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](#_Toc73458947)

[Tabellenverzeichnis ix](#_Toc73458948)

[1 Einleitung 1](#_Toc73458949)

[1.1 Problemumfeld 1](#_Toc73458950)

[1.2 Zielsetzung 2](#_Toc73458951)

[2 Aktueller Forschungsstand: Übersicht über Finger 3](#_Toc73458952)

[2.1 Ein Finger 3](#_Toc73458953)

[2.2 Min und Max Finger 4](#_Toc73458954)

[2.3 Fingersuche mit Direktion 5](#_Toc73458955)

[2.4 Finger als Leafs 5](#_Toc73458956)

[2.5 Lazy Finger in Binär-Suchbäumen 6](#_Toc73458957)

[2.6 Splay trees und Fingersuche 7](#_Toc73458958)

[2.7 Prediktion auf Input Streams für Lokalitätsprinzip 7](#_Toc73458959)

[2.8 K-server Problem und multiple Finger 7](#_Toc73458960)

[3 Modellierung des Problems als (Graph) 9](#_Toc73458961)

[5. Theoretische Untersuchung der Schrankenvermutun 9](#_Toc73458962)

[4 Programmierung der Simulation 11](#_Toc73458963)

[5 Evaluation der Ergebnisse 12](#_Toc73458964)

[6 Einordnung der Ergebnisse anhand des Forschungsstandes 13](#_Toc73458965)

[7 Zusammenfassung 14](#_Toc73458966)

[8 Fazit und Ausblick 15](#_Toc73458967)

[Begriffe und Anhang 16](#_Toc73458968)

[8.1 Begriffe 16](#_Toc73458969)

[8.1.1 Lokalitätsprinzip 16](#_Toc73458970)

[8.1.2 Daten/Values 16](#_Toc73458971)

[8.1.3 Datenstrucktur 17](#_Toc73458972)

[8.1.4 Baum 17](#_Toc73458973)

[8.1.5 Skipliste 18](#_Toc73458974)

[8.1.6 Finger 19](#_Toc73458975)

[8.1.7 Fingersearch und Fingersuche 19](#_Toc73458976)

[8.1.8 Splay tree 19](#_Toc73458977)

[Literaturverzeichnis 21](#_Toc73458978)

Abbildungsverzeichnis

[Abbildung 8.1: Beispiel 1 zum Einfügen einer Grafik 15](#_Toc73458979)

Tabellenverzeichnis

# 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.

## Problemumfeld

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 festen Minimalen Finger und einem festen Maximalen Finger in 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.

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.

## Zielsetzung

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. Diese Progrone der Verwendung kann man natürlich durch statische Finger als Programmierer selbst setzen, oder man überlegt sich eine klügere Alternative.

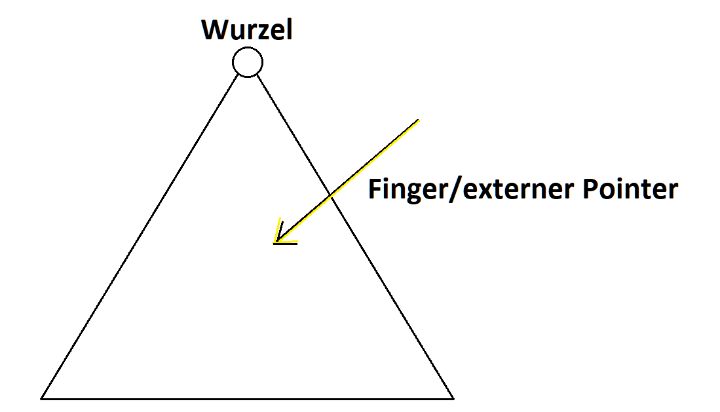
Ich möchte gerne aus den Laufzeiten ableiten, ob verscheidenen Fingersuchen sich je nach Verteilung der Suchanfragen besonders nützlich oder träge verhalten. Zu diesem Zweck erscheint mir sowohl eine theoretische Betrachtung als auch eine Simulation sinnvoll.

# Aktueller Forschungsstand: Übersicht über Finger

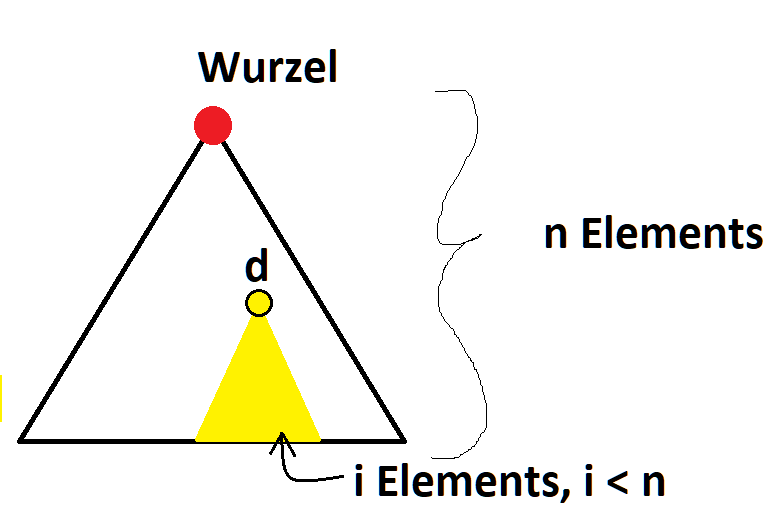
Dieses Kapitel soll ein Überblick der Finger, Fingersuchen und Datenstruckturen, wo sie angewendet werden, geben.

In dieser Arbeit gehe ich der übersichtshalber davon aus, dass eine Datenstrucktur beliebig, aber feste Anzahl n von Elementen in sich gespeichernt hat.

## Ein Finger

In diesem Paper (<https://www.cs.au.dk/~gerth/papers/finger05.pdf>) wird gleich im ersten Kapitel darauf hingewieden, dass Finger externe Pointer sind, welche genutzt werden, um von der Startposition d, statt von der Wurzel aus, zu suchen.

Hier wird vorrausgesetzt, dass der Finger schon so positioniert ist, dass die Fingersuche-Start-Position nächer zum Ergebniss liegt, als die Wurzelsuche und somit das Lokalitätsprinzip für das Suchergebniss und den Finger gilt.

Das führt zur Überlegung, dass die Wurzelsuch-startposition (rot) auch genannt Root-Finger (S. 181 <https://link.springer.com/content/pdf/10.1007%2F978-3-319-13075-0.pdf>)

in einem Binärbaum 2^n-1 Elemente durchsuchen kann um das Ergebniss zuliefern. Während der Ausschnitt (gelb), welcher durch den Finger-Startposition begrenzt wird, kann nach schon 2^i-1 Elementen das Ergebniss finden. Da i kleiner ist als n, kann man von einem Vorteil sprechen. Man könnte sogar sagen, je näher der Finger am Such-Ergebniss ist, desto exponenziell stärker ist dieser Effekt - veranschaulicht bedeutet das, dass die Fläche von dem gelben Bereich kleiner werden würde.

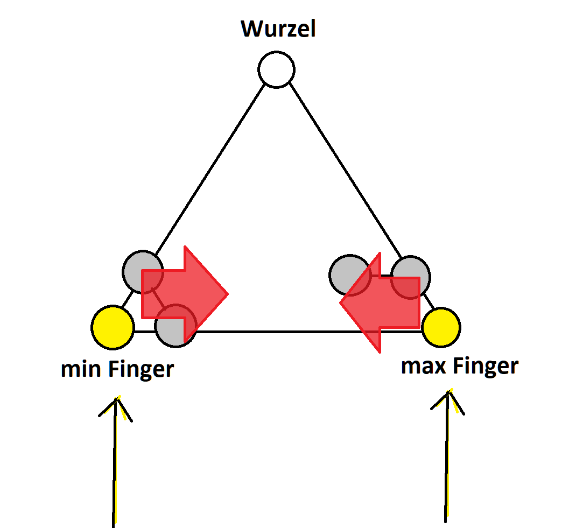
Mathematisch-Formal würde man sagen, die Wurzelsuche ist in der Laufzeitklasse O(log n), während die Distanz Suche auf i-Elementen arbeitet:

n > i = > O(log n) > O(log i).

Das ist der Vorteil von einem einzigen Finger.

## Min und Max Finger

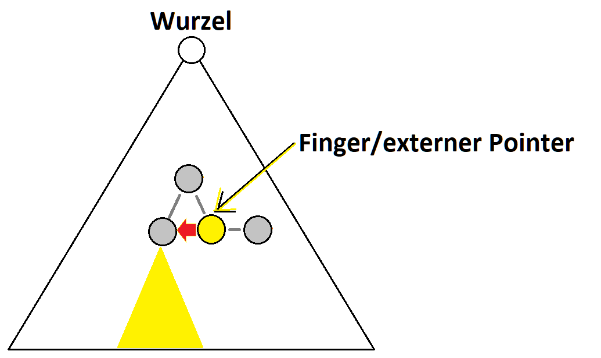
In diesem Buch (S 483, kap. 5.5 https://link.springer.com/content/pdf/10.1007/BF01940876.pdf ) wird die Idee unterbreitet der Fast Finger Search. Das Konzept beinhaltet, dass einfach von allen externen Fingern die Suche gleichzeitig gestartet werden soll. Wenn einer der Finger ein Ergebniss findet soll die Suche stoppen.

Als Spezial Fall wird der Min-Finger und der Max-Finger genannt. Der Min-Finger steht auf dem minimalen Node im Tree und der Max-Finger steht auf dem größten Node im Tree. Die Fingersuche beginnt zeitgleich. Diese Laufzeit ergibt deshlab:

O(log min{d, n - d})

In diesem Buch wird nähergelegt, dass versteckte Suchen wie in der Join, oder teile der Split-Operation auch von der Min-Max Fingersuche profitieren können, da sie schon auf Teilbäume zugreifen und man deshlab keine temporären Wurzeln von Teilbäumen zufinden braucht. (S 484 kap 5.7 und 5.9)

## Fingersuche mit Direktion

In diesem Paper aus dem Jahre 1977 (<http://citeseerx.ist.psu.edu/viewdoc/download;jsessionid=5A00C23223BF564B3107486113A2D1BE?doi=10.1.1.527.7294&rep=rep1&type=pdf> ) wird explizit eine Fingersuche vorgestellt, welche eine Suchrichtung besitzt. Wenn der Finger auf einer Position steht, kann es passieren, dass er seine Nachbar-Nodes durchsuchen muss, bevor eine runtergerichtete Suche - wie bei der Wurzelsuche - im Baum passiert.

Es wird auch der Fall betrachtet, dass Finger verschoben, also ein Update verlangen; am Beispiel von einer Insert Operation wird festgestellt, dass Finger unter umständen zu nahe beieinander liegen.

Finger werden in diesem Paper vorallem dazu genutzt um auf einzelne Nodes in Binären Suchbäumen, welche Listen enthalten, zuzugreifen.

In dieser Binärbaum-Finger Implementierung kann man also auf einer Ebene die Nodes durchsuchen, falls man einen Finger nicht sofort zur Verfügung hat.

## Finger als Leafs

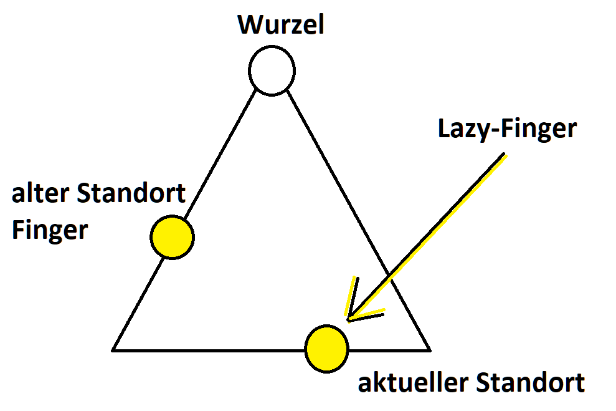
In diesem Buch seite 158, 3. Abs (http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.217.5131&rep=rep1&type=pdf ) werden auf Level-Linked-Binärbäumen Finger immer nur auf Leafs gesetzt. Es wird der Schluss gezogen, dass in der ammortisierten Laufzeit die Logarithmische Suche deutlich stärker zutragen kommt als die constante Laufzeit für die Erzeugung und Änderung der Finger.





Damit könnte man sagen, dass der wichtige zubetrachtende Aspekt in der Laufzeit die Fingersuche an sich ist.

## Lazy Finger in Binär-Suchbäumen

In einem etwas etwas aktuellerem Conference Paper aus dem Jahr 2014, S 181 (https://link.springer.com/content/pdf/10.1007%2F978-3-319-13075-0.pdf) werden die Laufzeiten von Lazy Fingern vorgestellt. Ein Lazy Finger ist ein Finger, der sich auf das zuletzt gesuchte Element im Baum stellt, und genau von diesem zuletzt gesuchten Element beginnt die neue Suchanfrage. Es wird sich mit der Fragestellung der Abstände von Suchergebniss und Lazyfinger beschäftigt und das im Zusammenhang von Entropien (Informationstheorie).

“Given the pairwise frequencies fX finding the tree that minimizes the execution time of search sequence X using lazy finger takes time O(n3).” Das wäre also die Laufzeit um eine Optimale Lazy-Finger Position in einem Binär-Baum zuberechnen. (3. Abs, S 190)

## Splay trees und Fingersuche

In diesem Paper aus dem Jahr 2018 (Absatz 3 https://drops.dagstuhl.de/opus/volltexte/2018/10003/pdf/LIPIcs-ISAAC-2018-55.pdf) steht in der Einleitung übersetzt: “Im Jahr 2000 haben Cole [16, 15] gezeigt, dass Splaytrees (asymptotisch) mit der Effizienz der Fingersuche übereinstimmt, die in diesem Zusammenhang als dynamische Fingereigenschaft bezeichnet wird. Dies ist bemerkenswert, da Splay keine expliziten Finger verwendet“

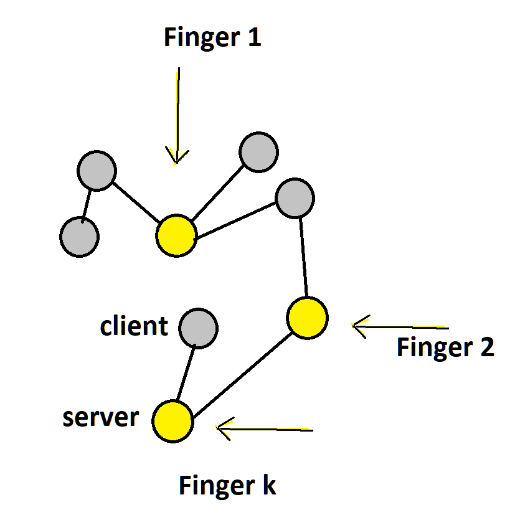
Das bedeutet, dass die Selbstanpassung, also die Splayfunktion im Splaytree selst sich wie eine indirekte Fingersuche verhalt. Die Suche in den Datenstruckturen ist also Laufzeittechnisch gleichwertig.

## Prediktion auf Input Streams für Lokalitätsprinzip

In diesem Paper 2003 (<https://dl.acm.org/doi/pdf/10.1145/781131.781159> ) wird der Sachverhalt der Vorhersage von Inputdaten auf verschiedene Systeme analysiert. Es wird explizit auf den Nutzen des Vorhersagens im Sinne des Lokalitätsprinzips erwähnt, und es wird ein Splaytree in diesem Zusammenhag benutzt, da er am schnellsten ist. Es gibt auch abgewandelte Variationen mit gewichteten Kanten oder geclusterten Nodes in Splaytree.

In diesem Paper wird nicht explizit von Fingern gesprochen, jedoch werden die Ergebnisse der Vorhersage für den Splaytree wie Finger verwendet.

## K-server Problem und multiple Finger



In diesem Paper aus dem Jahr 2018 (<https://drops.dagstuhl.de/opus/volltexte/2018/10003/pdf/LIPIcs-ISAAC-2018-55.pdf>) wird versucht dem k-server Problem mit k Fingern zubegegnen. Es wird davon ausgegangen, dass alle Server und alle Clients eine „Karte“, also die komplette Information zum Entscheiden haben.

Die Grundsätzliche Idee ist, dass jedem Server ein Finger zugewiesen wird und Anhand von diesen Fingern dann die nächstmöglichen Clients gefunden oder bedient werden. Der Algorithmus gibt also die Information zurück welcher Finger für welchen Client benutzt wurde.

# 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 8.1: Beispiel 1 zum Einfügen einer Grafik

# Begriffe und Anhang

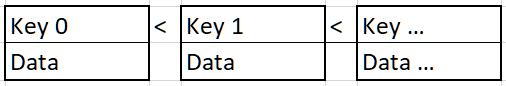
## Begriffe

### Lokalitätsprinzip

### Daten/Values

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) Daten und Values werden in dieser Arbeit analog sowohl in der schriftlichen Ausarbeitung als auch im Code für die Simulkation verwendet.

In meiner Arbeit wird vorallem die Ordinale Eigenschaft der Daten/Values vorrausgesetzt. Ordinal bedeutet, dass man die Daten ordnen oder einer Reihenfolge zuweisen kann. In dieser Arbeit haben Datenwerte/Values 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 zustellen. ([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 Kombination aus Datenwerten/Value mit Key heißen Knoten. Knoten können in einer Liste wie in der oberen Abbilung angeordnet werden.

### Datenstrucktur

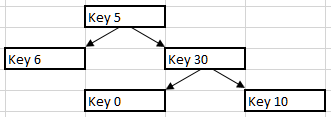
„Wie für viele fundamentale Begriffe der Informatik gibt es auch für […] Algorithmen und Datenstrukturen, nicht eine einzige, scharfe, allgemein akzeptierte Definition.“ ([https://www.fernuni-hagen.de/mi/studium/module/pdf/Leseprobe-komplett\_01662.pdf Seite 1](https://www.fernuni-hagen.de/mi/studium/module/pdf/Leseprobe-komplett_01662.pdf%20Seite%201), 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 oder Zugriffsmöglichkeiten auf die Daten: insert, delete und search.

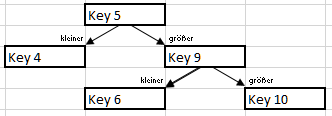
### Baum

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.

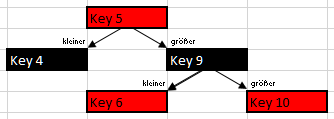


Die Binärsuche startet in der Wurzel – dem obersten Knoten - und folgt den Pointern zu den Kindern, den untergeordneten Knoten. Um diese Suche von der Fingersuche zu unterscheiden, nenne ich diese Suche Wurzelsuche.

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

Finger sind Pointer. In dieser Arbeit haben Finger die Besonderheit dass sie extern auf die jeweilige Datenstrucktur

### Fingersearch und Fingersuche

### Splay tree

Original paper https://dl.acm.org/doi/pdf/10.1145/3828.3835

# 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.

Kohm, M., & Morawski, J.-U. (2014). KOMA-Script, Die Anleitung. Abgerufen am 29. 10 2014 von https://www.rrzn.uni-hannover.de/fileadmin/kurse/

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)