Řešení domácích úkolů - 2. týden

Eulerova funkce ϕ

Eulerova funkce (totient) $\phi(n)$ je počet čísel z množiny $1, 2, \ldots n$, která jsou nesoudělná s n (t.j. jejich největší společný dělitel s n je 1). Jejím nejznámějším použitím je zobecněná Malá Fermatova věta, která tvrdí, že pro a, n nesoudělná platí $a^{\phi(n)} \equiv 1 \pmod{n}$.

Napište kus Pythonu, který ze standardního výstupu načte n a na standardní výstup vypíše $\phi(n)$.

Řešení

Analýza Existuje několik způsobů, jak spočíst totient pro dané n. Tady uvedeme řešení, které přímo vychází z definice $\phi(n)$: spočteme počet čísel od 1 do n-1, která jsou nesoudělná s n. Soudělnost $k < n$ budeme diagnostikovat tak, že spočteme největšího společného dělitele a pokud bude 1, budou čísla nesoudělná. Algoritmus na GCD je velice rychlý, $O(\log n)$, takže to nemusí být mimořádně neefektivní.

Vzorové řešení

```
n = int(input())

result = 1             # 1 započítáváme automaticky
for i in range(2, n): # hledáme nesoudělitelná čísla od 2 do n-1
    a = i              # počítáme gcd standardně modulením
    b = n
    while b > 0:
        a, b =b, a%b

    if a==1:            # pokud je gcd 1, započítáme do totientu
        result+=1

print(result)

```

Alternativní řešení

Alternativně můžeme počítat totient metodou připomínající Erastothenovo síto: nalezneme prvočíselný rozklad n a vyškrtneme všechny násobky každého *unikátního* prvočísla.

```
n = int(input())
n_pracovni = n
totient = n
p = 2
while n_pracovni > 1:
    if n_pracovni % p == 0:
        totient -= totient // p
        while n_pracovni % p == 0:
            n_pracovni //= p
    p += 1

print(totient)
```

Na stejném principu se zakládá i Eulerův vzorec pro výpočet totientu:

$$\phi(n) = n \prod_{p_k} \left(1 - \frac{1}{p_k}\right), \quad \prod_k p_k^{\alpha_k} = n$$

Obvyklé problémy

Tato úloha vyžaduje dobře si promyslet algoritmus, zorganizovat si kód a hlídat si složitost logiky. Pokud je váš kód složitý, použijte klidně `math.gcd()`, abyste si ho projasnili.