Introdução à Teoria dos Jogos

Laboratório de Programação Competitiva - 2020

Pedro Henrique Paiola

Teoria dos Jogos

- A teoria dos jogos é uma teoria matemática criada para se modelar fenômenos que podem ser observados quando dois ou mais “agentes de decisão” interagem entre si.
- É utilizada para se estudar assuntos tais como eleições, leilões, balança de poder, evolução genética, etc.
- Não confundir:
  - Teoria Econômica dos Jogos: com motivações predominantemente econômicas, procurando estabelecer métodos para se maximizar o ganho.
  - Teoria Combinatória dos Jogos: que se concentra nos aspectos combinatórios de jogos de mesa e não permite “elementos imprevisíveis”.

Teoria dos Jogos

- Em problemas de Programação Competitiva, os problemas que envolvem teoria dos jogos normalmente contém dois jogadores adversários, que jogam de forma ótima, e precisamos determinar qual jogador será vencedor em um certo cenário.

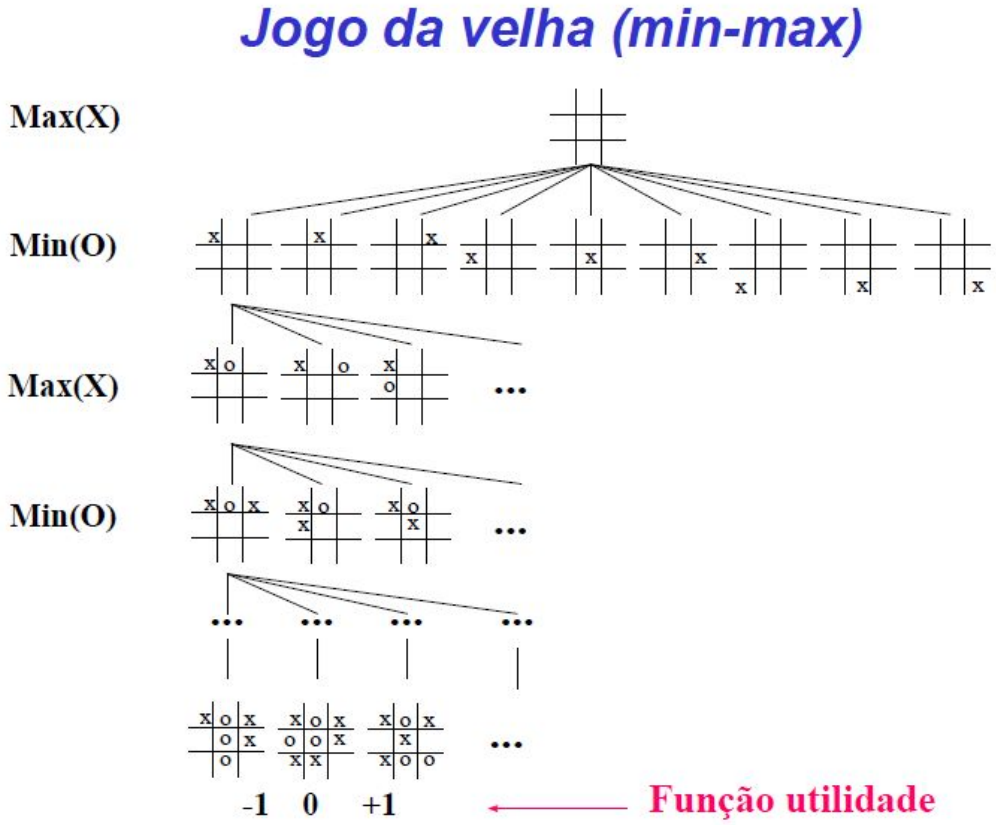
Árvores de decisão

- Uma solução possível para problemas deste tipo é utilizando **árvores de decisão**, em que podemos explorar todos os possíveis movimentos de cada jogador.
- Se temos sobreposição de subproblemas, então podemos aplicar Programação Dinâmica.
- Caso contrário, podemos aplicar um algoritmo de backtracking.
- Em cada estado, temos que considerar o jogador do momento, sendo que cada um está buscando ganhar o jogo.

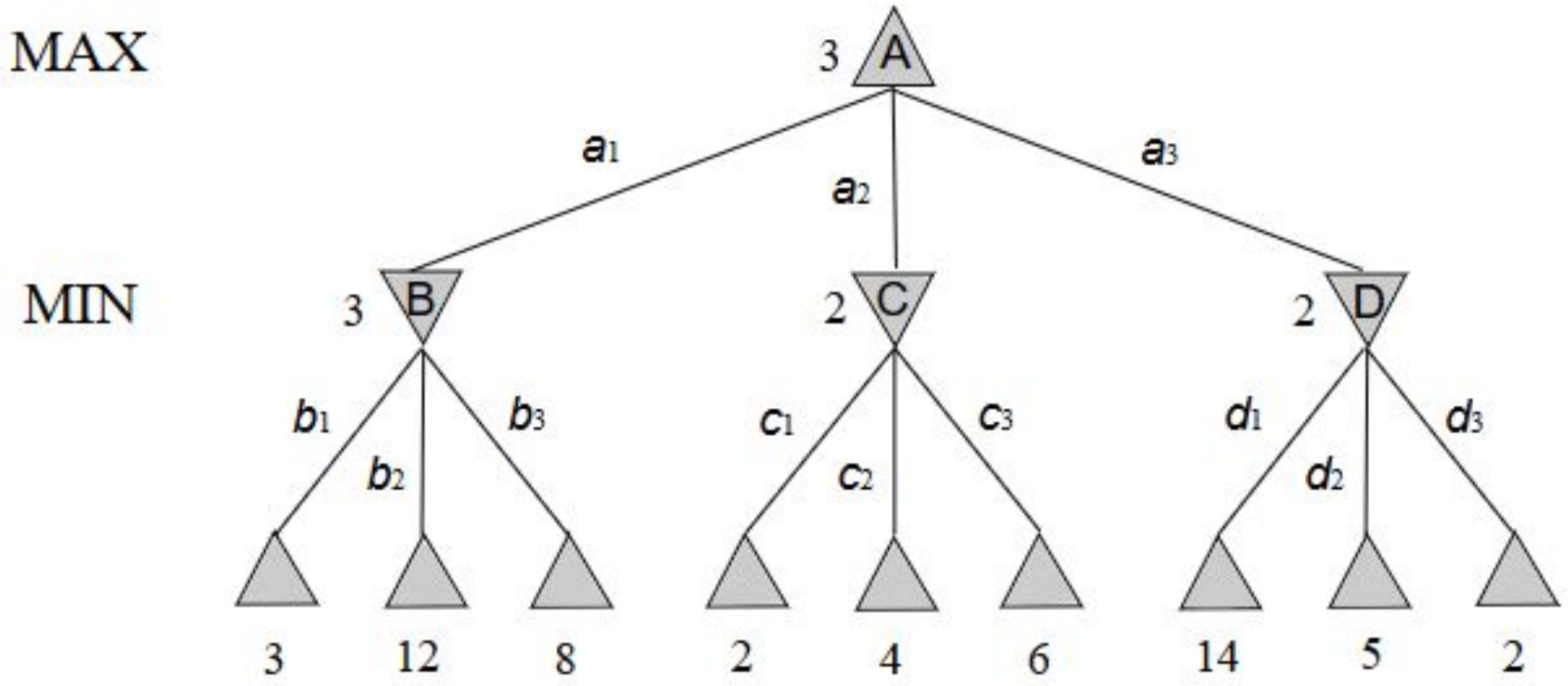
Algoritmo Minimax

- Se considerarmos a pontuação/ganho do jogador 1 como referência, podemos dizer que o jogador 1 está querendo **MAXIMIZAR** esse ganho, enquanto o jogador 2 tenta **MINIMIZAR**. Dessa estratégia obtemos o **Algoritmo Minimax.**
- Formulação:
  - Estado inicial: estado inicial do jogo + de quem é a vez
  - Estado final: posições em que o jogo acaba
  - Operadores: jogadas legais
  - Função de utilidade: valor numérico do resultado (pontuação)

Algoritmo Minimax



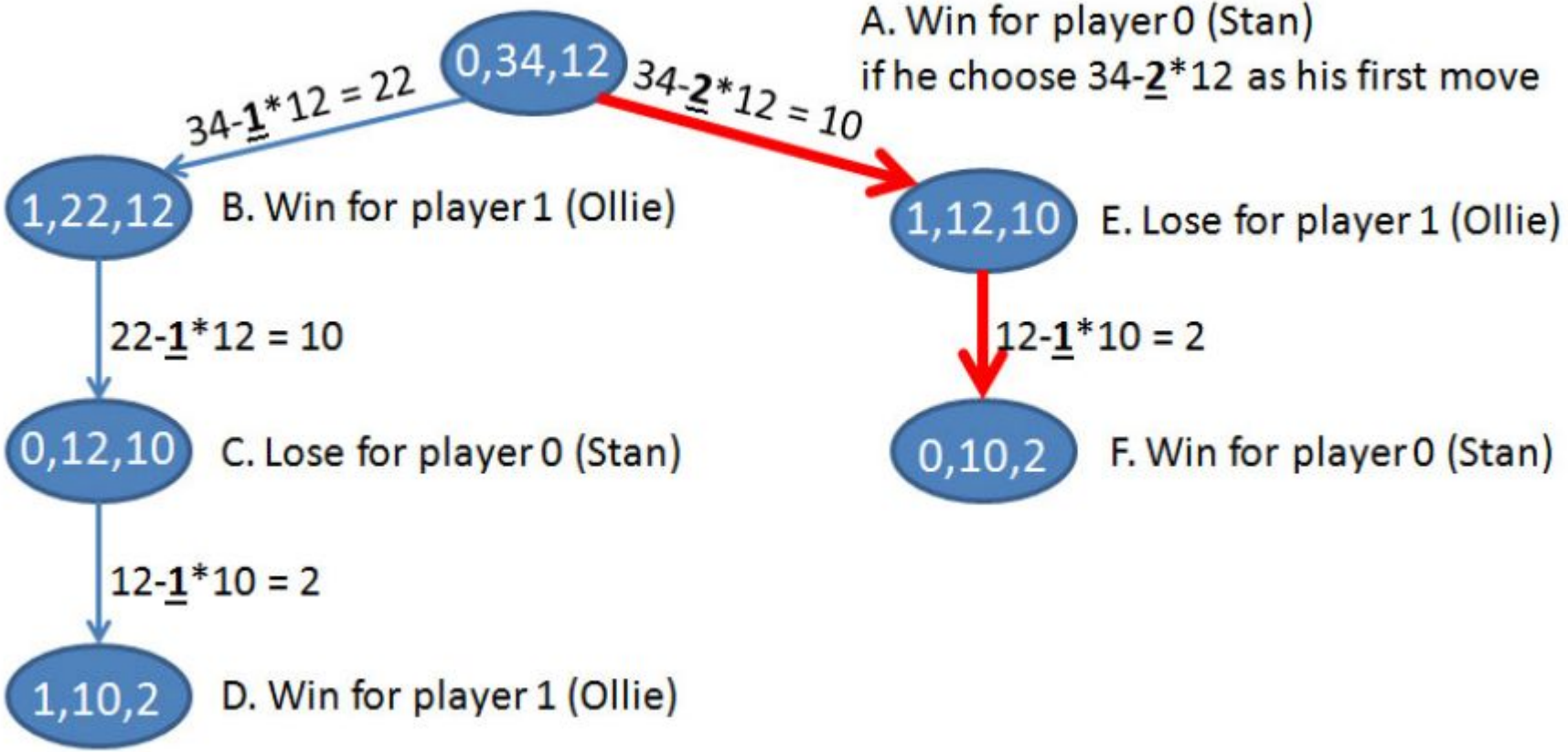
Algoritmo Minimax



Euclid's Game (UVA 10368)

- O estado do jogo consiste em uma tupla **(id, a, b)**, sendo a > b. O jogador atual **id** pode subtrair de **a** qualquer múltiplo de **b**, desde que o resultado seja maior ou igual a zero. Ganha o jogador que atingir o valor zero.

Euclid's Game (UVA 10368)



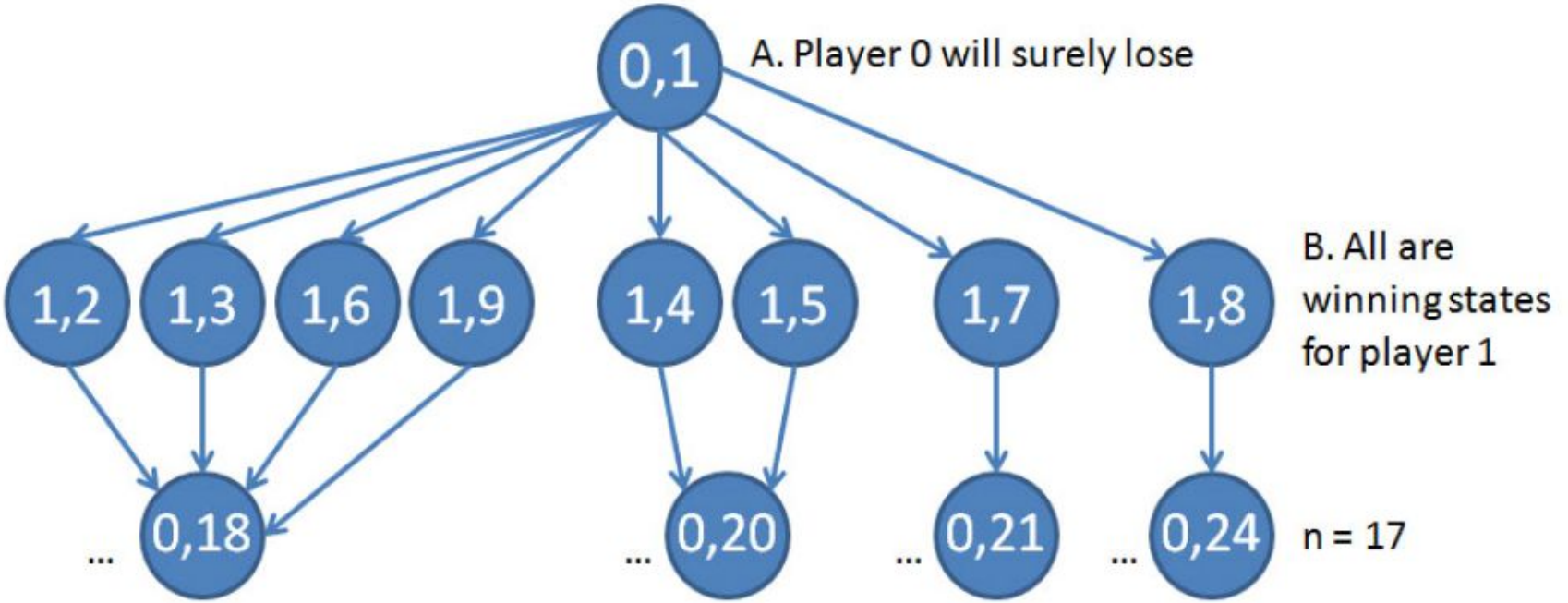
Insights para melhorar a solução

- Nem todos os jogos podem ser resolvidos explorando inteiramente a árvore de decisão, pois é comum que esta árvore seja muito grande.
- Porém, analisando melhor o problema pode-se obter alguns "*insights* matemáticos", percebendo padrões que podem nos guiar a solução de uma forma mais rápida, possivelmente até de forma gulosa/gananciosa.
- Explorar a árvore de decisão para problemas pequenos pode ajudar a perceber estes padrões.

A multiplication game (UVA 847)

- Neste jogo, o estado consiste no par **(id, p)**. O jogador atual **id** pode multiplicar o valor **p** por qualquer inteiro entre 2 e 9. Stan e Ollie jogam alternadamente, até que um deles seja capaz de obter um resultado maior ou igual a n (número alvo), sendo este jogador o vencedor.
- Stan é o primeiro jogador, começando com p = 1.

A multiplication game (UVA 847)



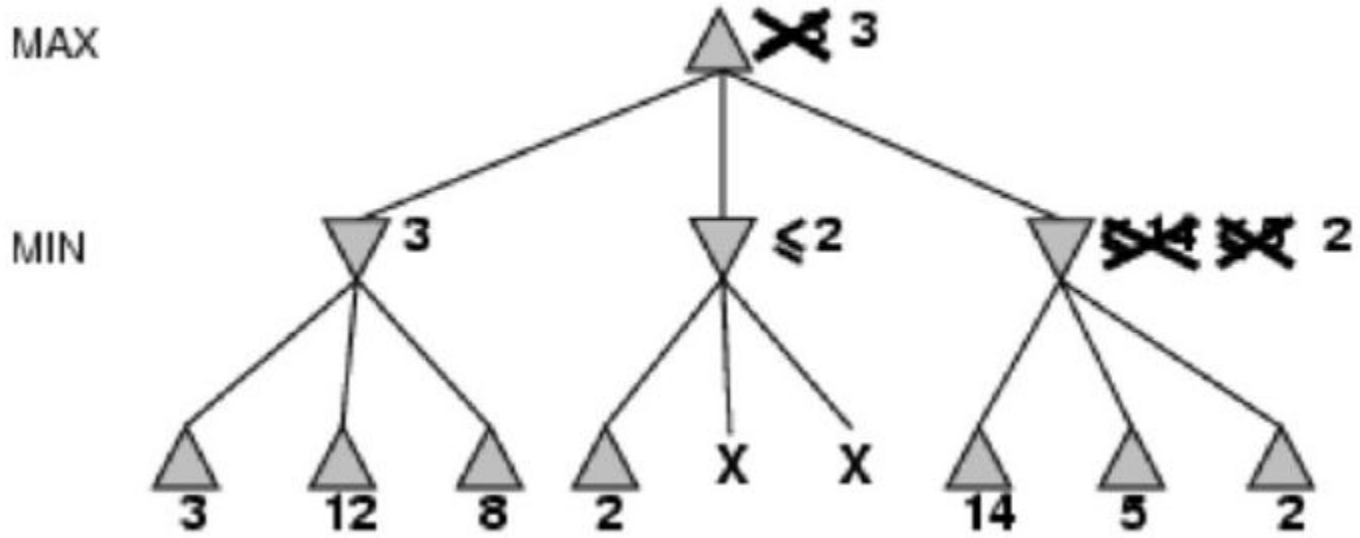
A multiplication game (UVA 847)

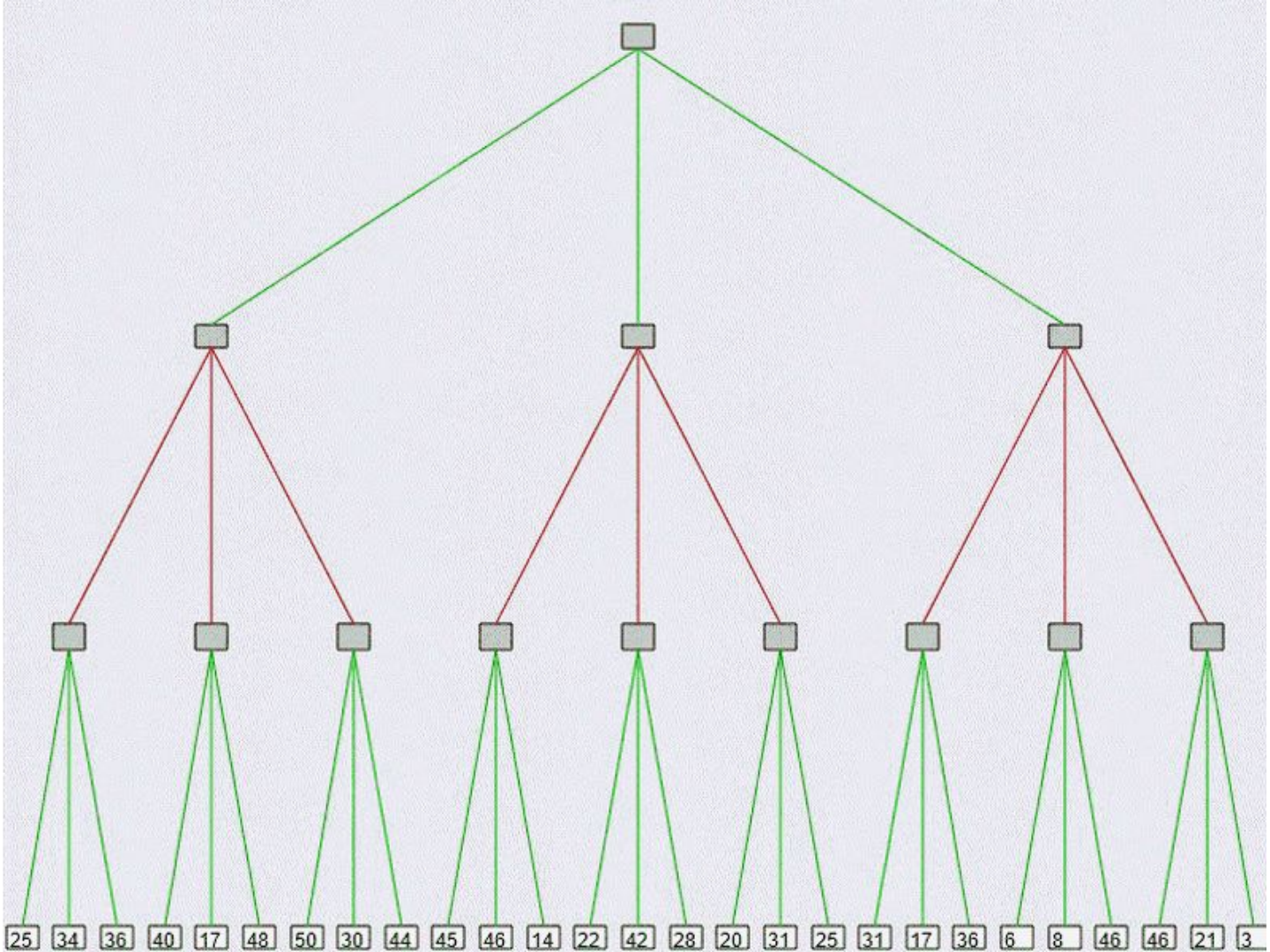
- Este é um exemplo de problema em que a árvore de decisão se torna muito grande, devido ao alto fator de ramificação (8).
- Porém, caso o competidor construa e analise esta árvore para casos mais simples, pode-se perceber que a seguinte estratégia corresponde a estratégia ótima para Stan:
  - Stan sempre multiplica p pelo maior valor possível (9)
  - Ollie sempre multiplica p pelo menor valor possível (2)

α-β Pruning

- Uma forma de otimizar a busca pelo algoritmo minimax é a **poda α-β**.
- Função: não expandir desnecessariamente nós durante o minimax.
- Ideia: se já encontramos uma solução, podemos eliminar ramos que sabemos que terão soluções piores.
- Para cada nó manteremos dois parâmetros:
  - **α:** melhor valor (no caminho) para MAX
  - **β:** melhor valor (no caminho) para MIN
- Teste de expansão:
  - **α** não pode diminuir (não pode ser menor que um ancestral)
  - **β** não pode aumentar (não pode ser maior que um ancestral)

α-β Pruning




25 34 36 40 17 48 50 30 44 45 46 14 22 42 28 20 31 25 31 17 36 6 8 46 46 21 3

Jogos combinatórios

- Caracterização:
  - Dois jogadores (alternando turnos)
  - Conjunto de estados
  - Regras de transição
  - Informação perfeita
  - Condição de vitória
    - Normal play: vence o último jogador a se movimentar
    - Misère play: vence o jogador que ficar sem movimentos
  - Não há opção de empate
- Exemplos: xadrez, damas, take-away, nim

Jogos combinatórios

- Em particular, nos concentraremos nos jogos combinatórios que possuem as seguintes restrições:
  - Imparciais: os dois jogadores possuem o mesmo conjunto de movimentos/ações possíveis.
  - Acíclicos: não podemos voltar a um estado anterior.
    - OBS: até podemos trabalhar com alguns problemas que permitem ciclos, porém que são derivados de ações “inúteis” e por isso podem ser desconsiderados.

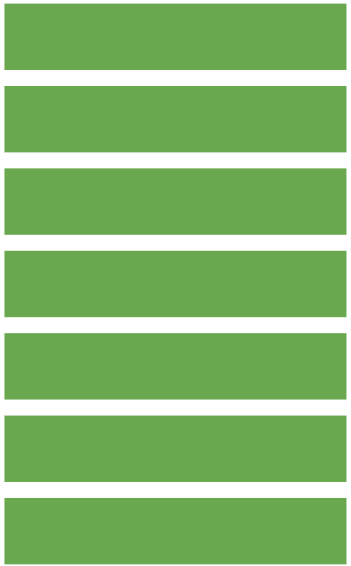
Jogos combinatórios x DAG

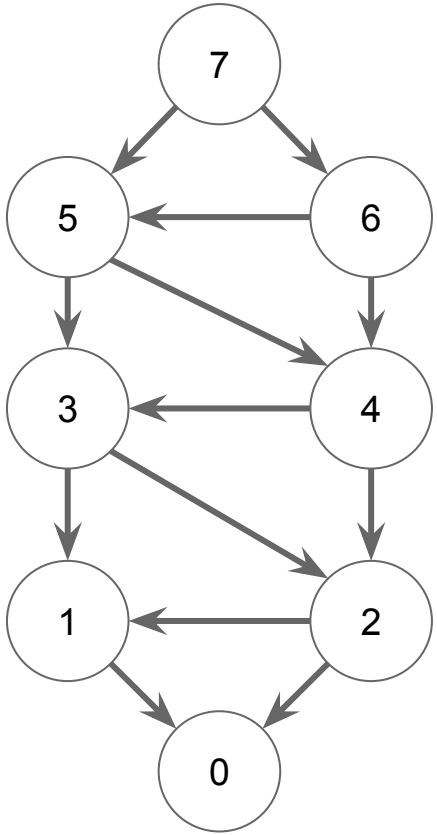
- Com esta caracterização, podemos modelar nosso problema como sendo um grafo direcionado acíclico, em que:
  - Os nós representam os estados;
  - As arestas representam as regras de transições (ações).
- E podemos determinar se um certo estado é vencedor ou perdedor baseado na seguinte recursão:
  - Caso base: nós terminais são perdedores (normal play);
  - Passo:
    - Um nó é vencedor se ele leva a PELO MENOS um nó perdedor;
    - Um nó é perdedor se ele leva SOMENTE a nós vencedores.

Exemplo: Take Away 1-2

- Temos uma pilha com N palitos
- A cada turno, o jogador atual remove 1 ou 2 palitos.
- Perde o jogador que não puder mais se movimentar (pilha vazia)

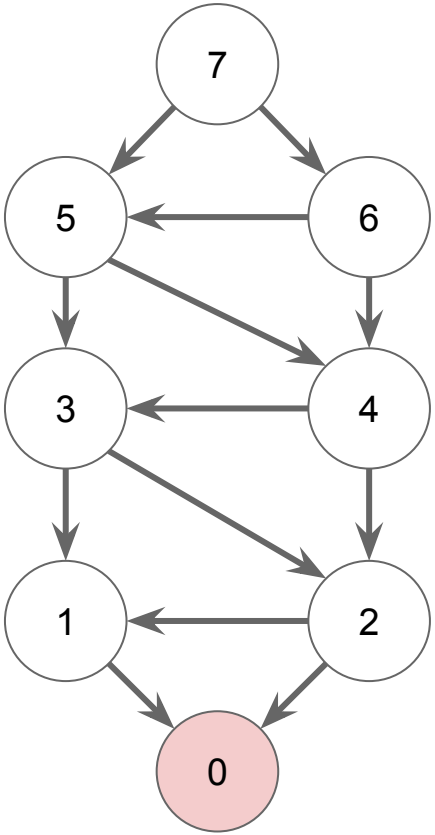
Exemplo: Take Away 1-2





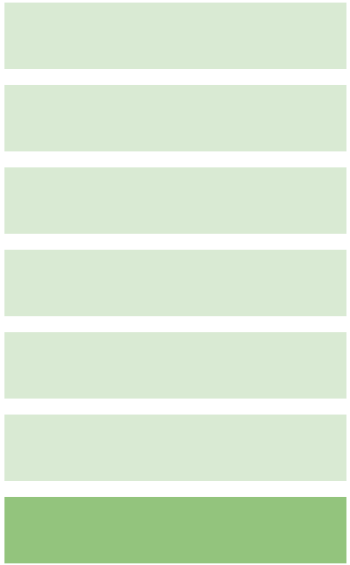
7	
6	
5	
4	
3	
2	
1	
0	

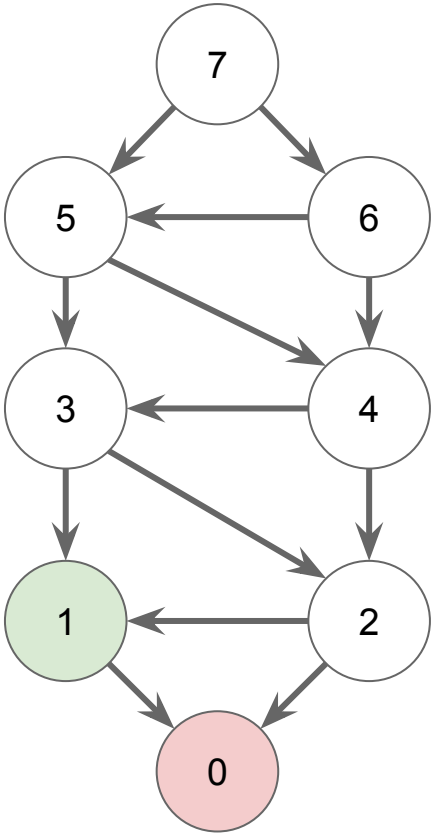
Exemplo: Take Away 1-2



7	
6	
5	
4	
3	
2	
1	
0	P

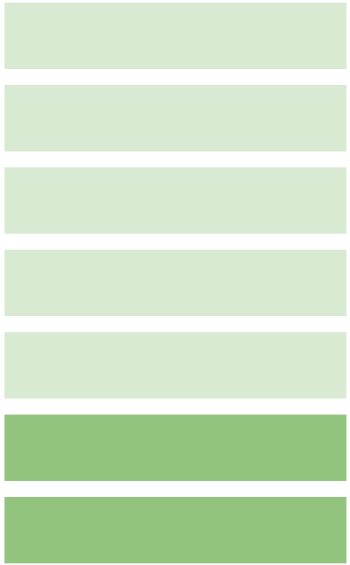
Exemplo: Take Away 1-2

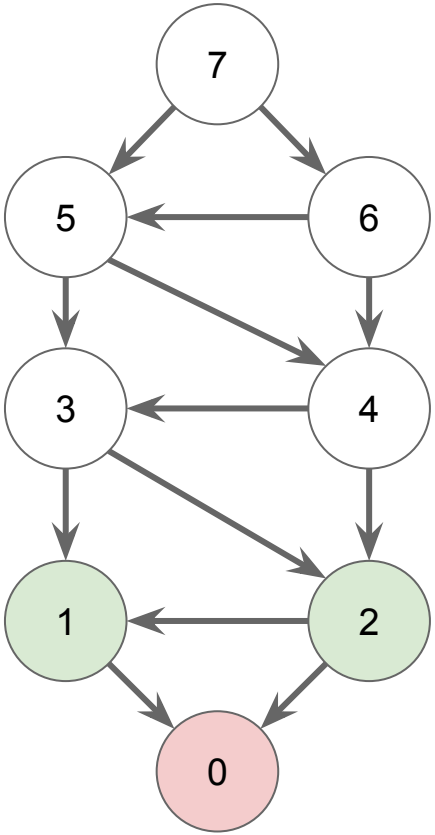




7	
6	
5	
4	
3	
2	
1	V
0	P

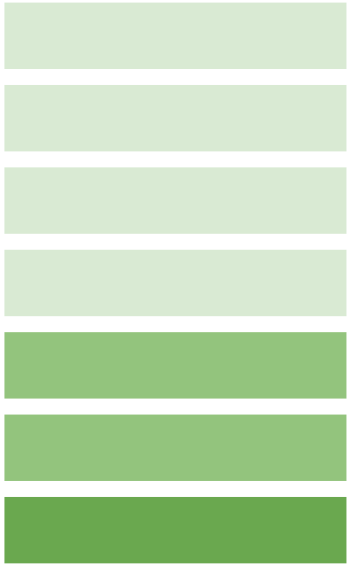
Exemplo: Take Away 1-2

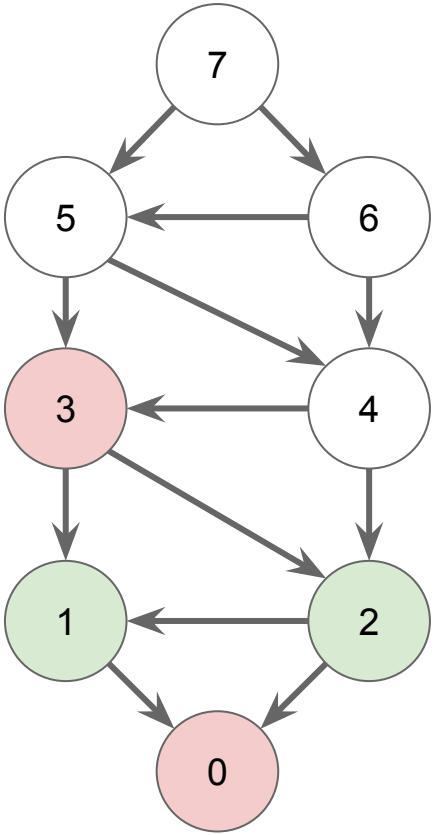




7	
6	
5	
4	
3	
2	V
1	V
0	P

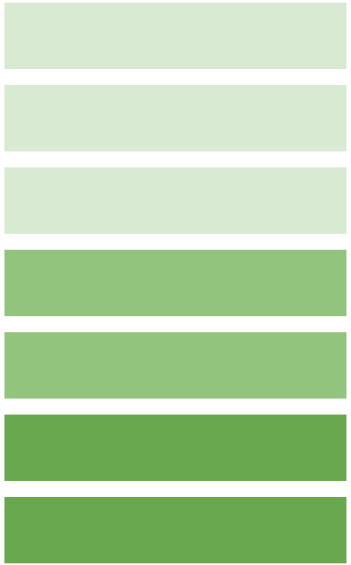
Exemplo: Take Away 1-2

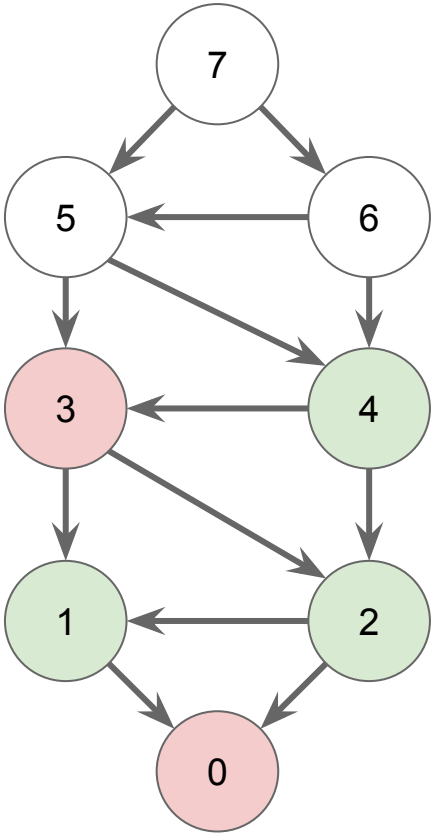




7	
6	
5	
4	
3	P
2	V
1	V
0	P

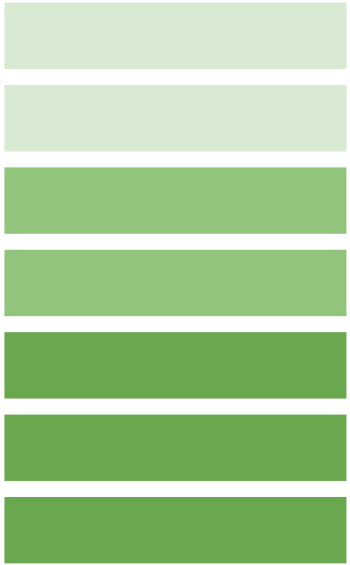
Exemplo: Take Away 1-2

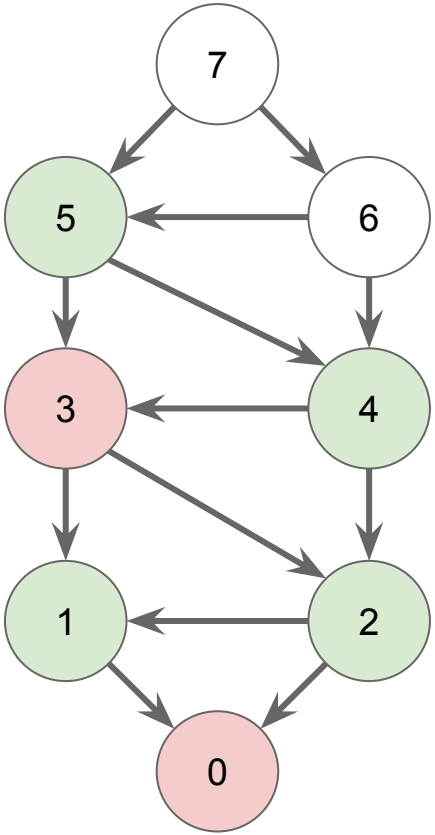




7	
6	
5	
4	V
3	P
2	V
1	V
0	P

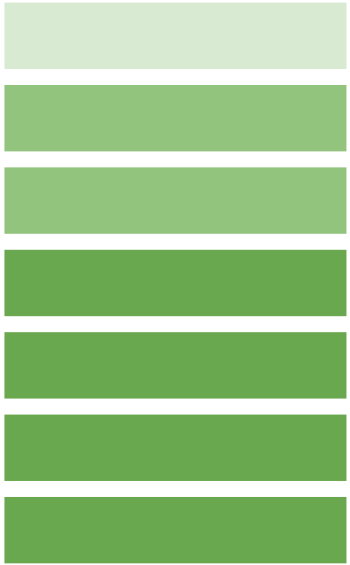
Exemplo: Take Away 1-2

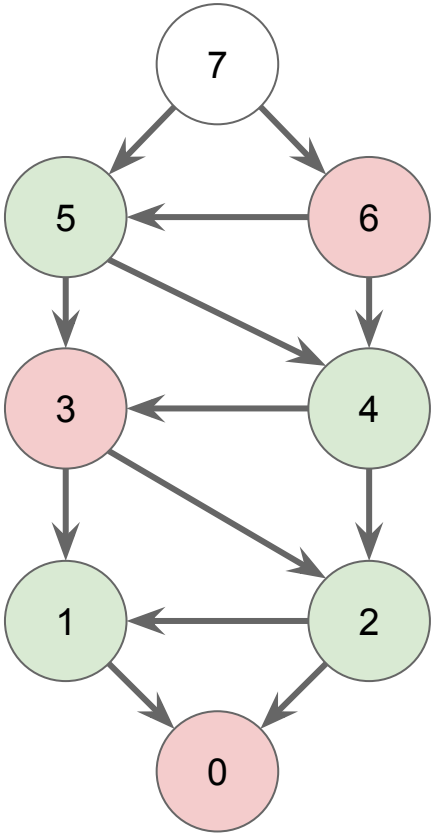




7	
6	
5	V
4	V
3	P
2	V
1	V
0	P

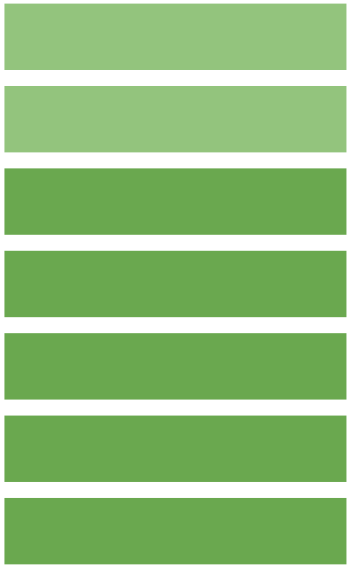
Exemplo: Take Away 1-2

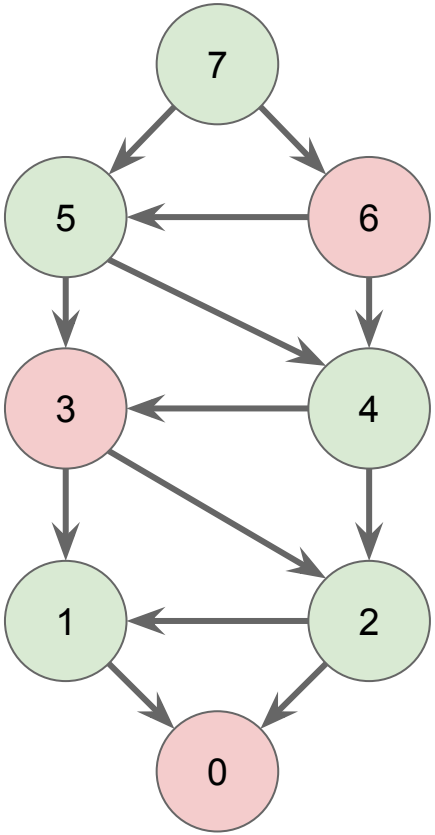




7	
6	P
5	V
4	V
3	P
2	V
1	V
0	P

Exemplo: Take Away 1-2





7	V
6	P
5	V
4	V
3	P
2	V
1	V
0	P

Nim

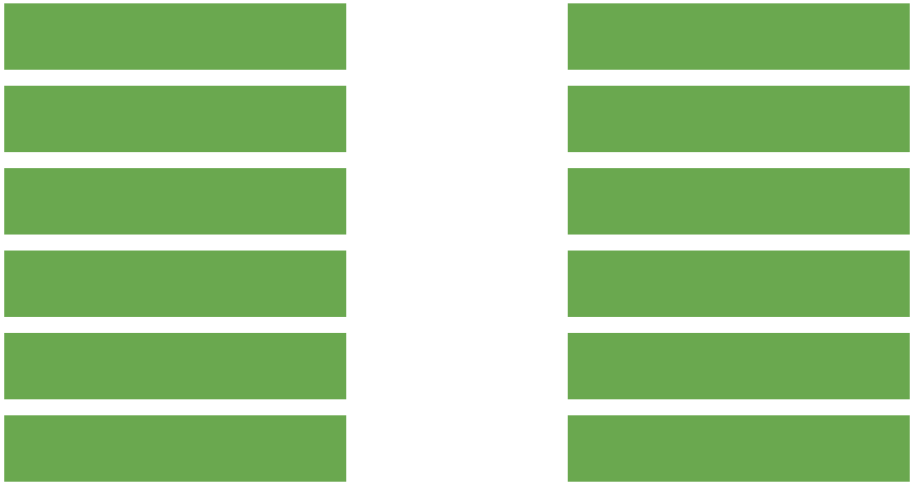
- Temos N pilhas de palitos.
  - A i-ésima pilha possui A_i palitos
- A cada turno, jogador atual escolhe uma pilha não vazia e remove quantos palitos desejar dela (pelo menos um)
- Perde o jogador que não puder mais se movimentar (quando todas as pilhas estiverem vazias)
- **Qualquer jogo imparcial entre dois jogadores com informação perfeita pode ser reduzido ao Nim.**

Nim

- Se N = 1
  - Caso degenerado: $A_0 = 0$ então o primeiro jogador perde
  - Caso contrário, o primeiro jogador sempre ganha

Nim

- Se N = 2
  - Jogador 1 só ganha se A_0 != A_1
  - Estratégia da cópia



Nim

- Se N = 2
  - Jogador 1 só ganha se A_0 != A_1
  - Estratégia da cópia
  - Se A_0= A_1 = 1, então é evidente que o jogador 1 perde

Nim

- Se N = 2
  - Jogador 1 só ganha se A_0 != A_1
  - Estratégia da cópia
  - Se A_0= A_1= x, e o jogador 1 retira y peças, o jogador 2 pode copiar a ação do jogador 1 na outra pilha, de forma que A_0=A_1=x-y. Depois de uma certa quantidade de passos, acabaremos no caso A_0=A_1=1.

Nim

- Se N = 2
  - Jogador 1 só ganha se A_0 != A_1
  - Estratégia da cópia
  - Se A_0= A_1= x, e o jogador 1 retira y peças, o jogador 2 pode copiar a ação do jogador 1 na outra pilha, de forma que A_0=A_1=x-y. Depois de uma certa quantidade de passos, acabaremos no caso A_0=A_1=1.

Nim

- Se N = 2
  - Jogador 1 só ganha se A_0 != A_1
  - Estratégia da cópia
  - Se A_0= A_1= x, e o jogador 1 retira y peças, o jogador 2 pode copiar a ação do jogador 1 na outra pilha, de forma que A_0=A_1=x-y. Depois de uma certa quantidade de passos, acabaremos no caso A_0=A_1=1.

Nim

- **Teorema**: O estado $(A_0, ..., A_{n-1})$ é perdedor sse $(A_0 \oplus ... \oplus A_{n-1}) = 0$.

  Exemplo: $3 \oplus 5 \oplus 3 \oplus 6 = 3 \neq 0$ ➜ Estado ganhador

Nim

- **Teorema**: O estado $(A_0, ..., A_{n-1})$ é perdedor sse $(A_0 \oplus ... \oplus A_{n-1}) = 0$.
- **Demonstração:** Por indução
  - Caso base: se todas as N pilhas estão vazias, então temos $0 \oplus ... \oplus 0 = 0$. Neste caso, estamos em uma posição perdedora.

Nim

- **Teorema**: O estado (A_0, ..., A_{n-1}) é perdedor sse ($A_0 \oplus ... \oplus A_{n-1}$) = 0.
- **Demonstração:** Por indução
  - Passo: considere $S = A_0 \oplus ... \oplus A_{n-1}$
    - Se **S = 0**, como esta é uma posição perdedora, então ao realizar qualquer ação, sempre levamos a um estado vencedor.
    - Considere um movimento qualquer em que reduzimos uma pilha de tamanho **x** para **y**. Usando as propriedades de $\oplus$ ($x \oplus x = 0$), sabemos que o *xor-sum* do novo estado é dada por:

      $$T = S \oplus x \oplus y = 0 \oplus x \oplus y = x \oplus y$$

    - Como $y < x$. Então $x \oplus y \neq 0$

Nim

- **Teorema**: O estado $(A_0, ..., A_{n-1})$ é perdedor sse $(A_0 \oplus ... \oplus A_{n-1}) = 0$.
- **Demonstração:** Por indução
  - Passo: considere $S = A_0 \oplus ... \oplus A_{n-1}$
    - Se **S != 0**, como esta é uma posição vencedora, podemos alcançar pelo menos um estado perdedor.
    - Considere a representação binária de S, sendo **d** o bit 1 mais significativo de S. A ação escolhida será pegar uma pilha que possua o bit **d** ligado, e diminuir o tamanho dessa pilha de **x** para **y = x** $\oplus$ **S**. Sendo assim, temos:

      $$T = S \oplus x \oplus y = S \oplus x \oplus (S \oplus x) = 0$$ ■

Valor de Grundy (nímero)

- **Mínimo excludente (mex):**
  - Seja X um conjunto de números inteiros não-negativos
  - Definimos mex(X) = min{x >= 0 / x ∉ X}
- Exemplos:
  - mex({0, 1, 2, 3}) = 4
  - mex({0, 1, 3, 5}) = 2
  - mex({1, 2, 5}) = 0

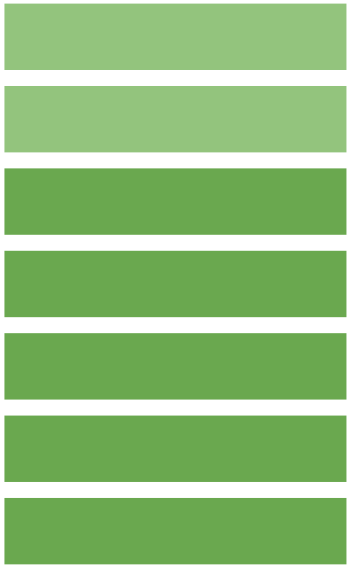
Valor de Grundy (nímero)

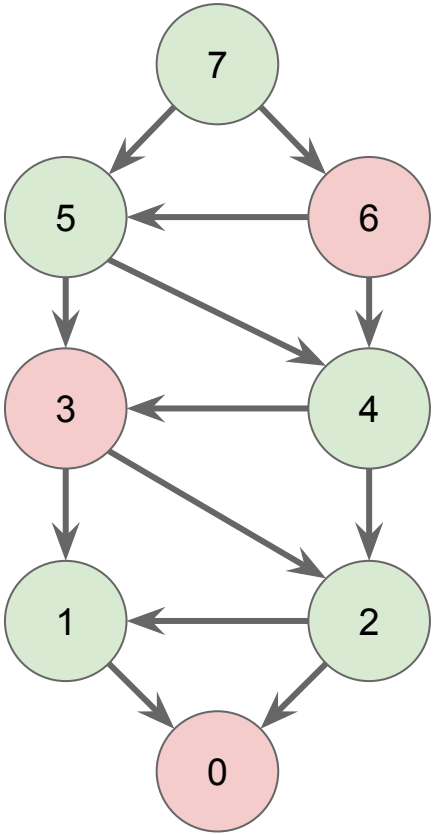
- **Função Sprague-Grundy**
  - Seja (X, F) um DAG
  - A função Sprague-Grundy de (X, F) é dada por

$$g(x) = \text{mex}\{g(y) \,/\, y \in F(x)\}$$

  - g(x) é dito o nímero (nimber) de x
- **Um estado *x* é perdedor sse g(x) = 0.**

Exemplo: Take Away 1-2





x		g(x)
7	V	1
6	P	0
5	V	2
4	V	1
3	P	0
2	V	2
1	V	1
0	P	0

Valor de Grundy (nímero)

- Calcular o valor de Grundy para alguns estados também facilita perceber possíveis padrões que ajudam a determinar mais fácil e eficientemente quando uma posição é vencedora ou perdedora.
- No problema Take Away 1-2, por exemplo, podemos perceber que a função **g(x)** é periódica, e que uma posição **x** é perdedora sse **x** é múltiplo de 3.

Exemplo: N Take Away 1-2

- Agora suponha que temos N “instâncias” do jogo Take Away 1-2, em uma espécie de Nim “modificado” em que limitamos a quantidade de palitos que podemos tirar de uma pilha.
- Partindo do exemplo para N = 2, podemos considerar que cada pilha representa dois jogos: (X_1, F_1) e (X_2, F_2).
- Como em cada turno uma ação só irá interferir em apenas um desses jogos, podemos definir o jogo (X, F) como a soma disjunta $(X_1, F_1) \oplus (X_2, F_2)$.
  - $X = X_1 \times X_2$
  - $F(x_1, x_2) = (F_1(x_1) \times \{x_2\}) \cup (\{x_1\} \times F_2(x_2))$

Exemplo: N Take Away 1-2

- Exemplo: para o estado (4,6) temos as seguintes possibilidades de ações:

  Ação no jogo 1

  - (3, 6)
  - (2, 6)

  Ação no jogo 2

  - (4, 5)
  - (4, 4)

Teorema de Sprague-Grundy

- Porém, nosso problema é simplificado devido ao seguinte teorema:
- **Teorema de Sprague-Grundy:** dado um jogo (X, F) formado pela soma disjunta de N subjogos (X_1, F_1), ..., (X_n, F_n), o nímero $g(x_1, ..., x_n)$ é dado por:

$$g(x_1, ..., x_n) = g_1(x_1) \oplus ... \oplus g_n(x_n)$$

- Com isso temos uma forma simples de “somar” jogos disjuntos (sendo que cada jogo nem precisa ser do mesmo tipo dos outros).

Exemplo: Crosses-crosses

- **O jogo:** considere um vetor de tamanho N com todas as posições vazias. Em um movimento, um jogador pode marcar uma posição com uma cruz(+), mas é proibido colocar duas cruzes em posições adjacentes. O jogador que ficar sem movimentos válidos perde.

		+				+				+			

Exemplo: Crosses-crosses

- **Solução:** quando o jogador coloca uma cruz em uma célula **i**, podemos considerar que ele dividiu o vetor em duas partes **independentes**, em dois subjogos, uma de tamanho **i - 2** e outro de **n - i - 1**. A partir disso, basta aplicarmos o Teorema de Sprague-Grundy.
- OBS:
  - Um estado **n** é perdedor se **n = 0**
  - Não esquecer dos casos específico em que **i = 0** ou **i = n-1**

$$g(n) = \text{mex}(\{g(n-2)\} \cup \{g(i-2) \oplus g(n-i-1) \mid 2 \leq i \leq n-1\})$$

Referências

Jonathan Queiroz. Aula sobre “jogos combinatórios” apresentada na Summer School 2019. https://www.youtube.com/watch?v=5kk_5HcwqOg

Prof. Dr. Ivan Rizzo Guilherme. Notas de aula da disciplina Inteligência Artificial.

Steven e Felix Halim. Competitive Programming 3

https://cp-algorithms.com/game_theory/sprague-grundy-nim.html

https://www.geeksforgeeks.org/combinatorial-game-theory-set-2-game-nim/?ref=rp

https://www.topcoder.com/community/competitive-programming/tutorials/algorithm-games/

https://medium.com/@lohitmarodia/game-theory-competitive-programming-98120cc14da3

https://www.ime.usp.br/~rvicente/IntroTeoriaDosJogos.pdf

https://www.geeksforgeeks.org/minimax-algorithm-in-game-theory-set-1-introduction/