3. Programmieraufgabe Computerorientierte Mathematik I

Abgabe: 26.11.2021 über den Comajudge bis 17 Uhr

Bitte beachten Sie: Die Herausgabe oder der Austausch von Code (auch von Teilen) zu den Programmieraufgaben führt für *alle* Beteiligten zum *sofortigen Scheinverlust.* Die Programmieraufgaben müssen von allen Teilnehmenden alleine bearbeitet werden. Auch Programme aus dem Internet dürfen nicht einfach kopiert werden.

1 Problembeschreibung

Eine Primzahl ist eine Zahl $p \in \mathbb{N}$, welche nur durch 1 und sich selbst teilbar ist. Es sei $n \in \mathbb{N}$ mit $n \geq 2$. Dann hat n eine bis auf die Reihenfolge der Faktoren eindeutige Primzahlzerlegung der Form

$$n = p_1^{e_1} \cdot p_2^{e_2} \cdots p_M^{e_M} = \prod_{k=1}^{M} p_k^{e_k}$$

Der Exponent e_k eines Primfaktors p_k wird die Vielfachheit von p_k genannt. Es gilt zum Beispiel

$$45 = 3^2 \cdot 5,$$

mit $p_1 = 3$, $e_1 = 2$ und $p_2 = 5$, $e_2 = 1$. Zwei Zahlen werden teilerfremd genannt, wenn sie keinen gemeinsamen Primfaktor p_k haben. Zum Beispiel ist

$$28 = 2^2 \cdot 7$$

teilerfremd zu 45, da die Mengen $\{3,5\}$ und $\{2,7\}$ disjunkt sind.
Die Teilerzahl $\tau(n)$ einer natürlichen Zahl $n \geq 1$ ist die Zahl der natürlichen Zahlen, welche n teilen. Es gilt

$$\tau(n) = (e_1+1)(e_2+1)\cdots(e_M+1), \tag{1}$$

wobei die e_i die Vielfachheiten der Primfaktoren von n sind.
Hierbei werden 1 und n mitgezählt. Zum Beispiel gilt $\tau(6) = 4$, da 6 von 1, 2, 3 und 6 geteilt wird. Für jede Primzahl p gilt $\tau(p) = 2$. Insbesondere gilt: Zwei natürliche Zahlen $n, m \geq 1$ sind genau dann teilerfremd, wenn gilt

$$\tau(n \cdot m) = \tau(n) \cdot \tau(m). \tag{2}$$

1.1 Das Sieb des Eratosthenes (Quelle: Wikipedia)

Das Sieb des Eratosthenes ist ein Algorithmus zur Bestimmung einer Liste oder Tabelle aller Primzahlen kleiner oder gleich einer vorgegebenen Zahl.
Zunächst werden alle Zahlen 2, 3, 4, ... bis zu einem frei wählbaren Maximalwert S aufgeschrieben. Die zunächst unmarkierten Zahlen sind potentielle Primzahlen. Die kleinste unmarkierte Zahl ist immer eine Primzahl. Nachdem eine Primzahl gefunden wurde, werden alle Vielfachen dieser Primzahl als zusammengesetzt markiert. Man bestimmt die nächstgrößere unmarkierte Zahl. Da sie kein Vielfaches von Zahlen kleiner als sie selbst ist (sonst wäre sie markiert worden), kann sie nur durch eins und sich selbst teilbar sein. Folglich muss es sich um eine Primzahl handeln.

Diese wird dementsprechend als Primzahl ausgegeben. Man streicht wieder alle Vielfachen und führt das Verfahren fort, bis man am Ende der Liste angekommen ist. Im Verlauf des Verfahrens werden alle Primzahlen ausgegeben.
Da mindestens ein Primfaktor einer zusammengesetzten Zahl immer kleiner gleich der Wurzel der Zahl sein muss, ist es ausreichend, nur die Vielfachen von Zahlen zu streichen, die kleiner oder gleich der Wurzel der Schranke S sind.

2 Aufgabenstellung und Anforderungen

Schreiben Sie folgende Funktionen:

1. `sieve(n)` Gibt eine aufsteigend geordnete Liste der Primzahlen p_k mit $p_k \leq n$ zurück. Die Funktion gibt `None` zurück, falls $n < 2$.
   Anmerkung: Es gibt viele Wege, dies zu realisieren. Eine Möglichkeit ist die Implementierung des oben beschriebenen Siebes des Eratosthenes.
   Beispielaufrufe:

```
>>> print (sieve(0))
None
>>> print (sieve(2))
[2]
>>> print (sieve(15))
[2, 3, 5, 7, 11, 13]
>>> print (sieve(13))
[2, 3, 5, 7, 11, 13]
```

2. `isprime(n)` Gibt `True` zurück, falls $n \geq 2$ eine Primzahl ist, und `False`, falls nicht. Die Funktion gibt `None` zurück, falls $n < 2$.
   Beispielaufrufe:

```
>>> print (isprime(1))
None
>>> print (isprime(2))
True
>>> print (isprime(10))
False
```

3. `factorization(n)` Gibt eine Liste von Listen der Länge 2 zurück. Jede der Listen enthält im ersten Eintrag einen der Primfaktoren p_k von n und im zweiten die Vielfachheit e_k von p_k. Die Primfaktoren sind nach aufsteigender Größe sortiert. Die Funktion gibt `None` zurück, falls $n < 2$.
   Anmerkung: Eine Liste von Kandidaten für Primfaktoren gibt Ihnen `sieve(m)`, wobei $m = \lceil n/2 \rceil$. Wenn n sich durch keinen Faktor in dieser Liste teilen läßt, ist es selbst eine Primzahl. Die Vielfachheit e_k eines Primfaktors p_k von n ist die Anzahl der Male, die sich n ohne Rest durch p_k teilen läßt.
   Beispielaufrufe:

```
>>> print (factorization(1))
None
>>> print (factorization(13))
[[13, 1]]
>>> print (factorization(64))
```

```
[[2, 6]]
>>> print (factorization(23432))
[[2, 3], [29, 1], [101, 1]]
```

4. `divisornumber(n)` Die Funktion gibt die Teilerzahl von n zurück. Sie gibt 1 zurück, falls $n=1$ und `None`, falls $n<1$.
   Anmerkung: Nutzen Sie die Ausgabe von `factorization(n)`, um die Teilerzahl anhand von Gleichung (1) zu berechnen.
   Beispielaufrufe:

```
>>> print (divisornumber(0))
None
>>> print (divisornumber(1))
1
>>> print (divisornumber(13))
2
>>> print (divisornumber(64))
7
>>> print (divisornumber(23432))
16
```

5. `iscoprime(n,m)` Die Funktion gibt `True` zurück, falls n und m teilerfremd sind und `False`, falls nicht. Die Funktion gibt `None` zurück, falls $n<1$ oder $m<1$.
   Anmerkung: Nutzen Sie die Funktion `divisornumber(n)` und Gleichung (2), um den Rückgabewert zu berechnen.
   Beispielaufrufe:

```
>>> print (iscoprime(0,64))
None
>>> print (iscoprime(1,64))
True
>>> print (iscoprime(1,1))
True
>>> print (iscoprime(13,64))
True
>>> print (iscoprime(23432,64))
False
```