UNISINOS

PIP/CA - ***Programa Interdisciplinar de Pós-Graduação***
Mestrado em Computação Aplicada da UNISINOS

2000/1 - 2o. Trimestre - AULA 01 / FSO

INTELIGÊNCIA ARTIFICIAL & SISTEMAS INTELIGENTES

• **Professores Responsáveis:**

Parte I - Profa. Dr. Renata Vieira
Web: ***http://www.inf.unisinos.br/~renata/iam.html***

Parte II - Prof. Dr. Fernando Osório
E-Mail: ***osorio@exatas.unisinos.br***
Web: ***http://www.inf.unisinos.br/~osorio/ia.html***

TEMAS DE ESTUDO

Resolução de problemas / Competindo com os seres humanos:

- Busca em espaços de estados:

> Busca Cega - Busca em Largura, Busca em Profundidade, British Museum Search

> Busca Heurística - Best First, Branch and Bound, A*

- IA em Jogos: Mini-Max, Alfa-Beta, Autômatos - Redes de Petri e Cadeias de Markov

Sistemas Inteligentes:

- Analogia: Case Based Reasoning (CBR), métricas de semelhança (Nearest Neighbors)

- Inferência: Sistemas Especialistas - Fatos, Regras e Mecanismo de Inferência

- KBS / RBS (Knowledge / Rule Based Systems): Forward Chaining,
Backward Chaining, Algoritmo Rete, ... ***Expert System Shells***

Representação de conhecimentos:

- Conhecimento incerto e impreciso: Fuzzy rules, Certainty Factor, Bayesian Networks

- Conhecimento: validação, explanação, meta-conhecimentos, aquisição

F. OSÓRIO - UNISINOS 2000

RESOLUÇÃO DE PROLEMAS E DE JOGOS

"A maioria dos problemas interessantes do ponto de vista da I.A. não dispõem de soluções algorítmicas, ou tem soluções algorítmicas conhecidas mas sua complexidade as torna impraticáveis"

Busca da Solução <==> Complexidade

Inteligência ?
Análise de Algoritmos ==> Problema da explosão combinatória
Pesquisa Operacional ==> Otimização

*** Problemas: Quebra-Cabeças e Jogos**

- **Ser humano é capaz de achar soluções**
- **Uma boa resposta é associada as "pessoas inteligentes"**
- ***Toy problems*: do simples ao mais complexo (jogo da velha => xadrez)**
- **Problemas auto-contidos: conhecimento sobre o problema é completo**

*** Busca da soluções: uso de técnicas de Inteligência Artificial**

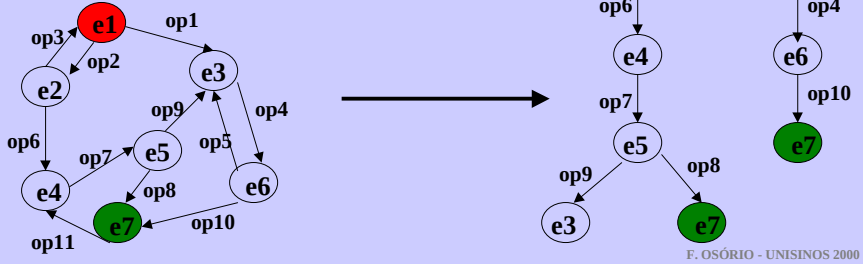
F. OSÓRIO - UNISINOS 2000

BUSCA EM ESPAÇO DE ESTADOS

Busca:

- **Achar a solução através de uma pesquisa nos possíveis estados do sistema (possíveis estados do sistema = espaço de estados)**

- Definição de um problema:
> Estados iniciais (1 ou mais)
> Estados Finais (0 ou mais soluções)
> Operadores que levam de um estado a outro
> Construção de uma "árvore de busca"



F. OSÓRIO - UNISINOS 2000

BUSCA EM ESPAÇO DE ESTADOS

Tipos de Busca - *Quanto a estratégia*:

1. BUSCA CEGA

- **1.1. Busca em Largura**
- **1.2. Busca em Profundidade**
- **1.3. Busca Exaustiva**

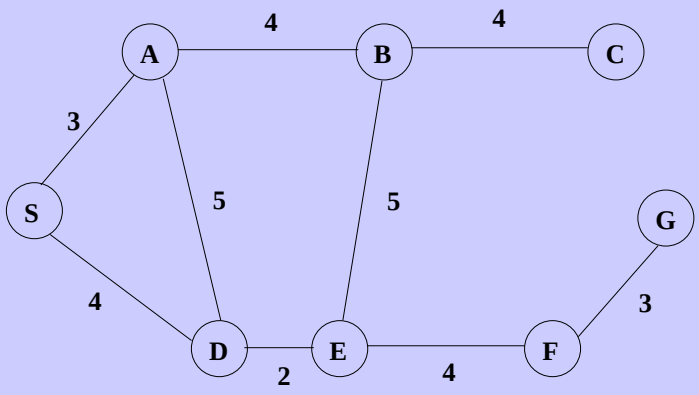
2. BUSCA HEURÍSTICA: A*

Tipos de Busca - *Quanto ao problema*:

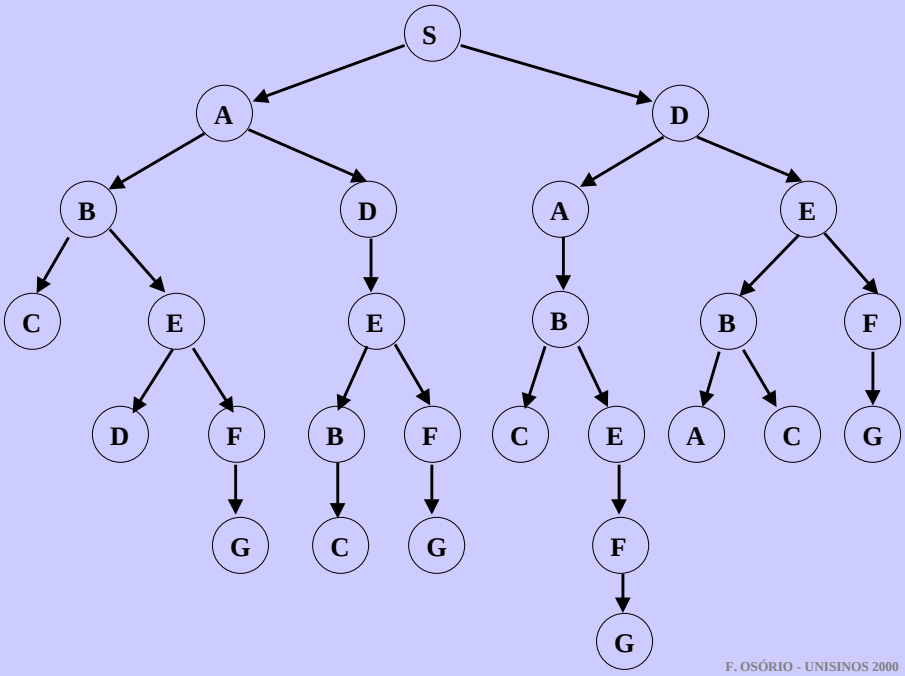
1. **Mecanismo de busca livre: Problemas em geral (quebra-cabeça)**

2. **Mecanismos de busca condicionada: Jogos com mais de 1 jogador**
   Presa as jogadas do oponente

BUSCA EM ESPAÇO DE ESTADOS

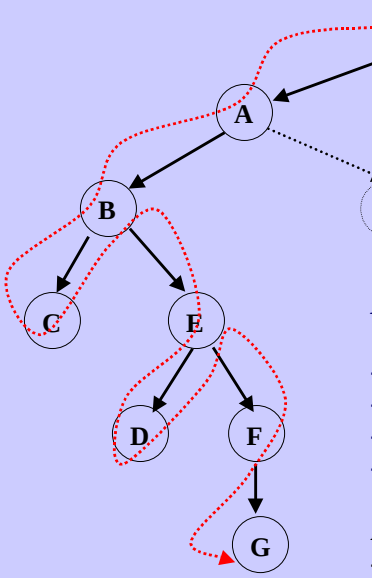


BUSCA EM ESPAÇO DE ESTADOS



F. OSÓRIO - UNISINOS 2000

BUSCA EM ESPAÇO DE ESTADOS



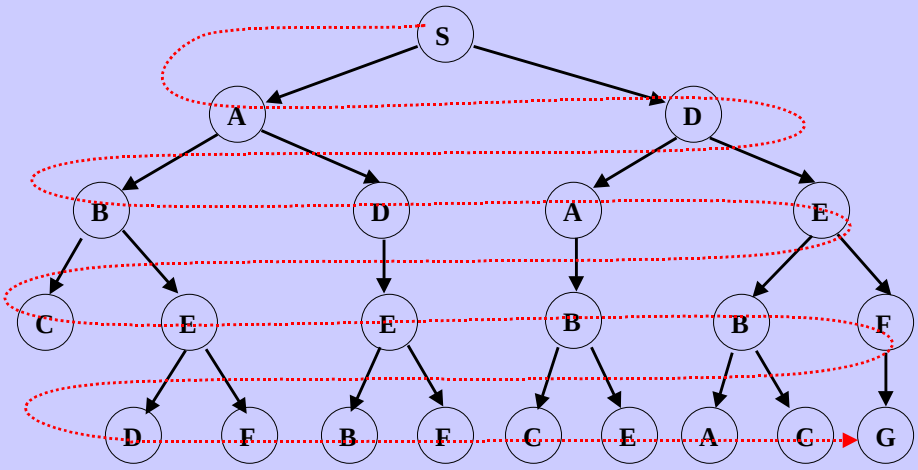
Depth-First Search:

- Busca em Profundidade
- Algoritmo "Leve"
- "Não ótimo" / Pode não chegar na solução
- Podemos inserir limite de profundidade

Algoritmo de Busca:
- Open Set / Closed Set = Pendente / Já visitou
- Select = Usa uma pilha, retira do topo
- Critérios de parada: Sucesso, Profundidade

F. OSÓRIO - UNISINOS 2000

BUSCA EM ESPAÇO DE ESTADOS



Breadth-First Search:

- Busca em Largura
- Algoritmo "Pesado"
- "Ótimo" / Deve chegar na solução (não se sabe quando)

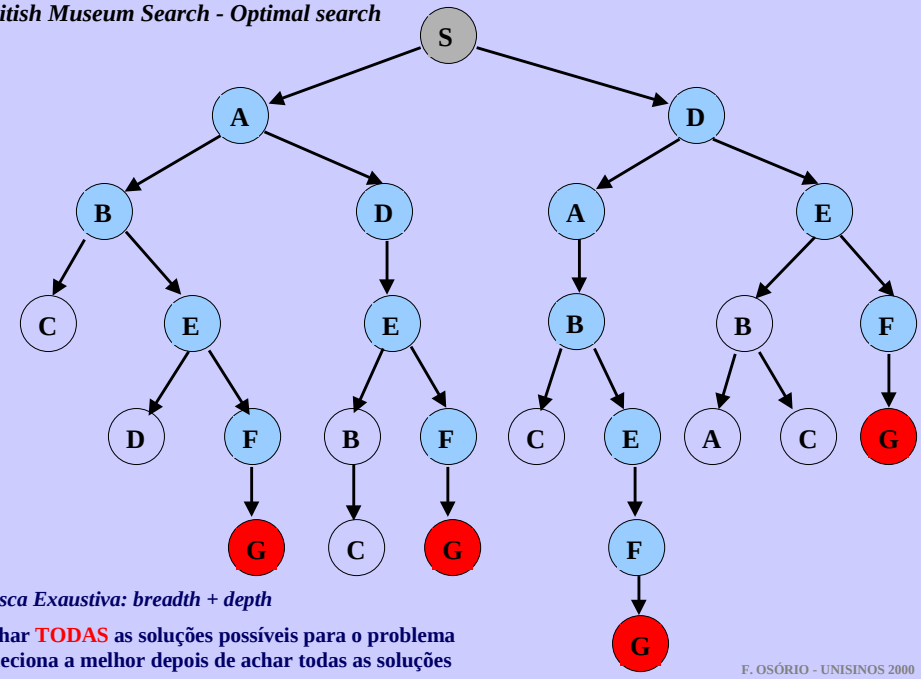
Algoritmo de Busca:

- Open Set / Closed Set
- Select = Usa uma fila, insere no final
- Critérios de parada: Sucesso

F. OSÓRIO - UNISINOS 2000

BUSCA EM ESPAÇO DE ESTADOS:

British Museum Search - Optimal search



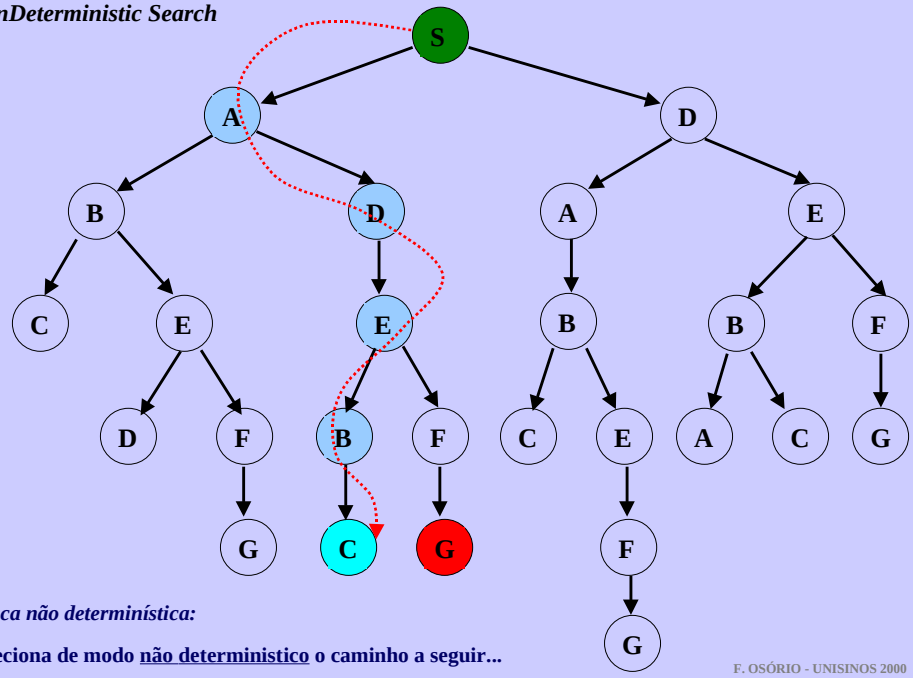
Busca Exaustiva: breadth + depth

Achar TODAS as soluções possíveis para o problema
Seleciona a melhor depois de achar todas as soluções

F. OSÓRIO - UNISINOS 2000

BUSCA EM ESPAÇO DE ESTADOS:

NonDeterministic Search



Busca não determinística:

Seleciona de modo não determinístico o caminho a seguir...

F. OSÓRIO - UNISINOS 2000

BUSCA EM ESPAÇO DE ESTADOS

Exemplos de Problemas tipo "quebra-cabeça":

1. **Caixeiro viajante**
2. **Torre de Hanoi**
3. **Labirinto**
4. **Puzzle 8 peças**
5. **Missionário e os canibais**
6. **Homem, lobo, carneiro e a alface**
7. **Problema dos baldes**
8. **Quadrados mágicos**
9. **Resta 1**
10. **Problema do depósito: alocação de espaço**

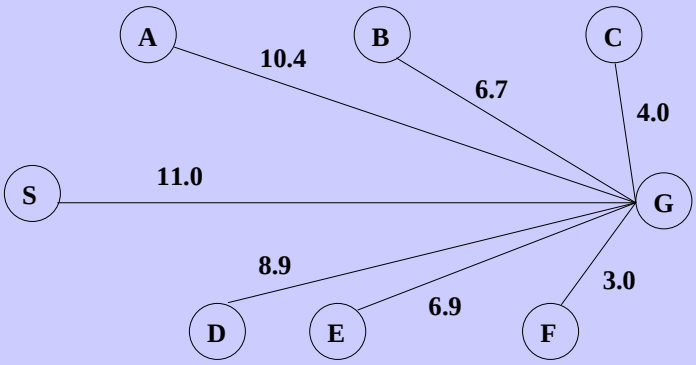
Alguns destes problemas tendem a se tornar intratáveis dependendo do "tamanho" do espaço de estados a ser analisado (exemplo: 1, 2, 3, 8, 9, 10).

Qual a solução ? **OTIMIZAR = USAR UMA HEURÍSTICA**

F. OSÓRIO - UNISINOS 2000

BUSCA EM ESPAÇO DE ESTADOS: Heurística

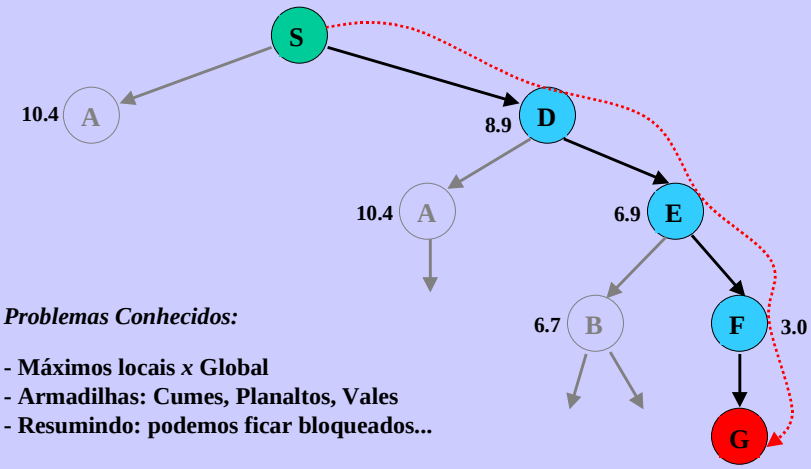
- Conhecer uma dica, uma informação que permita auxiliar na busca da solução do problema



Os valores indicam a distância em linha reta entre os pontos dados

BUSCA EM ESPAÇO DE ESTADOS: Heurística

Hill Climbing Search
- Conhecemos uma informação que permite avaliar os caminhos
- Heurística: depth-first + minimizar o "custo" (distância absoluta)



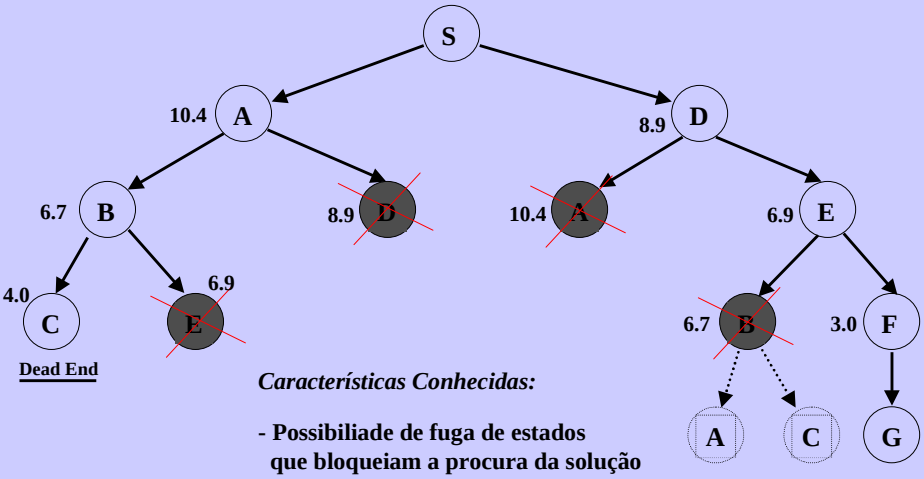
Problemas Conhecidos:

- Máximos locais *x* Global
- Armadilhas: Cumes, Planaltos, Vales
- Resumindo: podemos ficar bloqueados...

BUSCA EM ESPAÇO DE ESTADOS: Heurística

Beam Search

- Heurística: Breadth-first + seleciona apenas"N" caminhos por nível (procura selecionar os "n" melhores - tipo *Hill Climbing)*



Características Conhecidas:

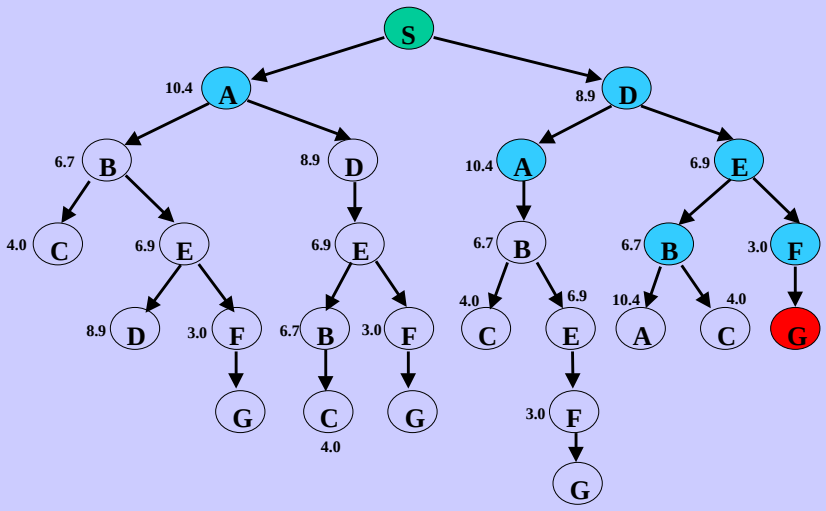
- **Possibiliade de fuga de estados que bloqueiam a procura da solução**
- **Menos pesado que o *Breadth-First***

BUSCA EM ESPAÇO DE ESTADOS: Heurística

Best-First Search

- Minimiza o custo, mas considera todos os que estão no "Open Set"

- No caso específico abaixo: equivale ao *Hill Climbing* (nunca se arrepende)

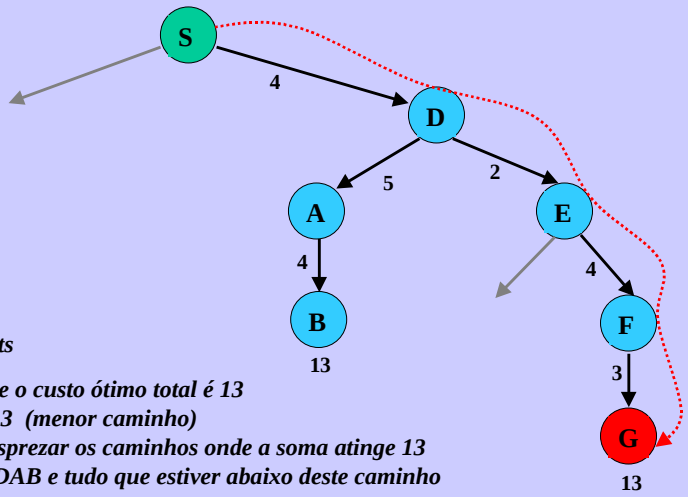


BUSCA EM ESPAÇO DE ESTADOS: Heurística - Optimal search

Branch-and-Bound Search

- Conhecemos uma informação que permite avaliar os caminhos e o custo total

- Heurística: avança e volta caso se "arrependa" do caminho adotado



**** Dicas - Hints***

- Sabemos que o custo ótimo total é 13
SDEFG = 13 (menor caminho)

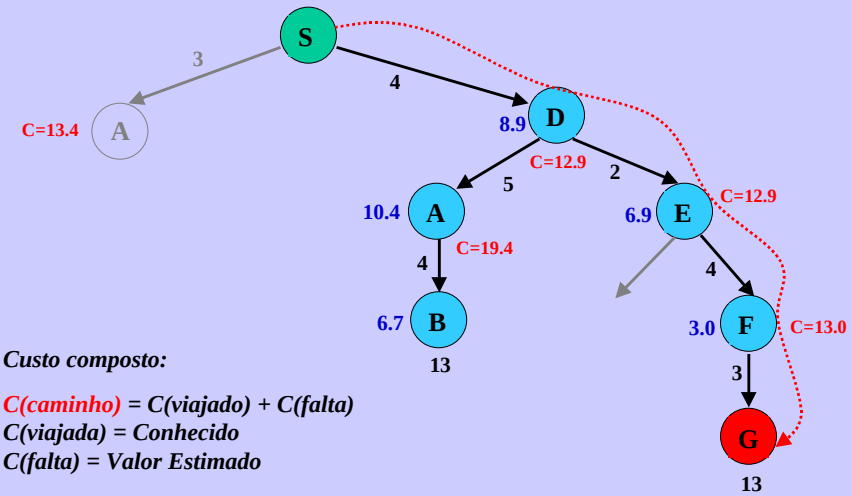
- Podemos desprezar os caminhos onde a soma atinge 13
Exemplo: SDAB e tudo que estiver abaixo deste caminho

BUSCA EM ESPAÇO DE ESTADOS: Heurística - Optimal search

Branch-and-Bound Search com estimativa

- Conhecemos uma informação que permite avaliar os caminhos e o custo total

- Heurística: avança e volta caso se "arrependa" do caminho adotado



**** Custo composto:***

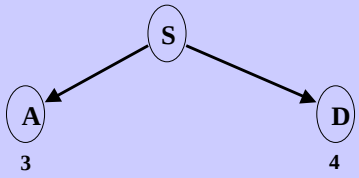
C(caminho) = C(viajado) + C(falta)
C(viajada) = Conhecido
C(falta) = Valor Estimado

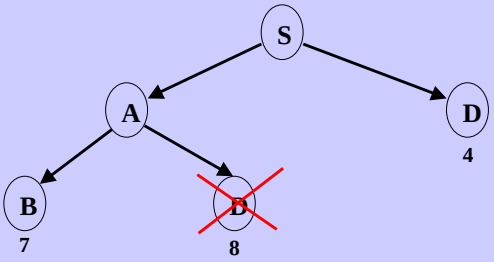
BUSCA EM ESPAÇO DE ESTADOS: Heurística - Optimal search

A* Search => Select: Sempre busca o melhor da lista Open

- Heurística: Custo (Caminho) = Custo (Caminho Percorrido)
+ Custo (Caminho Restante)

- Simplificação: eliminar caminhos redundantes
Exemplo => SD... SA...
SD... SAD... (D é novamente usado, caminho maior) SAB...





A* Search:
Branch-and-Bound + Simplificação

BUSCA EM ESPAÇO DE ESTADOS: Resumindo...

1. Busca Livre em espaço de estados - Problemas / Quebra-cabeças

1.1. Busca Cega
1.1.1. Busca em Profundidade (Depth-Search)
1.1.2. Busca em Largura (Breadth-Search)
1.1.3. Busca não determinística (Nondeterministic Search)
1.1.4. Busca exaustiva (British Museum Search - ótima)

1.2. Busca Heurística
1.2.1. Hill Climbing Search
1.2.2. Beam Search
1.2.3. Best-Fisrt Search
1.2.4 Optimal Search
1.2.4.1. Branch-and-Bound Search
1.2.4.2. A* Search

2. Busca Condicionada em espaços de estados - Jogos / Adversário externo

BUSCA EM ESPAÇO DE ESTADOS: Busca condicionada em Jogos

Jogos:

- **Caminhos possíveis dependem das "reações" do adversário**
- **Exemplo de jogos tratados pela I.A.:**

  Jogo da Velha
  Gamão
  Damas
  Xadrez
  Go
  Othello

- **Jogos também são uma procura do caminho em um espaço de estados, onde desejamos seguir o caminho que leva a vitória**

- **Heurísticas: Avaliar as jogadas (boa, ruim) e a situação/evolução do jogo**

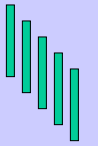
Algorítmos mais usados:

- ***Minimax***
- ***Alpha-Beta***

BUSCA EM ESPAÇO DE ESTADOS: Busca condicionada em Jogos

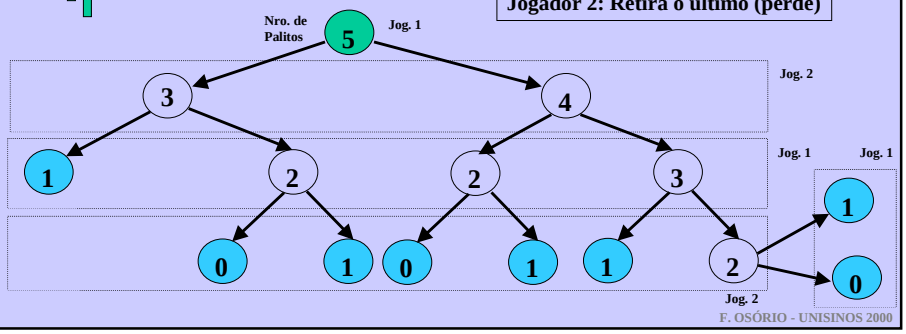
MiniMax Procedure
- **Alternância de jogadores**
- **Construção de uma árvore com camadas alternadas (Mini e Max)**

*** Exemplo:**
Jogo dos 5 palitos - Objetivo: pegar 1 ou 2 palitos e não ser o último a jogar

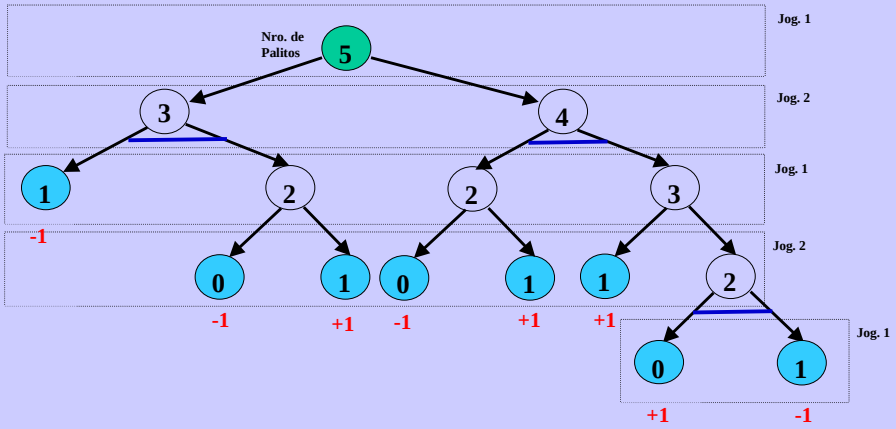


Cenário 1:
Jogador 1: Retira 2 palitos
Jogador 2: Retira 2 palitos
Jogador 1: Retira o último (perde)

Cenário 2:
Jogador 1: Retira 1 palito
Jogador 2: Retira 2 palitos
Jogador 1: Retira 1 palito
Jogador 2: Retira o último (perde)

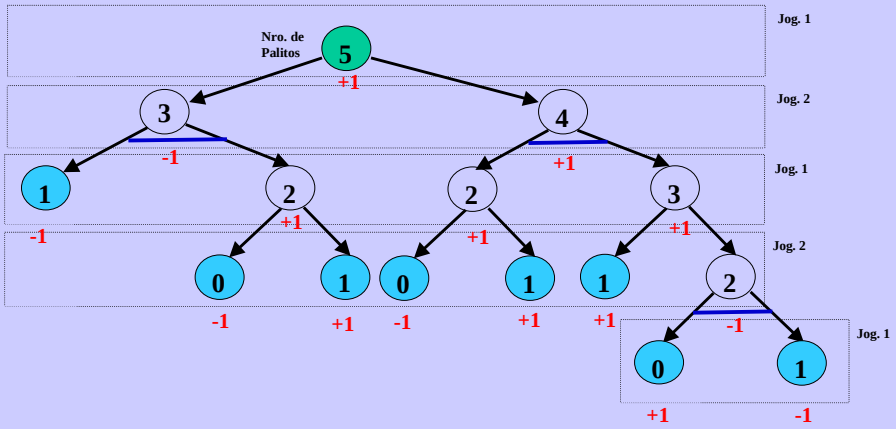


BUSCA EM ESPAÇO DE ESTADOS: Busca em Jogos - MiniMax



Etapas do algoritmo:
- **Pontuação nos nodos terminais da árvore de busca (usualmente: +1, 0, -1)**
- **Classificar os nodos como do tipo Max (jog. 1 - Livre escolha "ou" =)**
  ou Mini (jog. 2 - Adversário escolhe =)
- **Propagar os mini (menor dos dois filhos) e os max (maior dos dois filhos)**

BUSCA EM ESPAÇO DE ESTADOS: Busca em Jogos - MiniMax



Propagação dos Mini e Max...

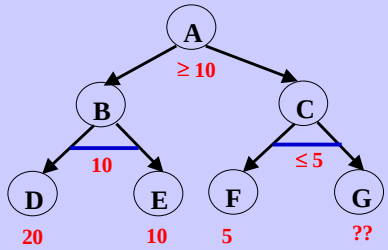
Para obter o melhor caminho, seguir a melhor pontuação!
(Em alguns casos podemos também limitar a profundidade da árvore)

=> Agora você pode fazer o mesmo para o Jogo da Velha!!

BUSCA EM ESPAÇO DE ESTADOS: Busca condicionada em Jogos

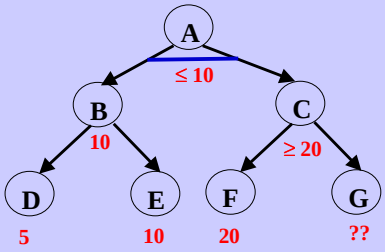
Alpha-Beta Procedure

- Limitar a procura tirando proveito das relações existentes entre as sub-árvores
- Procedimentos: Corte Alpha e Corte Beta



Corte Alpha:

Sabendo-se que A recebe o maior entre B e C
e que no lado de C existe um valor igual a 5
(logo um valor menor ou igual a 5 será selecionado em C => pois este é um nodo tipo Mini),
então não precisamos examinar a sub-arvore G.



Corte Beta:

Sabendo-se que o nodo A é o menor entre B e C, podemos desprezar a sub-árvore G pois esta é certamente maior ou igual a 20, e vamos guardar o menor valor entre B e C

Jogos Inteligentes:

Técnicas usadas...

- **Máquinas de estados**
- **Rule Based Systems / Sense & Act**
- **Redes de Petri**
- **Cadeias de Markov**
- **HFSM - Hierarquical Finite State Machines**

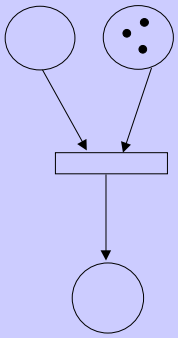
Integração de I.A. com outras áreas de pesquisa:
(referente ao tema estudado nesta aula)

I.A. e Pesquisa Operacional
I.A. e Análise de Algoritmos
I.A. e Computação Gráfica (jogos)

Tendências:

- Desafios... Xadrez, Go
- Jogos inteligentes: AI Wars (rules), FramSticks (artificial life), ...

Redes de Petri: Uma rápida introdução...



Estado:
Pré-Condições

Transição: Condições

Estado:
Pós-Condições

• → **Marcas: são deslocadas do estado prévio para o estado posterior**

Linguagens baseadas em Redes de Petri:

GrafCET - Comando-Etapa-Transição

Exemplo:

Se Encheu_Tanque E Atingiu_Limite
Então Abrir_Válvula_Escape

Se Aberta_Válvula_Escape E Tanque_Vazio
Então Fechar_Válvula_Escape