Maple als Visualisierungswerkzeug - WS17/18

Einfache Algorithmen und ihre Implementierung

Sebastian Wack

09. Dezember 2017

Universität des Saarlandes

Gliederung

1. Größter gemeinsamer Teiler
2. Gram-Schmidtsches Orthogonalisierungsverfahren
3. $\mathcal{O}$-Notation
4. Sortieralgorithmen

Größter gemeinsamer Teiler

Größter gemeinsamer Teiler - Definition

- Teiler einer ganzen Zahl: natürliche Zahl, durch die sich x ohne Rest teilen lässt
- $ggT(a, b)$: größte natürliche Zahl, durch die sich a und b teilen lassen
- Beispiele:
  - ggT(2,7)=1
  - ggT(4,16)=4
  - ggT(96,1024) = 32

Größter gemeinsamer Teiler - Implementierung

- naive Implementierung zu aufwendig!

Größter gemeinsamer Teiler - Implementierung

- naive Implementierung zu aufwendig!
- $ggT(a, b)$ ändert sich nicht, wenn man die kleinere von der größeren Zahl abzieht

Größter gemeinsamer Teiler - Implementierung

- naive Implementierung zu aufwendig!
- $ggT(a, b)$ ändert sich nicht, wenn man die kleinere von der größeren Zahl abzieht
- Beispiel: $ggT(7, 12)$

Größter gemeinsamer Teiler - Implementierung

- naive Implementierung zu aufwendig!
- $ggT(a, b)$ ändert sich nicht, wenn man die kleinere von der größeren Zahl abzieht
- Beispiel: $ggT(7, 12)$
  1. $ggT(7, 12) = ggT(7, 5)$, da $a \leq b$

Größter gemeinsamer Teiler - Implementierung

- naive Implementierung zu aufwendig!
- $ggT(a, b)$ ändert sich nicht, wenn man die kleinere von der größeren Zahl abzieht
- Beispiel: $ggT(7, 12)$
  1. $ggT(7, 12) = ggT(7, 5)$, da $a \leq b$
  2. $ggT(7, 5) = ggT(2, 5)$, da $a > b$

Größter gemeinsamer Teiler - Implementierung

- naive Implementierung zu aufwendig!
- $ggT(a, b)$ ändert sich nicht, wenn man die kleinere von der größeren Zahl abzieht
- Beispiel: $ggT(7, 12)$
  1. $ggT(7, 12) = ggT(7, 5)$, da $a \leq b$
  2. $ggT(7, 5) = ggT(2, 5)$, da $a > b$
  3. $ggT(2, 5) = ggT(2, 3)$, da $a \leq b$

Größter gemeinsamer Teiler - Implementierung

- naive Implementierung zu aufwendig!
- $ggT(a, b)$ ändert sich nicht, wenn man die kleinere von der größeren Zahl abzieht
- Beispiel: $ggT(7, 12)$
  1. $ggT(7, 12) = ggT(7, 5)$, da $a \leq b$
  2. $ggT(7, 5) = ggT(2, 5)$, da $a > b$
  3. $ggT(2, 5) = ggT(2, 3)$, da $a \leq b$
  4. $ggT(2, 3) = ggT(2, 1)$, da $a \leq b$

Größter gemeinsamer Teiler - Implementierung

- naive Implementierung zu aufwendig!
- $ggT(a, b)$ ändert sich nicht, wenn man die kleinere von der größeren Zahl abzieht
- Beispiel: $ggT(7, 12)$
  1. $ggT(7, 12) = ggT(7, 5)$, da $a \leq b$
  2. $ggT(7, 5) = ggT(2, 5)$, da $a > b$
  3. $ggT(2, 5) = ggT(2, 3)$, da $a \leq b$
  4. $ggT(2, 3) = ggT(2, 1)$, da $a \leq b$
  5. $ggT(2, 1) = ggT(1, 1)$, da $a > b$

Größter gemeinsamer Teiler - Implementierung

- naive Implementierung zu aufwendig!
- $ggT(a, b)$ ändert sich nicht, wenn man die kleinere von der größeren Zahl abzieht
- Beispiel: $ggT(7, 12)$
  1. $ggT(7, 12) = ggT(7, 5)$, da $a \leq b$
  2. $ggT(7, 5) = ggT(2, 5)$, da $a > b$
  3. $ggT(2, 5) = ggT(2, 3)$, da $a \leq b$
  4. $ggT(2, 3) = ggT(2, 1)$, da $a \leq b$
  5. $ggT(2, 1) = ggT(1, 1)$, da $a > b$
  6. $ggT(1, 1) = ggT(1, 0) = 1$, da $a \leq b$

Größter gemeinsamer Teiler - Implementierung

- naive Implementierung zu aufwendig!
- $ggT(a, b)$ ändert sich nicht, wenn man die kleinere von der größeren Zahl abzieht
- Beispiel: $ggT(7, 12)$
  1. $ggT(7, 12) = ggT(7, 5)$, da $a \leq b$
  2. $ggT(7, 5) = ggT(2, 5)$, da $a > b$
  3. $ggT(2, 5) = ggT(2, 3)$, da $a \leq b$
  4. $ggT(2, 3) = ggT(2, 1)$, da $a \leq b$
  5. $ggT(2, 1) = ggT(1, 1)$, da $a > b$
  6. $ggT(1, 1) = ggT(1, 0) = 1$, da $a \leq b$
- Aufgabe: Implementiere die iterative Variante der „klassischen Version“des größten gemeinsamen Teilers!

Größter gemeinsamer Teiler - euklidischer Algorithmus

- Idee: ersetze Subtraktion durch Division mit Rest:
  - Setze $b = r_0$
  - $a = q_1 * r_0 + r_1$
  - $r_0 = q_2 * r_1 + r_2$
  - $r_1 = q_3 * r_2 + r_3$
  - $\dots$
  - $r_{n-1} = q_{n+1} * r_n + 0$
- $ggT(a, b) = r_n$

Größter gemeinsamer Teiler - euklidischer Algorithmus

- Idee: ersetze Subtraktion durch Division mit Rest:
  - Setze $b = r_0$
  - $a = q_1 * r_0 + r_1$
  - $r_0 = q_2 * r_1 + r_2$
  - $r_1 = q_3 * r_2 + r_3$
  - $\dots$
  - $r_{n-1} = q_{n+1} * r_n + 0$
- $ggT(a, b) = r_n$
- Beispiel: $ggT(1248, 1956)$

Größter gemeinsamer Teiler - euklidischer Algorithmus

- Idee: ersetze Subtraktion durch Division mit Rest:
  - Setze $b = r_0$
  - $a = q_1 * r_0 + r_1$
  - $r_0 = q_2 * r_1 + r_2$
  - $r_1 = q_3 * r_2 + r_3$
  - $\dots$
  - $r_{n-1} = q_{n+1} * r_n + 0$
- $ggT(a, b) = r_n$
- Beispiel: $ggT(1248, 1956)$
  - $1248 = 0 * 1956 + 1248$

Größter gemeinsamer Teiler - euklidischer Algorithmus

- Idee: ersetze Subtraktion durch Division mit Rest:
  - Setze $b = r_0$
  - $a = q_1 * r_0 + r_1$
  - $r_0 = q_2 * r_1 + r_2$
  - $r_1 = q_3 * r_2 + r_3$
  - $\dots$
  - $r_{n-1} = q_{n+1} * r_n + 0$
- $ggT(a, b) = r_n$
- Beispiel: $ggT(1248, 1956)$
  - $1248 = 0 * 1956 + 1248$
  - $1956 = 1 * 1248 + 708$

Größter gemeinsamer Teiler - euklidischer Algorithmus

- Idee: ersetze Subtraktion durch Division mit Rest:
  - Setze $b = r_0$
  - $a = q_1 * r_0 + r_1$
  - $r_0 = q_2 * r_1 + r_2$
  - $r_1 = q_3 * r_2 + r_3$
  - $\dots$
  - $r_{n-1} = q_{n+1} * r_n + 0$
- $ggT(a, b) = r_n$
- Beispiel: $ggT(1248, 1956)$
  - $1248 = 0 * 1956 + 1248$
  - $1956 = 1 * 1248 + 708$
  - $1248 = 1 * 708 + 540$

Größter gemeinsamer Teiler - euklidischer Algorithmus

- Idee: ersetze Subtraktion durch Division mit Rest:
  - Setze $b = r_0$
  - $a = q_1 * r_0 + r_1$
  - $r_0 = q_2 * r_1 + r_2$
  - $r_1 = q_3 * r_2 + r_3$
  - $\dots$
  - $r_{n-1} = q_{n+1} * r_n + 0$
- $ggT(a, b) = r_n$
- Beispiel: $ggT(1248, 1956)$
  - $1248 = 0 * 1956 + 1248$
  - $1956 = 1 * 1248 + 708$
  - $1248 = 1 * 708 + 540$
  - $708 = 1 * 540 + 168$

Größter gemeinsamer Teiler - euklidischer Algorithmus

- Idee: ersetze Subtraktion durch Division mit Rest:
  - Setze $b = r_0$
  - $a = q_1 * r_0 + r_1$
  - $r_0 = q_2 * r_1 + r_2$
  - $r_1 = q_3 * r_2 + r_3$
  - $\dots$
  - $r_{n-1} = q_{n+1} * r_n + 0$
- $ggT(a, b) = r_n$
- Beispiel: $ggT(1248, 1956)$
  - $1248 = 0 * 1956 + 1248$
  - $1956 = 1 * 1248 + 708$
  - $1248 = 1 * 708 + 540$
  - $708 = 1 * 540 + 168$
  - $540 = 3 * 168 + 36$

Größter gemeinsamer Teiler - euklidischer Algorithmus

- Idee: ersetze Subtraktion durch Division mit Rest:
  - Setze $b = r_0$
  - $a = q_1 * r_0 + r_1$
  - $r_0 = q_2 * r_1 + r_2$
  - $r_1 = q_3 * r_2 + r_3$
  - $\dots$
  - $r_{n-1} = q_{n+1} * r_n + 0$
- $ggT(a, b) = r_n$
- Beispiel: $ggT(1248, 1956)$
  - $1248 = 0 * 1956 + 1248$
  - $1956 = 1 * 1248 + 708$
  - $1248 = 1 * 708 + 540$
  - $708 = 1 * 540 + 168$
  - $540 = 3 * 168 + 36$
  - $168 = 4 * 36 + 24$

Größter gemeinsamer Teiler - euklidischer Algorithmus

- Idee: ersetze Subtraktion durch Division mit Rest:
  - Setze $b = r_0$
  - $a = q_1 * r_0 + r_1$
  - $r_0 = q_2 * r_1 + r_2$
  - $r_1 = q_3 * r_2 + r_3$
  - $\dots$
  - $r_{n-1} = q_{n+1} * r_n + 0$
- $ggT(a, b) = r_n$
- Beispiel: $ggT(1248, 1956)$
  - $1248 = 0 * 1956 + 1248$
  - $1956 = 1 * 1248 + 708$
  - $1248 = 1 * 708 + 540$
  - $708 = 1 * 540 + 168$
  - $540 = 3 * 168 + 36$
  - $168 = 4 * 36 + 24$
  - $36 = 1 * 24 + 12$

Größter gemeinsamer Teiler - euklidischer Algorithmus

- Idee: ersetze Subtraktion durch Division mit Rest:
  - Setze $b = r_0$
  - $a = q_1 * r_0 + r_1$
  - $r_0 = q_2 * r_1 + r_2$
  - $r_1 = q_3 * r_2 + r_3$
  - $\dots$
  - $r_{n-1} = q_{n+1} * r_n + 0$
- $ggT(a, b) = r_n$
- Beispiel: $ggT(1248, 1956)$
  - $1248 = 0 * 1956 + 1248$
  - $1956 = 1 * 1248 + 708$
  - $1248 = 1 * 708 + 540$
  - $708 = 1 * 540 + 168$
  - $540 = 3 * 168 + 36$
  - $168 = 4 * 36 + 24$
  - $36 = 1 * 24 + 12$
  - $24 = 2 * 12 + 0 \Rightarrow ggT(1248, 1956) = 12$

Größter gemeinsamer Teiler - euklidischer Algorithmus

- Idee: ersetze die Subtraktion im ggT - Algorithmus durch Division mit Rest:
  - Setze $b = r_0$
  - $a = q_1 * r_0 + r_1$
  - $r_0 = q_2 * r_1 + r_2$
  - $r_1 = q_3 * r_2 + r_3$
  - $\dots$
  - $r_{n-1} = q_{n+1} * r_n + 0$
- $ggT(a, b) = r_n$
- Rekursiver Algorithmus
  - einfach zu implementieren
  - Rekursionsfall: $ggT(a, b) = ggT(b, a\%b)$
- Aufgabe: Implementiere den rekursiven euklidischen Algorithmus!

Gram-Schmidtsches Orthogonalisierungsverfahren

Gram-Schmidtsches Orthogonalisierungsverfahren

- Liste linear unabhängiger Vektoren $v_1, \cdots, v_n$ gegeben
- Berechne Orthogonalsystem, welches gleichen Untervektorraum erzeugt
- Orthogonalsystem für eine Menge M:
  - Nullvektor nicht im OGS enthalten
  - je zwei verschiedene Vektoren aus M sind orthogonal zueinander: $\forall v, w \in M : v \neq w \Rightarrow \langle v, w \rangle = 0$

Maple Demo

Gram-Schmidtsches Orthogonalisierungsv. - Algorithmus

- Eingabe: Liste von $v_1, \cdots, v_n$

Gram-Schmidtsches Orthogonalisierungsv. - Algorithmus

- Eingabe: Liste von $v_1, \cdots, v_n$
- Berechnung:

Gram-Schmidtsches Orthogonalisierungsv. - Algorithmus

- Eingabe: Liste von $v_1, \cdots, v_n$
- Berechnung:
  - $w_1 = v_1$

Gram-Schmidtsches Orthogonalisierungsv. - Algorithmus

- Eingabe: Liste von $v_1, \cdots, v_n$
- Berechnung:
  - $w_1 = v_1$
  - $w_2 = v_2 - \frac{\langle w_1, v_2 \rangle}{\langle w_1, w_1 \rangle} w_1$

Gram-Schmidtsches Orthogonalisierungsv. - Algorithmus

- Eingabe: Liste von $v_1, \cdots, v_n$
- Berechnung:
  - $w_1 = v_1$
  - $w_2 = v_2 - \frac{\langle w_1, v_2 \rangle}{\langle w_1, w_1 \rangle} w_1$
  - $\cdots$
  - $w_n = v_n - \sum_{i=1}^{n-1} \frac{\langle w_i, v_n \rangle}{\langle w_i, w_i \rangle} w_i$

Gram-Schmidtsches Orthogonalisierungsv. - Algorithmus

- Eingabe: Liste von $v_1, \cdots, v_n$
- Berechnung:
  - $w_1 = v_1$
  - $w_2 = v_2 - \frac{\langle w_1, v_2 \rangle}{\langle w_1, w_1 \rangle} w_1$
  - $\cdots$
  - $w_n = v_n - \sum_{i=1}^{n-1} \frac{\langle w_i, v_n \rangle}{\langle w_i, w_i \rangle} w_i$
- Ausgabe: Liste von $w_1, \cdots, w_n$

Gram-Schmidtsches Orthogonalisierungsv. - Algorithmus

- Eingabe: Liste von $v_1, \cdots, v_n$
- Berechnung:
  - $w_1 = v_1$
  - $w_2 = v_2 - \frac{\langle w_1, v_2 \rangle}{\langle w_1, w_1 \rangle} w_1$
  - $\cdots$
  - $w_n = v_n - \sum_{i=1}^{n-1} \frac{\langle w_i, v_n \rangle}{\langle w_i, w_i \rangle} w_i$
- Ausgabe: Liste von $w_1, \cdots, w_n$
- Aufgabe: Implementiere das Gram-Schmidtsche Orthogonalisierungsverfahren in Maple
- Hinweis: Normieren der berechneten Vektoren $w_1, ..., w_n$ liefert das Gram-Schmidtsche Orthonormalisierungsverfahren

Gram-Schmidtsches Orthogonalisierungsv. - Algorithmus

- Eingabe: Liste von $v_1, \cdots, v_n$
- Berechnung:
  - $w_1 = v_1$
  - $w_2 = v_2 - \frac{\langle w_1, v_2 \rangle}{\langle w_1, w_1 \rangle} w_1$
  - $\cdots$
  - $w_n = v_n - \sum_{i=1}^{n-1} \frac{\langle w_i, v_n \rangle}{\langle w_i, w_i \rangle} w_i$
- Ausgabe: Liste von $w_1, \cdots, w_n$
- Aufgabe: Implementiere das Gram-Schmidtsche Orthogonalisierungsverfahren in Maple
- Hinweis: Normieren der berechneten Vektoren $w_1, ..., w_n$ liefert das Gram-Schmidtsche Orthonormalisierungsverfahren
- Aufgabe: Implementiere das Gram-Schmidtsche Orthonormalisierungsverfahren in Maple!

Wiederholung von Listen in Maple

- Deklaration und Initialisierung: $L := [1, 6, 2, 4, 8, 16, 17, 21]$
- Zugreifen auf das i-te Element: $L[i] = 32$
- Anzahl der Elemente: $nops(M)$
- alle Elemente einer Liste: $op(M)$

$\mathcal{O}$-Notation

$\mathcal{O}$-Notation

- beschreibt das „Verhalten“ von Funktionen
- $\mathcal{O}(n)$ erfasst Wachstumsverhalten eines Algorithmus in Abhängigkeit der Eingabegröße n
- formale Definition:

$$f \in \mathcal{O}(g) \Leftrightarrow \limsup_{x \to n} \left| \frac{f(x)}{g(x)} \right| < \infty$$

- Beispiele:
  - n

$\mathcal{O}$-Notation

- beschreibt das „Verhalten“ von Funktionen
- $\mathcal{O}(n)$ erfasst Wachstumsverhalten eines Algorithmus in Abhängigkeit der Eingabegröße n
- formale Definition:

$$f \in \mathcal{O}(g) \Leftrightarrow \limsup_{x \to n} \left| \frac{f(x)}{g(x)} \right| < \infty$$

- Beispiele:
  - $n \in \mathcal{O}(n)$
  - $2n$

$\mathcal{O}$-Notation

- beschreibt das „Verhalten“ von Funktionen
- $\mathcal{O}(n)$ erfasst Wachstumsverhalten eines Algorithmus in Abhängigkeit der Eingabegröße n
- formale Definition:

$$f \in \mathcal{O}(g) \Leftrightarrow \limsup_{x \to n} \left| \frac{f(x)}{g(x)} \right| < \infty$$

- Beispiele:
  - $n \in \mathcal{O}(n)$
  - $2n \in \mathcal{O}(n)$
  - n^4

$\mathcal{O}$-Notation

- beschreibt das „Verhalten“ von Funktionen
- $\mathcal{O}(n)$ erfasst Wachstumsverhalten eines Algorithmus in Abhängigkeit der Eingabegröße n
- formale Definition:

$$f \in \mathcal{O}(g) \Leftrightarrow \limsup_{x \to n} \left| \frac{f(x)}{g(x)} \right| < \infty$$

- Beispiele:
  - $n \in \mathcal{O}(n)$
  - $2n \in \mathcal{O}(n)$
  - $n^4 \in \mathcal{O}(n^4)$
  - $18 * \log_2(n)$

$\mathcal{O}$-Notation

- beschreibt das „Verhalten“ von Funktionen
- $\mathcal{O}(n)$ erfasst Wachstumsverhalten eines Algorithmus in Abhängigkeit der Eingabegröße n
- formale Definition:

$$f \in \mathcal{O}(g) \Leftrightarrow \limsup_{x \to n} \left| \frac{f(x)}{g(x)} \right| < \infty$$

- Beispiele:
  - $n \in \mathcal{O}(n)$
  - $2n \in \mathcal{O}(n)$
  - $n^4 \in \mathcal{O}(n^4)$
  - $18 * \log_2(n) \in \mathcal{O}(\log_2(n))$
  - $n^3 + n^6 + 9$

$\mathcal{O}$-Notation

- beschreibt das „Verhalten“ von Funktionen
- $\mathcal{O}(n)$ erfasst Wachstumsverhalten eines Algorithmus in Abhängigkeit der Eingabegröße n
- formale Definition:

$$f \in \mathcal{O}(g) \Leftrightarrow \limsup_{x \to n} \left| \frac{f(x)}{g(x)} \right| < \infty$$

- Beispiele:
  - $n \in \mathcal{O}(n)$
  - $2n \in \mathcal{O}(n)$
  - $n^4 \in \mathcal{O}(n^4)$
  - $18 * \log_2(n) \in \mathcal{O}(\log_2(n))$
  - $n^3 + n^6 + 9 \in \mathcal{O}(n^6)$
  - 2^{n+3}

$\mathcal{O}$-Notation

- beschreibt das „Verhalten“ von Funktionen
- $\mathcal{O}(n)$ erfasst Wachstumsverhalten eines Algorithmus in Abhängigkeit der Eingabegröße n
- formale Definition:

$$f \in \mathcal{O}(g) \Leftrightarrow \limsup_{x \to n} \left| \frac{f(x)}{g(x)} \right| < \infty$$

- Beispiele:
  - $n \in \mathcal{O}(n)$
  - $2n \in \mathcal{O}(n)$
  - $n^4 \in \mathcal{O}(n^4)$
  - $18 * \log_2(n) \in \mathcal{O}(\log_2(n))$
  - $n^3 + n^6 + 9 \in \mathcal{O}(n^6)$
  - $2^{n+3} = 2^3 * 2^n$

$\mathcal{O}$-Notation

- beschreibt das „Verhalten“ von Funktionen
- $\mathcal{O}(n)$ erfasst Wachstumsverhalten eines Algorithmus in Abhängigkeit der Eingabegröße n
- formale Definition:

$$f \in \mathcal{O}(g) \Leftrightarrow \limsup_{x \to n} \left| \frac{f(x)}{g(x)} \right| < \infty$$

- Beispiele:
  - $n \in \mathcal{O}(n)$
  - $2n \in \mathcal{O}(n)$
  - $n^4 \in \mathcal{O}(n^4)$
  - $18 * \log_2(n) \in \mathcal{O}(\log_2(n))$
  - $n^3 + n^6 + 9 \in \mathcal{O}(n^6)$
  - $2^{n+3} = 2^3 * 2^n \in \mathcal{O}(2^n)$

Sortieralgorithmen

Sortieralgorithmen - SelectionSort (Sortieren durch Einfügen)

- teile Liste auf in sortierten und unsortierten Teil
- sortierter Teil initial leer, unsortierter Teil initial komplette Liste
- solange unsortierter Teil nicht leer ist: wähle kleinstes Element von unsortierter Liste, vertausche dieses Element mit dem ersten Element der unsortierten Liste und vergrößere sortierten Teil um 1 nach rechts

Sortieralgorithmen - SelectionSort (Sortieren durch Einfügen)

- Beispiel:

Index	**1**	**2**	**3**	**4**	**5**	**6**
0. Durchlauf	4	7	3	16	6	1

Sortieralgorithmen - SelectionSort (Sortieren durch Einfügen)

- Beispiel:

Index	**1**	**2**	**3**	**4**	**5**	**6**
0. Durchlauf	4	7	3	16	6	1
1. Durchlauf	1	7	3	16	6	4

Sortieralgorithmen - SelectionSort (Sortieren durch Einfügen)

- Beispiel:

Index	**1**	**2**	**3**	**4**	**5**	**6**
0. Durchlauf	4	7	3	16	6	1
1. Durchlauf	1	7	3	16	6	4
2. Durchlauf	1	3	7	16	6	4

Sortieralgorithmen - SelectionSort (Sortieren durch Einfügen)

- Beispiel:

Index	**1**	**2**	**3**	**4**	**5**	**6**
0. Durchlauf	4	7	3	16	6	1
1. Durchlauf	1	7	3	16	6	4
2. Durchlauf	1	3	7	16	6	4
3. Durchlauf	1	3	4	16	6	7

Sortieralgorithmen - SelectionSort (Sortieren durch Einfügen)

- Beispiel:

Index	**1**	**2**	**3**	**4**	**5**	**6**
0. Durchlauf	4	7	3	16	6	1
1. Durchlauf	1	7	3	16	6	4
2. Durchlauf	1	3	7	16	6	4
3. Durchlauf	1	3	4	16	6	7
4. Durchlauf	1	3	4	6	16	7

Sortieralgorithmen - SelectionSort (Sortieren durch Einfügen)

- Beispiel:

Index	1	2	3	4	5	6
0. Durchlauf	4	7	3	16	6	1
1. Durchlauf	1	7	3	16	6	4
2. Durchlauf	1	3	7	16	6	4
3. Durchlauf	1	3	4	16	6	7
4. Durchlauf	1	3	4	6	16	7
5. Durchlauf	1	3	4	6	7	16

Sortieralgorithmen - SelectionSort (Sortieren durch Einfügen)

- Beispiel:

Index	1	2	3	4	5	6
0. Durchlauf	4	7	3	16	6	1
1. Durchlauf	1	7	3	16	6	4
2. Durchlauf	1	3	7	16	6	4
3. Durchlauf	1	3	4	16	6	7
4. Durchlauf	1	3	4	6	16	7
5. Durchlauf	1	3	4	6	7	16
6. Durchlauf	1	3	4	6	7	16

Sortieralgorithmen - SelectionSort (Sortieren durch Einfügen)

- Beispiel:

Index	1	2	3	4	5	6
0. Durchlauf	4	7	3	16	6	1
1. Durchlauf	1	7	3	16	6	4
2. Durchlauf	1	3	7	16	6	4
3. Durchlauf	1	3	4	16	6	7
4. Durchlauf	1	3	4	6	16	7
5. Durchlauf	1	3	4	6	7	16
6. Durchlauf	1	3	4	6	7	16

- Worst-Case Laufzeit: $n + (n-1) + \cdots + 1 = \frac{n*(n-1)}{2}$

Sortieralgorithmen - SelectionSort (Sortieren durch Einfügen)

- Beispiel:

Index	1	2	3	4	5	6
0. Durchlauf	4	7	3	16	6	1
1. Durchlauf	1	7	3	16	6	4
2. Durchlauf	1	3	7	16	6	4
3. Durchlauf	1	3	4	16	6	7
4. Durchlauf	1	3	4	6	16	7
5. Durchlauf	1	3	4	6	7	16
6. Durchlauf	1	3	4	6	7	16

- Worst-Case Laufzeit: $n + (n-1) + \cdots + 1 = \frac{n*(n-1)}{2} \in \mathcal{O}(n^2)$

Sortieralgorithmen - SelectionSort (Sortieren durch Einfügen)

- Beispiel:

Index	1	2	3	4	5	6
0. Durchlauf	4	7	3	16	6	1
1. Durchlauf	1	7	3	16	6	4
2. Durchlauf	1	3	7	16	6	4
3. Durchlauf	1	3	4	16	6	7
4. Durchlauf	1	3	4	6	16	7
5. Durchlauf	1	3	4	6	7	16
6. Durchlauf	1	3	4	6	7	16

- Worst-Case Laufzeit: $n + (n-1) + \cdots + 1 = \frac{n*(n-1)}{2} \in \mathcal{O}(n^2)$
- Aufgabe: Implementiere den Selection Sort Algorithmus in Maple und sortiere anschließend die Liste $[4, 7, 1, 16, 32, 5, 2]$! Hinweis: Es kann davon ausgegangen werden, dass alle Elemente der erhaltenen Liste unterschiedlich sind.

Sortieralgorithmen - MergeSort

- Divide & Conquer
- wenn Liste nur ein Element enthält $\rightarrow$ fertig
- teile zu sortierende Liste gleichmäßig in 2 Teillisten auf
- rufe rekursiv MergeSort auf diesen beiden Teillisten auf und verbinde das Ergebnis wieder zu einer gemeinsamen Liste (Rückgabe)

Sortieralgorithmen - MergeSort

- Beispiel:

Index	**1**	**2**	**3**	**4**	**5**	**6**
0. Durchlauf	4	7	3	16	6	1

Sortieralgorithmen - MergeSort

- Beispiel:

Index	**1**	**2**	**3**	**4**	**5**	**6**
0. Durchlauf	4	7	3	16	6	1
1. Durchlauf	4	7	3	16	6	1

Sortieralgorithmen - MergeSort

- Beispiel:

Index	**1**	**2**	**3**	**4**	**5**	**6**
0. Durchlauf	4	7	3	16	6	1
1. Durchlauf	4	7	3	16	6	1
2. Durchlauf	4	7	3	16	6	1

Sortieralgorithmen - MergeSort

- Beispiel:

Index	**1**	**2**	**3**	**4**	**5**	**6**
0. Durchlauf	4	7	3	16	6	1
1. Durchlauf	4	7	3	16	6	1
2. Durchlauf	4	7	3	16	6	1
3. Durchlauf	4	7	3	16	6	1

Sortieralgorithmen - MergeSort

- Beispiel:

Index	**1**	**2**	**3**	**4**	**5**	**6**
0. Durchlauf	4	7	3	16	6	1
1. Durchlauf	4	7	3	16	6	1
2. Durchlauf	4	7	3	16	6	1
3. Durchlauf	4	7	3	16	6	1
	4	7	3	6	16	1

Sortieralgorithmen - MergeSort

- Beispiel:

Index	**1**	**2**	**3**	**4**	**5**	**6**
0. Durchlauf	4	7	3	16	6	1
1. Durchlauf	4	7	3	16	6	1
2. Durchlauf	4	7	3	16	6	1
3. Durchlauf	4	7	3	16	6	1
	4	7	3	6	16	1
	3	4	7	1	6	16

Sortieralgorithmen - MergeSort

- Beispiel:

Index	**1**	**2**	**3**	**4**	**5**	**6**
0. Durchlauf	4	7	3	16	6	1
1. Durchlauf	4	7	3	16	6	1
2. Durchlauf	4	7	3	16	6	1
3. Durchlauf	4	7	3	16	6	1
	4	7	3	6	16	1
	3	4	7	1	6	16
	1	3	4	6	7	16

Sortieralgorithmen - MergeSort

- Beispiel:

Index	**1**	**2**	**3**	**4**	**5**	**6**
0. Durchlauf	4	7	3	16	6	1
1. Durchlauf	4	7	3	16	6	1
2. Durchlauf	4	7	3	16	6	1
3. Durchlauf	4	7	3	16	6	1
	4	7	3	6	16	1
	3	4	7	1	6	16
	1	3	4	6	7	16

- Worst-Case Laufzeit: $n * log_2(n) \in \mathcal{O}(n * log(n))$

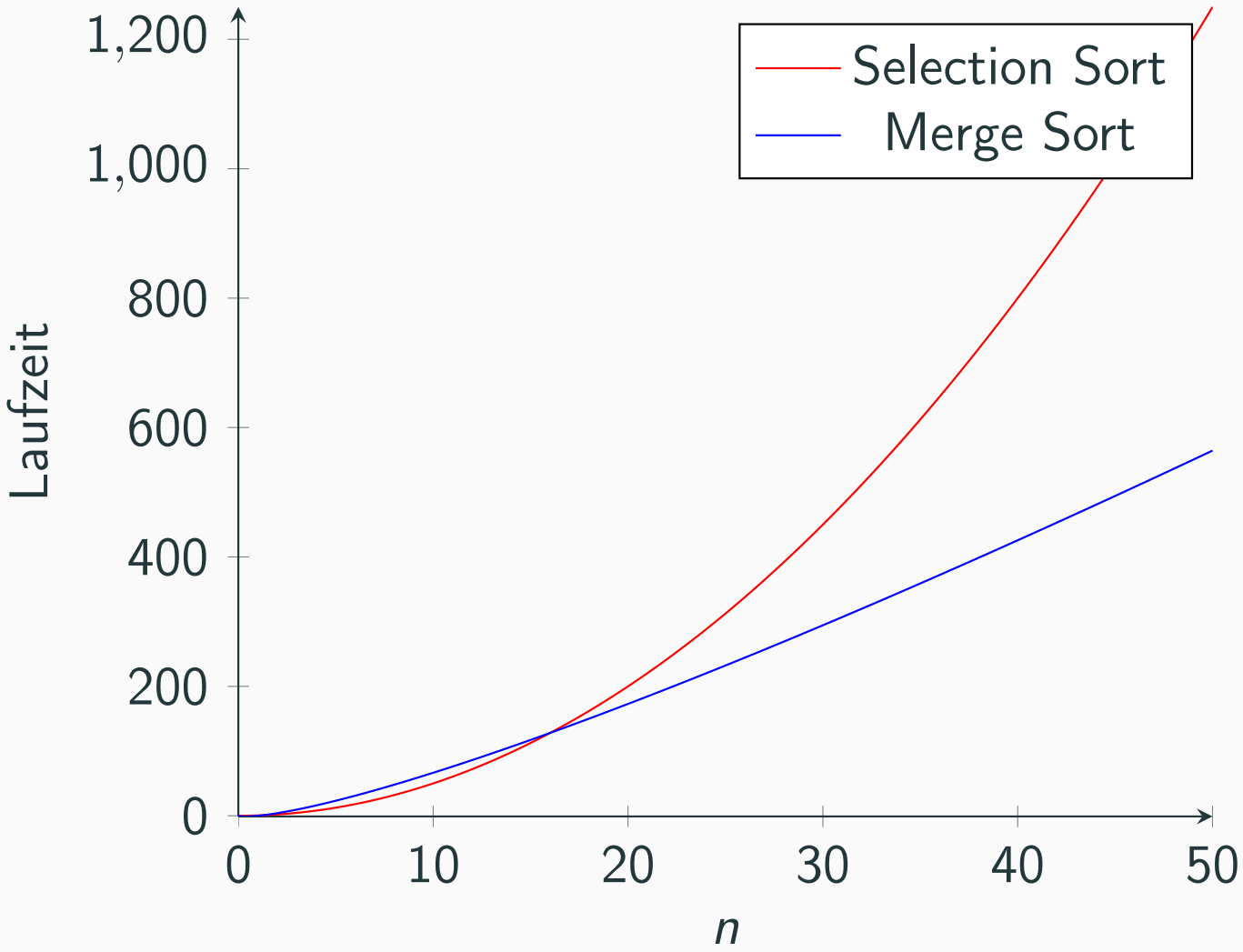
Sortieralgorithmen - MergeSort

- Beispiel:

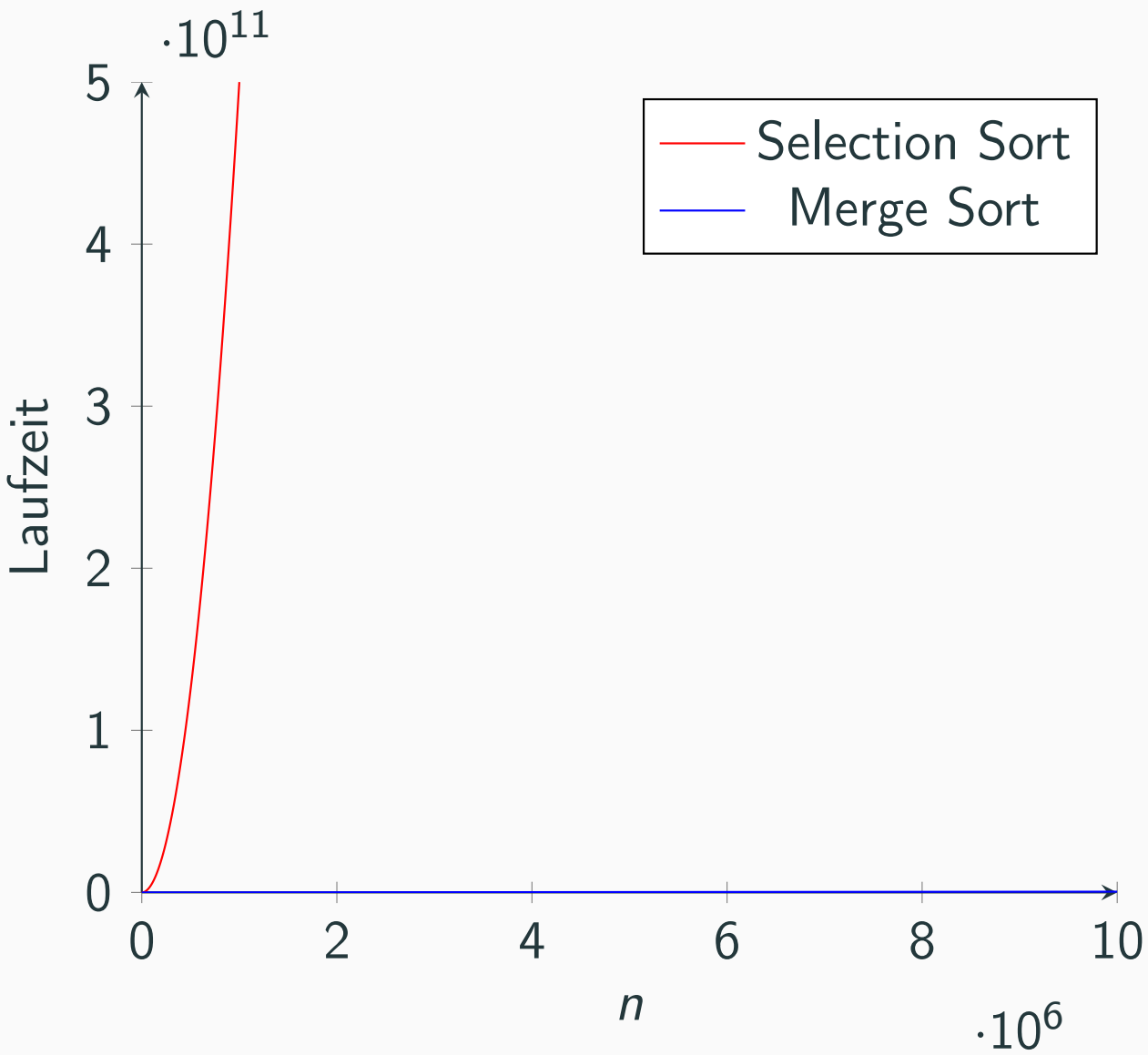
Index	**1**	**2**	**3**	**4**	**5**	**6**
0. Durchlauf	4	7	3	16	6	1
1. Durchlauf	4	7	3	16	6	1
2. Durchlauf	4	7	3	16	6	1
3. Durchlauf	4	7	3	16	6	1
	4	7	3	6	16	1
	3	4	7	1	6	16
	1	3	4	6	7	16

- Worst-Case Laufzeit: $n * log_2(n) \in \mathcal{O}(n * log(n))$

Laufzeitvergleich



Laufzeitvergleich



Vielen Dank für die Aufmerksamkeit!

Fragen?