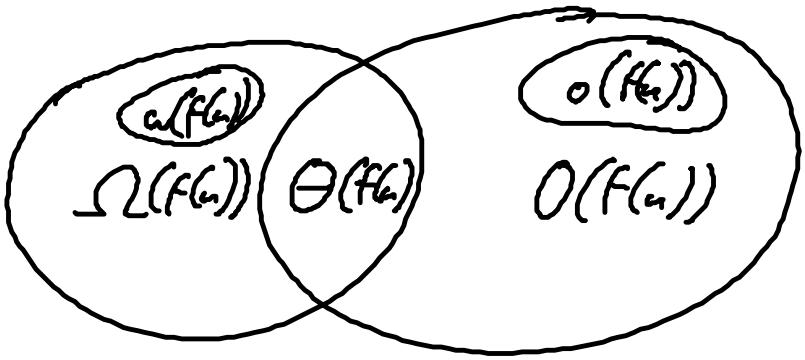
# CoMa-Übung 12, 04.02.13

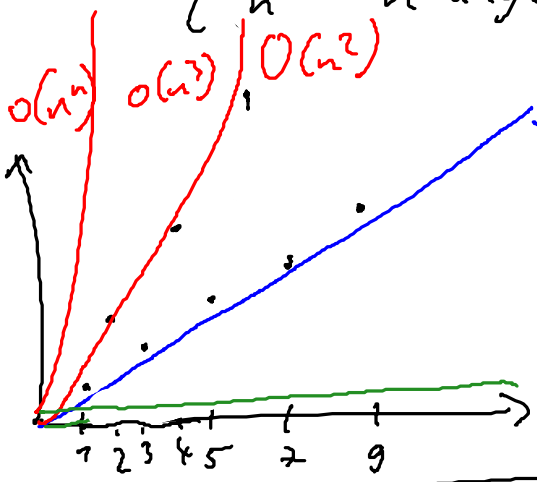
- Anmeldung zu Proberücksprachen
- " " Rücksprachen
- Wiederholung




$$f(n) = \begin{cases} n^2 & n \text{ gerade} \\ n & n \text{ ungerade} \end{cases}$$

$f(n) \in O(n^2)$ ✓

$f(n) \in \Omega(n)$ ✓




## Asymptotisches Wachstum

Asymptotisch langsamer

Konstante Funktionen $f(n) = 15$ → $\Theta(n)$ → z.B. Zugriff auf Array-Element

Logarithmische Funktionen $f(n) = \log n$ → z.B. Binäre Suche → $\Theta(\log n)$

Wurzel-Funktion $f(n) = \sqrt{n} = n^{\frac{1}{2}}$ → $\Theta(n^{\frac{1}{2}})$

Polynome $f(n) = \sum_{i=1}^{k} a_i \cdot n^i \rightarrow \Theta(n^k)$ $\quad n^3 + n^2 + n \rightarrow \Theta(n^3)$

$n^2 + \sqrt{n} \rightarrow \Theta(n^2)$

$= \sum a_i \cdot n^i \quad I \in \mathbb{R}$

$i \in I \searrow \Theta(n^{\max_{i \in I} i})$

Exponentielle Funktionen $f(n) = 2^n \Rightarrow$ Vollständige Enumerierungen

$\Rightarrow \Theta(2^n)$

Kombinatorische Funktionen $f(n) = n! \Rightarrow$ Alle Permutationen enumerieren

Asymptotisch schneller

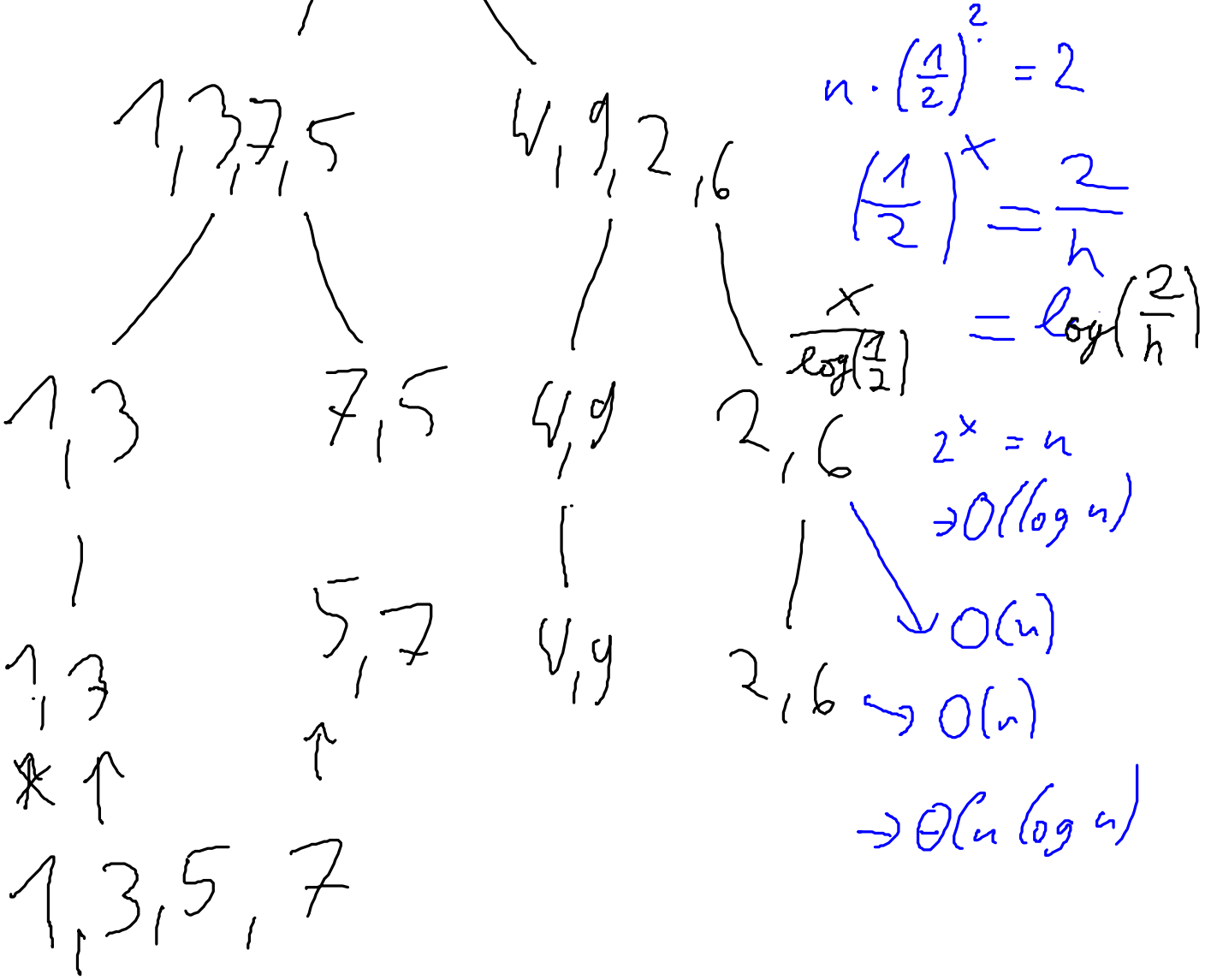
$f \in \Theta(n^2) \Leftrightarrow f \in \Omega(n^2) \wedge f \in O(n^2)$

---

```
public boolean reverse(String s)
{ if (s.length() == 0) return true;
  else
  { if (s.charAt(0) == s.charAt(s.length()-1))
      return (reverse(s.substring(1, s.length()-1));
    else return false; } }
```

1, 3, 7, 5, 4, 9, 2, 6

| \ | | | x




$n \cdot \left(\frac{1}{2}\right)^x = 2$

$\left(\frac{1}{2}\right)^x = \frac{2}{n}$

$x = \log\left(\frac{2}{n}\right)$

$\log\left(\frac{1}{2}\right)$

$2^x = n$

$\Rightarrow O(\log n)$

$O(n)$

$\Rightarrow O(n)$

$\Rightarrow \Theta(n \log n)$

```
public class axis {
private double[][] taxis;
public axis(double[][] i)
{*for(int a=0; a<i.length; a++)
```

```
{ for(int b=0; b<i[0].length; b++)
  { vars[a][b] = i[a][b]; } } }

*= vars = new double[i.length][i[0].length];

public axis()
{ this.axis(new double[2][2]); } }

public class matr extends axis implements mech
{ public matr(double[][] d)*
  { for(int f=0; f<d.length; f++)
    { for(int g=0; g<d[0].length; g++)
```

```
{ if (a[i][j] != a[j][i])
    throw new Exception(); } }
```

* Throws Exception

```
    super(a); }
```

```
public interface math
{
}
```

$F(1) = 3;\ F(2) = 2;\ F(3) = 1;$

$F(n) = 3 \cdot F(n-1) + F(n-2) \cdot F(n-3)$

```
public int recur(int n)
{ if (n == 1) return 3;
```

```
else if (n==2) return 2;
else if (n==3) return 1;
else return (2*recurs(n-1)+recurs(n-2)*recurs(n-3));
}
```

$2n^2 \in O(n^2)$

$f(n) \in O(g(n)) \Leftrightarrow \exists c \in \mathbb{R} \exists n_0 (c * f(n) \leq g(n))$

$\forall n \geq n_0$

$\Rightarrow$ ab dem $n_0$ dominiert das asymptotische Wachstum

$\Rightarrow c \cdot 2n^2 \leq n^2 \quad \forall c \in \mathbb{R}$
$\forall n \geq n_0$

T[]

ArrayList<T>

$\frac{n^3+1}{n^2} = n + \frac{1}{n^2}$

Class<T>

$n + \frac{1}{n^2} \in O(n)$

String.class

Integer.class

LinkedList<T>.class

T.class

public <T> T[] createArray(Class<T> c, int length);