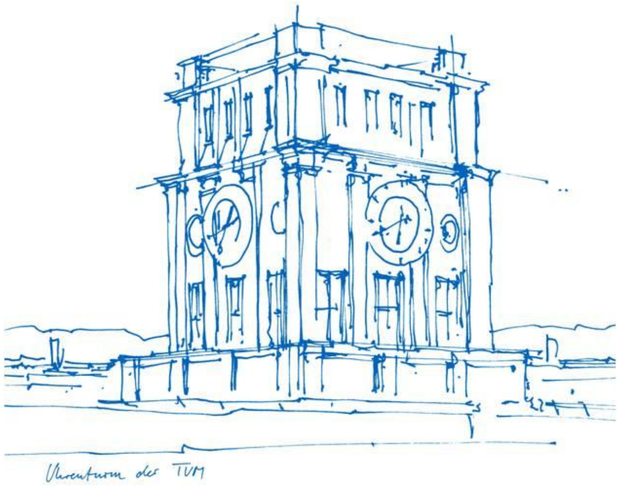
Praktikum Rechnerarchitektur

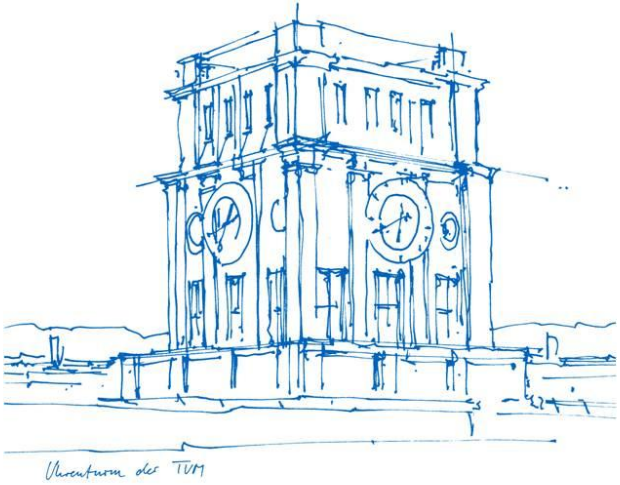
Geburtstagsparadox

Yesmine Maalej   Balkiss Nouri   Ismail Ben Ayed



Gliederung

1. ***Was ist Birthday Paradox?***
2. ***Taylor Reihe***
3. ***Look Up Tabelle***
4. ***Heron Verfahren***
5. ***Weitere Anwendungen***
6. ***K in Abhängigkeit von n***
7. ***Genauigkeit***
8. ***Performanzanalyse***
9. ***Genauigkeit vs Performanz***
10. ***Zusammenfassung***



1.Was ist Birthday Paradox?

Es geht um die Frage : wie viele Personen es geben muss, damit mit einer Wahrscheinlichkeit von 0,5 mindestens zwei Personen das gleiche Element aus einer Menge M von n Elementen teilen?

$$k \geq \frac{1 + \sqrt{8n \cdot \ln 2}}{2}$$

2.Reihendarstellung Methode:

- Formula :

$$f(x) = \sum_{n=0}^{\infty} \frac{f^{(n)}(a)(x-a)^n}{n!} \approx \sqrt{x}$$

- Berechnung von $f^n(x)$:
- $f^0(x) = x^{\frac{1}{2}}$
- $f^1(x) = \frac{1}{2} \cdot x^{\frac{-1}{2}}$
- $f^2(x) = \frac{-1}{2} \cdot \frac{1}{2} \cdot x^{\frac{-3}{2}}$
- $f^3(x) = \frac{-1}{2} \cdot \frac{1}{2} \cdot \frac{-3}{2} \cdot x^{\frac{-5}{2}}$
- Davon mittels induktions haben wir folgende Gleichung abgeleiten:
- $f^{n+1}(x) = \frac{(\frac{1}{2}-n) \cdot f^n(x)}{x}$

2.Reihendarstellung Methode:

- Berechnung von Koeffizienten für n>=1:

$$a_n = \frac{f^{(n)}(a)}{n!} \overset{\text{Per Induktion}}{=} \frac{(-1)^{n-1} \cdot \prod_{i=1}^{(n-1)} (2n-1)}{2^n \cdot n!}$$

- Aus dieser Formel haben wir die folgende Gleichung abgeleiten :

$$a_{n+1} = (-1)^n \cdot \frac{|a_n| \cdot (2(n-1)-1)}{2 \cdot (n+1)}$$

2.Reihendarstellung Methode:

Konvergenz der Reihe und wahl von x und a :

- die Formel $f(x) = \sum_{n=0}^{\infty} \frac{f^{(n)}(a)(x-a)^n}{n!}$ konvergiert gegen $\sqrt{x}$ falls:

  -1<x<1 und a=1

- wir möchten aber $\sqrt{1+8\cdot n\cdot ln(2)}$ berechnen mit 1+8.n ln(2)>1
- wir berechnen dann $\sqrt{(1+8\cdot n\cdot ln(2))\cdot 10^{-l}}$ mit l=Anzahl der ziffern von $(1+8\cdot n\cdot ln(2))$
- $\sqrt{1+8\cdot n\cdot ln(2)} = \sqrt{(1+8\cdot n\cdot ln(2))\cdot 10^{-l}}\cdot\sqrt{10^l}$ mit
- $\sqrt{10^l} = 10^{l/2}$ falls n gerade und $\sqrt{10^l} = 10^{(l-1)/2}\cdot\sqrt{10}$ falls n ungerade .

2.Reihendarstellung methode:

Performanz und genauigkeit von Taylorreihe:

1. Genauigkeit:

➢ Der Fehler ist maximal : $R_n(x) = \frac{f^{(n+1)}(z)}{(n+1)!}(x-a)^{n+1}.$

➢ n=50 durchläufe damit das Ergebnis für größere Zahler genauer ist

2. Performanz:

➢ Berechnung anzahl der Ziffern von x : $O(log(x))$

➢ Schleife für die Berechnung von Termen : im schlimmsten Fall 50 Durchläufe

➢ Schleife für die Berechnung von $\sqrt{10^l}$: $O(log(x))$

3. Look-Up-Methode

Die Wurzel von einer Fließkommazahl reduziert sich darauf, den Exponenten zu halbieren und die Wurzel von der Mantisse zu ziehen

$$\sqrt{2^e \cdot m} = 2^{\frac{e}{2}} \cdot \sqrt{m}$$

3. Look-Up-Methode

Wir speichern die Wurzeln der 15 höchstwertigen Bits der Mantisse im Bereich [1..4) in der Lookup tabelle.

Algorithm 1 build table()

---

for i : **integer** $0, 2^{16} - 1$

$f \leftarrow 1.i$

$table[i] \leftarrow mantissa(\sqrt{f});$

$f \leftarrow 2 \cdot 1.i$

$table[i + 2^{15}] \leftarrow mantissa(\sqrt{f});$

---

3. Look-Up-Methode

Algorithm 2 LUT sqrt

$e \leftarrow$ **exponent(V)**
$i \leftarrow$ **mantissa(V)**
if (e **bit-and** 1) **then** ▷ der Exponent ist ungerade
    das hohe Bit von i setzen
end if
$e \leftarrow \frac{e}{2}$ ▷ Dividiere e durch zwei (Bei dieser Division muss das Vorzeichen erhalten bleiben)
$j \leftarrow T[i]$
$U \leftarrow 2^e \cdot 1.j$

- Zeitkomplexität = O(1)

4. Heron Verfahren

- $a \leq \sqrt{n} \Rightarrow \frac{n}{a} \geq \sqrt{n}$
- $a \geq \sqrt{n} \Rightarrow \frac{n}{a} \leq \sqrt{n}$
- Iterationsvorschrift: $\boldsymbol{a_{i+1} = \frac{a_i + \frac{n}{a_i}}{2}}$
- je näher a_0 an $\sqrt{n}$ liegt, desto weniger Iterationen sind nötig, um das Ergebnis zu erreichen.
- $\sqrt{n}$ hat etwa halb so viele ganze Ziffern wie n selbst.
- $a_0 \in \{2, 7, 20, 70, 200, 700, \dots\}$

4. Heron Verfahren

- Ab einer bestimmten Iteration verdoppelt sich die Zahl der richtigen Ziffern in a_i.
- Beispiel für k=2:

x_i	**apoximation of** $\sqrt{0.25 + 2 * k * ln(2)}$
x_0	2.0000000000000000000
x_1	**1.7**556469440460205078**12**
x_2	**1.738**642334938049316406
x_3	**1.738559246063232421875**
x_4	**1.738559246063232421875**

Tabelle 1: die Entwicklung von $\sqrt{n}$ bei jeder Iteration

4.Heron Verfahren

- ➢ Maximal 7 Iterationen, um unsere Ergebnis zu erreichen.
- ➢ Zeitkomplexität zu Zählen von Anzahl der Ziffern = $O(log(n))$

$\longrightarrow$ Gute Performanz

- ➢ Abbruchbedingung der Schleife :Unterschied zwischen a_{i+1} und $a_i \geq e = 0.00000005$

$\longrightarrow$ Gute Genauigkeit unter Berücksichtigung der IEEE 745-Darstellung von Float.

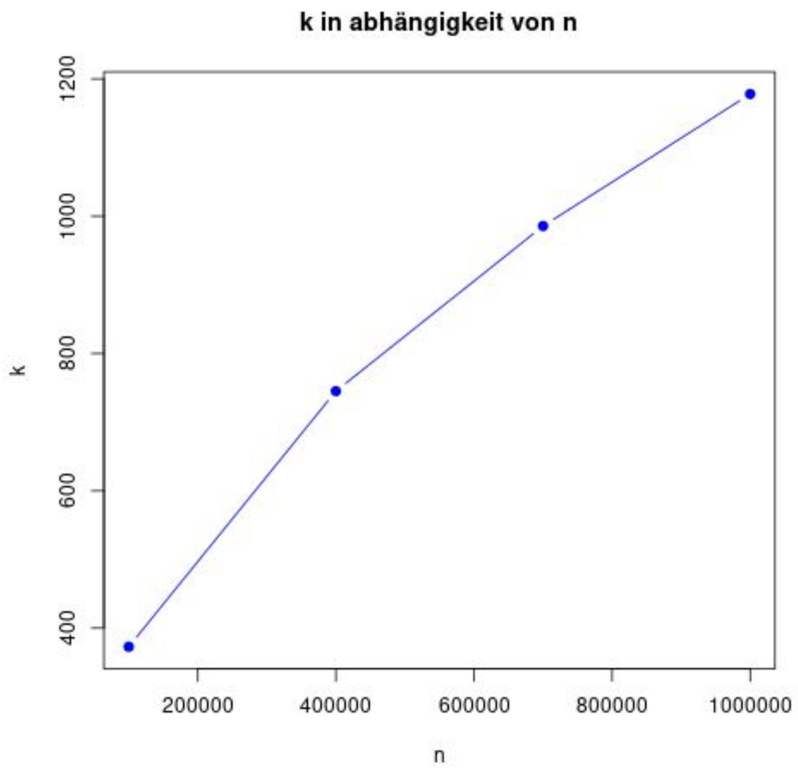
5.Weitere Anwendungen der birthday Formel

- die Berechnung der Anzahl k von Personen, so dass mindestens zwei Personen mit mindestens 0,5 Wahrscheinlichkeit das gleiche Element aus einer Menge M mit n Elementen haben.

Beispiele:

- Anzahl der Personen sodass mindestens zwei Leute mit einer Wahrscheinlichkeit von mindestens 0,5 die gleiche PIN für ihr Girokonto verwenden: Anzahl mögliche PIN : $n = 100000 \Rightarrow k = 119$
- Anzahl der MD5-Hashes um im Mittel mit mindestens fünfzig Prozent Wahrscheinlichkeit eine beliebige Kollision zu finden: $n = 2^{128} \Rightarrow k = 2.17 * 10^{19}$

6.K in Abhängigkeit von n



- ➢ Je größer die Grundmenge ist, desto mehr Elemente k brauchen wir, um eine Kollision mit 50 % Wahrscheinlichkeit zu haben.

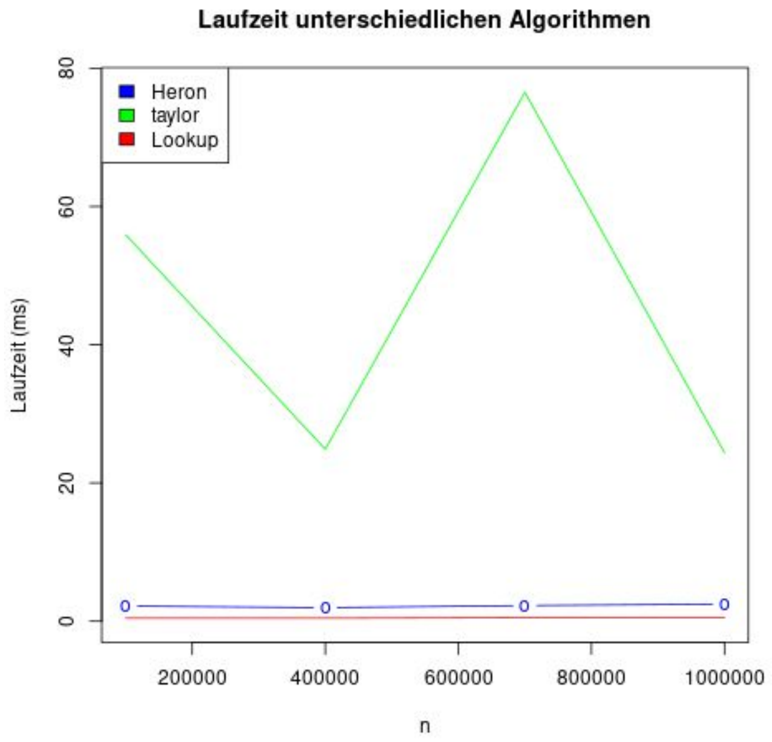
- ➢ Das bedeutet, dass man bei einer Hash-Funktion mit weniger Hash-Werten(n) eine 50%ige Chance hat, nach weniger Operationen(k) eine Kollision zu finden als bei einer Hash-Funktion mit mehr Hash-Werten(n) => weniger Hacking

- ➢ Das Birthday Paradox hat zu einer Verbesserung der Internetsicherheit geführt.

7.Genauigkeit

n	Heron	Taylor	Lookup
0	1.000000	**1.000000**	**1.000000**
1	1.779177	**1.779179**	**1.779175**
10^5	372.830017	**372.832520**	**372.820312**
10^{10}	117741.484375	**117742.265625**	**117738.500000**
10^{15}	37232968.000000	**37233216.000000**	**37232640.000000**
ULongMax	3575794432.000000	**3575842560.000000**	**3575775232.000000**

8.Performanzanalyse



9.Genauigkeit vs Performanz

$[0..10^9[$	$[10^9..ULONGMAX]$
➢ Performanz siegt über Genauigkeit ➢ LUT methode ➢ Grund: Der ganzzahlige Teil ist im Ergebnis am wichtigsten. In diesem Intervall spielt die Genauigkeit ihre Rolle im dezimalen Teil.	➢ Genauigkeit ist wichtiger ➢ Heron Verfahren ➢ Grund: Bei großen Zahlen beeinflusst die Genauigkeit auch den ganzzahligen Teil

10.Zusammenfassung

- Als Haupt Implementierung verwenden wir das Heron Verfahren da es die beste Genauigkeit anbietet und eine mittlere Laufzeit hat.

- Als reine Reihendarstellung bietet sich die Taylor Reihe , diese dauert länger als das Heron Verfahren und hat eine mittlere Genauigkeit.

- Die Look up Tabelle ist im gegenteil das schnellste aber geringere Genauigkeit anzeigt.

Vielen Dank für Ihre Aufmerksamkeit