Brazilian ICPC Summer School 2020

Jhúlia Graziella
jhuliagraziella@gmail.com

Upsolving - Final Brasileira 2019

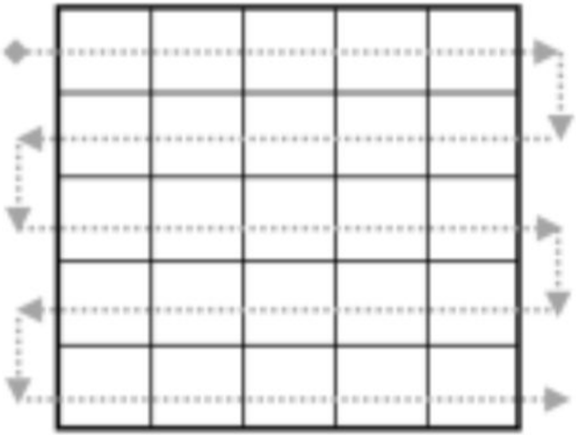
M - Mountain Ranges (29 ACs)
E - Eggfruit cake (19 ACs)
I - Improve SPAM (18 ACs)
K - Know your Aliens (15 ACs)
L - Leverage MDT (12 ACs)
G - Gluing Pictures (6 ACs)

L - Leverage MDT

Binary Search the answer

Entrada: grid com a qualidade do terreno.

Saída: maior área quadrada de terrenos bons.



K - Know your Aliens

Teoria dos números

Entrada: string de caracteres 'A' e 'H', representando a identidade de cada cidadão da população (alien ou humano).

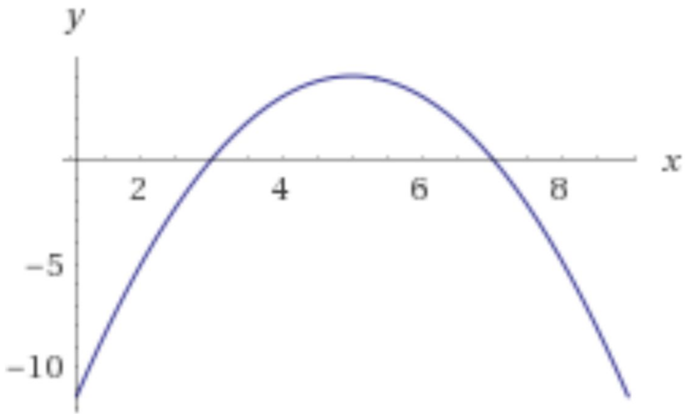
Saída: coeficientes do polinômio que codifica a população corretamente.

Cada par **`2i`** representa um cidadão, e a string **AHHA** representa que os cidadãos **2** e **8** são aliens, enquanto os cidadãos **4** e **6** são humanos.

Encontre o polinômio **`f(x)`** com menor grau possível, tal que:

```
f(2) < 0
f(4) > 0
f(6) > 0
f(8) < 0
```

```
f(x) = -x² + 10x - 21
```



Observações:

1) As raízes devem ser inteiras: nós temos as raízes!

2) O valor absoluto dos coeficientes é menor que **`2^63`**.

Teorema do resto: o resto **r** que resulta da divisão de um polinômio **`p(x)`** por **`(x-a)`**, é igual a **`p(a)`**.

Demonstração: **`p(x)=q(x)*(x-a)+r`** $\Longrightarrow$ **`p(a)=q(x)*(a-a)+r`**

Recorrências lineares

Qualquer termo pode ser calculado em função dos antecessores. Por exemplo, a sequência de Fibonacci é definida por **$f(n) = f(n-1) + f(n-2)$**, para **$n \geq 2$**.

Recorrências lineares

Qualquer termo pode ser calculado em função dos antecessores. Por exemplo, a sequência de Fibonacci é definida por **f(n) = f(n-1) + f(n-2)**, para **n ≥ 2**.

```
[f(0) f(1)] * |0 1| = [f(1) f(2)]
              |1 1|
```

Ônibus (https://www.urionlinejudge.com.br/judge/pt/problems/view/1474)

Temos **K** cores possíveis pra microônibus (tamanho 5) e **L** cores disponíveis pra ônibus (tamanho 10). De quantas maneiras podemos preencher uma fila de tamanho **N, N | 5** e **N < 10^15**?

Ônibus (https://www.urionlinejudge.com.br/judge/pt/problems/view/1474)

Temos **K** cores possíveis pra microônibus (tamanho 5) e **L** cores disponíveis pra ônibus (tamanho 10). De quantas maneiras podemos preencher uma fila de tamanho **N**, **$N \mid 5$** e **$N < 10^{15}$**?

```
f(0) = 1                          |f(0)| |0 l| | f(5)|
f(5) = k                       => |f(5)|*|1 k|=|f(10)|
f(10) = k*f(5) + l*f(0)
```

Runner's Problem (https://codeforces.com/contest/954/problem/F)

- Grid 3 x M (M ≤10^18)
- N (N ≤ 10^4) obstáculos ocupando uma linha no range **[l, r]**
- De quantas maneiras é possível chegar da posição **(2, 1)** até **(2, m)**?

Runner's Problem (https://codeforces.com/contest/954/problem/F)

- Dividir em blocos de acordo com os obstáculos.
- A resposta de cada bloco pode ser encontrada por exponenciação de matriz.

```
|a|    |1 1 0|    |a+b  |
|b| *  |1 1 1| =  |a+b+c|
|c|    |0 1 1|    |  b+c|
```

Prefix Sums (https://codeforces.com/contest/837/problem/F)

- **x** é um vetor de **n** números naturais **{x_1, x_2, ..., x_n}**
- A função **P(x)** retorna um vetor em que cada elemento **i** corresponde a soma dos **i** primeiros elementos de **x**, e.g. **P(x) = {x_1, x_1+x_2, $x_1+x_2+x_3$, ..., $x_1+x_2+...+x_n$}**
- Quantas vezes precisamos aplicar a função **P** em **x** até ter algum elemento maior ou igual a **k**?
- Limites: **n ≤ 2*10^5; x_i ≤ 10^9; k ≤ 10^18**

Prefix Sums (https://codeforces.com/contest/837/problem/F)

- Solução iterativa: enquanto **P[n-1] < k**, calcule novo **P**.
    - Pior caso: x = {1, 0, 0, ...}
    - Complexidade = O(n * ans)
- Solução com busca binária + exponenciação:
    - Chute **ans**, verifique se **x * mat**$^{\mathbf{ans}}$ é solução
    - Complexidade = O(log(k) * log(k) * n^3)
        - (busca binária, exponenciação, multiplicação de matriz)

Prefix Sums (https://codeforces.com/contest/837/problem/F)

- Solução iterativa: enquanto **P[n-1] < k**, calcule novo **P**.
    - Pior caso: x = {1, 0, 0, ...}
    - Complexidade = O(n * ans) → **funciona pra n ≥30**
- Solução com busca binária + exponenciação:
    - Chute **ans**, verifique se **x * mat**^{ans} é solução
    - Complexidade = O(log(k) * log(k) * n^3) → **funciona pra n < 30**
        - (busca binária, exponenciação, multiplicação de matriz)

Congruência modular

Pequeno teorema de Fermat: se **p** é primo e não divide **a**, então $\mathbf{a^{(p-1)} \equiv 1 \pmod{p}}$.

Teorema do Totient de Euler: se **a** e **n** são inteiros positivos primos entre si, então $\mathbf{a^{\varphi(n)} \equiv 1 \pmod{n}}$.

Inverso multiplicativo: o inverso multiplicativo de um número **n** módulo **m** é um número **a** tal que $\mathbf{n*a \equiv 1 \pmod{m}}$, denotado por $\mathbf{a = n^{-1}}$.

Congruência Modular

Teorema Chinês do Resto: para quaisquer $\mathbf{a}$, $\mathbf{b}$, $\mathbf{n_1}$ e $\mathbf{n_2}$, onde $\mathbf{n_1}$ e $\mathbf{n_2}$ são coprimos, existe um único $\mathbf{x \pmod{n_1 n_2}}$ onde $\mathbf{x \equiv a \pmod{n_1}}$ e $\mathbf{x \equiv b \pmod{n_2}}$.

Além disso, essa solução pode ser calculada por:

$$\mathbf{x = a * n_2^{-1} * n_2 + b * n_1^{-1} * n_1 \pmod{n_1 n_2}}$$

Isso se generaliza para qualquer número de equações.

Congruence equation (https://codeforces.com/contest/919/problem/E)

Quantos inteiros positivos **n ($n \le x$)** satisfazem a equação **$n * a^n \equiv b \pmod{p}$**, onde **$a$, b** e **p** são constantes.

Limites: **$x \le 10^{12}$**, **$p \le 10^6$**.

Congruence equation (https://codeforces.com/contest/919/problem/E)

Limites: **x <= 10^12, p <=10^6.**

Podemos calcular para cada **r ∈ [0, p-2]**.

$n * a^n \equiv b \pmod{p} \quad \Rightarrow n \equiv b * (a^r)^{-1} \pmod{p}$

$n * a^{n-(p-1)} * a^{p-1} \equiv b \pmod{p} \quad \Rightarrow n \equiv r \pmod{p-1}$

$n * a^{n \bmod (p-1)} * 1 \equiv b \pmod{p} \quad \exists \text{ sol } \bmod(p*(p-1))$

Contagem e probabilidade

Equacoes Diofantinas:

https://www.urionlinejudge.com.br/judge/pt/problems/view/1381

- Quantas soluções existem pra equação $x_1 + x_2 + x_3 = n$?
- Quantas soluções existem pra equação $x_1 + x_2 + x_3 = n$, onde $x_2 > 3$?
- Quantas soluções existem pra equação $x_1 + x_2 + x_3 = n$, onde $x_1 < 2$?

Contagem e probabilidade

Princípio da inclusão-exclusão é um método de contagem que generaliza como contar os elementos que pertencem a união de vários conjuntos não necessariamente disjuntos, expressado como:

$|A \cup B| = |A| + |B| - |A \cap B|$

Lei de Morgan: $\overline{X \cup Y} \leftrightarrow \overline{X} \cap \overline{Y}.$

$\overline{X \cap Y} \leftrightarrow \overline{X} \cup \overline{Y}$

Devu and Flowers (https://codeforces.com/contest/451/problem/E)

De quantas maneiras diferentes podemos escolher exatamente **s (s < 10^14)** flores, tendo no máximo **f_i (f < 10^12)** flores do tipo **i**, e **n < 20** tipos de flores?

Devu and Flowers (https://codeforces.com/contest/451/problem/E)

De quantas maneiras diferentes podemos escolher exatamente **s** **(s < 10^14)** flores, tendo no máximo $\mathbf{f_i}$ **(f < 10^12)** flores do tipo **i**, e **n < 20** tipos de flores?

Seja $\mathbf{x_i}$ a quantidade de flores do tipo **i**. Então temos:

$$\sum x_i = s, \text{ onde } x_i \leq f_i$$

Devu and Flowers (https://codeforces.com/contest/451/problem/E)

Pra **n = 2**:

A = soluções com $\mathbf{x_1 \geq f_1}$

B = soluções com $\mathbf{x_2 \geq f_2}$

A U B são as soluções que **NÃO** satisfazem.

Resposta = todas as soluções - **A U B**

Devu and Flowers (https://codeforces.com/contest/451/problem/E)

A = soluções com $x_1 \geq f_1$ => C $(s - f_1 + n - 1, n - 1)$

B = soluções com $x_2 \geq f_2$ => C $(s - f_2 + n - 1, n - 1)$

|A ∩ B| = soluções com $x_1 \geq f_1$ e $x_2 \geq f_2$ => C $(s - f_1 - f_2 + n - 1, n - 1)$

|A U B| = soluções com $x_1 \geq f_1$ ou $x_2 \geq f_2$ => $|A| + |B| - |A \cap B|$

U = todas as soluções => $C(s + n - 1, n - 1)$

Resposta = U - |A U B|

Winter is here (https://codeforces.com/problemset/problem/839/D)

Tem um exército com **$n < 2 * 10^5$** soldados com força **$a_i < 10^6$**. Podemos formar um clã de soldados se o **mdc** de suas forças for diferente de **1**. Um grupo de **k** soldados **$(i_1, i_2, ..., i_k)$** tem força **$k*mdc(a_{i1}, ..., a_{ik})$**. A força do exército é a soma das forças de cada clã.

Winter is here (https://codeforces.com/problemset/problem/839/D)

m(i) = quantos soldados tem força múltipla de **i.**

s(i) = soma das respostas para todos os clãs tal que **mdc(clã) = i**.

Isso inclui os clãs de qualquer tamanho.

$s(i) = m(i) * 2^{m(i)-1} - s(2*i) - s(3*i)...$