LEHRSTUHL FÜR RECHNERARCHITEKTUR UND PARALLELE SYSTEME

# Grundlagenpraktikum: Rechnerarchitektur

Gruppe 213 – Abgabe zu Aufgabe A326
Sommersemester 2023

Noah Schlenker Leon Baptist Kniffki Christian Krinitsin

## 1 Einleitung

Verwendung von Wurzel 2:

- Das verhältnis der beiden seitenlängen eines blattes im din-a-format beträgt 1 / sqrt(2) mit rundung auf ganze millimeter. Dadurch ist sichergestellt, dass bei halbierung des blattes entlang der längeren seite wieder ein blatt im din-a-format (mit um eins erhöhter nummerierung) entsteht.
- Die wurzel aus 2 ist das frequenzverhältnis zweier töne in der musik bei gleichschwebender stimmung, die einen tritonus, also eine halbe oktave bilden.
- In der elektrotechnik enthält die beziehung zwischen scheitelwert und effektivwert von sinusförmiger wechselspannung ebenfalls die konstante sqrt(2).

Geschichte über Näherung der Wurzel:

- Die alten Inder schätzen $\sqrt{2} \approx \frac{577}{408} = 1,414215686\ldots$ . Stimmt auf 5 Nachkommastellen. Abweichung beträgt nur +0,0001502 Prozent.
- Babylonier aus 1800 v. Chr.: $\frac{30547}{21600} = 1,414212962\ldots$ . Abweichung von -0,0000424 Prozent.

Möglichkeit, $\sqrt{2}$ mit einer unendlichen Präzision zu berechnen. Vorwegnahme: Bignums, Newton-Raphson-Division, Karazuba, Matrixexponentiation.
(0,75 Seiten)

## 2 Lösungsansatz

### 2.1 Big-Num

Diskussion über die Notwendigkeit von Bignums, Erklärung der Implementierung (Little Endian, Arithmetische Operationen, nicht Division). Laufzeiten thematisieren?
(1 Seite)

## 2.2 Karazuba-Multiplikation

Um zwei Zahlen miteinander zu multiplizieren läuft man bei der russischen Bauernmultiplikation des Multiplikators einmal den Multiplikanten ab, wenn man diesen auf das Zwischenergebnis addiert. Für zwei Zahlen der Längen $n, m \in \mathbb{N}$ hat die Bauernmultiplikation also Laufzeit von $\mathcal{O}(n * m)$ und im häufigen Fall von $n = m$ $\mathcal{O}(n^2)$. Die Multiplikation spielt auf der untersten Ebene für die Matrixmultiplikation und somit auch für die Berechnung von $\sqrt{2}$ eine wichtige Rolle.
Dank des Karazuba-Algorithmus kann die Laufzeit einer Multiplikation auf $\mathcal{O}(n^{1.59})$ verringert werden. Dafür werden die zu multiplizierenden Zahlen in die Form $a_0 + a_1 * 2^m$ gebracht, wobei $a_0$ und $a_1$ maximal die größe $\lceil \frac{n}{2} \rceil$ haben. Durch folgende Umformung kann $a * b$ mit nur noch drei $\lceil \frac{n}{2} \rceil$ großen Multiplikationen und 6 zu vernachlässigenden Additionen/Subtraktionen und einigen shifts ermittelt werden:

## 2.3 Schnelle Exponentiation

Die schnelle Exponentiation nutzt Assozitivität und Potenzgesetze, um die Zahl der der Multiplikationen bei der Exponentiation von $\mathcal{O}(n)$ auf $\mathcal{O}(\log n)$ zu verringern. Naiv kann eine Potenz $a^n$ mit $n \in \mathbb{N}$ nach der Schulmethode mit $\prod_1^n a$ berechnet werden. Dafür benötigt man allerdings $n - 1$ Multiplikationen, was bei großen Werten für $n$ zu einer langen Berechnung ausartet.
Um dieses Problem effizienter zu lösen, werfen wir erst einmal einen Blick auf die Potenzgesetze für assoziative Operatoren. Denn sowohl eine Multiplikation, als auch eine Addition im Exponenten kann aufgeteilt werden:

$$a^{n+m} = \overbrace{a * \cdots * a}^{n+m} = \overbrace{(a * \cdots * a)}^{n} * \overbrace{(a * \cdots * a)}^{m} = a^n * a^m \tag{1}$$

$$a^{n*m} = \overbrace{a * \cdots * a}^{n*m} = \underbrace{\overbrace{(a * \cdots * a)}^{n} * \cdots * \overbrace{(a * \cdots * a)}^{n}}_{m} = \left(a^n\right)^m \tag{2}$$

Wenn man also $a^n$ und $a^m$ effizienter als mit $n + m$ Multiplikationen berechnen kann, kann man auch $a^{n+m}$ mit 1 effizient berechnen.
Bei der schnellen Exponentiation berechnet man durch wiederholtes Quadrieren alle $a^{(2^k)}$ mit $2^k \leq n$. Denn nach 2 gilt:

$$\left(a^{(2^k)}\right)^2 = a^{(2^k * 2)} = a^{(2^{k+1})}$$

Um $a^n$ mit $n = 2^k$ zu berechnen, sind damit nur noch $k = \log_2 n$ Multiplikationen notwendig.
Potenzen mit der Form $a^{(2^k)}$ können also effizient berechnet werden, um nun auch Potenzen mit $n \in \mathbb{N}$ berechnen zu können, nutzt man 1. Jede Zahl $n \in \mathbb{N}$ kann durch Addition von Zweierpotenzen dargestellt werden (Binärsystem).
Sei $n$ in Binärdarstellung $b_0 * 2^0 + b_1 * 2^1 + b_2 * 2^2 \ldots$, so erhält man $a^n$ mit:

$$a^n = a^{b_0*2^0 + b_1*2^1 + b_2*2^2 \ldots} = a^{b_0*2^0} * a^{b_1*2^1} * a^{b_2*2^2} \ldots$$

Da $b_i$ nur die Werte $0$ und $1$ annehmen kann, ist es am Ende eine boolsche Entscheidung, ob der aktuelle Wert von $a^{(2^k)}$ auf das Zwischenergebnis aufmultipliziert wird, oder nicht.
Außerdem gilt:

$$a^n * a^m = a^m * a^n \tag{3}$$

Auch wenn diese Gleichung auf den ersten Blick nach einem Kommutativitgesetz aussieht, gilt sie aufgrund der Assozitivität, da nur die Klammerung geändert wird:

$$\overbrace{(a * \cdots * a)}^{n} * \overbrace{(a * \cdots * a)}^{m} = \overbrace{(a * \cdots * a)}^{m} * \overbrace{(a * \cdots * a)}^{n}$$

Zum Beispiel:

$$7^3 * 7^4 = \overbrace{(7 * 7 * 7)}^{3} * \overbrace{(7 * 7 * 7 * 7)}^{4} = \overbrace{(7 * 7 * 7 * 7)}^{4} * \overbrace{(7 * 7 * 7)}^{3} = 7^4 * 7^3$$

Demnach macht es keinen Unterschied, ob zuerst $a^{(2^k)}$ mit dem kleinsten oder dem größten $k$ aufmultipliziert wird.
Da $(\mathbb{N}^{2\times2}, *)$ eine Gruppe und damit assoziativ ist, kann die schnelle Exponentiation auch für das Lösen von $a \in \mathbb{N}^{2\times2}$ genutzt werden.

### 2.4 Newton-Raphson-Division

Erklärung des Algorithmus und der Umsetzung im Code.
(1 Seite)

## 3 Korrektheit/Genauigkeit

Wahrscheinlich Genauigkeit, da es die Aufgabe ist, $\sqrt{2}$ beliebig genau darzustellen.

Umfangreiche Erklärung darüber, wie die Matrix Elemente an $\sqrt{2}$ konvergiert und Newton-Raphson and die Division. Erklärung, wie die Kombination aus Bignum und Fixkommazahlen unendliche Genauigkeit ermöglicht, auf Kosten von Laufzeit, die im nächsten Kapitel beleuchtet wird.
(1,5 - 2 Seiten?)

## 4 Performanzanalyse

Newton Raphson Laufzeit erklären, mit Graphiken demonstrieren, das selbe mit Exponentiation.
Vergleichsimplementierungen ansetzen (SIMD, nicht SIMD? / Karazuba, normale Multiplikation / Bitshifts?), Graphisch laufzeiten vergleichen, tatsächliche Performanz erklären und schlussfolgern.
(2 - 3 Seiten)

# 5 Zusammenfassung und Ausblick

Ziel erreicht, unendliche Präzision ist gegeben. Nutzer können, abhängig von ihren Anforderungen oder "Computerspezifikationen", die Wurzel von 2 mit diesem Programm berechnen.
Ausblick: SIMD in AVX, mithilfe von 256 Bit kann man Multiplikationen noch schneller machen. Division lässt sich wahrscheinlich nicht optimieren, da SIMD nicht verwendet werden kann, und immer eine gewisse Anzahl an Iterationen gebraucht wird.
(0,75 - 1 Seite)
(Insgesamt 6 - 7,75 Seiten, 2 - 4 Seiten fehlen!)

# Literatur