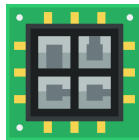


# AI aplicada a jogos adversariais

Afonso Fernandes, Gonalo Paredes, Ricardo Pereira



[dcc]



16 de Abril de 2019

# Conteúdo

1 Introdução

2 MiniMax

3 Alfa-Beta

4 MCTS

<https://afonsonf.github.io/ia-enemath-web/>

# Conteúdo

1 Introdução

2 MiniMax

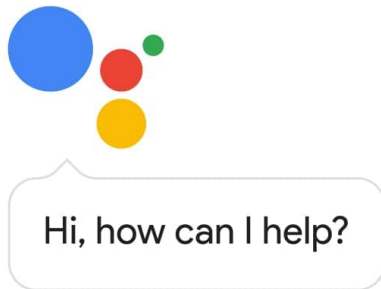
3 Alfa-Beta

4 MCTS



# Introdução - AI



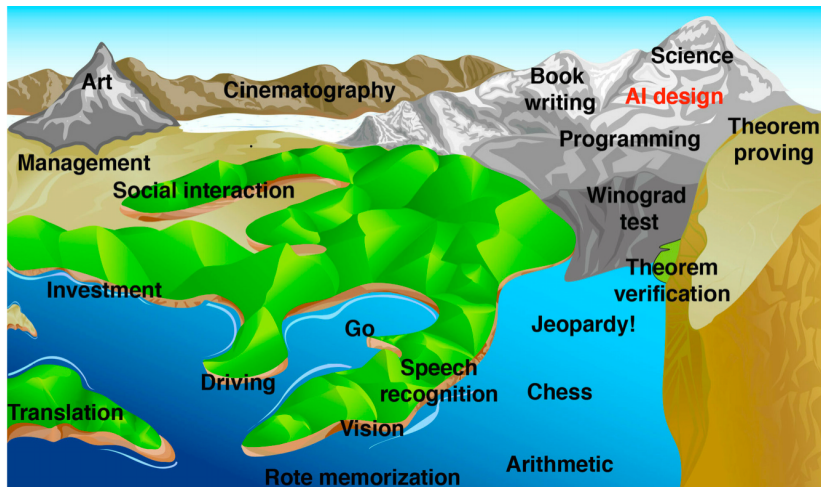


```
if(cond1)
{
    // ...
}
else if(cond2)
{
    // ...
}
else if(cond3)
{
    // ...
}
else if(cond4)
{
    // ...
}
else if(cond5)
{
    // ...
}
```



- Inteligência das máquinas
- Estuda agentes inteligentes
- Definição algo paradoxal
- Raciocínio Automático, Representação de Conhecimento, Processamento de Linguagem Natural, Planeamento, Machine Learning

# Introdução - AI



Landscape of Human Intelligence - Hans Moravec

# Introdução - AI



# Introdução - Jogo

- Estado inicial (posição + jogador)

- Estado inicial (posição + jogador)
- Função para gerar sucessores

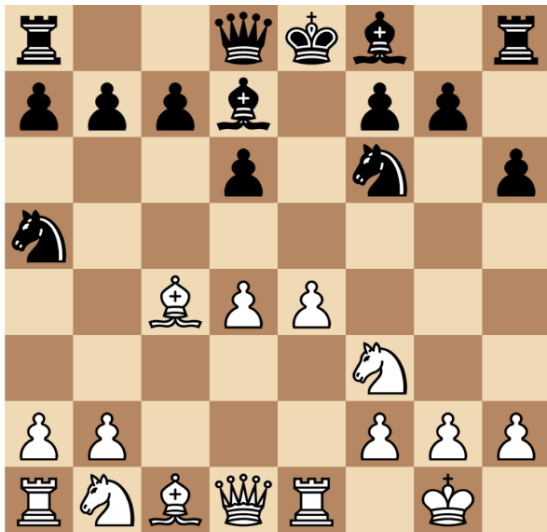
- Estado inicial (posição + jogador)
- Função para gerar sucessores
- Possível reconhecer estados finais

- Estado inicial (posição + jogador)
- Função para gerar sucessores
- Possível reconhecer estados finais
- Sequencial



- Estado inicial (posição + jogador)
- Função para gerar sucessores
- Possível reconhecer estados finais
- Sequencial
- Informação perfeita

# Introdução - Heurística



→ 73

# Conteúdo

1 Introdução

2 MiniMax

3 Alfa-Beta

4 MCTS

# Minimax

- 1 Método mais simples de decisão em inteligência artificial;

- 1 Método mais simples de decisão em inteligência artificial;
- 2 Baseia-se em fazer a escolha que minimiza a perda no pior caso possível;

- 1 Método mais simples de decisão em inteligência artificial;
- 2 Baseia-se em fazer a escolha que minimiza a perda no pior caso possível;
- 3 Originalmente desenvolvido para jogos de 2 jogadores de soma nula;

- 1 Método mais simples de decisão em inteligência artificial;
- 2 Baseia-se em fazer a escolha que minimiza a perda no pior caso possível;
- 3 Originalmente desenvolvido para jogos de 2 jogadores de soma nula;
- 4 Tem versões mais complexas, mas vamos apenas explorar a mais simples.



# Minimax

- 1 Explorar todas as jogadas possíveis;

- 1 Explorar todas as jogadas possíveis;
- 2 Escolher que minimiza a perda no pior caso possível, ou seja, escolher a que dá o menor valor possível ao adversário.

- 1 Explorar todas as jogadas possíveis;
- 2 Escolher que minimiza a perda no pior caso possível, ou seja, escolher a que dá o menor valor possível ao adversário.
- 3 No caso de uma jogada terminar o jogo atribuir um valor  $C$  ou  $-C$  consoante quem ganha;

- 1 Explorar todas as jogadas possíveis;
- 2 Escolher que minimiza a perda no pior caso possível, ou seja, escolher a que dá o menor valor possível ao adversário.
- 3 No caso de uma jogada terminar o jogo atribuir um valor  $C$  ou  $-C$  consoante quem ganha;
- 4 Para evitar cálculos que demorem anos, a partir de certa profundidade usa uma heurística para atribuir valor.

# Minimax - Pseudo-Código

```
function minimax(node, depth, maximizingPlayer) is
  if depth = 0 or node is a terminal node then
    return the heuristic value of node
  if maximizingPlayer then
    value :=  $-\infty$ 
    for each child of node do
      value := max(value, minimax(child, depth - 1, FALSE))
    return value
  else (* minimizing player *)
    value :=  $+\infty$ 
    for each child of node do
      value := min(value, minimax(child, depth - 1, TRUE))
    return value
```

# Minimax - Exemplo

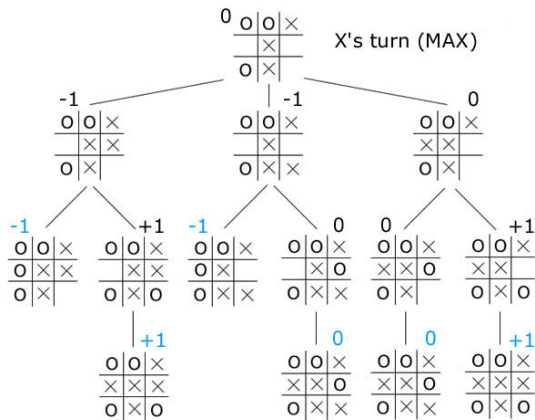


Figura: Exemplo de uma árvore de aplicação do Minimax no jogo do galo.

# Minimax - Problemas



- 1 Baseia-se em explorar árvores de possibilidades que normalmente crescem exponencialmente;

- 1 Baseia-se em explorar árvores de possibilidades que normalmente crescem exponencialmente;
- 2 Para valores pequenos de profundidade máxima joga fortemente de acordo com a heurística escolhida;
- 3 Para valores maiores demora muito a determinar a próxima jogada;

- 1 Baseia-se em explorar árvores de possibilidades que normalmente crescem exponencialmente;
- 2 Para valores pequenos de profundidade máxima joga fortemente de acordo com a heurística escolhida;
- 3 Para valores maiores demora muito a determinar a próxima jogada;
- 4 É um método muito próximo da força bruta, testando as possibilidades todas.

## Implementação

# Conteúdo

1 Introdução

2 MiniMax

3 Alfa-Beta

4 MCTS

# Alpha-beta pruning

# Alpha-beta pruning

- Baseia-se no minimax

# Alpha-beta pruning

- Baseia-se no minimax
- Melhora-o cortando nós que não é necessário explorar



# Alpha-beta pruning

- Baseia-se no minimax
- Melhora-o cortando nós que não é necessário explorar
- O ganho em tempo depende de vários fatores, mas geralmente é melhor que o minimax

# Alpha-beta - Pseudo-Código

```
function alphabeta(node, depth,  $\alpha$ ,  $\beta$ , maximizingPlayer) is
  if depth = 0 or node is a terminal node then
    return the heuristic value of node
  if maximizingPlayer then
    value :=  $-\infty$ 
    for each child of node do
      value := max(value, alphabeta(child, depth - 1,  $\alpha$ ,  $\beta$ , FALSE))
       $\alpha$  := max( $\alpha$ , value)
      if  $\alpha \geq \beta$  then
        break (*  $\beta$  cut-off *)
    return value
  else
    value :=  $+\infty$ 
    for each child of node do
      value := min(value, alphabeta(child, depth - 1,  $\alpha$ ,  $\beta$ , TRUE))
       $\beta$  := min( $\beta$ , value)
      if  $\alpha \geq \beta$  then
        break (*  $\alpha$  cut-off *)
    return value
```

# Alpha-beta - Exemplo

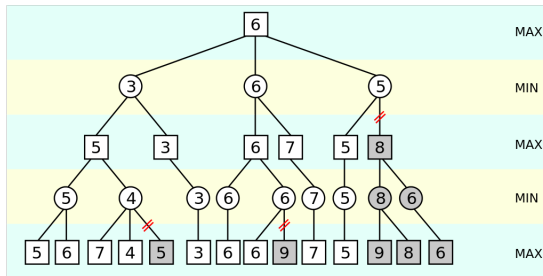


Figura: Exemplo de uma árvore de aplicação de *Alpha-beta pruning*.

# Implementação

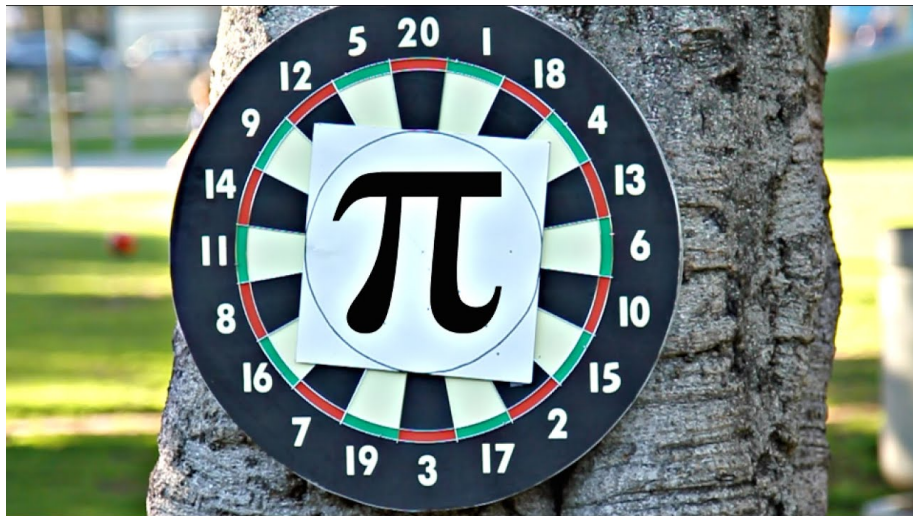
# Conteúdo

1 Introdução

2 MiniMax

3 Alfa-Beta

4 MCTS

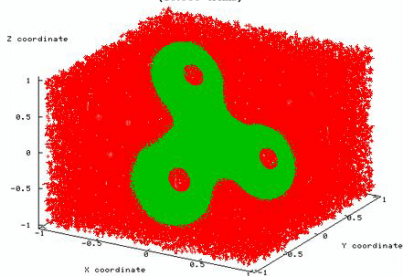


# MCTS - Métodos de Monte Carlo

1985: I bet they'll have flying cars  
in the future

2017:

Monte Carlo Integration of a Fidget Spinner  
(10,000 trials)



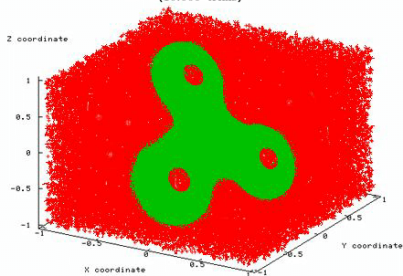
Integração

# MCTS - Métodos de Monte Carlo

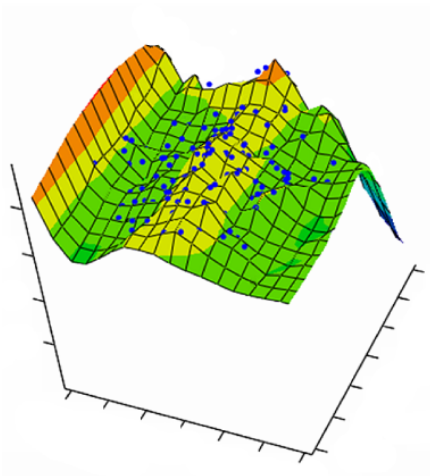
1985: I bet they'll have flying cars  
in the future

2017:

Monte Carlo Integration of a Fidget Spinner  
(10,000 trials)



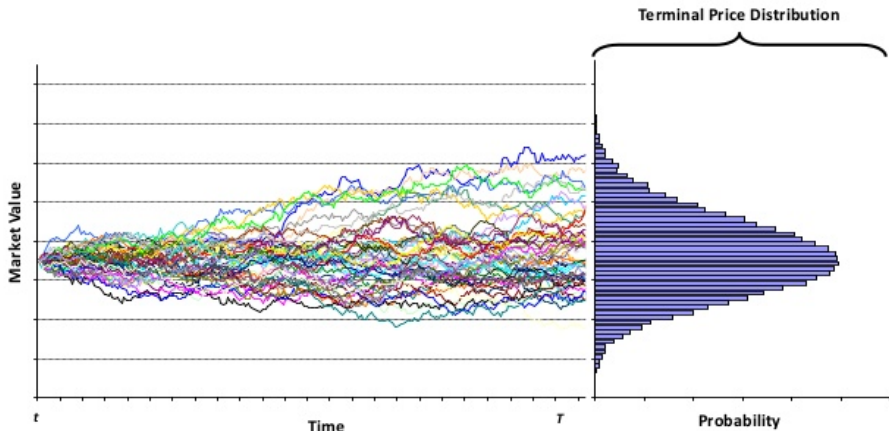
Integração



Otimização



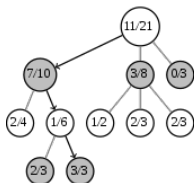
# MCTS - Métodos de Monte Carlo



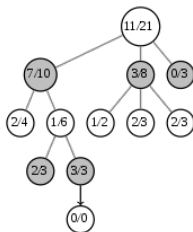
Mercado

# MCTS - Overview

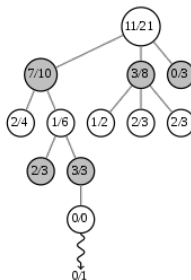
Selection



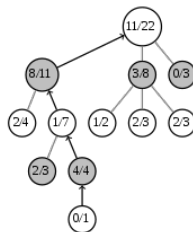
Expansion



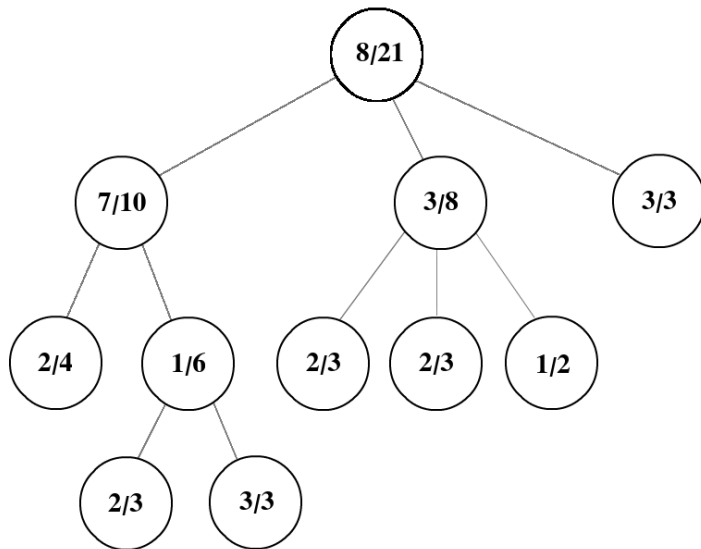
Simulation



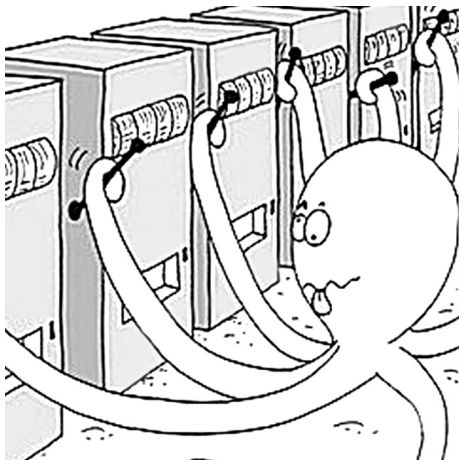
Backpropagation



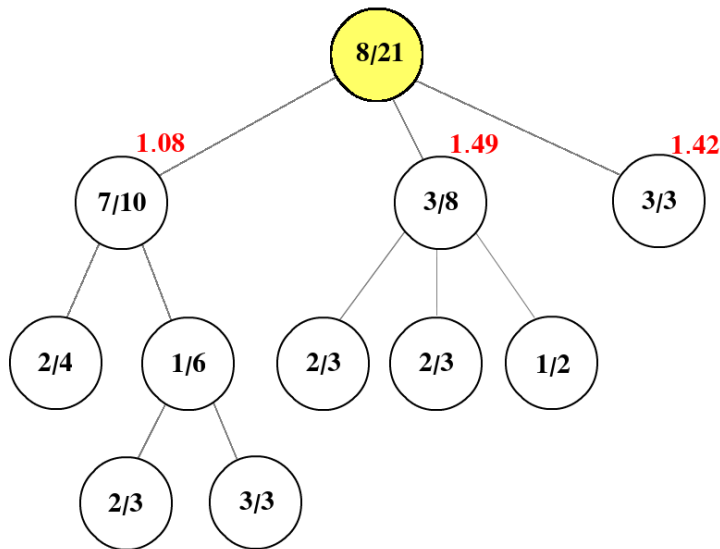
# MCTS - Estado Atual

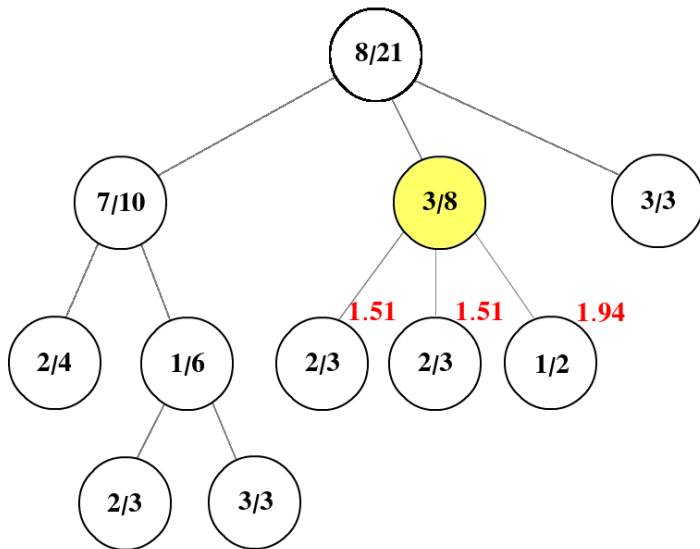


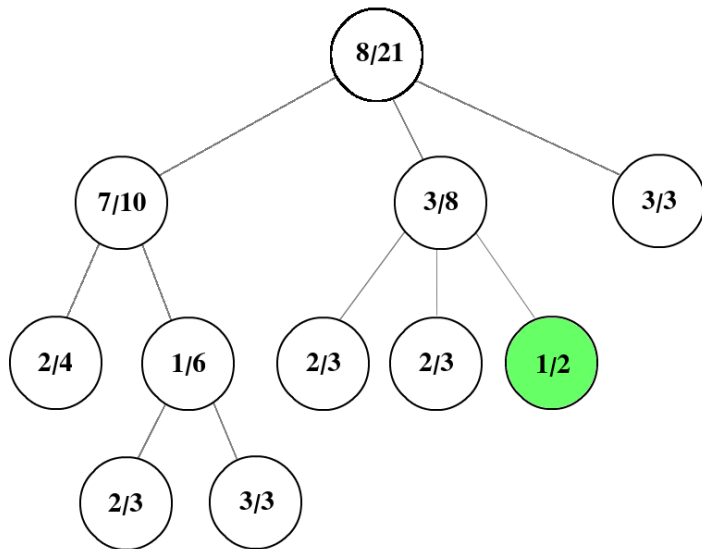
# MCTS - Multi Armed Bandit



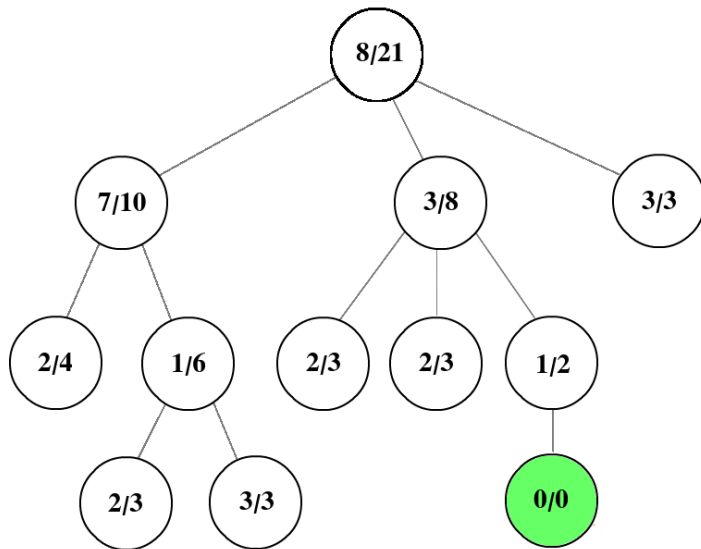
$$\frac{w}{n} + c\sqrt{\frac{\ln N}{n}}$$



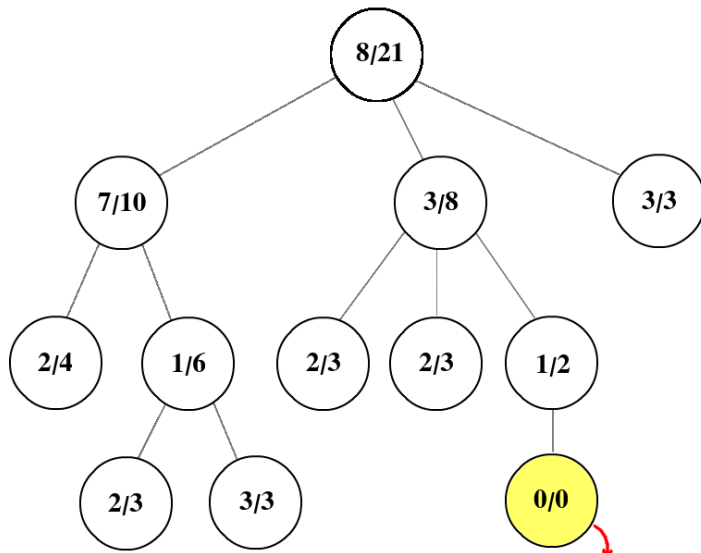


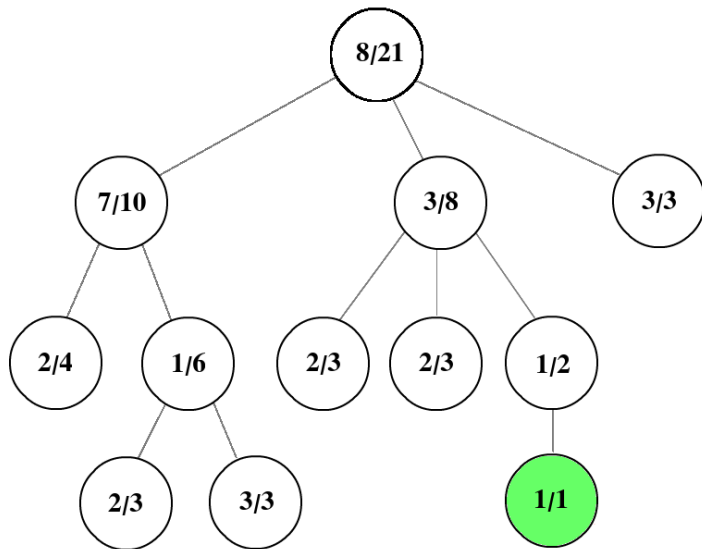


# MCTS - Expansão

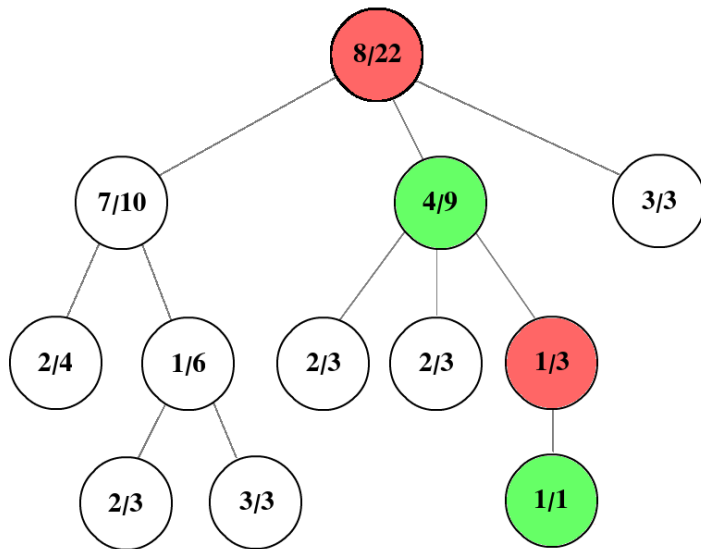


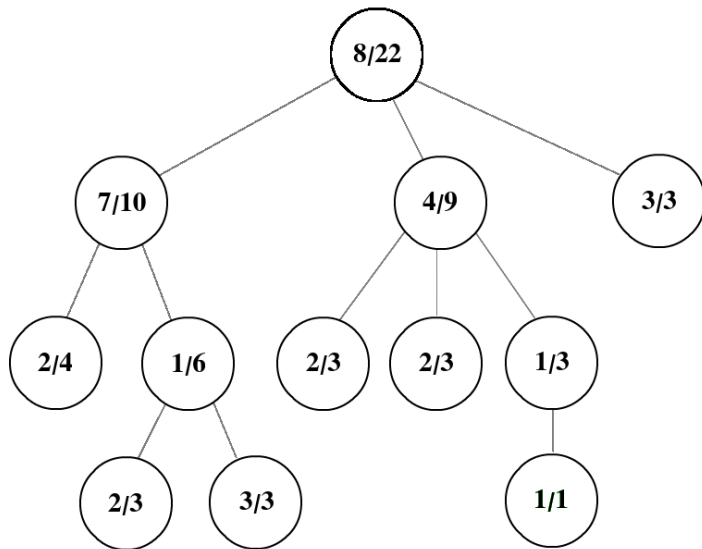






# MCTS - Backpropagation





# MCTS - Vantagens

- Não precisa de uma heurística decente

- Não precisa de uma heurística decente
- Crescimento assimétrico

- Não precisa de uma heurística decente
- Crescimento assimétrico
- Resposta "on demand"



# Implementação



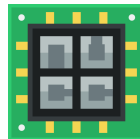
<https://afonsonf.github.io/ia-enemath-web/>

# AI aplicada a jogos adversariais

Afonso Fernandes, Gonalo Paredes, Ricardo Pereira



[dcc]



16 de Abril de 2019