Technische Universität Berlin **WiSe 2023/24**
Fakultät II, Institut für Mathematik
Sekretariat MA 5–2, Dorothea Kiefer-Hoeft
Prof. Dr. Max Klimm
Dr. Frank Lutz, Svenja M. Griesbach, Martin Knaack

4. Programmieraufgabe Computerorientierte Mathematik I

Abgabe: 01.12.2023 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. Eine natürliche Zahl $n \in \mathbb{N}$ mit $n \geq 2$ hat eine bis auf die Reihenfolge der Faktoren eindeutige *Primfaktorzerlegung* der Form

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

mit Primfaktoren p_k und Exponenten e_k. Der Exponent e_k wird auch als die Vielfachheit von p_k bezeichnet. Zum Beispiel gilt

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

mit $p_1 = 3, e_1 = 2$ und $p_2 = 5, e_2 = 1$.

Zwei natürliche Zahlen $m, n \geq 2$ werden *teilerfremd* genannt, wenn sie keinen gemeinsamen Primfaktor p_k haben. Zum Beispiel sind

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

und 45 teilerfremd, da die Mengen $\{3, 5\}$ und $\{2, 7\}$ disjunkt sind. Zudem definieren wir, dass die Zahlen 1 und n für alle natürlichen Zahlen $n \geq 1$ teilerfremd sind.

Die *Eulersche Phi-Funktion* $\phi(n)$ einer natürlichen Zahl $n \geq 1$ ist die Anzahl der natürlichen Zahlen $a \in \{1, 2, \ldots, n\}$ die zu n teilerfremd sind. Zum Beispiel gilt $\phi(6) = 2$, da 6 nur zu 1 und 5 teilerfremd ist. Für eine Primzahl $p \in \mathbb{N}$ und eine natürliche Zahl $e \geq 1$ gilt

$$\phi(p^e) = p^e - p^{e-1}, \tag{1}$$

da p^e nur zu Vielfachen von p nicht teilerfremd ist und da die Anzahl der Vielfachen von p die kleiner gleich p^e sind, gleich p^{e-1} entspricht $(1 \cdot p, 2 \cdot p, \ldots, p^{e-1} \cdot p)$.

Zudem gilt, dass zwei natürliche Zahlen $n, m \geq 1$ genau dann teilerfremd sind, wenn gilt

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

Somit gilt zum Beipsiel, dass

$$\phi(45) = \phi(3^2 \cdot 5) = \phi(3^2) \cdot \phi(5) = (3^2 - 3^1) \cdot (5^1 - 5^0) = 24.$$

1.1 Das Sieb des Eratosthenes

Das Sieb des Eratosthenes haben Sie bereits in der Vorlesung kennengelernt. Dieser Algorithmus dient zur Bestimmung aller Primzahlen kleiner oder gleich einer vorgegebenen Schranke $S \geq 2$. Dafür werden zunächst alle Zahlen $2, 3, \ldots, S$ abgespeichert. Im Laufe des Algorithmus bekommen alle Zahlen entweder die Markierung *Primzahl* oder *keine Primzahl*. Noch nicht betrachte Zahlen bleiben unmarkiert.

In jedem Schritt sucht der Algorithmus die kleinste unmarkierte Zahl. Anschließend werden alle ganzzahligen Vielfache dieser Zahl mit der Markierung *keine Primzahl* versehen. Dabei wird klar, dass es sich bei der kleinsten unmarkierten Zahl am Anfang um eine Primzahl handeln muss, da sie in den vorherigen Schritten nicht markiert wurde und somit nur die Teiler 1 und sich selbst besitzt. Sie wird also als *Primzahl* markiert und das Vorgehen wird wiederholt, bis man am Ende der Liste angekommen ist.

Da mindestens ein Primfaktor einer zusammengesetzten Zahl immer kleiner gleich der Wurzel der Zahl sein muss, reicht es aus, nur die Vielfachen von Zahlen zu streichen, die kleiner oder gleich der Wurzel von S sind.

2 Aufgabenstellung und Anforderungen

1. Schreiben Sie eine Funktion

   `sieve(n)`,

   die für $n \geq 2$ eine aufsteigend geordnete Liste der Primzahlen p_k mit $p_k \leq n$ zurückgibt. Andernfalls soll `None` ausgegeben werden.

   Anmerkung: Es gibt viele Wege, um dies zu realisieren. Eine Möglichkeit ist die Implementierung des oben beschriebenen Sieb des Eratosthenes.

   Beispielaufrufe:

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

2. Schreiben Sie eine Funktion

   `isprime(n)`,

   die `True` ausgibt, falls $n \geq 2$ eine Primzahl ist und `False`, falls nicht. Die Funktion soll `None` ausgeben, falls $n < 2$.

   Beispielaufrufe:

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

3. Schreiben Sie eine Funktion

   `factorization(n)`,

   die eine Liste von Listen der Länge 2 zurückgibt. 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 soll `None` zurückgeben, falls $n < 2$.

   Anmerkung: Eine Liste von Kandidaten für Primfaktoren gibt Ihnen `sieve(m)`, mit $m = \lceil n/2 \rceil$. Wenn n sich durch keinen Faktor in dieser Liste teilen lässt, dann ist n 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:

```
>>> L = factorization(1)
>>> print(L)
None
>>> L = factorization(13)
>>> print(L)
[[13, 1]]
>>> L = factorization(64)
>>> print(L)
[[2, 6]]
>>> L = factorization(23432)
>>> print(L)
[[2, 3], [29, 1], [101, 1]]
```

4. Schreiben Sie eine Funktion

   `euler_phi(n)`,

   die für $n \geq 1$ den Wert von $\phi(n)$ zurückgibt. Sie soll `None` zurückgeben, falls $n < 1$.

   Anmerkung: Nutzen Sie Gleichung (1) um die Funktion ϕ für Primzahlen auszuwerten. Für alle anderen Zahlen können Sie die Primfaktorzerlegung von `factorization(n)` und Gleichung (2) zur Berechnung verwenden.

   Beispielaufrufe:

```
>>> p = euler_phi(0)
>>> print(p)
None
>>> p = euler_phi(1)
>>> print(p)
1
>>> p = euler_phi(6)
>>> print(p)
2
>>> p = euler_phi(20)
>>> print(p)
8
```

5. Schreiben Sie eine Funktion

   `iscoprime(m,n)`,

   die `True` zurückgibt, falls m und n teilerfremd sind und `False`, falls nicht. Die Funktion soll `None` ausgeben, falls $n < 1$ oder $m < 1$.

   Anmerkung: Welche Probleme können auftreten, wenn man diese Aufgabe mit Gleichung (2) löst? Überlegen Sie sich eine andere Möglichkeit mit den bereits implementierten Funktionen.

   Beispielaufrufe:

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

3 Hinweise

1. Die Funktionen werden mit großen Eingaben getestet. Beachten Sie für eine effiziente Programmierung die Anmerkungen zu den Funktionen und die Informationen im Abschnitt 1.1.

2. Zu Ihrer Erinnerung: Sie müssen eine der Programmieraufgaben PA04, PA05 oder PA06 bei einem Tutor oder einer Tutorin in den Rechnerbetreuungen vorstellen. Die vorläufige Deadline ist der 22.12.23.