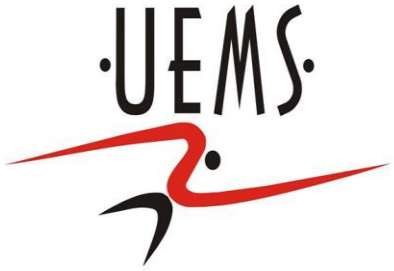


Resolução de Problemas por meio de Busca

Profa. Mercedes Gonzales Márquez

Tópicos

- Agentes de resolução de problemas.
  - Problemas e soluções de busca
  - Formulação de problemas
- Exemplos de problemas
  - Problemas Padronizados
  - Problemas do mundo real
- Algoritmos de busca
  - Busca pela melhor escolha
  - Estruturas de dados de busca

Resolução de Problemas por meio de busca

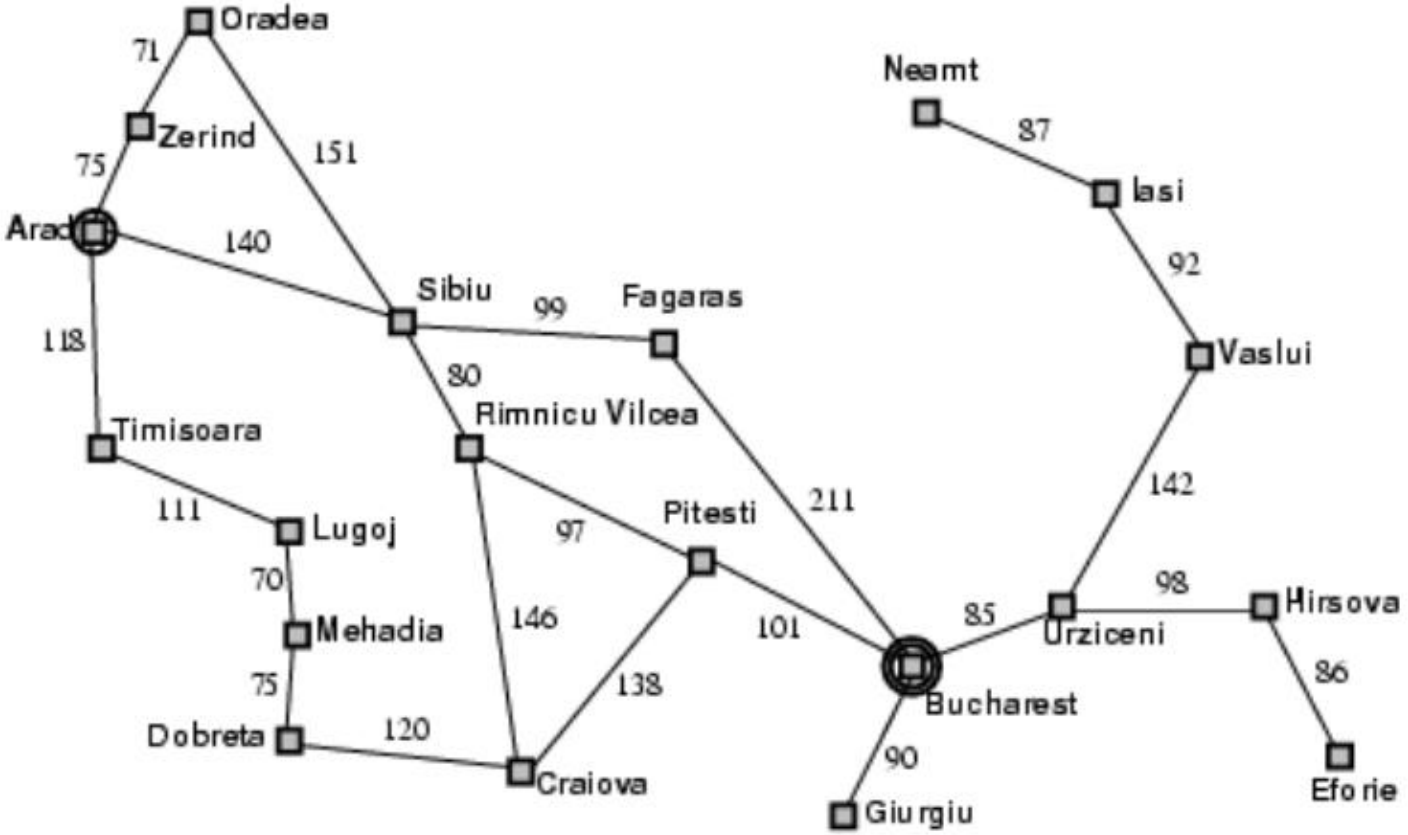
- Quando a ação correta a ser tomada não é imediatamente óbvia, pode-se considerar uma sequência de ações que formam um caminho até um estado meta. O agente que considera essa sequência é chamado de **agente de resolução de problemas**, e o processo computacional que ele realiza é chamado de **busca**.
- Consideremos ambientes mais simples: episódico, agente único, totalmente observável, determinístico, estático, discreto e conhecido.
- Distinguiremos entre algoritmos informados (estima a que distância está da meta) e não informados (sem nenhuma estimativa).

Agente de Resolução de Problemas

- Imagine um agente passando férias na Romênia e atualmente está em Arad.
- Vôo sai amanhã de Bucareste.
- Formulação da meta:
  - Estar em Bucareste
- Formulação do problema:
  - Elabora uma descrição dos estados e ações necessárias para alcançar a meta. Estados: cidades e Ações: dirigir entre as cidades.
- Busca: simulação de sequências de ações fazendo buscas até encontrar uma sequência de ações que alcance a meta. Essa sequência é a solução.

Agente de Resolução de Problemas

- Essa sequência é a solução. No exemplo:



Problemas e Soluções de Busca

- Um problema de busca pode ser formalmente definido :
  - Um conjunto de estados ou **espaço de estados**
  - O **estado inicial** em que o agente começa. Ex. Arad
  - Um conjunto de um mais **estados meta**. Ex. Bucareste. Às vezes a meta é definida por uma propriedade que se aplica a muitos estados. No mundo de aspiradores de pó a meta pode ser não ter sujeira.
  - **Ações** : As ações que estão disponíveis para o agente. Ex Ações(Arad) = {IrparaSibiu, irparaTimisoara, irparaZerind}
  - Um **modelo de transição**, o qual descreve o que cada ação faz. Ex. RESULTADO(Arad,IrparaZerind)=Zerind.
  - Uma **função de custo da ação** Custo-Ação(s,a): informa o custo numérico da aplicação da ação a no estado s.

Problemas e Soluções de Busca

- A função de custo deve refletir a medida de desempenho: ex. distância em milhas.
- Uma solução é uma sequência de ações que levam do estado inicial para o estado meta.
- Uma solução ótima é uma solução com o menor custo de caminho.
- O espaço de estados pode ser interpretado como um grafo em que os nós são estados e as arestas direcionadas entre eles são ações.

Formulação de problemas

- A formulação do problema de chegar a Bucareste é um modelo (uma descrição matemática abstrata) e não o mundo real.
- O processo de remover detalhes de uma representação é chamado de **abstração**.
- Uma viagem real cruzando o pais inclui muitos itens: companheiros de viagem, o programa de radio atual, a paisagem vista da janela, a proximidade de policiais, a distância até a próxima parada para descanso, as condições da estrada, o clima, o trânsito, e assim por diante.
- AbstraçãoSolução (abstrata) = conjunto de caminhos

Formulação de problemas

- Abstração válida: se qualquer solução abstrata pode ser expandida em uma solução mais detalhada (no mundo real);
- Abstração útil: se a execução de cada uma das ações na solução é mais fácil que o problema original.

Exemplos de problemas

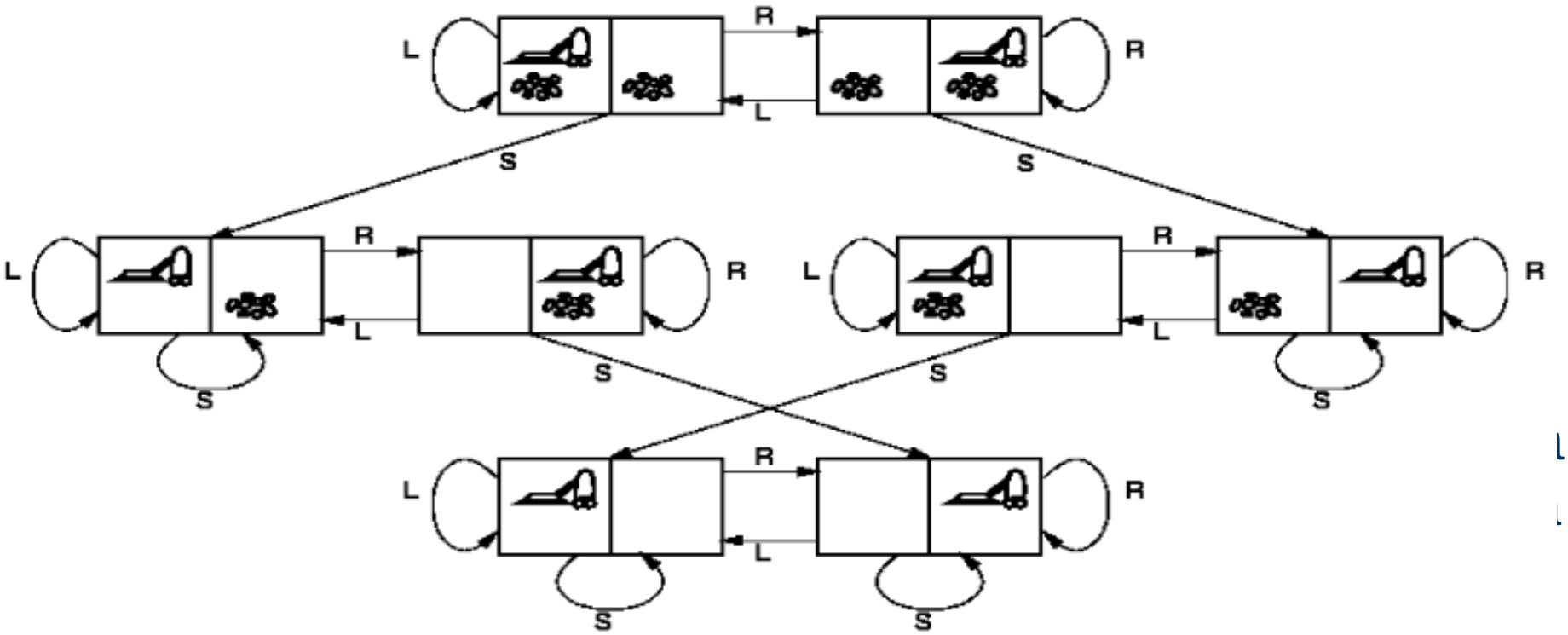
- Analisaremos dois tipos de problemas:
  - Problemas padronizados: abordaremos problemas do mundo de grade.
  - Problemas do mundo real, como a navegação por robô cuja formulação é fora de padrões, ex. cada robô tem diferentes sensores, que produzem diferentes dados.

Exemplos de problemas

- Analisaremos dois tipos de problemas:
  - Problemas padronizados: podem ter uma descrição concisa e exata como os problemas do mundo de grade. São utilizáveis como *benchmark* por diferentes pesquisadores para ilustrar, exercitar ou comparar.
  - Problemas do mundo real, como a navegação por robô cuja formulação é fora de padrões, ex. cada robô tem diferentes sensores, que produzem diferentes dados.

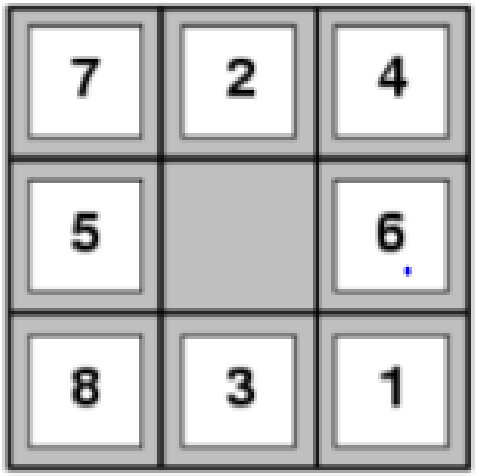
Problemas padronizados

- Um problema do mundo de grade é uma matriz retangular com células em que os agentes podem se mover de uma célula para outra. O mundo do aspirador de pó pode ser formulado como um problema do mundo de grade.

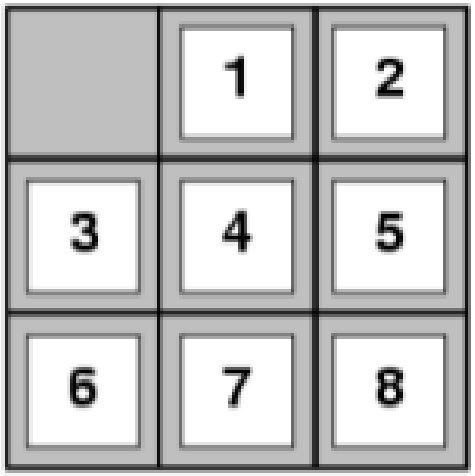


- Estados: O agente pode estar em uma entre duas posições, cada uma delas pode conter sujeira ou não.
- Estado inicial: qualquer
- Ações: aspirar, esquerda, direita.
- Modelo de Transição : aspirar: remove qualquer sujeira da célula do agente, etc.
- Estados meta: todos os quadrados limpos
- Custo: cada passo=1

O quebra--cabeça de 8 peças



Start State



Goal State

O quebra--cabeça de 8 peças

- Estados: Especifica a posição de cada uma das peças e do espaço vazio.
- Estado inicial: Qualquer um
- Modelo de transição: gera os estados válidos que resultam da tentativa de executar as
quatro ações (mover espaço vazio para esquerda, direita, acima ou abaixo)
- Estado meta: Estado com os números em ordem.
- Custo do caminho: Cada passo custa 1, e assim o custo do caminho é o número de passos do caminho

Problema Jarros

- Dados uma bica d'água, um jarro de capacidade de 3 litros e um jarro de capacidade de 4 litros (ambos vazios). O agente pode encher os jarros ou esvaziá-los despejando a água de um para o outro, ou derramando-a no chão. O agente precisa medir exatamente 2 litros de água em algum jarro.

Problema dos jarros

- Estados: Jarro3 e jarro4 cheio ou vazio ou parcialmente cheio.
- Estado inicial: jarro3 e jarro4 vazios.
- Ações: encher, derramar ou transferir.
- Modelo de transição: gera os estados válidos que resultam da tentativa de executar as três ações.
- Estado meta: Estado com um dos jarros com 2 litros.
- Custo do caminho: Cada passo custa 1, e assim o custo do caminho é o número de passos do caminho

Problemas do mundo real

- Problema de roteamento: encontrar a melhor rota de um ponto a outro (aplicações: redes de computadores, planejamento militar, planejamento de viagens aéreas)
- Problemas de roteiro de viagem: visitar cada ponto pelo menos uma vez. Caixeiro viajante, visitar cada cidade exatamente uma vez, encontrar o caminho mais curto.
- A navegação de robôs: generalização do problema de roteamento. Em vez de seguir um conjunto discreto de rotas, um robô pode se mover e criar suas próprias rotas. No caso do robô circular em movimentos sobre uma superfície plana, o espaço é bidimensional. Quando o

Problemas do mundo real

robô tem braços e pernas que também devem ser controlados, o espaço de busca passa a ter várias dimensões (uma para cada ângulo de junção. São necessárias técnicas avançadas apenas para tornar finito o espaço de busca essencialmente contínuo.

- Sequência automática de montagem: o objetivo é encontrar uma ordem na qual devem ser montadas as peças de algum objeto.
- Projeto de proteínas: encontrar uma sequência de aminoácidos que serão incorporados em uma proteína tridimensional para curar alguma doença.

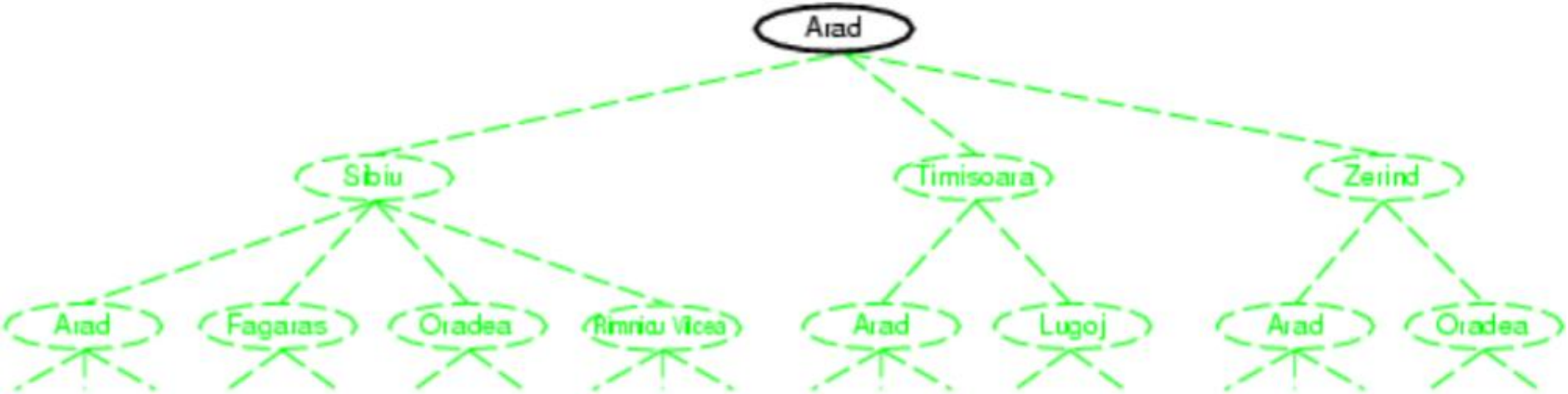
Algoritmos de busca

- Um algoritmo de busca recebe um problema de busca como entrada e devolve uma solução ou uma indicação de falha.
- Idéia: Percorrer o espaço de estados a partir de uma árvore de busca.
- O nó raiz da árvore corresponde ao estado inicial.
- Podemos expandir o nó (estado atual), considerando as AÇÕES disponíveis para esse estado, usando o modelo de transição, gerando um novo nó (nó filho) para cada uma das ações resultantes.

Algoritmos de busca

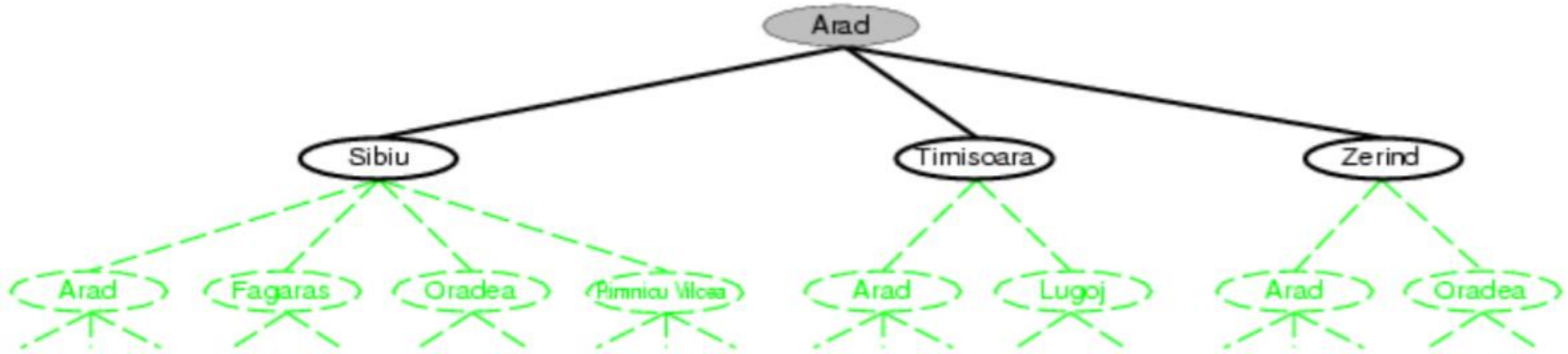
- Busca: consiste em escolher um entre os nós resultantes para expandi-lo, deixando os outros para depois.
- A estratégia de busca determina qual caminho seguir.
- Dizemos que qualquer estado que teve um nó gerado foi alcançado, sendo o nó expandido ou não.

Algoritmos de busca



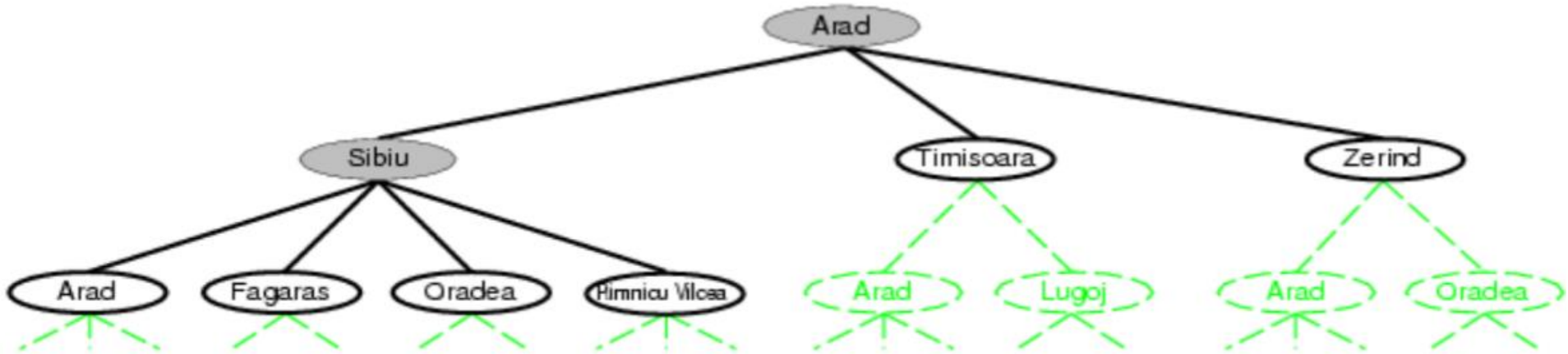
Estado Inicial.

Algoritmos de busca



Depois de expandir Arad.

Algoritmos de busca



Depois de expandir Sibiu.

Estrutura de dados de busca

Nó.ESTADO: o estado a que o nó corresponde

Nó.PAI: o nó na árvore de busca que gerou esse nó

Nó.AÇÃO: a ação que foi aplicada ao estado do pai para gerar esse nó.

Nó.CUSTO-CAMINHO: o custo total do caminho desde o estado inicial até este nó.

Estrutura de dados de busca

As operações sobre a fronteira da árvore são:

É VAZIA(fronteira): devolve V somente se não houver mais nós na fronteira.

POP(fronteira): remove e devolve o nó do topo da fronteira.

TOPO(fronteira): devolve (mas não remove) o nó do topo da fronteira.

INSERIR (nó,fronteira): insere um elemento em seu local apropriado na fila.

Estrutura de dados de busca

A estrutura de dados para armazenar a fronteira: três tipos de fila são usados nos algoritmos de busca.

- Uma fila de prioridade: remove primeiro o nó com menor custo, de acordo com alguma função avaliação f. Ela é usada na busca pela menor escolha.
- Uma fila FIFO: remove primeiro o nó que foi primeiro adicionado na fila; é usada na busca em largura.
- Uma fila LIFO (pilha): remove primeiro o nó que foi acrescentado por último; é usada na busca em profundidade.

Os estados alcançados podem ser armazenados como uma tabela de busca (ex. hash), em que cada chave é um estado e cada valor é o nó para esse estado.

Busca pela melhor escolha

- Como decidir qual nó da fronteira devemos expandir?
- Uma abordagem muito geral é chamada de busca pela melhor escolha, na qual escolhemos um nó com o valor mínimo de alguma função avaliação f(n).
- Veja o algoritmo:

```
função BUSCA-MELHOR-ESCOLHA(problema,f) devolve um nó solução ou falha
  nó ← NÓ(ESTADO=problema.INICIAL)
  fronteira ← uma fila de prioridade ordenada por f, com nó como elemento
  alcançado ← uma tabela de busca, com uma entrada com chave problema.INICIAL e valor nó
  enquanto não ESTÁ-VAZIA(fronteira) faça
      nó ← POP(fronteira)
      se problema.É-META(nó.ESTADO) então devolve nó
      para cada filho em EXPANDE(problema,nó) faça
          s ← filho.ESTADO
          se s não está em alcançado ou filho.CUSTO-CAMINHO < alcançado[s].
          CUSTO-CAMINHO então
            alcançado[s] ← filho
            adicione filho a fronteira
devolve falha

função EXPANDE(problema,nó) produz nós
s ← nó.ESTADO
para cada ação em problema.AÇÕES(s) faça
    s' ← problema.RESULTADO(s, ação)
    custo ← nó.CUSTO-CAMINHO + problema.CUSTO-AÇÃO(s,ação,s')
    produz NÓ(ESTADO=s', PAI=nó, AÇÃO=ação, CUSTO-CAMINHO=custo)
```

Medição de desempenho de um algoritmo de busca

- completude: o algoritmo sempre encontra a solução se ela existe?
- complexidade de tempo: número de nós gerados
- complexidade de espaço: número máximo de nós na memória
- otimização: a estratégia encontra a solução ótima?

- Complexidade de tempo e espaço são medidas em termos de:
  - b: máximo fator de ramificação da árvore (número máximo de sucessores de qualquer nó)
  - d: profundidade do nó objetivo menos profundo
  - m: o comprimento máximo de qualquer caminho no espaço de estados (pode ser ∞)

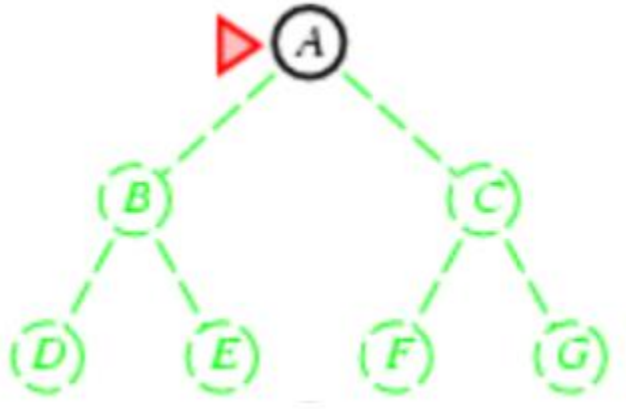
Estratégias de Busca Sem Informação (ou Busca Cega)

- Estratégias de busca sem informação não recebe qualquer indicação sobre a proximidade que um estado se encontra de sua meta. Veja o exemplo de Zarad no mapa.
- As estratégias de busca sem informação se distinguem pela ordem em que os nós são expandidos.

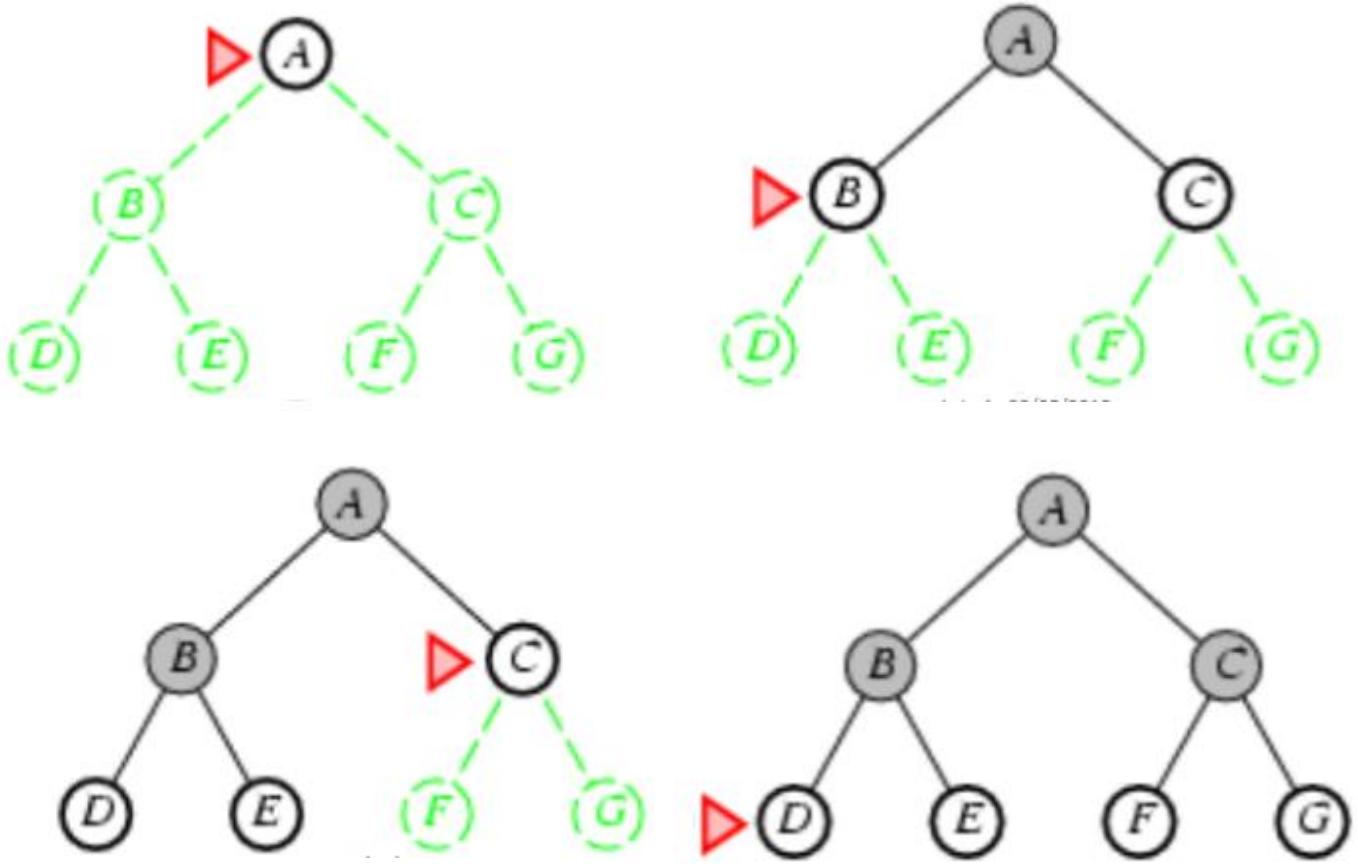
– Busca em largura, Busca de custo uniforme, Busca em profundidade, Busca em profundidade limitada e Busca de aprofundamento iterativo.

Busca em largura

- O nó raiz é expandido primeiro e, em seguida, todos os sucessores dele, depois todos os sucessores desses nós.
- Todos os nós em uma dada profundidade são expandidos antes de todos os nós do nível seguinte.



Busca em largura

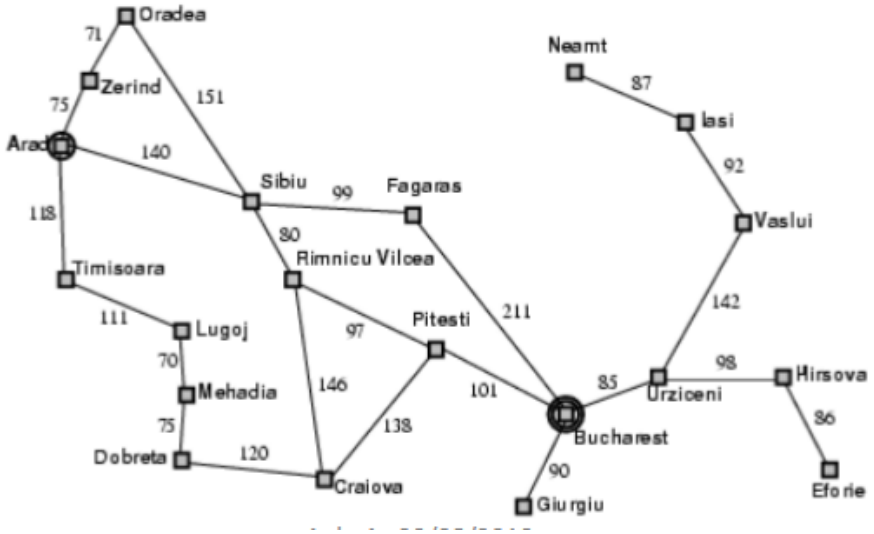


Busca em largura

- Propriedades da busca em largura
- Completa? Sim (se b é finito)
- Tempo? 1+b+b2+b3+… +b^d = O(b^d)
- Espaço? O(b^d) (mantém todos os nós na memória)
- Ótima? Sim (se todas as ações tiverem o mesmo custo.

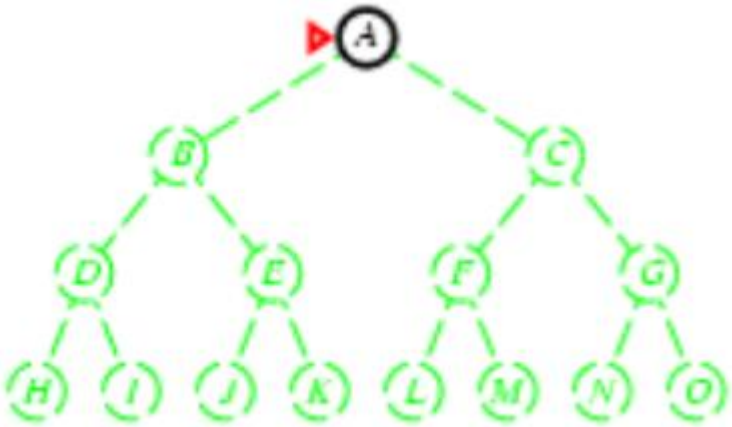
Busca de custo uniforme

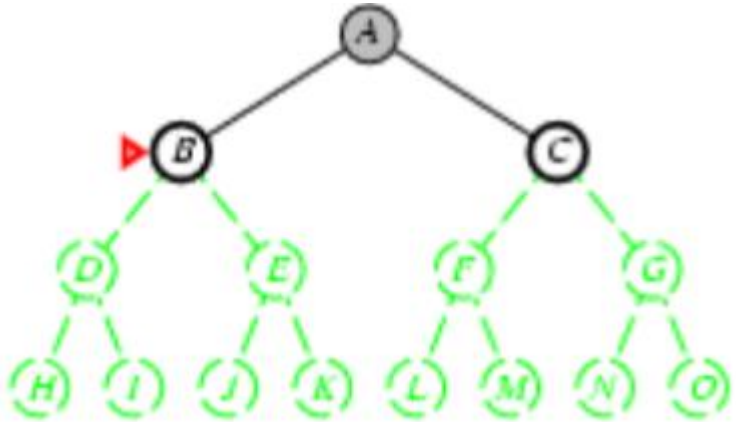
- Expande o nó que tenha o caminho de custo mais baixo.
- Aplicar busca de custo uniforme para achar o caminho mais curto entre Sibiu e Bucareste.



Busca em profundidade

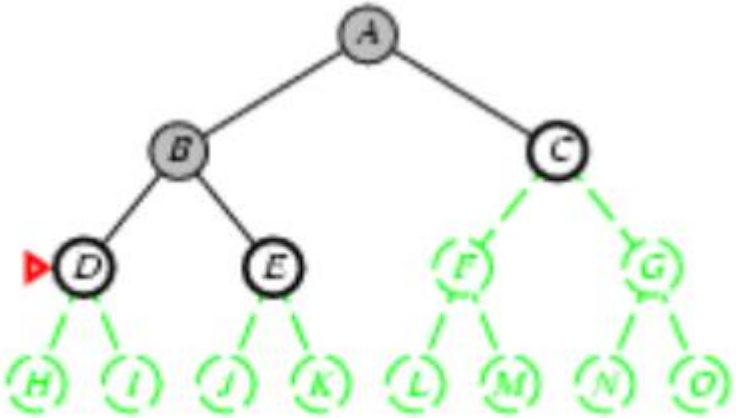
- Expande o nó não-expandido mais profundo.

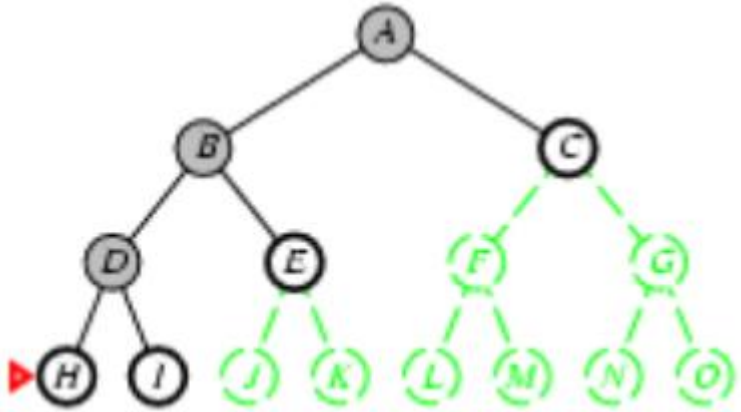




Busca em profundidade

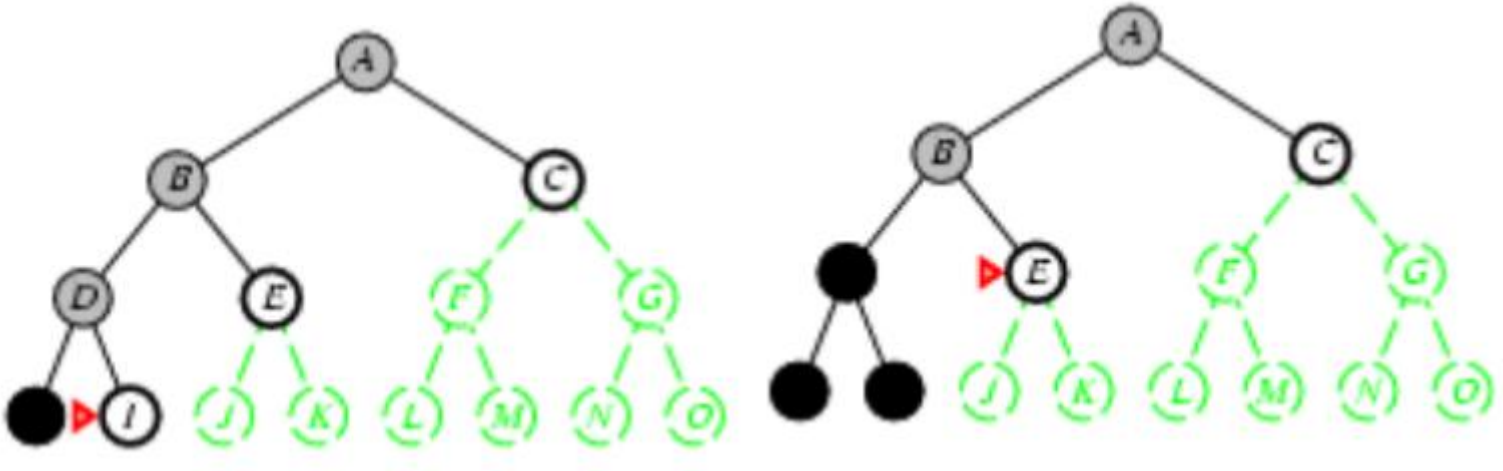
- Expande o nó não-expandido mais profundo.





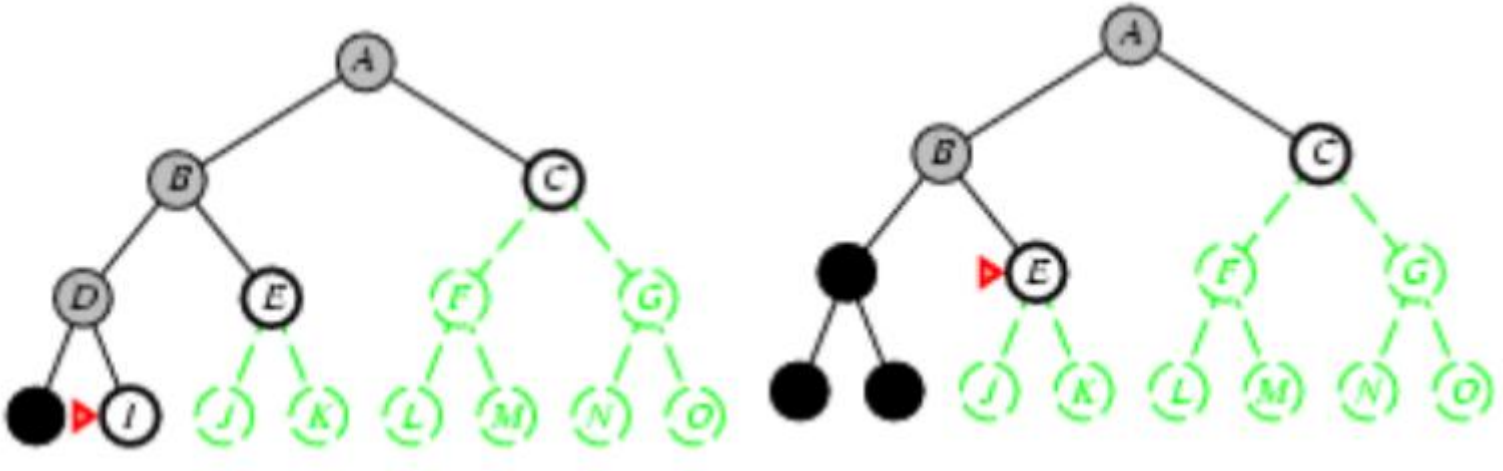
Busca em profundidade

- Expande o nó não-expandido mais profundo.



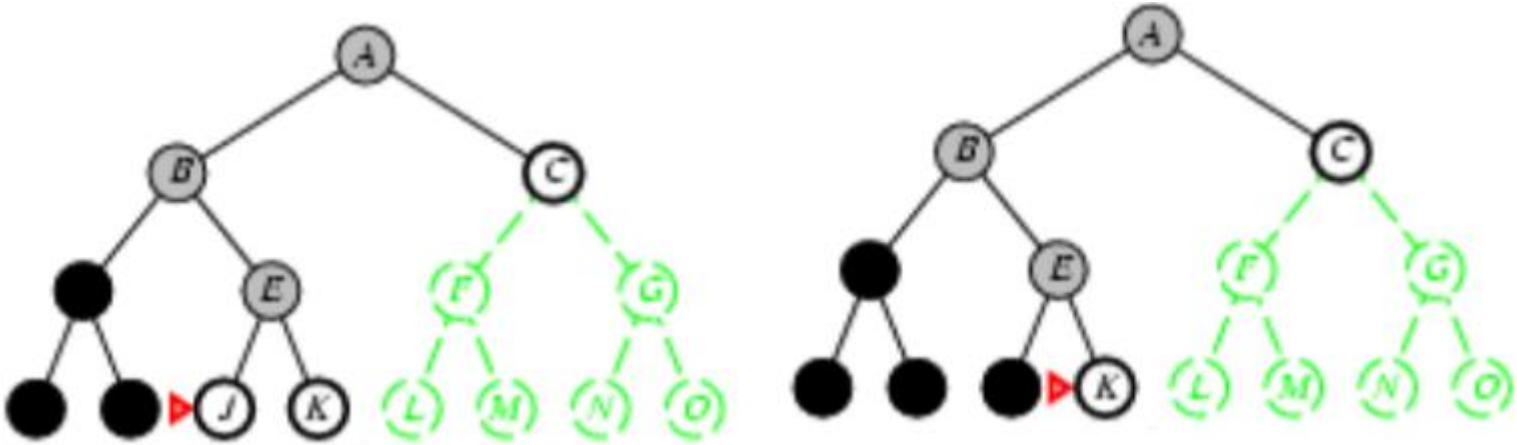
Busca em profundidade

- Expande o nó não-expandido mais profundo.



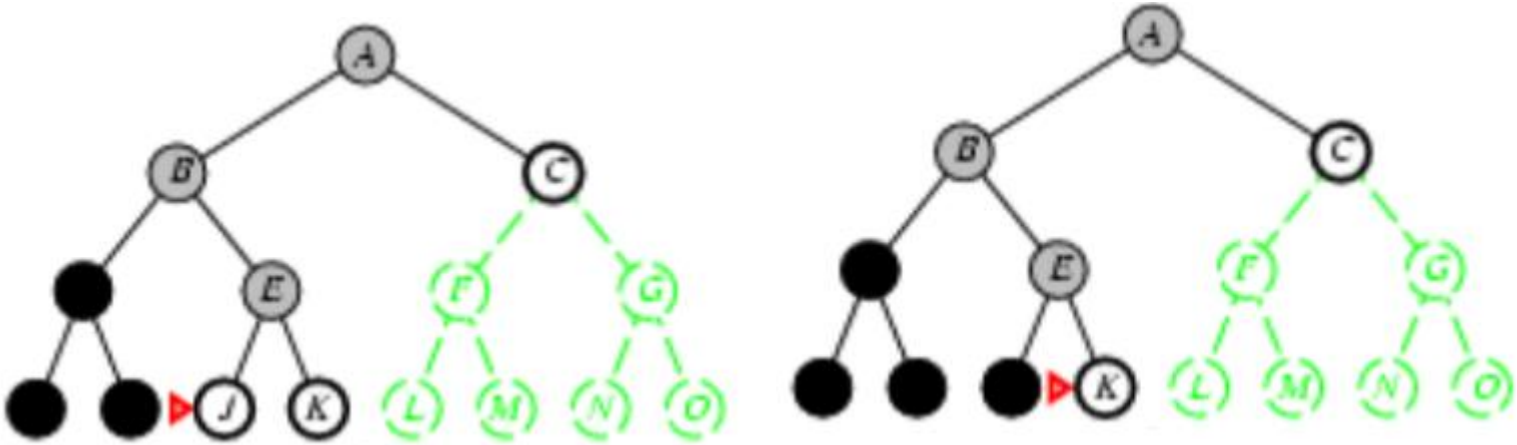
Busca em profundidade

- Expande o nó não-expandido mais profundo.



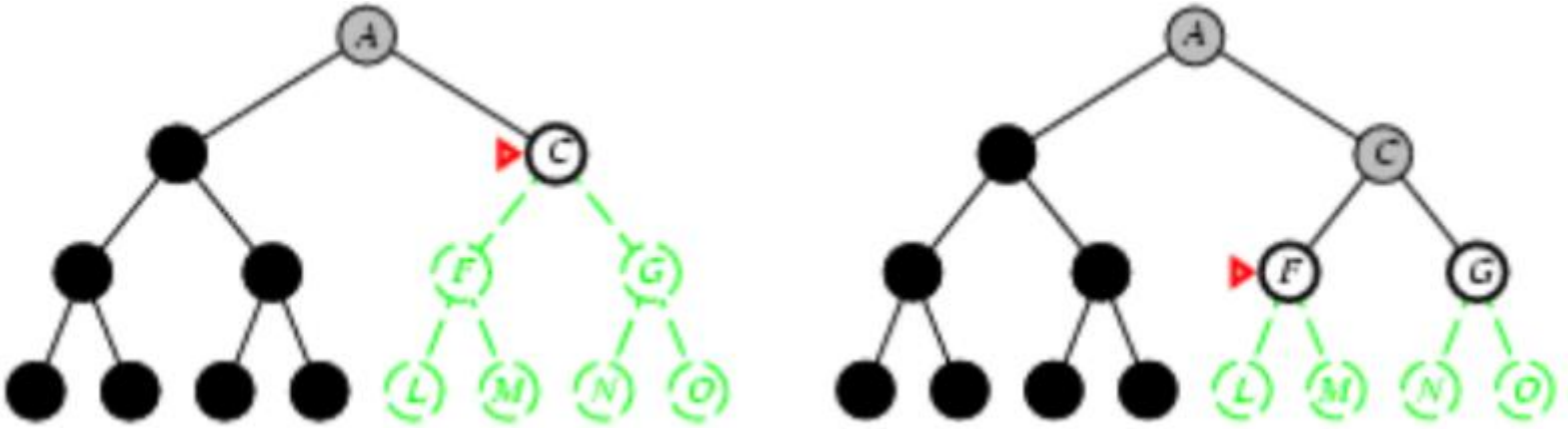
Busca em profundidade

- Expande o nó não-expandido mais profundo.



Busca em profundidade

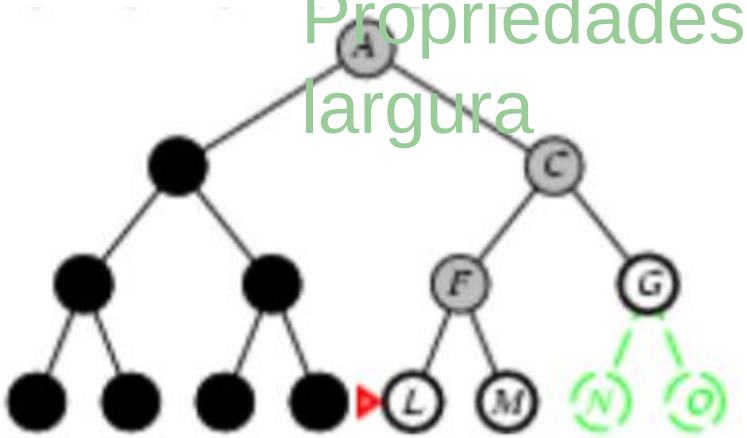
- Expande o nó não-expandido mais profundo.

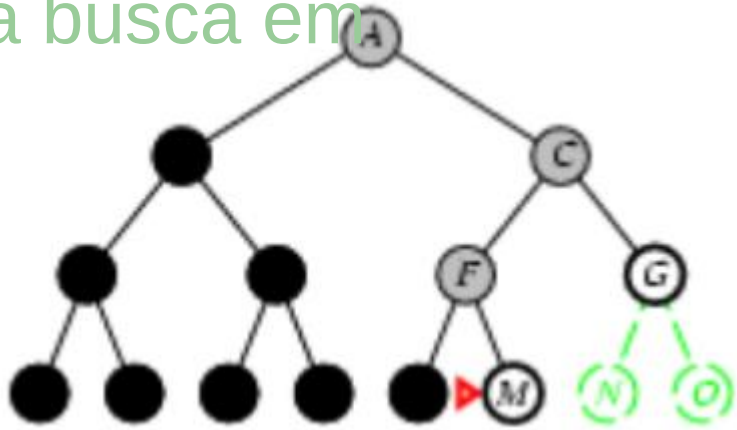


Busca em profundidade

- Expande o nó não-expandido mais profundo.

Propriedades da busca em largura





Busca em profundidade

- Propriedades da busca em profundidade.
- Completa?
- Tempo?
- Espaço?
- Ótima?