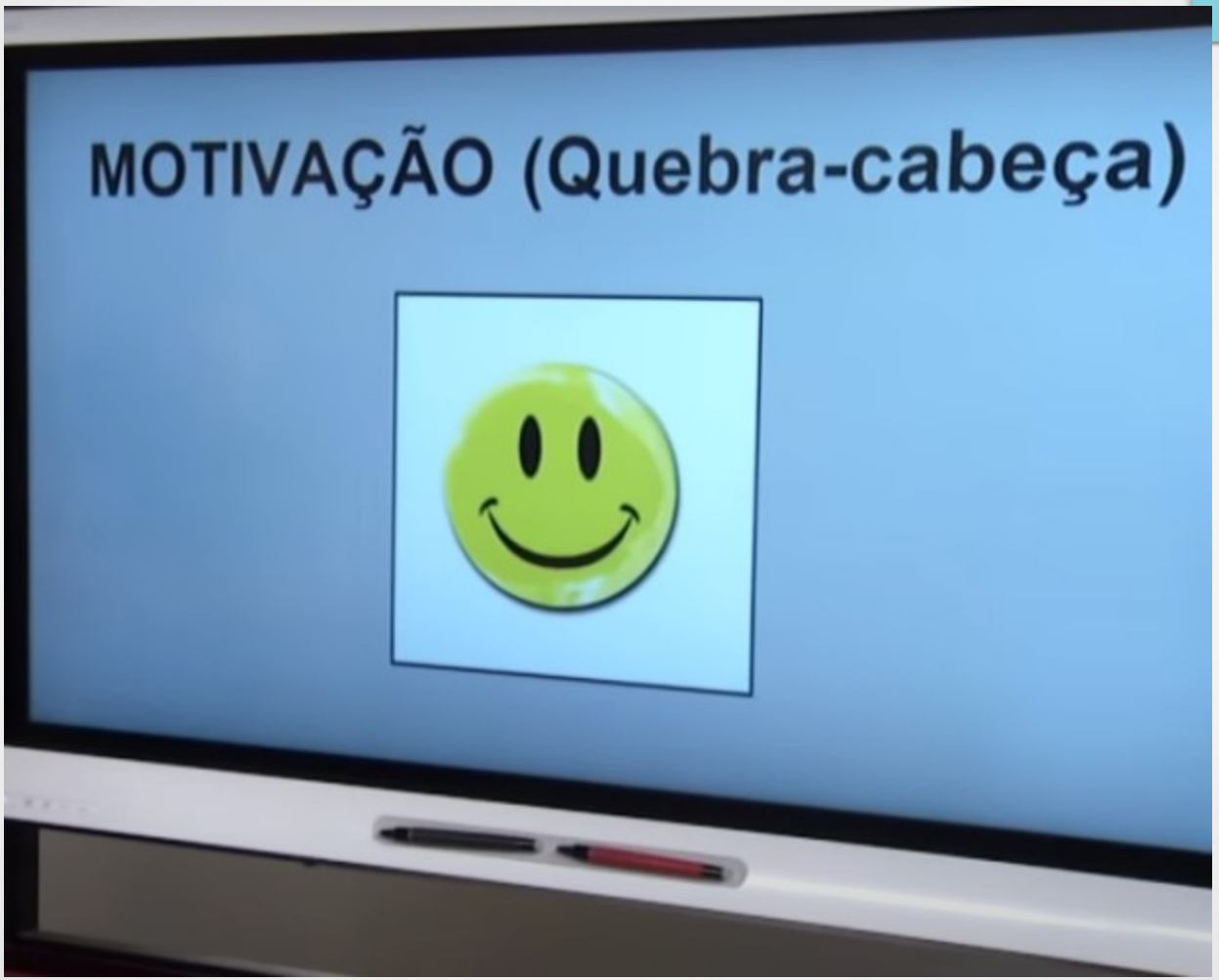
Inteligência Artificial
Paradigma simbolista
Buscas

Prof. Gizelle / Marcelo Dib

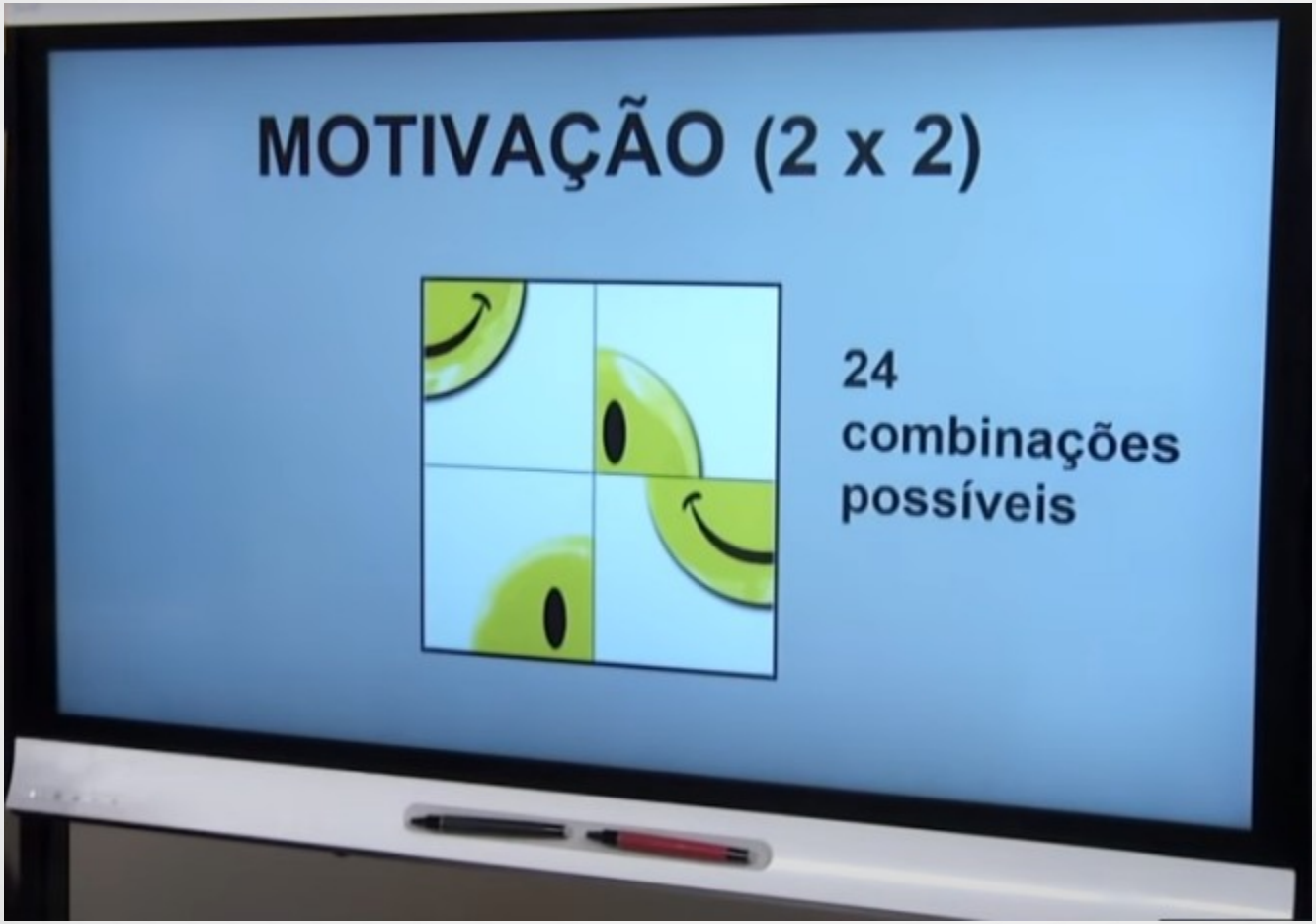
Buscas

No **Paradigma Simbolista**, os mecanismos efetuam transformações utilizando símbolos, letras, números ou palavras. Simulam, portanto, o raciocínio lógico por trás das linguagens com as quais os seres humanos se comunicam uns com os outros.

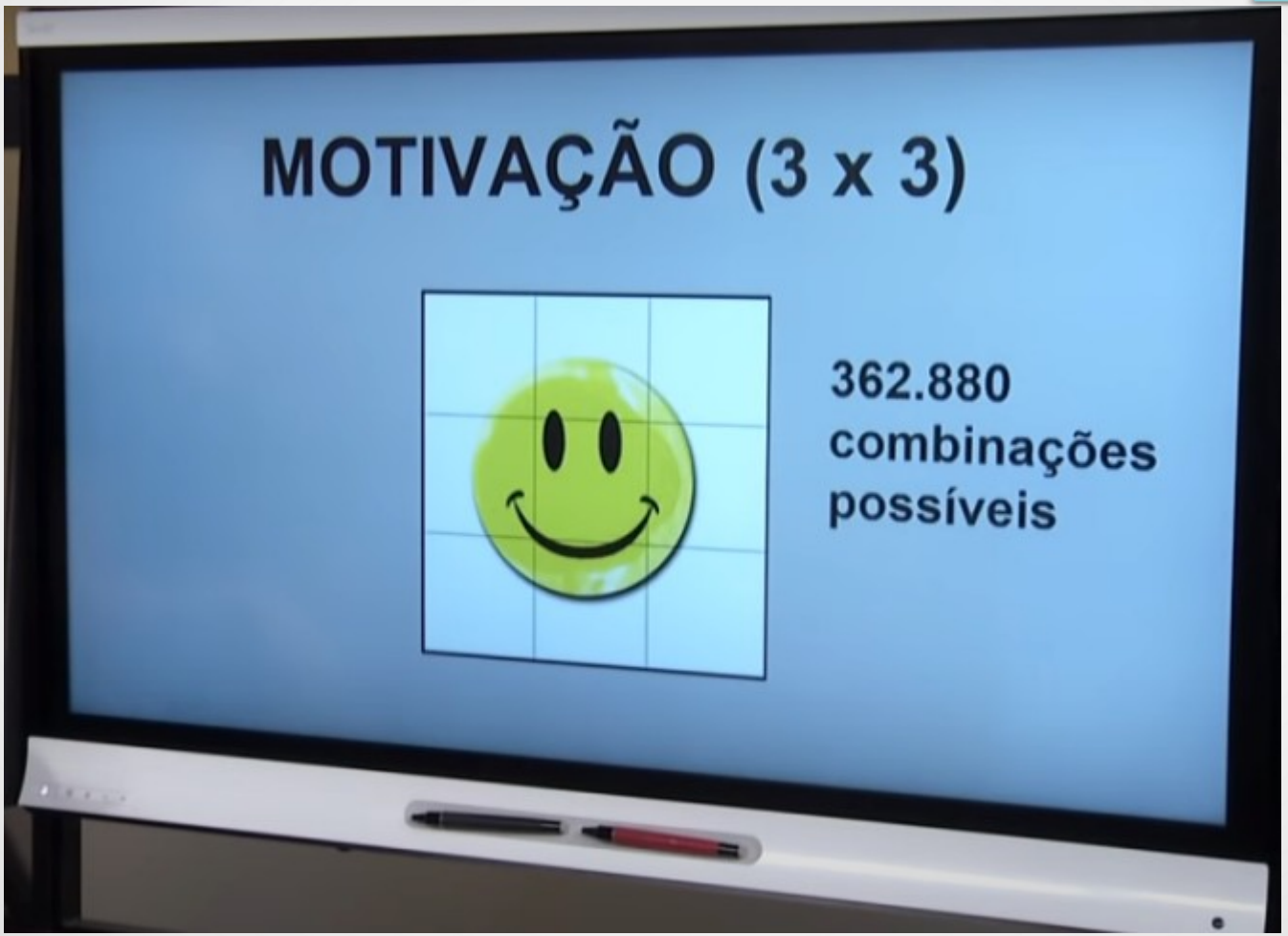
Buscas



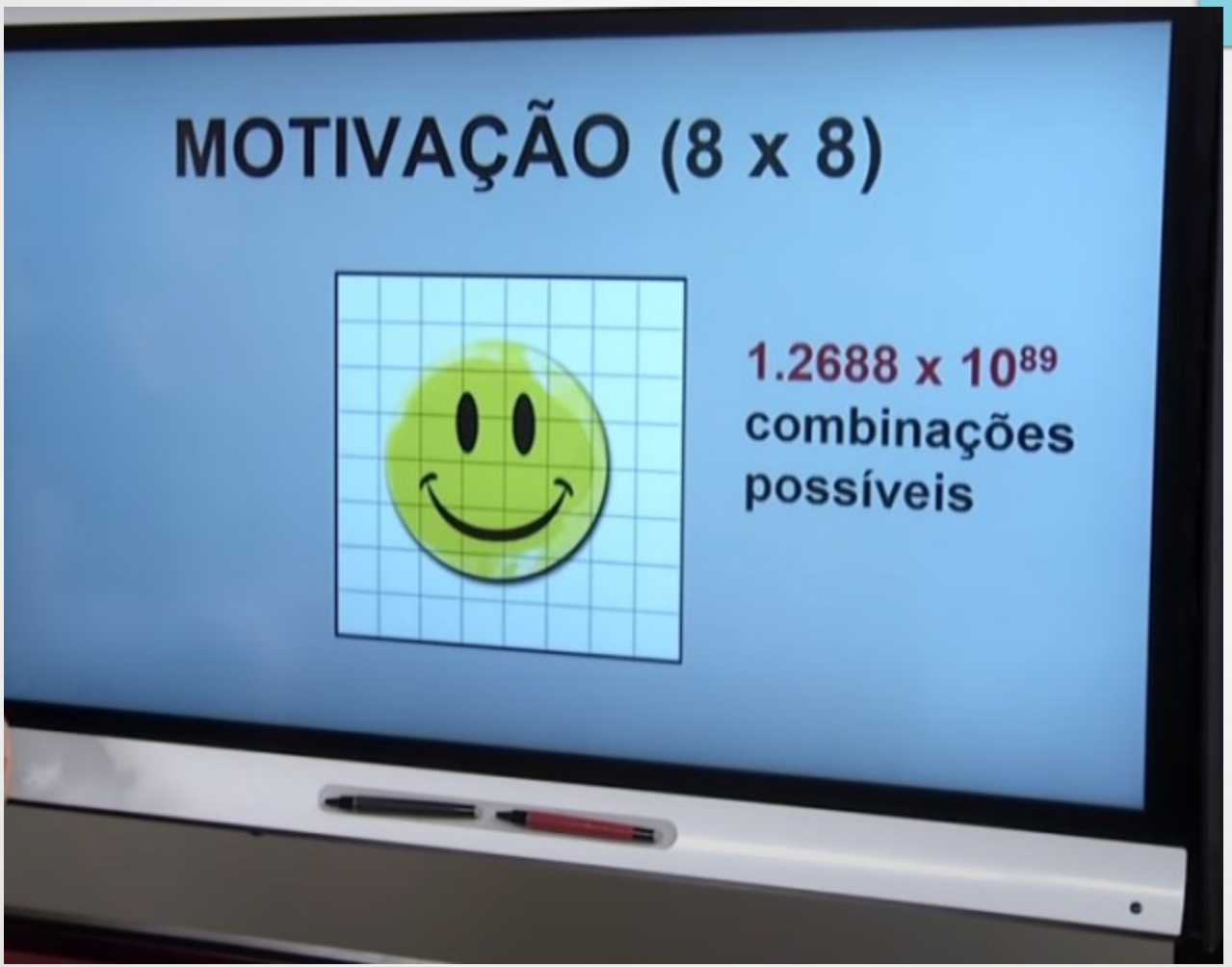
Buscas



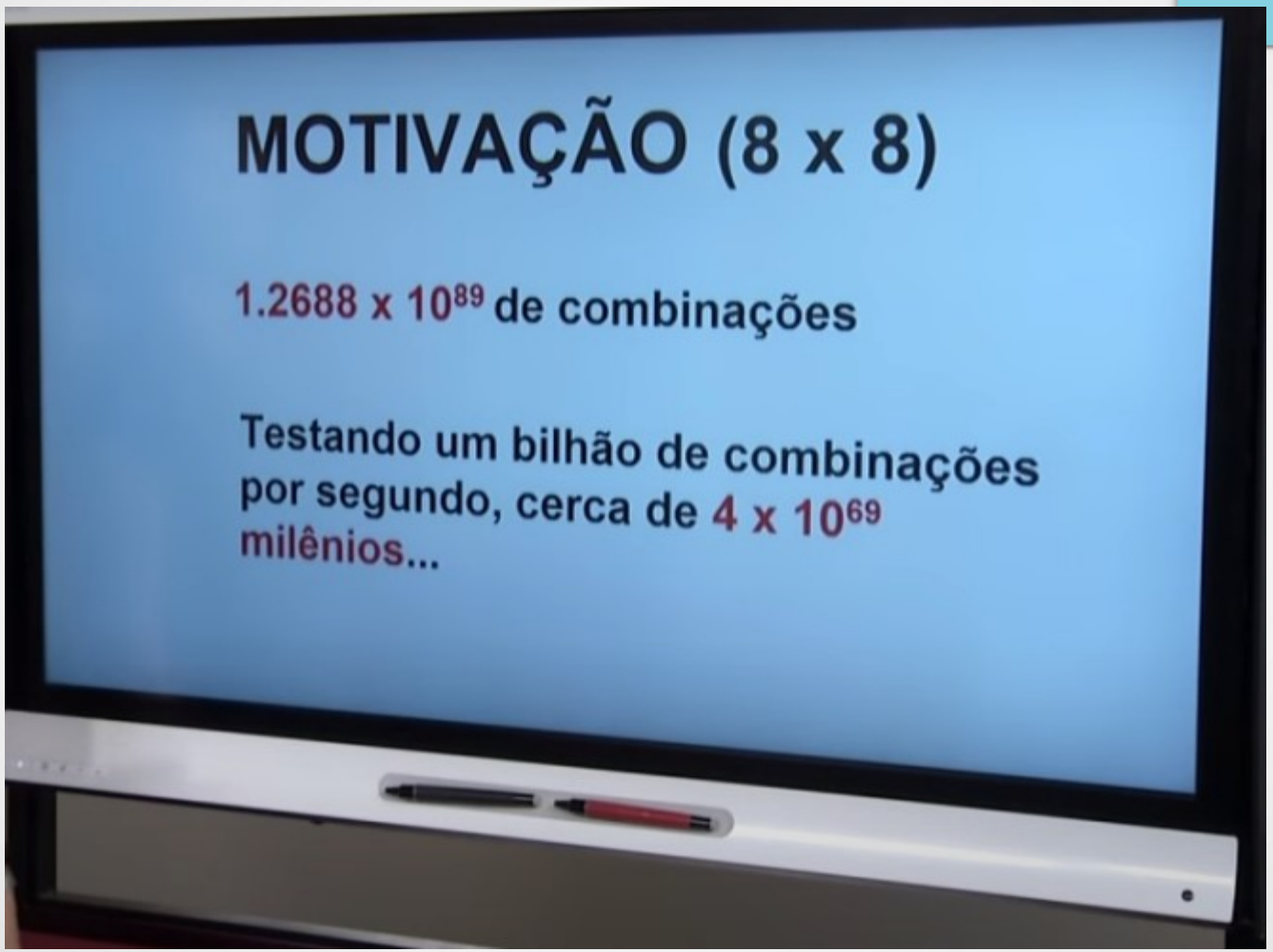
Buscas



Buscas



Buscas


MOTIVAÇÃO (8 x 8)
1.2688 x 10^89 de combinações
Testando um bilhão de combinações por segundo, cerca de 4 x 10^69 milênios...

Buscas

- Por que nós, humanos, podemos resolver muito mais rápido ?
  - Nós não testamos todas as possibilidades;
  - Utilizamos conhecimento do problema de forma inteligente !!!!

Buscas

- Alguns problemas de IA são aqueles onde não se conhece uma forma direta de resolvê-los.
- Como não há conhecimento sobre o caminho da solução, é preciso buscá-la de alguma forma.
- O objetivo em um Sistema de Produção é obter uma seqüência de operadores que levem um problema do estado inicial a um estado final.

Buscas

- Problemas com características que os tornam bons candidatos para a pesquisa em IA.
  - São solucionáveis por seres-humanos e, neste caso, sua solução está associada à inteligência;
  - Formam classes de complexidade variável existindo desde instâncias triviais (por exemplo, o jogo da velha, no caso dos jogos) até instâncias extremamente complexas (xadrez).

Buscas

- Problemas com características que os tornam bons candidatos para a pesquisa em IA.
  - São problemas de conhecimento total, isto é, tudo que é necessário para solucioná-los é conhecido, o que facilita sua formalização;
  - Suas soluções têm a forma de uma seqüência de situações legais e as maneiras de passar de uma situação para outra são em número finito e conhecidas.

Buscas

- Problema ?
  - O que é ?
  - Como representar ?
  - Como resolver ?

Buscas

- Problema ?
- De maneira geral, um problema de busca pode ser formalizado através da definição dos seguintes elementos:
    - Um conjunto de descrições chamado espaço de estados,onde cada elemento descreve uma situação possível do problema.
    - Um estado inicial que descreve a situação inicial do problema.

Buscas

- Problema ?
  - Um estado inicial que descreve a situação inicial do problema.
  - Um ou mais estados finais, isto é, as situações que se deseja alcançar.
  - Um conjunto de operadores, isto é, procedimentos que, dada a descrição de um estado, determinam todos os estados que podem ser alcançados a partir do estado dado.

Buscas

- PROBLEMA
  - Formular objetivo:
  - Formular problema:
  - Estados:
  - Operadores:
  - Achar solução
  - Custo

Buscas

- PROBLEMA
  - Estrutura de Dados utilizada
  - Arvores
  - Grafos

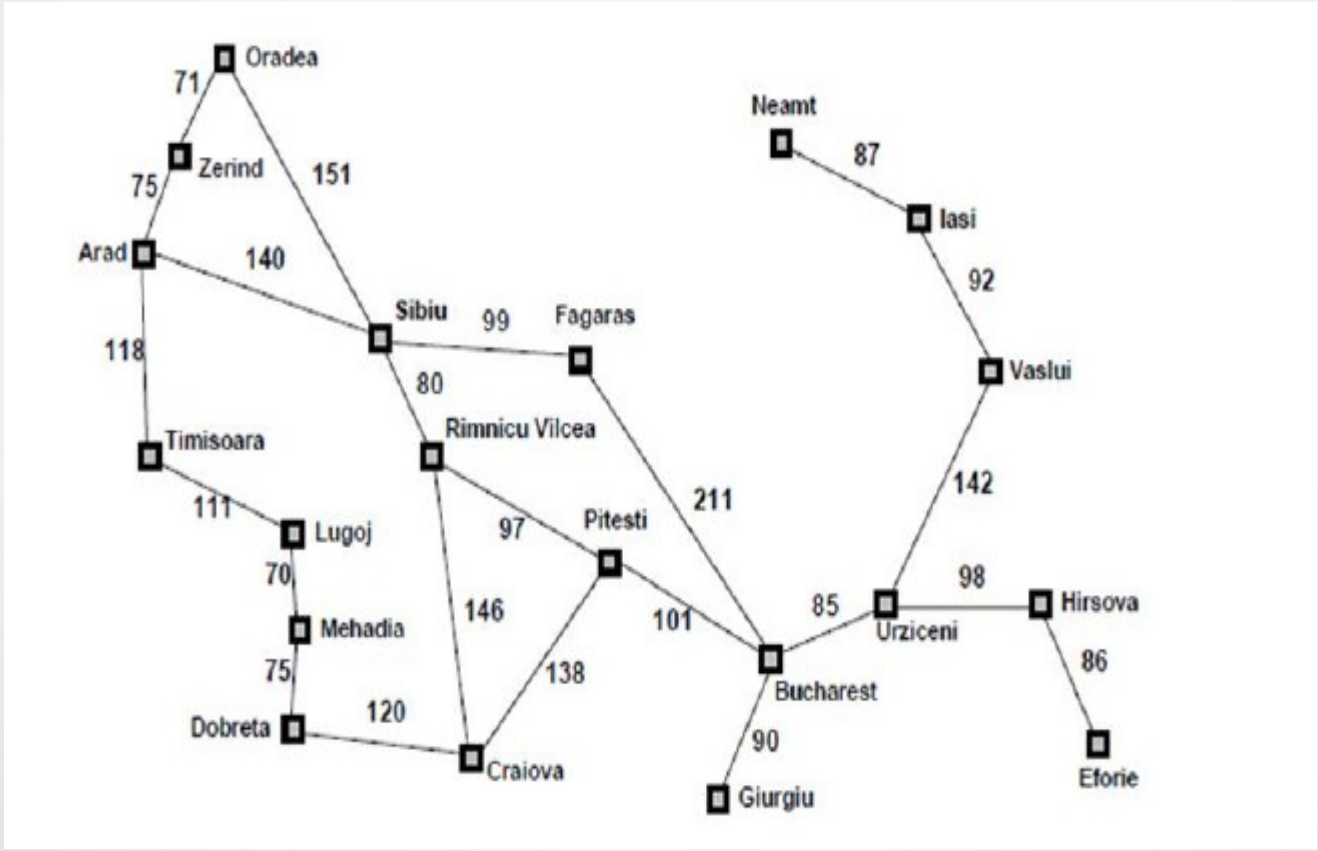
Buscas

- A forma sistemática de percorrer o grafo à procura desta seqüência é chamada de Algoritmo (ou Método) de Busca.
- Um algoritmo de busca constrói uma Árvore de Busca:
  - A raiz é o estado inicial,
  - Os nós filhos são obtidos pela aplicação dos operadores,

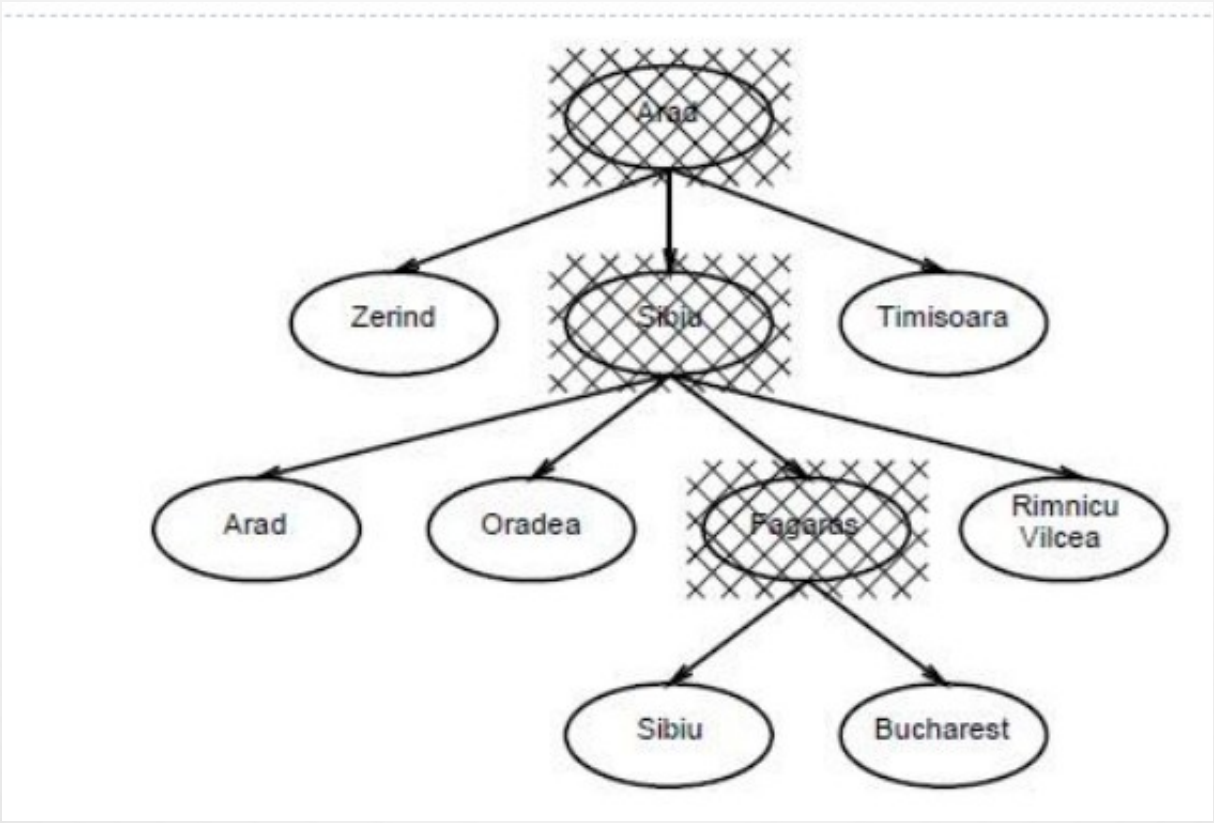
Buscas

- Exemplo1 : Em férias na Romênia, atualmente em Arad.
- Formular objetivo: Estar em Bucareste
- Formular problema:
  - Estados: cidades
  - Operadores: dirigir entre cidades
- Achar solução: seqüência de cidades
- Custo : distancia total percorrida

Buscas


Oradea
71
Neamt
87
Zerind
151
75
Iasi
Arad
140
92
Sibiu
99
Fagaras
118
Vaslui
80
Rimnicu Vilcea
Timisoara
142
211
111
Pitesti
Lugoj
97
98
70
85
Hirsova
Mehadia
146
101
Urziceni
75
86
138
Bucharest
Dobreta
120
90
Craiova
Eforie
Giurgiu

Buscas


Arad
Zerind
Sibiu
Timisoara
Arad
Oradea
Fagaras
Rimnicu Vilcea
Sibiu
Bucharest

Buscas

- Exemplo2:

Problema do aspirador: Um robô aspirador deve limpar duas salas contíguas. Modelar o problema significa abstrair as carcterísticas relevantes para o problema e desprezar as demais.

Formular objetivo: Limpeza total

Formular problema:

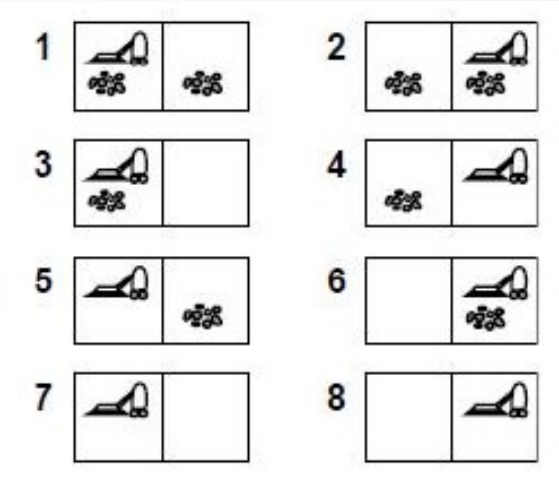
Estados:

Operadores: Esquerda,

Direita, Aspira

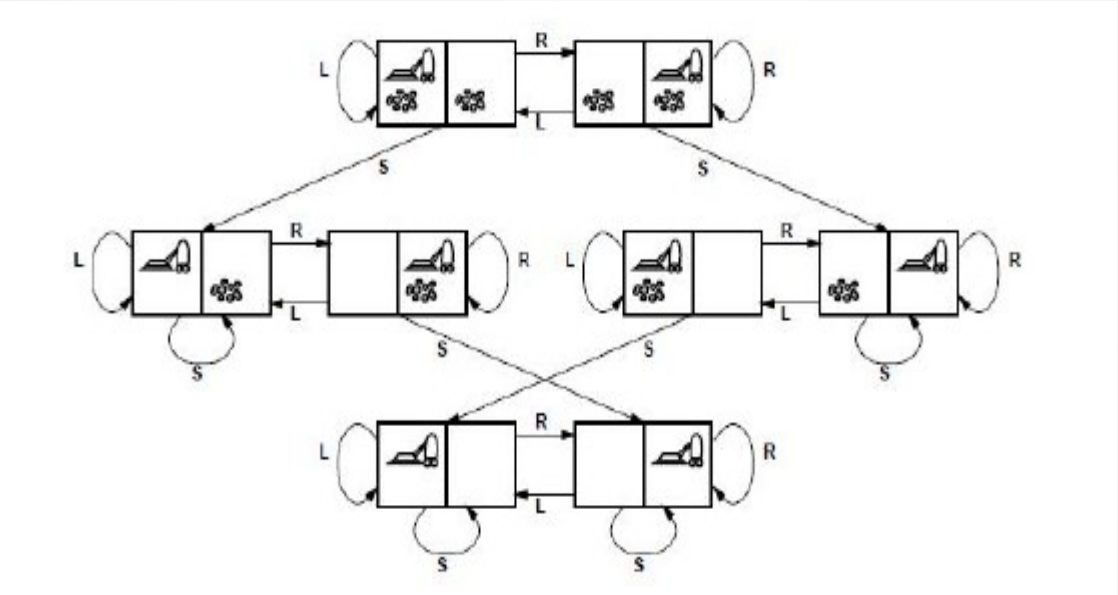
Achar solução: sequencia

Custo :



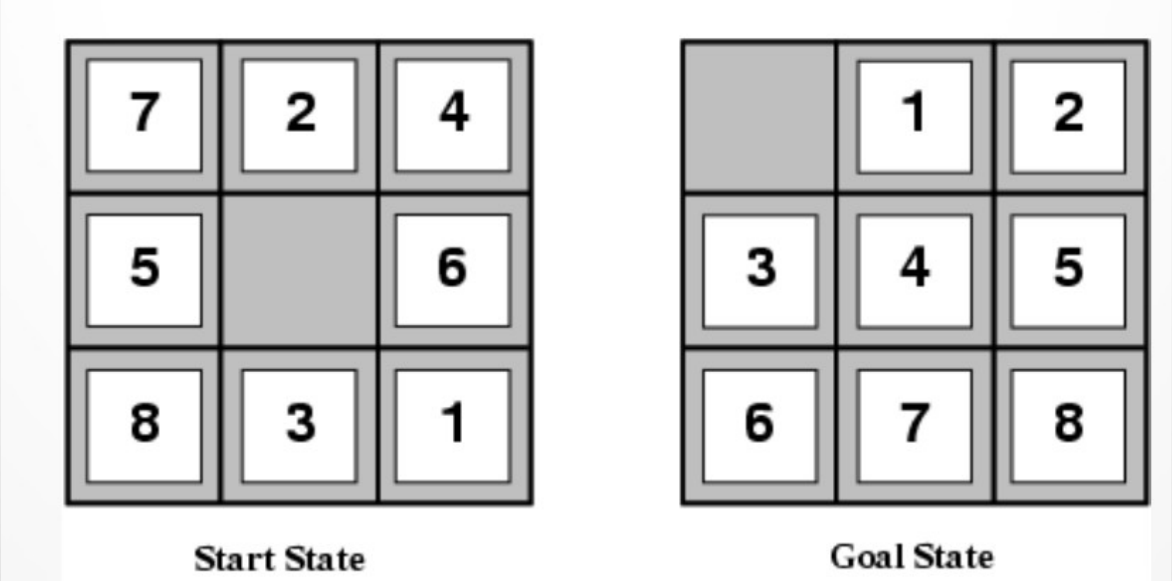
Buscas

- Exemplo 2:



Buscas

- Exemplo 3:
  - Quebra -cabeça de 8 peças
    - Tabuleiro 3x3 com 8 peças numeradas e um espaço vazio; Uma peça pode deslizar para o espaço



Buscas

- Formular objetivo: Dado um estado inicial, chegar ao estado final
- Formular problema:
- Estados: Qualquer configuração
- Operadores: Espaço vazio se desloca para esquerda, direita, cima ou baixo
- Achar solução – sequencia de passos para chegar a um determinado estado
- Custo – cada passo custa 1 e o custo do caminho é o numero de passos no caminho

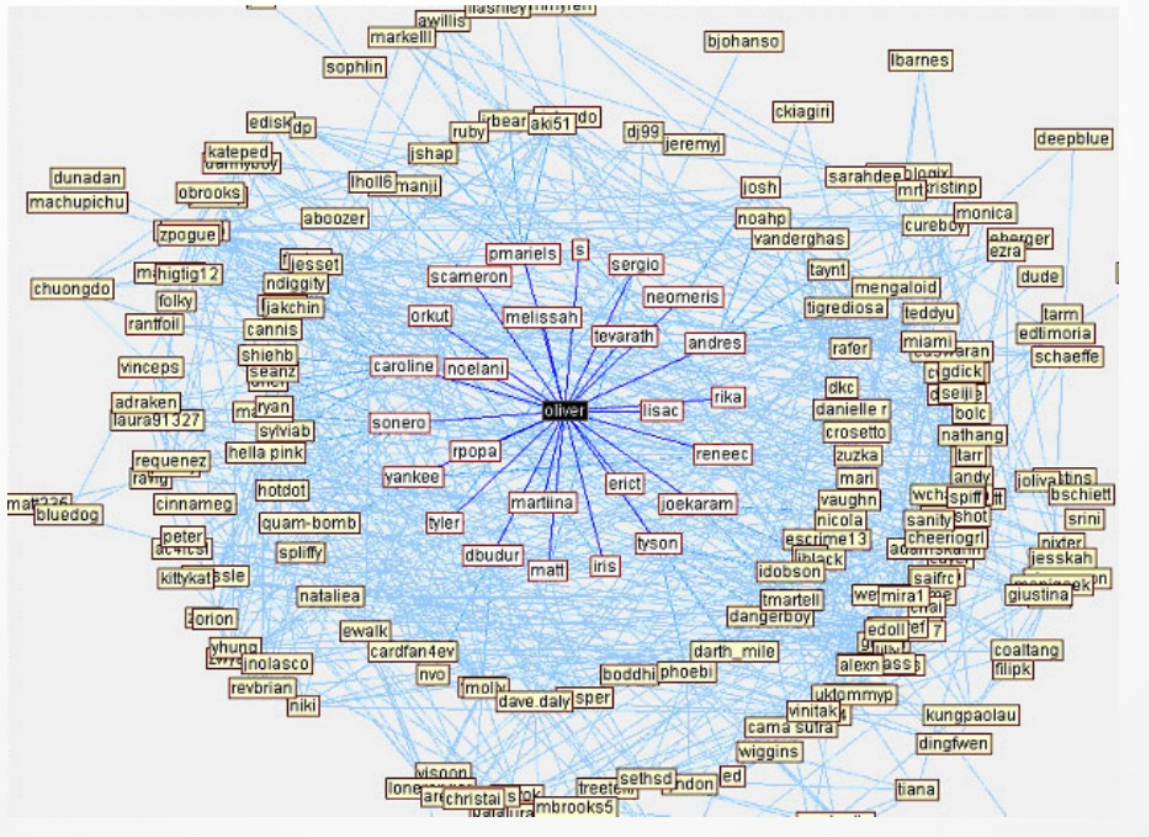
Buscas

Problema dos quebra-cabeças deslizantes

- NP-completo
  - Ainda não existem algoritmos determinísticos que encontram a solução em tempo polinomiais
    - 8 peças: 181440 estados possíveis
    - 15 peças: 1,3 trilhão de estados
    - 24 peças: 10^{25} estados

Buscas

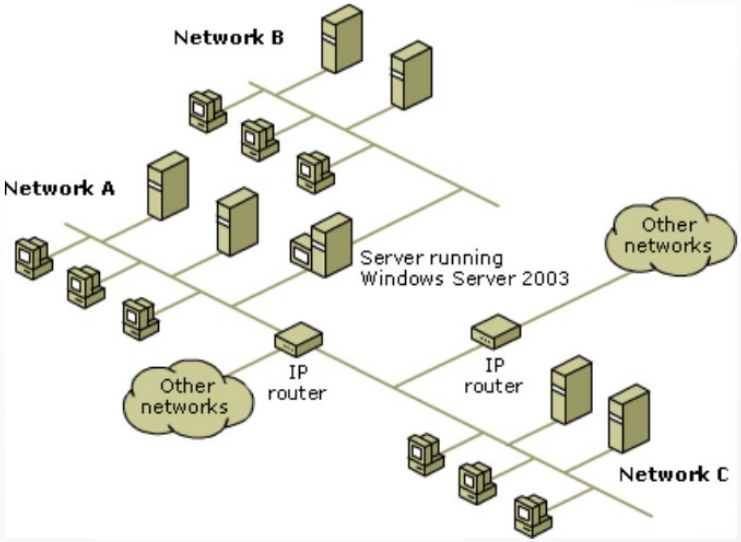
- Rede Social = Grafo



Buscas

Problema de roteamento

- Exemplos:
  - Roteamento de pacotes em rede de computadores
  - Planejamento de operações militares
  - Sistema de planejamento de viagens aéreas



Buscas

Navegação de robôs

- Generalização do problema de roteamento
- Lidar também com erros em sensores e controles

Buscas

Layout de VLSI

- Posicionamento de componentes e conexões em chip minimizando
  - Área
  - Retardos de circuitos
  - Capacitâncias de fuga
- E maximizando
  - Rendimento industrial

Buscas

- Sistema de controle:
  1. Inicializar a base de dados.
  2. Inicializar a condição de término.
  3. Até que a base de dados satisfaça a condição de término, faça:

  (É no selecionar das regras que está a sua “inteligência”, segundo os simbolistas)

     1. Selecione a regra aplicável existente.
     2. Aplique a regra.
  1. Fim.

Buscas

- Na execução dos algoritmos de busca, alguns caminhos são melhores do que outros.
- Cada caminho será definido pela ordem de escolha das regras.
- Para que o sistema de controle possa decidir pela melhor regra a cada instante, ele deve fazer uso das heurísticas.
- A inteligência do sistema está de fato nas heurísticas, que são conselhos sobre que regras aplicar em cada situação.

Buscas

- **Algoritmos de busca**
  - **não Informados**
    - usam somente definição do problema;
    - não fornece informações adicionais sobre o problema, exceto sua especificação;
  - **Informados**
    - usam conhecimento sobre o domínio para encontrar as etapas para a solução.
    - conhecimento na forma de heurísticas

Buscas

- **Busca Informada**
- Características
  - conhecimento domínio + problema;
  - função avaliação f(n);
  - conhecimento na forma de heurísticas;
  - Expande o nó com melhor avaliação;
  - Estratégia de busca depende da função;

Buscas

- **Busca Informada**
- Para verificar o custo da expansão da solução de um problema em uma arvore, implementando uma estratégia de busca informada, os **n** nós mais promissores são inseridos na função heurística **h(n).**
- Em seguida, a função retorna um número real não negativo, que é um custo de caminho aproximado calculado do nó n para o nó de destino.

Buscas

- **Busca Informada**
- Existe um consumo de tempo para que o algoritmo escolha a melhor heurística no momento e outro tempo para escolher a melhor regra, segundo essa heurística.
- O tempo total, soma dos dois anteriores, é chamado de “custo de seleção”:
- T1 = calcula a heurística -> Ri
- T2 = gerar uma nova base aplicando Ri
- CT = T1 + T2

Buscas

- **Busca Informada**
  - Busca Gulosa (greedy)
  - Busca A*

Buscas

- **Busca não Informada**
- Características

– busca cega

– utiliza somente informação contida no problema

– definidos pela ordem em que os nós são expandidos

Buscas

- **Busca não Informada**
  - necessário eliminar ciclos (organizar soluções em uma árvore)
  - detectar estados redundantes
  - complexidade mais baixa

Buscas

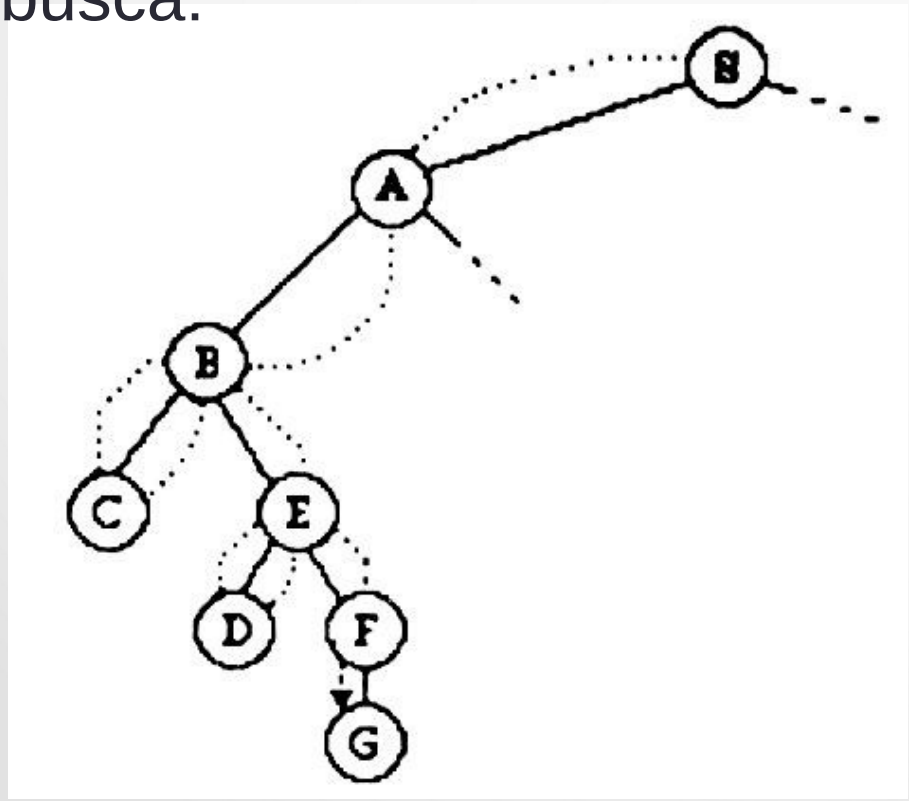
- **Busca não Informada**
  - Busca em Profundidade
  - Busca em Largura

Busca em Profundidade

- Quando procuramos apenas um caminho qualquer entre dois nós, uma boa idéia é escolher uma alternativa em cada nó e trabalhar para frente.
- Outras alternativas no mesmo nível são ignoradas, enquanto houver esperança de que se possa alcançar o destino seguindo o caminho corrente.

Busca em Profundidade

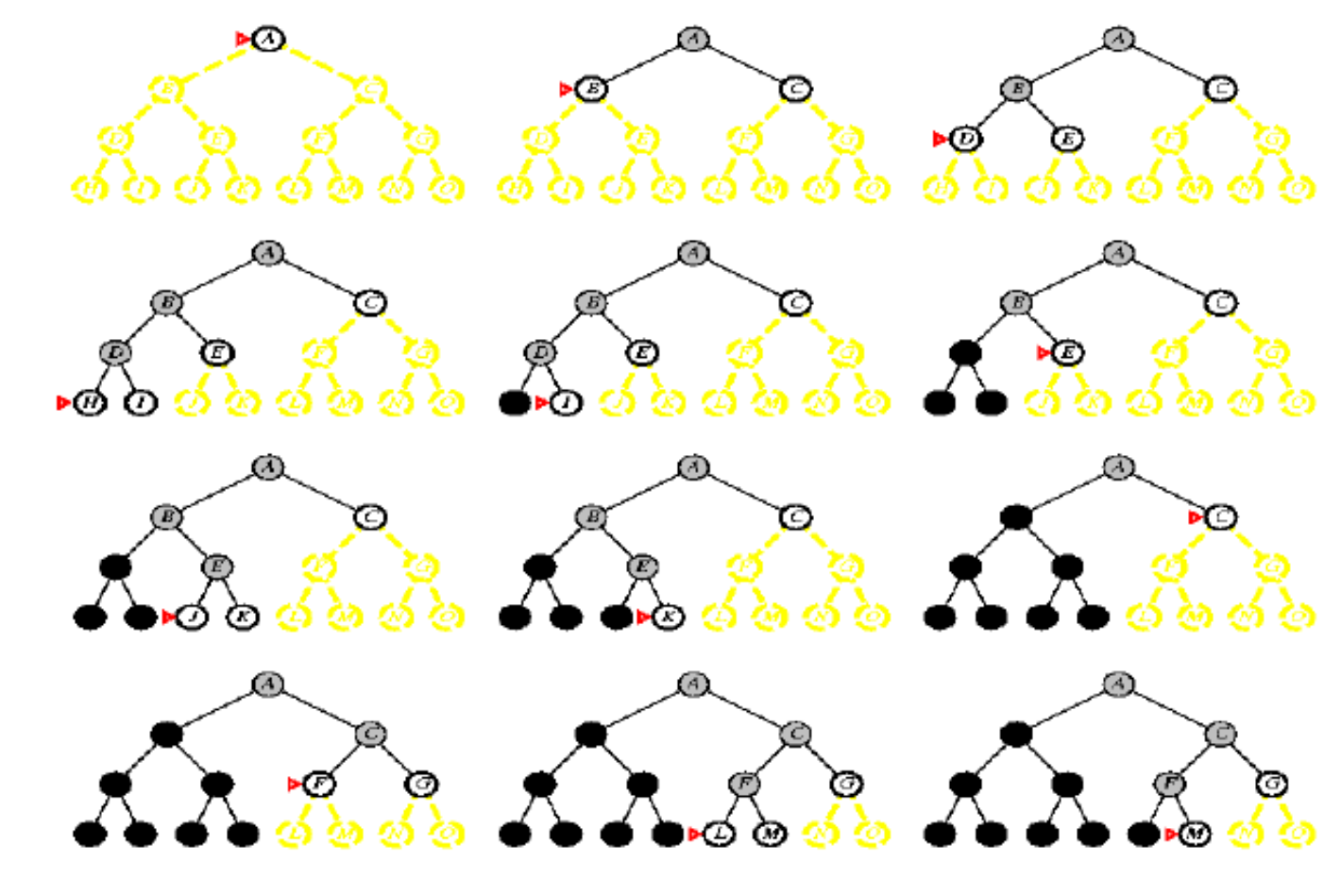
- Se trabalharmos sempre da direita para a esquerda, iremos criar o seguinte caminho na nossa árvore de busca:



Como o traçado até o nó C não alcança G, tivemos que retornar até B e continuar novamente daí.

Se esse caminho também não tivesse funcionado, teríamos que retornar até A (pois a outra alternativa para E seria B, que já havia sido visitado).

Busca em Profundidade

- 

Busca em Profundidade

- Algoritmo:
  - Forme uma Pilha de um elemento, contendo o nó raiz.
  - Até que a Pilha esteja vazia ou que o destino tenha sido alcançado, determine se o primeiro elemento da Pilha é o nó destino.
    - Se for, não faça nada.
    - Se o primeiro elemento não é o nó destino, remova o primeiro elemento da Pilha e adicione os filhos do primeiro elemento, se existentes, na frente da Pilha;
  - Se o nó destino tiver sido encontrado: sucesso. Senão: fracasso.

Busca em Profundidade

- Na busca em profundidade, cada nó é explorado até que se alcance um estado final, ou que não haja mais regras aplicáveis. (não é completa e não é ótima)
- Apenas no último caso um outro nó irmão desse será explorado.
- Nesse algoritmo, pontos de retorno são marcados e, caso se conclua que uma regra foi mal aplicada, retorna-se a um desses pontos e aplica-se outra regra. (só guarda nós do caminho atual)

Busca em Profundidade

- O mecanismo de retorno é conhecido como *backtraking*.
- Agindo desta forma, o algoritmo garante que, caso haja alguma solução, ela será encontrada, já que todos os caminhos são investigados.

Busca em Profundidade

- Na Busca em Profundidade, fazemos o backtracking nas seguintes situações:
  - Quando for gerada uma configuração já gerada na trajetória atual de busca
  - Quando a profundidade do backtracking for maior que um limite pré-estabelecido (piso).

Busca em Largura

- A busca em largura, ou amplitude, procura pelo nó destino entre todos os nós do mesmo nível antes de usar os filhos desses nós para prosseguir.
- Neste método, investigamos todos os operadores que podem ser aplicados ao nó, gerando todos os filhos.

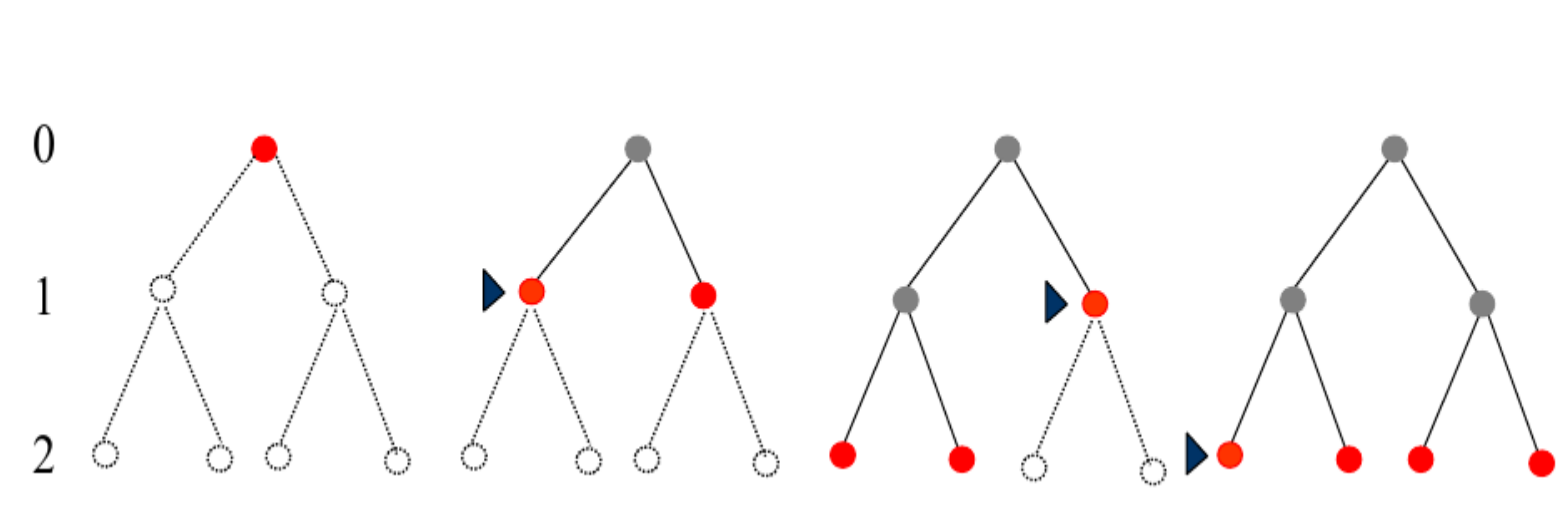
Busca em Largura

- Algoritmo:
  - Forme uma fila de um elemento, contendo o nó raiz.
  - Até que a fila esteja vazia ou que o destino tenha sido alcançado, determine se o primeiro elemento da fila é o nó destino.
    - Se for, não faça nada.
    - Se o primeiro elemento não é o nó destino, remova o primeiro elemento da fila e adicione os filhos do primeiro elemento, se existentes, no final da fila.
  - Se o nó destino tiver sido encontrado: sucesso. Senão: fracasso.

Busca em Largura

- Essa busca não é adequada no caso em que todos os caminhos que levam ao nó destino possuírem mais ou menos a mesma profundidade.(completa)
- Ela sempre encontra a solução, caso ela exista. Como expande um nível de cada vez, o método também encontra o caminho mais curto. (Ótima)
- Entretanto, a quantidade de possibilidades avaliadas é muito grande, o que torna o método computacionalmente intenso.(mantém todos os nós na memória)

Busca em Largura



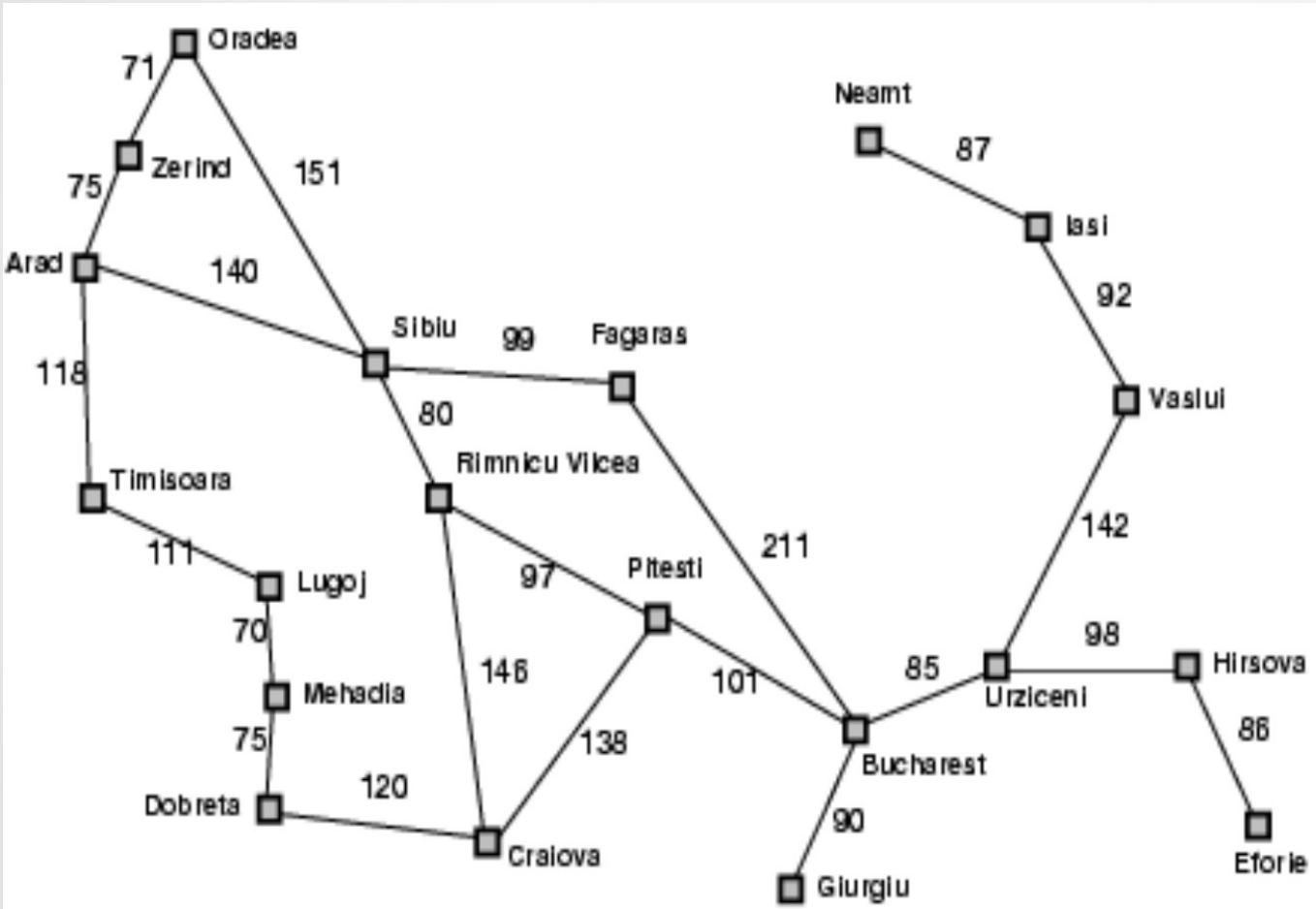
Busca Informada

- **Busca Informada**
  - Busca Gulosa (greedy)
  - Busca A*

Busca Informada

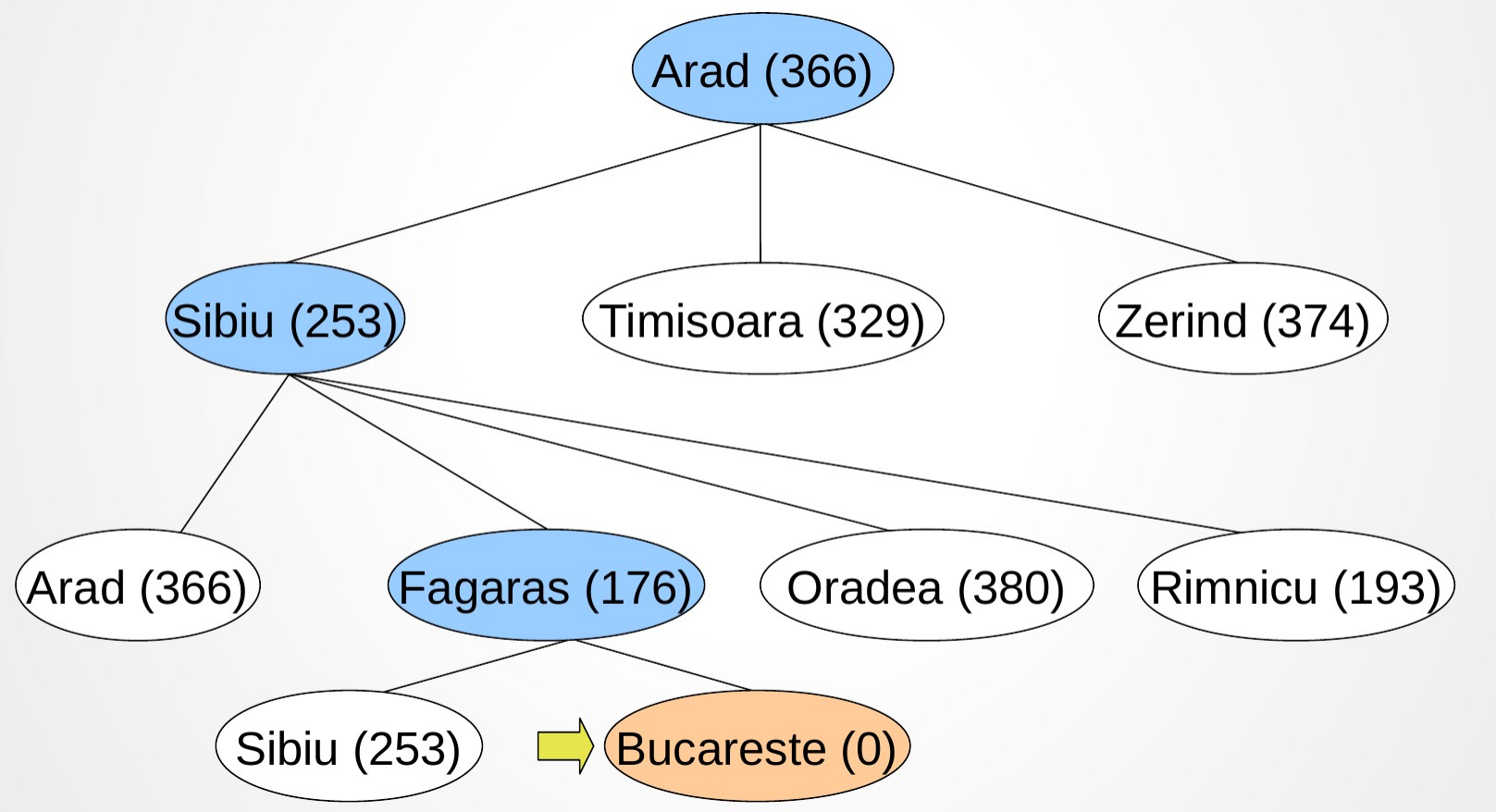
- Busca Gulosa
- Busca Best-First:
  - O nó é selecionado para expansão baseado em uma função de avaliação f(n)=h(n) (heurística) = estimativa de custo de n até o objetivo.
  - Busca gulosa pela melhor escolha expande o nó que parece mais próximo ao objetivo de acordo com a função heurĩstica;

Busca Informada



Straight–line distance to Bucharest	
Arad	366
Bucharest	0
Craiova	160
Dobreta	242
Eforie	161
Fagaras	176
Giurgiu	77
Hirsova	151
Iasi	226
Lugoj	244
Mehadia	241
Neamt	234
Oradea	380
Pitesti	100
Rimnicu Vilcea	193
Sibiu	253
Timisoara	329
Urziceni	80
Vaslui	199
Zerind	374

Busca Informada



Busca Informada

- Boas heurísticas devem ter a forma f(n) que calcule ou avalie com certa precisão o custo da melhor trajetória do nó inicial *s* ao final *t*, obrigada a passar por *n*.
- A função f(n) é a soma de duas estimativas:
  - g(n): o custo da trajetória de s a n.
  - h(n): o custo da trajetória de n a t.

- Então, o custo real será f(n) = g(n) + h(n)

Busca Informada

- Uma boa tentativa para g(n) é tomar, a cada instante, o custo da menor trajetória de s a n gerada até aquele instante.
- Já aproximar h(n) é mais complexo.

Busca Informada

Algoritmo

1. Criar uma árvore de busca contendo, inicialmente apenas um nó (nó raiz) com a base de dados S.
2. Colocar S em uma lista chamada ABERTOS.
3. Criar outra lista chamada FECHADOS, inicialmente vazia.
4. Se ABERTOS é vazia, parar com FALHA. Do contrário, retire o primeiro nó de ABERTOS e coloque em FECHADOS.
5. Se este nó satisfaz à condição de término, parar com sucesso. Do contrário, gerar os filhos do nó e coloca-los em ABERTOS.
6. Segundo uma heurística, ordene os nós em ABERTOS e volte para o passo 4.

Busca Informada

- Para evitar que a busca nunca pare, estabelecemos um piso. Essa busca é muito próxima de uma busca por *backtracking*.
- A chave está no procedimento de ordenação dos nós!
- Deste procedimento geral podemos gerar diferentes buscas em função de como os nós em ABERTOS são ordenados:
  - Se os nós mais recentemente gerados são os primeiros da lista, temos a Busca em Profundidade.
  - Se ordenarmos os nós da lista ABERTOS de forma que os mais antigos estejam primeiro, temos a Busca em Largura

Busca A*

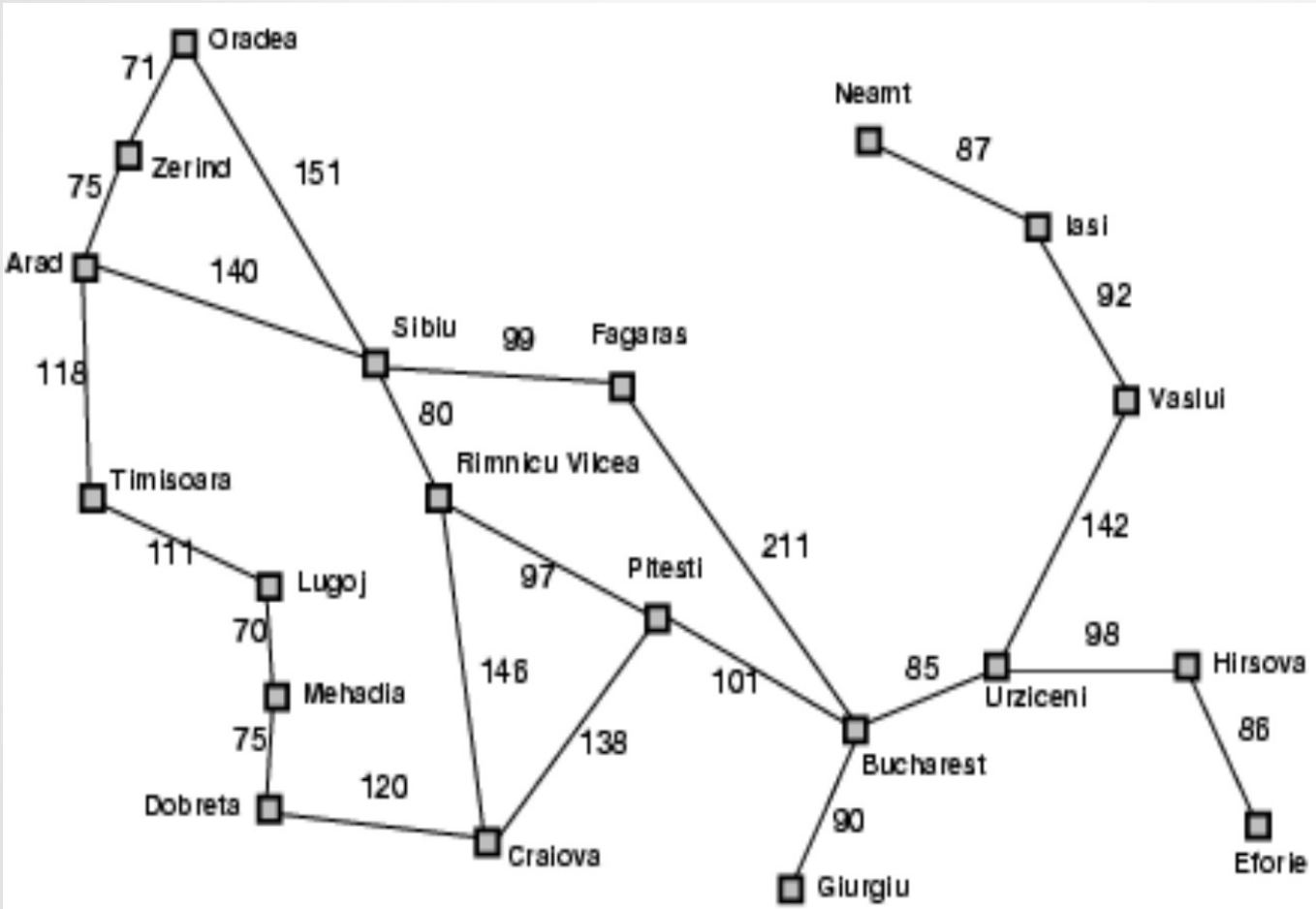
- Na Busca A*, a função de custo para avaliar a qualidade de um nó N possui a seguinte forma:
  - f(N) = g(N) + h(N) onde:
  - g(N) = custo do caminho até N
  - h(N) = estimativa do custo do caminho de N até um nó terminal
- A função h(N) é uma heurística sobre o menor caminho do nó N até o nó final e a qualidade da solução depende desta estimativa.
- Se soubermos como calcular h(N), ou seja, se tivermos uma boa estimativa do custo de caminho de N até nosso objetivo, podemos orientar a busca por esta estimativa.

Busca A*

Algoritmo

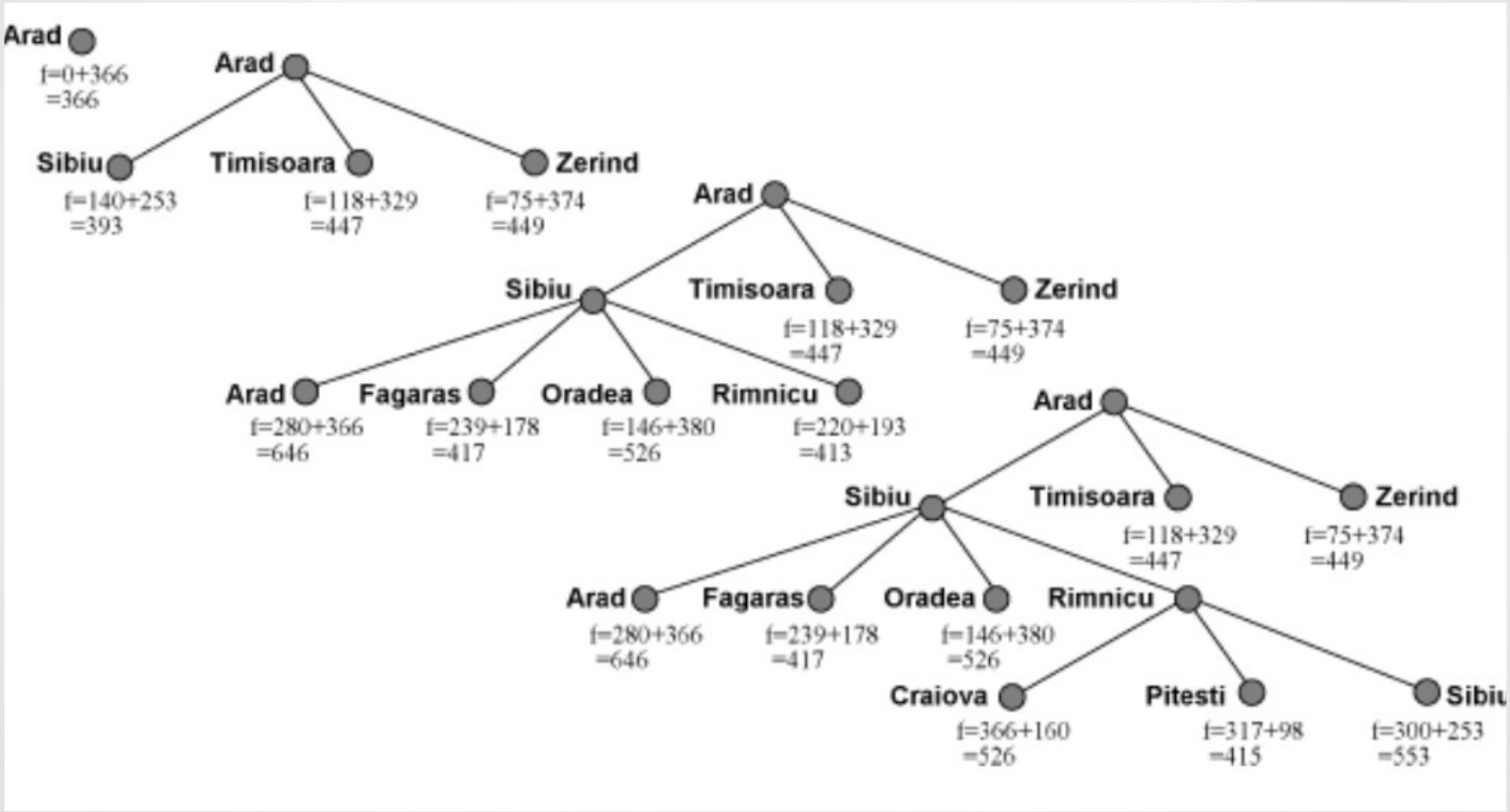
1. Expandir todas as possibilidades de um nó que esteja aberto;
2. Para cada nó N gerado
   - Calcular f(N)
   - Se N já está na árvore (aberto), eliminar aquele com maior f(N)
3. Escolher o nó que tenha o menor f(N) na árvore;
4. Se não foi achado o destino, retornar ao passo 1 ;

Busca A* - Exemplo



Straight–line distance to Bucharest	
Arad	366
Bucharest	0
Craiova	160
Dobreta	242
Eforie	161
Fagaras	176
Giurgiu	77
Hirsova	151
Iasi	226
Lugoj	244
Mehadia	241
Neamt	234
Oradea	380
Pitesti	100
Rimnicu Vilcea	193
Sibiu	253
Timisoara	329
Urziceni	80
Vaslui	199
Zerind	374

Busca A* - Exemplo



Busca Ordenada

- Referëncias

- Prof. Eduardo R. Hruschka -

http://wiki.icmc.usp.br/images/3/32/Aula3_BuscaInformada.pdf

- Russell, S., Norvig, P., Artificial Intelligence – A Modern Approach, Second Edition, Prentice Hall
- Huei Diana Lee -

http://www.dainf.ct.utfpr.edu.br/~fabro/IA_I/busca/IA_Estrategias_Busca_Inf.pdf

- Prof Fernando Gomide -

http://www.dca.fee.unicamp.br/~gomide/courses/EA072/transp/EA072EstruturasEstrategiasBusca4.pdf

Prof. Fabio Augusto Faria - http://www.somos.unifesp.br/professores/view/3353