1.6. Laufzeit von Algorithmen

Definition 1.1. Die Laufzeit eines Algorithmus mit gegebenem Input ist die Anzahl elementarer Rechenoperationen (Vergleiche, Zuweisungen, Additionen, ...), die der Algorithmus für diesen Input durchführt. Die Laufzeit eines Algorithmus für beliebige Inputs wird als Funktion eines oder mehrerer Inputparameter gemessen.

Beispiel. Die Laufzeit eines Primzahltests wird als Funktion der im Input gegebenen Zahl n angegeben. Beobachtungen:

Algorithm 1 Schneller Primzahltest

```
if n ≥ 1 then
    antwort ← False
else
    antwort ← True
end if
for i ← 2 to √n do
    if i ist Teiler von n then
        antwort ← False
    end if
end for
return Antwort
```

- Der Algorithmus liefert das korrekte Ergebnis, die Laufzeit des Algorithmus ist in $\mathcal{O}(\sqrt{n})$.
- Es gibt einen schnellerern Algorithmus zum Primzahltest mit Laufzeit in $\mathcal{O}(\log(n)^k)$ für eine Konstante k.
- Es ist ein offenes Forschungsproblem, ob es einen Algorithmus mit Laufzeit $\mathcal{O}(\log(n)^k)$ für eine Konstante k gibt, der zu einer gegebenen Zahl n deren Primfaktorzerlegung bestimmt.

1.7. Sieb des Erathostenes

Algorithm 2 Sieb des Erathostenes

```
for i ← 2 to n do
    p(i) ← True
end for
for i ← 2 to √n do
    if p(i) = True then
        for j ← i to n/i do
            p(i · j) ← False
        end for
    end if
end for
for i ← 2 to n do
    if p(i) = True then
        return i
    end if
end for
```

Satz 1.2. *Der Algorithmus ist korrekt und hat eine Laufzeitfunktion in* $\mathcal{O}(n \log(n))$.

Korrektheitsbeweis. Zeige, dass für alle $k \in \{2, \ldots, n\}$ gilt:

$$k \text{ ist Teil der Ausgabe} \iff k \text{ ist Primzahl}$$

Es sei also $k \in \{2, \ldots, n\}$.
$\Longleftarrow$ Beweis per Kontraposition:
Wir nehmen an, dass k nicht Teil der Ausgabe ist. Also wird $p(k)$ im Laufe des Algorithmus auf False gesetzt. Folglich ist $k = i \cdot j$ mit $2 \le i \le j$ und k nicht prim.

$\Longrightarrow$ Beweis per Kontraposition:
Ist k nicht prim, so besitzt k einen kleinsten Teiler $i \ge 2$ mit $j := \frac{k}{i} \ge i$. Insbesondere ist i prim, $2 \le i \le \sqrt{n}$ und $i \le j \le \lfloor \frac{n}{i} \rfloor$. Da i prim, gilt die ganze Zeit $p(i) = \text{True}$ (wegen $\Longleftarrow$). Daher wird $p(i \cdot j)$ auf False gesetzt. Da $k = i \cdot j$, ist und bleibt $p(k) = \text{False}$. □

Laufzeitanalyse. Teile 1 und 3 benötigen offensichtlich $\mathcal{O}(n)$ Operationen. Die Anzahl der Operationen in Teil 2 ist nach oben beschränkt durch

$$c \cdot \sum_{i=2}^{\sqrt{n}} \left\lfloor \frac{n}{i} \right\rfloor$$

Wir können diese Abschätzung vereinfachen:

$$\sum_{i=2}^{\sqrt{n}} \left\lfloor \frac{n}{i} \right\rfloor \le \sum_{i=2}^{n} \frac{n}{i} \le n \cdot \sum_{i=2}^{n} \int_{i-1}^{i} \frac{1}{x} dx = n \int_{1}^{n} \frac{1}{x} dx = n \ln n$$

Die Laufzeitfunktion ist also in $\mathcal{O}(n \cdot \log n)$. □