Números Inteiros e Criptografia, 2020.1

Lista de Exercícios 6

Submeta as soluções das questões marcadas com *****
até 29 de janeiro às 18:00 salvando um arquivo na sua pasta no Google Drive

Justifique todas as questões.

***Questão 1.** Considere as seguintes funções definidas para n natural:

- $\omega(n)$ = número de fatores primos de n *distintos*.
- $\Omega(n)$ = número de fatores primos de n *contando todas as repetições!*
- $d(n)$ = número de divisores positivos de n
- $S(n)$ = soma dos divisores positivos de n
- $h(n) = n^{123456789}$
- $j(n) = 123456789 \cdot n$

Exemplos de valores das funções ω, Ω, d, S estão dados na tabela abaixo:

n	fatoração em primos	$\omega(n)$	$\Omega(n)$	divisores	$d(n)$	$S(n)$
1	—	0	0	1	1	1
2	2	1	1	1, 2	2	3
3	3	1	1	1, 3	2	4
4	2^2	1	2	1, 2, 4	3	7
8	2^3	1	3	1, 2, 4, 8	4	15
15	$3 \cdot 5$	2	2	1, 3, 5, 15	4	24
120	$2^3 \cdot 3 \cdot 5$	3	5	1, 2, 3, 4, 5, 6, 8, 10, 12, 15, 20, 24, 30, 40, 60, 120	16	360

Dizemos que uma função $f : \mathbb{N} \to \mathbb{N}$ é:

- *aditiva* se, para todos $n, m \in \mathbb{N} \setminus \{0\}$:

$$\text{se } \operatorname{mdc}(n,m) = 1 \text{ então } f(n \cdot m) = f(n) + f(m).$$

- *completamente aditiva* se, para todos $n, m \in \mathbb{N} \setminus \{0\}$:

$$f(n \cdot m) = f(n) + f(m).$$

- *multiplicativa* se, para todos $n, m \in \mathbb{N} \setminus \{0\}$:

$$\text{se } \operatorname{mdc}(n,m) = 1 \text{ então } f(n \cdot m) = f(n) \cdot f(m).$$

- *completamente multiplicativa* se, para todos $n, m \in \mathbb{N} \setminus \{0\}$:

$$f(n \cdot m) = f(n) \cdot f(m).$$

Para cada uma das funções ω, Ω, d, S, h e j definidas acima, e para cada uma das propriedades *aditiva*, *completamente aditiva*, *multiplicativa* e *completamente multiplicativa*, diga se a função tem a propriedade ou não, provando cada caso positivo e dando um contra-exemplo para cada caso negativo. (*Dica:* poupe um pouco do seu trabalho notando que uma função ser *completamente aditiva* já implica que ela seja *aditiva* [qual a contrapositiva dessa implicação?], e analogamente para *completamente multiplicativa* e *multiplicativa*.)

Questão 2. Dado um natural n, definimos o *primorial* de n, denotado $n^{\#}$, da seguinte forma:

$$\begin{cases} n^{\#} = 1, & \text{se } n < 2 \\ n^{\#} = \text{o produto de todos os primos} & \\ \qquad \text{de 2 até (inclusive, se for o caso) } n, & \text{se } n \geq 2 \end{cases}$$

*** a.** Prove que, para todos naturais n e p, se p é primo e $p \leq n$, então p não divide $n^{\#} + 1$.

*** b.** Use o item anterior para provar a *infinitude dos primos* na seguinte forma:

$$\text{para todo natural } n, \text{ existe um primo } p > n.$$

Questão 3. Faça uma função de `python` que receba dois naturais `n` e `limite` e retorne `True` caso todos os fatores primos de `n` sejam menores ou iguais ao `limite`, e retorne `False` em caso contrário. *Atenção*! Em <u>nenhum</u> momento a sua função pode testar se um número `k` é divisível por algum número `m`, se `m` não for primo. Quando `limite` for um número pequeno (digamos, com até 5 algarismos), a sua função <u>deve</u> ser executada rapidamente mesmo para `n` enorme (digamos, com 500 algarismos).

Você não precisa provar formalmente a corretude e terminação do seu programa, mas deve incluir uma explicação de como e por que ele funciona.

***Questão 4.** Dado um número natural $k > 0$, vamos chamar de *primos k-gêmeos* os números primos que estão à distância k um do outro. Por exemplo, para $k = 2$ temos 3 & 5, 5 & 7, 11 & 13, etc., para $k = 4$ temos 3 & 7, 7 & 11, 13 & 17, etc.

Faça uma função em `python` que receba dois naturais `k` e `limite` e retorne a lista de todos os pares de primos k-gêmeos que existem até (incluindo, se for o caso) o `limite`.

Para receber a pontuação máxima, sua função deve funcionar em tempo razoável (no máximo alguns segundos no Google Colab) pelo menos para `limite` com até 5 algarismos.