# Algoritmos

Caique Paiva

Maio 2023

Um algoritmo é basicamente uma sequência de operações. Vendo assim, parece simples, porém, existe uma série de problemas que, surpreendentemente, sai usando algum algoritmo. Normalmente, usamos algoritmos em combinatória, porém, eles também combinam muito bem com álgebra e teoria dos números, pois eles nos ajudam a construir coisas.

Além disso, como algoritmos é uma sequência de passos, e problemas de sequência de passos normalmente combinam muito bem com invariantes e monovariantes, esses dois conteúdos andam lado a lado em resolução de problemas.

Existem 2 tipos de algoritmos principais, os gulosos e os indutivos. Vamos estudar eles dois nesse material!

## §1 Algoritmos Gulosos

Um algoritmo guloso é um algoritmo no qual nós fazemos uma operação que tenta maximizar a resposta imediatamente, e então maximizando a repsosta para todos. Nem sempre esse algoritmo vai funcionar, porque ele pode afetar a resposta de outros passos, porém ele ainda é bem útil. Vamos ver algumas aplicações dele!

**Exemplo 1.1**

Seja $G$ um grafo no qual o grau máximo de todos os vértices é $\Delta$. Prove que podemos colorir o grafo usando no máximo $\Delta + 1$ cores, de modo que, se dois vértices forem adjacentes, eles tem cores diferentes.

Vamos fazer o seguinte algoritmo: Primeiro, ordene os vértices em uma maneira aleatória, e vamos numerar as cores de $1, 2, 3, \cdots x$, então,

- Passo $i$: Pegue o primeiro vértice da ordenação que ainda não foi colorido, e vamos colorir ele com a menor cor que podemos usar atualmente nele (Por exemplo, se esse vértice for adjacente a vértices com cores $1, 2, 4$, então, pintamos ele com a cor 3.

Veja que esse algoritmo obviamente gera uma coloração válida, então, vamos provar que $x \leq \Delta + 1$. Vamos provar o seguinte lema sobre o algoritmo

**Lema 1.2**

Seja $v$ um vértice em $G$. Então, sua coloração é no máximo $\deg v + 1$.

*Prova:* A prova é bem simples. Veja que, pela construção do algoritmo, se a cor $i$ não pertence a nenhum dos vizinhos de $v$, então, a cor de $v$ é no máximo $i$, então, por PCP, se eu tenho $\deg v + 1$ cores para colocar em $\deg v$ vértices, então, alguma cor não vai aparecer

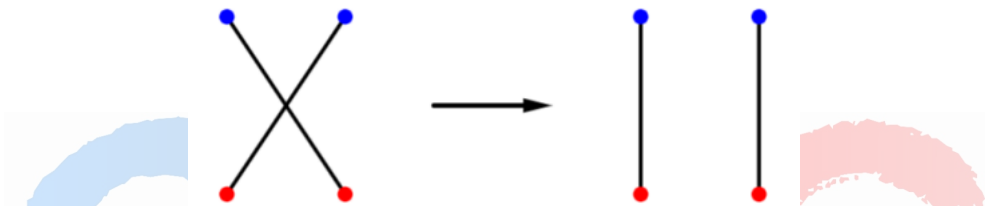
Então, temos que $x \leq \max(\deg v) + 1 = \Delta + 1$, como queriamos provar.

A maior parte de problemas gulosos são assim. O algoritmo normalmente é muito simples de construir, porém, provar que ele funciona é o mais complicado. Vamos ver mais um exemplo.

**Exemplo 1.3** (Putnam 1979)

Temos $n$ pontos vermelhos e $n$ pontos azuis no plano, sem 3 colineares. Prove que podemos desenhar $n$ segmentos, juntando um ponto azul com um ponto vermelho, de modo que nenhum par de segmentos se intersecte.

Em problemas de combinatória, é sempre bom fazermos casos pequenos. Vamos fazer o caso $n = 2$.



‘ Nesse caso, se escolhermos os segmentos aleatoriamente, podemos ou chegar numa configuração boa, ou nós cortamos eles dois, mas, se nós cortamos, é fácil desembolar eles (Só fazer a operação da figura).

Portanto, seria bom se pudessemos fazer isso sempre... e nós podemos!! Começe com uma ligação de vermelhos e azuis aleatória, e então, vamos fazer o seguinte algoritmo

- Passo $i$: Escolha dois segmentos que se cortam, e faça a operação da figura acima. Faça isso até que não tenha mais segmentos se cortando.

Certo, só temos um problema nesse algoritmo, como conseguimos garantir que ele acaba? Para isso, vamos atrás de uma monovariante! Vamos provar o seguinte lema:

**Lema 1.4**

Seja $ABCD$ um quadrilátero, de modo que $A, B$ são pontos pintados de vermelho e $C, D$ são pontos pintados de azul. Suponha que $\overline{AC} \cap \overline{BD}$ existe, então, temos que

$$AD + BC < AC + BD$$

*Proof.* Seja $P = \overline{AC} \cap \overline{BD}$, então, pela desigualdade triângular, temos que

$$\begin{cases} AP + PD > AD \\ BP + PC > BD \end{cases} \implies AP+PD+BP+PC > AD+BD \implies AC+BD > AD+BC$$

□

Com isso, vamos pegar nossa invariante para ser a soma dos tamanhos dos segmentos!! Como ela sempre diminui quando fazemos uma operação por conta do lema acima (E temos valores finitos para essas somas de segmentos), eventualmente, vamos chegar na configuração de tamanho mínimo, e não vamos mais poder fazer operações, ou seja, chegamos num conjunto de segmentos que não se intersectam.

A chave para matar esse problema foi fazer o caso $n = 2$, e tentar repetir ele várias vezes até dá certo. Isso é algo que sempre nos ajuda em problemas de algoritmo: Se temos uma operação que nos ajuda a chegar mais perto do final do algoritmo, aplique ela quantas vezes você puder.

**Exemplo 1.5** (IMO SL C2 2020)

Em um polígono regular de 100 lados, 41 vértices são coloridos de preto e 59 são coloridos de branco. Prove que existe 24 quadriláteros convexos tal que

- Os quadriláteros são disjuntos 2 a 2
- Todo quadrilátero tem 3 vértices de uma cor e 1 vértice da outra cor

Primeiro, o que seria bom que acontecesse para a gente? Bem, se no polígono, nós temos 4 pontos consecutivos de modo que 3 sejam de uma cor e 1 seja da outra cor, nós podemos pegar esses 4 pontos, e eles nunca se intersectariam com nenhum outro ponto! Com isso, vamos formular o seguinte algoritmo

- Passo $i$: Seja $P_1P_2\cdots P_{4k}$ o nosso polígono atual, então, suponha sem perda de generalidade, que temos mais brancos do que pretos. Então, escolha um $i$ tal que $P_i, P_{i+1}, P_{i+2}, P_{i+3}$ seja um conjunto de pontos com 3 brancos e um preto, então, remova-os do polígono.

Perfeito, só temos um problema com esse algoritmo... Como garantimos que esse $i$ existe? Vamos provar que essa quádrupla sempre existe!

**Lema 1.6**

Seja $P_1P_2\cdots P_n$ um polígono regular de $4n$ pontos, com $x$ desses pontos de brancos e $y$ de pretos, com $x + y = n$ e com $x \neq y, x \geq 1, y \geq 1$. Suponha sem perda de generalidade que temos mais brancos do que pretos, então, existe $P_iP_{i+1}P_{i+2}P_{i+3}$ de modo que 3 desses pontos são brancos e 1 é preto.

*Proof.* Para cada ponto $P_i$, defina $x_i$ de modo que

$$x_i = \begin{cases} 1 & \text{se } P_i \text{ é branco} \\ -1 & \text{se } P_i \text{ é preto} \end{cases}$$

Primeiro, se temos 4 pontos consecutivos brancos, o problema acabou por continuidade discreta! Então, só precisamos provar que $\exists i$ tal que

$$x_i + x_{i+1} + x_{i+2} + x_{i+3} > 0$$

Então, suponha por absurdo que $x_i + x_{i+1} + x_{i+2} + x_{i+3} \leq 0$ para todo $i$, então, somando todas as equações para todos os $i$, temos que

$$4\sum_{i=1}^{n} x_i \leq 0 \Longrightarrow \sum_{i=1}^{n} x_i \leq 0$$

Absurdo! pois sabemos que temos mais brancos do que pretos no polígono! □

Perfeito, conseguimos provar esse lema! Porém, ele tem uma condição muito importante, que é que a quantidade de pontos pretos é diferente da quantidade de pontos brancos. Além disso, veja que o problema pede 24 polígonos, e não 25 polígonos, ou seja, podemos descartar 4 pontos do problema! Então, vamos descartar um ponto branco, e então, a quantidade de pontos vai ser um número ímpar, e então, nunca vamos ter a quantidade de pontos brancos igual a quantidade de pontos pretos.

Com isso, basta rodar esse algorítmo várias vezes! Vou escrever em formato de um par, onde o primeiro número é a quantidade de brancos e o segundo número é a quantidade de pretos. Logo, vou fazer uma simulação do algoritmo

$$\begin{aligned}(58,41) &\to (55,40) \to (52,39) \to (49,38) \to (46,37) \to (43,36)\\ &\to (40,35) \to (37,34) \to (34,33) \to (31,32) \to (30,29)\\ &\to (27,28) \to (26,25) \to (23,24) \to (22,21) \to (19,20)\\ &\to (18,17) \to (15,16) \to (14,13) \to (11,12) \to (10,9)\\ &\to (7,8) \to (6,5) \to (3,4) \to (2,1).\end{aligned}$$

Veja que sobra 2 pontos brancos e 1 ponto preto, e como ainda podemos descartar 3 pontos, descartamos eles 3, e o problema acabou!

# §2 Algoritmos Indutivos

Bem, a ideia para algoritmos indutivos é bem simples: Pegar o caso com $n$ elementos, e reduzir ele para um caso com menos elementos. Normalmente é ir de $n$ elementos para $n-1$ elementos, porém temos outros casos posíveis também. Vamos ver alguns aqui!

**Exemplo 2.1** (IMO SL 2005 C1)

Uma casa tem um número par de lâmpadas distribuidas em quartos de modo que em cada quarto temos pelo menos 3 lâmpadas. Cada lâmpada está conectada exatamente com um outra lâmpada por meio de um interruptor. Cada vez que mudamos o interruptor, as duas lâmpadas mudam de estado simultaneamente. Prove que para qualquer estado inicial das lâmpadas, existe alguma sequência de movimento de interruptores de modo que em cada quarto temos pelo menos uma lâmpada ligada e uma lâmpada desligada.

Vamos chamar um quarto de ruim se todas as lâmpadas dele estão ligadas ou todas estão desligadas. Seja $k$ a quantidade de quartos ruins. Vamos fazer um algoritmo que diminui $k$ para $k-1$.

Começe por um quarto $Q_1$ ruim, e então, escolha uma lâmpada aleatória desse quarto e mude o estado dela. Se essa lâmpada estiver conectada com outra lâmpada desse mesmo quarto, o quarto vai ficar bom! Pois tem pelo menos 3 lâmpadas em cada quarto, e então vamos conseguir o que queriamos. Agora, se ele estiver conectado a outro quarto $Q_2$, e quando virarmos essa lâmpada o quarto ficar bom, nós também vamos ter conseguido diminuir a quantidade de quartos ruins! Agora, o problema é se o quarto $Q_2$ era bom, e ficou ruim depois de mudar essa lâmpada, e então, o que fazer?

Uma coisa muito importante em matemática é saber quando desistir das suas ideias. Essa ideia não parece promissora, porque quando eu tento diminuir a quantidade de quartos ruins, pode aparecer novos quartos ruins nesse processo. Porém, e se a gente continuar tentando fazer esse processo, o que vai acontecer?

Com isso, vamos definir o sequinte algoritmo, começando pelo quarto $Q_2$.

- Passo $i \geq 2$: Seja $Q_i$ o quarto que estamos atualmente, $Q_{i-1}, L_{i-1}$ o quarto que estavamos no passo anterior e a lâmpada que mudamos. Então, se $Q_i$ for bom, pare o algorimto, e a gente conseguiu o que queriamos! Agora, se $Q_i$ for ruim, escolha uma lâmpada $L$ diferente de $L_{i-1}$, e mude o estado dela. Seja $L_i$ a lâmpada que estava conectada a $L$. Se, depois de mudarmos $L_i$, o quarto que tem $L_i$ ficar bom, paramos o algoritmo e conseguimos o nosso objetivo. **Se o quarto novo ficar ruim, e nós já passamos por ele, paramos o algoritmo**, e caso contrário, continuamos o algoritmo com o próximo quarto sendo o que contem $L_i$.

O único caso em que esse algoritmo pode não funcionar é, se voltarmos para um quarto já visitado, podemos não diminuir a quantidade de quartos ruins, mas vamos provar que ele funciona até nesse caso!

Seja $m, n$ os números dos passos onde $Q_m = Q_n$. Veja que, quando entramos em $Q_n$ pela primeira vez, o quarto é ruim, e então fazemos uma operação para deixar ele bom, mudando somente uma lâmpada. Agora, quando estamos em $Q_{m-1}$, e vamos mudar uma lâmpada para voltar para $Q_m$, nós vamos mudar uma lâmpada diferente da que mudamos primeiro, pois essa lâmpada estava conectada a $Q_{n+1}$, então, nós mudamos duas lâmpadas diferentes em $Q_m$, então, esse quarto é bom! (O quarto começa ruim, então, temos pelo menos 3 lâmpadas do mesmo estado, e se trocarmos duas lâmpadas diferentes, é impossivel que as 3 lâmpadas sejam iguais).

Portanto, esse algoritmo realmente funciona! E o problema acabou.

Esse problema mostra bem como funciona um algoritmo indutivo: Uma indução onde a gente tem que fazer várias operações dentro do próprio passo indutivo. Outra maneira de fazer um algoritmo indutivo é fazer uma indução com uma transição estranha... Vamos ver um exemplo.

**Exemplo 2.2**

Ana adora moedas. Ela tem $3^n$ moedas de mesmo peso. Miguel, como o menino travesso que ele é, trocou uma das moedas por outra moeda idêntica, só que com peso menor. Ana percebeu que tinha algo estranho com as moedas, então, ela pegou uma balança de dois pratos para descobrir qual é a moeda falsa. **Ela sabe que a moeda falsa tem peso menor que as outras**. Prove que ela sempre consegue descobrir em $n$ perguntas.

Veja que, se em cada pesagem, conseguimos reduzir o nosso conjunto de respostas de $x$ para $\frac{x}{3}$, o problema acabou, pois aí gastariamos $n$ operações no total, então, vamos montar um algoritmo que realize isso para a gente! Nós começamos com $3^n$ moedas que podem ser falsas.

- Passo $i$: Suponha que temos $3^x$ moedas que podem ser falsas (Se $x = 0$, pare o algoritmo), então, escolhemos $3^{x-1}$ moedas aleatórias para colocar na balança esquerda, e outras $3^{x-1}$ moedas aleatórias para colocar na balança direita. Então, temos 3 possibilidades
  - Se a balança esquerda tiver um peso maior do que a da direita, significa que a moeda falsa está no conjunto de $3^{x-1}$ moedas da direita, e então, vamos para o passo $i + 1$ com essas $3^{x-1}$ moedas
  - Se a balança direita tiver um peso maior do que a esquerda, temos um caso analogo ao de cima
  - Se for o mesmo peso, significa que a moeda falsa está na pilha de $3^{x-1}$ que não foi colocada na balança, então, vamos para o passo $i + 1$ com essa pilha.

Logo, conseguimos um algoritmo que leva $x$ para $\frac{x}{3}$, e o problema acabou.

Problemas desse tipo são bem comuns, em que levamos $x$ para $\frac{x}{2}$ ou $\frac{x}{3}$ por uma sequência de passos.

# §3 Alguns Problemas mais Apimentados

**Exemplo 3.1** (OBM 2022 - P5)

Seja $n$ um inteiro positivo, e defina $S(n)$ para ser o menor inteiro positivo tal que $S(n)$ e $n$ tem a mesma paridade, $S(n) \geq n$ e tais que **não** existam inteiros positivos $k, x_1, x_2, \cdots, x_k$ tais que $x_1 + x_2 + \cdots + x_k = n$ e $x_1^2 + x_2^2 + \cdots x_k^2 = S(n)$. Prove que existe uma constante $c$ tal que $S(n) \geq cn^{\frac{3}{2}}$

Primeiro, veja que $x_1 = x_2 = \cdots = x_n = 1$ é solução, então $S(n) > n$.

A ideia principal do problema vai ser a seguinte: Suponha que temos uma sequência $x_1, x_2, \cdots, x_k$ tal que $x_1 + x_2 + \cdots x_k = n$ e que $x_1^2 + x_2^2 + \cdots x_k^2 = t$, conseguimos construir outra sequência $y_1, y_2, \cdots, y_k$ de modo que $y_1 + y_2 + \cdots y_k = n$ e $y_1^2 + y_2^2 + \cdots + y_k^2 = t + 2$? Enquanto conseguimos construir essa sequência, conseguimos melhorar a cota para $S(n)$, então, vamos tentar fazer isso!

Se temos dois valores iguais $x_i = x_j = v$, então, podemos fazer $x_i = v - 1$ e $x_j = v + 1$, e então

$$\begin{aligned} &x_1^2 + \cdots + (v-1)^2 + \cdots + (v+1)^2 + \cdots + x_k^2 \\ &= x_1^2 + \cdots + v^2 - 2v + 1 + \cdots + v^2 + 2v + 1 + \cdots x_k^2 \\ &= t + 2v + 1 - 2v + 1 = t + 2 \end{aligned}$$

Então, caso tenha dois valores iguas na sequência, conseguimos aumentar ela!! Logo, vamos montar o seguinte algoritmo guloso: Comeqe com $x_1 = x_2 = \cdots = x_k$, então

- Passo $i$: Seja $X$ o maior valor em $x_i$ de modo que ele aparece duas vezes na sequência, sejam eles $x_i$ e $x_j$, e faça $x_i = x_i - 1$ e $x_j = x_j + 1$, e caso $x_i - 1$ for igual a 0, tire-o da sequência. Pare o algoritmo quando todos os números forem diferentes.

Seja $a_1, a_2, \cdots, a_k$ a sequência que resta depois do algoritmo acabar, então, queremos cotar $a_1^2 + a_2^2 + \cdots a_k^2$, pois sabemos que $S(n) \geq a_1^2 + \cdots + a_k^2$. Agora, sabemos que

$$\begin{aligned} n = a_1 + a_2 + \cdots + a_k \geq 1 + 2 + \cdots + k = \frac{k(k+1)}{2} \\ \implies k \leq \sqrt{2n} \end{aligned}$$

Também temos que, por MQ $\geq$ MA, temos que

$$a_1^2 + \cdots + a_k^2 \geq \frac{(a_1 + \cdots + a_k)^2}{k} = \frac{n^2}{k} \geq \frac{n^2}{\sqrt{2n}}$$

Portanto, $S(n) \geq \frac{1}{\sqrt{2}} n^{3/2}$, como queriamos provar.

Esse problema tem duas grandes dificuldades. A primeira é essa interpretação combinatórica do problema, de trocar $x, x$ por $x - 1, x + 1$, mas visto isso, concluir o algoritmo não é difícil, o problema mesmo é ver porque o algoritmo funciona, e para isso temos uma ideia bem legal, que é olhar para os números no final do algoritmo, pois ele tem a propiedade de serem todos diferentes.

Bem, já vimos vários exemplos de algoritmos em combinatória, porém, tal ideia também pode ser aplicada nas outras matérias! Vamos ver uma aplicação em teoria dos números para exemplificar.

**Exemplo 3.2** (Teste Cone-Sul 2017 - T2)

Boris escreve, na lousa, o produto a seguir:

$$1 \cdot 2 \cdot 3 \cdots n$$

Boris pode escolher alguns números, ou todos se ele quiser, e colocar um sinal de exclamação ao lado, transformando tal número em fatorial. Para quais $n > 2$, ele pode colocar sinais de exclamação de tal forma de modo que o produto resultante seja um quadrado perfeito?

Vamos ver alguns casos iniciais:

- $n = 3$: Impossível, pois o fator 3 só pode aparecer uma vez no produto
- $n = 4$: Possível! Faça $1 \cdot 2 \cdots 3 \cdot 4! = 4 \cdot (3!)^2$ que é quadrado perfeito
- $n = 5$: Impossível, pois o fator 5 só pode aparecer uma vez no produto
- $n = 6$: Possível! Colocando um fatorial no 6, temos o número $(5!)^2 \cdot 2 \cdot 3$ no quadro, então, colocado um fatorial no número 4, temos o números $5!^2 \cdot 3!^2$
- $n = 7$: Impossível, pois o fator 7 só pode aparecer uma vez no produto
- $n = 8$: Possível! Colocando um fatorial no 8, temos o número $(7!)^2 \cdot 2^3$ no quadro, então, coloque um fatorial no 3, e temos o número $(7!)^2 \cdot 2^4$
- $n = 9$: Possível! Colocando um fatorial no 9, temos o número $(8!)^2 \cdot 3^2$ no quadro, que é quadrado perfeito.

Bem, com os casos iniciais já temos um avanço claro: Se $n$ é primo, logo não conseguimos fazer essa operação, pois o fator primo $n$ só pode aparecer uma vez. Agora, se $n$ é composto, nós conseguimos montar?

A resposta é sim! Vamos montar o seguinte algoritmo, começando com $n \cdot (n-1)!^2$, ou seja, colocando um fatorial no $n$:

- Passo $i$: Seja $x$ o número atual. Seja $p$ o maior primo que tem expoente impar em $x$, e então, coloque um fatorial no $p+1$. Pare o algoritmo quando não tivermos mais fatores ímpares em $x$.

Vamos mostrar um exemplo antes de provar que funciona:

Faça $n = 12 = 4 \cdot 3$, então, 3 é o maior fator primo impar no número, então coloque um fatorial em 4, e logo, o número $x$ é igual a $12 \cdot 11!^2 \cdot 3!$, e então, o maior fator primo é 2, então, coloque um fatorial no 3, e conseguimos $12 \cdot 11!^2 \cdot 3! \cdot 2!$, que é quadrado perfeito!

Primeiro, se $n$ for primo, o algorimto não começa, pois não poderiam colocar um fatorial em $n+1$. Logo, para provar que o algoritmo sempre funciona, veja que, sendo $p$ o maior fator primo com expoente impar de $x$ no passo $i$, depois de colocarmos um fatorial em $p+1$, o $p$ agora tem expoente par, e então, o maior fator primo de expoente impar diminúi de tamanho! Ou seja, em cada passo diminúi, e no final, vai ficar sem fatores primos de expoentes impares, ou seja, um quadrado perfeito.

Para finalizar, vamos para o problema que fez com que algoritmos começasse a ser estudado a fundo pelos olímpicos.

**Exemplo 3.3** (IMO 2010 - P5)

Seis caixas $B_1, B_2, B_3, B_4, B_5, B_6$ estão colocadas em uma linha. Cada caixa contém exatamente uma moeda. Podemos fazer dois tipos de movimentos:

- **Movimento 1:** Se $B_k$, com $1 \leq k \leq 5$, contém pelo menos uma moeda, você pode remover uma moeda de $B_k$ e colocar duas em $B_{k+1}$.
- **Movimento 2:** Se $B_k$, com $1 \leq k \leq 4$, contém pelo menos uma moeda, você pode remover uma moeda de $B_k$ e trocar a quantidade de moedas de $B_{k+1}, B_{k+2}$ de lugar.

Determine se existe uma sequência finita de operações de modo que as caixas $B_1, B_2, \cdots B_5$ fiquem todas vazias e $B_6$ tenha $2010^{2010^{2010}}$ moedas.

A primeira coisa que tu pensa quando vê esse problema é: "pfff, com certeza a resposta é não". Bem, surpreendentemente é sim. Dito isso, a primeira coisa que podemos ver é a seguinte

**Lema:** Se temos pelo menos $X/4$ moedas em $B_4$, com $4 \mid X$ e nenhuma moeda em $B_5$, $B_6$, então, podemos aplicar a operação 2 em $B_4$ várias vezes, até que tenhamos exatamente $X/4$ moedas em $B_4$, e então aplicar a operação 1 duas vezes, em $B_4, B_5$, então, conseguimos cheagr em $X$ em $B_6$.

Com isso, temos que esse número estranho do enunciado não é importante! Basta chegarmos em um número muito grande de moedas em $B_4$, e manter 0 moedas em $B_5, B_6$, que conseguimos terminar o problema!

Agora, vamos fazer dois algoritmos para aumentar o número de moedas exponencialmente.

- Defina $(x, y, z) \to (x', y', z')$ três caixas consecutivas com tais números de moedas nelas. Então, conseguimos fazer $(a, 0, 0)$ virar $(0, 2^a, 0)$
- Defina $(a, b, c, d) \to (a', b', c', d')$ quatro caixas consecutivas com tais números de moedas nelas. Então, conseguimos fazer $(a, 0, 0, 0)$ virar $(0, P_a, 0, 0)$, onde $P_0 = 1$ e $P_n = 2^{P_{n-1}}$ (Por exemplo, $P_3 = 2^{2^2}$)

Para o primeiro algoritmo, faça $(a, 0, 0) \to (a-1, 2, 0)$ com o primeiro movimento, e então $\to (a-1, 0, 4)$, então, faça $(a-2, 4, 0)$, e então $(a-2, 0, 8)$ e então $(a-3, 8, 0)$, e assim vai, até acabar as moedas na primeira caixa, e temos $(0, 2^a, 0)$.

Agora, para o segundo algoritmo, vamos aplicar o primeiro repetidamente, e para explicar, é mais fácil explicar isso com uma indução.

Vamos provar que conseguimos chegar $(a, 0, 0, 0) \to (a-k, P_k, 0, 0)$, em particular, com $k = a$, conseguimos chegar em $(0, P_a, 0, 0)$. Suponha que estamos em $(a-k, P_k, 0, 0)$, então, aplique o algoritmo acima para fazer $(a-k, 0, 2^{P_k}, 0) = (a-k, 0, P_{k+1}, 0) \to (a-k-1, P_{k+1}, 0, 0)$, como queriamos provar.

Perfeito! Com esses dois algoritmos conseguimos acabar o problema! Primeiro, faça a seguinte sequência de operações:

$$(1,1,1,1,1,1) \to (1,1,1,1,0,3) \to (1,1,1,0,3,0) \to \cdots (0,3,0,0,0,0) \to (0,0,P_3,0,0,0)$$

$$\to (0,0,0,P_{16},0,0)$$

Que não é dificil de provar que $P_{16} > 2010^{2010^{2010}}$, e dai o problema acaba.

# §4 Problemas Extras

**Problema 1:** (Teste Cone-Sul 2021 -T2) Em um país existem um número finito de cidades. Algumas cidades são conectadas por duas rotas de voo, uma em cada direção. Porém, o governo decidiu que, em meses de baixa demanda, uma das direções de cada rota de voo será fechada. Após tal remoção, uma cidade estaria sobrecarregada se tivesse pelo menos dois voos de entrada a mais que voos de saída, ou dois voos de saída a mais que voos de entrada. Mostre que, independentemente de como as cidades estiverem conectadas inicialmente, é possível fazer a remoção de uma das rotas de cada par de cidades conectadas de modo que nenhuma cidade fique sobrecarregada.

**Problema 2:** (USA TST 2015 P2) Um torneio é um grafo direcionado onde, para todo par de vértices não ordenados, existe uma aresta que liga esses dois vértices. Dizemos que uma coloração de arestas é boa se ela colore todas as arestas e não tem três vértices $u, v, w$ tais que $\overrightarrow{uv}$ e $\overrightarrow{vw}$ não são da mesma cor. Para cada $n$, onde $n$ é a quantidade de vértices do grafo, qual é a quantidade mínima de cores que podemos usar para uma coloração ser boa?

**Problema 3:** (IMO 2007 - P3) Em uma competição matemática, alguns competidores são amigos, que é uma relação mútua. Chame um grupo de competidos um clique se cada par deles são amigos. O número de membros num clique é chamado de tamanho. Sabe-se que o tamamho do maior clique é par. Prove que os competidores podem ser divididos em duas salas de modo que o tamanho do maior clique de uma é o mesmo da otura sala.

**Problema 4:** (IMO SL 2001 C4) Um conjunto $\{x, y, z\}$ de inteiros não negativos é chamado de histórico se $x < y < z$ e $z - y, y - x = 1776, 2016$. Mostre que os inteiros não negativos podem ser escritos como um conjunto disjuntos de conjuntos históricos.

**Problema 5:** Dado um grafo $G$ no qual todo vértice tem grau pelo menos $n - 1$, e uma árvore $T$ com $n$ vértices, mostre que existe um subgrafo de $G$ que é isomorfo a $T$.

**Problema 6:** (Rússia) Em um grid de $2 \times n$ com números reais positivos colocados neles, de modo que as somas de cada uma das colunas é 1. Prove que é possível selecionar um número em cada coluna de modo que a soma dos números selecionados em cada linha é menor ou igual que $\frac{n+1}{4}$.

**Problema 7:** (IMO SL 2013 C3) Um físico maluco descobre um noco tipo de partícula que ele chama de imon, depois de ter aparecido misteriosamente no laboratório. Alguns pares de imons no laboratório podem se ligar, e cada imon pode participar em vários ligamentos. O físico descobriu uma maneira de fazer essas duas operações com as partículas, uma de cada vez:

1. Se algum imon está ligado com um número impar de imons no laboratório, então o físico pode destrui-lo

2. Em qualquer momento, ele pode pegar todos os imons no laboratorio e dobrar eles, criando uma cópia $I'$ de cada imon $I$. Durante esse procedimento, $v'$ e $u'$ estão ligados se, e somente se, $v$ está ligado em $u$. Além disso, $v'$ está ligado em $v$.

Prove que o físico pode aplicar uma sequência de operações resultando numa família de imons, onde não há dois deles ligados.

**Problema 8:** (Codeforces # 1270 G) Você tem uma sequência de $n$ inteiros, $a_1, a_2, \cdots a_n$ tal que $i - n \leq a_i \leq i - 1$ para todo $1 \leq i \leq n$. Prove que tem uma subsequência não vazia dos elementos com soma igual a 0.

# §5 Referências

1. Olympiad Combinatorics - Pranav A. Sriram
2. Algorithms - Howard Halim - Canada Training IMO 2021
3. Mathlinks.ro
4. Treinamento Cone Sul