felizeseducados

March 2, 2026

1 Números Felizes e Números Educados

Prof. Doherty Andrade

2 1. Números Felizes

Dado um número natural $n = d_k d_{k-1} \dots d_1 d_0$ definimos a função S dada por

$$S(n) = \sum_{i=0}^{k} d_i^2.$$

Dizemos que n é um número feliz se existe um natural $m > 0$ tal que

$$S^m(n) = 1.$$

Por exemplo, 7 é um número feliz. De fato:

$$\begin{aligned} S(7) &= 7^2 = 49 \\ S^2(7) &= 4^2 + 9^2 = 97 \\ S^3(7) &= 9^2 + 7^2 = 130 \\ S^4(7) &= 0^2 + 1^2 + 3^2 = 10 \\ S^5(7) &= 0^2 + 1^2 = 1. \end{aligned}$$

Da definição, vemos que acrescentando dígitos zero em qualquer posição em n ainda obtemos um número feliz. Também é claro que realizando permutações dos dígitos de n ainda teremos um número feliz. Apenas com este jogo de zeros e permutações deduzimos que existem infinitos números felizes.

Mas nem todo número natural é feliz. O processo pode pode não terminar em 1. Caso o processo entre em um ciclo que não atinja 1, o número é chamad de número infeliz.

}. Do mesmo modo, permutando os dígitos de um número infeliz ou realizando inclusões de dígitos zeros obtemos de novo um número infeliz. E, portanto, podemos afirmar que existem infintios números infelizes.

A sequência $S^k(4)$ entra no ciclo

$$4 - -6 - -37 - -58 - -89 - -145 - -42 - -20 - -4.$$

Também é o caso do número $n = 197$, cuja sequência é:

$$197, 131, 11, 2, 4, 16, 37, 58, 89, 145, 42, 20, 4.$$

2.1 2. Código para números felizes

Decide se um dado número natural é feliz e apresenta a sequência.

[12]:
```python
# Implementação formal dos números felizes

def soma_quadrados_digitos(n: int) -> int:
    """Implementa a função S(n) = soma dos quadrados dos dígitos"""
    return sum(int(d) ** 2 for d in str(n))

def sequencia_feliz(n: int) -> list:
    """Gera a sequência S(n), S²(n), S³(n), ... até repetir"""
    sequencia = []
    vistos = set()
    atual = n

    while atual not in vistos:
        vistos.add(atual)
        sequencia.append(atual)
        atual = soma_quadrados_digitos(atual)

    sequencia.append(atual)  # Adiciona o primeiro elemento repetido
    return sequencia

def eh_feliz(n: int) -> bool:
    """Determina se um número é feliz (atinge 1)"""
    seq = sequencia_feliz(n)
    return seq[-1] == 1
```

2.2 2. Exemplos

[13]:
```python
eh_feliz(7)
```

[13]:
```
True
```

[14]:
```python
eh_feliz(197)
```

[14]:
```
False
```

[15]:
```python
eh_feliz(2026)
```

[15]:
```
True
```

[16]:
```python
sequencia_feliz(197)
```

[16]:
```
[197, 131, 11, 2, 4, 16, 37, 58, 89, 145, 42, 20, 4]
```

[17]:
```python
sequencia_feliz(7)
```

[17]: [7, 49, 97, 130, 10, 1, 1]

2.3 3. Números Educados

Um número é dito educado se pode ser escrito como a soma de dois ou mais números inteiros positivos consecutivos.

O número 15 é um número educado. De fato, temos:

$$\begin{aligned} 15 &= 4+5+6 \\ 15 &= 7+8 \\ 15 &= 1+2+3+4+5. \end{aligned}$$

O número 121 também é educado. De fato,

$$121 = 6+7+8+9+10+11+12+13+14+15+16.$$

2.4 4. Código

Decide se um dado número natural é educado e apresenta as suas possíveis sequências.

[18]:
```python
# Números educados
def divisores_impares(n: int) -> list:
    """Retorna todos os divisores ímpares de n maiores que 1"""
    import math
    divisores = []
    limite = int(math.sqrt(n))

    for i in range(1, limite + 1, 2):
        if n % i == 0:
            if i > 1:
                divisores.append(i)
            outro = n // i
            if outro != i and outro > 1 and outro % 2 == 1:  # Garantindo que␣
 ↪seja ímpar
                divisores.append(outro)

    return sorted(divisores)

def sequencias_educadas(n: int) -> list:
    """Encontra todas as representações de n como soma de consecutivos"""
    if n < 3:
        return []

    sequencias = []
    dobro = 2 * n
    divisores = divisores_impares(dobro)

    for k in divisores:  # k é o número de termos
```

```
        termo = dobro // k
        a = (termo - k + 1) // 2  # primeiro termo
        if a >= 1 and k >= 2:
            sequencia = list(range(a, a + k))
            if sum(sequencia) == n:
                sequencias.append(sequencia)

    return sorted(sequencias, key=lambda x: (len(x), x[0]))

# Função principal para testar
def verificar_numero_educado(n: int):
    """Verifica se um número é educado e exibe as sequências"""
    sequencias = sequencias_educadas(n)

    if len(sequencias) == 0:
        print(f"O número {n} NÃO é um número educado.")
    else:
        print(f"O número {n} é um número educado!")
        print(f"Foram encontradas {len(sequencias)} sequência(s):")
        for i, seq in enumerate(sequencias, 1):
            print(f"  {i}: {' + '.join(map(str, seq))} = {sum(seq)}")

# Exemplos de uso
if __name__ == "__main__":
    # Testando com alguns números
    numeros_teste = [3, 5, 6, 9, 15, 2, 8,4]

    for numero in numeros_teste:
        print("-" * 40)
        verificar_numero_educado(numero)
```

```
----------------------------------------
O número 3 NÃO é um número educado.
----------------------------------------
O número 5 NÃO é um número educado.
----------------------------------------
O número 6 é um número educado!
Foram encontradas 1 sequência(s):
  1: 1 + 2 + 3 = 6
----------------------------------------
O número 9 é um número educado!
Foram encontradas 1 sequência(s):
  1: 2 + 3 + 4 = 9
----------------------------------------
O número 15 é um número educado!
Foram encontradas 2 sequência(s):
  1: 4 + 5 + 6 = 15
  2: 1 + 2 + 3 + 4 + 5 = 15
```

```
----------------------------------------
O número 2 NÃO é um número educado.
----------------------------------------
O número 8 NÃO é um número educado.
----------------------------------------
O número 4 NÃO é um número educado.
```

2.5 5. Exemplos

[19]:
```
verificar_numero_educado(121)
```

```
O número 121 é um número educado!
Foram encontradas 1 sequência(s):
  1: 6 + 7 + 8 + 9 + 10 + 11 + 12 + 13 + 14 + 15 + 16 = 121
```

[20]:
```
verificar_numero_educado(15)
```

```
O número 15 é um número educado!
Foram encontradas 2 sequência(s):
  1: 4 + 5 + 6 = 15
  2: 1 + 2 + 3 + 4 + 5 = 15
```

[21]:
```
verificar_numero_educado(808)
```

```
O número 808 NÃO é um número educado.
```

2.6 Números primos felizes

Eles existem?

No episódio 42 (ano 2007) da série da TV britânica DoctorWho, a sequência 313, 331, 367, ____ precisa ser completada para o desbloqueio da porta de uma nave espacial que está prestes a colidir com uma estrela. The Doctor (interpretado nessa temporada pelo ator David Tennant) diz que a sequênciadeve ser completada com 379, que é o próximo número primo feliz, e ainda ironiza aqueles que desconfiam da sua lógicadizendo: "- Será que não se ensina mais matemática recreativanas escolas?".

[22]:
```
def soma_quadrados_digitos(n: int) -> int:
    """Implementa a função S(n) = soma dos quadrados dos dígitos"""
    return sum(int(d) ** 2 for d in str(n))

def sequencia_feliz(n: int) -> list:
    """Gera a sequência S(n), S²(n), S³(n), ... até repetir"""
    sequencia = []
    vistos = set()
    atual = n

    while atual not in vistos:
        vistos.add(atual)
        sequencia.append(atual)
```

```
        atual = soma_quadrados_digitos(atual)

    sequencia.append(atual)  # Adiciona o primeiro elemento repetido
    return sequencia

def eh_feliz(n: int) -> bool:
    """Determina se um número é feliz (atinge 1)"""
    seq = sequencia_feliz(n)
    return seq[-1] == 1

def eh_primo(n: int) -> bool:
    """
    Verifica se um número é primo.
    Retorna True se for primo, False caso contrário.
    """
    if n < 2:
        return False
    if n == 2:
        return True
    if n % 2 == 0:
        return False

    # Verifica divisores ímpares até a raiz quadrada de n
    for i in range(3, int(n ** 0.5) + 1, 2):
        if n % i == 0:
            return False
    return True

def eh_feliz_e_primo(n: int) -> bool:
    """
    Verifica se um número é feliz E primo.
    Retorna True se for feliz e primo, False caso contrário.
    """
    return eh_feliz(n) and eh_primo(n)

def testar_numeros(intervalo: range) -> None:
    """
    Testa números em um intervalo e mostra quais são felizes e primos.
    """
    print("Números felizes e primos encontrados:")
    print("-" * 40)
    encontrados = 0

    for num in intervalo:
        if eh_feliz_e_primo(num):
            print(f"{num}: Feliz e Primo  ")
            encontrados += 1
```

```
    print("-" * 40)
    print(f"Total encontrado: {encontrados}")

# Exemplo de uso
if __name__ == "__main__":
    # Testando números de 1 a 100
    print("Testando números de 1 a 501:")
    testar_numeros(range(1, 501))

    # Teste individual
    print("\n" + "=" * 50)
    numero_teste = 7
    print(f"Testando o número {numero_teste}:")
    print(f"É feliz? {eh_feliz(numero_teste)}")
    print(f"É primo? {eh_primo(numero_teste)}")
    print(f"É feliz E primo? {eh_feliz_e_primo(numero_teste)}")

    # Sequência do número
    print(f"\nSequência do número {numero_teste}:␣
 ↪{sequencia_feliz(numero_teste)}")
```

```
Testando números de 1 a 501:
Números felizes e primos encontrados:
----------------------------------------
7: Feliz e Primo
13: Feliz e Primo
19: Feliz e Primo
23: Feliz e Primo
31: Feliz e Primo
79: Feliz e Primo
97: Feliz e Primo
103: Feliz e Primo
109: Feliz e Primo
139: Feliz e Primo
167: Feliz e Primo
193: Feliz e Primo
239: Feliz e Primo
263: Feliz e Primo
293: Feliz e Primo
313: Feliz e Primo
331: Feliz e Primo
367: Feliz e Primo
379: Feliz e Primo
383: Feliz e Primo
397: Feliz e Primo
409: Feliz e Primo
487: Feliz e Primo
```

```
----------------------------------------
Total encontrado: 23

================================================
Testando o número 7:
É feliz? True
É primo? True
É feliz E primo? True

Sequência do número 7: [7, 49, 97, 130, 10, 1, 1]
```

[ ]: