Fatec São Paulo · Centro Paula Souza · Governo de São Paulo

Números Primos

Conceitos & Algoritmos

URI Online Judge — Problems & Contests · Maratona de Programação InterFatecs · Maratona de Programação

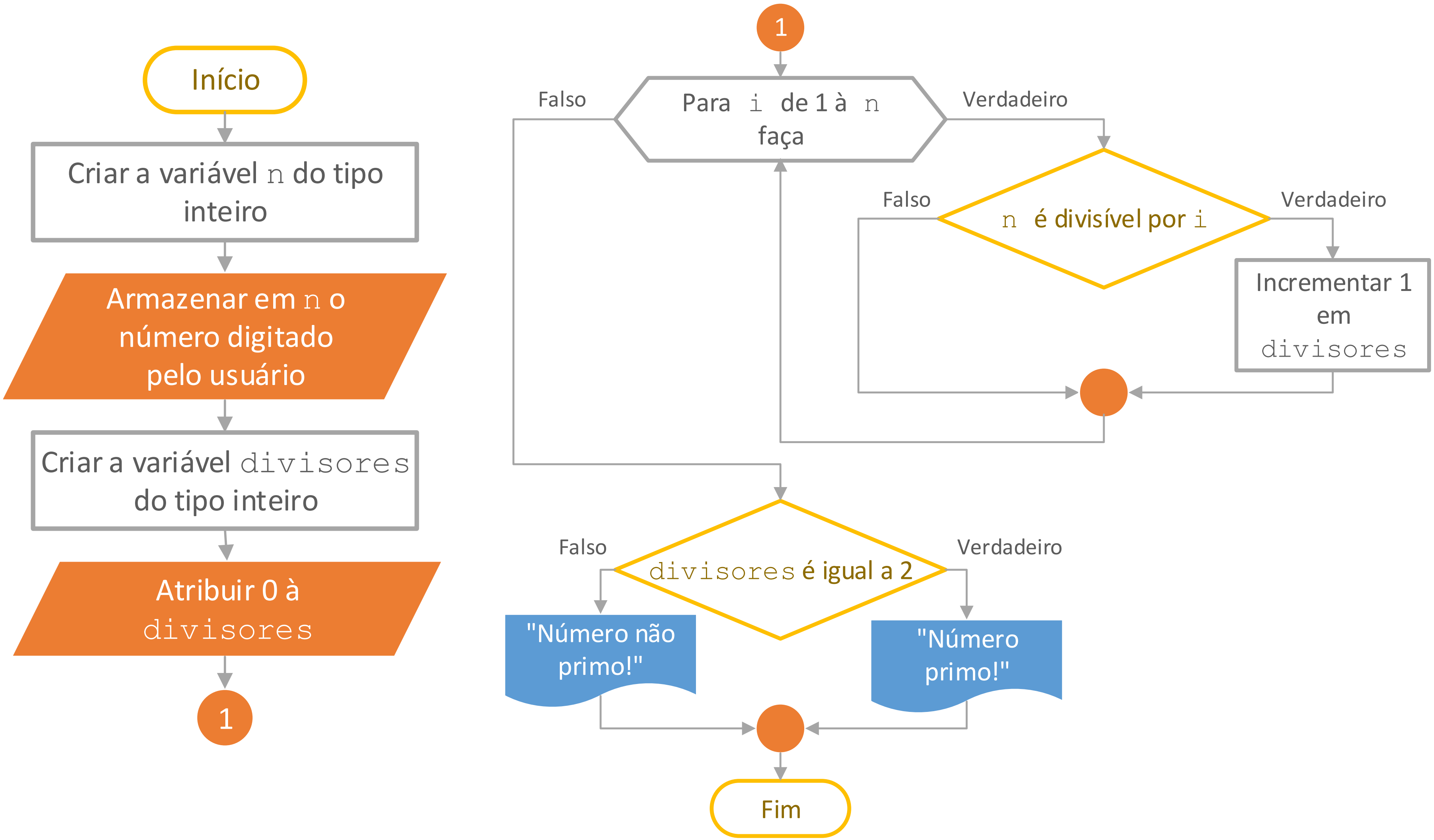
Lucio Nunes de Lira • Grace Anne Pontes Borges

O QUE É?

Simplificadamente, um **número primo** é um número natural que tem exatamente **dois divisores naturais**: o número 1 e ele próprio.

Exemplos: 2, 3, 5, 7, 11, 13, 17, 19...

FLUXOGRAMA


Início
Criar a variável n do tipo inteiro
Armazenar em n o número digitado pelo usuário
Criar a variável divisores do tipo inteiro
Atribuir 0 à divisores
1
1
Para i de 1 à n faça
Falso
Verdadeiro
n é divisível por i
Falso
Verdadeiro
Incrementar 1 em divisores
divisores é igual a 2
Falso
Verdadeiro
"Número não primo!"
"Número primo!"
Fim

```python
# Conta os divisores entre 1 e n.
def primo_1(n):
    divisores = 0
    for i in range(1, n + 1):
        if n % i == 0:
            divisores += 1
    if divisores == 2:
        return True
    else:
        return False
```

1ª VERSÃO

HORA DO

```python
# Conta os divisores entre 1 e n.
def primo_2(n):
    divisores = 0
    for i in range(1, n + 1):
        if n % i == 0:
            divisores += 1
    return divisores == 2
```

2ª VERSÃO

HORA DO

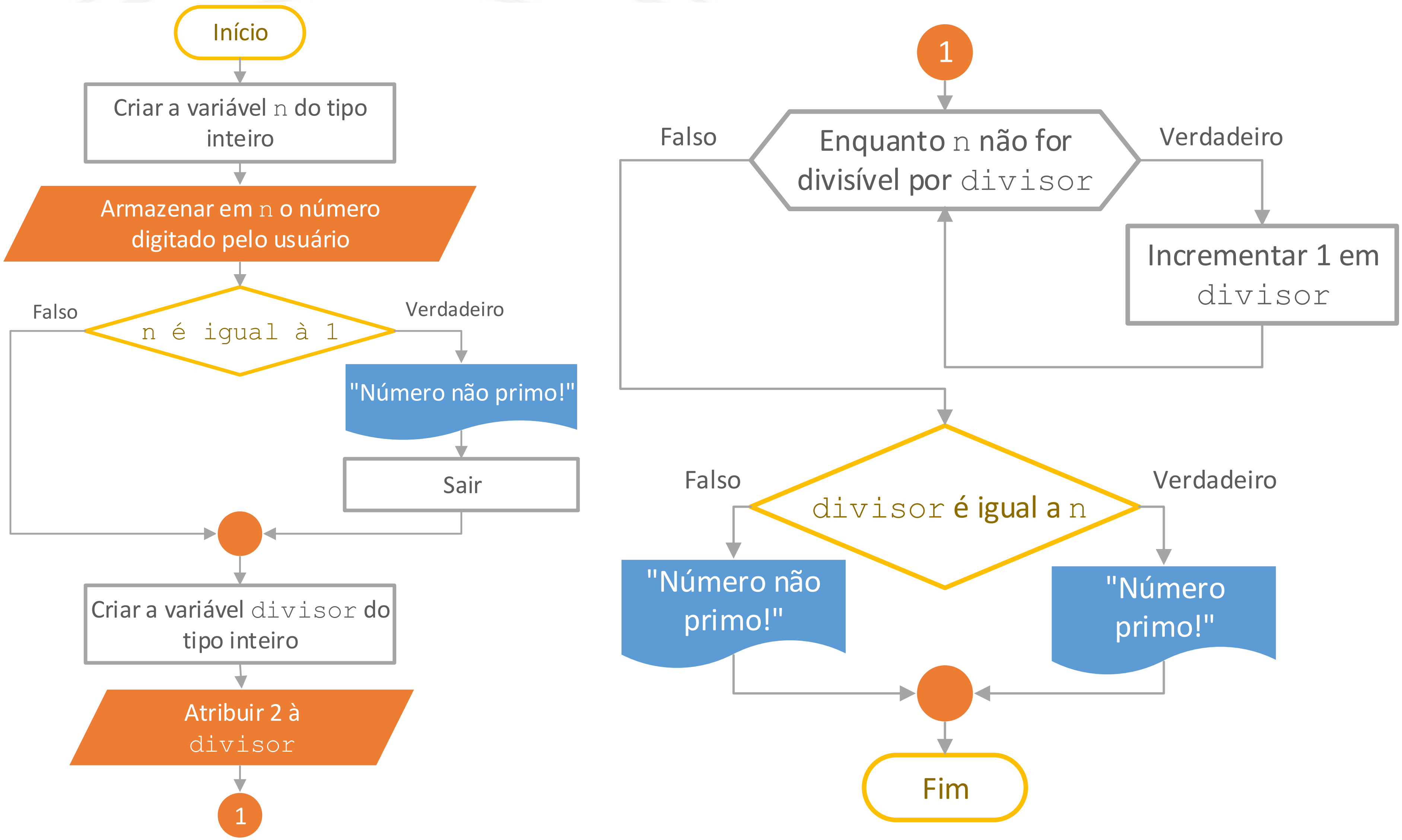
```python
# Conta os divisores entre 2 e n-1.
def primo_3(n):
    divisores = 0
    for i in range(2, n):
        if n % i == 0:
            divisores += 1
    return n > 1 and divisores == 0
```

3ª VERSÃO

HORA DO

UPGRADE

Fluxograma



Início
Criar a variável n do tipo inteiro
Armazenar em n o número digitado pelo usuário
n é igual à 1
Falso
Verdadeiro
"Número não primo!"
Sair
Criar a variável divisor do tipo inteiro
Atribuir 2 à divisor
1
1
Enquanto n não for divisível por divisor
Falso
Verdadeiro
Incrementar 1 em divisor
divisor é igual a n
Falso
Verdadeiro
"Número não primo!"
"Número primo!"
Fim

```python
# Termina as tentativas de divisão na
# primeira ocorrência de uma divisão
# com resto zero.

def primo_4(n):
    if n == 1: return False
    divisor = 2
    while n % divisor != 0:
        divisor += 1
    return divisor == n
```

4ª VERSÃO

HORA DO

UPGRADE

```python
# Testa apenas divisores ímpares.
def primo_5(n):
    if n == 1: return False
    if n % 2 == 0: return n == 2
    divisor = 3
    while n % divisor != 0:
        divisor += 2
    return divisor == n
```

5ª VERSÃO

HORA DO

UPGRADE

- O quociente da **divisão inteira** de um número natural n $(n > 0)$ por um natural m $(m > 0)$ é 1 quando $\left\lfloor \frac{n}{2} \right\rfloor < m \leq n$.
- O quociente da **divisão inteira** de um número natural n $(n > 0)$ por um natural m $(m > 0)$ é 2 quando $\left\lfloor \frac{n}{3} \right\rfloor < m \leq \left\lfloor \frac{n}{2} \right\rfloor$.
- Sendo **mais direto**: não existe um divisor, que gere resto zero, acima da metade de um número n que não seja o próprio n.

```python
# Testa divisores até a metade de n.
def primo_6(n):
    if n % 2 == 0: return n == 2
    divisor = 3
    metade = n // 2
    while divisor <= metade and n % divisor != 0:
        divisor += 2
    return n > 1 and divisor > metade
```

6ª VERSÃO

HORA DO

UPGRADE

- Um número natural é **composto** quando tem mais de dois divisores naturais **distintos**.

- Todo número composto pode ser **decomposto** em um produto de **pelo menos** dois fatores **primos** (não necessariamente distintos).

- Exemplos: **$21 = 3 \times 7$** | **$27 = 3 \times 3 \times 3$** | **$121 = 11 \times 11$**

- Seja $\boldsymbol{n}$ um número natural. Note as **características**:

$$\boldsymbol{n} = \sqrt{\boldsymbol{n}} \times \sqrt{\boldsymbol{n}}$$

Suponha $\boldsymbol{n} = \boldsymbol{a} \times \boldsymbol{b}$

se $\boldsymbol{a} = \sqrt{\boldsymbol{n}}$, então $\boldsymbol{b} = \sqrt{\boldsymbol{n}}$

se $\boldsymbol{a} > \sqrt{\boldsymbol{n}}$, então $\boldsymbol{b} < \sqrt{\boldsymbol{n}}$

se $\boldsymbol{a} < \sqrt{\boldsymbol{n}}$, então $\boldsymbol{b} > \sqrt{\boldsymbol{n}}$

- Se n for um número composto, pode, como citado antes, ser decomposto por no mínimo **dois fatores** primos.

- Portanto, **um dos fatores** deve ser menor ou igual a $\sqrt{n}$. Logo, n tem pelo menos um divisor $\leq \sqrt{n}$.

É necessário **importar** algumas funções para a próxima atualização do algoritmo:

```
# Importação das funções necessárias
from math import sqrt, floor
```

A importação precisa ser feita apenas **uma vez** e antes da execução da função.

```python
# Testa divisores até a raiz de n.
def primo_7(n):
    if n % 2 == 0: return n == 2
    divisor = 3
    raiz = floor(sqrt(n))
    while divisor <= raiz and n % divisor != 0:
        divisor += 2
    return n > 1 and divisor > raiz
```

7ª VERSÃO

VAMOS EXERCITAR!

URI Online Judge | 1165

Número Primo

Adaptado por Neilor Tonin, URI Brasil

Timelimit: 1

1165

Na matemática, um Número Primo é aquele que pode ser dividido somente por 1 (um) e por ele mesmo. Por exemplo, o número 7 é primo, pois pode ser dividido apenas pelo número 1 e pelo número 7.

Entrada

A entrada contém vários casos de teste. A primeira linha da entrada contém um inteiro **N** ($1 \leq N \leq 100$), indicando o número de casos de teste da entrada. Cada uma das **N** linhas seguintes contém um valor inteiro **X** ($1 < X \leq 10^7$), que pode ser ou não, um número primo.

Saída

Para cada caso de teste de entrada, imprima a mensagem "**X** eh primo" ou "**X** nao eh primo", de acordo com a especificação fornecida.

Exemplo de Entrada	Exemplo de Saída
3 8 51 7	8 nao eh primo 51 nao eh primo 7 eh primo

Crivo de Eratóstenes

	2	3	4	5
6	7	8	9	10
11	12	13	14	15
16	17	18	19	20
21	22	23	24	25

- O **Crivo de Eratóstenes** é um algoritmo que permite encontrar todos os números primos até um valor **limite**.

- Foi criado pelo bibliotecário-chefe da Biblioteca de Alexandria, **Eratóstenes** (285-194 a.C.).

Os passos do **algoritmo** são:

1) Determinar o valor *limite*;
2) Criar uma lista com números naturais de 2 até *limite*;
3) Selecionar o primeiro número da lista. Ele é primo;
4) Remover da lista todos os múltiplos desse primo;
5) Selecionar o próximo número da lista. Ele é primo;
6) Repetir os passos 4 e 5 até ultrapassar a raiz do *limite*;
7) Os valores não removidos são **números primos**.

Simulação

LIMITE: 25 | RAIZ: 5

	2	3	4	5
6	7	8	9	10
11	12	13	14	15
16	17	18	19	20
21	22	23	24	25

Simulação

LIMITE: 25 | RAIZ: 5

	2	3	4	5
6	7	8	9	10
11	12	13	14	15
16	17	18	19	20
21	22	23	24	25

SIMULAÇÃO

LIMITE: 25 | RAIZ: 5

	2	3	4	5
6	7	8	9	10
11	12	13	14	15
16	17	18	19	20
21	22	23	24	25

SIMULAÇÃO

LIMITE: 25 | RAIZ: 5

	2	3	4	5
6	7	8	9	10
11	12	13	14	15
16	17	18	19	20
21	22	23	24	25

Simulação

LIMITE: 25 | RAIZ: 5

	2	3	4	5
6	7	8	9	10
11	12	13	14	15
16	17	18	19	20
21	22	23	24	25

SIMULAÇÃO

LIMITE: 25 | RAIZ: 5

	2	3	4	5
6	7	8	9	10
11	12	13	14	15
16	17	18	19	20
21	22	23	24	25

Simulação

LIMITE: 25 | RAIZ: 5

	2	3	4	5
6	7	8	9	10
11	12	13	14	15
16	17	18	19	20
21	22	23	24	25

SIMULAÇÃO

LIMITE: 25 | RAIZ: 5

	2	3	4	5
6	7	8	9	10
11	12	13	14	15
16	17	18	19	20
21	22	23	24	25

Simulação

LIMITE: 25 | RAIZ: 5

	2	3	4	5
6	7	8	9	10
11	12	13	14	15
16	17	18	19	20
21	22	23	24	25

Simulação

LIMITE: 25 | RAIZ: 5

	2	3	4	5
6	7	8	9	10
11	12	13	14	15
16	17	18	19	20
21	22	23	24	25

Simulação

LIMITE: 25 | RAIZ: 5

	2	3	4	5
6	7	8	9	10
11	12	13	14	15
16	17	18	19	20
21	22	23	24	25

Simulação

LIMITE: 25 | RAIZ: 5

	2	3	4	5
6	7	8	9	10
11	12	13	14	15
16	17	18	19	20
21	22	23	24	25

Simulação

LIMITE: 25 | RAIZ: 5

	2	3	4	5
6	7	8	9	10
11	12	13	14	15
16	17	18	19	20
21	22	23	24	25

Simulação

LIMITE: 25 | RAIZ: 5

	2	3	4	5
6	7	8	9	10
11	12	13	14	15
16	17	18	19	20
21	22	23	24	25

Simulação

LIMITE: 25 | RAIZ: 5

	2	3	4	5
6	7	8	9	10
11	12	13	14	15
16	17	18	19	20
21	22	23	24	25

Simulação

LIMITE: 25 | RAIZ: 5

	2	3	4	5
6	7	8	9	10
11	12	13	14	15
16	17	18	19	20
21	22	23	24	25

Simulação

LIMITE: 25 | RAIZ: 5

	2	3	4	5
6	7	8	9	10
11	12	13	14	15
16	17	18	19	20
21	22	23	24	25

Simulação

LIMITE: 25 | RAIZ: 5

	2	3	4	5
6	7	8	9	10
11	12	13	14	15
16	17	18	19	20
21	22	23	24	25

Simulação

LIMITE: 25 | RAIZ: 5

	2	3	4	5
6	7	8	9	10
11	12	13	14	15
16	17	18	19	20
21	22	23	24	25

SIMULAÇÃO

LIMITE: 25 | RAIZ: 5

	2	3	4	5
6	7	8	9	10
11	12	13	14	15
16	17	18	19	20
21	22	23	24	25

```python
# Algoritmo "Crivo de Eratóstenes"
# lim = valor do limite
def crivo(lim):
    lista = [n for n in range(lim + 1)]
    lista[1] = 0
    raiz = floor(sqrt(lim))
    for i in range(2, raiz + 1):
        marca_multiplos(i, lista, lim)
    return filtra(lista)
```

```python
# Marca os múltiplos de n com zero.
# Obs.: a partir do segundo múltiplo.
def marca_multiplos(n, lista, lim):
    for i in range(n * 2, lim + 1, n):
        lista[i] = 0


# Devolve uma nova lista sem zeros.
def filtra(lista):
    return [n for n in lista if n != 0]
```

VAMOS EXERCITAR!

Marianne e os Primos Gêmeos

Por Gustavo Ribeiro, IFPB - Campina Grande Brazil

Timelimit: 1

1926

Marianne está criando um jogo chamado "Herói da Guitarra". É um trabalho extremamente cansativo, que requer bastante empenho e tempo, mas nada que uma greve não resolva. Ao abrir o seu email, Mari se deparou com um problema bastante curioso proposto pelos primos Renè e Leonhard e pelos gêmeos Isaac e Carl.

O problema é descrito da seguinte forma:

"Um número natural é dito primo, se ele possui exatamente dois divisores naturais distintos: o número um e ele mesmo. Um número é dito primo gêmeo, se e somente se, ele for primo e houver outro número primo qualquer cuja diferença absoluta entre esse dois números primos seja igual a dois. Por exemplo, o número 3 é um primo gêmeo, pois ele é primo e existe outro primo (5) tal que $|3 - 5| = 2$, já o número 23, apesar de ser primo, não é um primo gêmeo. Você poderia nos dizer quantos número primos gêmeos existem entre *x* e *y*, inclusive?"

Marianne adora resolver esse tipo de problema, mas está muito ocupada criando o seu próprio jogo de Herói da Guitarra. Você pode ajudar?

Entrada

A primeira linha de entrada irá conter um inteiro $1 \leq \mathbf{Q} \leq 10^5$, o número de consultas, cada uma das próximas **Q** linhas irá contér dois inteiros, $1 \leq \mathbf{X}, \mathbf{Y} \leq 10^6$.

Saída

Para cada uma das **Q** consultas, imprima a quantidade de número primos gêmeos entre **X** e **Y**, inclusive.

Viagem à Marte na Velocidade de Primo

Por Neilor Tonin, URI Brazil

Timelimit: 1

2180

Um grupo de cientistas está fazendo novas experiências para criar uma nave que possibilite a viagem muito mais rápida até Marte do que é possível atualmente. Esta nave utilizará dois foguetes e um novo combustível recém criado, muito mais eficiente que os utilizados até hoje. Só que a velocidade que os novos foguetes podem proporcionar à nave está relacionada diretamente com o peso do combustível armazenado nestes foguetes (em kg) e, por incrível que pareça, uma relação deste peso com números primos. Por exemplo, se o peso total do combustível dos foguetes for 1010 kg, a velocidade atingida (em km/h) é a soma dos 10 números primos à partir de 1010 (incluindo ele se for primo): 1013 -> 1019 -> 1021 -> 1031 -> 1033 -> 1039 -> 1049 -> 1051 -> 1061 -> 1063, ou seja, 10380 km/h.

Os cientistas estão muito intrigados com esta relação matemática existente e querem que você construa um programa que calcule quanto tempo aproximado (em horas e em dias) uma nave levaria para ir da terra até marte com este novo combustível, dado um determinado peso de foguetes (claro, eles estão tentando criar os maiores foguetes possíveis) assumindo que a distância da terra até marte no dia do lançamento, será 60 milhões de kms.

Entrada

A entrada contém um único valor inteiro **Peso** (1000 < **Peso** ≤ 60000) indicando o peso máximo de combustível (em kg) que os foguetes podem armazenar.

Contato

Lucio Nunes

lucio.nunes@hotmail.com

Grace Borges

graceapborges@fatecsp.br

interfatecs@fatecsp.br