Inhaltsverzeichnis

1 Tafelanschrieb 12.04.2022

1.1 Exponentialfunktion (o-Notation)

$$
\begin{aligned}
n^l &= o(\exp(n)) \text{ für alle } l > 0 \\
\exp(n) = e^n &= \sum_{j=0}^{\infty} \frac{n^j}{j!} \text{ wobei } j! = 1 \cdot \dots \cdot (j-1) \cdot j = \Pi_{i=1}^{j} i \\
0! &= 1
\end{aligned}
$$

Wähle $k \in \mathbb{N}$, $k > l$ Dann:

$$
\begin{aligned}
\frac{n^l}{\exp(n)} &= \frac{n^l}{\sum_{j=0}^{\infty} \frac{n^j}{j!}} \\
&\leq \frac{n^l}{\frac{n^k}{k!}} \\
&= \frac{n^l(k!)}{n^k} \\
&= k! \cdot n^{\overbrace{l-k}^{<0}} \\
&= \frac{k!}{n^{k-l}} \qquad n \to \infty, \to 0
\end{aligned}
$$

$$
\begin{aligned}
&f(n) = o(y(n)) \text{ falls } \lim_{n\to\infty} \frac{f(n)}{g(n)} = 0 \\
&\to n^l = o(\exp(n))
\end{aligned}
$$

Allgemeiner:

$$
\begin{aligned}
\forall l > 0, c > 1 \\
n^l = o(c^n)
\end{aligned}
$$

Logarithmieren:

$$
\begin{aligned}
\log &= \ln \\
\log x &= \int_1^x \frac{1}{z} dz \\
\log n &= o(n^\alpha) \text{ für jedes } \alpha > 0
\end{aligned}
$$

1.2 Rechnen mit o(g(n)) und O(g(n))

$2^{O(n)}$ heißt $2^{f(n)}$ für irgendeine Funktion $f(n) = O(n)$

$$
f(n) = \frac{1}{n} = O(1)
$$

$$\begin{aligned}
&f(n) = 10 + \sin(\frac{1}{n}) = \Theta(1)\\
&f(n) = o(1) \rightarrow f(n) = O(1)\\
&f(n) = n = \Omega(1)\\
&f(n) = \sin(n^2) \rightarrow \text{ erfüllt nicht } o(1), \Omega(1), \Theta(1)
\end{aligned}$$

$$f(n) \sim g(n) \text{ heißt } \frac{f(n)}{g(n)} \rightarrow_{n\rightarrow n} 1$$

Beispiel:

$$\begin{aligned}
&f(n) = n^2 - 10n + 100\log n\\
&g(n) = n^2\\
&\frac{f(n)}{g(n)} = 1 - \frac{\overbrace{10}^{\rightarrow 0}}{n} + \frac{\overbrace{100\log n}^{\rightarrow 0}}{n^2} \rightarrow_{n\rightarrow\infty} 1
\end{aligned}$$

1.3 Quicksort-Algorithmus

Worst-Case: $\frac{n(n+1)}{2}$

1.4 Permutation erklärt

$$\begin{tikzcd}
1 \arrow[ddr] & 1 \\
2 \arrow[r] & 2 \\
3 \arrow[uur] & 3
\end{tikzcd}$$

1.5 Beweis der Laufzeit vom randomisierten Quicksort

X_n = erwartete # Vergleiche auf einer zufälligen n-Permutation
Es gilt:

$$X_n = \overbrace{n}^{\text{Vrgl. mit Pivot}} + \frac{1}{n}\sum_{k=1}^{n}(X_{k-1} + X_{n-k}) \qquad X_1 = 1 \qquad X_0 = 0$$

k	. . . . . .

$$\begin{aligned}
n \cdot X_n &= n^2 + \sum_{k=1}^{n}(X_{k-1} + X_{n-k})\\
&= n^2 + 2 \cdot \sum_{k=0}^{n-1} X_k
\end{aligned}$$

$$n+1)X_{n+1} = (n+1)^2 + 2 \cdot \sum_{k=0}^{n} X_k$$

$$\begin{aligned}(n+1)X_{n+1} - n \cdot X_n = (n+1)^2 - n^2 + 2X_n \\ = 2n+1+2X_n \\ (n+1)X_{n+1} = 2n+1+(n+2)X_n\end{aligned}$$

$$\begin{aligned}\frac{X_{n+1}}{n+2} &= \frac{2n+1}{(n+1)(n+2)} + \frac{X_n}{n+1} \\ &\leq \frac{2}{n+2} + \frac{X_n}{n+1}\end{aligned}$$

$$\begin{aligned}\frac{X_{n+1}}{n+2} &\leq \frac{2}{n+2} + \frac{X_n}{n+1} \\ &\leq \frac{2}{n+2} + \frac{2}{n+1} + \frac{X_{n-1}}{n} \\ &\leq \sum_{k=0}^{n+1} \frac{2}{k+1} + \underbrace{\frac{X_0}{1}}_{=0} = 2H_{n+1} \\ &\Rightarrow X_{n+1} \leq 2(n+2)H_{n+1} \, \forall n = 2H_{n+1}\end{aligned}$$

$$\begin{aligned}H_n &\geq \log(n) \\ &\geq H_n - 1 \\ H_n &= \log(n) + O(1)\end{aligned}$$

<u>Quicksort:</u>

$$2(n+1)H_n = 2(n+1)\log(n) + O(n) \sim 2n\log(n) = \underline{\underline{O(n \log n)}}$$

2 Tafelanschrieb 14.04.2022

2.1 Quicksort nach Dijkstra in C

Live-Programmierung: Pipe-Operator: |(output des linken Teils wird als Input des rechtens gehandhabt)

shuf mischt die Eingabe

```
seq 10 | shuf
```

Tafel:

1	2	3	4	5	6	7	8	9

$\overset{Input}{\rightarrow}$ A

Wie bekommt man es hin, dass die der Speicher nicht zu groß wird aus der Eingabe heraus?
Erster Vorschlag: $1 \longrightarrow 2 \longrightarrow 3$ Zweiter Vorschlag:
Man erstellt einen Speicher und wenn der vom Platz nicht reicht wird, dieser einfach verdoppelt:

1			
1	2		
1	2	3	4

Live-Programmierung:
Programm für die Speicherallocation bei zu großen Eingaben

```
/* Dieser Code wird ins Vorlesungs Git-Repo hochgeladen */
#include<stdlib.h>
#include<stdio.h>

char *input=NULL;
int length=0;

int read_input(void) {
    int size=0; /* size of memory block */
    int fill=0; /* number characters read */
    int c; /* next character */

    while((c=getchar()) != EOF){
        if(fill>=size){ /* wenn der Speicher nicht reicht */
            int i,newsize=2*size+1;
            char *newinput;

            /*Bestell Speicher in der entsprechenden Groesse */
            if((newinput=(char *)malloc(sizeof(char)*newsize))==NULL)
                return -1;
            for(i=0;i<size;i++) newinput[i]=input[i];
            if(input!=NULL) free(input);
            input=newinput;
            size=newsize;
        }
        input[fill++]=(char)c; /*leg das neue Symbol ab */
    }
    length=fill;
    return length; /* Gibt neue Laenge zurueck */
}

int main(int argc, char **argv){
    int i;

    if(read_input()<=0) return 1;
    for(i=0;i<length;i++) putchar(input[i]);
}
```

Es muss jetzt noch die Eingabe in richtige Integer-Werte übersetzt werden. Dies haben wir an der Stelle leider nicht geschafft. Er bereitet dies zum nächsten Mal vor und zeigt uns anschließend das Ergebnis.

3 Tafelanschrieb 19.04.2022

3.1 Quicksort

Live-Programmierung Wird demnächst bereitgestellt.

Tafelanschrieb Quicksort hat eine Laufzeit von $\Theta(n \log n)$

Vergleichsbasierte Sortieralgorithmen

Laufzeit

$\Theta(n^2)$	$\Theta(n \log n)$
InsertionSort	Quicksort
BubbleSOrt	MargeSort
	Heapsort

Wir werden zeigen:

$$\log(n!) = \Theta(n \vee n)$$

S_n = Menge aller n-Permutationen

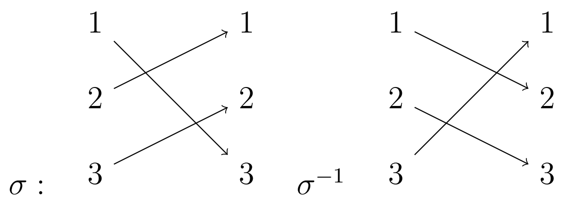
$$\sigma \in S_n$$

$$(\sigma^{-1})^{-1} = \sigma \begin{array}{|c|c|}\hline \sigma(1) & 1 \\ \hline \sigma(2) & 2 \\ \hline \sigma(3) & 3 \\ \hline \vdots & \vdots \\ \hline \sigma(n) & n \\ \hline \end{array} \xrightarrow[\text{nach der ersten Spalte}]{\text{Sortiere}} \begin{array}{|c|c|}\hline 1 & \sigma^{-1}(1) \\ \hline 2 & \sigma^{-1}(2) \\ \hline 3 & \sigma^{-1}(3) \\ \hline \vdots & \vdots \\ \hline n & \sigma^{-1}(n) \\ \hline \end{array}$$

$$\sigma : \{1, \ldots, n\} \to \{1, \ldots, n\}$$

bijektiv

σ^{-1} : Umkehrabbildung



3.2 Beweis des Satzes

Angenommen T ist die erwartete Anzahl von Vergleichen zum Sortieren einer zufälligen n-Permutation. Dann gibt es mindestens $\frac{n!}{2}$ n-Permutationen, für die der Algorithmus $\geq 2 \cdot T$ Vergleiche braucht. Der Algorithmus stellt also $X \leq 2 \cot T$ Vergleichsanfragen.
Die Antworten können durch einen Vektor aus $\{0, +1, -1\}$ beschrieben werden. Weil beim Sortiervorgang keine Information verloren geht, folgt:

$$3^{2T+1} \geq \sum_{X=1}^{2 \cdot T} 3^X \geq \frac{n!}{2} \Rightarrow 2T + 1 \geq \log_3(n!) \Rightarrow T \geq \frac{1}{2} \log_3(n!) - 1 \Rightarrow T = \boxed{\Omega(\log(n!))}$$

$$\log_3 z = \frac{\log(z)}{\log 3}$$

4 Tafelanschrieb 21.04.2022

4.1 Stirlingformel

$\Omega(\log n!)$

$$\begin{aligned}
& n! \sim \sqrt{2\pi n}(\frac{n}{e})^n && \text{Stirling Formel} \\
& \Rightarrow \log n! \sim \log(\sqrt{2\pi n}(\frac{n}{e})^n) && \log(x+y) = \log x + \log y \\
& = \underbrace{\frac{1}{2}\log(2\pi)}_{O(1)} + \underbrace{\log n}_{O(\log n)} + n\log n \underbrace{-n}_{O(n)} && \log 1 = 0 \\
& \sim n \log n
\end{aligned}$$

4.2 Herleitung vom Wallisschen Produkt

<u>Beweis:</u> Übungen

$$\begin{aligned}
& \int_0^{\frac{\pi}{2}} \sin^n x\ dx = \frac{n-1}{n}\int_0^{\frac{\pi}{2}} \sin^{n-2} x\ dx && (n \geq 2) \\
& \int_0^{\frac{\pi}{2}} \sin^{2n} x\ dx = \left[\Pi_{i=1}^n \frac{2i-1}{2i}\right] \cdot \frac{\pi}{2} \\
& \int_0^{\frac{\pi}{2}} \sin^{2n+1} x\ dx = \left[\Pi_{i=1}^n \frac{2i-1}{2i}\right] \underbrace{\int_0^{\frac{\pi}{2}} \sin x\ dx}_{1} = \Pi_{i=1}^n \frac{2i-1}{2i}
\end{aligned}$$

$$\begin{aligned}
& 1 \geq \frac{\int_0^{\frac{\pi}{2}} \sin^{2n+1} x\ dx}{\int_0^{\frac{\pi}{2}} \sin^{2n} x\ dx} \text{ weil } \sin^{2n+1} x \leq \sin^{2n} x\ \forall x, n \text{ da } \sin x \in [0,1] \text{ für } x \in [0, \frac{\pi}{2}] \\
& \geq \frac{\int_0^{\frac{\pi}{2}} \sin^{2n+2} x\ dx}{\int_0^{\frac{\pi}{2}} \sin^{2n} x\ dx} = \frac{\frac{\pi}{2} \cdot \Pi_{i=1}^{n+1} \frac{2i-1}{2i}}{\frac{\pi}{2} \cdot \Pi_{i=1}^{n} \frac{2i-1}{2i}} = \frac{2n+1}{2n+2} \xrightarrow[n\to\infty]{} 1 \\
& \Rightarrow \lim_{n\to\infty} \frac{\int_0^{\frac{\pi}{2}} \sin^{2n+1} x\ dx}{\int_0^{\frac{\pi}{2}} \sin^{2n} x\ dx} = 1 \\
& \Rightarrow 1 = \lim_{n\to\infty} \frac{\Pi_{i=1}^n \frac{2}{2i+1}}{\frac{\pi}{2}\Pi_{i=1}^n \frac{2i-1}{2i}} = \frac{2}{\pi} \lim_{n\to\infty} \Pi_{i=1}^n \frac{4i^2}{4i^2-1} && \square
\end{aligned}$$

4.3 Korollar zur Sterlinformel

Beweis:

$$\text{Wallis } \Rightarrow \frac{\pi}{2} = \lim_{n\to\infty} \Pi_{i=1}^{n} \frac{4i^2}{4i^2-1} = \lim_{n\to\infty} \Pi_{i=1}^{n} \frac{2i}{2i+1}\frac{2i}{2i-1}$$
$$= \lim_{n\to\infty} \frac{1}{2n+1} \Pi_{i=1}^{n} \left(\frac{2i}{2i-1}\right)^2 = \lim_{n\to\infty} \frac{1}{2n} \Pi_{i=1}^{n} \left(\frac{2i}{2i-1}\right)^2$$

$$\Rightarrow \boxed{\sqrt{\pi n}} \sim \Pi_{i=1}^{n} \frac{2i}{2i-1} = \boxed{\frac{\Pi_{i=1}^{n} 2i}{\Pi_{i=1}^{n} 2i-1}}$$
$$\boxed{\Pi_{i=1}^{n} 2i = 2n^2(n!)}$$
$$\Pi_{i=1}^{n} 2i-1 = \frac{\Pi_{i=1}^{n} i}{\Pi_{i=1}^{n} 2i} = \boxed{\frac{(2n)!}{2^n n!}} \qquad \square$$

4.4 Sterling

$$n! \sim \sqrt{2\pi n}(\frac{n}{e})^n$$

Beweis:
Definiere $a_n = \frac{\sqrt{n}(\frac{n}{e})^n}{n!}$ und $b_n = a_n \cdot \exp(\frac{1}{12n}) \geq a_n$

$$\log a_n = \frac{1}{2} \cdot \log n - n + \log \frac{n^n}{n!} = \frac{1}{2} \cdot \log n - n + \sum_{i=1}^{n} \log \frac{n}{i}$$

$$\log b_n = \log a_n + \frac{1}{12n} = \frac{1}{2} \log n - n + \sum_{i=1}^{n} \log \frac{n}{i} + \frac{1}{12n}$$

$$\log a_{n+1} - \log a_n = \frac{1}{2} \log(n+1) - (n+1) + \sum_{i=1}^{n+\cancel{1}} \log \frac{n+i}{i}$$
$$- \left(\frac{1}{2} \log n - n + \sum_{i=1}^{n} \log \frac{n}{i}\right)$$
$$= \frac{1}{2} \log \frac{n+1}{n} - 1 + n \log \frac{n+1}{n} = \left(n + \frac{1}{2}\right) \log \frac{n+1}{n} - 1$$

$$\log b_{n+1} - \log b_n = \log a_{n+1} - \log a_n + \frac{1}{12n} - \frac{1}{12(n+1)} = \left(\frac{1}{2} + n\right) \log \frac{n+1}{n} - 1 - \frac{1}{12n(n+1)}$$

HIlfsfunktionen:

$$\phi(x) = \frac{1}{2}\log\frac{1+x}{1-x} - x \qquad\qquad \psi(x) = \phi(x) - \frac{x^3}{3(1-x^2)}$$
$$\phi(0) = 0 \qquad\qquad \psi(0) = 0$$

$$\phi'(x) = \frac{x^2}{1-x^2} \geq 0 \quad (0 \leq x < 1) \qquad\qquad \psi'(x) = -\frac{x^4}{6(1-x^2)^2} \leq 0 \quad (0 \leq x < 1)$$
$$\Rightarrow \phi(x) \geq 0 \quad \text{für } 0 \leq x \leq 1 \qquad\qquad \psi(x) \leq 0 \quad \text{für } 0 \leq x \leq 1$$

Einsetzen:

$$x = \frac{1}{2n+1}$$

Anmerkung:

$$\begin{aligned}
&\Rightarrow 0 \leq \frac{1}{2}\log\frac{1+x}{1-x} - x \leq \frac{x^3}{3(1-x^2)} \\
&0 \leq \frac{1}{2}\log\frac{1+\frac{1}{2n+1}}{1-\frac{1}{2n+1}} - \frac{1}{2n+1} \leq \frac{\left(\frac{1}{2n+1}\right)^3}{3\left(1-\frac{1}{2n+1}\right)^2} \\
&= \frac{1}{2}\log\frac{n+1}{n} - \frac{1}{2n+1} \leq \frac{1}{12n(n+1)(2n+1)} \\
&\Rightarrow 0 \leq (n+\frac{1}{2})\log\frac{n+1}{n} - 1 \leq \frac{1}{12n(n+1)}
\end{aligned}$$

Daraus folgt

$$\begin{aligned}
&\Rightarrow \log a_{n+1} \geq \log a_n \\
&\log bn + 1 \leq b_n \\
&\Rightarrow a_{n+1} \geq a_n \\
&b_{n+1} \leq b_n
\end{aligned}$$

Also wissen wir:

$$a_n \leq a_{n+1} \leq b_{n+1} \leq b_n \text{ und}$$
$$\lim_{n\to\infty}\frac{a_n}{b_n} = \lim_{n\to\infty}\exp\left(-\frac{1}{12n}\right) = 1$$

Also existiert

$$x = \lim_{n\to\infty} a_n \text{ und } c = \lim_{n\to\infty} b_n$$
$$\Rightarrow \lim_{n\to\infty}\frac{\sqrt{n}\left(\frac{n}{e}\right)^n}{n!} = c > 0$$

Gerade Werte einsetzen:

$$
\begin{aligned}
0 < c = \lim_{n\to\infty} a_{2n} = \lim_{n\to\infty} \frac{\sqrt{2n}\left(\frac{2n}{e}\right)^{2n}}{(2n)!} &= \sqrt{2} \lim_{n\to\infty} \underbrace{\frac{n!^2 2^{2n}}{(2n)!\sqrt{n}}}_{\to\sqrt{n}} \sqrt{n} \left(\underbrace{\sqrt{\pi}\left(\frac{n}{e}\right)^n}_{a_n} \right)^2 \\
&= \sqrt{2\pi} \lim_{n\to\infty} a_n^2 = \sqrt{2\pi} \cdot c^2 \\
\Rightarrow c = \frac{1}{\sqrt{2\pi}} & \qquad \square
\end{aligned}
$$

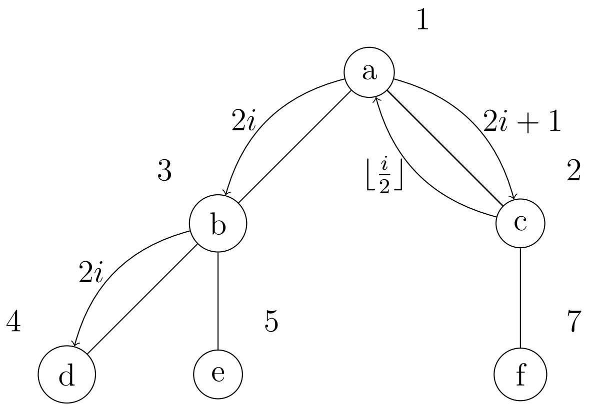
$$
\begin{aligned}
\log a_n = \log \frac{\sqrt{n}\left(\frac{n}{e}\right)^n}{n!} &= \log\sqrt{n} + \log\left(\left(\frac{n}{e}\right)^n\right) - \log(n!) \\
&= \frac{1}{2}\log n + n \log\left(\frac{n}{e}\right) - \log \Pi_{i=1}^n i \\
&= \frac{1}{2}\log n + n(\log n - \underbrace{\log e}_{1}) - \sum_{i=1}^n \log i \\
&= \frac{1}{2}\log n - n + \sum_{i=1}^n \log\frac{n}{i}
\end{aligned}
$$

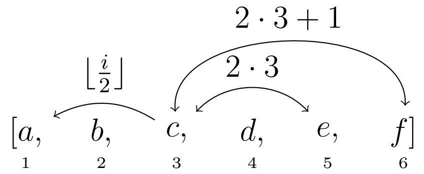
4.5 Pascallsches Dreieck

$$
\begin{array}{ccccccccc}
 & & & & 1 & & & & \\
 & & & 1 & & 1 & & & \\
 & & 1 & & 2 & & 1 & & \\
 & 1 & & 3 & & 3 & & 1 & \\
1 & & 4 & & 6 & & 4 & & 1
\end{array}
$$

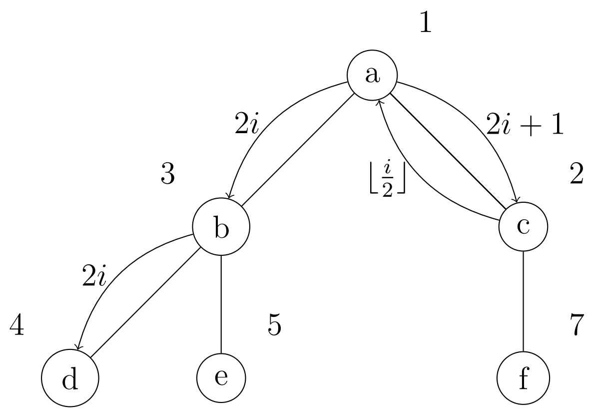
5 Tafelanschrieb 26.04.2022

5.1 Heapsort





$$\log_2(n) = \frac{\log_e(n)}{\log_e(2)}$$



$$\sum_{h=0}^{\lfloor \log n \rfloor} \frac{n}{2^{h+1}} \cdot O(h) = O(n \cdot \underbrace{\sum_{h=0}^{\lfloor \log n \rfloor} \frac{h}{2^{h+1}}}_{c}) = O(n)$$

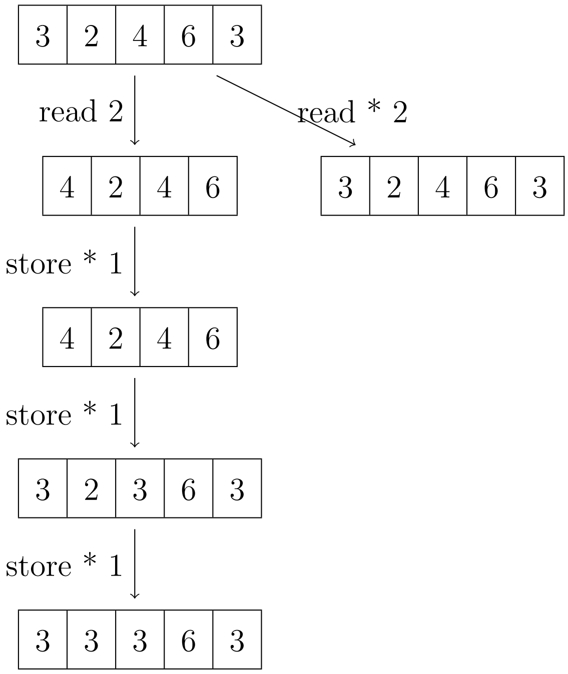
5.2 Weiteres

Es gibt an dieser Stelle keine weiteren Mitschriften zu dieser Vorlesung, da die Tafelanschriften nicht hilfreich sind zum nachlesen.

Unter dem folgenden Link ist der Heapsort Algorithmus nocheinmal erklärt, wie in den Vorlesungen, sowohl in der Baumdarstellung, als auch in der Array-Darstellung.
`wikiversity`
Oder auch in den Folien von DAP2 aus dem Jahr 2019, ist der Algorithmus deutlich erklärt:(Wenn man in den Kurs eingeschrieben ist)
`moodle-tudortmund`

6 Tafelanschrieb 28.04.2022

6.1 Die Registermaschine



6.1.1 Beispiel für jump

```
store 2
load k
jzero 9
add -1
store 1
load 2
add 1
jump 3
halt
```

2	0	0	0

FÜr k setzen wir mal 3 ein.

3	0	2	0

r_0	r_1	r_2	r_3	...	Zähler
2	0	0	0	...	t = 0 (Programmstart)
2	0	2	0	...	z = 1
3	0	2	0	...	z = 2
2	0	2	0	...	z = $\cancel{3}$ 4
2	2	2	0	...	z = 5
2	2	2	0	...	z = 6
2	2	2	0	...	z = 7
3	2	2	0	...	z = 8
3	2	2	0	...	$z = 3 \Rightarrow 4$
2	0	0	0	...	

$$r_0 = r_0 + r_j$$

```
while r_j > 0
    r_0 = r_0 + 1
    r_j  --
```

Die Laufzeit beträgt normal: $O(r_j)$

6.2 Effiziente Algorithmen

$$\begin{aligned}
\mathcal{T}_M(e) &= \begin{cases} |e| & \text{, wenn } e \text{ gerade} \\ c & \text{, sonst} \end{cases} \\
&= \mathcal{T}_{\mathcal{M}}(n) = \max\{\mathcal{T}_{\mathcal{M}}(e) \log_2(2 + |e|) \leq n\} \\
&= O(2^n) \\
&\max\{\mathcal{T}_{\mathcal{M}}(2^k) | k \leq n\} \\
&= \max\{\mathcal{T}_{\mathcal{M}}(2^n), \mathcal{T}_{\mathcal{M}}(2^{n-1})\} \\
&= O(2^n)
\end{aligned}$$

Länge n größte e $|e| = 2^{n-1}$
$\Rightarrow$ Maximum aller e mit $|e| \leq 2^n$

	mit Addition	ohne Addition
$r_0 = r_0 + r_j$	$O(1)$	$O(\lvert r_j \rvert)$
$r_0 = r_0 * r_j$	$O(\lvert r_j \rvert \cdot r_j \rvert)$	$O(1)$
n-mal $r_0 * r_0$ 2^{2^n}		

$$\begin{cases} \text{mult-Befehl:} & O(n) \\ \text{add } r_j & O(2^n) \end{cases}$$