Die asymptotische Notation

$f, g : \mathbb{N} \to \mathbb{R}_{\geq 0}$ seien Funktionen, die einer Eingabelänge $n \in \mathbb{N}$ eine nicht-negative Laufzeit $f(n)$, bzw. $g(n)$ zuweisen.

Die asymptotische Notation

$f, g : \mathbb{N} \to \mathbb{R}_{\geq 0}$ seien Funktionen, die einer Eingabelänge $n \in \mathbb{N}$ eine nicht-negative Laufzeit $f(n)$, bzw. $g(n)$ zuweisen.

- Die Groß-Oh Notation: $f = O(g) \Leftrightarrow$ Es gibt eine positive Konstante $c > 0$ und eine natürliche Zahl $n_0 \in \mathbb{N}$, so dass

$$f(n) \leq c \cdot g(n)$$

für alle $n \geq n_0$ gilt: f wächst höchstens so schnell wie g.

Die asymptotische Notation

$f, g : \mathbb{N} \to \mathbb{R}_{\geq 0}$ seien Funktionen, die einer Eingabelänge $n \in \mathbb{N}$ eine nicht-negative Laufzeit $f(n)$, bzw. $g(n)$ zuweisen.

- Die Groß-Oh Notation: $f = O(g) \Leftrightarrow$ Es gibt eine positive Konstante $c > 0$ und eine natürliche Zahl $n_0 \in \mathbb{N}$, so dass

$$f(n) \leq c \cdot g(n)$$

  für alle $n \geq n_0$ gilt: f wächst höchstens so schnell wie g.
- $f = \Omega(g) \Leftrightarrow g = O(f)$: f wächst mindestens so schnell wie g.

Die asymptotische Notation

$f, g : \mathbb{N} \to \mathbb{R}_{\geq 0}$ seien Funktionen, die einer Eingabelänge $n \in \mathbb{N}$ eine nicht-negative Laufzeit $f(n)$, bzw. $g(n)$ zuweisen.

- Die Groß-Oh Notation: $f = O(g) \Leftrightarrow$ Es gibt eine positive Konstante $c > 0$ und eine natürliche Zahl $n_0 \in \mathbb{N}$, so dass

  $$f(n) \leq c \cdot g(n)$$

  für alle $n \geq n_0$ gilt: f wächst höchstens so schnell wie g.
- $f = \Omega(g) \Leftrightarrow g = O(f)$: f wächst mindestens so schnell wie g.
- $f = \Theta(g) \Leftrightarrow f = O(g)$ und $g = O(f)$:
  f und g wachsen gleich schnell.

Die asymptotische Notation

$f, g : \mathbb{N} \to \mathbb{R}_{\geq 0}$ seien Funktionen, die einer Eingabelänge $n \in \mathbb{N}$ eine nicht-negative Laufzeit $f(n)$, bzw. $g(n)$ zuweisen.

- Die Groß-Oh Notation: $f = O(g) \Leftrightarrow$ Es gibt eine positive Konstante $c > 0$ und eine natürliche Zahl $n_0 \in \mathbb{N}$, so dass

  $$f(n) \leq c \cdot g(n)$$

  für alle $n \geq n_0$ gilt: f wächst höchstens so schnell wie g.
- $f = \Omega(g) \Leftrightarrow g = O(f)$: f wächst mindestens so schnell wie g.
- $f = \Theta(g) \Leftrightarrow f = O(g)$ und $g = O(f)$:
  f und g wachsen gleich schnell.
- Die Klein-Oh Notation: $f = o(g) \Leftrightarrow \lim_{n \to \infty} \frac{f(n)}{g(n)} = 0$:
  f wächst langsamer als g.

Wie schnell dominiert die Asymptotik?

Annahme: Ein einfacher Befehl benötigt 10^{-9} Sekunden.

Wie schnell dominiert die Asymptotik?

Annahme: Ein einfacher Befehl benötigt 10^{-9} Sekunden.

n	n^2	n^3	n^{10}	2^n	$n!$
16	256	4.096	$\geq 10^{12}$	65536	$\geq 10^{13}$
32	1.024	32.768	$\geq 10^{15}$	$\geq 4 \cdot 10^9$	$\geq 10^{31}$
64	4.096	262.144	$\geq 10^{18}$	$\geq 6 \cdot 10^{19}$	
128	16.384	2.097.152	mehr als 10 Jahre	mehr als 600 Jahre	mehr als 10^{14} Jahre
256	65.536	16.777.216			
512	262.144	134.217.728			
1024	1.048.576	$\geq 10^9$			
Million	$\geq 10^{12}$	$\geq 10^{18}$			
	mehr als 15 Minuten	mehr als 10 Jahre			

Grenzwerte

Grenzwerte sollten das Wachstum vorausssagen!

Grenzwerte

Grenzwerte sollten das Wachstum vorausssagen!

Der Grenzwert der Folge $\frac{f(n)}{g(n)}$ existiere und es sei $\lim_{n\to\infty} \frac{f(n)}{g(n)} = c$.

Grenzwerte

Grenzwerte sollten das Wachstum vorausssagen!

Der Grenzwert der Folge $\frac{f(n)}{g(n)}$ existiere und es sei $\lim_{n\to\infty} \frac{f(n)}{g(n)} = c$.

- Wenn $c = 0$, dann ist

Grenzwerte

Grenzwerte sollten das Wachstum vorausssagen!

Der Grenzwert der Folge $\frac{f(n)}{g(n)}$ existiere und es sei $\lim_{n\to\infty} \frac{f(n)}{g(n)} = c$.

- Wenn $c = 0$, dann ist $f = o(g)$:
  f wächst langsamer als g.

Grenzwerte

Grenzwerte sollten das Wachstum vorausssagen!

Der Grenzwert der Folge $\frac{f(n)}{g(n)}$ existiere und es sei $\lim_{n\to\infty} \frac{f(n)}{g(n)} = c$.

- Wenn $c = 0$, dann ist $f = o(g)$:
  f wächst langsamer als g.
- Wenn $0 < c < \infty$, dann ist

Grenzwerte

Grenzwerte sollten das Wachstum vorausssagen!

Der Grenzwert der Folge $\frac{f(n)}{g(n)}$ existiere und es sei $\lim_{n\to\infty} \frac{f(n)}{g(n)} = c$.

- Wenn $c = 0$, dann ist $f = o(g)$:
  f wächst langsamer als g.
- Wenn $0 < c < \infty$, dann ist $f = \Theta(g)$:
  f und g wachsen gleich schnell.

Grenzwerte

Grenzwerte sollten das Wachstum vorausssagen!

Der Grenzwert der Folge $\frac{f(n)}{g(n)}$ existiere und es sei $\lim_{n\to\infty} \frac{f(n)}{g(n)} = c$.

- Wenn $c = 0$, dann ist $f = o(g)$:
  f wächst langsamer als g.
- Wenn $0 < c < \infty$, dann ist $f = \Theta(g)$:
  f und g wachsen gleich schnell.
- Wenn $0 \leq c < \infty$, dann ist

Grenzwerte

Grenzwerte sollten das Wachstum vorausssagen!

Der Grenzwert der Folge $\frac{f(n)}{g(n)}$ existiere und es sei $\lim_{n\to\infty} \frac{f(n)}{g(n)} = c$.

- Wenn $c = 0$, dann ist $f = o(g)$:
  f wächst langsamer als g.
- Wenn $0 < c < \infty$, dann ist $f = \Theta(g)$:
  f und g wachsen gleich schnell.
- Wenn $0 \leq c < \infty$, dann ist $f = O(g)$:
  f wächst höchstens so schnell wie g.

Grenzwerte

Grenzwerte sollten das Wachstum vorausssagen!

Der Grenzwert der Folge $\frac{f(n)}{g(n)}$ existiere und es sei $\lim_{n\to\infty} \frac{f(n)}{g(n)} = c$.

- Wenn $c = 0$, dann ist $f = o(g)$:
  f wächst langsamer als g.
- Wenn $0 < c < \infty$, dann ist $f = \Theta(g)$:
  f und g wachsen gleich schnell.
- Wenn $0 \leq c < \infty$, dann ist $f = O(g)$:
  f wächst höchstens so schnell wie g.
- Wenn $0 < c \leq \infty$, dann ist

Grenzwerte

Grenzwerte sollten das Wachstum vorausssagen!

Der Grenzwert der Folge $\frac{f(n)}{g(n)}$ existiere und es sei $\lim_{n\to\infty} \frac{f(n)}{g(n)} = c$.

- Wenn $c = 0$, dann ist $f = o(g)$:
  f wächst langsamer als g.
- Wenn $0 < c < \infty$, dann ist $f = \Theta(g)$:
  f und g wachsen gleich schnell.
- Wenn $0 \leq c < \infty$, dann ist $f = O(g)$:
  f wächst höchstens so schnell wie g.
- Wenn $0 < c \leq \infty$, dann ist $f = \Omega(g)$:
  f wächst mindestens so schnell wie g.

Die Regel von de l'Hospital

$\lim_{n\to\infty} \frac{f(n)}{g(n)} = \lim_{n\to\infty} \frac{f'(n)}{g'(n)}$, falls der letzte Grenzwert existiert und falls $\lim_{n\to\infty} f(n) = \lim_{n\to\infty} g(n) \in \{0, \infty\}$.

Die Regel von de l'Hospital

$\lim_{n\to\infty} \frac{f(n)}{g(n)} = \lim_{n\to\infty} \frac{f'(n)}{g'(n)}$, falls der letzte Grenzwert existiert und falls $\lim_{n\to\infty} f(n) = \lim_{n\to\infty} g(n) \in \{0, \infty\}$.

- $\log_2 n = o(n)$. Warum?
  - $\lim_{n\to\infty} \log_2 n = \lim_{n\to\infty} n = \infty$,
  - der Grenzwert $\lim_{n\to\infty} \frac{\log_2'(n)}{n'} = \lim_{n\to\infty} \frac{(\log_2 e \ln(n))'}{n'} = \lim_{n\to\infty} \log_2 e \cdot \frac{1/n}{1} = 0$ existiert und
  - $\lim_{n\to\infty} \frac{\log_2(n)}{n} = 0$ folgt mit der Regel von de l'Hospital.

Die Regel von de l'Hospital

$\lim_{n\to\infty} \frac{f(n)}{g(n)} = \lim_{n\to\infty} \frac{f'(n)}{g'(n)}$, falls der letzte Grenzwert existiert und falls $\lim_{n\to\infty} f(n) = \lim_{n\to\infty} g(n) \in \{0, \infty\}$.

- $\log_2 n = o(n)$. Warum?
  - $\lim_{n\to\infty} \log_2 n = \lim_{n\to\infty} n = \infty$,
  - der Grenzwert $\lim_{n\to\infty} \frac{\log_2'(n)}{n'} = \lim_{n\to\infty} \frac{(\log_2 e \ln(n))'}{n'} = \lim_{n\to\infty} \log_2 e \cdot \frac{1/n}{1} = 0$ existiert und
  - $\lim_{n\to\infty} \frac{\log_2(n)}{n} = 0$ folgt mit der Regel von de l'Hospital.
- Weitere Anwendungen:
  - $\log_2 \log_2 n = o(\log_2 n)$.

Die Regel von de l'Hospital

$\lim_{n\to\infty} \frac{f(n)}{g(n)} = \lim_{n\to\infty} \frac{f'(n)}{g'(n)}$, falls der letzte Grenzwert existiert und falls $\lim_{n\to\infty} f(n) = \lim_{n\to\infty} g(n) \in \{0, \infty\}$.

- $\log_2 n = o(n)$. Warum?
  - $\lim_{n\to\infty} \log_2 n = \lim_{n\to\infty} n = \infty$,
  - der Grenzwert $\lim_{n\to\infty} \frac{\log_2'(n)}{n'} = \lim_{n\to\infty} \frac{(\log_2 e \ln(n))'}{n'} = \lim_{n\to\infty} \log_2 e \cdot \frac{1/n}{1} = 0$ existiert und
  - $\lim_{n\to\infty} \frac{\log_2(n)}{n} = 0$ folgt mit der Regel von de l'Hospital.
- Weitere Anwendungen:
  - $\log_2 \log_2 n = o(\log_2 n)$.
  - $\log_2 \log_2 \log_2 n = o(\log_2 \log_2 n)$.

Die Regel von de l'Hospital

$\lim_{n\to\infty} \frac{f(n)}{g(n)} = \lim_{n\to\infty} \frac{f'(n)}{g'(n)}$, falls der letzte Grenzwert existiert und falls $\lim_{n\to\infty} f(n) = \lim_{n\to\infty} g(n) \in \{0, \infty\}$.

- $\log_2 n = o(n)$. Warum?
  - $\lim_{n\to\infty} \log_2 n = \lim_{n\to\infty} n = \infty$,
  - der Grenzwert $\lim_{n\to\infty} \frac{\log_2'(n)}{n'} = \lim_{n\to\infty} \frac{(\log_2 e \ln(n))'}{n'} = \lim_{n\to\infty} \log_2 e \cdot \frac{1/n}{1} = 0$ existiert und
  - $\lim_{n\to\infty} \frac{\log_2(n)}{n} = 0$ folgt mit der Regel von de l'Hospital.
- Weitere Anwendungen:
  - $\log_2 \log_2 n = o(\log_2 n)$.
  - $\log_2 \log_2 \log_2 n = o(\log_2 \log_2 n)$.
  - $\log_2^{(k+1)} n = o(\log_2^{(k)} n)$ für jedes k.

Eine Wachstums-Hierarchie

- $f_1(n) = 1$, dann ist $f_1 = o(\log_2 \log_2 n)$,
  $\lim_{n\to\infty} \log_2 \log_2 n = \infty$.

Eine Wachstums-Hierarchie

- $f_1(n) = 1$, dann ist $f_1 = o(\log_2 \log_2 n)$,
  $\lim_{n\to\infty} \log_2 \log_2 n = \infty$.
- $\log_2 \log_2 n = o(\log_2 n)$,
  wende de l'Hospital an.

Eine Wachstums-Hierarchie

- $f_1(n) = 1$, dann ist $f_1 = o(\log_2 \log_2 n)$,
  $\lim_{n\to\infty} \log_2 \log_2 n = \infty$.
- $\log_2 \log_2 n = o(\log_2 n)$,
  wende de l'Hospital an.
- $\log_2 n = \Theta(\log_a n)$ für jedes $a > 1$,
  $\log_2 n = \log_2 a \cdot \log_a n$.

Eine Wachstums-Hierarchie

- $f_1(n) = 1$, dann ist $f_1 = o(\log_2 \log_2 n)$,
  $\lim_{n\to\infty} \log_2 \log_2 n = \infty$.
- $\log_2 \log_2 n = o(\log_2 n)$,
  wende de l'Hospital an.
- $\log_2 n = \Theta(\log_a n)$ für jedes $a > 1$,
  $\log_2 n = \log_2 a \cdot \log_a n$.
- $\log_2 n = o(n^{1/k})$ für jedes $k > 1$,
  wende de l'Hospital an.

Eine Wachstums-Hierarchie

- $f_1(n) = 1$, dann ist $f_1 = o(\log_2 \log_2 n)$,
  $\lim_{n\to\infty} \log_2 \log_2 n = \infty$.
- $\log_2 \log_2 n = o(\log_2 n)$,
  wende de l'Hospital an.
- $\log_2 n = \Theta(\log_a n)$ für jedes $a > 1$,
  $\log_2 n = \log_2 a \cdot \log_a n$.
- $\log_2 n = o(n^{1/k})$ für jedes $k > 1$,
  wende de l'Hospital an.
- $n^{1/k} = o(n)$ und $n = o(n \cdot \log_2 n)$ für jedes $k > 1$,
  $\lim_{n\to\infty} \frac{n^{1/k}}{n} = \lim_{n\to\infty} \frac{1}{n^{1-1/k}} = 0$ und $\lim_{n\to\infty} \log_2 n = \infty$.

Eine Wachstums-Hierarchie

- $f_1(n) = 1$, dann ist $f_1 = o(\log_2 \log_2 n)$,
  $\lim_{n\to\infty} \log_2 \log_2 n = \infty$.
- $\log_2 \log_2 n = o(\log_2 n)$,
  wende de l'Hospital an.
- $\log_2 n = \Theta(\log_a n)$ für jedes $a > 1$,
  $\log_2 n = \log_2 a \cdot \log_a n$.
- $\log_2 n = o(n^{1/k})$ für jedes $k > 1$,
  wende de l'Hospital an.
- $n^{1/k} = o(n)$ und $n = o(n \cdot \log_2 n)$ für jedes $k > 1$,
  $\lim_{n\to\infty} \frac{n^{1/k}}{n} = \lim_{n\to\infty} \frac{1}{n^{1-1/k}} = 0$ und $\lim_{n\to\infty} \log_2 n = \infty$.
- $n \cdot \log_2 n = o(n^k)$ für jedes $k > 1$.

Eine Wachstums-Hierarchie

- $f_1(n) = 1$, dann ist $f_1 = o(\log_2 \log_2 n)$,
  $\lim_{n\to\infty} \log_2 \log_2 n = \infty$.
- $\log_2 \log_2 n = o(\log_2 n)$,
  wende de l'Hospital an.
- $\log_2 n = \Theta(\log_a n)$ für jedes $a > 1$,
  $\log_2 n = \log_2 a \cdot \log_a n$.
- $\log_2 n = o(n^{1/k})$ für jedes $k > 1$,
  wende de l'Hospital an.
- $n^{1/k} = o(n)$ und $n = o(n \cdot \log_2 n)$ für jedes $k > 1$,
  $\lim_{n\to\infty} \frac{n^{1/k}}{n} = \lim_{n\to\infty} \frac{1}{n^{1-1/k}} = 0$ und $\lim_{n\to\infty} \log_2 n = \infty$.
- $n \cdot \log_2 n = o(n^k)$ für jedes $k > 1$.
- $n^k = o(b^n)$ für jedes $b > 1$,
  $n^k = b^{k \cdot \log_b n}$ und

Eine Wachstums-Hierarchie

- $f_1(n) = 1$, dann ist $f_1 = o(\log_2 \log_2 n)$,
  $\lim_{n\to\infty} \log_2 \log_2 n = \infty$.
- $\log_2 \log_2 n = o(\log_2 n)$,
  wende de l'Hospital an.
- $\log_2 n = \Theta(\log_a n)$ für jedes $a > 1$,
  $\log_2 n = \log_2 a \cdot \log_a n$.
- $\log_2 n = o(n^{1/k})$ für jedes $k > 1$,
  wende de l'Hospital an.
- $n^{1/k} = o(n)$ und $n = o(n \cdot \log_2 n)$ für jedes $k > 1$,
  $\lim_{n\to\infty} \frac{n^{1/k}}{n} = \lim_{n\to\infty} \frac{1}{n^{1-1/k}} = 0$ und $\lim_{n\to\infty} \log_2 n = \infty$.
- $n \cdot \log_2 n = o(n^k)$ für jedes $k > 1$.
- $n^k = o(b^n)$ für jedes $b > 1$,
  $n^k = b^{k \cdot \log_b n}$ und
  $\lim_{n\to\infty} \frac{n^k}{b^n} = \lim_{n\to\infty} \frac{b^{k\cdot\log_b n}}{b^n} = \lim_{n\to\infty} b^{k\cdot\log_b n - n} = 0$.

Eine Wachstums-Hierarchie

- $f_1(n) = 1$, dann ist $f_1 = o(\log_2 \log_2 n)$,
  $\lim_{n\to\infty} \log_2 \log_2 n = \infty$.
- $\log_2 \log_2 n = o(\log_2 n)$,
  wende de l'Hospital an.
- $\log_2 n = \Theta(\log_a n)$ für jedes $a > 1$,
  $\log_2 n = \log_2 a \cdot \log_a n$.
- $\log_2 n = o(n^{1/k})$ für jedes $k > 1$,
  wende de l'Hospital an.
- $n^{1/k} = o(n)$ und $n = o(n \cdot \log_2 n)$ für jedes $k > 1$,
  $\lim_{n\to\infty} \frac{n^{1/k}}{n} = \lim_{n\to\infty} \frac{1}{n^{1-1/k}} = 0$ und $\lim_{n\to\infty} \log_2 n = \infty$.
- $n \cdot \log_2 n = o(n^k)$ für jedes $k > 1$.
- $n^k = o(b^n)$ für jedes $b > 1$,
  $n^k = b^{k \cdot \log_b n}$ und
  $\lim_{n\to\infty} \frac{n^k}{b^n} = \lim_{n\to\infty} \frac{b^{k\cdot\log_b n}}{b^n} = \lim_{n\to\infty} b^{k\cdot\log_b n - n} = 0$.
- Und $b^n = o(n!)$ für jedes $b > 1$.

For-und While-Schleifen

- `for (i=1; i < n ; i++)`
  `vier einfache Anweisungen;`
  - Laufzeit =

For-und While-Schleifen

- `for (i=1; i < n ; i++)`
  `vier einfache Anweisungen;`
  - Laufzeit = $O(n)$.

For-und While-Schleifen

- ```
  for (i=1; i < n ; i++)
    vier einfache Anweisungen;
  ```
  - Laufzeit = $O(n)$.
- ```
  for (i=1; i < n ; i++)
    for (j=i+1; j < n ; j++)
      zwei einfache Anweisungen;
  ```
  - Laufzeit =

For-und While-Schleifen

- ```
  for (i=1; i < n ; i++)
    vier einfache Anweisungen;
  ```
  - Laufzeit = $O(n)$.
- ```
  for (i=1; i < n ; i++)
    for (j=i+1; j < n ; j++)
      zwei einfache Anweisungen;
  ```
  - Laufzeit = $O(n^2)$.

For-und While-Schleifen

- ```
  for (i=1; i < n ; i++)
    vier einfache Anweisungen;
  ```
  - Laufzeit = $O(n)$.
- ```
  for (i=1; i < n ; i++)
    for (j=i+1; j < n ; j++)
      zwei einfache Anweisungen;
  ```
  - Laufzeit = $O(n^2)$.
- `while (n >1) n = 3n/4;`
  - Laufzeit =

For-und While-Schleifen

- ```
  for (i=1; i < n ; i++)
    vier einfache Anweisungen;
  ```
  - Laufzeit = $O(n)$.
- ```
  for (i=1; i < n ; i++)
    for (j=i+1; j < n ; j++)
      zwei einfache Anweisungen;
  ```
  - Laufzeit = $O(n^2)$.
- `while (n >1) n = 3n/4;`
  - Laufzeit = $O(\log_2 n)$.

For-und While-Schleifen

- ```
  for (i=1; i < n ; i++)
    vier einfache Anweisungen;
  ```
  - Laufzeit = $O(n)$.
- ```
  for (i=1; i < n ; i++)
    for (j=i+1; j < n ; j++)
      zwei einfache Anweisungen;
  ```
  - Laufzeit = $O(n^2)$.
- `while (n >1) n = 3n/4;`
  - Laufzeit = $O(\log_2 n)$.
- ```
  while (n >1)
    {for (i=1; i < n; i++)
      {drei einfache Anweisungen;}
    n = 4n/5; }
  ```
  - Laufzeit =
  ```

For-und While-Schleifen

- `for (i=1; i < n ; i++)`
  `vier einfache Anweisungen;`
  - Laufzeit = $O(n)$.
- `for (i=1; i < n ; i++)`
  `for (j=i+1; j < n ; j++)`
  `zwei einfache Anweisungen;`
  - Laufzeit = $O(n^2)$.
- `while (n >1) n = 3n/4;`
  - Laufzeit = $O(\log_2 n)$.
- `while (n >1)`
  `{for (i=1; i < n; i++)`
  `{drei einfache Anweisungen;}`
  `n = 4n/5; }`
  - Laufzeit = $O(n)$.

Anwendungen des Sortierens

Wann soll was sortiert werden?

Anwendungen des Sortierens

Wann soll was sortiert werden?

1. Wie bestimmt man den Durchschnitt von n Intervallen $[a_i, b_i]$ möglichst schnell?

Anwendungen des Sortierens

Wann soll was sortiert werden?

1. Wie bestimmt man den Durchschnitt von n Intervallen $[a_i, b_i]$ möglichst schnell?
2. Wie bestimmt man den Durchschnitt von n Rechtecken $[a_i, b_i] \times [c_i, d_i]$ möglichst schnell?

Anwendungen des Sortierens

Wann soll was sortiert werden?

1. Wie bestimmt man den Durchschnitt von n Intervallen $[a_i, b_i]$ möglichst schnell?
2. Wie bestimmt man den Durchschnitt von n Rechtecken $[a_i, b_i] \times [c_i, d_i]$ möglichst schnell?
3. Wir sollen die Vereinigung von n Intervallen $[a_i, b_i]$ möglichst schnell als eine Vereinigung von disjunkten Intervallen darstellen. Wie?

Anwendungen des Sortierens

Wann soll was sortiert werden?

1. Wie bestimmt man den Durchschnitt von n Intervallen $[a_i, b_i]$ möglichst schnell?
2. Wie bestimmt man den Durchschnitt von n Rechtecken $[a_i, b_i] \times [c_i, d_i]$ möglichst schnell?
3. Wir sollen die Vereinigung von n Intervallen $[a_i, b_i]$ möglichst schnell als eine Vereinigung von disjunkten Intervallen darstellen. Wie?
4. Die k größten Zahlen aus einem unsortierten Array A sollen bestimmt werden.
   - Sollte man A sortieren
   - oder einen Heap aus A bauen und k mal Delete_Max anwenden
   - oder ....?

Anwendungen des Sortierens

Wann soll was sortiert werden?

1. Wie bestimmt man den Durchschnitt von n Intervallen $[a_i, b_i]$ möglichst schnell?
2. Wie bestimmt man den Durchschnitt von n Rechtecken $[a_i, b_i] \times [c_i, d_i]$ möglichst schnell?
3. Wir sollen die Vereinigung von n Intervallen $[a_i, b_i]$ möglichst schnell als eine Vereinigung von disjunkten Intervallen darstellen. Wie?
4. Die k größten Zahlen aus einem unsortierten Array A sollen bestimmt werden.
   - Sollte man A sortieren
   - oder einen Heap aus A bauen und k mal Delete_Max anwenden
   - oder ....?
5. Wir sollen feststellen, ob alle Schlüssel im Array A auch im Array B vorkommen. Wie?

Wasserkrüge

Gegeben sind n blaue und rote Wasserkrüge, wobei

- alle blauen Krüge unterschiedliche Kapazität besitzen
- und es zu jedem blauen Krug genau einen roten Krug gleicher Kapazität gibt.

Finde alle Paare blauer und roter Krüge gleicher Kapazität mit möglichst wenigen Vergleichen.

Wasserkrüge

Gegeben sind n blaue und rote Wasserkrüge, wobei

- alle blauen Krüge unterschiedliche Kapazität besitzen
- und es zu jedem blauen Krug genau einen roten Krug gleicher Kapazität gibt.

Finde alle Paare blauer und roter Krüge gleicher Kapazität mit möglichst wenigen Vergleichen.

- Eine Vergleichsoperation: Fülle einen blauen Krug b mit Wasser und entleere Krug b in einen roten Krug r.
- Wieviele Vergleichsoperationen sind ausreichend?

Sortieren: Zusammenfassung

- Quicksort:
  - Ein schnelles in place Sortierverfahren. Quicksort ist ideal, wenn alle Daten in den Hauptspeicher passen.

Sortieren: Zusammenfassung

- Quicksort:
  - Ein schnelles in place Sortierverfahren. Quicksort ist ideal, wenn alle Daten in den Hauptspeicher passen.
  - Die zufällige Pivotwahl garantiert, dass **jede** Folge von n Zahlen in **erwarteter Zeit** $O(n \cdot \log_2 n)$ sortiert wird.

Sortieren: Zusammenfassung

- Quicksort:
  - Ein schnelles in place Sortierverfahren. Quicksort ist ideal, wenn alle Daten in den Hauptspeicher passen.
  - Die zufällige Pivotwahl garantiert, dass **jede** Folge von n Zahlen in **erwarteter Zeit** $O(n \cdot \log_2 n)$ sortiert wird.
  - Eine Quicksort-Variante löst das Auswahlproblem in Zeit $O(n)$.

Sortieren: Zusammenfassung

- Quicksort:
  - Ein schnelles in place Sortierverfahren. Quicksort ist ideal, wenn alle Daten in den Hauptspeicher passen.
  - Die zufällige Pivotwahl garantiert, dass **jede** Folge von n Zahlen in **erwarteter Zeit** $O(n \cdot \log_2 n)$ sortiert wird.
  - Eine Quicksort-Variante löst das Auswahlproblem in Zeit $O(n)$.
- Mergesort:
  - Sortiert jede Folge von n Zahlen in **worst case Zeit** $O(n \cdot \log_2 n)$.

Sortieren: Zusammenfassung

- Quicksort:
  - Ein schnelles in place Sortierverfahren. Quicksort ist ideal, wenn alle Daten in den Hauptspeicher passen.
  - Die zufällige Pivotwahl garantiert, dass **jede** Folge von n Zahlen in **erwarteter Zeit** $O(n \cdot \log_2 n)$ sortiert wird.
  - Eine Quicksort-Variante löst das Auswahlproblem in Zeit $O(n)$.
- Mergesort:
  - Sortiert jede Folge von n Zahlen in **worst case Zeit** $O(n \cdot \log_2 n)$.
  - Eine nicht-rekursive Variante sortiert n auf einem Externspeicher gespeicherte Zahlen mit höchstens $O(\frac{n}{B} \log_{\frac{M}{B}} \frac{n}{M})$ Speicherzugriffen, solange B Zahlen in einem „Schwung“ vom Externspeicher in den Hauptspeicher der Größe M gebracht werden können.

Sortieren: Zusammenfassung

- Quicksort:
  - Ein schnelles in place Sortierverfahren. Quicksort ist ideal, wenn alle Daten in den Hauptspeicher passen.
  - Die zufällige Pivotwahl garantiert, dass **jede** Folge von n Zahlen in **erwarteter Zeit** $O(n \cdot \log_2 n)$ sortiert wird.
  - Eine Quicksort-Variante löst das Auswahlproblem in Zeit $O(n)$.
- Mergesort:
  - Sortiert jede Folge von n Zahlen in **worst case Zeit** $O(n \cdot \log_2 n)$.
  - Eine nicht-rekursive Variante sortiert n auf einem Externspeicher gespeicherte Zahlen mit höchstens $O(\frac{n}{B} \log_{\frac{M}{B}} \frac{n}{M})$ Speicherzugriffen, solange B Zahlen in einem „Schwung“ vom Externspeicher in den Hauptspeicher der Größe M gebracht werden können.
- Jedes vergleichsorientierte Sortierverfahren benötigt mindestens $\Omega(n \cdot \log_2 n)$ Vergleiche, aber

Sortieren: Zusammenfassung

- Quicksort:
  - Ein schnelles in place Sortierverfahren. Quicksort ist ideal, wenn alle Daten in den Hauptspeicher passen.
  - Die zufällige Pivotwahl garantiert, dass **jede** Folge von n Zahlen in **erwarteter Zeit** $O(n \cdot \log_2 n)$ sortiert wird.
  - Eine Quicksort-Variante löst das Auswahlproblem in Zeit $O(n)$.
- Mergesort:
  - Sortiert jede Folge von n Zahlen in **worst case Zeit** $O(n \cdot \log_2 n)$.
  - Eine nicht-rekursive Variante sortiert n auf einem Externspeicher gespeicherte Zahlen mit höchstens $O(\frac{n}{B} \log_{\frac{M}{B}} \frac{n}{M})$ Speicherzugriffen, solange B Zahlen in einem „Schwung“ vom Externspeicher in den Hauptspeicher der Größe M gebracht werden können.
- Jedes vergleichsorientierte Sortierverfahren benötigt mindestens $\Omega(n \cdot \log_2 n)$ Vergleiche, aber
- Radixsort sortiert n Zahlen aus der Menge $\{0, \ldots, n^L - 1\}$ in Zeit $O(n \cdot L)$ ohne irgendeinen Vergleich auszuführen.