Übung 4

Philipp Hanslovsky, Frederick Bluemnthal

May 22, 2012

Aufgabe 1

1a)

- sei $h(x) \in \mathcal{O}(c f(x)) \Rightarrow \exists\, \tilde{c} > 0, x_0 \in \mathbb{R}$ so dass $\forall x \geq x_0$ gilt

$$h(x) \leq \tilde{c}\, c\, f(x)$$

$\Rightarrow \exists\, c' = \tilde{c} c > 0, x_0 \in \mathbb{R}$, so dass $\forall x \geq x_0$ gilt

$$h(x) \leq \tilde{c}\, c\, f(x) = c' f(x)$$

$$\Rightarrow \underline{\underline{h(x) \in \mathcal{O}(f(x))}}$$

- sei $h(x) \in \mathcal{O}(f(x))$, $k(x) \in \mathcal{O}(g(x))$

$$\left.\begin{array}{l} \Rightarrow \exists\, c_1 > 0, x_1 \in \mathbb{R} \text{ so dass } \forall x \geq x_1 \text{ gilt } h(x) \leq c_1 f(x) \\ \text{und } \exists\, c_2 > 0, x_2 \in \mathbb{R} \text{ so dass } \forall x \geq x_2 \text{ gilt } k(x) \leq c_2\, g(x) \end{array}\right\} (*)$$

$\Rightarrow h(x) \cdot k(x) \leq \max(c_1, c_2)\, g(x) f(x) \quad \forall x \geq \max(x_1, x_2)$

$\Rightarrow \underline{\underline{h(x) \cdot k(x) \in \mathcal{O}(g(x) f(x))}}$ mit $c = \max(c_1, c_2)$
$x_0 = \max(x_1, x_2)$

sei $h(x) \in O(f(x))$, $k(x) \in O(g(x))$

~~$\Rightarrow$ Es gilt (*) (siehe Seite 1)~~

~~$\Rightarrow h(x) + k(x)$~~

$\Rightarrow$ Es gilt (*) (siehe Seite 1)

Sei $g(x) \in O(f(x))$, also

$\exists\ c > 0,\ x_0 \in \mathbb{R}$ so dass $\forall x \geq x_0$ gilt $g(x) \leq c \cdot f(x)$

$$\Rightarrow k(x) \leq c_2\, g(x) \leq \overbrace{c_2 c_0}^{=:c_3} f(x) \quad \forall x \geq \underbrace{\max(x_0, x_2)}_{=:x_3}$$

$\Rightarrow k(x) \in O(f(x))$

$$\Rightarrow h(x) + k(x) \leq c_1 f(x) + c_3 f(x) \quad \forall x \geq \max(x_3, x_1)$$

$$= (c_1 + c_3) f(x)$$

$$\Rightarrow \underline{\underline{h(x) + k(x) \in O(f(x))}}$$

Aufgabe 1b

Behauptung: Es gilt

$$f_E < f_B < f_D < f_F < f_A < f_C$$

(i) $f_A < f_C$

$$\lim_{x\to\infty} \frac{3^x}{x^x} \overset{\text{für } x \geq 6}{\leq} \lim_{x\to\infty} \frac{3^x}{6^x} = \lim_{x\to\infty} \left(\frac{1}{2}\right)^x \overset{\text{geometrische Folge}}{=} \underline{\underline{0}}$$

(ii) $f_F < f_A$:

$$\lim_{x\to\infty} \frac{x^3+12x^2+200x+999}{3^x} \overset{\text{l'Hopital}}{=} \lim_{x\to\infty} \frac{\frac{d^4 f_F}{dx^4}}{\frac{d^4 f_A}{dx^4}} = \lim_{x\to\infty} \frac{0}{\log^4(3)} \underline{\underline{=0}}$$

da (es gilt $\frac{d^4 3^x}{dx^4} = \log^4(3)$)

(iii) $f_D < f_F$

$$\lim_{x\to\infty} \frac{x \log_2 x}{x^3} = \lim_{x\to\infty} \frac{\log_2 x}{x^2} \overset{\text{l'hopital}}{=} \lim_{x\to\infty} \frac{1}{x \cdot 2x^2 \ln(2)} = \underline{\underline{0}}$$

$$\left(\log_2 x = \frac{\ln x}{\ln(2)}\right)$$

(iv) $f_B < f_D$

$$\lim_{x\to\infty} \frac{\sqrt{x}+\log_2 x}{x \log_2 x} = \lim_{x\to\infty} \frac{\frac{\sqrt{x}}{\log_2 x}+1}{x} \leq \lim_{x\to\infty} \frac{\sqrt{x}+1}{x} \leq \lim_{x\to\infty} \frac{\sqrt{x}+\sqrt{x}}{x}$$

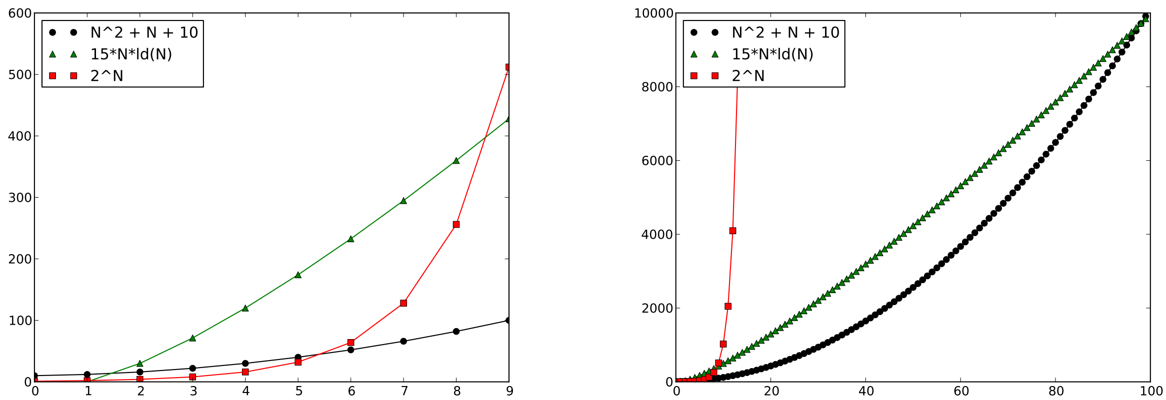
$$= \lim_{x\to\infty} \frac{2}{\sqrt{x}} = \underline{\underline{0}}$$

4.2

Die Laufzeiten der Algorithmen sind in Tabelle 1 in den jeweiligen Intervallen in aufsteigender Reihenfolge sortiert und in Abb. 1 in den Intervallen 0...9 bzw. 0...100 graphisch dargestellt. In Tabelle 2 sind die entsprechenden asymptotischen Komplexitäten β_i dargestellt. In Abb. 2 sind die β_i graphisch dargestellt. Hier wird ersichtlich, dass asymptotische Komplexität und Laufzeit erst im Intervall 98... übereinstimmen, d.h. für große N, daher auch der Name asymptotische Komplexität.

Intervall	1	2	3
0 ... 1	$\alpha_2(N)$	$\alpha_3(N)$	$\alpha_1(N)$
1 ... 5	$\alpha_3/N)$	$\alpha_1(N)$	$\alpha_2(N)$
5 ... 8	$\alpha_1(N)$	$\alpha_3(N)$	$\alpha_2(N)$
8 ...98	$\alpha_1(N)$	$\alpha_2(N)$	$\alpha_3(N)$
98...	$\alpha_2(N)$	$\alpha_1(N)$	$\alpha_3(N)$

Table 1: Intervalle und Laufzeiten der Algorithmen in aufsteigender Reihenfolge (vgl. Abb. 1)



(a) Intervall 0...9 **(b)** Intervall 0...100

Figure 1: Laufzeiten der Algorithmen für die ersten 9 bzw. 100 N.

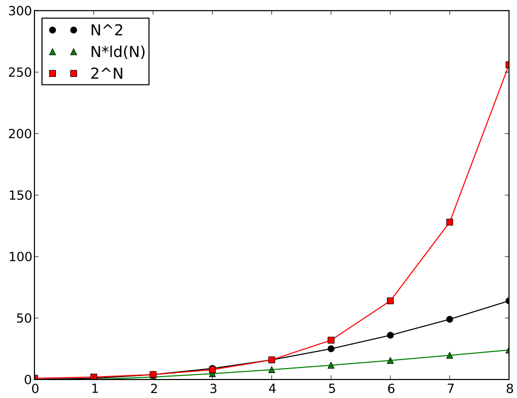
4.3

Für jede Primzahl p muss man $\frac{N}{p}$ Labels setzen (Vielfache der Primzahl als Nichtprim markieren). Hinzu kommen $\sqrt{N}-1$ Vergleiche, ob eine Zahl bereits als Nichtprim markiert wurde (die Schleife läuft von 2 bis $\sqrt{N}$. Das führt zum Ergebnis aus der Aufgabenstellung (vgl. Gl. 1. In Abb. 3 wird diese Komplexität auch bestätigt.

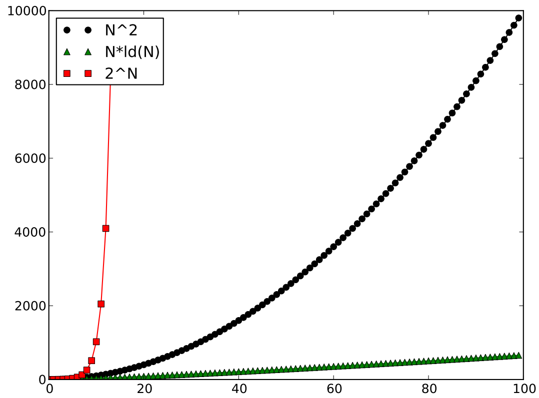
$$O(\sqrt{N} - 2 + \sum_{p \leq N} \frac{N}{p}) \approx O(N \sum_{p \leq N} \frac{1}{p}) = O(N \log \log(N)) \tag{1}$$

Algorithmus	asymptotische Komplexität
$\alpha_1(N)$	$\beta_1(N) = N^2$
$\alpha_2(N)$	$\beta_2(N) = N \log_2(N)$
$\alpha_3(N)$	$\beta_3(N) = 2^N$

Table 2: Algorithmen und zugehörige asymptotische Komplexitäten



(a) Intervall 0...9



(b) Intervall 0...100

Figure 2: Asymptotische Komplexitäten für die ersten 9 bzw. 100 N.

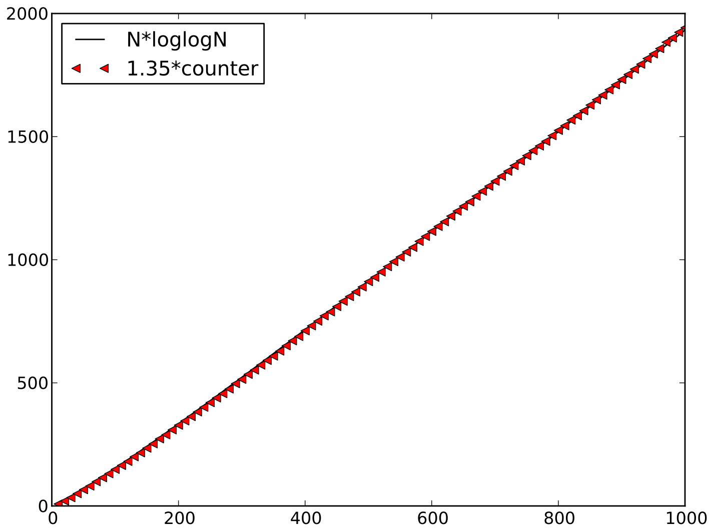


Figure 3: Komplexität und Zählvariable mit einer Konstante multipliziert

4.4.1

Die Zeit, die für append() benötigt wird, ist nicht konstant, das Python Array ist also dynamisch (vgl. Abb. 4).

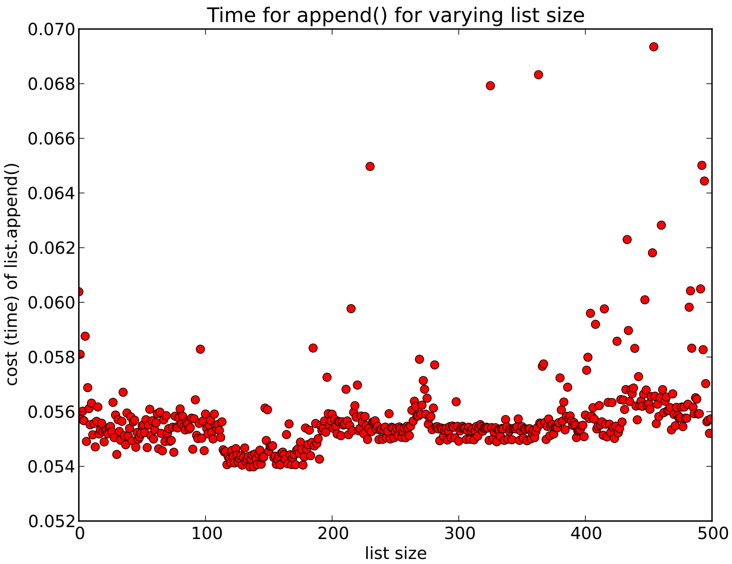


Figure 4: Zeit für append() in Abhängigkeit von der Arraylänge

4.4.2

siehe Datei matrixmulti.py

Bonus

```
int
PyList_Append(PyObject *op, PyObject *newitem)
{
    if (PyList_Check(op) && (newitem != NULL))
        return app1((PyListObject *)op, newitem);
    PyErr_BadInternalCall();
    return -1;
}
```

Falls das anzufügende Element kein Nullzeiger ist, wird app1((PyListObject *)op, newitem) zurückgegeben.

```
static int
app1(PyListObject *self, PyObject *v)
{
    Py_ssize_t n = PyList_GET_SIZE(self);

    assert (v != NULL);
    if (n == PY_SSIZE_T_MAX) {
        PyErr_SetString(PyExc_OverflowError,
            "cannot add more objects to list");
        return -1;
    }

    if (list_resize(self, n+1) == -1)
        return -1;

    Py_INCREF(v);
```

```
    PyList_SET_ITEM(self, n, v);
    return 0;
}
```

Falls die Listenlänge innerhalb des erlaubten Maximums für Python ist (PY_SSIZE_T_MAX), wird die Liste vergrößert (list_resize(self, n+1)). Ob und um wie viel die Liste vergrößert wird, hängt hierbei von der Kapazität und der Belegung der Liste ab. Das Python Array ist also dynamisch.