



Resolução de problemas por meio de busca

(Capítulo 3 - Russell)

Inteligência Artificial

Professor: Rosalvo Ferreira de Oliveira Neto

1. Agente de resolução de problema
2. Tipos de problema
3. Formulação de problema
4. Exemplos de problemas
5. Algoritmos de busca cega
6. Exercícios

Agentes de resolução de problemas

- Agentes reativos não funcionam em ambientes para os quais o número de regras condição-ação é grande demais para armazenar.
- Nesse caso podemos construir um tipo de agente baseado **em objetivo** chamado de agente de resolução de problemas.

Agentes de resolução de problemas

- Um agente com várias opções imediatas pode decidir o que fazer comparando diferentes seqüências de ações possíveis.
- Esse processo de procurar pela melhor seqüência é chamado de **busca**.
- Formular objetivo → buscar → executar

Agentes de resolução de problemas

Os Agentes de resolução de problemas decidem o que fazer encontrando seqüências de ações que levam a estados desejáveis.

Agentes de resolução de problemas

- Primeiro passo:
- Formulação do objetivo baseado:
 - Situação atual
 - Medida de desempenho



Agentes de resolução de problemas

A formulação do problema

Processo de decidir que ações e estados devem ser considerados, dado um objetivo.

A tarefa do agente é descobrir que sequencia de ações o levará a um estado objetivo.

Tipos de problema

A abordagem de resolução de problemas é aplicada a uma série de ambientes de tarefas.

Inicialmente existem duas classes:

- Miniproblemas
- Problemas do mundo real

Miniproblemas

Se destina a ilustrar problemas ou exercitar diversos métodos de resolução de problemas. Ele pode ter uma descrição concisa e exata.

Isso significa que ele pode ser usado com facilidade por diferentes buscadores, com a finalidade de comparar o desempenho de algoritmos.

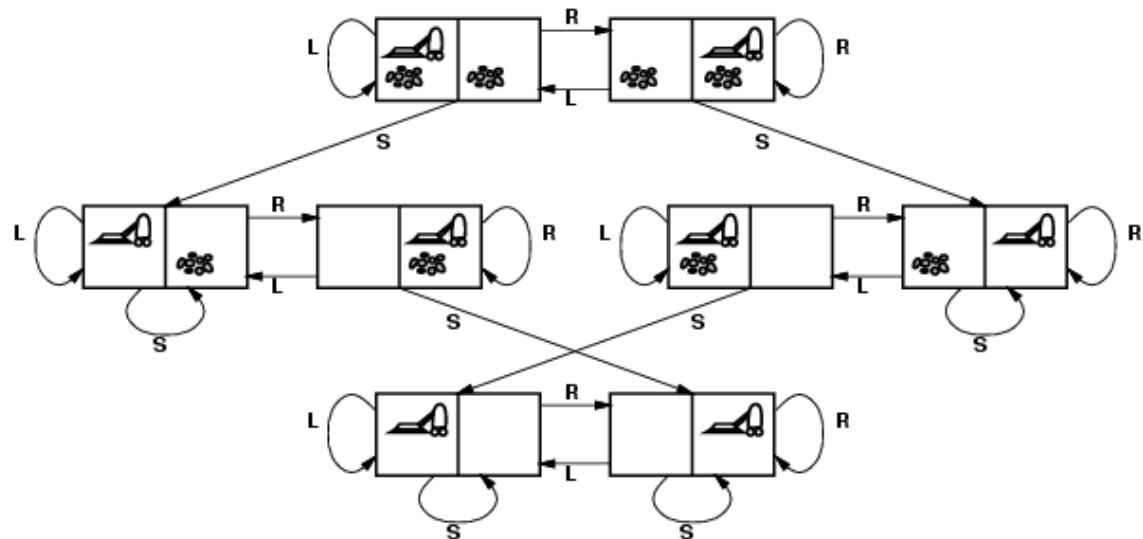
Características do ambiente:

- Determinístico;
- Completamente observáveis;
- Estáticos;
- Completamente conhecidos;

Exemplos de Miniproblemas

Mundo do aspirador de pó

O agente aspirador de pó percebe em que quadrado está e se existe sujeira no quadrado. Ele pode mover-se para direita ou para esquerda, aspirar a sujeira ou não fazer nada.



Exemplos de Miniproblemas

Quebra-cabeça de 8 peças

1	2	3
8		4
7	6	5

	1	2
3	4	5
6	7	8

Problemas do mundo real

São aqueles cujas soluções de fato preocupam as pessoas. Eles tendem a não representar uma única descrição consensual, mas tentaremos dar uma idéia geral de suas formulações.

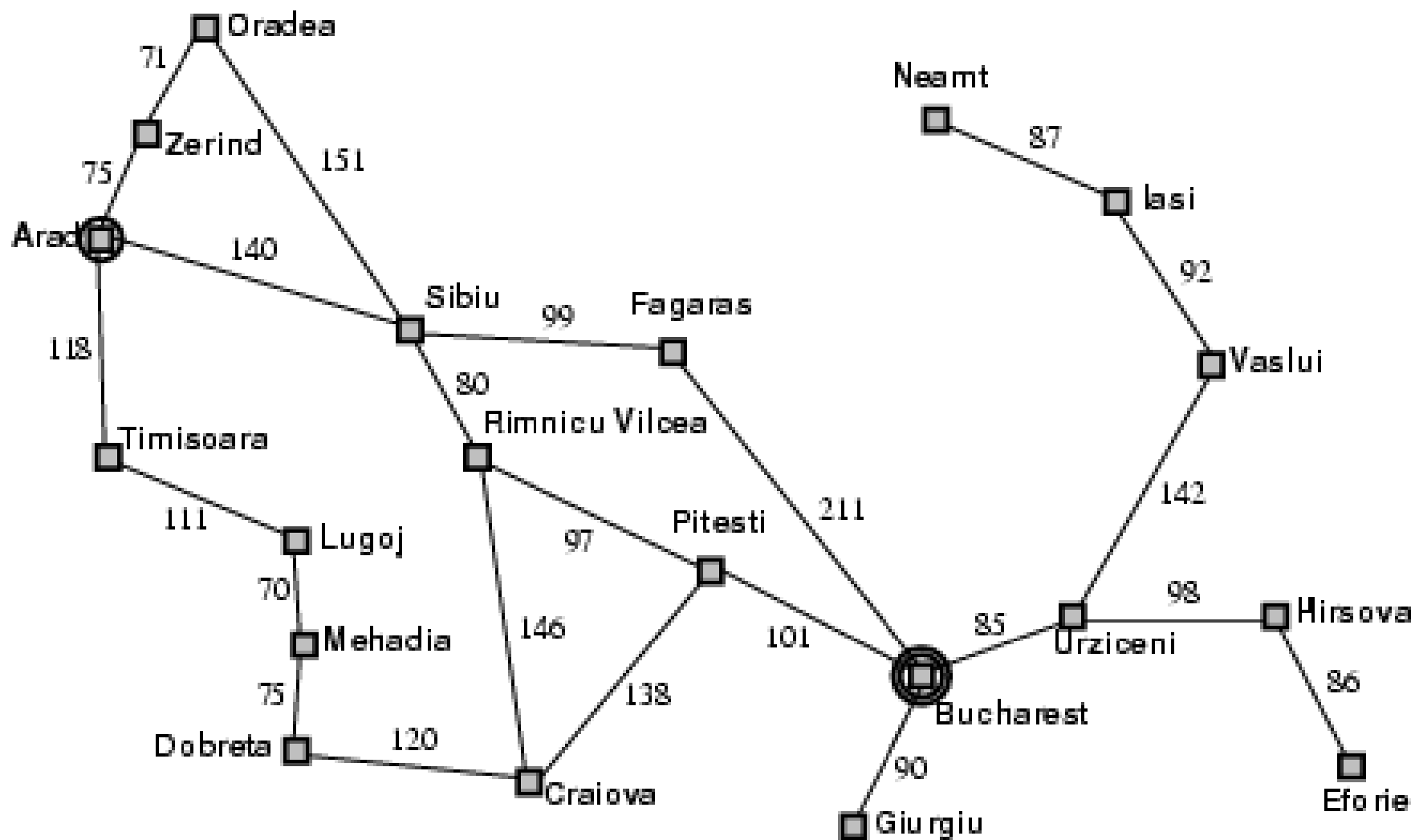
Problemas do mundo real

- Problema de roteamento
 - Encontrar a melhor rota de um ponto a outro (aplicações: redes de computadores, planejamento militar, planejamento de viagens aéreas)
- Problemas de tour
 - Visitar cada ponto pelo menos uma vez
- Caixeiro viajante
 - Visitar cada cidade exatamente uma vez
 - Encontrar o caminho mais curto
- Layout de VLSI
 - Posicionamento de componentes e conexões em um chip

Exemplos de problemas (Romênia)

- De férias na Romênia; atualmente em Arad.
- Vôo sai amanhã de Bucareste.
- Formular objetivo:
 - Estar em Bucareste
- Formular problema:
 - estados: cidades
 - ações: dirigir entre as cidades
- Encontrar solução:
 - sequência de cidades, ex., Arad, Sibiu, Fagaras, Bucareste.

Exemplos de problemas (Romênia)



Formulação de problemas

Um **problema** é definido por quatro itens:

1. **Estado inicial** ex., “em Arad”
 2. **Ações ou função sucessor** $S(x)$ = conjunto de pares estado-ação
 - ex., $S(\text{Arad}) = \{ \langle \text{Arad} \rightarrow \text{Zerind}, \text{Zerind} \rangle, \dots \}$
 3. **Teste de objetivo**, pode ser
 - explícito, ex., $x = \text{“em Bucharest”}$
 - implícito, ex., $\text{Cheque-mate}(x)$
 -
 4. **Custo de caminho** (aditivo)
 - ex., Soma das distâncias, Número de ações executadas, etc.
- Uma **solução** é uma seqüência de ações que levam do estado inicial para o estado objetivo.
 - Uma **solução ótima** é uma solução com o menor custo de caminho.

Algoritmo de Resolução de Problemas

```
function SIMPLE-PROBLEM-SOLVING-AGENT(percept) returns an action
  static: seq, an action sequence, initially empty
           state, some description of the current world state
           goal, a goal, initially null
           problem, a problem formulation

  state  $\leftarrow$  UPDATE-STATE(state, percept)
  if seq is empty then do
    goal  $\leftarrow$  FORMULATE-GOAL(state)
    problem  $\leftarrow$  FORMULATE-PROBLEM(state, goal)
    seq  $\leftarrow$  SEARCH(problem)
  action  $\leftarrow$  FIRST(seq)
  seq  $\leftarrow$  REST(seq)
  return action
```

Supõe que ambiente é estático, observável, discreto e determinístico.

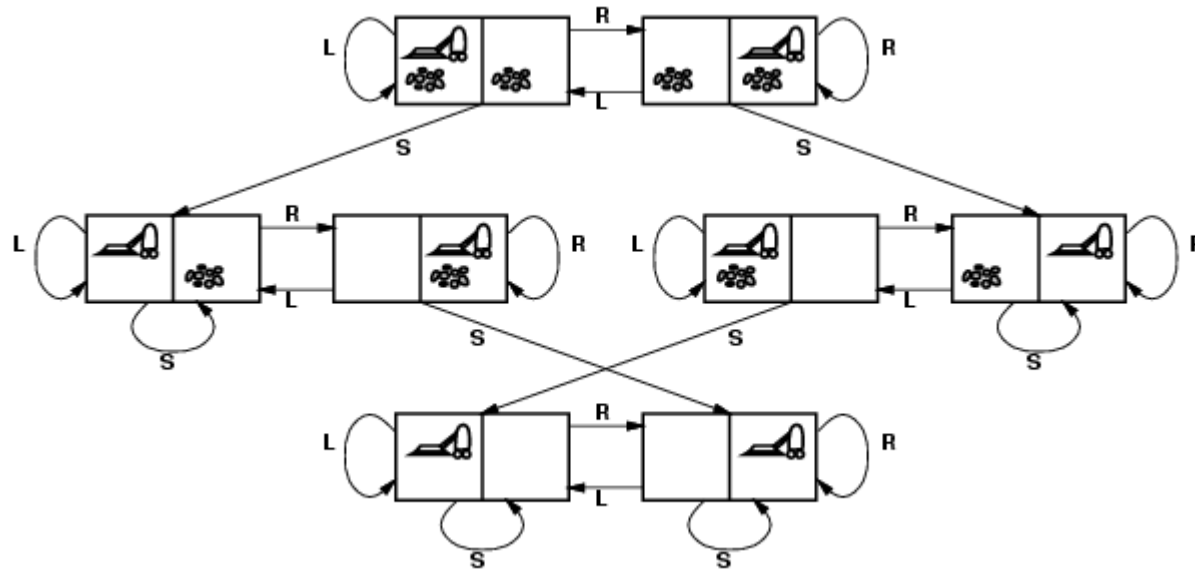
Espaço de estados

- O conjunto de todos os estados acessíveis a partir de um estado inicial é chamado de espaço de estados.
- **Como os estados são acessados?**
 - Os estados acessíveis são aqueles dados pela função sucessora.
- **Como representar o espaço de estado?**
 - *O espaço de estados pode ser interpretado como um grafo em que os nós são estados e os arcos são ações.*

Selecionando um espaço de estados

- O mundo real é absurdamente complexo
→ o espaço de estados é uma **abstração**
- Estado (abstrato) = conjunto de estados reais
- Ação (abstrata) = combinação complexa de ações reais
- Solução (abstrata) = conjunto de caminhos reais que são soluções no mundo real

Exemplo de Espaço de Estados



- **Estados:** Definidos pela posição do robô e sujeira (8 estados)
- **Estado inicial:** Qualquer um
- **Função sucessor:** pode-se executar qualquer uma das ações em cada estado (esquerda, direita, aspirar)
- **Teste de objetivo:** Verifica se todos os quadrados estão limpos
- **Custo do caminho:** Cada passo custa 1, e assim o custo do caminho é o número de passos do caminho

Exemplo de Espaço de Estados

7	2	4
5		6
8	3	1

Start State

	1	2
3	4	5
6	7	8

Goal State

- **Estados:** Especifica a posição de cada uma das peças e do espaço vazio
- **Estado inicial:** Qualquer um
- **Função sucessor:** gera os estados válidos que resultam da tentativa de executar as quatro ações (mover espaço vazio para esquerda, direita, acima ou abaixo)
- **Teste de objetivo:** Verifica se o estado corresponde à configuração objetivo.
- **Custo do caminho:** Cada passo custa 1, e assim o custo do caminho é o número de passos do caminho.

Estrutura de um Nó

- Estado
- Ação
- Nó-Pai
- Custo-Do-Caminho
- Profundidade

Borda

- A coleção de nós que foram gerados, mas ainda não foram expandidos é chamada de borda.

A borda pode ser representado por uma fila que deve possuir as seguintes operações:

- Vazia?
- Inserir
- Remover primeiro
- InserirTodos



Algoritmo de busca geral em árvore

Em sala

Estratégias de busca

- Uma estratégia de busca é definida pela escolha da **ordem da expansão de nós**
- Estratégias são avaliadas de acordo com os seguintes critérios:
- **completeza**: o algoritmo sempre encontra a solução se ela existe?
- **otimização**: a estratégia encontra a solução ótima?
- **complexidade de tempo**: número de nós gerados
- **complexidade de espaço**: número máximo de nós na memória

Algoritmos de Busca sem informação

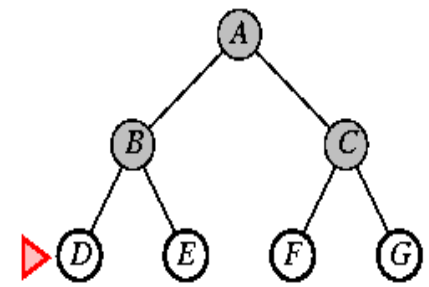
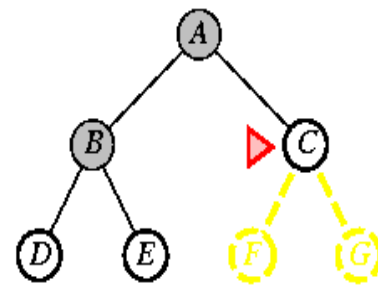
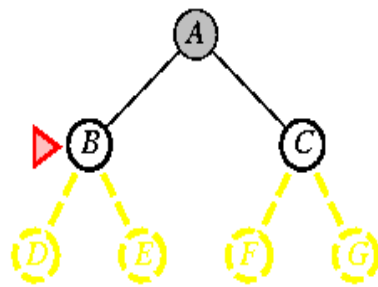
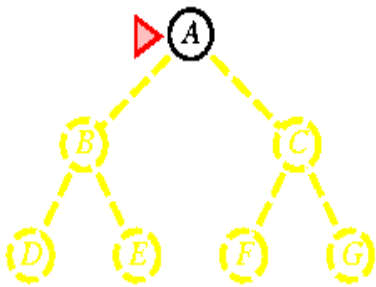
Busca Cega

- Os algoritmos que serão vistos são **sem informação**, também chamados de busca cega.
- São os mais simples, uma vez que não possuem nenhuma informação adicional além de sua definição.
- Ou seja, estes algoritmos são utilizados para ambientes com as seguintes dimensões:

Busca em extensão

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

Busca em extensão



Busca em extensão

- Completa: sempre encontra o objetivo que esteja em uma profundidade d (desde que b seja finito);
- Ótima??

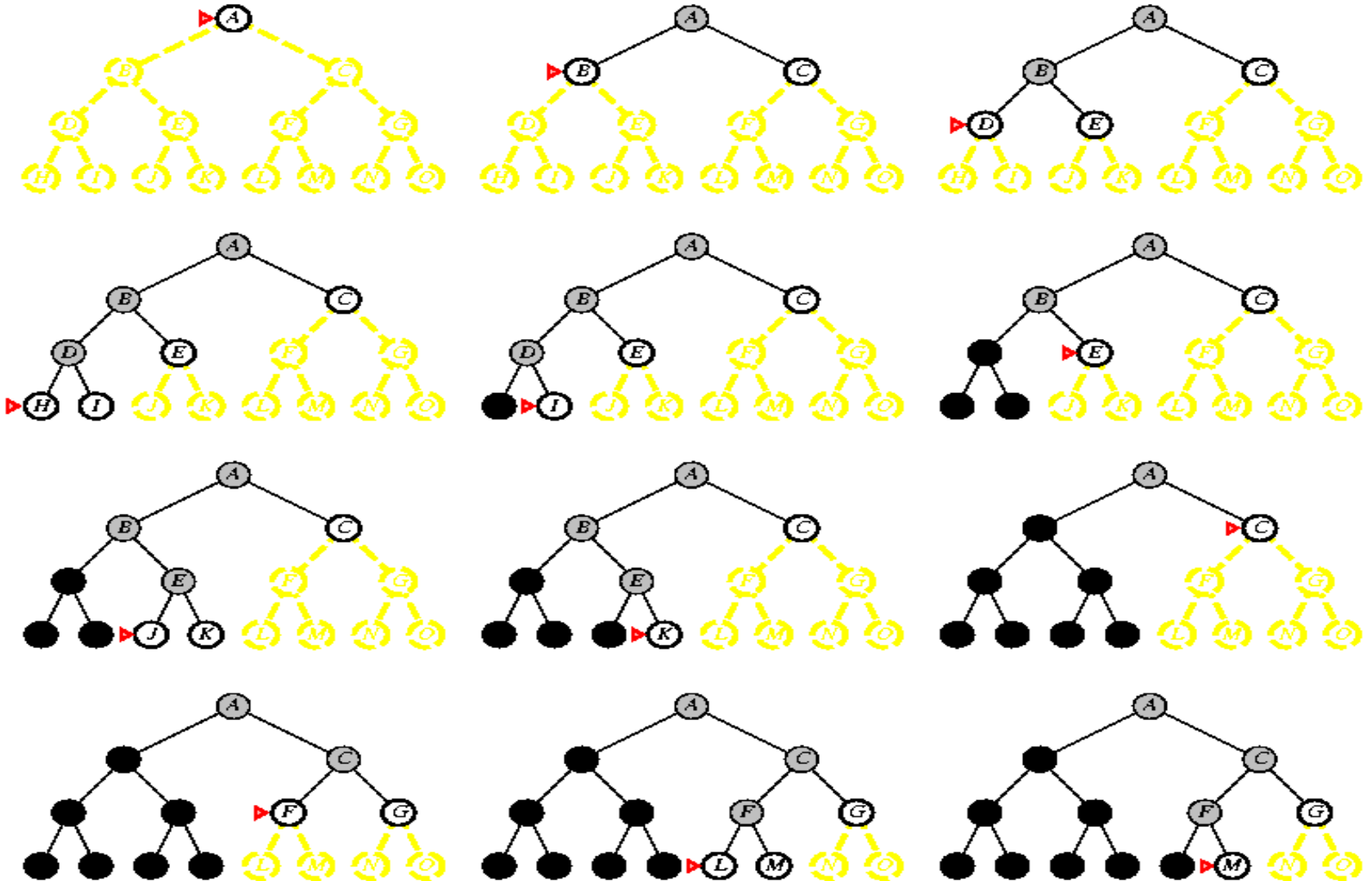
O nó objetivo mais raso não é necessariamente o nó ótimo

será ótimo se o custo do caminho for uma função não-decrescente da profundidade do nó.

Busca em profundidade

- Expande o nó mais profundo na borda atual da árvore;
- Não havendo mais sucessores, a busca retorna à próxima profundidade acima que não foi explorada.

Busca em profundidade



Como implementar?

- Existem muitas maneiras de representar um nó, mas partiremos do princípio de que um nó é uma estrutura de dados com cinco componentes:
- **ESTADO:** O estado no espaço de estados a que o nó corresponde.
- **NÓ-PAI:** O nó na árvore de busca que gerou esse nó.
- **AÇÃO:** A ação que 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 pai.

Algoritmo geral de busca em árvore

```
function TREE-SEARCH(problem, strategy) returns a solution, or failure
  initialize the search tree using the initial state of problem
  loop do
    if there are no candidates for expansion then return failure
    choose a leaf node for expansion according to strategy
    if the node contains a goal state then return the corresponding solution
    else expand the node and add the resulting nodes to the search tree
```

