Teoria dos Jogos
Exercícios

Laboratório de Programação Competitiva - 2020

Pedro Henrique Paiola

Coins Game (SPOJ MCOINS)

- **Problema**: dada uma torre de N moedas, cada jogador pode retirar 1, K ou L moedas da torre. O ganhador é o jogador que retirar as últimas moedas da torre.
- **Restrições:**
  - $1 < K < L < 10$
  - $1 <= N < 1000000$

Coins Game (SPOJ MCOINS)

- **Solução:** podemos modelar esse problema como um DAG em que:
  - Nós/estados: quantidade de moedas na torre
  - Arestas/transições: dadas pela retirada de 1, K ou L moedas.
- Como temos sobreposição de subproblemas então podemos aplicar Programação Dinâmica, considerando a seguinte recorrência:

$$g(n) = \begin{cases} 0 & \text{se } n = 0 \\ mex\{g(n-1), g(n-k), g(n-l)\} & c.c \end{cases}$$

Win or Freeze (CodeForces 150A)

- **Problema:** Dado um número Q, em cada turno o jogador deve substituí-lo por um de seus divisores não triviais (!= 1 e != Q). Ganha o jogador que não tiver movimentos possíveis (misère play)
- **Restrições:**
  - $1 <= Q <= 10^{13}$

Win or Freeze (CodeForces 150A)

- **Solução 1:** primeiramente, podemos tentar uma solução mais intuitiva modelando o problema como um DAG e aplicando Programação Dinâmica.
- Estado: valor q
- Transições: dada pelos divisores não triviais
- Estados vencedores: números primos (só possuem divisores triviais)
- Complexidade: O(n.sqrt(n))

Win or Freeze (CodeForces 150A)

- **Solução 2:** mas podemos analisar um pouco melhor o problema, chegando em uma solução mais direta.
- Considere a fatoração de um número **Q** em **x** números primos.
- Todos os possíveis divisores não triviais de Q são dados pelo produto de 1 ou mais (até **x - 1**) desses fatores.

Win or Freeze (CodeForces 150A)

- Se Q é primo, então este é um estado vencedor.
- Se Q possui apenas dois fatores primos, então este é um estado perdedor, pois o jogador deve substituir o número por um destes fatores, o que levará o adversário a vitória

$$Q = p_1 . p_2 \leftrightarrow \text{estado perdedor}$$

- Caso contrário, então estamos em um estado vencedor, porque podemos selecionar o produto de quaisquer dois fatores, o que levará o próximo jogador a situação anterior.

$$Q = \mathbf{p_1 . p_2} p_3 ... p_4 \leftrightarrow \text{estado vencedor}$$

Win or Freeze (CodeForces 150A)

- Sendo assim, basta fatorar o número Q e, a partir da quantidade de fatores, é possível saber qual jogador irá ganhar o jogo.
- Se x = 1
  - Jogador 1 vence e sem fazer nenhuma jogada.
- Se x = 2
  - Jogador 2 vence.
- Caso contrário (x > 2)
  - Jogador 1 vence, substituindo Q pelo produto de dois fatores primos.
- Complexidade: O(sqrt(n))

28042797	PedroHP98	CodeForces	150A	Accepted	202
28042795	PedroHP98	CodeForces	150A	Accepted	31

Lieges of Legendre (CodeForces 603C)

- **Problema**: neste jogo, temos **n** pilhas de palitos. Em cada jogada, um jogador pode remover um palito de uma pilha ou pegar uma pilha com um número par de palitos e substituí-la por **k** pilhas de **n/2** palitos.
- **Restrições:**
  - $1 <= n <= 100000$
  - $1 <= k <= 10^9$
  - $1 <= a_i <= 10^9$

Lieges of Legendre (CodeForces 603C)

- Em primeiro lugar, pelo Teorema de Sprague-Grundy sabemos que podemos considerar cada pilha como um jogo separado e, calculando o nímero $\mathbf{g(a_i)}$ para cada pilha **i** podemos calcular $\mathbf{g(a_1, ..., a_n) = g(a_1) \wedge ... \wedge g(a_n)}$.
- Sendo assim, vamos nos concentrar em analisar o jogo com uma única pilha.

Lieges of Legendre (CodeForces 603C)

- Para cada estado possível, determinado pelo tamanho da pilha (**a**), temos duas possibilidades:
  - Retirar um palito: **a - 1**
  - Se **a** é par, então substituir a pilha por **k** pilhas de tamanho **a/2**
    - Neste caso, este jogo será dividido em **k** subjogos. Novamente podemos considerar o Teorema de Sprague-Grundy e dizer que o nímero desse jogo é **g(a/2) ^ .... ^ g(a/2)**
      k vezes

Lieges of Legendre (CodeForces 603C)

- Porém, sabemos que **x ^ x = 0** e **x ^ 0 = x**. Sendo assim, se k é par, então **g(a/2) ^ .... ^ g(a/2) = 0**, caso contrário **g(a/2) ^ .... ^ g(a/2) = g(a/2).**
- Agora, considerando ambos os movimentos possíves, temos que

$$g(a) = \begin{cases} mex\{g(a-1)\} & \text{se } a \text{ é ímpar} \\ mex\{g(a-1), g(a/2)\} & \text{se } k \text{ é ímpar} \\ mex\{g(a-1), 0\} & c.\,c \end{cases}$$

Lieges of Legendre (CodeForces 603C)

- No entanto, como o tamanho de cada pilha pode ser muito grande, isso inviabiliza a aplicação de PD. Por isso precisamos refletir um pouco mais sobre o problema na busca de padrões.

Lieges of Legendre (CodeForces 603C)

- Se **k** é par
  - Para **a** > 2
    - Se **a** é par, então **g(a) = 1**
    - Se **a** é ímpar, então **g(a) = 0**
  - É fácil perceber que quando **a** é par estamos em um estado ganhador, afinal, g(a) = mex{g(a-1), 0} > 0
  - Agora, a partir de **a** ímpar, só podemos ir para o estado **a-1**, que é par, logo **g(a-1)>0**. Então, **g(a) = 0**

0	0
1	1
2	2
3	**0**
4	**1**
5	**0**
6	**1**

Lieges of Legendre (CodeForces 603C)

- Se **k** é ímpar
  - Para **a** > 4
    - Se **a** é par, então **g(a) > 0**
    - Se **a** é ímpar, então **g(a) = 0**
  - Se **a** é ímpar, ele só pode ir ao estado **a-1**. Ou seja, ele só pode ser um estado ganhador se **g(a-1)** for zero.
  - Porém, de **a** par também podemos ir para o estado **a-1**, e **g(5)** = 0, logo a partir disso surge um ciclo de alternância entre estados perdedores e vencedores.

0	0
1	1
2	0
3	1
4	2
5	**0**
6	**2**

Lieges of Legendre (CodeForces 603C)

- Diferente do caso que **k** é par, isso não nos dá todos os nímeros imediatamente.
- Sabemos que **g(a) = 0** se **a** é ímpar, mas se for par então **g(a)** pode ser 1 ou 2.
- Para saber disso, basta calcular **mex{0, g(a/2)}**, o que pode ser resolvido recursivamente em O(log n).

0	0
1	1
2	0
3	1
4	2
5	**0**
6	**2**

Marbles (Regional 2018)

- **Problema:** dado um tabuleiro com N bolinhas de gude, em cada turno um jogador pode movimentar qualquer bolinha de gude da posição (l, c) para a posição:
  - (l - u, c- u) ou
  - (l - u, c) ou
  - (l, c - u)
- Vence o primeiro jogador a mover alguma bolinha para (0, 0)
- **Restrições:**
  - 1 <= N <= 1000
  - 1 <= L, C <= 100

Marbles (Regional 2018)

- **Solução:** em primeiro lugar, consideraremos cada bolinha separadamente e "uniremos" os resultados pelo Teorema de Sprague-Grundy.
- Como ambos os jogadores jogam de forma ótima, não iremos considerar as jogadas em que uma bolinha seja movida para uma posição em que
  - l = 0 ou
  - c = 0 ou
  - l = c
- Nesse caso, as posições perdedoras são (1,2) e (2,1), onde não temos como mover uma bolinha para uma posição que não se encaixe em uma das situações acima.