Freie Universität Berlin

Institut für Informatik

AG Theoretische Informatik



Bachelorarbeit

**Was ist der beste Trade-off für die Anzahl der Finger in Fingerbäumen und wie kann man in der Praxis Informationen über die Struktur ausnutzen?**

Anna Schapiro

Gutachter: Prof. Dr. Wolfgang Mulzer

Verfasser: Anna Hannah Schapiro

Matrikel-Nr.: 5039650

E-Mail: anna@schapiro.berlin

Telefon: 0152 5706 6134

Abgabetermin: 24.08.2021 (Campus Management)

**Abstract**

In dieser Arbeit geht es darum externe Pointer – Finger – einzuführen und den Suchprozess in Datenstruckturen durch sie zu beschläunigen. Die Laufzeiten der Such-Funktion werden durch extra Finger in einer Datenstruktur von O(log n) auf O(log d) reduziert [2]. Diese Art der Suche heißt Distanzsuche, da von einer Distanz zur Wurzel der Suchauftrag startet.

Ein gängiges Beispiel einer Distanzsuche ist der Lazy-Finger. Der Finger merkt sich nur seine zuletzt verwendete Position in der Datenstrucktur und startet neue Suchanfragen von dieser Position. Unbrauchbar wird der Lazy-Finger, wenn abwechselnd immer ein maximales und ein mininimales Node in der Datenstruktur gesucht wird. Denn bei so einer Nutzung der Datenstruktur kommt der Lazy-Finger nicht mehr hinterher und verhält sich sogar schlechter als die herkönnliche Wurzelsuche.

Deshalb ist das Ziel dieser Arbeit, eine Simulation der Fingersuche zuprogramieren, denn ich denke die Position und Anzahl der Finger ist eine implizite Vorhersage, wie eine Datenstruktur verwendet wird. Es könnte performanter sein, wenn die Fingersuche sich an den Such-Anfragen der Datenstrucktur orientiert, als wenn man diese struckturelle Information nicht berücksichtigt wird und mit der Wurzelsuche gesucht wird.

**Abstract - english**

Inhaltsverzeichnis

[Abbildungsverzeichnis vii](#_Toc71033837)

[Tabellenverzeichnis ix](#_Toc71033838)

[1 Einleitung 1](#_Toc71033839)

[1.1 Problemumfeld 2](#_Toc71033840)

[1.2 Zielsetzung 3](#_Toc71033841)

[2 Begriffe 6](#_Toc71033842)

[2.1.1 Daten 6](#_Toc71033843)

[2.1.2 Datenstrucktur 6](#_Toc71033844)

[2.1.3 Baum 7](#_Toc71033845)

[2.1.4 Skipliste 8](#_Toc71033846)

[2.1.5 Finger 8](#_Toc71033847)

[2.1.6 Fingersearch und Fingersuche 8](#_Toc71033848)

[3 Aktueller Forschungsstand: Übersicht über Finger in 9](#_Toc71033849)

[4 Modellierung des Problems als (Graph) 10](#_Toc71033850)

[5. Theoretische Untersuchung der Schrankenvermutun 10](#_Toc71033851)

[5 Programmierung der Simulation 12](#_Toc71033852)

[6 Evaluation der Ergebnisse 13](#_Toc71033853)

[7 Einordnung der Ergebnisse anhand des Forschungsstandes 14](#_Toc71033854)

[8 Zusammenfassung 15](#_Toc71033855)

[9 Fazit und Ausblick 16](#_Toc71033856)

[Literaturverzeichnis 17](#_Toc71033857)

Abbildungsverzeichnis

[Abbildung 1.1: Beispiel 1 zum Einfügen einer Grafik 2](#_Toc416951569)

[Abbildung 2.1: Beispiel 2 zum einfügen einer Abbildung 7](#_Toc416951570)

[Abbildung 3.1: Beispiel 3 zum Einfügen einer Abbildung 10](#_Toc416951571)

Tabellenverzeichnis

[Tabelle 3.1: Beispiel für das Einfügen einer Tabelle 11](#_Toc416951531)

# Einleitung

Alle Datenstrukturen haben im Wesentlichen immer die gleichen Funktionen Search, Insert und Delete. In dieser Arbeit geht es darum externe Pointer, sogenannte Finger, einzuführen und den Suchprozess in Datenstruckturen durch sie zu beschläunigen.

Die Laufzeiten der Search-Funktion werden durch extra Finger in einer Datenstruktur von O(log n) auf O(log d) reduziert [2]. Viele theoretischen Ansätze versuchen dieses d möglichst klein zu halten und nennen diese Art der Suche Distanzsuche.

Ein gängiges Beispiel einer Distanzsuche ist der Lazy-Finger. Der Finger merkt sich nur seine zuletzt verwendete Position in der Datenstrucktur und startet neue Suchanfragen von dieser Position. Er ist besonders praktisch, falls die Ergebnisse der Search-Funktion dem lokalitäts Prinzip folgen, also struckturell nahe beisammen in der Datenstrucktur anzutreffen sind. Wenn die Distanzsuche sofort in diesem Bereich anfangen kann zusuchen, dann ist es logisch, dass Ergebnisse in der Datenstrucktur schneller zufinden sind, als wenn die Search-funktion von der Wurzel aus anfangen würde zu suchen.

Unbrauchbar wird der Lazy-Finger, wenn abwechselnd immer ein maximales und ein mininimales Node in der Datenstruktur gesucht wird. Denn bei so einer Nutzung der Datenstruktur kommt der Lazy-Finger nicht mehr hinterher und verhält sich sogar schlechter als die Wurzelsuche.

Man könnte mit der Min-Max Fingersuche diesem Problem begegen. Die Min-Max-Fingersuche besteht aus einem Minimalen Finger und einem Maximalen Finger der Datenstrucktur. Die Fingersuche würde selbst entscheiden, ob die Startposition der Distanzsuche vom Maximalen oder vom Minimalsten Node in der Datenstrucktur beginnen sollte.

Diese Fingersuche würde zwar bei den Suchanfragen welche immer das Maximum und das Minimum suchen oder zumindest in deren Nähe suchen deutsch besser performieren als der Lazy-Finger. Jedoch kann man sich auch bei dieser Fingersuche einen ungünstigen Fall konstuieren, wo die Wurzelsuche besser performiert als die Fingersuche.

Z.B. Könnte man annenehmen, dass die Datenstrucktur ein Binärer Suchbaum ist. Die am meisten gesuchten Nodes würden immer vom linken Teilbaum ganz rechts sein und anschließend der ganz linke Node vom rechten Teilbaum würde gesucht werden. Auch hier würde sich die Wurzelsuche im Vergleich zu der Distanzsuche schneller verhalten, denn die Distanzsuche müsste immer bis zur Wurzel hochgehen, nur um dann als normale Wurzelsuche weiterzusuchen.

Man könnte aus diesen Beispielen schließen, dass die Fingersuche am besten alle Finger, also Startpositionen der Distanzsuche, in alle hochfriquentierten Bereiche der Datenstrucktur stellen sollte. Am Sinnvollsten wären das Bereiche von den man aus der Vergangenheit weiß, dass in diesen Bereichen besonders oft gesucht wird.

Im Prinzip wäre somit die Aufgabe der Fingersuche eine Datenstrucktur zusein, welche das Wissen über die besonders oft nachgefragten Startpositionen der Distanzsuche in sich vereint.

Man könnte sagen, die Fingersuche lernt mit jeder neuen Suche dazu wie die Datenstrucktur genutzt wird und versucht für die Zukunft Prognosen über die Nutzung in sich zuspeichern.

## Problemumfeld

Wie in der Einleitung angedeutet, verhalten sich unterscheidliche Distanzsuchen unterscheidlich jenachdem, welche Anfragen an die Suche gestellt werden und auch wie oft diese dem Lokalitätsprinzip folgen.

Ich würde gerne die Suchanfragen deshalb mit einer Verteilung betrachten. Z.B. wenn immer nur nach dem Maximum der Datenstrucktur gefragt wird oder ob immer das Maximum und Minimum nachgefragt wird oder ob vielleicht eine andere Verteilung wie die Gaus-Verteilung usw… verwendet wird, wie performieren dann Fingersuchen? Vielleicht schaffe ich es in der Arbeit nachzuvollziehen, wie schnell unterscheidliche Fingersuchen sich an neue Verteilungen der Anfragen anpassen können?

Vermutlich werden Fingersuchen besonders nützlich sein bei sehr großen Datenstruckturen, welche nur an wenigen Clustern angefragt werden? Fingersuchen sind auch nützlich falls nicht die gesammte Datenstrucktur bekannt ist, weil sie z.b. unendlich wächst.

## Zielsetzung

Das Ziel dieser Arbeit ist, einige Konzepte der Fingersuche auf Binärbaume, auf Rotschwarz-Bäume, sowie Skiplisten anzuwenden und deren Laufzeiten zumessen.

Ich möchte gerne aus den Laufzeiten ableiten, ob verscheidenen Fingersuchen sich je nach Verteilung der Suchanfragen besonders nützlich oder träge verhalten.

Eine Simulation der Fingersuche ist hier sinnvoll, denn ich denke die Position und Anzahl der Finger ist eine implizite Vorhersage, wie die Datenstruktur verwendet wird. Diese Information möchte ich gerne in der Fingersuche zum Vorteil der Laufzeit ausnutzen und zeigen, dass dieses Wissen zu schnelleren Suchen führen kann.

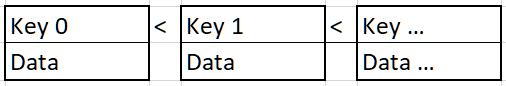
# Begriffe

### Daten

Laut dem Duden sind Daten durch Beobachtungen, Messungen, statistische Erhebungen u.a. gewonnene Zahlenwerte, welche auf Beobachtungen, Messungen, statistischen Erhebungen oder Angaben formulierbare Befunde sind. (<https://www.duden.de/rechtschreibung/Daten>, 04.05.2021, 14:00 Uhr)

In meiner Arbeit wird vorallem die Ordinale Eigenschaft der Daten vorrausgesetzt. Ordinal bedeutet, dass man die Daten ordnen oder einer Reihenfolge zuweisen kann. In dieser Arbeit haben Datenwerte deshlab jeweils einen Key, dieser Key unterliegt der ordinalen Eigenschaft. Insbesondere kann man über die Key‘s aussagen, dass wenn sie verglichen werden, es möglich ist, dass die Keys entweder im Verhältniss größer, gleich oder kleiner zueinender stehen. ([https://de.wikipedia.org/wiki/Ordinalzahl 04.05.2021](https://de.wikipedia.org/wiki/Ordinalzahl%2004.05.2021), 04.05.21 14:07 Uhr) ([https://www.unibw.de/hum-bildungswissenschaft/professuren/swm/methodenskripte/deskriptive-statistik.pdf 04.05.2021](https://www.unibw.de/hum-bildungswissenschaft/professuren/swm/methodenskripte/deskriptive-statistik.pdf%2004.05.2021), 14:11 Uhr )

Die Ordinale Eigenschaft kann auch durch Nummerierung oder zufälligem zuweisen des Keys zu den jeweiligen Datenwerten erreicht werden, sie muss keiner sinnvollen oder intepretierbaren Logik folgen, jedoch wäre das praktisch um mit Datenstruckturen wirklich effizient arbeiten zukönnen.



Die Kombination aus Datenwerten mit Key heißen Knoten. Knoten können in einer Liste wie in der oberen Abbilung angeordnet werden.

### Datenstrucktur

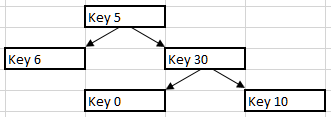
„Wie für viele fundamentale Begriffe der Informatik gibt es auch für […] Algorithmen und Datenstrukturen, nicht eine einzige, scharfe, allgemein akzeptierte Definition.“ ([https://www.fernuni-hagen.de/mi/studium/module/pdf/Leseprobe-komplett\_01662.pdf Seite 1](https://www.fernuni-hagen.de/mi/studium/module/pdf/Leseprobe-komplett_01662.pdf%20Seite%201), erster Satz, 04.05.13:50 Uhr). Daher halte ich mich an die Definition aus Wikipedia:

„In der Informatik und Softwaretechnik ist eine Datenstruktur ein Objekt, welches zur Speicherung und Organisation von Daten dient. Es handelt sich um eine Struktur, [in der] die Daten in einer bestimmten Art und Weise angeordnet und verknüpft werden, um den Zugriff auf sie und ihre Verwaltung effizient zu ermöglichen.“ ([https://de.wikipedia.org/wiki/Datenstruktur Absatz 1](https://de.wikipedia.org/wiki/Datenstruktur%20Absatz%201), 04.05.2021 13:54 Uhr)

In dieser Arbeit untersuchten Datenstruckturen haben drei wesentliche Opetationen, Zugriffsmöglichkeiten auf die Daten: insert, delete und search.

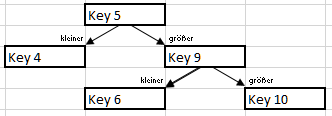
### Baum

Das ist eine Datenstruktur, mit dem sich hierarchische Strukturen abbilden oder erstellen lassen. Dabei können ausgehend von der Wurzel mehrere gleichartige Objekte mit den Keys miteinander verkettet werden, sodass eine lineare Struktur einer Liste aufgebrochen wird und eine Verzweigung stattfindet. Da Bäume zu den meistverwendeten Datenstrukturen in der Informatik gehören, gibt es viele Spezialisierungen und Deutungen. (<https://de.wikipedia.org/wiki/Baum_(Datenstruktur)> 04.05.2021 14:47 Uhr)



#### Binär-Baum

Binärbäume sind in der Informatik die an der häufigsten verwendeten Unterart der Bäume. Im Gegensatz zu anderen Arten von Bäumen können die Knoten eines Binärbaumes nur höchstens zwei direkte Nachkommen haben. Meistens werden an den jeweiligen Verbindungen oder Pointern ordinäre Vergleiche verwendet, wie größer oder kleiner als der aktuelle Knoten.

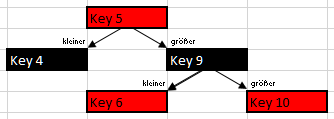


Die Binärsuche startet in der Wurzel – dem obersten Knoten - und folgt den Pointern zu den Kindern, den untergeordneten Knoten. Um diese Suche von der Fingersuche zu unterscheiden, nenne ich diese Suche Wurzelsuche.

Die Wurzelsuche hat eine Worst-case Laufzeit von O (log n), wobei n die Anzahl der Knoten ist. In meiner implementierung haben die Funktionen Insert und Delete eine Worst-case Laufzeit von O (1), wobei diesen beiden Operationen eine Suche vorgeschaltet ist.

#### Rot-Schwarz-Baum

Rotschwarz-Bäume sind eine spezielle Art der Binärbäume. Die enthaltenen Knoten werden je nach Ebene des Baumes entweder rot oder schwarz gefärbt. Diese Datenstrucktur hat den Vorteil, dass es einen Algorythnus der balancierung gibt, welcher die zusätzliche Information der unterscheidliochen Knotenfarben ausnutzt und deshlab schneller ist.



(<https://en.wikipedia.org/wiki/Red%E2%80%93black_tree> 04.05.2021 15:06 Uhr)

### Skipliste

### Finger

### Fingersearch und Fingersuche

# Aktueller Forschungsstand: Übersicht über Finger in

# Modellierung des Problems als (Graph)

5. Theoretische Untersuchung der Schrankenvermutun**g**

# Programmierung der Simulation

# Evaluation der Ergebnisse

# Einordnung der Ergebnisse anhand des Forschungsstandes

# Zusammenfassung

# Fazit und Ausblick



Abbildung 1.1: Beispiel 1 zum Einfügen einer Grafik

# Literaturverzeichnis

Brink, A. (2005). *Anfertigung wissenschaftlicher Arbeiten.* Oldenbourg.

Forbes, M. H. (1994). An exact algorithm for multiple depot bus scheduling. *European Journal of Operational Research, 72 (1)* , S. 115 - 124.

Gopen, G., & Swan, J. (1990). The Science of Scientific Writing If the reader is to grasp what the writer means, the writer must understand what the reader needs. *American Scientist 78 (6)* , S. 550 - 558.

Kalyan T. Talluri, G. J. (2005). *Revenue Management.* New York: Springer.

Kliewer, N. (2005). Optimierung des Fahrzeugeinsatzes im öffentlichen Personennahverkehr. *Dissertation* . Universität Paderborn.

Hiermit erkläre ich an Eides statt, dass ich die vorliegende Arbeit selbständig und ohne unerlaubte fremde Hilfe angefertigt, andere als die angegebenen Quellen und Hilfsmittel nicht benutzt und die den benutzten Quellen und Hilfsmitteln wörtlich oder inhaltlich entnommenen Stellen als solche kenntlich gemacht habe.

Berlin, den 1. Februar 2042

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(Unterschrift des Verfassers)