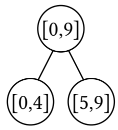
Aufgabe 1

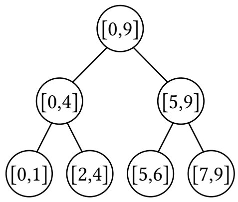
Sortiere $[-5, 13, -32, 7, -3, 17, 23, 12, -35, 19]$ mit ...

- MergeSort
  1. Divide

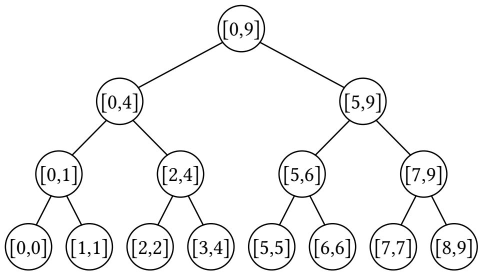
$[-5,13,-32,7,-3,17,23,12,-35,19]$ $\Rightarrow$



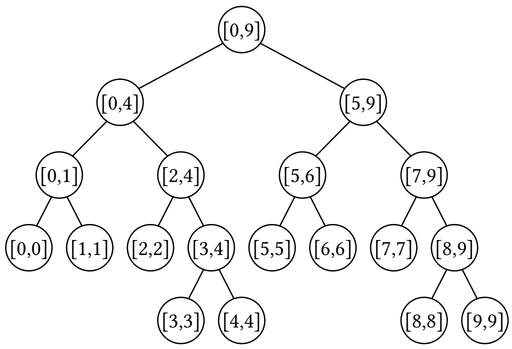
$[-5,13,-32,7,-3,|,17,23,12,-35,19]$ $\Rightarrow$



$[-5,13,|,-32,7,-3,|,17,23,|,12,-35,19]$ $\Rightarrow$

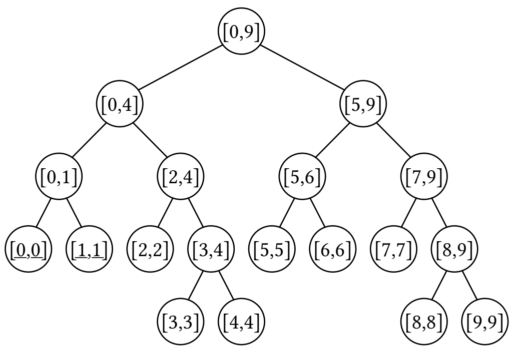


$[-5,|,13,|,-32,|,7,-3,|,17,|,23,|,12,|,-35,19]$ $\Rightarrow$

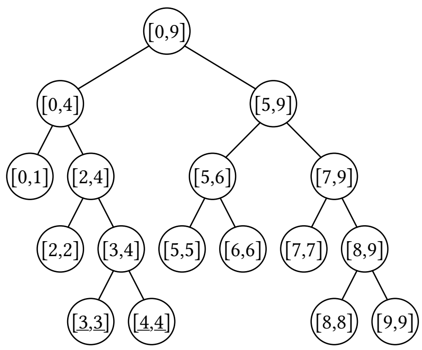


$[-5,|,13,|,-32,|,7,|,-3,|,17,|,23,|,12,|,-35,|,19]$

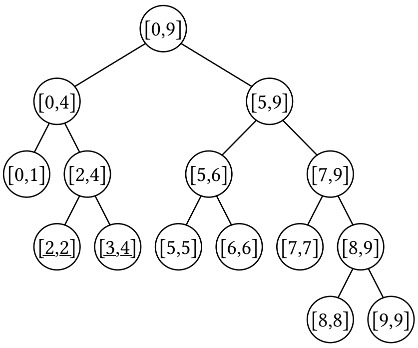
  2. Conquor / Merge



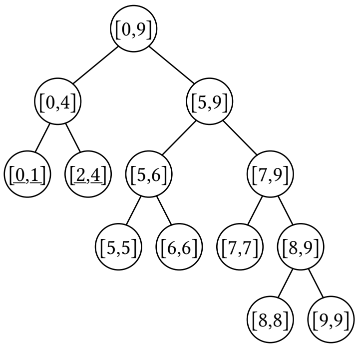
$[-5,|,13,|,-32,|,7,|,-3,|,17,|,23,|,12,|,-35,|,19]$ $\Rightarrow$



$[-5,|,13,|,-32,|,7,|,-3,|,17,|,23,|,12,|,-35,|,19]$ $\Rightarrow$

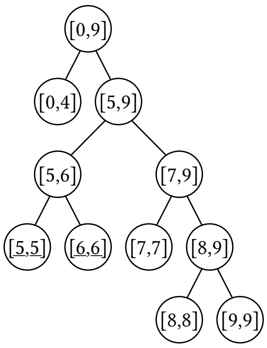


$\Rightarrow$

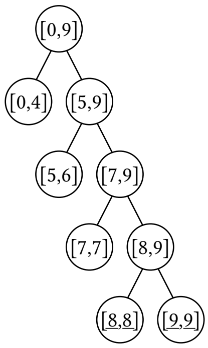


$\Rightarrow$

$[-5,13,|,7,|,-32,-3,|,17,|,23,|,12,|,-35,|,19]$ $\quad$ $[-5,13,|,-32,-3,7,|,17,|,23,|,12,|,-35,|,19]$

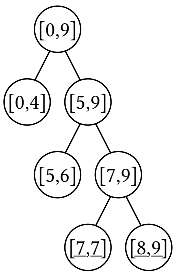


$\Rightarrow$

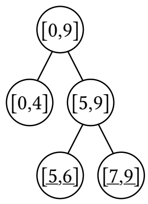


$\Rightarrow$

$[-32,-5,-3,7,13,|,17,|,23,|,12,|,-35,|,19]$ $\quad$ $[-32,-5,-3,7,13,|,17,23,|,12,|,-35,|,19]$

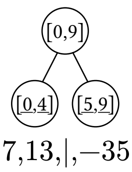


$\Rightarrow$



$\Rightarrow$

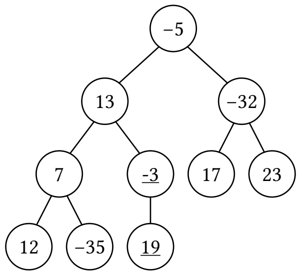
$[-32,-5,-3,7,13,|,17,23,|,12,|,-35,19]$ $\quad$ $[-32,-5,-3,7,13,|,17,23,|,-35,12,19]$



$\Rightarrow$

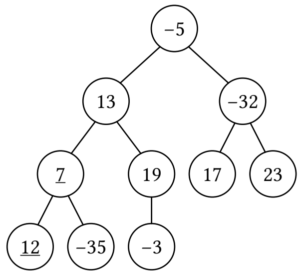
$[-32,-5,-3,7,13,|,-35,12,17,19,23]$ $\quad$ $[-35,-32,-5,-3,7,12,13,17,19,23]$

- HeapSort
  1. Heapify



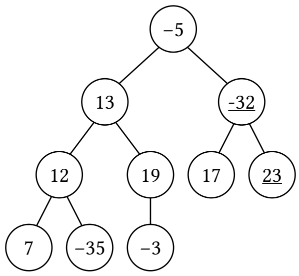
$[-5,13,-32,7,-3,17,23,12,-35,19]$

⇒



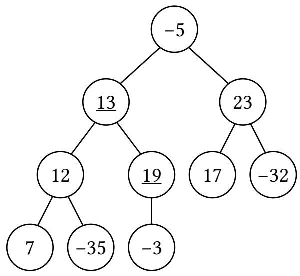
$[-5,13,-32,7,19,17,23,12,-35,-3]$

⇒



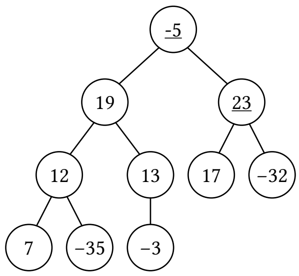
$[-5,13,-32,12,19,17,23,7,-35,-3]$

⇒



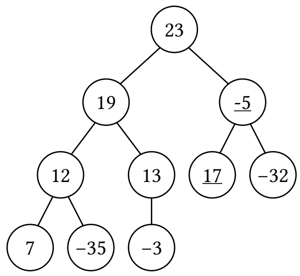
$[-5,13,23,12,19,17,-32,7,-35,-3]$

⇒



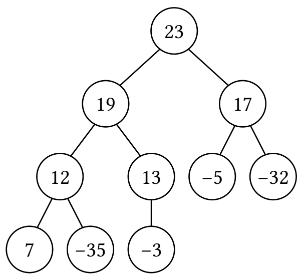
$[-5,19,23,12,13,17,-32,7,-35,-3]$

⇒



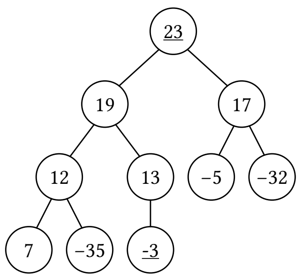
$[23,19,-5,12,13,17,-32,7,-35,-3]$

⇒



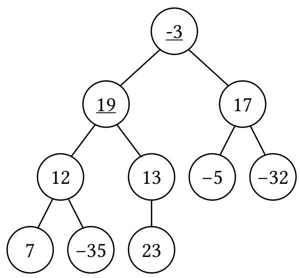
$[23,19,17,12,13,-5,-32,7,-35,-3]$

  2. Sort



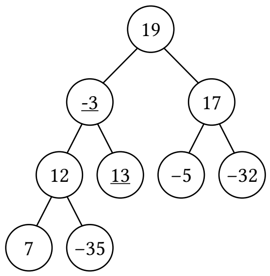
$[23,19,17,12,13,-5,-32,7,-35,-3]$

⇒



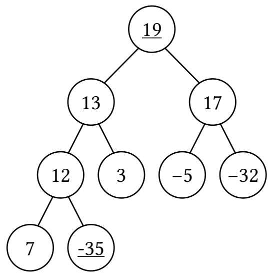
$[-3,19,17,12,13,-5,-32,7,-35,23]$

⇒



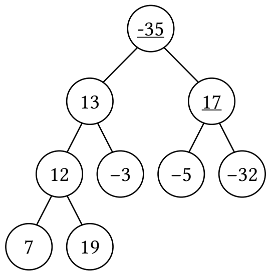
[19,−3,17,12,13,−5,−32,7,−35,23]

⇒



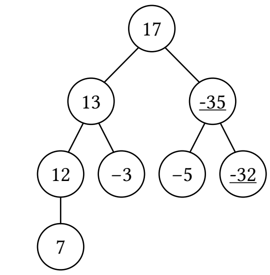
[19,13,17,12,−3,−5,−32,7,−35,23]

⇒



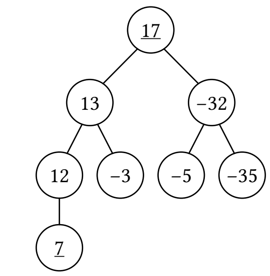
[−35,13,17,12,−3,−5,−32,7,19,23]

⇒



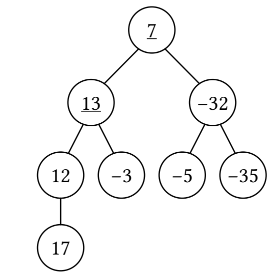
[17,13,−35,12,−3,−5,−32,7,19,23]

⇒



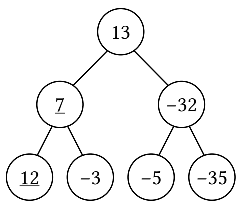
[17,13,−32,12,−3,−5,−35,7,19,23]

⇒



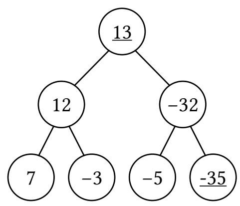
[7,13,−32,12,−3,−5,−35,17,19,23]

⇒



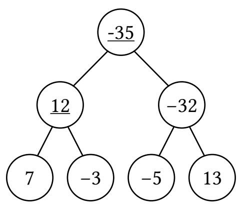
[13,7,−32,12,−3,−5,−35,17,19,23]

⇒



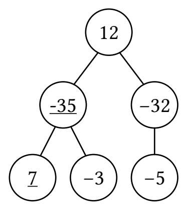
[13,12,−32,7,−3,−5,−35,17,19,23]

⇒



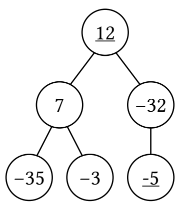
[−35,12,−32,7,−3,−5,13,17,19,23]

⇒



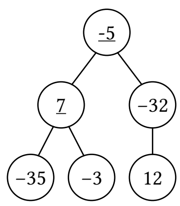
[12,−35,−32,7,−3,−5,13,17,19,23]

⇒



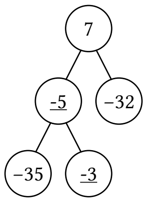
[12,7,−32,−35,−3,−5,13,17,19,23]

⇒

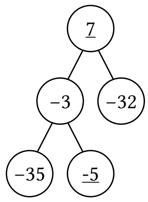


[−5,7,−32,−35,−3,12,13,17,19,23]

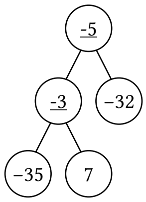
⇒



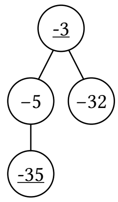
$[7, -5, -32, -35, -3, 12, 13, 17, 19, 23]$ $\Rightarrow$



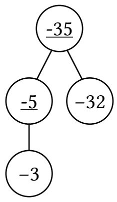
$[7, -3, -32, -35, -5, 12, 13, 17, 19, 23]$ $\Rightarrow$



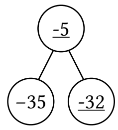
$[-5, -3, -32, -35, 7, 12, 13, 17, 19, 23]$ $\Rightarrow$



$[-3, -5, -32, -35, 7, 12, 13, 17, 19, 23]$ $\Rightarrow$

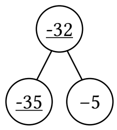


$[-35, -5, -32, -3, 7, 12, 13, 17, 19, 23]$ $\Rightarrow$

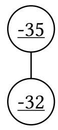


$[-5, -35, -32, -3, 7, 12, 13, 17, 19, 23]$ $\Rightarrow$

,



$[-32, -35, -5, -3, 7, 12, 13, 17, 19, 23]$ $\Rightarrow$



$[-35, -32, -5, -3, 7, 12, 13, 17, 19, 23]$ $\Rightarrow$

$\Rightarrow$

$[-35, -32, -5, -3, 7, 12, 13, 17, 19, 23]$

$\Rightarrow [-35, -32, -5, -3, 7, 12, 13, 17, 19, 23]$

Aufgabe 2

Zeige folgende Aussagen:

1. Ein Heap mit n Elementen hat die Höhe $\lfloor \log n \rfloor$
   Sei $h \in \mathbb{N}$ die Höhe des Heap
   Sei $n_i \in \mathbb{N}_0$ die Anzahl auf der i-ten Ebene
   Es gilt:
   $$\forall i \in \mathbb{N} : 1 \leq n_i \leq 2^i$$
   Da ein Heap linksvoll ist muss auch gelten:
   $$n = \sum_{i=0}^{h} n_i = \sum_{i=0}^{h-1} 2^i + n_h \underset{\text{endl. geom. Reihe}}{=} \frac{2^{h-1+1} - 1}{2-1} + n_h = \frac{2^h - 1}{2-1} + n_h = 2^h - 1 + n_h$$
   $$\Rightarrow n = 2^h + n_h - 1 \Rightarrow 2^h - 1 < n \leq 2^{h+1} - 1 \Rightarrow \#^{\text{min}}_{\text{Höhe h}} = 2^h \wedge \#^{\text{max}}_{\text{Höhe h}} = 2^{h+1} - 1$$
   $$\Rightarrow \log(2^h - 1) < \log n \leq \log(2^{h+1} - 1)$$
   $$\Rightarrow \log(2^h - 1) < \log n \leq \log(2^{h+1} - 1) < \log(2^{h+1}) \Rightarrow \log(2^h - 1) < \log n < \log(2^{h+1})$$
   $$\Rightarrow \log(2^h) \leq \log n < \log(2^{h+1}) \Rightarrow h \leq \log n < h + 1 \Rightarrow h = \lfloor \log n \rfloor$$

2. Ein Heap mit n Elementen hat höchstens $\left\lceil \frac{n}{2^{h+1}} \right\rceil$ viele Knoten der Höhe h
   Sei H ein linksvoller Heap mit n Elementen. Dann gilt: $\max(\#_{\text{Knoten Höhe h}}) = \left\lceil \frac{n}{2^{h+1}} \right\rceil$

   Sei $\text{index}_{\text{Vorgänger}}(i) = \left\lfloor \frac{i-1}{2} \right\rfloor$ und $A(i) = i + 1$ die Anzahl der Elemente bis zu einem Index
   IA: $h = 0$
   $$\#_{\text{Blätter}} = n - A(\text{index}_{\text{Vorgänger}}(n-1)) = n - \left(\left\lfloor \frac{n-2}{2} \right\rfloor + 1 \right)$$
   $$= \begin{cases} n - \left(\frac{n-2}{2} + 1\right), & n \bmod 2 = 0 \\ n - \left(\frac{n-1-2}{2} + 1\right), & n \bmod 2 = 1 \end{cases}$$
   $$= \begin{cases} \frac{n}{2}, & n \bmod 2 = 0 \\ \frac{n}{2} + \frac{1}{2}, & n \bmod 2 = 1 \end{cases} = \left\lceil \frac{n}{2} \right\rceil$$
   IV: angenommen die Behauptung gilt für ein festes aber beliebiges h-1
   IS: $h - 1 \to h$
   Aus dem Heap H wird durch streichen aller Blätter der Heap H'. Beobachtung: alle Knoten in H' die Höhe h-1 haben in H die Höhe h (eigentlich noch durch Induktion Beweisen)
   nach IV:
   $$\#^{\text{H'}}_{\text{Blätter}} = \left\lceil \frac{n - \left\lceil \frac{n}{2} \right\rceil}{2^{(h-1)+1}} \right\rceil = \left\lceil \frac{\left\lfloor \frac{n}{2} \right\rfloor}{2^h} \right\rceil \leq \left\lceil \frac{\frac{n}{2}}{2^h} \right\rceil = \left\lceil \frac{n}{2^{h+1}} \right\rceil \Rightarrow \#_{\text{Knoten mit Höhe h}} \leq \left\lceil \frac{n}{2^{h+1}} \right\rceil$$
   $$\Rightarrow \text{Die Aussage gilt für alle h}$$

3. Für alle x mit $|x| < 1 : \sum_{k=0}^{\infty} kx^k = \frac{x}{(1-x)^2}$

$$\forall x \in \mathbb{R} \land |x| < 1 : \sum_{k=0}^{\infty} kx^k = \frac{x}{(1-x)^2}$$

$$\text{geometrische Reihe} : \sum_{k=0}^{\infty} x^k = \frac{1}{1-x}$$

$$\frac{\partial}{\partial x} \sum_{k=0}^{\infty} x^k = \frac{\partial}{\partial x} \frac{1}{1-x} \Leftrightarrow \sum_{k=0}^{\infty} kx^{k-1} = \frac{1}{(1-x)^2}$$

$$\Rightarrow \forall 0 < |x| < 1 : x \sum_{k=0}^{\infty} kx^{k-1} = x \frac{1}{(1-x)^2} \Leftrightarrow \sum_{k=0}^{\infty} kx^k = \frac{x}{(1-x)^2}$$

Aufgabe 3

$o_{i,j}$ gilt: $o_{i,j} = \sum_{k=1}^{n} (m_{i,k} \cdot n_{k,i})$

1.

- Variante 1:

$$O_{11} = M_{11} \cdot N_{11} + M_{12} \cdot N_{12} = \mathrm{MN1} + \mathrm{MN2} =$$

$$\begin{pmatrix} m_{1,1} & m_{1,2} & \cdots & m_{1,\frac{n}{2}} \\ m_{2,1} & m_{2,2} & \cdots & m_{2,\frac{n}{2}} \\ \vdots & \vdots & \ddots & \vdots \\ m_{\frac{n}{2},1} & \cdots & \cdots & m_{\frac{n}{2},\frac{n}{2}} \end{pmatrix} \cdot \begin{pmatrix} n_{1,1} & m_{1,2} & \cdots & n_{1,\frac{n}{2}} \\ n_{2,1} & m_{2,2} & \cdots & n_{2,\frac{n}{2}} \\ \vdots & \vdots & \ddots & \vdots \\ n_{\frac{n}{2},1} & \cdots & \cdots & n_{\frac{n}{2},\frac{n}{2}} \end{pmatrix} + \begin{pmatrix} m_{1,\frac{n}{2}} & m_{1,\frac{n}{2}+1} & \cdots & m_{1,n} \\ m_{2,\frac{n}{2}} & m_{2,\frac{n}{2}+1} & \cdots & m_{2,n} \\ \vdots & \vdots & \ddots & \vdots \\ m_{\frac{n}{2},\frac{n}{2}} & \cdots & \cdots & m_{\frac{n}{2},n} \end{pmatrix} \cdot \begin{pmatrix} m_{\frac{n}{2}+1,1} & m_{\frac{n}{2}+1,2} & \cdots & m_{\frac{n}{2}+1,\frac{n}{2}} \\ m_{\frac{n}{2}+2,1} & m_{\frac{n}{2}+2,2} & \cdots & m_{\frac{n}{2}+2,\frac{n}{2}} \\ \vdots & \vdots & \ddots & \vdots \\ m_{n,1} & \cdots & \cdots & m_{n,\frac{n}{2}} \end{pmatrix}$$

$$i, j \in \left[1, \frac{n}{2}\right] : \mathrm{MN1}_{i,j} = \sum_{k=1}^{\frac{n}{2}} m_{i,k} \cdot n_{k,i} \land \mathrm{MN2}_{i,j} = \sum_{k=\frac{n}{2}+1}^{n} m_{i,k} \cdot n_{k,i}$$

$$\Rightarrow \mathrm{MN1}_{i,j} + \mathrm{MN2}_{i,j} = \sum_{k=1}^{\frac{n}{2}} m_{i,k} \cdot n_{k,i} + \sum_{k=\frac{n}{2}+1}^{n} m_{i,k} \cdot n_{k,i} = \sum_{k=1}^{n} m_{i,k} \cdot n_{k,i} = o_{i,j}$$

gilt für $O_{1,2}, O_{2,1}, O_{2,2}$ analog

- Variante 2:

$$\begin{aligned}
H_1 &:= (\mathrm{M}_{1,1} + \mathrm{M}_{2,2}) \cdot (\mathrm{N}_{1,1} + \mathrm{N}_{2,2}) \\
H_2 &:= (\mathrm{M}_{2,1} + \mathrm{M}_{2,2}) \cdot \mathrm{N}_{1,1} \\
H_3 &:= \mathrm{M}_{1,1} \cdot (\mathrm{N}_{1,2} - \mathrm{N}_{2,2}) \\
H_4 &:= \mathrm{M}_{2,2} \cdot (\mathrm{N}_{2,1} - \mathrm{N}_{1,1}) \\
H_5 &:= (\mathrm{M}_{1,1} + \mathrm{M}_{1,2}) \cdot \mathrm{N}_{2,2} \\
H_6 &:= (\mathrm{M}_{2,1} - \mathrm{M}_{1,1}) \cdot (\mathrm{N}_{1,1} + \mathrm{N}_{1,2}) \\
H_7 &:= (\mathrm{M}_{1,2} - \mathrm{M}_{2,2}) \cdot (\mathrm{N}_{2,1} + \mathrm{N}_{2,2}) \\
O_{1,1} &:= H_1 + H_4 - H_5 + H_7 \\
O_{1,2} &:= H_3 + H_5 \\
O_{2,1} &:= H_2 + H_4 \\
O_{2,2} &:= H_1 - H_2 + H_3 + H_6
\end{aligned}$$

Umformungen:

$$
\begin{aligned}
O_{1,1} &= H_1 + H_4 - H_5 + H_7 \\
&= \left(\mathrm{M}_{1,1} + \mathrm{M}_{2,2}\right) \cdot \left(\mathrm{N}_{1,1} + \mathrm{N}_{2,2}\right) + \mathrm{M}_{2,2} \cdot \left(\mathrm{N}_{2,1} - \mathrm{N}_{1,1}\right) - \left(\mathrm{M}_{1,1} + \mathrm{M}_{1,2}\right) \cdot \mathrm{N}_{2,2} + \\
&\quad \left(\mathrm{M}_{1,2} - \mathrm{M}_{2,2}\right) \cdot \left(\mathrm{N}_{2,1} + \mathrm{N}_{2,2}\right) \\
&= \mathrm{M}_{1,1} \cdot \mathrm{N}_{1,1} + \mathrm{M}_{1,1} \cdot \mathrm{N}_{2,2} + \mathrm{M}_{2,2} \cdot \mathrm{N}_{1,1} + \mathrm{M}_{2,2} \cdot \mathrm{N}_{2,2} + \mathrm{M}_{2,2} \cdot \mathrm{N}_{2,1} - \mathrm{M}_{2,2} \cdot \mathrm{N}_{1,1} - \\
&\quad \mathrm{M}_{1,1} \cdot \mathrm{N}_{2,2} - \mathrm{M}_{1,2} \cdot \mathrm{N}_{2,2} + \mathrm{M}_{1,2} \cdot \mathrm{N}_{2,1} + \mathrm{M}_{1,2} \cdot \mathrm{N}_{2,2} - \mathrm{M}_{2,2} \cdot \mathrm{N}_{2,1} - \mathrm{M}_{2,2} \cdot \mathrm{N}_{2,2} \\
&= \left(\mathrm{M}_{1,1} \cdot \mathrm{N}_{1,1} + \mathrm{M}_{1,2} \cdot \mathrm{N}_{2,1}\right) + \left(\mathrm{M}_{1,1} \cdot \mathrm{N}_{2,2} - \mathrm{M}_{1,1} \cdot \mathrm{N}_{2,2}\right) + \left(\mathrm{M}_{2,2} \cdot \mathrm{N}_{1,1} - \mathrm{M}_{2,2} \cdot \mathrm{N}_{1,1}\right) \\
&\quad + \left(\mathrm{M}_{2,2} \cdot \mathrm{N}_{2,2} - \mathrm{M}_{2,2} \cdot \mathrm{N}_{2,2}\right) + \left(\mathrm{M}_{2,2} \cdot \mathrm{N}_{2,1} - \mathrm{M}_{2,2} \cdot \mathrm{N}_{2,1}\right) + \left(-\mathrm{M}_{1,2} \cdot \mathrm{N}_{2,2} + \mathrm{M}_{1,2} \cdot \mathrm{N}_{2,2}\right) \\
&= \mathrm{M}_{1,1} \cdot \mathrm{N}_{1,1} + \mathrm{M}_{1,2} \cdot \mathrm{N}_{2,1} \\
O_{1,2} &= H_3 + H_5 \\
&= \mathrm{M}_{1,1} \cdot \left(\mathrm{N}_{1,2} - \mathrm{N}_{2,2}\right) + \left(\mathrm{M}_{1,1} + \mathrm{M}_{1,2}\right) \cdot \mathrm{N}_{2,2} \\
&= \mathrm{M}_{1,1} \cdot \mathrm{N}_{1,2} - \mathrm{M}_{1,1} \cdot \mathrm{N}_{2,2} + \mathrm{M}_{1,1} \cdot \mathrm{N}_{2,2} + \mathrm{M}_{1,2} \cdot \mathrm{N}_{2,2} \\
&= \left(\mathrm{M}_{1,1} \cdot \mathrm{N}_{1,2} + \mathrm{M}_{1,2} \cdot \mathrm{N}_{2,2}\right) - \mathrm{M}_{1,1} \cdot \mathrm{N}_{2,2} + \mathrm{M}_{1,1} \cdot \mathrm{N}_{2,2} \\
&= \mathrm{M}_{1,1} \cdot \mathrm{N}_{1,2} + \mathrm{M}_{1,2} \cdot \mathrm{N}_{2,2} \\
O_{2,1} &= H_2 + H_4 \\
&= \left(\mathrm{M}_{2,1} + \mathrm{M}_{2,2}\right) \cdot \mathrm{N}_{1,1} + \mathrm{M}_{2,2} \cdot \left(\mathrm{N}_{2,1} - \mathrm{N}_{1,1}\right) \\
&= \mathrm{M}_{2,1} \cdot \mathrm{N}_{1,1} + \mathrm{M}_{2,2} \cdot \mathrm{N}_{1,1} + \mathrm{M}_{2,2} \cdot \mathrm{N}_{2,1} - \mathrm{M}_{2,2} \cdot \mathrm{N}_{1,1} \\
&= \mathrm{M}_{2,1} \cdot \mathrm{N}_{1,1} + \mathrm{M}_{2,2} \cdot \mathrm{N}_{2,1} \\
O_{2,2} &= H_1 - H_2 + H_3 + H_6 \\
&= \left(\mathrm{M}_{1,1} + \mathrm{M}_{2,2}\right) \cdot \left(\mathrm{N}_{1,1} + \mathrm{N}_{2,2}\right) - \left(\mathrm{M}_{2,1} + \mathrm{M}_{2,2}\right) \cdot \mathrm{N}_{1,1} + \mathrm{M}_{1,1} \cdot \left(\mathrm{N}_{1,2} - \mathrm{N}_{2,2}\right) + \\
&\quad \left(\mathrm{M}_{2,1} - \mathrm{M}_{1,1}\right) \cdot \left(\mathrm{N}_{1,1} + \mathrm{N}_{1,2}\right) \\
&= \left(\mathrm{M}_{1,1} \cdot \mathrm{N}_{1,1} + \mathrm{M}_{1,1} \cdot \mathrm{N}_{2,2} + \mathrm{M}_{2,2} \cdot \mathrm{N}_{1,1} + \mathrm{M}_{2,2} \cdot \mathrm{N}_{2,2}\right) - \left(\mathrm{M}_{2,1} \cdot \mathrm{N}_{1,1} + \mathrm{M}_{2,2} \cdot \mathrm{N}_{1,1}\right) + \\
&\quad \left(\mathrm{M}_{1,1} \cdot \mathrm{N}_{1,2} - \mathrm{M}_{1,1} \cdot \mathrm{N}_{2,2}\right) + \left(\mathrm{M}_{2,1} \cdot \mathrm{N}_{1,1} + \mathrm{M}_{2,1} \cdot \mathrm{N}_{1,2} - \mathrm{M}_{1,1} \cdot \mathrm{N}_{1,1} - \mathrm{M}_{1,1} \cdot \mathrm{N}_{1,2}\right) \\
&= \mathrm{M}_{1,1} \cdot \mathrm{N}_{1,1} + \mathrm{M}_{1,1} \cdot \mathrm{N}_{2,2} + \mathrm{M}_{2,2} \cdot \mathrm{N}_{1,1} + \mathrm{M}_{2,2} \cdot \mathrm{N}_{2,2} - \mathrm{M}_{2,1} \cdot \mathrm{N}_{1,1} - \mathrm{M}_{2,2} \cdot \mathrm{N}_{1,1} + \\
&\quad \mathrm{M}_{1,1} \cdot \mathrm{N}_{1,2} - \mathrm{M}_{1,1} \cdot \mathrm{N}_{2,2} + \mathrm{M}_{2,1} \cdot \mathrm{N}_{1,1} + \mathrm{M}_{2,1} \cdot \mathrm{N}_{1,2} - \mathrm{M}_{1,1} \cdot \mathrm{N}_{1,1} - \mathrm{M}_{1,1} \cdot \mathrm{N}_{1,2} \\
&= \left(\mathrm{M}_{2,1} \cdot \mathrm{N}_{1,2} + \mathrm{M}_{2,2} \cdot \mathrm{N}_{2,2}\right) + \left(\mathrm{M}_{1,1} \cdot \mathrm{N}_{1,1} - \mathrm{M}_{1,1} \cdot \mathrm{N}_{1,1}\right) + \left(\mathrm{M}_{2,2} \cdot \mathrm{N}_{1,1} - \mathrm{M}_{2,2} \cdot \mathrm{N}_{1,1}\right) + \\
&\quad \left(-\mathrm{M}_{2,1} \cdot \mathrm{N}_{1,1} + \mathrm{M}_{2,1} \cdot \mathrm{N}_{1,1}\right) + \left(\mathrm{M}_{1,1} \cdot \mathrm{N}_{1,2} - \mathrm{M}_{1,1} \cdot \mathrm{N}_{1,2}\right) - \mathrm{M}_{1,1} \cdot \mathrm{N}_{2,2} + \mathrm{M}_{1,1} \cdot \mathrm{N}_{2,2} \\
&= \mathrm{M}_{2,1} \cdot \mathrm{N}_{1,2} + \mathrm{M}_{2,2} \cdot \mathrm{N}_{2,2}
\end{aligned}
$$

2.

für n gelte: $n, k \in \mathbb{N}_0 : n = 2^k$

- Variante 1:
  Matrix Additionen werden durch die Konstante $4n^2$ abgeschätzt

$$\Rightarrow T(n) = 8T\left(\frac{n}{2}\right) + 4n^2$$

mit Master Methode: $4n^2 \in O\left(n^{\log_2 8}\right) \Rightarrow T(n) \in \Theta\left(n^3\right)$

- Variante 2:
  Matrix Additionen werden durch die Konstante $18n^2$ abgeschätzt

$$\Rightarrow 7T\left(\frac{n}{2}\right) + 18n^2$$

$$\text{mit Master Methode: } \; 18n^2 \in O\left(n^{\log_2 7}\right) \Rightarrow T(n) \in \Theta\left(n^{\log_2 7}\right)$$

Aufgabe 4

```cpp
main.cpp:
#include "matrix.hpp"
#include <chrono>
#include <iostream>
#include <climits>

int main() {
    for (int i = 2; i < INT_MAX; i = i * 2) {
        std::cout << i << std::endl;
        Matrix M(i,i);
        Matrix N(i,i);

        M.randomFill();
        N.randomFill();

        auto tempRecStart = std::chrono::system_clock::now();
        Matrix res1 = M.mult(N);
        auto durationRec = std::chrono::system_clock::now() - tempRecStart;

        auto tempNormStart = std::chrono::system_clock::now();
        Matrix res2 = M*N;
        auto durationNorm = std::chrono::system_clock::now() - tempNormStart;

        if (durationRec <= durationNorm) {
            std::cout << "break even: " << i << std::endl;
            break;
        }
    }

}

matrix.hpp:
#pragma once
class Matrix {
private:
    unsigned int m, n;
    int** val;

public:
    Matrix();
    Matrix(unsigned int m, unsigned n);
    Matrix(const Matrix& other);
    ~Matrix();
    void print() const;
    void input();
```

```
    void randomFill();
    Matrix mult(const Matrix&) const;
    void operator=(const Matrix& B);
    Matrix operator+(const Matrix& B) const;
    Matrix operator-(const Matrix& B) const;
    Matrix operator*(const Matrix& B) const;
    const int* operator[](int) const;
    void setCell(unsigned int, unsigned int, int);

    friend void recursiveMult(const Matrix& M, const Matrix& N, Matrix& O);
};

matrix.cpp:
#include "matrix.hpp"
#include <iostream>
#include <string>
#include <random>
#include <climits>
#include <cmath>
#include <exception>

Matrix::Matrix(): m(0), n(0), val(nullptr) {}
Matrix::Matrix(unsigned int m, unsigned int n): m(m), n(n) {
    this->val = new int*[m];
    for (int i = 0; i < m; i++) {
        this->val[i] = new int[n];
        for (int j = 0; j < n; j++)
            this->val[i][j] = 0;
    }
}

Matrix::Matrix(const Matrix& other): m(other.m), n(other.n) {
    this->val = new int*[m];
    for (int i = 0; i < m; i++) {
        this->val[i] = new int[n];
        for (int j = 0; j < n; j++)
            this->val[i][j] = other.val[i][j];
    }
}

Matrix::~Matrix() {
    if (this->val != nullptr) {
        for (int i = 0; i < m; i++)
            delete[] this->val[i];
        delete[] this->val;
    }
```

```
}

void Matrix::print() const {
    std::cout << "--------------------" << std::endl;
    for (int i = 0; i < n; i++) {
        for (int j = 0; j < m; j++)
            std::cout << this->val[i][j] << "\t";
        std::cout << std::endl;
    }
    std::cout << "--------------------" << std::endl;
}

void Matrix::operator= (const Matrix& other) {
    if (val != nullptr)
        delete this->val;
    this->m = other.m;
    this->n = other.n;
    this->val = new int*[m];
    for (int i = 0; i < m; i++) {
        this->val[i] = new int[n];
        for (int j = 0; j < n; j++)
            this->val[i][j] = other.val[i][j];
    }
}

const int* Matrix::operator[] (int index) const {
    if (index < 0 || index >= m)
        throw std::runtime_error("index out of bounds");
    return this->val[index];

}

void Matrix::input() {
    for (int i = 0; i < n; i++) {
        for (int j = 0; j < m; j++) {
            int tmp = 0;

            std::cout << "please give a value for input: (" << i << "/" << j << "):
            ";
            while (true) {
                std::cin >> tmp;
                if (std::cin.fail()) {
                    std::cin.clear();
                    std::cin.ignore(10000000000, '\n');
                    std::cout << "please give a valid value for input: (" << i <<
                    "/" << j << "): ";
```

```
                } else
                    break;
            }

            this->val[i][j] = tmp;
        }
    }
}

void Matrix::randomFill() {
    static std::random_device rd;
    static std::mt19937 gen (rd());
    static std::uniform_int_distribution<int> dist(-100, 100);

    for (int i = 0; i < m; i++) {
        for (int j = 0; j < n; j++) {
            this->val[i][j] = dist(gen);
        }
    }
}

Matrix Matrix::operator+ (const Matrix& other) const {
    if (this->n != other.n || this->m != other.m)
        throw std::runtime_error("cannot add two Matrices that don't have the same
        dimension");
    unsigned int iterations = 0;

    Matrix res(this->m, this->n);
    for (int i = 0; i < this->m; i++) {
        for (int j = 0; j < this->n; j++) {
            iterations++;
            res.val[i][j] = this->val[i][j] + other.val[i][j];
        }
    }
    return res;
}

Matrix Matrix::operator- (const Matrix& other) const {
    if (this->n != other.n || this->m != other.m)
        throw std::runtime_error("cannot add two Matrices that don't have the same
        dimension");
    unsigned int iterations = 0;

    Matrix res(this->m, this->n);
    for (int i = 0; i < this->m; i++) {
        for (int j = 0; j < this->n; j++) {
```

```
            iterations++;
            res.val[i][j] = this->val[i][j] - other.val[i][j];
        }
    }
    return res;
}

Matrix Matrix::operator* (const Matrix&  other) const {
    if (this->n != other.m || this->m != other.n)
        throw std::runtime_error("cannot multiply two Matrices where rows and column
        not match");
    Matrix res(other.m, this->n);
    int iterations = 0;

    for (int row = 0; row < res.n; row++) {
        for (int column = 0; column < res.m; column++) {
            int cij = 0;
            for (int index = 0; index < res.m; index++) {
                iterations++;
                cij += this->val[row][index] * other.val[index][column];
            }
            res.val[row][column] = cij;
        }
    }


    return res;
}

void Matrix::setCell(unsigned int y, unsigned int x, int value) {
    if (y >= m || x >= n)
        throw std::runtime_error("index out bounds error");
    this->val[y][x] = value;
}

void recursiveMult(const Matrix& M, const Matrix& N, Matrix& O) {
    if (M.n == 2) {
        O.setCell(0, 0, M[0][0] * N[0][0] + M[0][1] * N[1][0]);
        O.setCell(0, 1, M[0][0] * N[0][1] + M[0][1] * N[1][1]);
        O.setCell(1, 0, M[1][0] * N[0][0] + M[1][1] * N[1][0]);
        O.setCell(1, 1, M[1][0] * N[0][1] + M[1][1] * N[1][1]);
        return;
    }

    int nHalf = M.n/2;
    Matrix M11(nHalf, nHalf);
```

```
    Matrix M12(nHalf, nHalf);
    Matrix M21(nHalf, nHalf);
    Matrix M22(nHalf, nHalf);
    Matrix N11(nHalf, nHalf);
    Matrix N12(nHalf, nHalf);
    Matrix N21(nHalf, nHalf);
    Matrix N22(nHalf, nHalf);

    for (int i = 0; i < nHalf; i++) {
        for (int j = 0; j < nHalf; j++) {
            M11.setCell(i, j, M[i        ][j        ]);
            M12.setCell(i, j, M[i        ][nHalf + j]);
            M21.setCell(i, j, M[nHalf + i][j        ]);
            M22.setCell(i, j, M[nHalf + i][nHalf + j]);
            N11.setCell(i, j, N[i        ][j        ]);
            N12.setCell(i, j, N[i        ][nHalf + j]);
            N21.setCell(i, j, N[nHalf + i][j        ]);
            N22.setCell(i, j, N[nHalf + i][nHalf + j]);
        }
    }



    Matrix H1(nHalf, nHalf);
    Matrix H2(nHalf, nHalf);
    Matrix H3(nHalf, nHalf);
    Matrix H4(nHalf, nHalf);
    Matrix H5(nHalf, nHalf);
    Matrix H6(nHalf, nHalf);
    Matrix H7(nHalf, nHalf);

    recursiveMult(M11 + M22, N11 + N22, H1);
    recursiveMult(M21 + M22, N11      , H2);
    recursiveMult(M11      , N12 - N22, H3);
    recursiveMult(M22      , N21 - N11, H4);
    recursiveMult(M11 + M12, N22      , H5);
    recursiveMult(M21 - M11, N11 + N12, H6);
    recursiveMult(M12 - M22, N21 + N22, H7);

    for (int i = 0; i < nHalf; i++) {
        for (int j = 0; j < nHalf; j++) {
            O.setCell(i, j, H1[i][j] + H4[i][j] - H5[i][j] + H7[i][j]);
            O.setCell(i        , nHalf + j, H3[i][j] + H5[i][j]);
            O.setCell(nHalf + i, j        , H2[i][j] + H4[i][j]);
            O.setCell(nHalf + i, nHalf + j, H1[i][j] - H2[i][j] + H3[i][j] + H6[i]
            [j]);
```

```
        }
    }
}

Matrix Matrix::mult(const Matrix& other) const {
    if (n == 0 || m == 0 || other.n == 0 || other.m == 0) {
        throw std::runtime_error("cannot multiply empty Matrices");
    }else if (n != other.n || m != other.m || n != m) {
        throw std::runtime_error("cannot multiply non square shaped matrices recursively");
    } else if (std::fmod(log2(n), 1) != 0) {
        throw std::runtime_error("cannot multiply matrices that are not in R^(2^ix2^i) recursively");
    }

    Matrix O(n,n);
    recursiveMult(*this, other, O);

    return O;
}
```