Karatsuba Algorithmus

by Qianli Wang

16.06.2020

Gliederung

- Einführung
- Addition zweier Zahlen mit Länge n
- Multiplikation einer Zahl mit einer Ziffer
- Grundschulmethode zur Multiplikation
- Karatsuba Algorithmus
- Zusammenfassung
- Literatur

- In der Grundschule gelernt.

- In der Grundschule gelernt.
- **Anwendungen:** Verschlüssung von Nachrichten, Kryptographie, polynomielle Multiplikation oder Lösung von geometrischen Problemen usw.

- In der Grundschule gelernt.
- **Anwendungen:** Verschlüssung von Nachrichten, Kryptographie, polynomielle Multiplikation oder Lösung von geometrischen Problemen usw.
- Ineffizient bei der Multiplikation zweier großen Zahlen.

Vordefinition

Definition 1:

Eine **Grundoperation** ist eine Operation, die der Prozessor direkt unterstützt. z.B. Addition, Multiplikation

Vordefinition

Definition 1:

Eine **Grundoperation** ist eine Operation, die der Prozessor direkt unterstützt. z.B. Addition, Multiplikation

Beobachtung 2:

Die zifferweise Addition und Multiplikation kostet nur konstante Zeit, also $\mathcal{O}(1)$.

Vorbemerkung

Vorbemerkung:

Wir können eine Zahl als ein Array von Ziffern betrachten:
12345 $\rightarrow$ ['1', '2', '3', '4', '5']
Anzahl von Ziffern $\Leftrightarrow$ Länge vom Array

Addition zweier Zahlen mit Länge n

$$
\begin{array}{lr}
 & 6789 \\
+ & 5678 \\
 & {\scriptstyle 1\,1\,1\,1} \\
\hline
 & 12467
\end{array}
$$

Schritte:

1) Schreiben die beiden Zahlen untereinander

Addition zweier Zahlen mit Länge n

```
      6789
+     5678
     1111
----------------
     12467
```

Schritte:

1) Schreiben die beiden Zahlen untereinander
2) Gehen von rechts nach links alle Spalten durch und rechnen.

Addition zweier Zahlen mit Länge n

```
     6789
+    5678
    1111
---------
    12467
```

Schritte:

1) Schreiben die beiden Zahlen untereinander
2) Gehen von rechts nach links alle Spalten durch und rechnen.
3) Tragen Überträge eine Stelle davor

Addition zweier Zahlen mit Länge n

```
        6789
+       5678
       1111
------------
       12467
```

Schritte:

1) Schreiben die beiden Zahlen untereinander
2) Gehen von rechts nach links alle Spalten durch und rechnen.
3) Tragen Überträge eine Stelle davor
4) Schreiben den letzten Übertrag als linkeste Ziffer des Ergebnis hin.

Analyse von der Addition

- **Angenommen:** Wir haben zwei Zahlen mit Länge n.

Satz 3:

Die Addition zweier Zahlen benötigt n Grundoperationen. Die Laufzeit liegt in $\mathcal{O}(n)$

Analyse von der Addition

- **Angenommen:** Wir haben zwei Zahlen mit Länge n.

Satz 3:

Die Addition zweier Zahlen benötigt n Grundoperationen. Die Laufzeit liegt in $\mathcal{O}(n)$

Beweis:

Die Länge n entspricht dann n ziffernweise Additionen. Man bekommt die linkeste Ziffer ohne weitere Berechnung.
$\rightarrow$ Laufzeit: $\mathcal{O}(n)$.

Multiplikation einer Zahl mit einer Ziffer

```
        1 2 3 4
*             6
---------------------------
            2 4
          1 8
        1 2
+     6
---------------------------
        7 4 0 4
```

Schritte

1) Berechne alle Teilprodukte von zifferweise jeweiligen Multiplikationen.

Multiplikation einer Zahl mit einer Ziffer

```
      1 2 3 4
*           6
---------------------------
          2 4
        1 8
      1 2
+   6
---------------------------
    7 4 0 4
```

Schritte

1) Berechne alle Teilprodukte von zifferweise jeweiligen Multiplikationen.
2) Schreiben Teilprodukte von rechts nach links schräg auf.

Multiplikation einer Zahl mit einer Ziffer

```
        1 2 3 4
*             6
---------------------------
            2 4
          1 8
        1 2
+     6
---------------------------
      7 4 0 4
```

Schritte

1) Berechne alle Teilprodukte von zifferweise jeweiligen Multiplikationen.
2) Schreiben Teilprodukte von rechts nach links schräg auf.
3) Addieren alle Ziffer in der selben Spalte.

Analyse

- **Angenommen:** a ist eine Zahl mit Länge n, b ist eine Ziffer.

Satz 4

Laufzeit der Multiplikation einer Zahl mit einer Ziffer ist $\mathcal{O}(n)$. Die Multiplikation braucht 2n Grundoperationen.

Analyse

- **Angenommen:** a ist eine Zahl mit Länge n, b ist eine Ziffer.

Satz 4

Laufzeit der Multiplikation einer Zahl mit einer Ziffer ist $\mathcal{O}(n)$. Die Multiplikation braucht 2n Grundoperationen.

Beweis:

Jeder Ziffer von a soll jeweils einmal mit b multiplizieren
$\rightarrow$ n Multiplikationen.
Anschließend sollen alle Teilprodukte addiert werden $\rightarrow$ n Additionen.
$\rightarrow$ Insgesamt 2n Grundoperationen.
$\rightarrow$ $\mathit{Laufzeit} : \mathcal{O}(n)$

Grundschulmethode zur Multiplikation

- **Als Beispiel:** Wir haben zwei Zahlen 1234 und 5678, die beiden aus 4 Ziffern bestehen.

```
          1 2 3 4
*         5 6 7 8
---------------------------
          9 8 7 2
        8 6 3 8
      7 4 0 4
+   6 1 7 0
---------------------------
    7 0 0 6 6 5 2
```

Schritte

1) Es werden die Teilprodukte nebeneinander geschrieben.

Grundschulmethode zur Multiplikation

- **Als Beispiel:** Wir haben zwei Zahlen 1234 und 5678, die beiden aus 4 Ziffern bestehen.

```
            1 2 3 4
*           5 6 7 8
---------------------------
            9 8 7 2
          8 6 3 8
        7 4 0 4
+     6 1 7 0
---------------------------
      7 0 0 6 6 5 2
```

Schritte

1) Es werden die Teilprodukte nebeneinander geschrieben.
2) Beginnt mit der niedrigsten Stelle und man kann es danach leicht addieren.

Analyse von Grundoperationen

```
        1 2 3 4
*       5 6 7 8
---------------------------
        9 8 7 2
      8 6 3 8
    7 4 0 4
+ 6 1 7 0
---------------------------
  7 0 0 6 6 5 2
```

Analyse von der Multiplikation:

1) "Zahl mal Ziffer" Multiplikation
2) n solche Multiplikationen notwendig

$\rightarrow$ Anzahl der Grundoperationen $= 2n \cdot n = 2n^2$

Analyse von Grundoperationen

```
              1 2 3 4
    *         5 6 7 8
----------------------
        0 0 0 9 8 7 2
        0 0 8 6 3 8 0
        0 7 4 0 4 0 0
+       6 1 7 0 0 0 0
----------------------
        7 0 0 6 6 5 2
```

Analyse von der Addition:

1) Teilprodukt kann max. $(n+1)$ Ziffern haben.
2) Höchstens $(n+1+n-1) = 2n$ spaltenweise Additionen

$\rightarrow$ Anzahl der Grundoperationen zur Addition $= 2n \cdot (n-1) = 2n^2 - 2n$

$\Rightarrow$ Sämtliche Anzahl der Grundoperationen $= 2n^2 - 2n + 2n^2 = 4n^2 - 2n$

$\Rightarrow$ *Laufzeit* : $\mathcal{O}(n^2)$

Teile und Herrsche

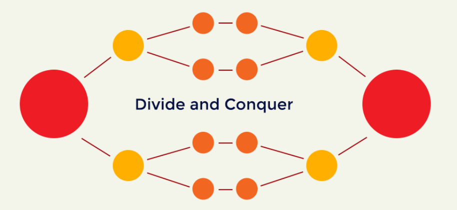


Abbildung: Quelle: https://medium.com/@gaurav_52429/divide-and-conquer-paradigm-in-algorithms-ef43fb2222f5

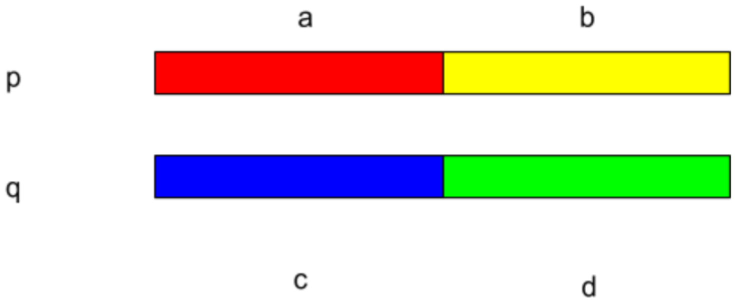
Definition:

Teile und Herrsche zerlegt ein Problem rekursiv in zwei oder mehrere Unterprobleme desselber oder verwandter Art, bis diese einfach genug werden, um dirkt gelöst werden zu können.

Splitmultiply

- **Angenommen,** dass wir zwei Zahlen p und q haben mit Länge n. Wir definieren $m = \lfloor \frac{n}{2} \rfloor$. Dann können wir p und q so darstellen:

$$p = 10^m \cdot a + b$$
$$q = 10^m \cdot c + d$$
$$(wobei\ a,\ b,\ c,\ d \in \mathbb{Z})$$



Darstellung von p und q

- **Distributivitätsgesetz** verwenden:

$$
\begin{aligned}
p \cdot q &= (10^m \cdot a + b) \cdot (10^m \cdot c + d) && (1)\\
&= a \cdot c \cdot 10^{2m} + (a \cdot d + b \cdot c) \cdot 10^m + b \cdot d && (2)\\
&= a \cdot c \cdot 10^n + (a \cdot d + b \cdot c) \cdot 10^{\frac{n}{2}} + b \cdot d && (3)
\end{aligned}
$$

Pseudocode

1 SPLITMULTIPLY(x, y, n):

Require: $n \geq 1$
Ensure: *Get the correct product of x and y*

```
if n = 1 then
    return  x · y
else
    m ← ⌊n/2⌋
    a ← ⌊x/10^m⌋;   b ← x mod 10^m;   c ← ⌊y/10^m⌋;   d ← y mod 10^m
    e ← SPLITMULTIPLY(a, c, m)
    f ← SPLITMULTIPLY(b, d, m)
    g ← SPLITMULTIPLY(b, c, m)
    h ← SPLITMULTIPLY(a, d, m)
    return  10^(2m) · e + 10^m (g + h) + f
end if
```

Analyse

- $p \cdot q = a \cdot c \cdot 10^n + (a \cdot d + b \cdot c) \cdot 10^{\frac{n}{2}} + b \cdot d \quad (3)$

Beobachtung 5:

4 unterschiedlichen Multiplikationen mit Länge $\lfloor \frac{n}{2} \rfloor$ erforderlich.
Darüber hinaus sind 3 Additionen mit Länge $\lfloor \frac{n}{2} \rfloor$ benötigt.

Analyse

- $p \cdot q = a \cdot c \cdot 10^n + (a \cdot d + b \cdot c) \cdot 10^{\frac{n}{2}} + b \cdot d \quad (3)$

Beobachtung 5:

4 unterschiedlichen Multiplikationen mit Länge $\lfloor \frac{n}{2} \rfloor$ erforderlich.
Darüber hinaus sind 3 Additionen mit Länge $\lfloor \frac{n}{2} \rfloor$ benötigt.

Satz 6:

Laufzeit des einfachen divide and conquer Ansatzes ist $\mathcal{O}(n^2)$.

Analyse

Beweis vom Satz 6:

$\rightarrow$ 4 Multiplikationen braucht. Dann können wir die Rekursionsformel so formulieren:

$$T(n) = 4 \cdot T(\lfloor \tfrac{n}{2} \rfloor) + \mathcal{O}(n)$$
$$T(\tfrac{n}{2}) = 4 \cdot T(\lfloor \tfrac{n}{4} \rfloor) + \mathcal{O}(\tfrac{n}{2})$$
...
$$T(2) = 4 \cdot T(1) + \mathcal{O}(1) \quad \text{(wobei } T(1) = \mathcal{O}(1)\text{)}$$

Beweis der Rekursionsformel

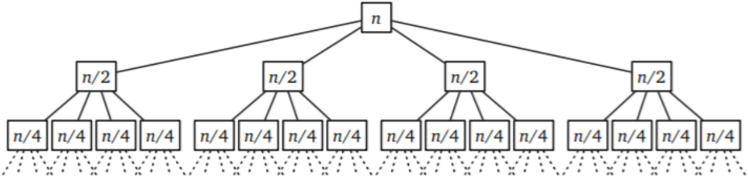


Abbildung: Quelle:
http://jeffe.cs.illinois.edu/teaching/algorithms/book/Algorithms-JeffE.pdf

Beweis:

Tiefe: $log_2 n$. Auf der **ersten** Ebene: 4^1 Rekursionen. Auf der **zweiten** Ebene: 4^2 Rekursionen, usw. Das heißt, auf der **letzten** Ebene gibt es dann $4^{\log_2 n}$ Rekursionen. Und die Laufzeit jeder Rekursion auf der letzten Ebene ist $T(1)$ *bzw.* $\mathcal{O}(1)$.

$\rightarrow$ Laufzeit: $4^{\log_2 n} \cdot \mathcal{O}(1) = \mathcal{O}(4^{\log_2 n}) = \mathcal{O}(n^{\log_2 4}) = \mathcal{O}(n^2)$

Einführung vom Karatusba Algorithmus

Abbildung: Quelle: https://en.wikipedia.org/wiki/Anatoly$_K$*aratsuba*

- Von Anatoli Alexejewitsch Karazuba in 1962 veröffentlicht

Einführung vom Karatusba Algorithmus

Abbildung: Quelle: https://en.wikipedia.org/wiki/Anatoly$_K$*aratsuba*

- Von Anatoli Alexejewitsch Karazuba in 1962 veröffentlicht
- Schneller als die quadratische Grundschulmethode

Einführung vom Karatusba Algorithmus

Abbildung: Quelle: https://en.wikipedia.org/wiki/Anatoly$_K$*aratsuba*

- Von Anatoli Alexejewitsch Karazuba in 1962 veröffentlicht
- Schneller als die quadratische Grundschulmethode
- Laufzeit: $\mathcal{O}(n^{\log_2 3}) \approx \mathcal{O}(n^{1.585})$

Einführung vom Karatusba Algorithmus

Abbildung: Quelle: https://en.wikipedia.org/wiki/Anatoly$_K$*aratsuba*

- Von Anatoli Alexejewitsch Karazuba in 1962 veröffentlicht
- Schneller als die quadratische Grundschulmethode
- Laufzeit: $\mathcal{O}(n^{\log_2 3}) \approx \mathcal{O}(n^{1.585})$
- Eine Multiplikation eingespart

Vorbereitung

Kernidee:

4 Unterprobleme auf 3 Unterproblemen reduzieren.

Vorbereitung

Kernidee:

4 Unterprobleme auf 3 Unterproblemen reduzieren.

Fallunterscheidung von den Eingaben:

Fall 1: Multiplikation zweier einstelligen Zahlen ($n = 1$).

Fall 2: Multiplikation zweier Zahlen mit Länge n, wobei n größer 1 ist.

Vorbereitung

Kernidee:

4 Unterprobleme auf 3 Unterproblemen reduzieren.

Fallunterscheidung von den Eingaben:

Fall 1: Multiplikation zweier einstelligen Zahlen ($n = 1$).

Fall 2: Multiplikation zweier Zahlen mit Länge n, wobei n größer 1 ist.

Beobachtung 7

Im ersten Fall kann man sofort Ergebnis bekommen, weil dafür genau nur eine Grundoperation benötigt ist, die vom Prozessor direkt unterstützt ist.

Karatsuba Algorithmus

Umformen:

Wir haben zwei Zahlen p und q mit Länge n und definieren $m = \lfloor \frac{n}{2} \rfloor$.

$$p = 10^m \cdot a + b$$
$$q = 10^m \cdot c + d$$
$$(wobei\ a,\ b,\ c,\ d \in \mathbb{Z})$$

Dann können wir $p \cdot q$ so formulieren:

Karatsuba Algorithmus

Umformen:

Wir haben zwei Zahlen p und q mit Länge n und definieren $m = \lfloor \frac{n}{2} \rfloor$.

$$p = 10^m \cdot a + b$$
$$q = 10^m \cdot c + d$$
$$(wobei\ a,\ b,\ c,\ d \in \mathbb{Z})$$

Dann können wir $p \cdot q$ so formulieren:

$$\begin{aligned} p \cdot q &= (10^m \cdot a + b) \cdot (10^m \cdot c + d) \\ &= a \cdot c \cdot 10^{2m} + (a \cdot d + b \cdot c) \cdot 10^m + b \cdot d \quad (Formel 3) \\ &= a \cdot c \cdot 10^{2m} + (a \cdot c + b \cdot d - (b - a) \cdot (d - c)) \cdot 10^m + b \cdot d \\ &= a \cdot c \cdot 10^n + (a \cdot c + b \cdot d - (b - a) \cdot (d - c)) \cdot 10^{\frac{n}{2}} + b \cdot d \end{aligned}$$

Beweis der Korrektheit vom Umformen:

$$
\begin{aligned}
&(a \cdot c + b \cdot d - (b - a) \cdot (d - c)) \\
&= a \cdot c + b \cdot d - (b \cdot d - b \cdot c - a \cdot d + a \cdot c) \\
&= a \cdot c + b \cdot d - b \cdot d + b \cdot c + a \cdot d - a \cdot c \\
&= b \cdot c + a \cdot d
\end{aligned}
$$

Genauere Beschreibung:

Seiene:

$$
\begin{aligned}
u &= a \cdot c \\
v &= (b - a) \cdot (d - c) \\
w &= b \cdot d
\end{aligned}
$$

Also wir können $p \cdot q$ als folgendes umschreiben:

$$p \cdot q = u \cdot 10^n + (u + w - v) \cdot 10^{\frac{n}{2}} + w$$

$\rightarrow$ Eine Multiplikation eingespart.

Karatsuba Algorithmus

Beispiel:

Wir wollen das Produkt von 12345 und 6789 berechnen. Seien a = 12, b = 345 und c = 6, d = 789, nämlich:

$$12345 = 12 \cdot 1000 + 345$$
$$6789 = 6 \cdot 100 + 789$$

Zwischenergebnisse

$$u = a \cdot c = 12 \cdot 6 = 72$$
$$v = (b - a) \cdot (d - c) = (345 - 12) \cdot (789 - 6) = 260739$$
$$w = b \cdot d = 345 \cdot 789 = 272205$$

$\Rightarrow 12345 \cdot 6789 = u \cdot 10^6 + (u + w - v) \cdot 10^3 + v = 83810205$

Pseudocode

2 karatsuba(num1, num2):

Require: *The length of each number is* $\geq$ 1.
Ensure: *Get the correct product of* $num1$ *and* $num2$

```
if (num1 < 10) or (num2 < 10) then
    return num1 · num2
else
    m ← max(size_base10(num1), size_base10(num2))
    m_2 ← floor(m/2)
    high1, low1 ← split_at(num1, m_2)
    high2, low2 ← split_at(num2, m_2)
    u ← karatsuba(high1, high2)
    v ← karatsuba((high1 − low1), (high2 − low2))
    w ← karatsuba(low1, low2)
    return (u · 10^(m_2·2)) + ((u + w − v) · 10^(m_2)) + w
end if
```

Laufzeitsanalyse

Beobachtung 9

Außer den Multiplikationen werden noch $\mathcal{O}(n)$ Additionen benötigt.

Laufzeitsanalyse

Beobachtung 9

Außer den Multiplikationen werden noch $\mathcal{O}(n)$ Additionen benötigt.

Satz 10

Die Laufzeit von Karatsuba Algorithmus ist $\mathcal{O}(n^{\log_2 3})$.

Laufzeitsanalyse

Beobachtung 9

Außer den Multiplikationen werden noch $\mathcal{O}(n)$ Additionen benötigt.

Satz 10

Die Laufzeit von Karatsuba Algorithmus ist $\mathcal{O}(n^{\log_2 3})$.

Beweis

Initialisierung: $\mathrm{T}(1) = \mathcal{O}(1)$
Rekursionsformel: $T(n) = 3 \cdot T(\frac{n}{2}) + \mathcal{O}(n)$
Angenommen: $\mathrm{n} = 2^k \Leftrightarrow \log_2 n = k$
$\Rightarrow f(k) = \mathrm{T}(2^k)$

Laufzeitsanalyse

Beweis:

$$\Rightarrow \; f(k) = 3 \cdot \; f(k-1) + 2^k$$

$$\Leftrightarrow \; \frac{f(k)}{3^k} = \frac{f(k-1)}{3^{k-1}} + (\frac{2}{3})^k$$

$$\Rightarrow \; f(k) \leqslant \; 3^{k+1} \quad (*)$$

Dann können wir die in die Definition von f einsetzen:

$\mathrm{T(n)} = \mathrm{T}(2^k) = f(k) = f(\log_2 n) \leqslant \; 3^{\log 2n+1}$
$\Rightarrow \; T(n) = \mathcal{O}(3^{\log_2 n}) = \mathcal{O}(2^{\log_2 3 \cdot \log_2 n}) = \mathcal{O}(n^{\log_2 3})$

Beweis

(*)

$$\frac{f(k)}{3^k} = \frac{f(k-1)}{3^{k-1}} + (\frac{2}{3})^k$$

$$\frac{f(k-1)}{3^{k-1}} = \frac{f(k-2)}{3^{k-2}} + (\frac{2}{3})^{k-1}$$

$$\dots$$

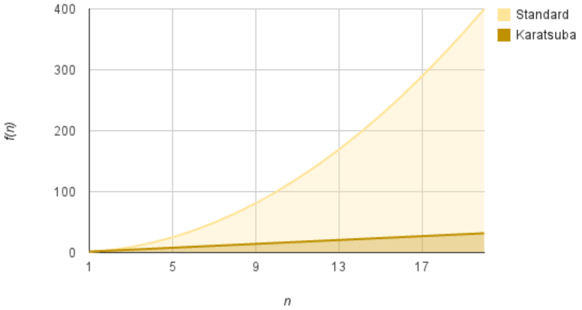
$$\frac{f(1)}{3^1} = \frac{f(0)}{3^0} + (\frac{2}{3})^1$$

$$\Rightarrow \frac{f(k)}{3^k} - \frac{f(0)}{3^0} = \sum_{i=1}^{k}(\frac{2}{3})^i = \frac{\frac{2}{3}\cdot(1-\frac{2}{3})^k}{1-\frac{2}{3}} = \frac{2}{3^k}$$

$$\Leftrightarrow \ f(k) \ = \ 3^k \cdot (1 + 2 \cdot \frac{1}{3^k}) = 3^k + 2$$

$$\Rightarrow \ f(k) \leqslant \ 3^{k+1}$$

Laufzeitsvergleich



Länge	Karatsuba	Grundschulmethode
1,024=2^{10}	59,049	1,048,675
1,048,576 $= 2^{20}$	3,486,784,401	1,099,511,627,776
$n = 2^k$	3^k	4^k

Schlussfolgerung

- Wenn die Zahlen ziemlich lang sind, ist Karatsuba Algorithmus viel besser.
- Ab welcher Länge?

Andere Methoden

1) **Schönhage-Strassen-Algorithmus** in $\mathcal{O}(n \cdot \log n \cdot \log \log n)$ (FTT)
($Quelle : https : //de.wikipedia.org/wiki/Sch\%C3\%B6nhage-Strassen-Algorithmus$)
2) **Toom-Cook-Algorithmus** in $\mathcal{O}(n \cdot \log n \cdot 2^{\sqrt{log(n)}})$
(Quelle: $https : //de.wikipedia.org/wiki/Toom - Cook - Algorithmus$)

Zusammenfassung

- Multiplikation auf kleinere Unterprobleme zurückführen
  $\rightarrow$ Divide and Conquer
- 3 anstatt 4 Multiplikationen
- Laufzeit: $\mathcal{O}(n^{\log_2 3})$

Literatur

- https://en.wikipedia.org/wiki/Karatsuba_algorithm
- https://iq.opengenus.org/content/images/2018/05/Karatsuba-Complexity.png
- https://de.wikipedia.org/wiki/Sch%C3%B6nhage-Strassen-Algorithmus
- https://de.wikipedia.org/wiki/Toom-Cook-Algorithmus

Vielen Dank für Ihre Aufmerksamkeit!