

# Inteligência Artificial

Estratégias de busca heurística

# Sumário

- Introdução
- Busca com informação

# Introdução

# Introdução

- Um problema pode ser definido por 5 componentes
  - Estado inicial
  - Ações
  - Modelo de transição
  - Teste de objetivo
  - Custo do caminho

# Introdução

- Uma solução é uma sequência de ações que levam do estado inicial para o estado objetivo
- Uma solução ótima é uma solução com o menor custo de caminho

# Algoritmo Geral de Busca em árvore

```
function TREE-SEARCH( problem, fringe) returns a solution, or failure
  fringe  $\leftarrow$  INSERT(MAKE-NODE(INITIAL-STATE[problem]), fringe)
  loop do
    if fringe is empty then return failure
    node  $\leftarrow$  REMOVE-FRONT(fringe)
    if GOAL-TEST[problem](STATE[node]) then return SOLUTION(node)
    fringe  $\leftarrow$  INSERTALL(EXPAND(node, problem), fringe)
```

---

```
function EXPAND( node, problem) returns a set of nodes
  successors  $\leftarrow$  the empty set
  for each action, result in SUCCESSOR-FN[problem](STATE[node]) do
    s  $\leftarrow$  a new NODE
    PARENT-NODE[s]  $\leftarrow$  node; ACTION[s]  $\leftarrow$  action; STATE[s]  $\leftarrow$  result
    PATH-COST[s]  $\leftarrow$  PATH-COST[node] + STEP-COST(node, action, s)
    DEPTH[s]  $\leftarrow$  DEPTH[node] + 1
    add s to successors
  return successors
```

**Busca com informação**

# Busca com informação

## *Definição*

- Utiliza conhecimento específico sobre o problema para encontrar soluções de forma mais eficiente do que a busca cega
- Conhecimento específico além da definição do problema



# Busca com informação

## *Definição*

- Abordagem geral: busca pela melhor escolha.
  - Utiliza uma função de avaliação para cada nó
  - Expande o nó que tem a função de avaliação mais baixa
  - Dependendo da função de avaliação, a estratégia de busca muda

# Busca com informação

*Busca pela melhor escolha*

- Idéia: usar uma função de avaliação  $f(n)$  para cada nó
  - Estimativa do quanto aquele nó é desejável
  - Expandir nó mais desejável que ainda não foi expandido
- Implementação:
  - Ordenar nós na borda em ordem decrescente de acordo com a função de avaliação

# Busca com informação

*Casos especiais*

- Busca gulosa pela melhor escolha
- Busca  $A^*$

# Busca Gulosa

- Função de avaliação
  - $f(n) = h(n)$  (heurística) = estimativa do custo de  $n$  até o objetivo
  - ex.,  $h_{DLR}(n)$  = distância em linha reta de  $n$  até Bucareste.
- Busca gulosa pela melhor escolha expande o nó que parece mais próximo ao objetivo de acordo com a função heurística

# Busca Gulosa

- Não é ótima, pois segue o melhor passo considerando somente o estado atual
  - Pode haver um caminho melhor seguindo algumas opções piores em alguns pontos da árvore de busca
- Minimizar  $h(n)$  é suscetível a falsos inícios
  - Ex. Ir de Iasi a Fagaras
    - Heurística sugerirá ir a Neamt, que é um beco sem saída
    - Se repetições não forem detectadas a busca entrará em loop

# Busca A\*

- Idéia: evitar expandir caminhos que já são caros
- Função de avaliação  $f(n) = g(n) + h(n)$ 
  - $g(n)$  = custo até o momento para alcançar  $n$
  - $h(n)$  = custo estimado de  $n$  até o objetivo
  - $f(n)$  = custo total estimado do caminho através de  $n$  até o objetivo

# Busca A\*

## *Heurística Admissível*

- Uma heurística  $h(n)$  é admissível se para cada nó  $n$ ,  $h(n) \leq h^*(n)$ , onde  $h^*(n)$  é o custo verdadeiro de alcançar o estado objetivo a partir de  $n$
- Uma heurística admissível nunca superestima o custo de alcançar o objetivo, isto é, ela é otimista.
- Exemplo:  $h_{DLR}(n)$ 
  - distância em linha reta nunca é maior que distância pela estrada
- Teorema: Se  $h(n)$  é admissível, A\* usando algoritmo BUSCA-EM-ARVORE é ótima

# Busca A\*

## *Heurística Admissível*

- Para o quebra-cabeça de 8 peças:
  - $h1(n)$  = número de peças fora da posição
  - $h2(n)$  = distância “Manhattan” total (para cada peça calcular a distância em “quadras” até a sua posição)

7	2	4
5		6
8	3	1

	1	2
3	4	5
6	7	8



# Heurísticas Admissíveis

*Como criar*

1. A solução de uma simplificação de um problema (problema relaxado) é uma heurística para o problema original
  - Admissível: a solução do problema relaxado não vai superestimar a do problema original
  - É consistente para o problema original se for consistente para o relaxado

# Heurísticas Admissíveis

*Como criar*

- Exemplo: Quebra-cabeça de 8
  - h1 daria a solução ótima para um problema “relaxado” em que as peças pudessem se deslocar para qualquer lugar
  - h2 daria a solução ótima para um problema “relaxado” em que as peças pudessem se mover um quadrado por vez em qualquer direção

# Heurísticas Admissíveis

*Como criar*

2. Usar o custo da solução de um subproblema do problema original

*	2	4
*		*
*	3	1

Start State

	1	2
3	4	*
*	*	*

Goal State

Calcular o custo da solução exata sem se preocupar com os \*  
Limite inferior do custo do problema completo

# Heurísticas Admissíveis

*Como criar*

## 3. Banco de dados de padrões:

- Armazenar o custo exato das soluções de muitos subproblemas
- Para um determinado estado procurar o subproblema referentes àquele estado
- Exemplo: todas as configurações das 4 peças na figura anterior