14º Contest

Laboratório de Programação Competitiva I

Pedro Henrique Paiola

Rene Pegoraro

Wilson M Yonezawa

A – T-Primes (CodeForces 230B)

- **Objetivo:** Determinar se um número é T-primo (possui exatamente 3 divisores distintos)
- $1 \leq n \leq 10^5$
- $1 \leq x_i \leq 10^{12}$

A – T-Primes (CodeForces 230B)

- Um T-Primo n têm 3 divisores: 1, x e n.
- x deve ser um número primo, caso contrário os divisores de x também seriam divisores de n.

A – T-Primes (CodeForces 230B)

- Um T-Primo n têm 3 divisores: 1, x e n.
- x deve ser um número primo, caso contrário os divisores de x também seriam divisores de n.
- x também deve ser exatamente $\sqrt{n}$
  - Se x é divisor de n, então $\frac{n}{x}$ também é. Sendo assim, a única forma de n continuar com apenas 3 divisores é se $x = \frac{n}{x}$, ou seja, $x = \sqrt{n}$

A – T-Primes (CodeForces 230B)

$n = 2^3 . 7^2 . 11^4$

- Um T-Primo n têm 3 divisores: 1, x e n.
- x deve ser um ~~número primo~~, caso contrário os divisores de x também seriam divisores de n.
- x também deve ser exatamente $\sqrt{n}$
  - Se x é divisor de n, então $\frac{n}{x}$ também é. Sendo assim, a única forma de n continuar com apenas 3 divisores é se $x = \frac{n}{x}$, ou seja, $x = \sqrt{n}$
- Se $n = x^2$, então é garantido que o número só possui três divisores. Afinal, x é um número primo, logo $n = x^2$ é exatamente a fatoração de n.

A – T-Primes (CodeForces 230B)

- Um T-Primo n têm 3 divisores: 1, x e n.
- x deve ser um número primo, caso contrário os divisores de x também seriam divisores de n.
- x também deve ser exatamente $\sqrt{n}$
  - Se x é divisor de n, então $\frac{n}{x}$ também é. Sendo assim, a única forma de n continuar com apenas 3 divisores é se $x = \frac{n}{x}$, ou seja, $x = \sqrt{n}$
- Se $n = x^2$, então é garantido que o número só possui três divisores. Afinal, x é um número primo, logo $n = x^2$ é exatamente a fatoração de n.
- Resumindo: n deve ser um quadrado perfeito e $\sqrt{n}$ deve ser um número primo.

B – DDF (UVA 547)

- Uma sequência $x_1, x_2, \ldots, x_n$ é uma DDF se $\underline{x_{i+1}}$ é a soma dos dígitos de todos os fatores de x_i

DDF =

```
17, 9, 13, 5, 6, ...
positive factor of 17 = 1, 17
1 + (1 + 7) = 9
positive factor of 9 = 1, 3, 9
1 + 3 + 9 = 13
positive factor of 13 = 1, 13
1 + (1 + 3) = 5
positive factor of 5 = 1, 5
1 + 5 = 6
```

B – DDF (UVA 547)

- **Objetivo:** dado um intervalo definido por m, n encontrar a maior DDF possível que se inicie com um número entre $[\min(m,n), \max(m,n)]$
- $m, n \leq 3000$

B – DDF (UVA 547)

- **Solução**
  - 1) Para cada número de 1 a 3000 vamos descobrir qual o próximo termo da DDF para ele, ou seja, qual a soma dos dígitos dos seus fatores.
  - Para fazer isso, vamos nos basear no algoritmo do Crivo, percorrendo todos os números de 1 a 3000, calcular a soma dos seus dígitos e adicionar esse valor para todos os seus múltiplos

```
for(int x = 1; x < MAXN; x++){
        somaDig = somaDigitos(x);
        for(int y = x; y < MAXN; y += x){
            prox[y] += somaDig;
        }
}
```

Departamento de Computação

Programação Competitiva unesp

B – DDF (UVA 547)

- **Solução**
  - 2) Para determinar o tamanho de cada DDF, podemos realizar uma busca com memorização, partindo da seguinte recorrência:

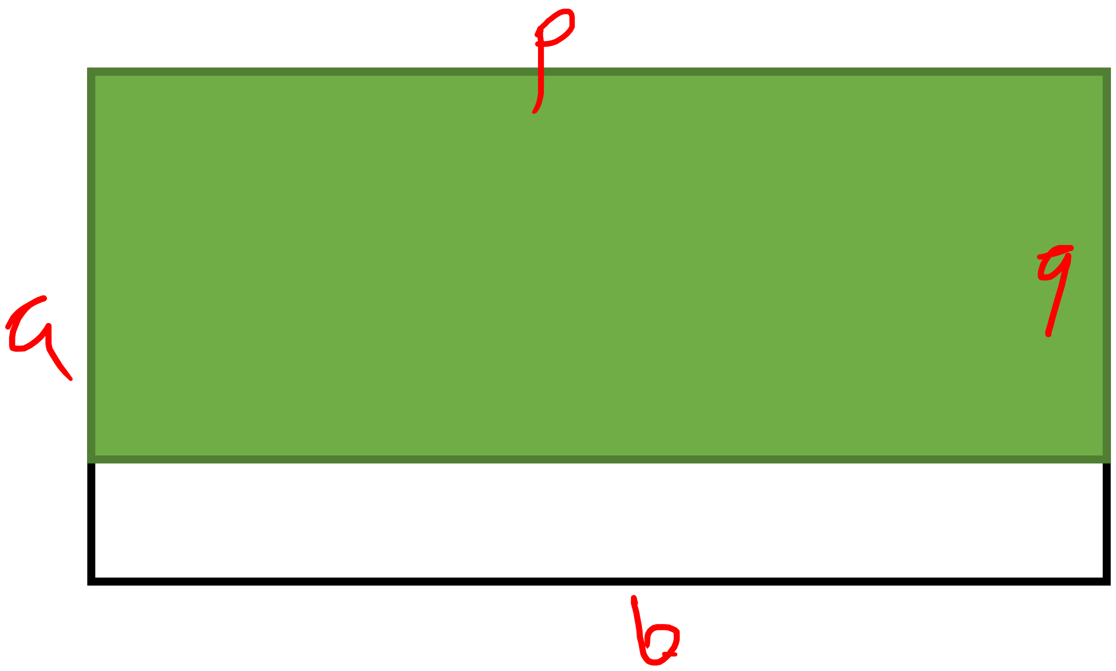
$$tam(x) = \begin{cases} 1 & \text{se } x = prox(x) \\ 1 + tam(prox(x)) & c.\,c \end{cases}$$

B – DDF (UVA 547)

- **Solução**
  - 3) Por fim, com os vetores $prox$ e tam já processados, determinar qual a maior DDF em um certo intervalo pode ser feito por força-bruta mesmo, sem nenhum problema.
  - OBS: os vetores $prox$ e tam podem ser pré-processados por um programa a parte e inseridos já prontos direto no código (solução *off-line*).
    - Neste caso, isso não traz um ganho de desempenho muito grande, pois não eram processamentos muito pesados.
    - Mas ainda assim, poderia ser útil justamente caso não conseguíssemos modelar uma solução suficientemente eficiente.

C – Monitor (CodeForces 16C)

- **Objetivo:** quais as dimensões do maior retângulo possível dentro da área $a\ x\ b$ que possua uma proporção $x: y$



$$\frac{p}{q} = \frac{x}{y}$$

C – Monitor (CodeForces 16C)

- **Solução 1:** sem pensar muito na parte matemática, pode-se fazer uma busca binária.
  - Buscando a maior dimensão possível $p \times q$, em que $p \leq a$, $q \leq b$ e $\frac{p}{q} = \frac{x}{y}$
  - Busca binária em relação à p no intervalo $[0, a]$. O valor q é dependente de p ($q = p.\frac{y}{x}$). Se $q > b$, diminuímos p, senão aumentamos, e continuamos a busca binária até achar o melhor p possível.

C – Monitor (CodeForces 16C)

- **Solução 2:** sem busca, solução direta.
  1. Simplificamos a razão $\frac{x}{y}$:
     1. $x \leftarrow \frac{x}{\gcd(x,y)}$
     2. $y \leftarrow \frac{y}{\gcd(x,y)}$
  2. Com isso podemos dizer que a proporção $\frac{p}{q}$ almejada é um múltiplo de $\frac{x}{y}$, pois este está na forma mais simplificada possível. Ou seja, $p = x.m$ e $q = y.m$. Também sabemos que $p \leq a$ e $q \leq b$. Logo $x.m \leq a$ e $y.m \leq b$.
  3. Como queremos que m tenha o maior valor possível, vamos calcular m considerando as igualdades $x.m = a$ e $y.m = b$, e ficar com o menor valor (para obedecer ambas as restrições)

D – Bishops (SPOJ BISHOPS)

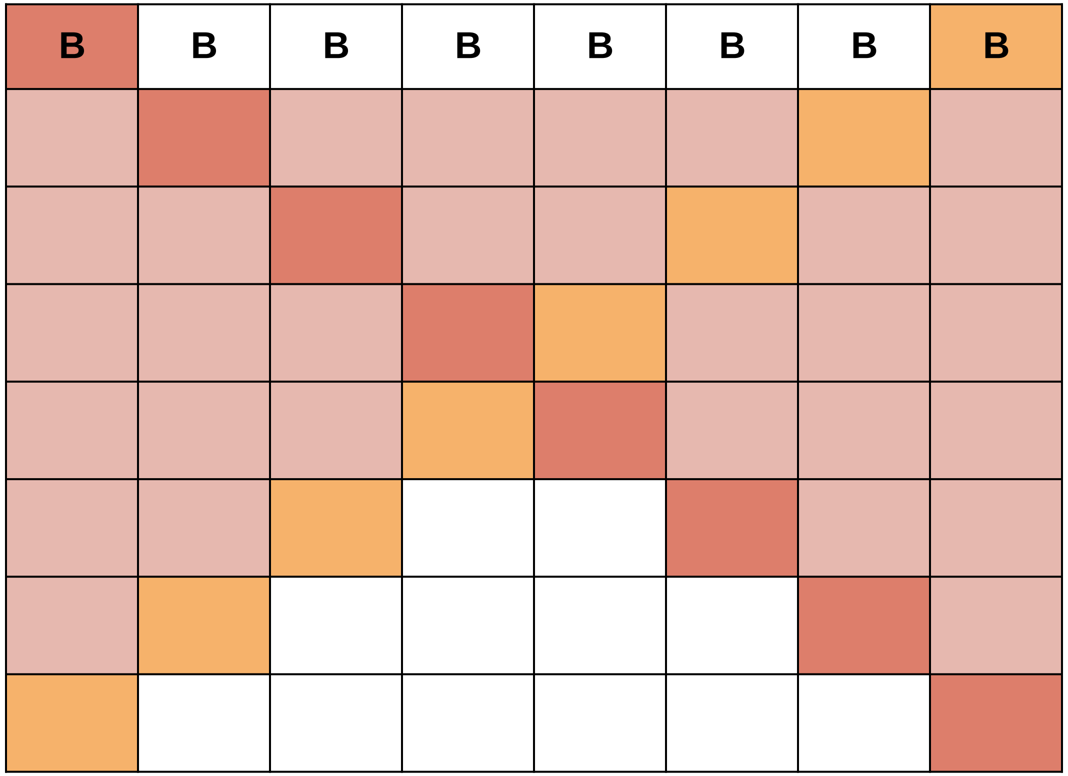
- **Objetivo:** determinar o número máximo de bispos que podem ser posicionados em um tabuleiro de tamanho $n \times n$.
- $1 \leq n \leq 10^{100}$

D – Bishops (SPOJ BISHOPS)

- Perceba que n pode assumir um valor muito grande, extrapolando o valor máximo de uma variável do tipo *long long int*. Logo:
  - Teremos que tratar números desta proporção de alguma forma, implementando uma classe BigInteger em C++, ou migrando para outra linguagem como Java ou Python.
  - Uma busca exaustiva é totalmente inviável pelo tamanho do problema. Logo deve haver algum tipo de estratégia gulosa para posicionar os bispos de forma ótima.

D – Bishops (SPOJ BISHOPS)

- Estratégia
  1. Preencher toda a primeira linha com bispos



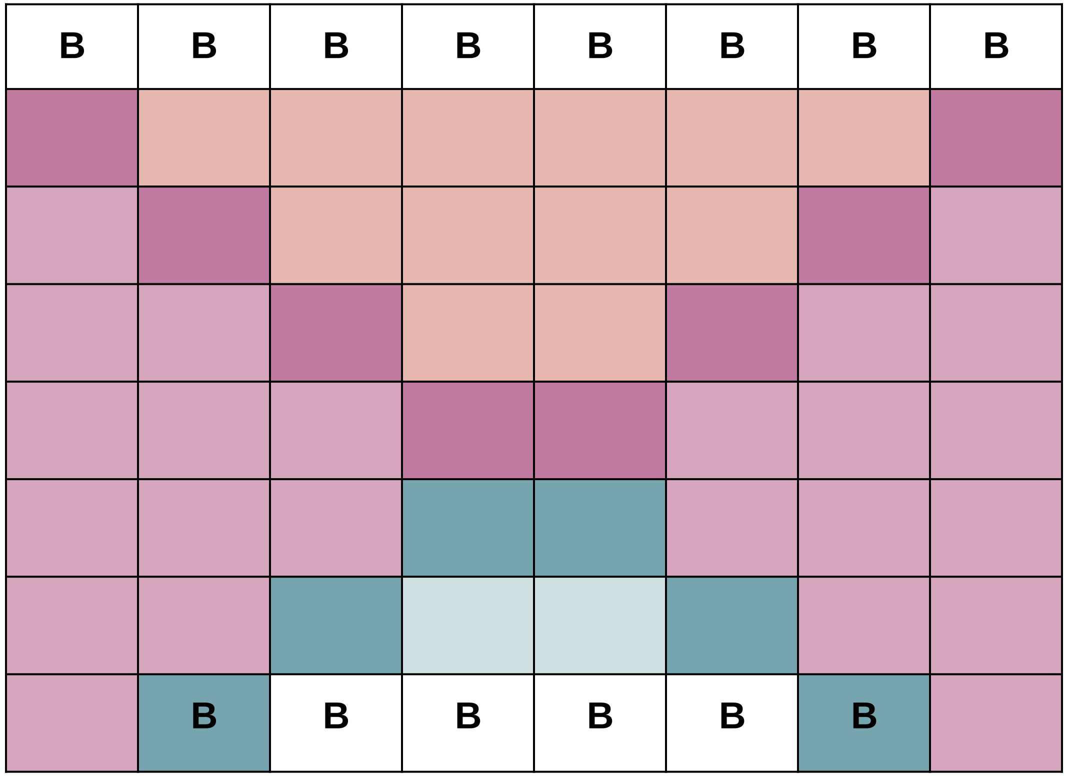
D – Bishops (SPOJ BISHOPS)

- Estratégia
    1. Preencher toda a primeira linha com bispos
    2. Preencher toda a última linha, com exceção das extremidades

B	B	B	B	B	B	B	B
	B	B	B	B	B	B	

D – Bishops (SPOJ BISHOPS)

- Solução: $2n - 2$
  - Caso específico: $n = 1$



n

$n-2$

E – Get AC in one go (CodeChef COPR16G)

- **Problema:** dado dois valores de moedas a e b, determinar o menor valor n tal que todos os valores $\geq n$ possam ser trocados utilizando as moedas a e b. Caso esse número não existe, imprimir -1.

E – Get AC in one go (CodeChef COPR16G)

- Em primeiro lugar, vamos buscar uma forma de determinar se há solução ou não.
- Se houver solução, então existe n para que sejam válidas as equações:

$$ax + by = \cancel{n}$$
$$au + bv = \cancel{n} + 1$$

- Subtraindo estas equações chegamos em outra equação diofantina

$$ar + bs = 1$$

- E esta é uma eq. diofantina que possui solução sse $\gcd(a, b) = 1$

E – Get AC in one go (CodeChef COPR16G)

- Agora, se o problema possui solução, então precisamos determina-lá.
- Para isso iremos utilizar o Teorema do *Chicken McNugget* que diz que dados dois números a e b coprimos, o maior número n que não pode ser escrito na forma $ax + by$ é $ab - a - b$.
  - https://artofproblemsolving.com/wiki/index.php/Chicken_McNugget_Theorem
- Logo, nossa resposta é $ab - a - b + 1$