Vorlesung 1: Algorithmenbegriff und Rechnen mit ganzen Zahlen

1 Was ist ein Algorithmus?

Einfache Beispiele aus dem Alltag:

- Kochrezepte
- LEGO / IKEA Bauanleitungen

Mögliche Definition:

Vollständige **Beschreibung** einer **Abfolge** von elementaren **Schritten** (Operationen), üblicherweise zur Lösung eines Problems

> *Was sind die zu lösenden Probleme bei diesen Beispielen?*
>
> - *Zutaten werden zu Speisen*
> - *Bausteine werden zu einem Piratenschiff*
>
> *Was sind elementare Schritte bei diesen Beispielen?*
>
> - *schneiden, vermischen, anbraten, …*
> - *Baustein platzieren*

2 Algorithmen in der Informatik

Computer können Algorithmen auf **Daten** ausführen mit sehr hoher Geschwindigkeit (viele Millionen Operationen pro Sekunde)

Computeralgorithmen sind zentral für bedeutsamen Fortschritte der Menschheit, zum Beispiel:

- Internet (Suchmaschinen, …)
- moderne Medizin (bildgebende Verfahren, …)
- Naturwissenschaften

  > *Teilchenbeschleuniger produzieren unglaubliche Datenmengen, die nur mit Computeralgorithmen analysiert werden können*

- Luft- und Raumfahrt (Autopilot)

Algorithmen sind das zentrale Thema der Informatik

> *Informatik wird oft als Wissenschaft der Algorithmen definiert*

3 Diese Vorlesung

Einführung in Entwurf und mathematische Analyse von Algorithmen

4 Rechnen mit ganzen Zahlen

> *Wichtige Beispiele von Algorithmen, die auf Daten ausgeführt werden sowohl von Menschen als auch von Computern (hier stellen die Daten Zahlen dar)*

5 "Schriftliche Multiplikation"

bekannt aus Primarschule (auch Schulmethode genannt)

```
    8765 × 4321
---------------
       35060
        26295
         17530
+          8765
---------------
       37873565
```

- berechne Teilprodukte
- addiere alle Teilprodukte

Geht es besser?

Was heisst besser?

Was sind geeignete elementare Operationen?

6 Addition

```
  175300
+   8765
   1100
--------
  184065
```

Allgemein: Addition zwei n-stellige Zahlen a und b

```
     a_{n-1} … a₁ a₀ (Summand a)
     b_{n-1} … b₁ b₀ (Summand b)
+ cₙ c_{n-1} … c₁ c₀ (Übertrag / carry c)
```

```
------------------
  sₙ s_{n-1} … s₁ s₀ (Summe a+b)
```

Hier gilt: c_0, $s_n = c_n$ und $10 \cdot c_{i+1} + s_i = a_i + b_i + c_i$ für alle $i \in \{0,\ldots,n-1\}$.

> *Als elementare Operation eignet sich hier die Addition von drei einstelligen Zahlen mit zweistelligem Ergebnis*

Elementare Operation ADD:

> *(v,w) ← ADD(x,y,z) // 10 v + w = x + y + z*

Eingabe: drei Ziffern x,y,z Ausgabe: zwei Ziffern v und w, so dass $10 \cdot v + w = x+y+z$ gilt.

Algorithmus für Addition:

> $c_0 \leftarrow 0$
> $(c_1, s_0) \leftarrow ADD(a_0, b_0, c_0)$
> $(c_2, s_1) \leftarrow ADD(a_1, b_1, c_1)$
> …
> $(c_n, s_{n-1}) \leftarrow ADD(a_{n-1}, b_{n-1}, c_{n-1})$
> $s_n \leftarrow c_n$

Hier: $c_1, \ldots, c_{n-1}$ sind sogenannte *Hilfsvariablen* (weder Ausgabe noch Eingabe; nur zur Berechnung)

> Koennen wir den Algorithmus klarer / einfacher formulieren?

Klarere Formulierung mit **FOR**-Schleife:

> $c_0 = 0$
>
> ***FOR** i **FROM** 1 **TO** n-1 {*
> $(c_{i+1}, s_i) \leftarrow ADD(a_i, b_i, c_i)$
> *}*
>
> $s_n \leftarrow c_n$

> Wie viele ADD Operationen führt der Algorithmus aus, um die Summe zweier n-stelligen Zahlen zu berechnen?

Anzahl ADD Operationen: n

7 Teilprodukte

```
  8765 × 4
----------
 3222       (Übertrag von einstelligen Multiplikationen)
+ 2840
------
 35060
```

> *Um zwei n-stellige Zahlen $a=a_{n-1}\cdots a_0$ und $b=b_{n-1}\cdots b_0$ nach der Schulmethode zu multiplizieren müssen wir Teilprodukte berechnen*

Allgemein: Multiplikation zweier n-stelligen Zahlen $a=a_{n-1}\cdots a_0$ und $b=b_{n-1}\cdots b_0$

j-tes Teilprodukt: $p_j = a_{n-1}\cdots a_0 \cdot b_j$

		a_{n-1}	⋯	a_1	a_0	×	b_j
	c_n	c_{n-1}	⋯	c_1	0		
+	0	d_{n-1}	⋯	d_1	d_0		
	$p_{j,n}$	$p_{j,n-1}$	⋯	$p_{j,1}$	$p_{j,0}$		

Hier gilt: $10\cdot c_{i+1} + d_i = a_i \cdot b_i$ für alle $i\in\{0,\ldots,n-1\}$ und $p_j=p_{j,n}\cdots p_{j,0}$ ist Summe von $c_n\, c_{n-1}\cdots c_1\, 0$ und $0\, d_{n-1}\cdots d_1\, d_0$

Weitere elementare Operation MULT:

> *(v,w) = MULT(x,y) // 10 v + w = x · y*

Eingabe: zwei Ziffern x und y

Ausgabe: zwei Ziffern v und w mit 10 v + w = x · y

Algorithmus für Teilprodukt

> ***FOR** i **FROM** 0 **TO** n-1 {*
> *$(c_{i+1}, d_i) \leftarrow MULT(a_i, b_j)$*
> *}*
>
> *$p_j \leftarrow$ Summe von $c_n\, c_{n-1}\cdots c_1\, 0$ und $0\, d_{n-1}\cdots d_1\, d_0$ // Algorithmus für Addition*

> Wie viele MULT und ADD Operationen führt der Algorithmus aus, um ein Teilprodukt einer n-stelligen Zahl zu berechnen?

Anzahl MULT Operationen: n

Anzahl ADD Operationen: n+1

8 Allgemeine Multiplikation

```
    8765 × 4321
---------------
        35060      p₃
         26295     p₂
          17530    p₁
+           8765   p₀
---------------
        37873565
```

Algorithmus für Multiplikation:

> $p_0 \leftarrow$ *0. Teilprodukt* $a \cdot b_0$
> $p_1 \leftarrow$ *1. Teilprodukt* $a \cdot b_1$
> ...
> $p_{n-1} \leftarrow$ *(n-1)-tes Teilprodukt* $a \cdot b_{n-1}$
>
> $q_0 \leftarrow p_0$
> $q_1 \leftarrow q_0 + p_1\,0$ *// eine Null angehängt*
>
> $q_2 \leftarrow q_1 + p_2\,00$ *// zwei Nullen angehängt*
> ...
> $q_{n-1} \leftarrow q_{n-2} + p_{n-1}\,0 \cdots 0$ *// n-1 Nullen angehängt*
>
> ***RETURN*** q_{n-1}

Kompakter und klarer:

> $q_0 \leftarrow a \cdot b_0$ *// Algorithmus für Teilprodukt*
> ***FOR*** *j* ***FROM*** *1* ***TO*** *n-1 {*
> $q_j \leftarrow q_{j-1} + a \cdot b_j \cdot 10^j$ *// Algorithmen für Addition und Teilprodukt*
> *}*
> ***RETURN*** q_{n-1}

> Wie viele MULT Operationen macht der Algorithmus?

Anzahl MULT Operationen: $n \cdot n = n^2$

> *Um all Teilprodukte zu berechnen, führen wir nämlich MULT Operationen MULT(a_i, b_j) aus für alle* $i,j \in \{0,\ldots,n-1\}$
>
> *Es gibt n Möglichkeiten für i und n Möglichkeiten für j. Die Anzahl aller Kombinationen ist damit* $n \cdot n = n^2$

> Wie viele ADD Operationen macht der Algorithmus?

Anzahl ADD Operationen: $\leq \underbrace{n \cdot (n+1)}_{\text{Berechnung der Teilprodukte}} + \underbrace{(n-1) \cdot 2n}_{\text{Summierung der Teilprodukte}} \leq 3n^2$

> *Wir verwenden hier dass alle Zahlen q_j und $p_j \cdot 10^j$ höchstens 2n-stellig sind (da $q_j \leq a \cdot b$ und $p_j \cdot 10^j \leq a \cdot b$)*

Geht es besser?

Vermutung (Kolmogorov 1956): Kein Algorithmus kommt mit wesentlich weniger als n^2 elementaren Operationen aus

> *Diese Vermutung formalisiert und quantifiziert ein intuitives Empfinden, dass Multiplizieren viel schwieriger ist als Addieren*

Vermutung ist falsch: Karatsuba findet 1960 besseren Algorithmus

> *Tatsächlich zeigen spätere Algorithmen, dass Multiplizieren kaum schwieriger als Addieren ist — sie brauchen nur $C \cdot n \cdot \log(n)$ Operationen für eine Konstante $C \geq 1$*
>
> *Schnelle Algorithmen zur Multiplikation sind wichtig für Anwendungen in den Naturwissenschaften und der Kryptographie*
>
> *Der aktuell beste Algorithmus wurde erst dieses Jahr (2019) gefunden (von Harvey und v. d. Hoefen)*