

Inteligência Artificial

BCC35G

Diego Bertolini

diegobertolini@utfpr.edu.br

Aula 003

- **Aula Anterior:**
 - Agentes Inteligentes ;
- **Aula de Hoje:**
 - Resolução de Problemas por Meio de Busca ;

Objetivo

O que vocês devem saber ao final da aula:

Discutiremos como um agente pode encontrar uma sequencia de ações que alcança seus objetivos, quando nenhuma ação isolada seja capaz de fazê-lo.

Introdução

Agentes Autônomos:

Entidades capazes de observar o ambiente e agir de forma autônoma com o objetivo de atingir um determinado objetivo.

Tipos de Agentes:

Agentes reativos simples;

Agentes reativos baseado em modelo;

Agentes baseados em objetivos;

Agentes baseados na utilidade;

Agentes baseados em aprendizado;

Problema de Busca

Objetivo: Conjunto de estados que satisfazem o objetivo.

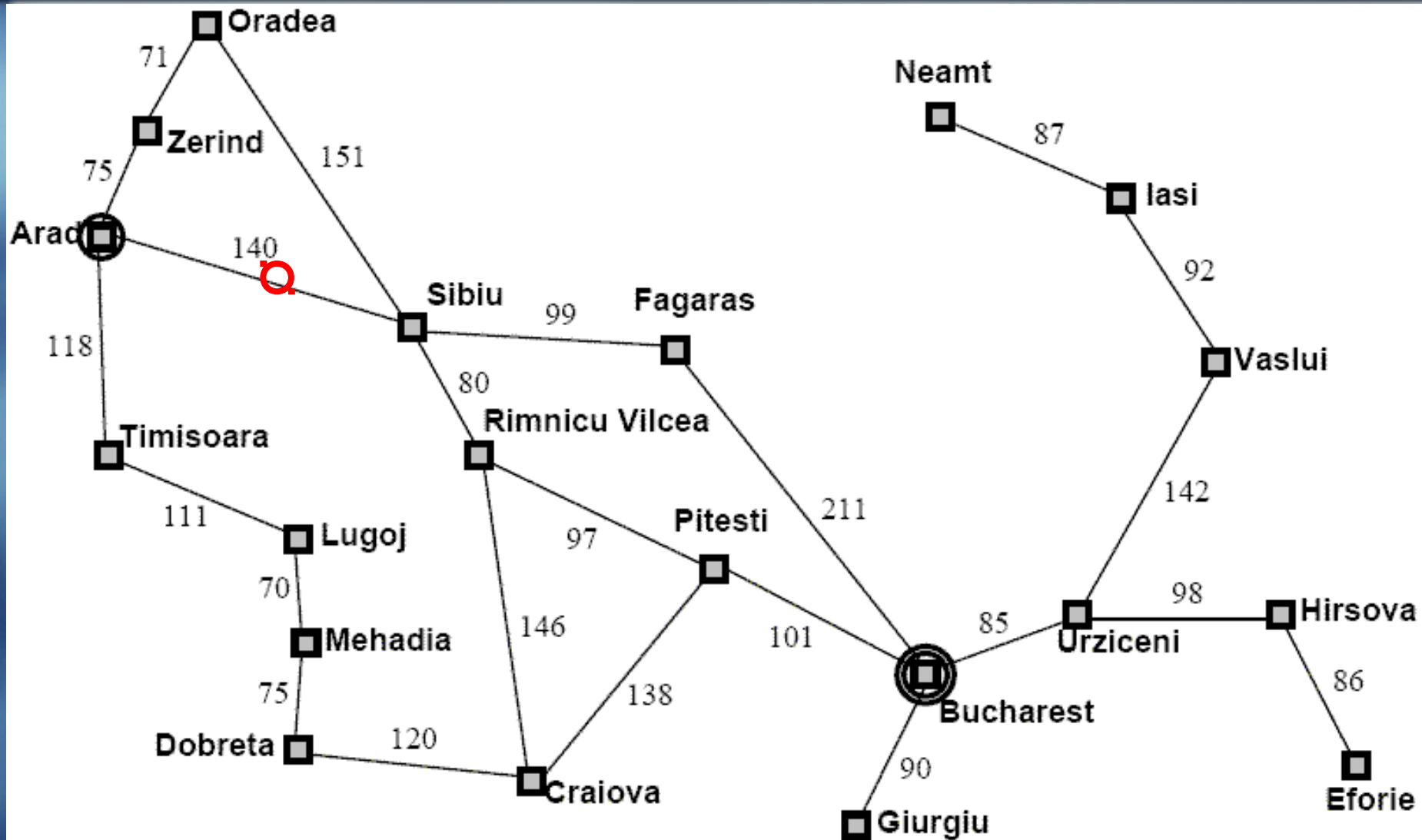
Tarefa de Busca: Encontrar a sequencia de ações que leva do estado atual até um estado objetivo.

Quais são os estados?

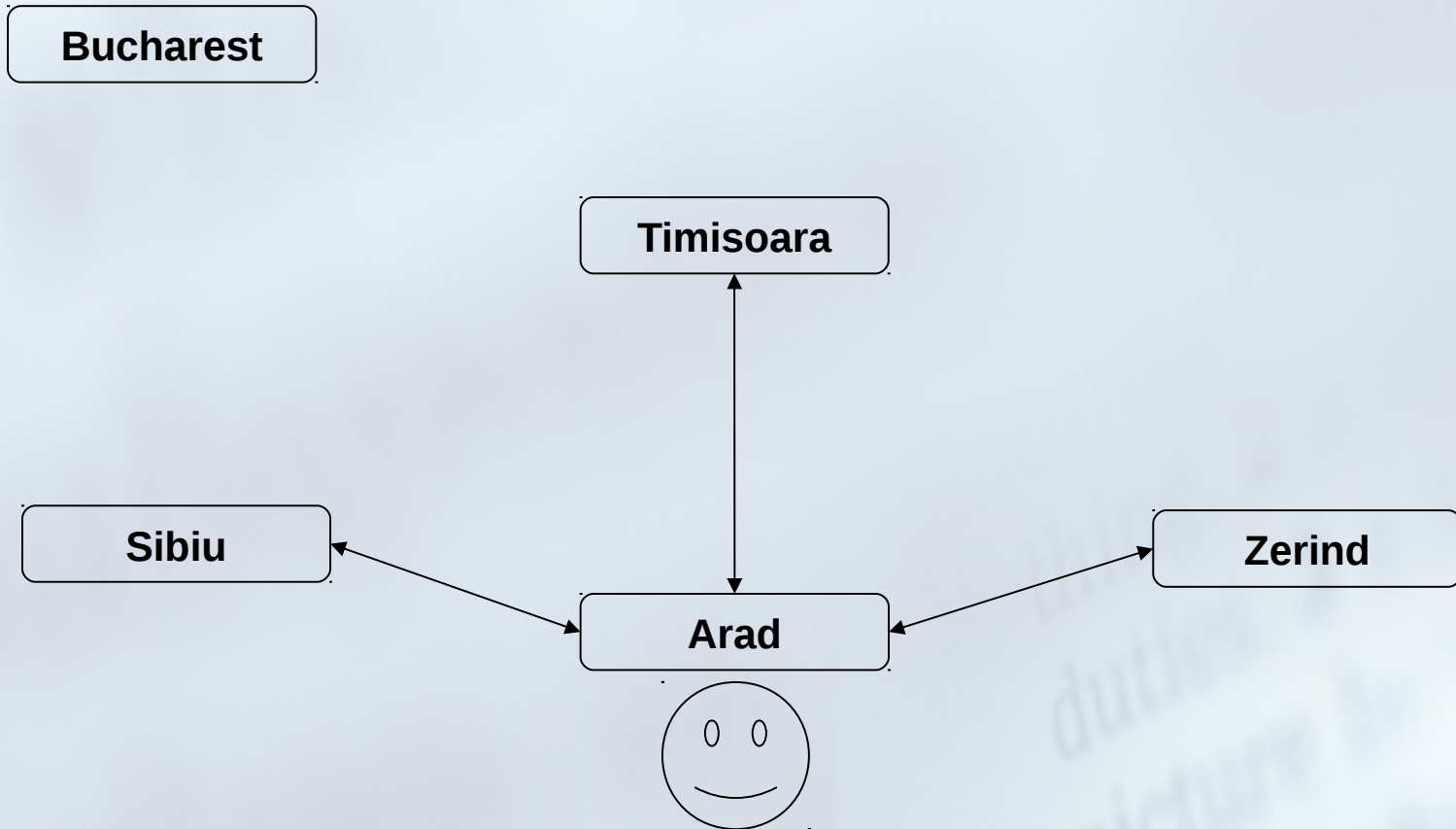
Quais são as ações?

Nível de abstração?

Problema de Busca



Problema de Busca



Problema de Busca

O processo de tentar encontrar uma sequencia de ações que leva de um estado até um estado objetivo é chamado de busca.

Uma vez encontrada a solução, o agente pode executar a sequencia de ações para chegar no objetivo.

Fases:

Formular objetivo

Buscar objetivo

Executar sequencia de ações

Definição do Problema

A definição do problema é a primeira e mais importante etapa do processo de resolução de problemas de inteligência artificial por meio de buscas.

Consiste em analisar o espaço de possibilidades de resolução do problema, encontrar sequências de ações que levem a um objetivo desejado.

Definição de um Problema

Estado Inicial: Estado inicial do agente.

Ex: Em(Arad)

Estado Final: Estado buscado pelo agente.

Ex: Em(Bucharest)

Ações Possíveis: Conjunto de ações que o agente pode executar.

Ex: Ir(Cidade, PróximaCidade)

Espaço de Estados: Conjunto de estados que podem ser atingidos a partir do estado inicial.

Ex: Mapa da Romênia.

Considerações em Relação ao Ambiente

Estático:

O Ambiente não pode mudar enquanto o agente está realizando a resolução do problema.

Observável:

O estado inicial do ambiente precisa ser conhecido previamente.

Determinístico:

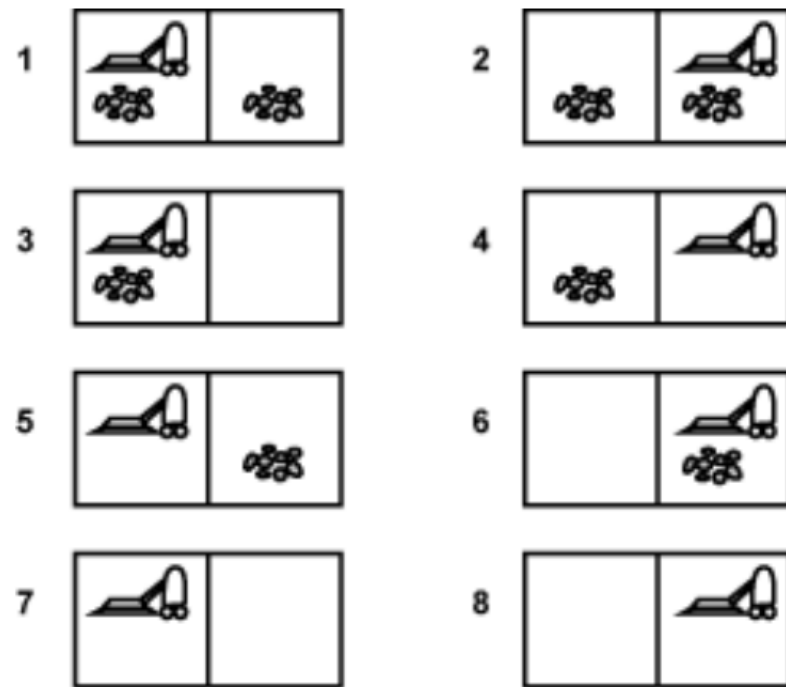
O próximo estado do agente deve ser determinado pelo estado atual + ação. A execução da ação não pode falhar.

Exemplo: Aspirador de Pó

Espaço de Estados: 8 estados possíveis (figura ao lado);

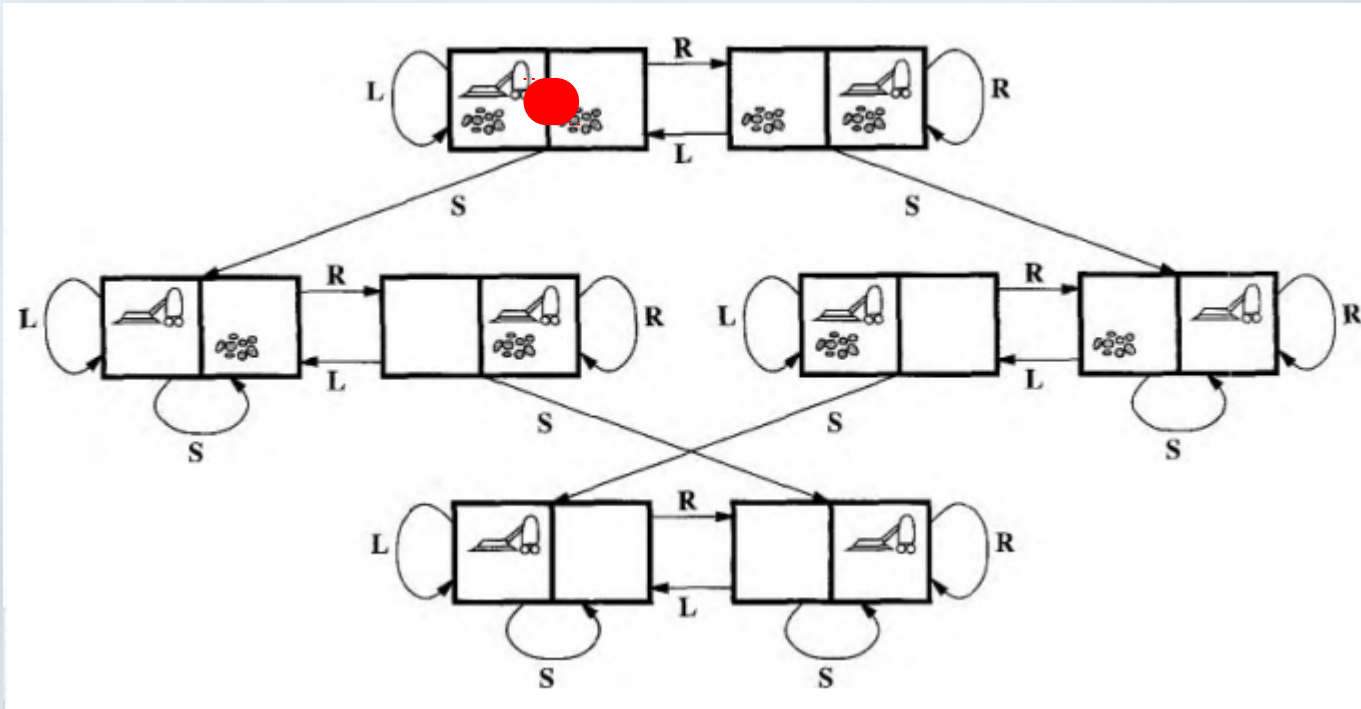
Estado Inicial: Qualquer estado;

Estado Final: Estado 7 ou 8 (ambos quadrados limpos);



Ações Possíveis: Mover

Exemplo: Aspirador de Pó



Exemplo: 8-Puzzle

Espaço de Estados: 181.440 possíveis estados;

Estado Inicial: Qualquer estado;

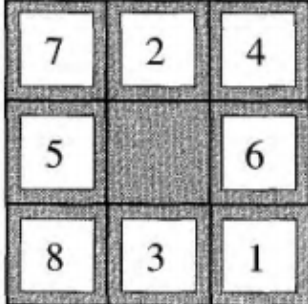
Estado Final: Figura ao lado – Goal State;

Ações Possíveis: Mover o quadrado vazio para direita, para esquerda, para cima ou para baixo;

Custo: Cada passo tem o custo 1, assim o custo do caminho é definido pelo número de passos;

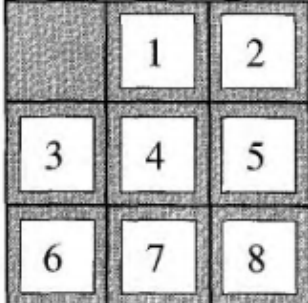
15-puzzle (4x4) – 1.3 trilhões estados possíveis.

24-puzzle (5x5) – 10^{25} estados possíveis.



| | | |
|---|---|---|
| 7 | 2 | 4 |
| 5 | | 6 |
| 8 | 3 | 1 |

Start State



| | | |
|---|---|---|
| | 1 | 2 |
| 3 | 4 | 5 |
| 6 | 7 | 8 |

Goal State

Exemplo: Xadrez

Espaço de Estados: Aproximadamente 10^{40} possíveis estados (Claude Shannon, 1950);

Estado Inicial: Posição inicial de um jogo de xadrez;

Estado Final: Qualquer estado onde o rei adversário está sendo atacado e o adversário não possui movimentos válidos;

Ações Possíveis: Regras de movimentação de cada peça do xadrez;

Custo: Quantidade de posições examinadas;



Exemplo: 8 Rainhas (Incremental)

Espaço de Estados: Qualquer disposição de 0 a 8 rainhas no tabuleiro (1.8×10^{14} possíveis estados);

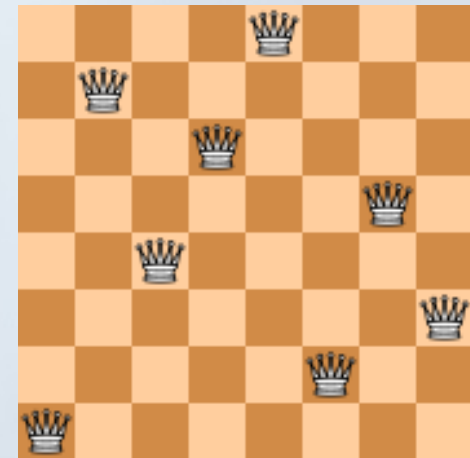
Estado Inicial: Nenhuma rainha no tabuleiro;

Estado Final: Qualquer estado onde as 8 rainhas estão no tabuleiro e nenhuma esta sendo atacada;

Ações Possíveis: Colocar uma rainha em um espaço vazio do tabuleiro;

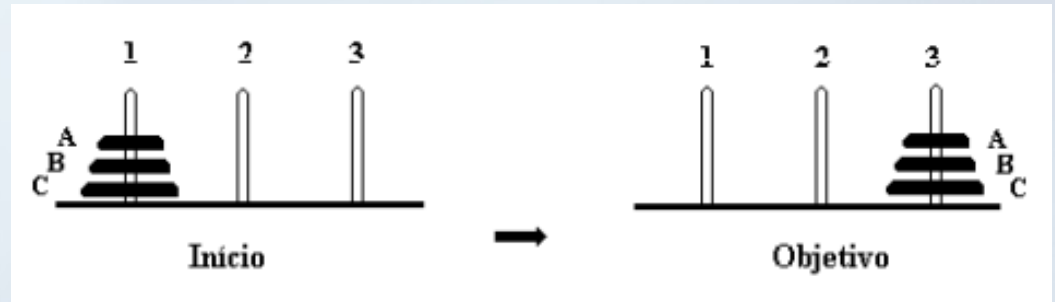
Custo: Não importa nesse caso;

* O jogo possui apenas 92 possíveis soluções (considerando diferentes rotações e reflexões). E apenas 12 soluções únicas.

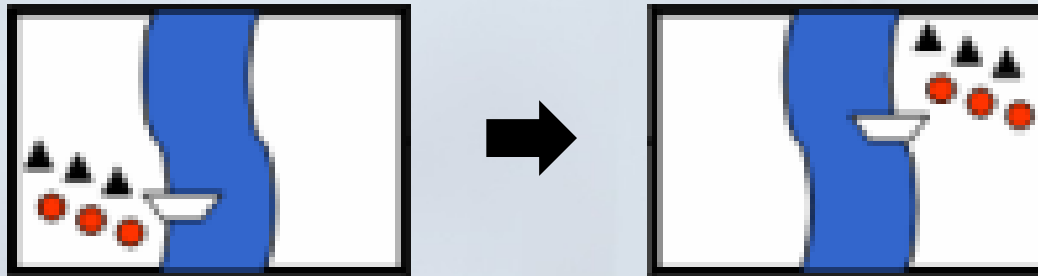


Exercícios

Torre de Hanói?



Canibais e Missionários?



Exercícios

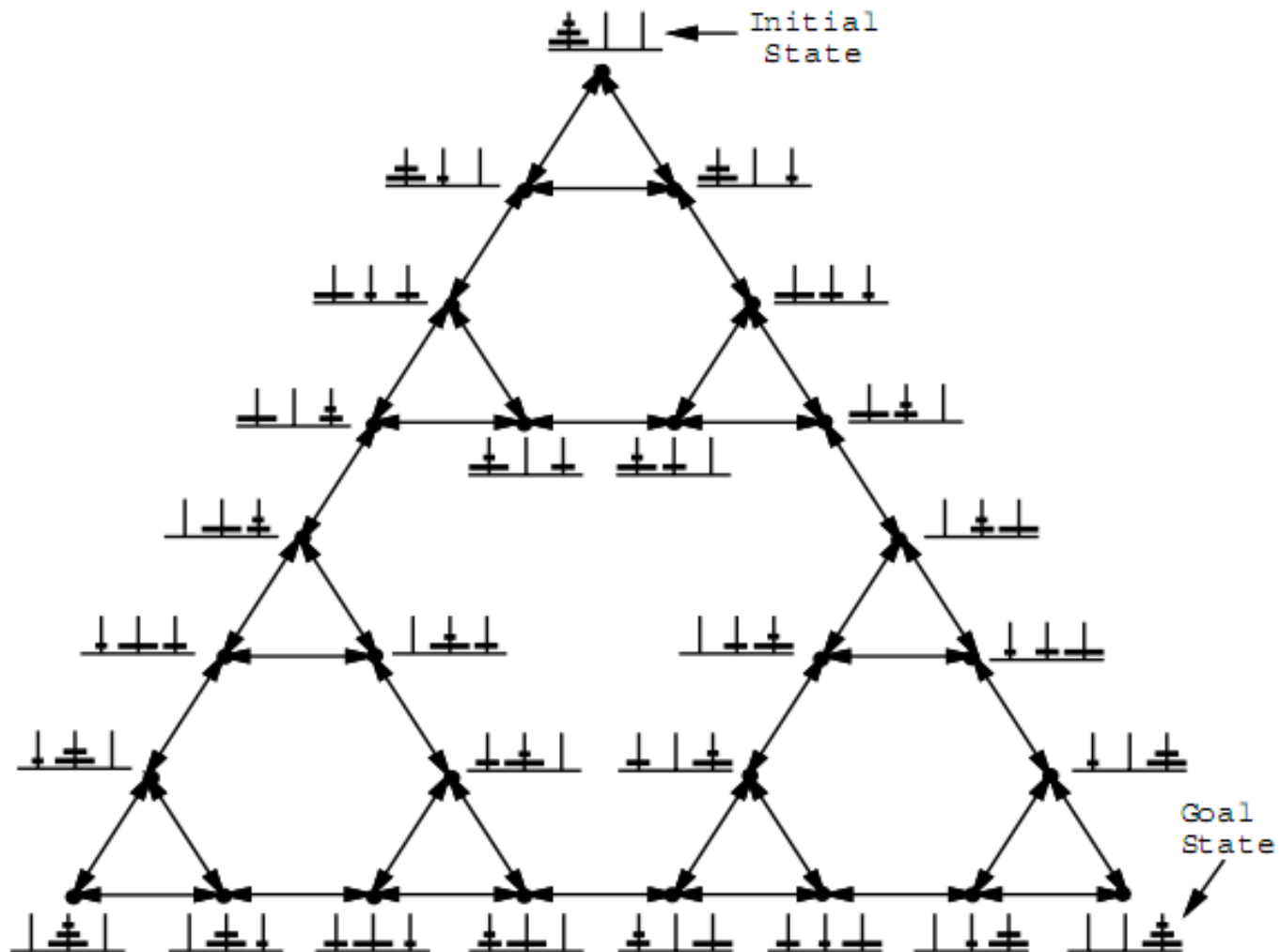
Torre de Hanói:

Espaço de Estados: Todas as possíveis configurações de argolas em todos os pinos (27 possíveis estados).

Ações Possíveis: Mover a primeira argola de qualquer pino para o pino da direita ou da esquerda.

Custo: Cada movimento tem 1 de custo.

Exercícios



Exercícios

Canibais e Missionários:

Espaço de Estados: Todas as possíveis configurações validas de canibais e missionários em cada lado do rio (16 possíveis estados).

Ações Possíveis: Mover 1 ou 2 personagens (canibais ou missionários) para o outro lado do rio. O número de canibais em um determinado lado do rio não pode ser maior do que o número de missionários.

Custo: Cada movimento tem 1 de custo.

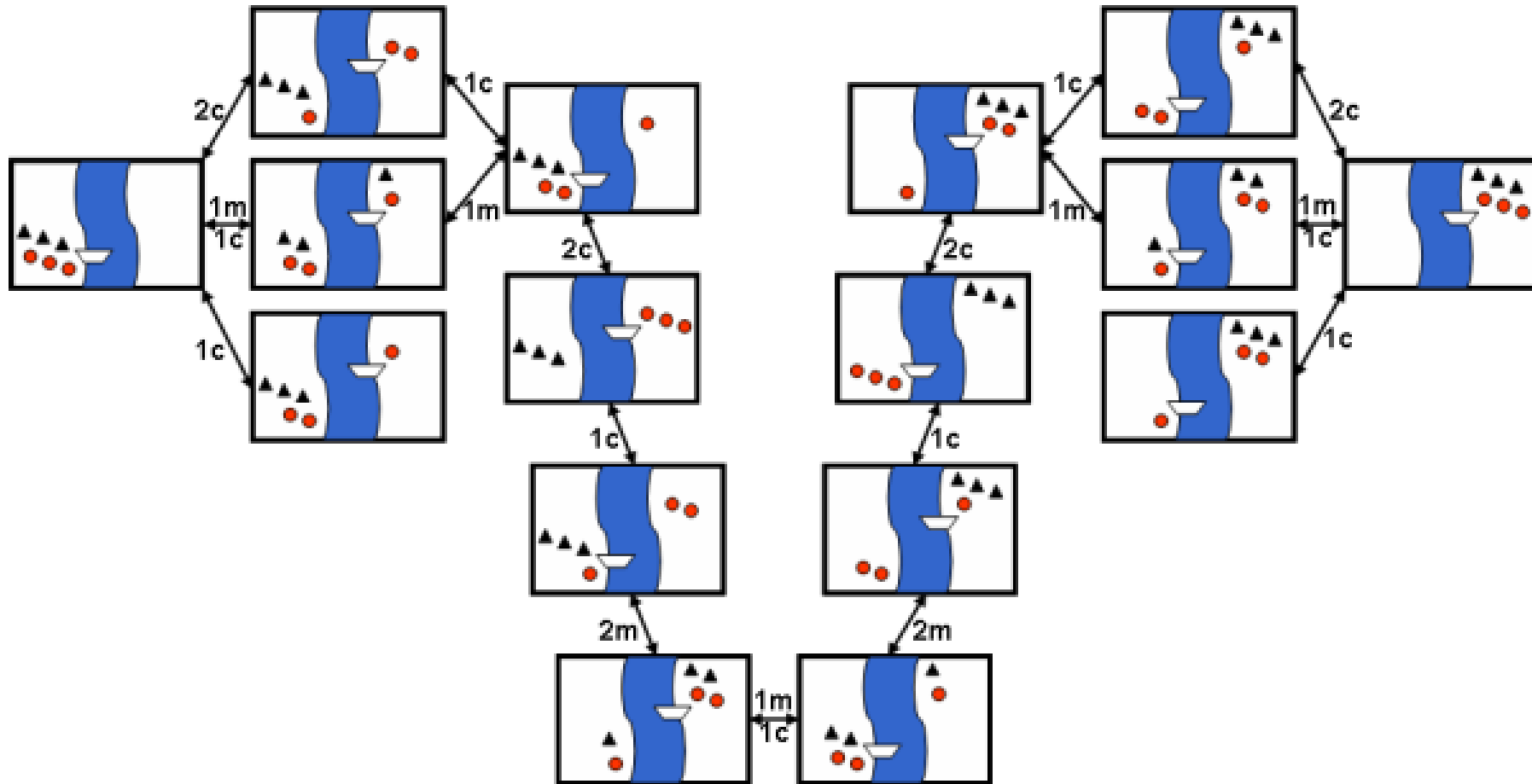
Exercícios



Canibal



Missionario



Aplicações em Problemas Reais

Cálculo de Rotas:

Planejamento de rotas de aviões;

Sistemas de planejamento de viagens;

Caixeiro viajante;

Rotas em redes de computadores;

Jogos de computadores (rotas dos personagens);

Alocação

Salas de aula;

Máquinas industriais;

Aplicações em Problemas Reais

Circuitos Eletrônicos:

Posicionamento de componentes;

Rotas de circuitos;

Robótica:

Navegação e busca de rotas em ambientes reais;

Montagem de objetos por robôs;

Como Encontrar a Solução?

Uma vez o problema bem formulado, o estado final (objetivo) deve ser “buscado” no espaço de estados.

A busca é representada em uma árvore de busca:

Raiz: corresponde ao estado inicial;

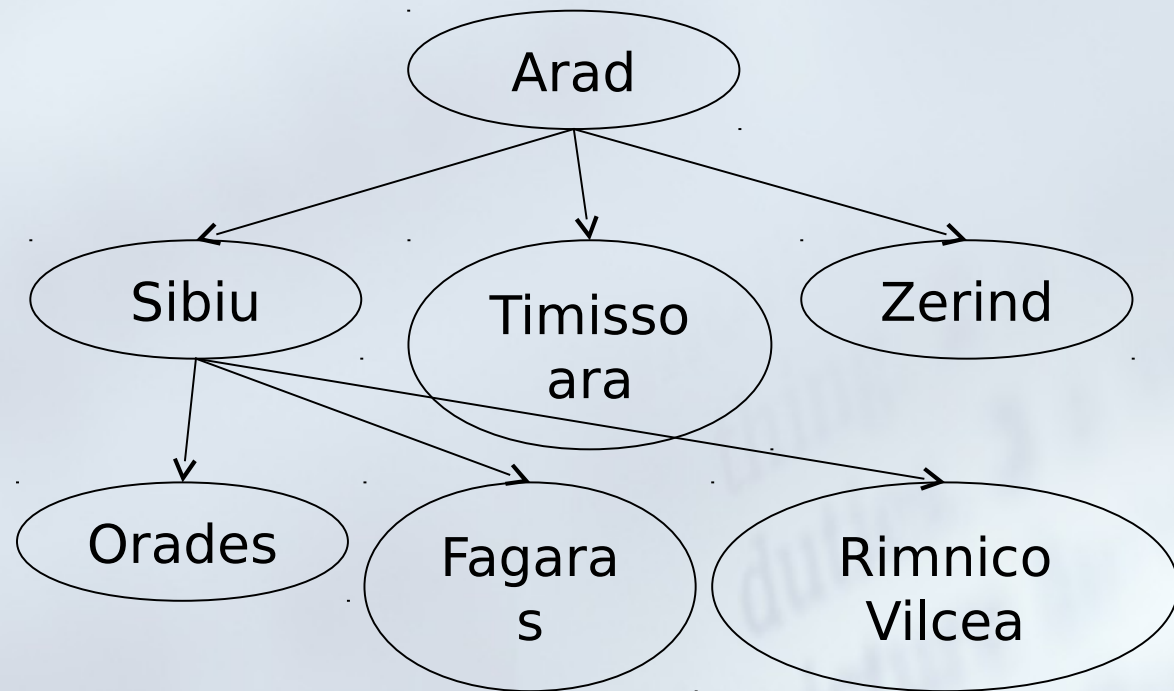
Expandem-se o estado corrente, gerando um novo conjunto de sucessores;

Escolhe-se o próximo estado a expandir seguindo uma estratégia de busca;

Prossegue-se até chegar ao estado final (solução) ou falhar na busca pela solução;

Buscando Soluções

Exemplo: Ir de Arad para Bucharest



Buscando Soluções

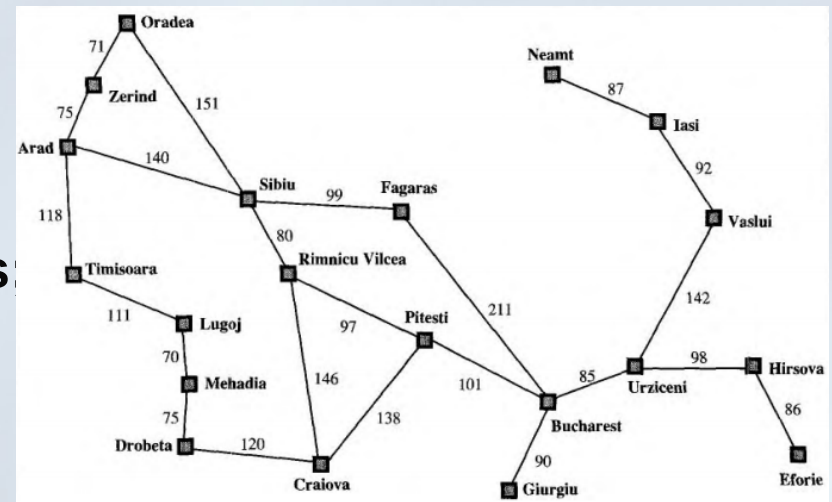
O espaço de estados é diferente da árvore de buscas.

Exemplo:

20 estados no espaço de estados

Número de caminhos infinito;

Árvore com infinitos nós;



Medida de Desempenho

Desempenho do Algoritmo:

(1) O algoritmo encontrou alguma solução?

(2) É uma boa solução?

Custo de caminho (qualidade da solução).

(3) É uma solução computacionalmente barata?

Custo da busca (tempo e memória).

Custo Total

Custo do Caminho + Custo de Busca.

Métodos de Busca

Busca Cega ou Exaustiva:

Não sabe qual o melhor nó da fronteira a ser expandido. Apenas distingue o estado objetivo dos não objetivos.

Busca Heurística:

Estima qual o melhor nó da fronteira a ser expandido com base em funções heurísticas.

Busca Local:

Operam em um único estado e movem-se para a vizinhança deste estado.

Busca Cega

Algoritmos de Busca Cega:

Busca em largura;

Busca de custo uniforme;

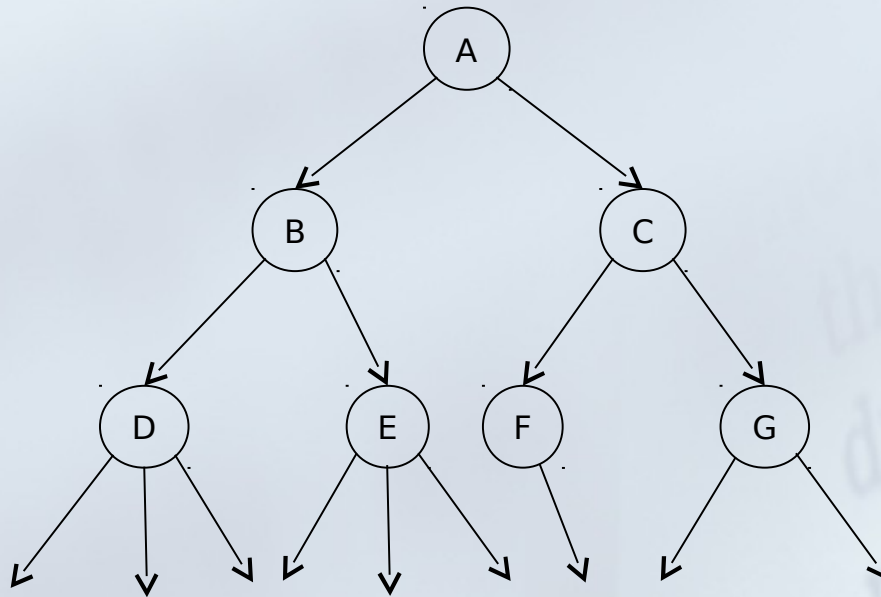
Busca em profundidade;

Busca com aprofundamento iterativo;

Busca em Largura

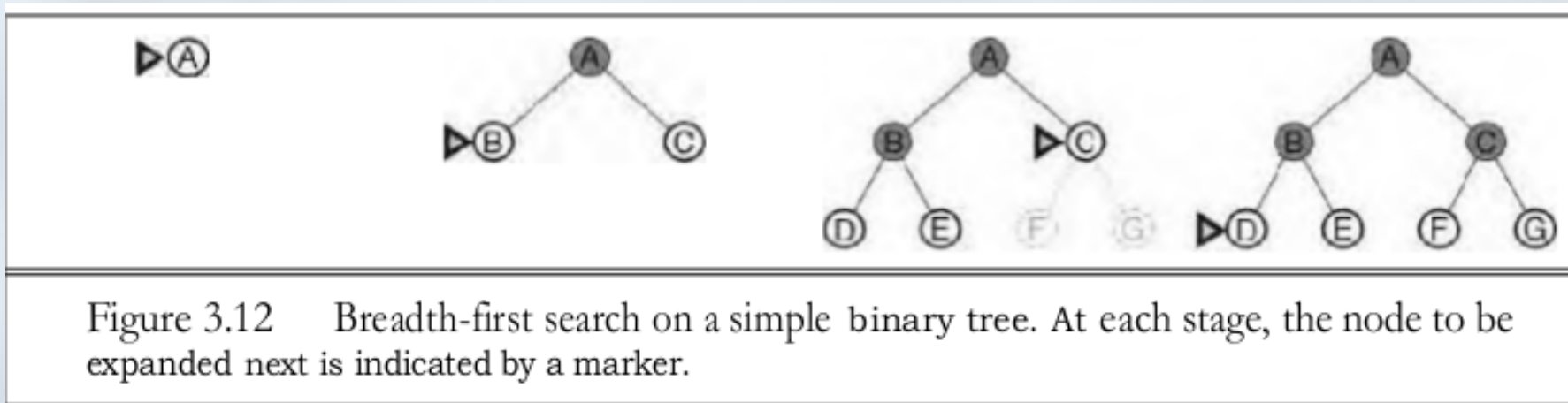
Estratégia:

O nó raiz é expandido, em seguida todos os nós sucessores são expandidos, então todos próximos nós sucessores são expandidos, e assim em diante.



Busca em Largura

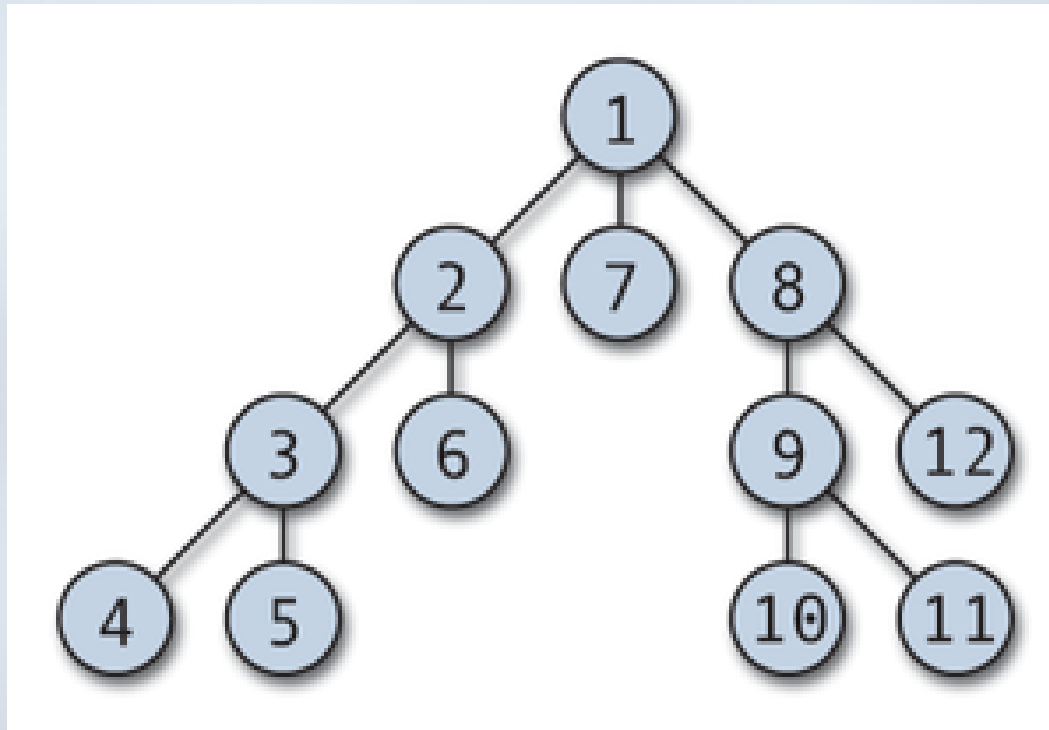
Estratégia:



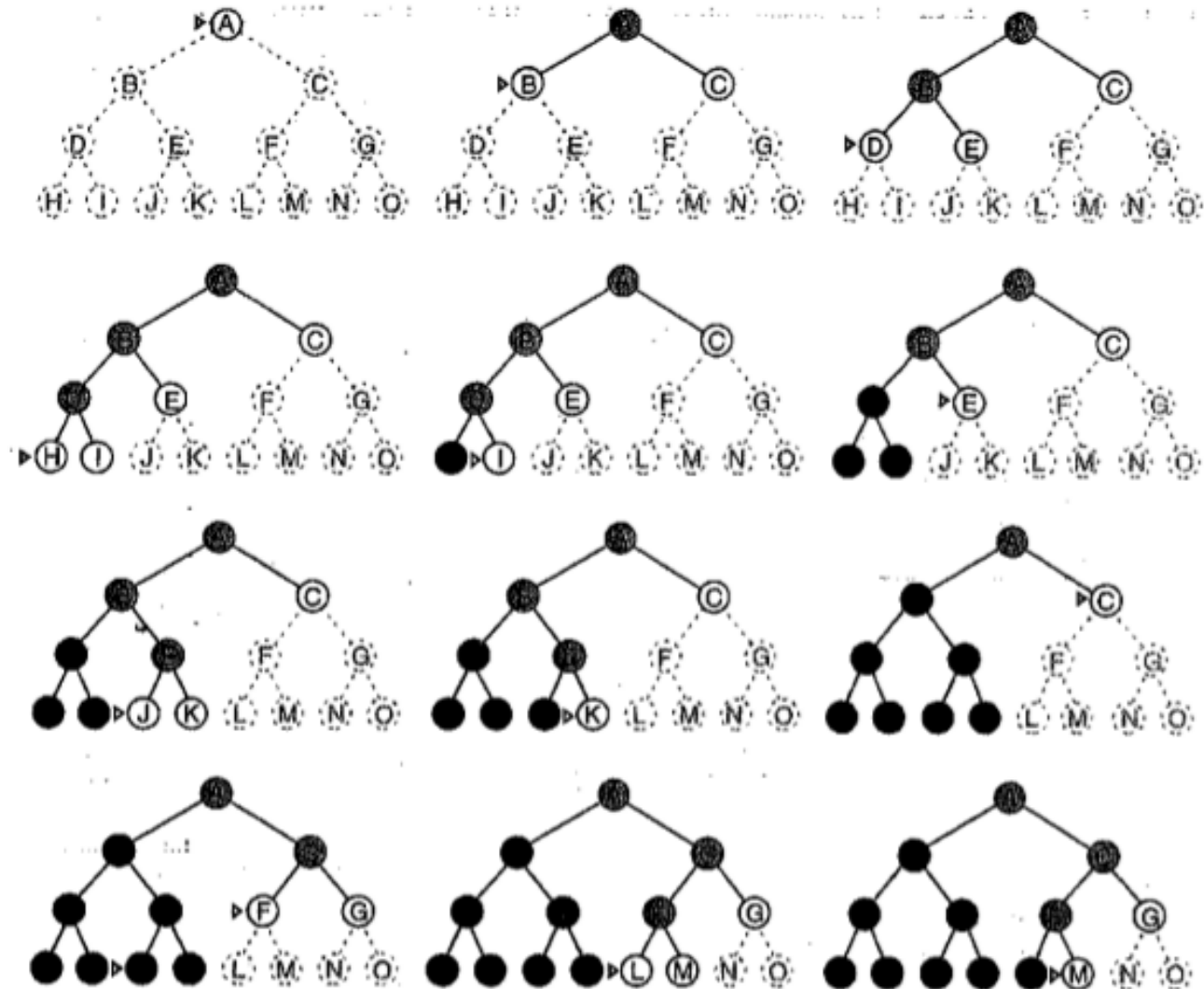
Busca em Profundidade

Estratégia:

Expande os nós da vizinhança até o nó mais profundo.



Busca em Profundidade



Busca em Profundidade

Pode ser implementado com base no pseudocódigo da função “BuscaEmArvore” apresentado anteriormente. Utiliza-se uma estrutura de pilha (last-in-first-out) para armazenar os nós da fronteira.

Pode também ser implementado de forma recursiva.

Consome pouca memória, apenas o caminho de nós sendo analisados precisa armazenado. Caminhos que já foram explorados podem ser descartados da memória.

Busca em Profundidade

Uso de memória pela busca em largura em uma árvore com 12 de profundidade: 1000 TB.

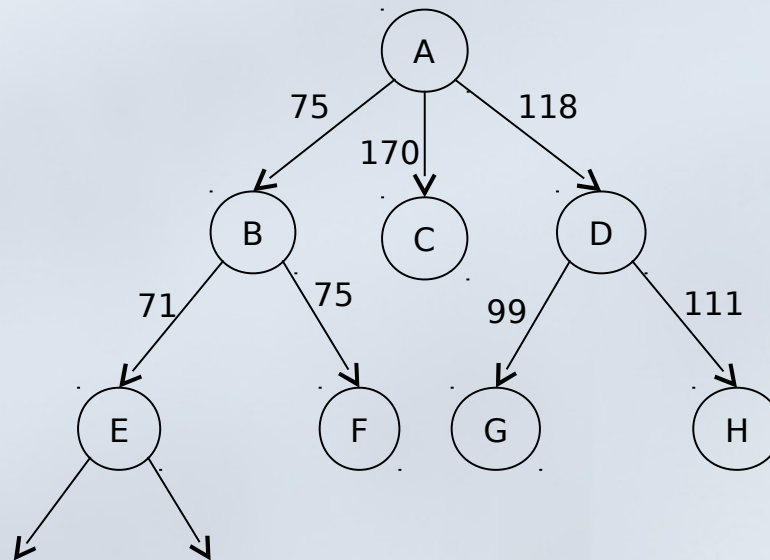
Uso de memória pela busca em profundidade em uma árvore com 12 de profundidade: 118 KB.

Problema: O algoritmo pode fazer uma busca muito longa mesmo quando a resposta do problema está localizado a poucos nós da raiz da árvore.

Busca de Custo Uniforme

Estratégia:

Expande sempre o nó de menor custo de caminho. Se o custo de todos os passos for o mesmo, o algoritmo acaba sendo o mesmo que a busca em largura.



Busca de Custo Uniforme

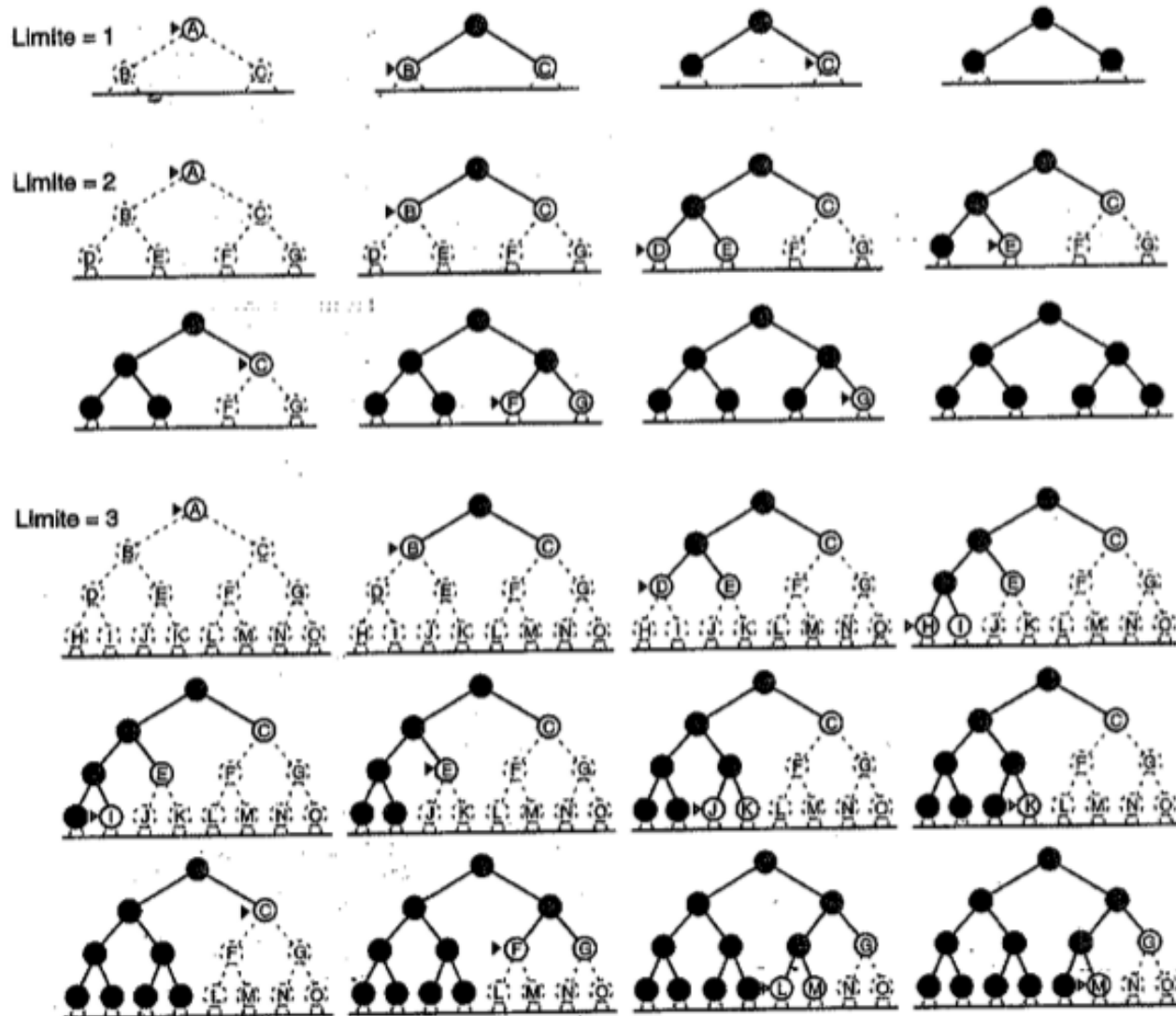
A primeira solução encontrada é a solução ótima se custo do caminho sempre aumentar ao longo do caminho, ou seja, não existirem operadores com custo negativo.

Implementação semelhante a busca em largura. Adiciona-se uma condição de seleção dos nós a serem expandidos.

Busca com Aprofundamento Iterativo

Estratégia: Consiste em uma busca em profundidade onde o limite de profundidade é incrementado gradualmente.

Busca com Aprofundamento Iterativo



Busca com Aprofundamento Iterativo

Combina os benefícios da busca em largura com os benefícios da busca em profundidade.

Evita o problema de caminhos muito longos ou infinitos.

A repetição da expansão de estados não é tão ruim, pois a maior parte dos estados está nos níveis mais baixos.

Cria menos estados que a busca em largura e consome menos memória.

Como evitar estados repetidos?

Estados repetidos sempre vão ocorrer em problema onde os estados são reversíveis.

Como evitar?

Não retornar ao estado “pai”.

Não retorna a um ancestral.

Não gerar qualquer estado que já tenha sido criado antes (em qualquer ramo).

Requer que todos os estados gerados permaneçam na memória.

Bibliografia e Materiais.

Estes slides foram adaptados do Livro:

Russell, S. and Norvig, P. Artificial Intelligence: a Modern Approach, 2nd Edition, Prentice-Hall, 2003. Capítulo 3: Solving Problems by Searching;

Adaptado das Aulas do Professor: Ederley – PU

