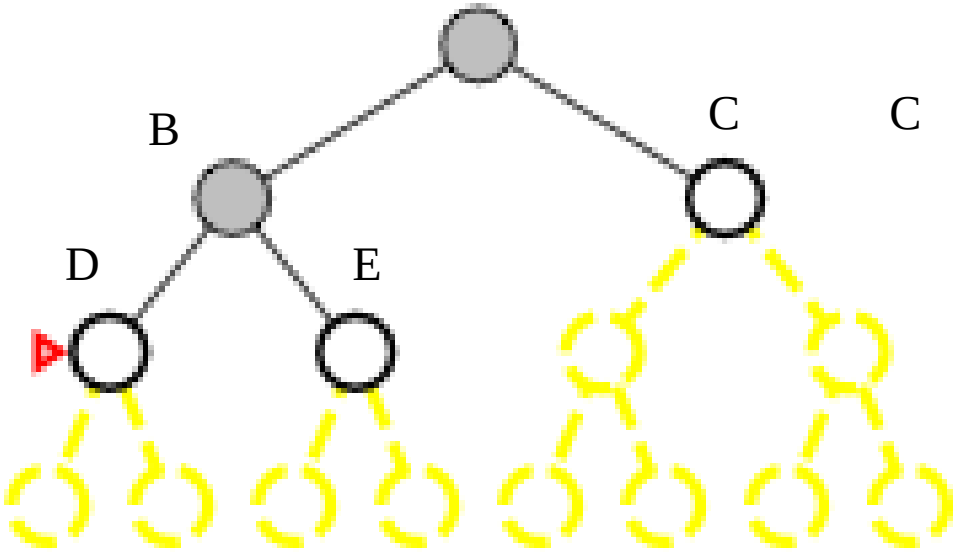
- Expande o nodo não expandido mais profundo.
- Implementação: *fringe* (nós a serem expandidos) como uma pilha.



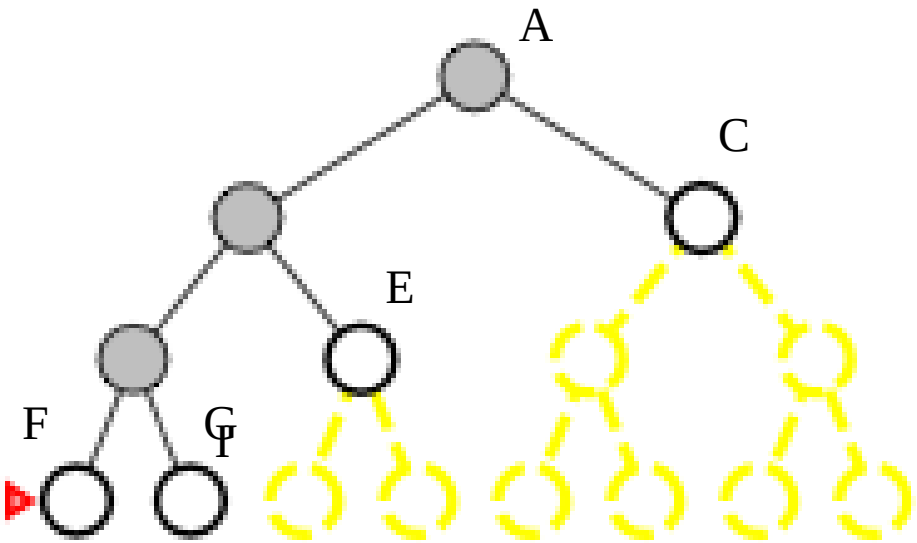
Busca em Profundidade



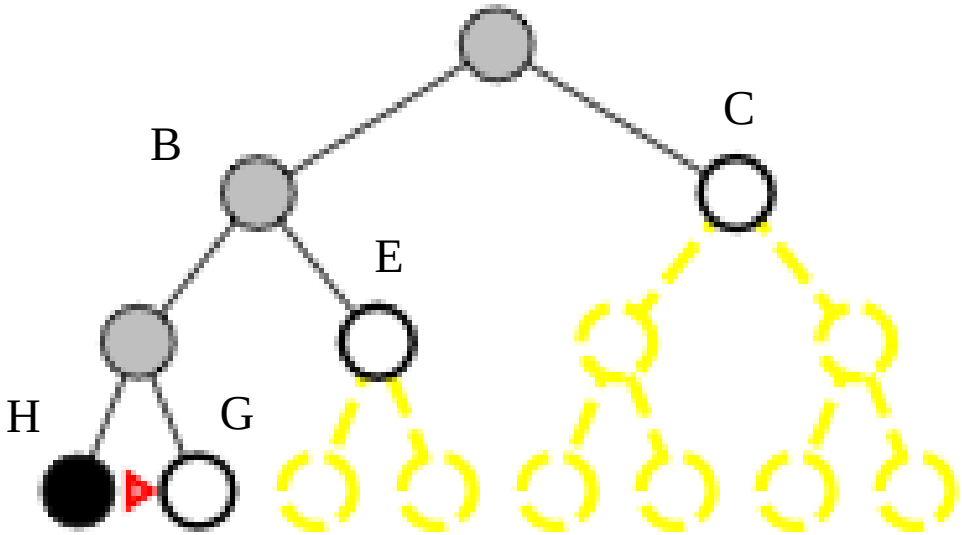
Busca em Profundidade



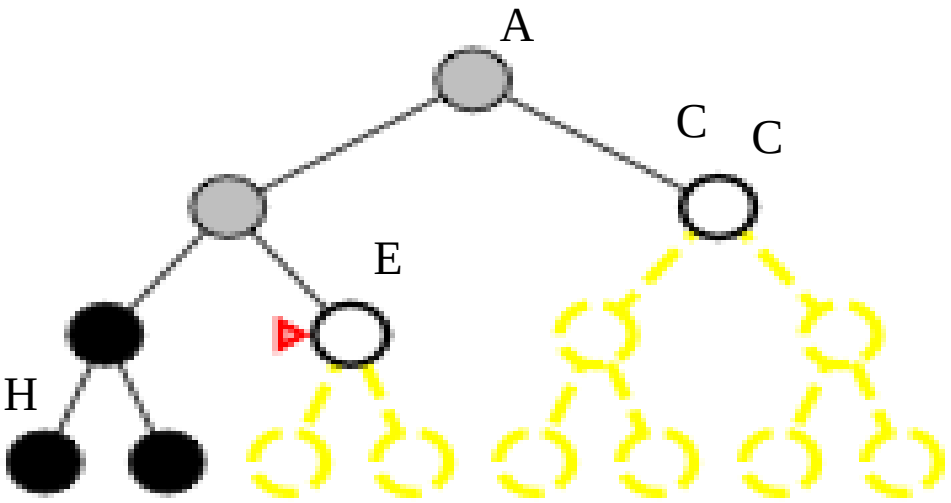
Busca em Profundidade



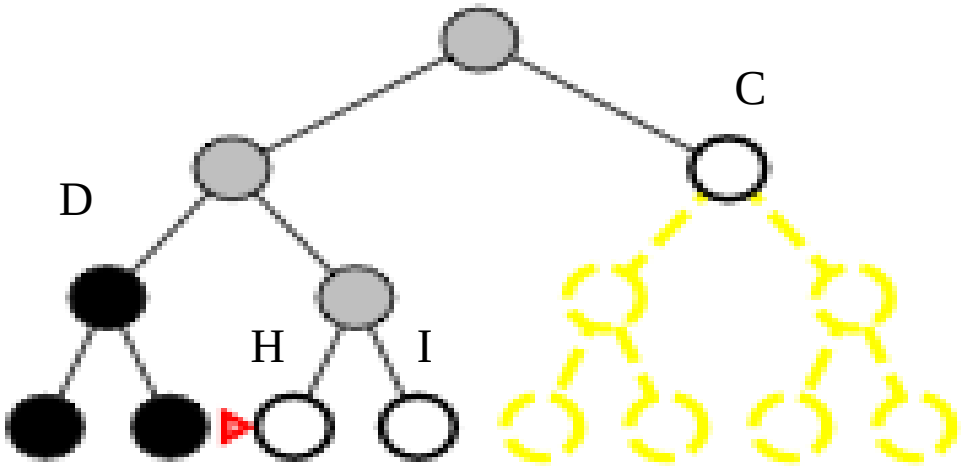
Busca em Profundidade



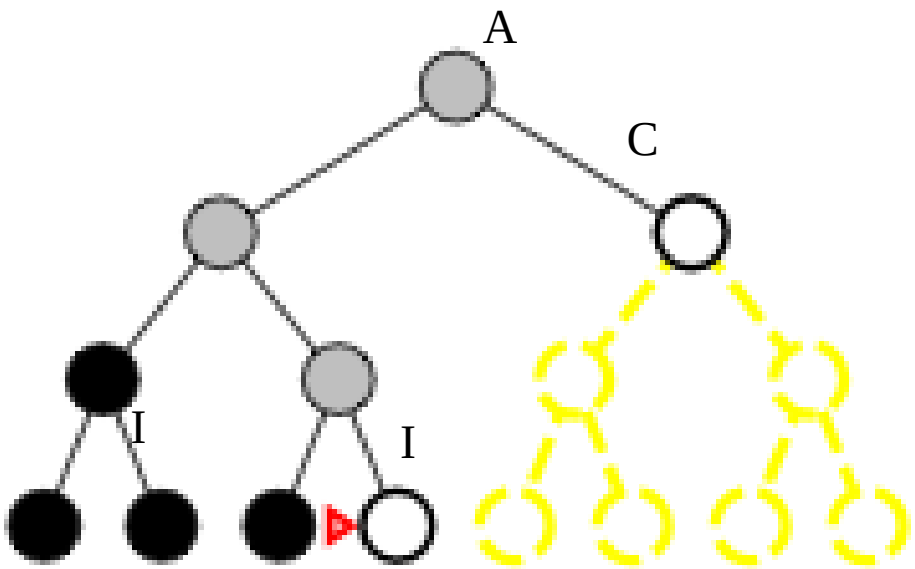
Busca em Profundidade



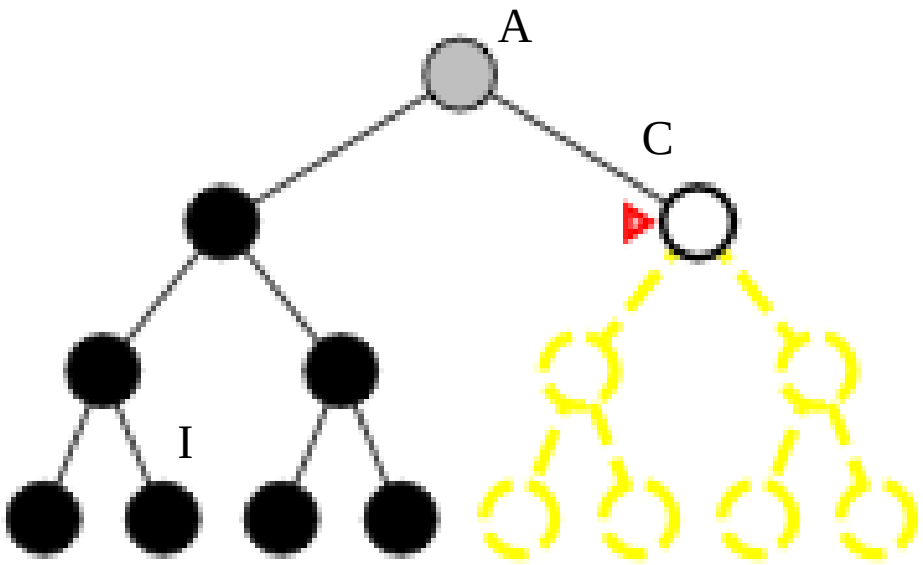
Busca em Profundidade



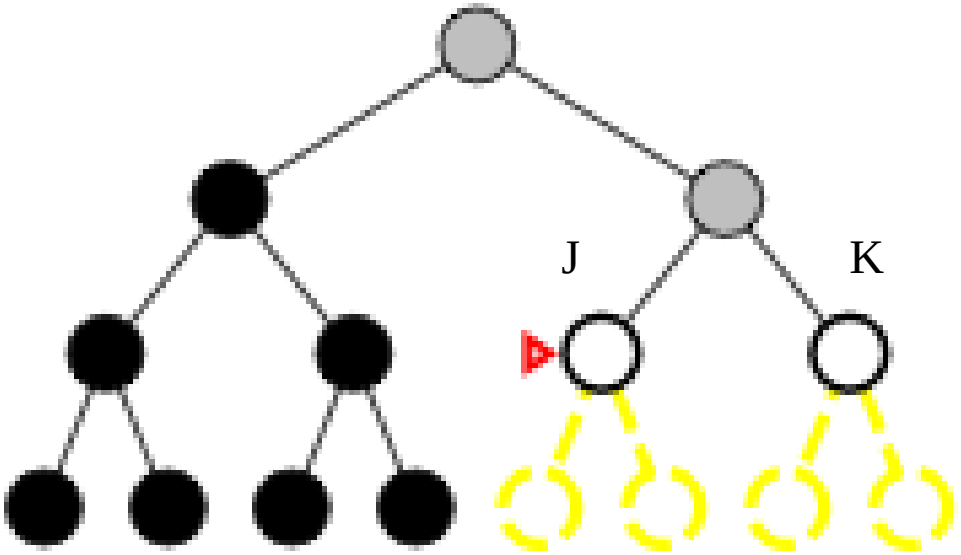
Busca em Profundidade



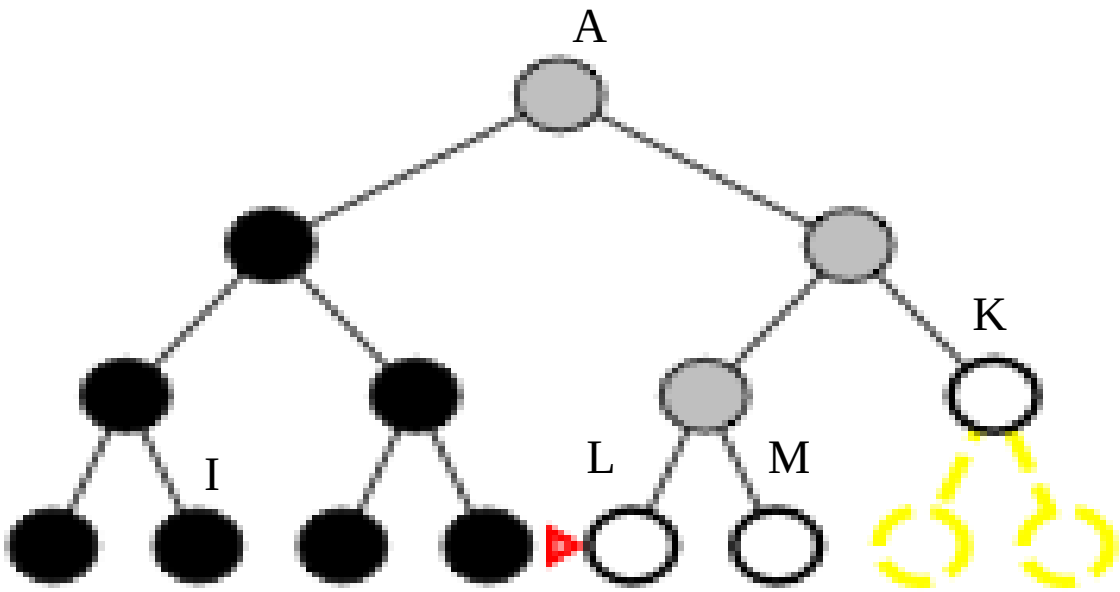
Busca em Profundidade



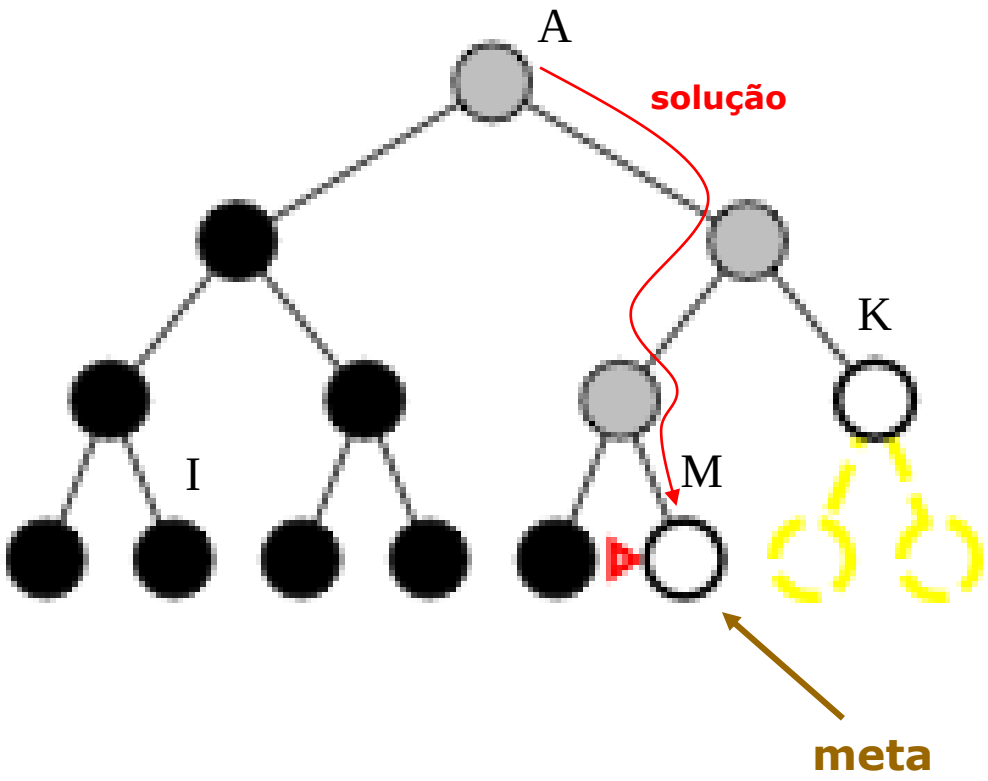
Busca em Profundidade



Busca em Profundidade



Busca em Profundidade



Busca em Profundidade

- Completude:
  - *Sempre encontra uma solução se existir?*
    - Não.
    - Caminho muito longo (infinito) que não contenha a meta pode impedir de que esta seja encontrada em outro caminho.
- Otimalidade:
  - *Sempre encontra a solução de menor custo ?*
    - Não.

Busca em Profundidade Limitada

- Trata-se da busca em profundidade com um limite de profundidade *l*.
  - Ou seja, nodos na profundidade *l* não possuem sucessores.
  - Conhecimento de domínio pode ser utilizado: No mapa da Romênia (20 cidades) qualquer solução $\rightarrow$ máximo d=19.
- Resolve o problema de árvores infinitas.
- Se $l < d$ então a estratégia não é completa.
- Se $l > d$ a estratégia não é ótima.

Busca de Aprofundamento Iterativo

- Estratégia geral para encontrar o melhor limite de profundidade *l*.
  - Limite é incrementado até *d* (desconhecido).
    - Meta é encontrada na profundidade *d*, a profundidade do nodo meta mais raso.
- Combina os benefícios das buscas em profundidade (espaço) e largura (possivelmente completude e otimalidade).

Busca de Aprofundamento Iterativo

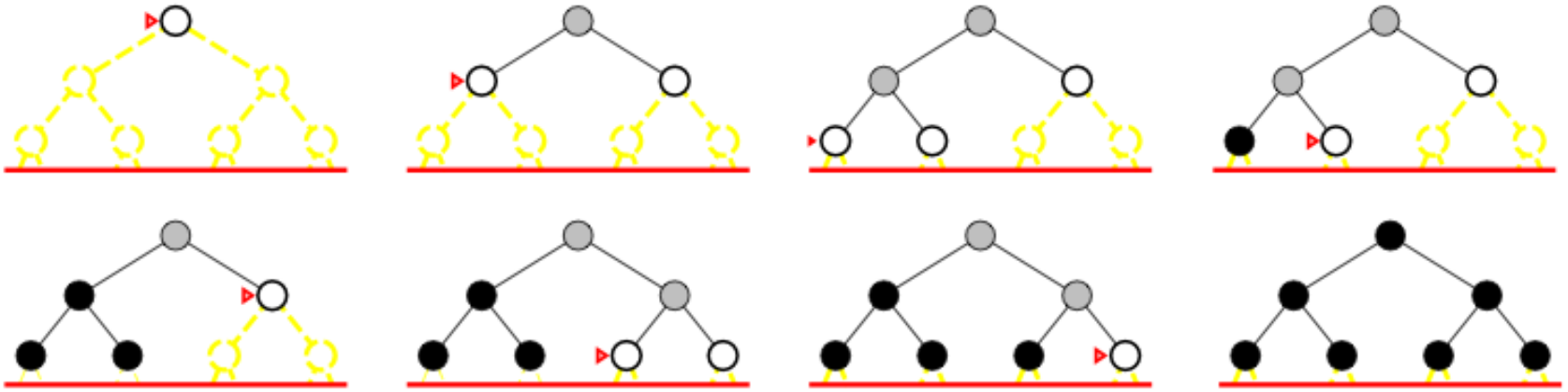
- Limite=0

Busca de Aprofundamento Iterativo

- Limite=1

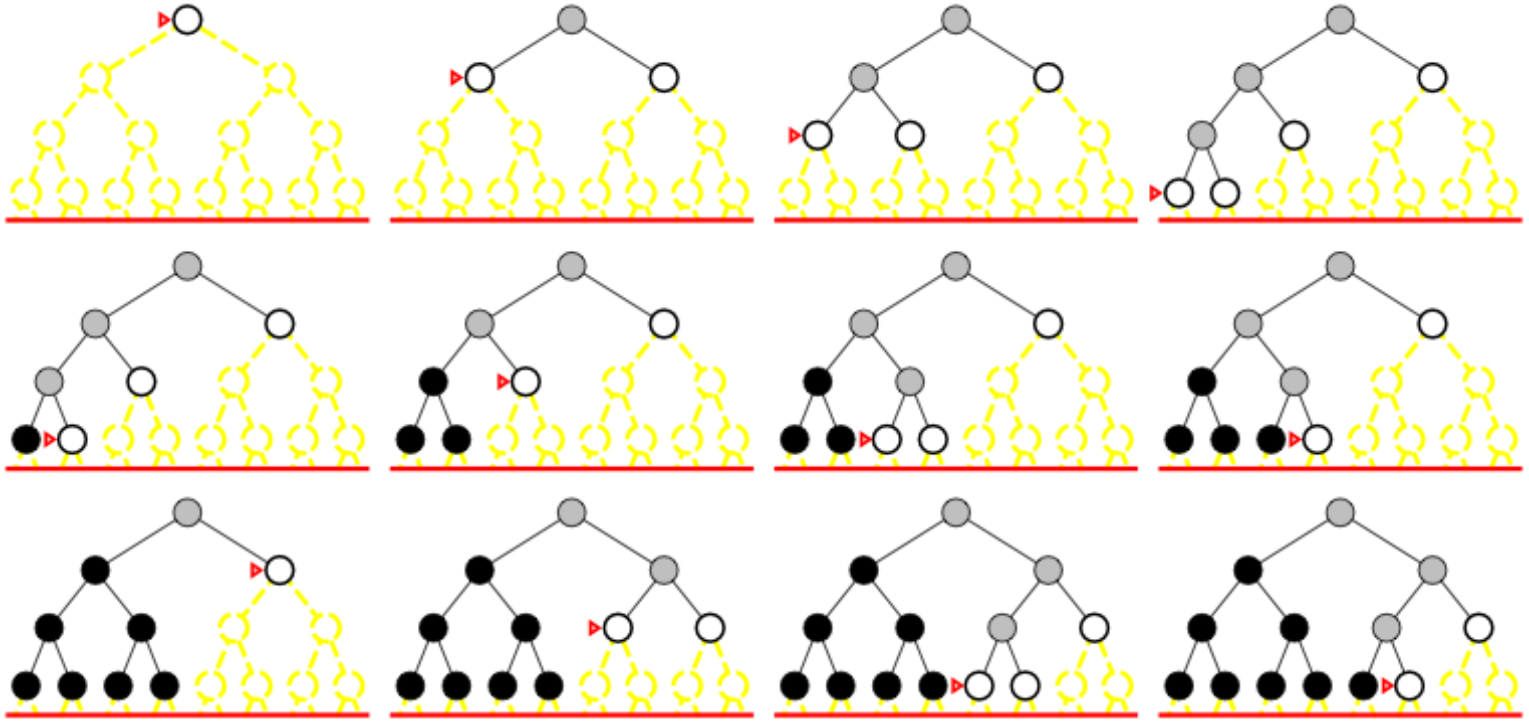
Busca de Aprofundamento Iterativo

- Limite=2



Busca de Aprofundamento Iterativo

- Limite=3



Busca de Aprofundamento Iterativo

- Semelhantemente à *busca em profundidade*, possui custo de memória reduzido;
- Tal como a *busca em largura:*
  - É completa quando o fator de ramificação é finito;
  - É ótima quando o custo do caminho é uma função não decrescente da profundidade do nó.

Considerações:

- Análises detalhadas sobre complexidade de tempo e de espaço são elaboradas no livro texto;
- Problemas de busca com complexidade exponencial não podem ser resolvidos por métodos de busca não-informada exceto para as instâncias pequenas.
- Busca Informada.

Exercício: simule busca em largura e em profundidade na árvore abaixo para encontrar os nós 10 e 20. Compare tanto o número de nós visitados como a memória usada por cada tipo de busca.