IAD – Übungsblatt Nr. 4

Mona Scheerer (bv237), Jannnick Borowitz (ak236)
Tutor: Felix Lübbe

Aufgabe 1

(a) $(x+20)\log_{10}(x+20) \in O(x\log_{10}(x+20)$, denn nach VL ist skalare Multiplikation egal und dann genügt es nach der Sequenzregel $x\log_{10}(x+20)$ zu betrachten. Nun ist für $x \to \infty$ die Addition $+20$ irrelevant und die Komplexität der Logarithmen zu verschiedenen Basen äquivalent, daher:

$$(x+20)\log_{10}(x+20) \in O(x\log_2(x))$$

(b) Es ist $e^{\sqrt{log_2(x)}*\ln(2)} \prec \sqrt{x} + \log_2(x)$, denn:

$$\lim_{x\to\infty} \frac{e^{\sqrt{log_2(x)}*\ln(2)}}{\sqrt{x}+\log_2(x)} = \lim_{x\to\infty} \frac{\frac{e^{\sqrt{\log_2(x)}*\ln(x)}\sqrt{log(2)}}{2x\sqrt{log(x)}}}{\frac{1}{2\sqrt{x}} + \frac{1}{x\log(2)}} =$$

Es ist $\sqrt{x} + \log_2(x) \prec x\log_2(x)$, denn:

$$\lim_{x\to\infty} \frac{\sqrt{x}+\log_2(x)}{x\log_2(x)} = \lim_{x\to\infty} \frac{\frac{1}{2\sqrt{x}} + \frac{1}{x\log(2)}}{\log_2(x) + \frac{x}{x\log(2)}} = \frac{0+0}{\lim_{x\to\infty}\log_2(x) + \frac{x}{x\log(2)}} = 0$$

Es ist $x\log_2(x) \prec x^3 + 12x^2 + 200x + 999$, denn:

$$\lim_{x\to\infty} \frac{x\log_2(x)}{x^3+12x^2+200x+999} = \lim_{x\to\infty} \frac{\log_2(x) + \frac{1}{\log(2)}}{3x^2+24x+200} = \lim_{x\to\infty} \frac{\frac{1}{x\log(2)}}{6x+24} = 0$$

Es ist $x^3 + 12x^2 + 200x + 999 \prec e^{x\ln(3)}$

$$\lim_{x\to\infty} \frac{x^3+12x^2+200x+999}{e^{x\ln(3)}} = \lim_{x\to\infty} \frac{3x^2+24x+200}{\ln(3)e^{x\ln(3)}} = \lim_{x\to\infty} \frac{6x+24}{\ln(3)^2 e^{x\ln(3)}} = \lim_{x\to\infty} \frac{6}{\ln(3)^3 e^{x\ln(3)}} = 0$$

Es ist $e^{x\ln(3)} \prec x^x$

$$\lim_{x\to\infty} \frac{e^{x\ln(3)}}{x^x} = \lim_{x\to\infty} \frac{\ln(3)e^{\ln(3)x}}{(\ln(x)+1)e^{x\ln(x)}} = \lim_{x\to\infty} \frac{\ln(3)}{(\ln(x)+1)} e^{x(\ln(3)-\ln(x))}$$

Da $\ln(x)$ streng monoton steigend ist, gilt für alle $x \in \mathbb{R}_{>3} : \ln(x) > ln(3)$, woraus wir folgern können:

$$\lim_{x\to\infty} \frac{\ln(3)}{\ln(x)+1} e^{x(\ln(3)-\ln(x))} = 0 * e^{-\infty} = 0$$

Aufgabe 2

(a) Falls $primes[k] == 0$ und k keine Primzahl ist - sie also ein Viefaches einer Zahl <k ist, folgt, dass bereits alle k-Vielfache gestrichen sind. In Variante 1 wird dieser Fall nicht geprüft, in Variante 2 spart man sich durch eine Abfrage diese unnötigen Streichungen.

(b) Wir setzen $N = len(primes)$

```
def sieve1(N):
    primes = list(range(N+1))                       #O(1)
    primes[1] = 0 # Zahl 1 streichen                #O(1)
    stop = N                                        #O(1)
    k = 2                                           #O(1)
    while k <= stop:                                #O(stop-k+1)=O(N-k+1)
        j = 2*k                                         #O(1)
        while j <= N:                                   #O(N+1-2*k)
            primes[j] = 0 # j streichen                     #O(1)
            j += k                                          #O(1)
        k += 1                                          #O(1)
    return [k for k in primes if k!=0]              # O(1)
```

Durch die Schachtelungregel ergibt sich folgender Term in Abhängigkeit von N für die Streichungen:

$$\sum_{k=2}^{stop} \sum_{\substack{j=2k,\\ j+=k}}^{N} 1 \leq \sum_{k=2}^{stop} \frac{N-2k}{k} = \sum_{k=2}^{stop} \left(\frac{N}{k} - 2 \right)$$

$$= N \cdot \sum_{k=2}^{stop} \frac{1}{k} - \sum_{k=2}^{stop} 2 \overset{stop=N}{=} N \cdot \sum_{k=2}^{N} \frac{1}{k} - \sum_{k=2}^{N} 2$$

$$= N \cdot \sum_{k=2}^{N} \frac{1}{k} - (2N-2) = N \cdot \sum_{k=2}^{N} \frac{1}{k} - 2N + 2 = N(\sum_{k=2}^{N} \frac{1}{k} - 2) + 2 \in O(N \ln N)$$

Nach der Schachtelungsregel in den while-Schleifen erhalten wir $O(N \ln(N))$ und der Sequenzregel letztendlich $O(N \ln(N))$

(c)

```
def sieve2(N):
    primes = list(range(N+1))       #O(1)
    primes[1] = 0                   #O(1)
    stop = N                        #O(1)
    k = 2                           #O(1)
    while k <= stop:                #O(stop+1-k)
        if primes[k] != 0:              #O(1)
            j = 2*k                         #O(1)
            while j <= N:                   #O(N+1-2*k)
                primes[j] = 0               #O(1)
                j += k                      #O(1)
        k += 1                          #O(1)
    return [k for k in primes if k!=0]  #O(1)
```

$$\sum_{\substack{k=2,\\ k \text{ prime}}}^{stop} \sum_{\substack{j=2k,\\ j+=k}}^{N} 1 \leq \sum_{k=2}^{stop} \frac{N-2k}{k} = \sum_{\substack{k=2,\\ k \text{ prime}}}^{stop} \left(\frac{N}{k} - 2\right)$$
$$= N \cdot \sum_{\substack{k=2,\\ k \text{ prime}}}^{stop} \frac{1}{k} - \sum_{\substack{k=2,\\ k \text{ prime}}}^{stop} 2 \overset{stop=N}{=} N \cdot \sum_{\substack{k=2,\\ k \text{ prime}}}^{N} \frac{1}{k} - \sum_{\substack{k=2,\\ k \text{ prime}}}^{N} 2$$
$$= N \cdot \sum_{\substack{k=2,\\ k \text{ prime}}}^{N} \frac{1}{k} - (2N-2) = N \cdot \sum_{\substack{k=2,\\ k \text{ prime}}}^{N} \frac{1}{k} - 2N + 2 = N(\sum_{\substack{k=2,\\ k \text{ prime}}}^{N} \frac{1}{k} - 2) + 2 \in O(N \ln\ln N)$$

(d) Begründung, dass $stop = \sqrt{N}$ funktioniert: Sei z eine Zahl $\leq n$, die keine Primzahl ist, dann hat sie einen Teiler $\leq \sqrt{z}$, der ganzzahlig ist und nicht gleich 1 ist. Da z maximal n ist, folgt $stop = \sqrt{n}$. Es hätte eine Auswirkung auf die Komplexität. Denn die äußere while-Schleife würde nur $\sqrt{N}$-mal durchlaufen. Nach der Schachtelungsregel erhielten wir dann:

$$O(\sqrt{N} \ln(\ln(N)))$$

als Komplexität der zweiten Variante bzw. $O(\sqrt{N}(\ln(N)))$ bei der ersten Variante.