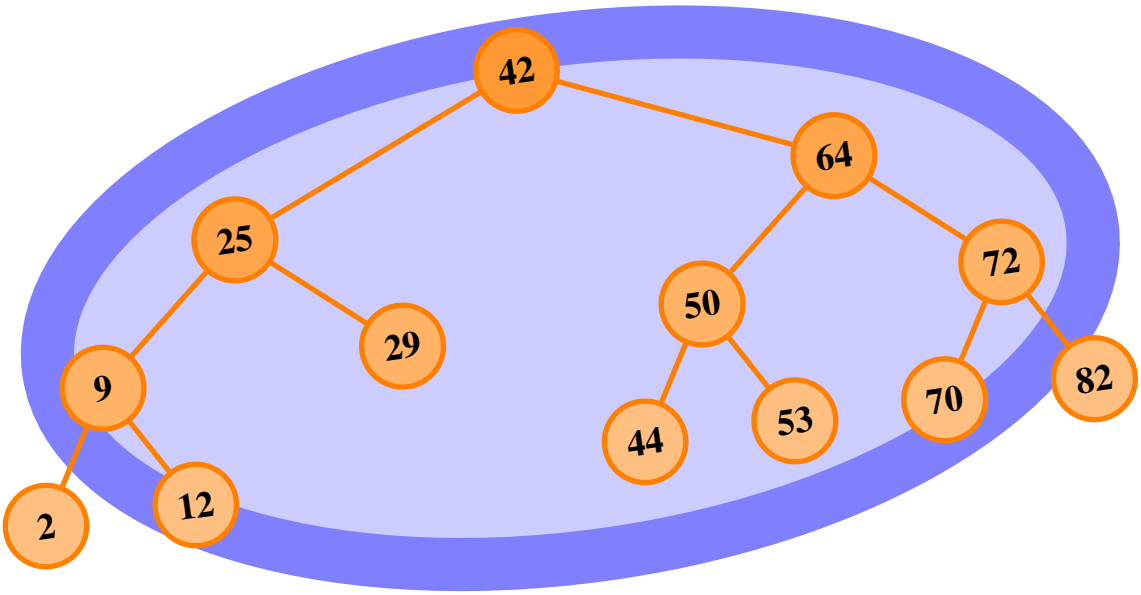
Leitprogramm Informatik

Einführung in verzweigte Datenstrukturen

Schultyp	Gymnasium/Gesamtschule
Jahrgangsstufe	EF/11
Fachliche Vorkenntnisse	Lineare Liste
Bearbeitungsdauer	$\sim$ 5-6 Schulstunden
Originalautor	Christian Baart
Erste Fassung	28. Mai 2012
Erste Schulerprobung	Jahrgangsstufe 11 an der Gesamtschule Barmen



Einleitung

Mit wachsenden Speichergrößen und immer günstiger werdenden Speicherpreisen ist absehbar, dass große Mengen von Daten verwaltet werden müssen. Dass hierfür effiziente Verfahren notwendig sind, ist einleuchtend. Niemand möchte erst lange warten, bis eine angeklickte MP3 auf der Festplatte lokalisiert und in den Speicher geladen wird – das Informatiksystem muss innerhalb von Sekundenbruchteilen »wissen«, wo genau die Datei sich befindet, um sie laden zu können.

Ein Verfahren zur Verwaltung von dynamisch wachsenden und sich verändernden Datenbeständen haben Sie bereits kennengelernt: Die lineare Liste. Mit ihrer Hilfe ist es möglich, Daten zu einer bestehenden Liste hinzuzufügen, Elemente daraus zu entfernen und die Liste mittels der linearen Suche elementweise zu durchsuchen.

Wie Sie sich vorstellen können, stößt diese Datenstruktur schnell an ihre Grenzen, wenn es darum gehen soll, »schnell mal« ein bestimmtes Element zu finden. Darum werden Sie in diesem Leitprogramm eine neue Datenstruktur kennen lernen, die Ihnen neue Möglichkeiten eröffnet. Wir beginnen damit allerdings nicht im oder am Informatiksystem, sondern stellen uns vor ein CD-Regal …

Inhaltsverzeichnis

Arbeitsanleitung

Ein Leitprogramm ist ein Unterrichtsheft zum selbstständigen Arbeiten. Das Leitprogramm besteht dabei aus mehreren Kapiteln, die Sie nacheinander durcharbeiten. Sie beginnen mit Kapitel 1 und melden sich, wenn Sie sich gut vorbereitet fühlen, bei der Informatiklehrkraft für den **Kapiteltest**. Wenn Sie diesen erfolgreich absolvieren, erhalten Sie das nächste Kapitel ausgehändigt.

Die einzelnen Kapitel setzen sich aus folgenden Bestandteilen zusammen:

Übersicht

Jedes Kapitel beginnt mit einer kurzen Übersicht darüber, was Sie in diesem Kapitel erwartet.

Lernziele

Mit den Lernzielen/Kompetenzen werden die Kenntnisse und Fähigkeiten angegeben, die Sie durch die Bearbeitung des Kapitels erwerben. Diese werden auch im anschließenden Kapiteltest überprüft.

Theorie

An dieser Stelle erhalten Sie theoretische Hinweise und Definitionen, die zum weiteren Bearbeiten des Leitprogramms wichtig sind.

Aufgabe 0.1

Aufgaben, die auf diese Weise gekennzeichnet sind, bearbeiten Sie mit Stift und Papier.

Aufgabe 0.2

An einigen Stellen werden Sie auch Aufgaben direkt mit einem Informatiksystem bearbeiten.

–

Lösungen

Zu allen Aufgaben gibt es am Ende jedes Kapitels die zugehörigen Lösungen, mit denen Sie Ihre Antworten selbstständig überprüfen können. Verwenden Sie diese jedoch erst, wenn Sie sich hinreichend mit einer Aufgabe beschäftigt haben.

Nothelfer

Unter einigen Aufgaben finden Sie einen sogenannten Nothelfer. Dieser kann Ihnen Tipps und Anregungen geben, die zum Bearbeiten der Aufgaben hilfreich sind. Ihnen steht frei, zu entscheiden, ob Sie diese verwenden wollen oder nicht.

Lernkontrolle

Mithilfe der Lernkontrolle können Sie feststellen, ob Sie gut auf den Kapiteltest vorbereitet sind. Wenn Sie die Lernkontrolle ohne Probleme lösen können, können Sie beruhigt zum Kapiteltest gehen. Ansonsten solltest Sie die Stellen noch einmal wiederholen, an denen Probleme aufgetreten sind.

In der Regel werden Sie das Leitprogramm selbstständig bearbeiten. Bei Fragen oder Problemen können Sie sich aber gerne an Ihre Lehrkraft wenden.

Kapitel 1

Das neue CD-Regal

Übersicht

Das erste Kapitel frischt Ihr Vorwissen zur linearen Liste, sowie deren Nachteile auf, wenn es um die Suche eines bestimmten Elementes geht. Im Laufe des Kapitels entwickeln Sie eine Methode, die sich von der linearen Suche in einer Liste abgrenzt und sich als wesentlich effektiver erweist.

Lernziele

Nach der Bearbeitung dieses Kapitels…

- sind Sie in der Lage, zu benennen, warum sich die lineare Liste nicht zur Verwaltung großer Datenmengen eignet.
- kennen Sie das Prinzip der »Binären Suche« und können es auf eine Menge von Elementen anwenden.

1.1 Auf der Suche nach Sunrise Avenue

Trotz der Möglichkeit, sich Lieder über Online-Shops direkt herunterzuladen, haben Sie eine große Sammlung von Musik-CDs in einer Schachtel in Ihrem Zimmer stehen. Als Sie eines Tages aus der Schule kommen, strahlt Ihre Mutter Sie an: »Heute sind die neuen Regale für dein Zimmer gekommen. Ich habe sie direkt mal angebracht und deine CDs eingeräumt. Schau mal, wie’s dir gefällt!«

In Ihrem Zimmer nehmen Sie die Arbeit Ihrer Mutter in Augenschein – alle Ihre CDs sind fein säuberlich aufgereiht. Da Ihnen jetzt der Sinn nach etwas Musik steht, entschließen Sie sich, ein Album aus Ihrer Sammlung herauszusuchen und sich danach an die Hausaufgaben zu machen.

Flogging Molly – Speed of Darkness
Fall Out Boy – Folie à Deux
die ärzte – auch
Linkin Park – A Thousand Suns
Green Day – American Idiot
AC/DC – Back in Black
Alan Jackson – Good Time
Script – Science & Faith
Lostprophets – Weapons
The Black Keys – El Camino
Katy Perry – Teenage Dream
Rizzle Kicks – Stereo Typical
One Direction – Up all Night
Gotye – Making Mirrors
Alabama Shakes – Boys & Girls
Alpa Gun – Ehrensache
Kings of Leon – Only By The Night
Boy – Mutual Friends
Ed Sheeran – +
Culcha Candela – Flätrate
Rea Garvey – Can't Stand The Silence
Eisblume – Ewig
Rihanna – Talk That Talk
Mike Candys – Smile
Luxuslärm – Carousel
Die Toten Hosen – Tage wie dieser
Michel Teló – Ai Se Eu Te Pego
Coldplay – Mylo Xyloto
Nickelback – Here and Now
Stefanie Heinzmann – Stefanie Heinzmann
Sunrise Avenue – Out of Style
Unisonic – Unisonic
Roxette – Travelling
Caligola – Back to Earth
Tim Bendzko – Wenn Worte meine Sprache wären
David Guetta – Nothing but the Beat
The Overtones – Gambling Man
Kraftclub – Mit K
The Boss Hoss – Liberty of Action
Bruce Springsteen – Wrecking Ball
Lana Del Rey – Born to die
Madonna – M D N A
Katie Melua – Secret Symphony
Adele – 21
Deichkind – Befehl von ganz unten
Ivy Quainoo – Ivy
Santiano – Bis ans Ende der Welt
Xavier Naidoo – Danke fürs Zuhören
Silbermond – Himmel auf
Unheilig – Lichter der Stadt

Aufgabe 1.1

(a) Suchen Sie das Album von Sunrise Avenue aus Ihrer CD-Sammlung oben heraus. Notieren Sie dabei, wie viele CDs Sie sich angesehen haben, bis Sie das Album gefunden haben.

(b) Suche Sie nun das Album von Linkin Park aus der CD-Sammlung heraus. Notieren Sie sich auch hier, wie viele CDs Sie sich angesehen haben.

(c) Beschreiben Sie, welchen Zusammenhang Sie zu einer Ihnen bekannten Datenstruktur sehen. Welche Nachteile ergeben sich bei der Suche nach einer bestimmten CD? Und was bedeutet dies für den Fall, dass Sie nach einer CD suchen, die gar nicht im Regal steht?

(d) Sehen Sie Möglichkeiten, die Suche zu beschleunigen?

1.2 Geht das auch einfacher?

Einen arbeitsreichen Nachmittag später haben Sie die CDs in eine neue Reihenfolge gebracht: sie sind jetzt nach den Namen der Interpreten aufsteigend sortiert.

Aufgabe 1.2

(a) Suchen Sie nun erneut die CD von Sunrise Avenue aus Ihrer unten stehenden Sammlung heraus und notieren Sie sich wiederum die Anzahl der CDs, die Sie sich ansehen mussten. Verfahren Sie ebenso mit dem Album von Linkin Park, sowie mit dem Album von Michael Jackson.

(b) Tauschen Sie sich anschließend mit einem Partner darüber aus, wie Sie bei der Suche vorgegangen sind. Lässt sich diese Vorgehensweise verallgemeinern?

AC/DC – Back in Black
Adele – 21
Alabama Shakes – Boys & Girls
Alan Jackson – Good Time
Alpa Gun – Ehrensache
Boy – Mutual Friends
Bruce Springsteen – Wrecking Ball
Caligola – Back to Earth
Coldplay – Mylo Xyloto
Culcha Candela – Flätrate
David Guetta – Nothing but the Beat
Deichkind – Befehl von ganz unten
Die Toten Hosen – Tage wie dieser
Die Ärzte – auch
Ed Sheeran – +
Eisblume – Ewig
Fall Out Boy – Folie à Deux
Flogging Molly – Speed of Darkness
Gotye – Making Mirrors
Green Day – American Idiot
Ivy Quainoo – Ivy
Katie Melua – Secret Symphony
Katy Perry – Teenage Dream
Kings of Leon – Only By The Night
Kraftclub – Mit K
Lana Del Rey – Born to die
Linkin Park – A Thousand Suns
Lostprophets – Weapons
Luxuslärm – Carousel
Madonna – M D N A
Michel Teló – Ai Se Eu Te Pego
Mike Candys – Smile
Nickelback – Here and Now
One Direction – Up all Night
Rea Garvey – Can't Stand The Silence
Rihanna – Talk That Talk
Rizzle Kicks – Stereo Typical
Roxette – Travelling
Santiano – Bis ans Ende der Welt
Script – Science & Faith
Silbermond – Himmel auf
Stefanie Heinzmann – Stefanie Heinzmann
Sunrise Avenue – Out of Style
The Black Keys – El Camino
The Boss Hoss – Liberty of Action
The Overtones – Gambling Man
Tim Bendzko – Wenn Worte meine Sprache wären
Unheilig – Lichter der Stadt
Unisonic – Unisonic
Xavier Naidoo – Danke fürs Zuhören

1.3 Die binäre Suche

Eine Möglichkeit, sich die Suche in einer sortierten Menge zu vereinfachen, besteht in der sogenannten *binären Suche*. Die Grundidee liegt darin, mit jedem Suchschritt die Anzahl der möglichen Elemente zu verkleinern und sich so insgesamt nur wenige Elemente anschauen zu müssen. Als Menschen haben wir einen Vorteil gegenüber dem Informatiksystem, wenn es darum geht, nach bestimmten Elementen zu suchen. Vor allem bei einer noch einigermaßen übersichtlichen Menge von Daten, wie den 50 CDs, können wir uns häufig auf Erfahrungswerte berufen:

Wenn wir beispielsweise nach einem Interpreten mit dem Anfangsbuchstaben **M** suchen und bei unserer Suche beim Buchstaben N ankommen, werden wir direkt in der unmittelbaren Umgebung dieses Buchstabens suchen. Problematisch wird diese Methode jedoch, wenn es sich statt 50 um beispielsweise 10.000 CDs handelt. Denn in diesem Fall können zwischen zwei Anfangsbuchstaben leicht 400 CDs liegen. Also braucht man eine festgelegte Vorgehensweise, die allgemeingültig ist und mit jedem Suchschritt möglichst viele CDs ausschließt.

Aufgabe 1.3

- Überlegen Sie sich eine Methode, wie man mit jedem Suchschritt eine gewisse Anzahl an CDs direkt ausschließen kann. **Tipp:** Beginnen Sie Ihre Suche in der Mitte des Regals. Wie viele CDs können Sie so direkt nach dem ersten Vergleich ausschließen?
- Wenden Sie diese Methode anschließend auf das Regal an und suchen Sie zwei CDs heraus. Notieren Sie die Anzahl der CDs, die Sie dabei betrachten mussten und vergleichen Sie diese mit der Anzahl der benötigten Versuche in den vorherigen Durchgängen.

Nothelfer

Betrachten Sie nach jedem Vergleich immer nur noch die Hälfte der CDs und überlegen Sie, ob Sie die linke oder rechte Hälfte betrachten müssen.

Das so entstandene Verfahren nennt man die binäre Suche. Binär leitet sich dabei vom lateinischen *binarius* ab und bedeutet so viel wie »zweiteilig«. Der Name erklärt sich daher, dass sich bei jedem Suchschritt die Anzahl der zu betrachtenden Elemente halbiert. Dies ist ein enormer Fortschritt gegenüber der ersten Version unser linearen Suche.

Im nächsten Kapitel werden Sie sehen, wie man aus der Erkenntnis der binären Suche eine neue Datenstruktur entwickeln kann, die schon durch ihren Aufbau die Suche nach bestimmten Elementen unterstützt.

Lernkontrolle

Aufgabe 1: Worin liegen Nachteile der Suche in einer linearen Liste?

- ☐ Die Suche kann bei einer großen Anzahl von Elementen sehr lange dauern.
- ☐ Die Elemente müssen vor dem Durchlaufen sortiert werden.
- ☐ Es müssen immer alle Elemente von Anfang an durchlaufen werden, auch wenn das letzte Element gesucht wird.

Aufgabe 2: Die binäre Suche funktioniert in jeder beliebigen Liste von Elementen.

- ☐ Richtig
- ☐ Falsch

Aufgabe 3: Wenden Sie die binäre Suche auf die folgende Menge an, um das Element »Frankreich« zu finden. Markieren Sie dabei alle Elemente, die Sie während Ihrer Suche betrachtet haben.

1	Afghanistan
2	Ägypten
3	Albanien
4	Algerien
5	Andorra
6	Angola
7	Antigua und Barbuda
8	Äquatorialguinea
9	Argentinien
10	Armenien
11	Aserbaidschan
12	Äthiopien
13	Australien
14	Bahamas
15	Bahrain
16	Bangladesh
17	Barbados
18	Belgien
19	Belize
20	Benin
21	Bhutan
22	Bolivien
23	Bosnien-Herzegowina
24	Botswana
25	Brasilien
26	Brunei
27	Bulgarien
28	Burkina-Faso
29	Burundi
30	Chile
31	China
32	Costa Rica
33	Côte d'Ivoire
34	Dänemark
35	Deutschland
36	Dominica
37	Dominikanische Republik
38	Ecuador
39	El Salvador
40	Eritrea
41	Estland
42	Fidschi
43	Finnland
44	Frankreich
45	Gabun
46	Gambia
47	Georgien
48	Ghana
49	Grenada
50	Griechenland
51	Großbritannien
52	Guatemala
53	Guinea
54	Guinea-Bissau
55	Guyana
56	Haiti
57	Honduras
58	Indien
59	Indonesien
60	Irak
61	Iran
62	Irland
63	Island
64	Israel
65	Italien
66	Jamaika
67	Japan
68	Jemen
69	Jordanien
70	Jugoslawien
71	Kambodscha
72	Kamerun
73	Kanada
74	Kap Verde
75	Kasachstan
76	Katar
77	Kenia
78	Kirgisistan
79	Kiribati
80	Kolombien
81	Komoren
82	Kongo, Republik
83	Kroatien
84	Kuba
85	Kuwait
86	Laos
87	Lesotho
88	Lettland
89	Libanon
90	Liberia
91	Libyen
92	Liechtenstein
93	Litauen
94	Luxemburg
95	Madagaskar
96	Makedonien
97	Malawi
98	Malaysia
99	Malediven
100	Mali
101	Malta
102	Marokko
103	Marshallinseln
104	Mauritanien
105	Mauritius
106	Mexiko
107	Mikronesien
108	Moldawien
109	Monaco
110	Mongolei
111	Mosambik
112	Myanmar
113	Namibia
114	Nauru
115	Nepal
116	Neuseeland
117	Nicaragua
118	Niederlande
119	Niger
120	Nigeria
121	Nordkorea
122	Norwegen
123	Oman
124	Österreich
125	Pakistan
126	Palau-Inseln
127	Panama
128	Papua-Neuguinea
129	Paraguay
130	Peru
131	Philippinen
132	Polen
133	Portugal
134	Ruanda
135	Rumänien
136	Russland
137	Saint Kitts-Nevis
138	Saint Lucia
139	Saint Vincent und Grenadinen
140	Sambia
141	San Marino
142	Sao Tomé und Príncipe
143	Saudi-Arabien
144	Schweden
145	Schweiz
146	Senegal
147	Seychellen
148	Sierra Leone
149	Simbabwe
150	Singapur
151	Slowakische Republik
152	Slowenien
153	Solomonen
154	Somalia
155	Spanien
156	Sri Lanka
157	Südafrika
158	Sudan
159	Südkorea
160	Surinam
161	Swasiland
162	Syrien
163	Tadschikistan
164	Taiwan
165	Tansania
166	Thailand
167	Togo
168	Tonga
169	Trinidad und Tobago
170	Tschad
171	Tschechische Republik
172	Tunesien
173	Türkei
174	Turkmenistan
175	Tuvalu
176	Uganda
177	Ukraine
178	Ungarn
179	Uruguay
180	Usbekistan
181	Vanuatu
182	Vatikanstadt
183	Venezuela
184	Vereinigte Arabische Emirate
185	Vereinigte Staaten von Amerika
186	Vietnam
187	Weißrußland
188	Westsahara
189	Westsamoa
190	Zaire
191	Zentralafrikanische Republik
192	Zypern

Lösungen zu Kapitel 1:

Aufgabe 1.1:

(c) Das CD-Regal lässt sich durch eine lineare Liste repräsentieren. Da die CDs keinerlei Ordnung unterliegen, muss jede CD einzeln betrachtet werden, um ein bestimmtes Album zu finden. Um festzustellen, dass eine CD nicht vorhanden ist, muss also immer die komplette Liste durchlaufen werden – in diesem Fall sind es 50 Einträge.

(d) Eine Möglichkeit, die Suche zu beschleunigen, besteht darin, die CDs in irgendeiner sinnvollen Weise zu sortieren, beispielsweise nach Interpreten oder Albumnamen.

Aufgabe 1.2: Da jeder Mensch bei der Suche ein wenig anders vorgehen wird, lässt sich an dieser Stelle nur eine allgemeine Aussagen treffen. Dadurch, dass die CDs sortiert sind, bietet sich ein enormer Vorteil. So kann die Suche allein dadurch beschleunigt werden, dass man intuitiv abschätzt, an welcher Stelle etwa ein Buchstabe liegen könnte und sich anschließend nur die Alben im näheren Umfeld ansieht. Durch die Sortierung wird gleichzeitig eine »Richtung« vorgegeben, in die man die Suche fortsetzen muss. Sucht man beispielsweise eine CD vom Interpreten Luxuslärm und beginnt seine Suche beim Buchstaben »O«, so ist klar, dass die Suche im linken Teil der Liste weitergehen muss.

Um festzustellen, dass ein Album nicht im Regal vorhanden ist, genügt es, sich die Alben anzusehen, die denselben Anfangsbuchstaben haben – somit kann auch für diesen Fall die Anzahl der Suchvorgänge erheblich reduziert werden.

Aufgabe 1.3: Umgangssprachlich lässt sich die Suche folgendermaßen formulieren:

- Betrachte das mittlere Element!
- Ist es das gesuchte Element?

  JA Gib das Element aus und beende die Suche

  NEIN Liegt das gesuchte Element links oder rechts vom aktuellen Element?

    LINKS Springe zur Hälfte der linken Teilliste und beginne dort von vorne

    RECHTS Springe zur Hälfte der rechten Teilliste und beginne dort von vorne

Sollte der Fall eintreten, dass die verbleibende Liste nicht mehr halbiert werden kann, so ist das gesuchte Album nicht darin enthalten.

Wir verdeutlichen uns den Ablauf an einem Beispiel. Zum besseren Verständnis sind die einzelnen Alben im nachfolgenden Bild nummeriert worden. Wir suchen nach dem Album von Coldplay.

Schritt 1: Da es sich um eine gerade Anzahl von CDs handelt, berechnen wir das mittlere Element durch $\frac{50}{2} = 25$.

1 – AC/DC – Back in Black
2 – Adele – 21
3 – Alabama Shakes – Boys & Girls
4 – Alan Jackson – Good Time
5 – Alpa Gun – Ehrensache
6 – Boy – Mutual Friends
7 – Bruce Springsteen – Wrecking Ball
8 – Caligola – Back to Earth
9 – Coldplay – Mylo Xyloto
10 – Culcha Candela – Flätrate
11 – David Guetta – Nothing but the Beat
12 – Deichkind – Befehl von ganz unten
13 – Die Toten Hosen – Tage wie dieser
14 – Die Ärzte – auch
15 – Ed Sheeran – +
16 – Eisblume – Ewig
17 – Fall Out Boy – Folie a Deux
18 – Flogging Molly – Speed of Darkness
19 – Gotye – Making Mirrors
20 – Green Day – American Idiot
21 – Ivy Quainoo – Ivy
22 – Katie Melua – Secret Symphony
23 – Katy Perry – Teenage Dream
24 – Kings of Leon – Only By The Night
25 – Kraftclub – Mit K
26 – Lana Del Rey – Born to die
27 – Linkin Park – A Thousand Suns
28 – Lostprophets – Weapons
29 – Luxuslärm – Carousel
30 – Madonna – M D N A
31 – Michel Teló – Ai Se Eu Te Pego
32 – Mike Candys – Smile
33 – Nickelback – Here and Now
34 – One Direction – Up all Night
35 – Rea Garvey – Can't Stand The Silence
36 – Rihanna – Talk That Talk
37 – Rizzle Kicks – Stereo Typical
38 – Roxette – Travelling
39 – Santiano – Bis ans Ende der Welt
40 – Script – Science & Faith
41 – Silbermond – Himmel auf
42 – Stefanie Heinzmann – Stefanie Heinzmann
43 – Sunrise Avenue – Out of Style
44 – The Black Keys – El Camino
45 – The Boss Hoss – Liberty of Action
46 – The Overtones – Gambling Man
47 – Tim Bendzko – Wenn Worte meine Sprache wären
48 – Unheilig – Lichter der Stadt
49 – Unisonic – Unisonic
50 – Xavier Naidoo – Danke fürs Zuhören

Beim Interpreten von Album 25 handelt es sich um Kraftclub. Unser gesuchtes Album liegt also in der linken Hälfte der Liste, alle Elemente rechts der Mitte und die Mitte selbst brauchen wir nicht mehr weiter zu betrachten.

Schritt 2: Schon nach dem ersten Schritt hat sich Anzahl der Elemente beträchtlich verkleinert. Da wir nun noch über 24 Elemente verfügen, betrachten wir jetzt Element 12 – Deichkind.

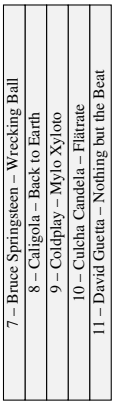
1 – AC/DC – Back in Black
2 – Adele – 21
3 – Alabama Shakes – Boys & Girls
4 – Alan Jackson – Good Time
5 – Alpa Gun – Ehrensache
6 – Boy – Mutual Friends
7 – Bruce Springsteen – Wrecking Ball
8 – Caligola – Back to Earth
9 – Coldplay – Mylo Xyloto
10 – Culcha Candela – Flätrate
11 – David Guetta – Nothing but the Beat
12 – Deichkind – Befehl von ganz unten
13 – Die Toten Hosen – Tage wie dieser
14 – Die Ärzte – auch
15 – Ed Sheeran – +
16 – Eisblume – Ewig
17 – Fall Out Boy – Folie à Deux
18 – Flogging Molly – Speed of Darkness
19 – Gotye – Making Mirrors
20 – Green Day – American Idiot
21 – Ivy Quainoo – Ivy
22 – Katie Melua – Secret Symphony
23 – Katy Perry – Teenage Dream
24 – Kings of Leon – Only By The Night

Das Album ist wieder nicht das gesuchte, es verrät uns aber, dass unser Element wiederum in der linken Teilliste liegen muss.

Schritt 3: Wir wiederholen das bekannte Verfahren für die restlichen 11 Elemente. Da es sich um eine ungerade Anzahl von Elementen handelt, runden wir das Ergebnis auf und erhalten so $\frac{11}{2} = 5,5 \approx 6$. Das mittlere Element ist in diesem Fall: »Boy«. Unsere CD liegt demnach in der rechten Hälfte.

1 – AC/DC – Back in Black
2 – Adele – 21
3 – Alabama Shakes – Boys & Girls
4 – Alan Jackson – Good Time
5 – Alpa Gun – Ehrensache
6 – Boy – Mutual Friends
7 – Bruce Springsteen – Wrecking Ball
8 – Caligola – Back to Earth
9 – Coldplay – Mylo Xyloto
10 – Culcha Candela – Flätrate
11 – David Guetta – Nothing but the Beat

Schritt 4: Betrachten wir nun erneut das mittlere Element der verbleibenden Liste – Element 9 –, so zeigt sich, dass wir unser Element nach lediglich 4 Schritten gefunden haben.



Lernkontrolle

Aufgabe 1: Worin liegen Nachteile der Suche in einer linearen Liste?

- ☒ Die Suche kann bei einer großen Anzahl von Elementen sehr lange dauern.
- ☐ Die Elemente müssen vor dem Durchlaufen sortiert werden.
- ☒ Es müssen immer alle Elemente von Anfang an durchlaufen werden, auch wenn das letzte Element gesucht wird.

Aufgabe 2: Die binäre Suche funktioniert in jeder beliebigen Menge von Elementen.

- ☐ Richtig
- ☒ Falsch – Die Elemente müssen einer Ordnung unterliegen und sortierbar sein.

Aufgabe 3: Reihenfolge der besuchten Elemente:

- Wir beginnen die Suche bei Element $\frac{192}{2} = 96$, Makedonien. Das gesuchte Element befindet sich also unterhalb von Element 96.
- Es verbleiben nun noch 95 Elemente, da Makedonien ebenfalls ausgeschlossen werden kann. Die Mitte ergibt sich also als $\frac{95}{2} = 47,5 \approx 48$, Ghana. Wiederum kein Treffer, es verbleiben 47 Elemente.
- Die neue Mitte ergibt sich als $\frac{47}{2} = 23,5 \approx 24$, Botswana. Das gesuchte Element liegt oberhalb von Element 24. Wir betrachten jetzt nur noch die Elemente von 25 bis 47, da alle anderen bereits ausgeschlossen wurden.
- Wir betrachten noch 23 Elemente, also bestimmen wird die Mitte: $\frac{23}{2} = 11,5 \approx 12$. Da wird nun nicht mehr die Elemente ab der 1 mitbetrachten, benötigen wir das zwölfte der verbleibenden Elemente, Nr. 36 – Dominica, welches ebenfalls nicht das gesuchte Element ist. Demnach betrachten wir noch die Elemente 37 bis 47.

- Erneutes Berechnen des mittleren Elementes liefert $\frac{11}{2} = 5,5 \approx 6$, und somit Element 42 – Fidschi. Es verbleiben die Elemente 43 bis 47.
- Die Mitte berechnet sich als $\frac{5}{2} = 2,5 \approx 3$, also Element 45, Gabun. Wir betrachten noch die Elemente 43 bis 44.
- Die Mitte berechnet sich für diese zwei Elemente als $\frac{2}{2} = 1$, welche somit Element 44, Frankreich, ist.

Die Suche endet damit nach insgesamt 7 betrachteten Elementen.

Kapitel 2

Der erste Baum

Übersicht

In diesem Kapitel dreht sich alles darum, aus den Ergebnissen der binären Suche eine neue Datenstruktur zu konzipieren, die deren Vorteile in sich beinhaltet.

Lernziele

Nach Bearbeitung dieses Kapitels

- können Sie die Eigenschaften eines binären Suchbaums und dessen einzelnen Bestandteile benennen.
- können Sie einen binären Baum auf eine bestimmte Art und Weise durchlaufen.

2.1 Wir entwerfen eine neue Datenstruktur

Nachdem Sie im letzten Kapitel das Prinzip der »Binären Suche« kennengelernt haben, soll es in diesem Kapitel darum gehen, eine Datenstruktur zu entwickeln, die die Vorteile der binären Suche ausnutzt. Der Einfachheit halber arbeiten wir an dieser Stelle zuerst nur mit einer Menge von Zahlen. Generell lässt sich dieses Prinzip aber auf jede beliebige Art von Elementen übertragen – vorausgesetzt, dass die Elemente auf irgendeine Art und Weise miteinander vergleichbar sind (oder mathematisch ausgedrückt: einer Ordnungsrelation unterliegen). Wenn das Prinzip soweit klar ist, werden wir uns wieder unserer CD-Sammlung zuwenden.

Das Problem an der linearen Liste ist, dass die Zahlen nach wie vor darin nur in aufsteigend sortierter Reihenfolge vorliegen, ein direkter Zugriff auf die Elemente aber so, wie wir sie in der binären Suche benötigen, nicht möglich ist. Was liegt also näher, als eine neue Datenstruktur zu entwickeln, die die Vorteile der Suche nach einem bestimmten Element schon bei ihrem Aufbau berücksichtigt und so die Suchanfragen minimiert.

Damit wollen wir uns auf den folgenden Seiten beschäftigen.

Aufgabe 2.1

Gegeben ist die Liste der Zahlen von 1 bis 9 in aufsteigender Reihenfolge:

1, 2, 3, 4, 5, 6, 7, 8, 9.

(a) Wenden Sie die binäre Suche an, um das Element 2 in der Liste zu finden. Notieren Sie dabei wiederum jede Zahl, die Sie betrachten. Verfahren Sie ebenso mit der Zahl 9.

Erinnerung: Betrachten wir eine ungerade Zahl von Elementen, runden wir das Ergebnis für die Mitte auf!

(b) Überlegen Sie eine Möglichkeit, Ihr Ergebnis und der Weg, den Sie von Element zu Element gegangen sind, graphisch in einer Zeichnung darzustellen. Beachten Sie dabei auch, in welche »Richtung« Sie sich von Element zu Element bewegt haben. Vervollständigen Sie diese Zeichnung so, dass die Suche nach allen Elementen (also auch nach der 1,3,4,...) daraus abgelesen werden kann. (Schauen Sie nach dieser Aufgabe noch nicht in die Lösungen, sondern bearbeiten Sie erst Aufgabe (c).)

(c) Zeichnen Sie Ihre Idee auf ein DIN A4-Blatt und vergleichen Sie diese mit einem Partner. Versuchen sie, ihre Ideen zu kombinieren, dass eine sinnvolle Darstellung entsteht.

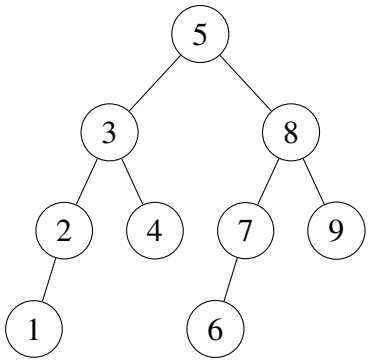
Sollten Sie mit der Aufgabe nicht klarkommen, können Sie sich im Kasten unten eine Anregung holen.

Nothelfer

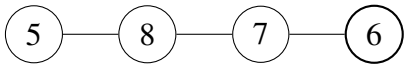
1. Mit welchem der Elemente beginnt Ihre Suche immer? Dieses Element sollte in Ihrer Darstellung oben stehen.
2. Wie viele Möglichkeiten haben Sie, um von einem Element zum nächsten zu gelangen? Zeichnen Sie die möglichen nachfolgenden Elemente unterhalb des aktuellen Elements und beachten Sie dabei die »Richtung«, in die Sich sich auf sie zu bewegst.

2.2 Der binäre Suchbaum

Die untenstehende Darstellung zeigt Ihnen einen sogenannten **binären Suchbaum**.



Der binäre Suchbaum ist aus den einzelnen Suchanfragen entstanden und zeichnet den Weg nach, den wir bei der Suche nach einem bestimmten Element zurückgelegt haben. Haben wir beispielsweise nach dem Element (6) gesucht, so wurde dabei der folgende Weg zurückgelegt:



Theorie

Bäume sind eine der wichtigsten dynamischen Datenstrukturen in der Informatik. Bevor wir auf weiterführende Eigenschaften der Bäume zu sprechen kommen, folgen zuerst einige grundlegende Definitionen:

- Die Elemente, aus denen ein Baum besteht, werden **Knoten** genannt. Der oberste Knoten in einem Baum heißt **Wurzel**[a].
- Die Verbindungslinien zwischen zwei Knoten heißen **Kanten**.
- Da es sich bei unserem Suchbaum um einen *binären* Baum[b] handelt, kann ein Knoten **maximal zwei** direkte Nachfolger haben. Knoten, die keinen Nachfolger haben, nennt man **Blatt**.
- Die **Höhe** eines Baums ist definiert als der längste Weg von der Wurzel zu einem Blatt. Wir bewegen uns dabei von oben nach unten entlang der Kanten. Im obigen Beispiel hat der Baum eine Höhe von $h = 4$.

Binäre Suchbäume dienen u. a. dazu, Daten (vor allem Daten, die sich häufig verändern) aufzunehmen und zu speichern. Durch ihre Struktur (auf die wir später noch genauer zu sprechen kommen werden), lassen sich Bäume effizient nach bestimmten Daten durchsuchen.

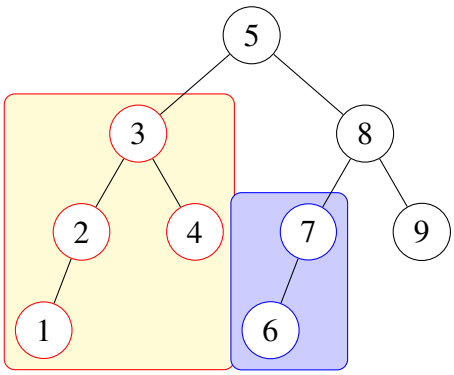
[a] Dies entspricht nicht unbedingt der Intuition, da die Wurzel in unseren Bäumen immer oben ist. Man kann sich den Begriff leichter merken, wenn man sich klarmacht, dass alle anderen Knoten an der Wurzel »hängen«.

[b] Erinnern Sie sich daran, woher der Begriff »Binäre Suche« stammt.

Es gibt viele verschiedene Arten von Bäumen in der Informatik, wir beschäftigen uns im Rahmen dieses Leitprogramms aber überwiegend mit den binären Suchbäumen. Warum dieser Baum »binärer« Baum heißt, haben wir uns bereits verdeutlicht. Im Folgenden wollen wir einige Eigenschaften untersuchen, die binäre Suchbäume für uns interessant machen.

2.3 Eigenschaften des binären Suchbaums

Bevor wir uns mit der elementarsten Eigenschaft des binären Suchbaums befassen können, benötigen wir eine weitere Definition: Den sogenannten Teilbaum.



Ein **Teilbaum** eines binären Suchbaums sind alle Knoten, die von einem bestimmten Knoten aus erreichbar sind. Im Beispiel sind zwei solche Teilbäume eingezeichnet. Den obersten Knoten dieser Teilbäume nennen wir ebenfalls **Wurzel**, diesmal jedoch die Wurzel eines Teilbaums. Zusätzlich unterscheiden wir zwischen einem **linken** und einem **rechten** Teilbaum: Im oberen Beispiel ist der linke Teilbaum von 3 der Teilbaum mit den Knoten 1 und 2, der rechte Teilbaum ist der Knoten 4. Enthält ein Teilbaum keine Knoten, bezeichnen wir ihn als **leer**.

Es folgen einige Aufgaben, mit denen Sie Ihr Verständnis für den Begriff des Teilbaums überprüfen können.

Aufgabe 2.2

(a) Aus welchen Knoten besteht der Teilbaum mit der Wurzel 8?

(b) Welche Knoten gehören zum linken, welche zum rechten Teilbaum des blau eingezeichneten Teilbaums?

(c) Ist der Teilbaum mit einem Blatt als Wurzel ebenfalls ein Teilbaum?

Im Folgenden wollen wir eine Eigenschaft untersuchen, die besonders im Hinblick darauf wichtig ist, wenn wir unseren Suchbaum »von Grund auf aufbauen« oder Elemente in den Baum einfügen wollen, ohne seine leichte Durchsuchbarkeit direkt zu zerstören. Bearbeiten Sie hierzu folgende Aufgabe:

Aufgabe 2.3

Betrachten Sie einen beliebigen Teilbaum mit der Wurzel w. Welche elementare Eigenschaft fällt Ihnen auf, wenn Sie die Werte innerhalb der Knoten miteinander vergleichen? Begründen Sie dies!

Nothelfer

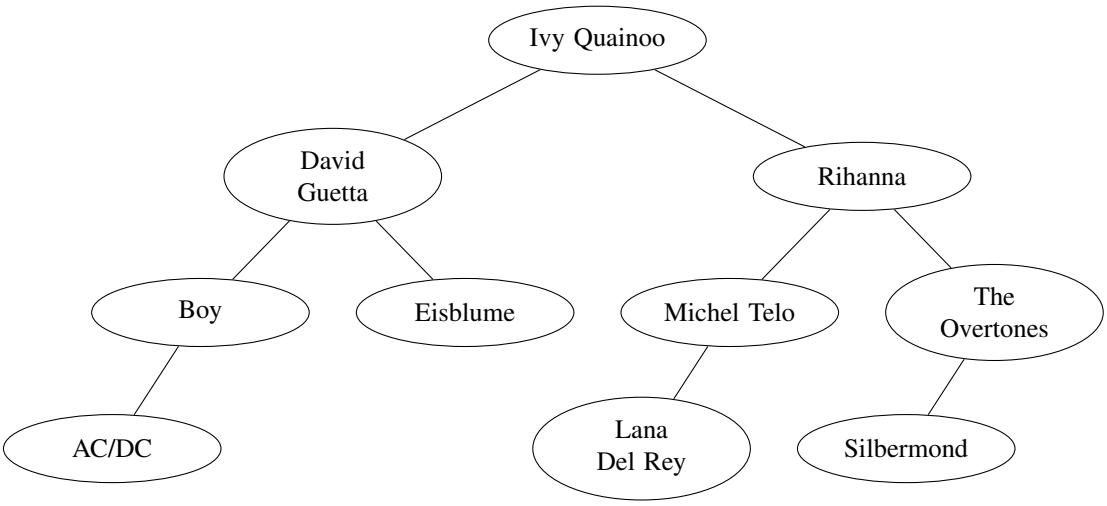
Betrachen Sie zuerst einen einzelnen Knoten und dessen linken und rechten Nachfolger. Welche Relation besteht zwischen diesem Knoten und seinen Nachfolgern? Versuchen Sie anschließend, Ihr Ergebnis auf Teilbäume auszuweiten.

2.4 Wo ist die Ordnung hin?

Nach den ziemlich theoretischen letzten Abschnitten kehren wir nun wieder etwas mehr zur konkreten Anwendung zurück.

Wir erinnern uns: Unser erster binärer Suchbaum ist als Produkt aus der binären Suche entstanden, indem wir die Reihenfolge der besuchten Elemente in eine graphische Darstellung gebracht haben. Das würde allerdings bedeuten, dass wir, um einen Baum aufzubauen, immer zuerst eine sortierte Menge von Elementen benötigen, auf die wir dann einzeln die binäre Suche anwenden, um unseren Baum Knoten um Knoten aufzubauen. Das klingt insgesamt nicht sehr effizient und nutzt auch nicht die Vorteile aus, die der Baum als Datenstruktur hat. Unser Ziel sollte es also sein, einen Baum durch die Vorgabe einer Menge von Elementen so aufbauen zu können, dass die Eigenschaft aus Abschnitt 2.3 erfüllt bleibt. Und natürlich sollte unser Baum ebenfalls die Möglichkeit haben, die in ihm abgelegten Elemente auch wieder in der richtigen Reihenfolge ausgeben zu können – denn wenn man einen Baum an sich betrachtet, ist es nicht unbedingt direkt möglich, zu sagen, in welche Reihenfolge die Elemente ausgegeben werden müssen.

Den Aspekt des Aufbaus verschieben wir nun vorerst auf das nächste Kapitel und beschäftigen uns hier mit dem sogenannten »Baumdurchlauf« und welche Möglichkeiten es hierfür gibt. Hierzu betrachten wir folgenden Baum mit einigen der Interpreten aus Ihrer CD-Sammlung:



Wie Sie sich sicher schnell vergewissern können, ist die Eigenschaft aus Abschnitt 2.3 auch für diesen Baum erfüllt. Wie wollen uns nun einen Algorithmus überlegen, nach dem man vorgehen kann, um die Elemente in sortierter Reihenfolge aus dem Baum abzulesen.

Aufgabe 2.4

Entwickeln Sie einen Algorithmus in Umgangssprache, der den Baum in der richtigen Reihenfolge wieder ausgibt.

Im unteren Kasten sind einige Fragen aufgelistet, die Ihnen Anhaltspunkte liefern können, worauf Sie bei der Formulierung achten müssen.

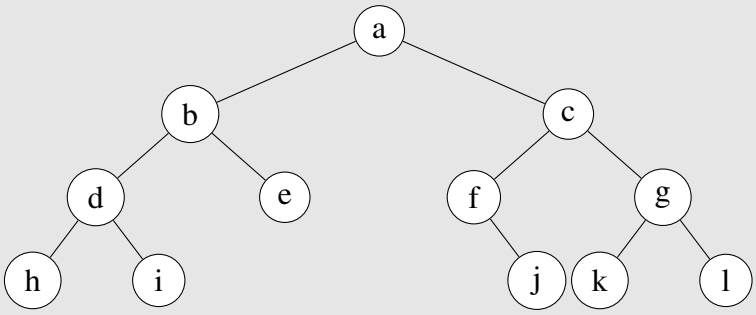
Nothelfer

- Welches Element ist das erste in der Liste? Welchen Weg müssen Sie zurücklegen, um dieses Element zu erreichen? Danken Sie dabei immer an die in 2.3 festgestellte Eigenschaft!
- Betrachten Sie den Teilbaum mit »David Guetta« als Wurzel. Bewegen Sie sich nun entlang der Kanten zum ersten Element, danach zum zweiten usw.. Was sagt dies über die Reihenfolge aus, die Sie bei der Ausgabe beachten mussten?
- Versuchen Sie bei Ihrem Algorithmus die Teilbäume eines Knotens in Betracht zu ziehen. Welchen der Teilbäume müssen Sie zuerst bearbeiten?

Diesen Algorithmus nennt man *Inorder-Durchlauf*, da er die Knoten in der richtigen Reihenfolge (also »in order« ausgibt).

Lernkontrolle

Gegeben ist folgender binärer Baum:



Die Buchstaben sind hier Stellvertreter für verschiedene vergleichbare Elemente, die auch nach der Eigenschaft aus Abschnitt 2.3 angeordnet sind.
Bearbeiten Sie die nachfolgenden Aufgaben:

Aufgabe 1: Die Blätter des Baums sind h, i, j, k und l.

- ☐ Falsch
- ☐ Richtig

Aufgabe 2: In einem binären Suchbaum findet man das größte Elemente, indem man so lange den Kanten nach rechts folgt, bis man ein Blatt erreicht hat.

- ☐ Falsch
- ☐ Richtig

Aufgabe 3: Die Nachfolger von b sind:

- ☐ d
- ☐ e
- ☐ h
- ☐ i

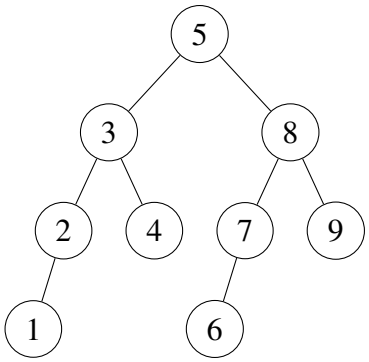
Aufgabe 4: Geben Sie die Reihenfolge an, in der die Knoten beim »Inorder-Durchlauf« besucht werden. Denken Sie daran, dass die eingetragenen Buchstaben nur Platzhalter sind!

Lösungen zu Kapitel 2:

Aufgabe 2.1:

(a) Zur 2: 5, 3, 2
Zur 9: 5, 8, 9

(b) Eine mögliche Darstellung für alle Suchdurchläufe:



Die Idee dahinter ist, dass man – ausgehend von der betrachteten Zahl – eine Linie nach rechts oder links zeichnet, die »getroffene« Zahl dort notiert und diesen Vorgang wiederholt, bis die Zahl gefunden wurde.

Aufgabe 2.2:

(a) 8 – 7 – 6 – 9

(b) Der linke Teilbaum besteht aus dem Knoten mit der Zahl 6, der rechte Teilbaum ist leer.

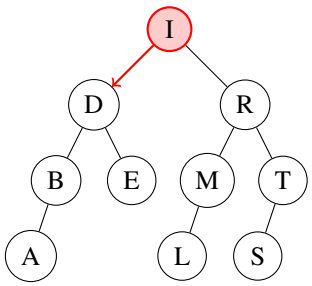
(c) Ja, denn es kommt nicht darauf an, ob ein Knoten tatsächlich Nachfolger hat oder nicht.

Aufgabe 2.3: Bei einem gegebenen Teilbaum mit Wurzel w sind alle Knoten in der linken Seite des Teilbaums kleiner als die Wurzel selbst. Die Knoten auf der rechten Seite dagegen sind alle größer als das Element in der Wurzel.

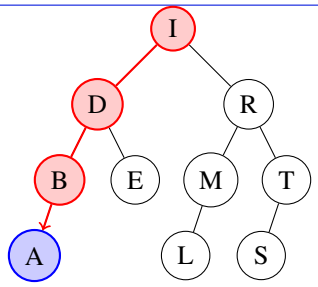
Verdeutlicht man sich, wie der Baum aufgebaut wurde, so wird diese Eigenschaft unmittelbar ersichtlich: Das mittlere Element (hier die 5) hat die Menge der Zahlen genau in zwei Hälften geteilt: eine mit kleineren und eine mit größeren Zahlen darin. Somit wird diese Zahl von uns als Wurzel festgelegt. Dieser Schritt wird nun für jede der Hälften wiederholt und jede der Zahlen wird damit zur Wurzel eines eigenen Teilbaums.

Aufgabe 2.4: Wir verdeutlichen uns den Ablauf zuerst anhand des Beispiels. Der Einfachheit halber (und da alle Anfangsbuchstaben verschieden sind), sind im Baum nur die Anfangsbuchstaben der jeweiligen Interpreten eingetragen.

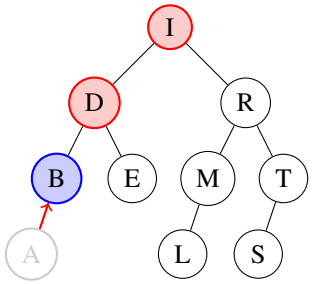
Wir beginnen an der Wurzel. Da uns bekannt ist, dass alle Elemente links der Wurzel definitiv vor der Wurzel ausgegeben werden müssen, steigen wir in den linken Teilbaum ein.



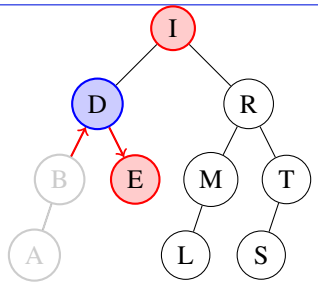
Wir betrachten erneut den linken Teilbaum, diesmal mit Knoten D als Wurzel (Begründung: s.o.) und setzen dieses Verfahren so lange fort, bis wir beim kleinsten Element angelangt sind. Dieses Element geben wir anschließend aus.



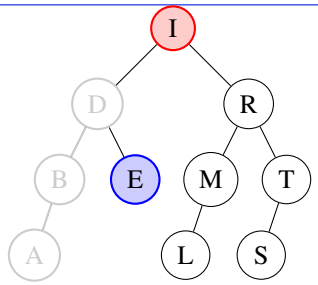
Nun steigen wir eine Ebene im Baum auf. Wir wissen, dass die Elemente im rechten Teilbaum von B (falls er existiert) größer sind als B und demnach erst danach ausgegeben werden müssen. Alle kleineren Elemente sind bereits abgearbeitet, also können wir an dieser Stelle das B ausgeben.



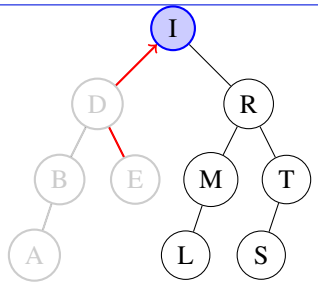
Wir steigen wiederum eine Ebene auf, da B's rechter Teilbaum leer ist, und geben das D aus. Danach steigen wir in den rechten Teilbaum von D ein, da sich dort alle Elemente befinden, die größer sind als D, aber kleiner als I.



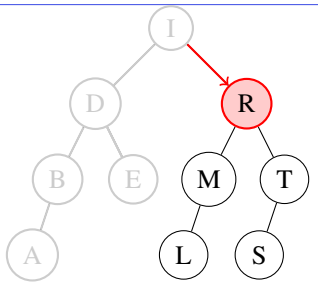
Anschließend geben wir das E aus, da es sich um ein Blatt handelt und sowohl rechter als auch linker Teilbaum leer sind.



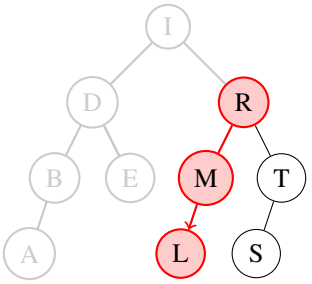
Da nun alle Knoten des linken Teilbaums der Wurzel I ausgegeben wurden (und somit alle Elemente, die kleiner sind als I), steigen wir auf bis zur Wurzel und geben diese aus.



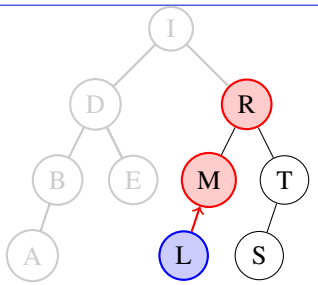
Anschließend müssen die Knoten des rechten Teilbaums ausgegeben werden. Wir steigen in den rechten Teilbaum ein, hin zum R.



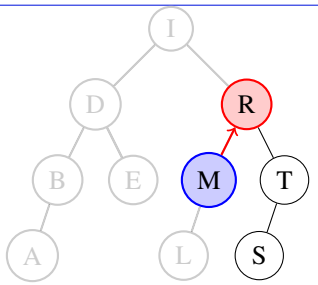
An dieser Stelle verdeutlichen wir uns nochmals die Eigenschaft aus 2.3 - alle Knoten im rechten Teilbaum von I sind größer als I. Für R gilt wiederum: Alle Knoten im linken Teilbaum von R sind kleiner als R, alle im rechten Teilbaum größer. Also steigen wir als nächstes in den linken Teilbaum von R hinab. Dort verfahren wir nach dem gleichen Schema.



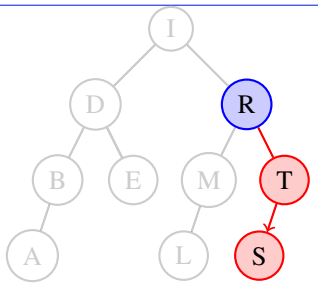
Da wir nun bei einem Blatt angekommen sind, geben wir L aus und steigen eine Ebene im Baum auf zum M.



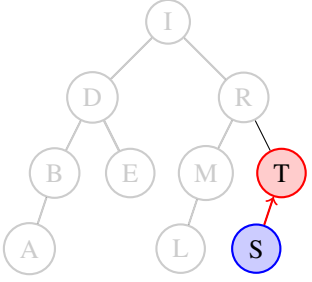
M's rechter Teilbaum ist leer, demnach geben wir M aus und steigen auf zum R.

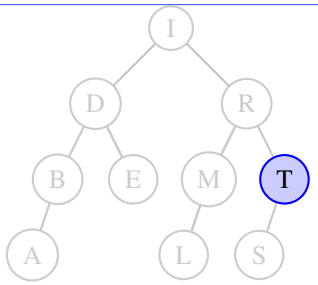


Da alle Elemente im linken Teilbaum von R ausgegeben wurden, geben wir R aus und steigen danach in den rechten Teilbaum von R ein.



Wir kommen beim Blatt S an, geben es aus und steigen zum letzten verbleibenden Knoten T auf.



Anschließen geben wir T aus. Da T's rechter Teilbaum leer ist, beenden wir die Ausgabe an dieser Stelle.	

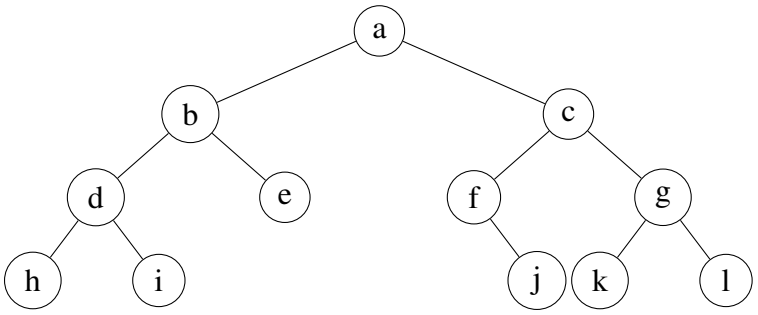
Die endgültige Ausgabe lautet demnach:

A, B, D, E, I, L, M, R, S, T.

Wir können den Algorithmus nun in eine umgangssprachliche Form bringen:

- Steige an der Wurzel in den Baum ein
- Wiederhole für jeden Knoten – von der Wurzel aus – folgende Schritte:
  - Steige in den linken Teilbaum ein und beginne für jeden Knoten diese Abfolge von Schritten. Ist der linke Teilbaum leer, gehe zum nächsten Schritt.
  - Gib den Knoten aus.
  - Steige in den rechten Teilbaum ein und beginne für jeden Knoten diese Abfolge von Schritten.

Lernkontrolle



Die Buchstaben sind hier Stellvertreter für verschiedene vergleichbare Elemente, die auch nach der Eigenschaft aus Abschnitt 2.3 angeordnet sind.

Bearbeiten Sie die nachfolgenden Aufgaben:

Aufgabe 1: Die Blätter des Baums sind h, i, j, k und l.

☒ Falsch – e fehlt.
☐ Richtig

Aufgabe 2: In einem binären Suchbaum findet man das größte Elemente, in dem man so lange den Kanten nach rechts folgt, bis man ein Blatt erreicht hat.

- ☐ Falsch
- ☒ Richtig

Aufgabe 3: Die Nachfolger von b sind:

- ☒ d
- ☒ e
- ☐ h
- ☐ i

Aufgabe 4:

h, d, i, b, e, a, f, j, c, k, g, l.

Kapitel 3

Die Modellierung des binären Suchbaums

Übersicht

Im vorherigen Kapitel haben wir den binären Suchbaum als Produkt der binären Suche für eine Menge von sortierbaren Elementen kennengelernt. Unser Ziel ist es jedoch, den binären Suchbaum als eigenständige und unabhängige Datenstruktur verwenden zu können und die Vorteile, die uns die binäre Suche bietet, in diese Datenstruktur einzuarbeiten.
In diesem Kapitel werden die Grundlagen für eine mögliche spätere Implementierung des Baums gelegt, indem wir den Baum mit Hilfe von Objekten modellieren. Anschließend verdeutlichen wir uns die Abläufe, die beim Einfügen von neuen Knoten in den Baum vorgehen, indem wir unser objektorientiertes Modell zugrunde legen.

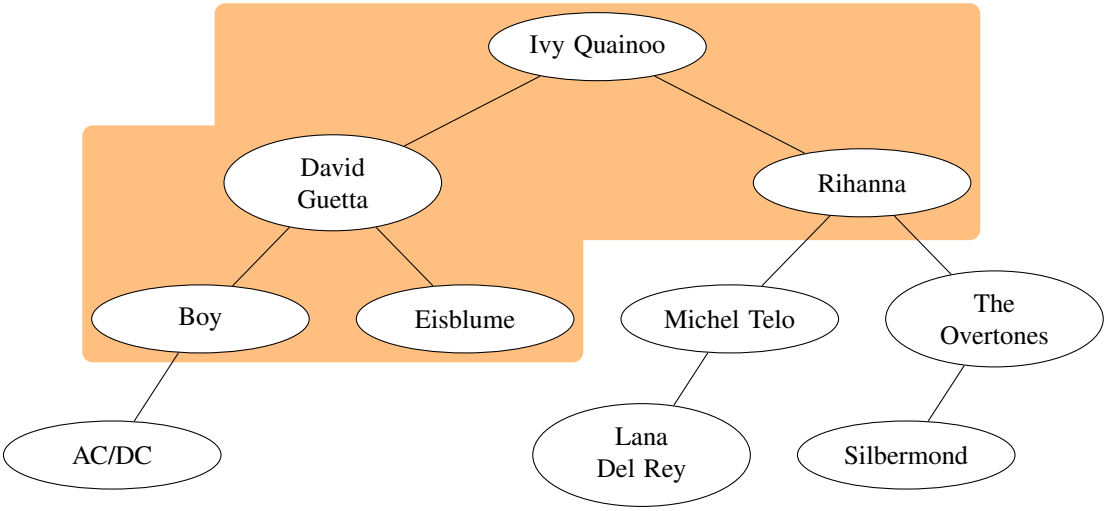
Lernziele

Nach der Bearbeitung dieses Kapitels,

- kennen Sie ein Modell für den binären Suchbaum.
- kennen Sie den Algorithmus zum Einfügen neuer Knoten.

3.1 Modellierung

Um uns der Modellierung eines binären Suchbaums stückweise zu nähern, beginnen wir wiederum mit einem Beispiel eines bereits bestehenden Baums:



Um uns ein wenig Arbeit vorweg abzunehmen, beschäftigen wir uns bei der objektorientierten Modellierung vorerst nur mit den markierten fünf Elementen.

Aufgabe 3.1

(a) Geben Sie die Objektkarten für die markierten fünf Knoten des Baums an. Beachten Sie bei der Modellierung, dass die Knoten im restlichen Baum trotzdem als existent angesehen werden sollen. Denken Sie an Methoden, mit denen die Attribute ausgelesen werden können.

(b) Kennzeichnen Sie die Beziehungen zwischen den Objekten. Falls Beziehungen zu Objekten aufgebaut werden sollen, die außerhalb des markierten Bereichs liegen, so genügt es, wenn Sie einen stellvertretenden Text an die passende Stelle schreiben.

Nachdem Sie sich klar gemacht haben, welche Attribute und Methoden die einzelnen Objekte benötigen, sollte es Ihnen möglich sein, daraus eine Klasse für die Knoten abzuleiten.

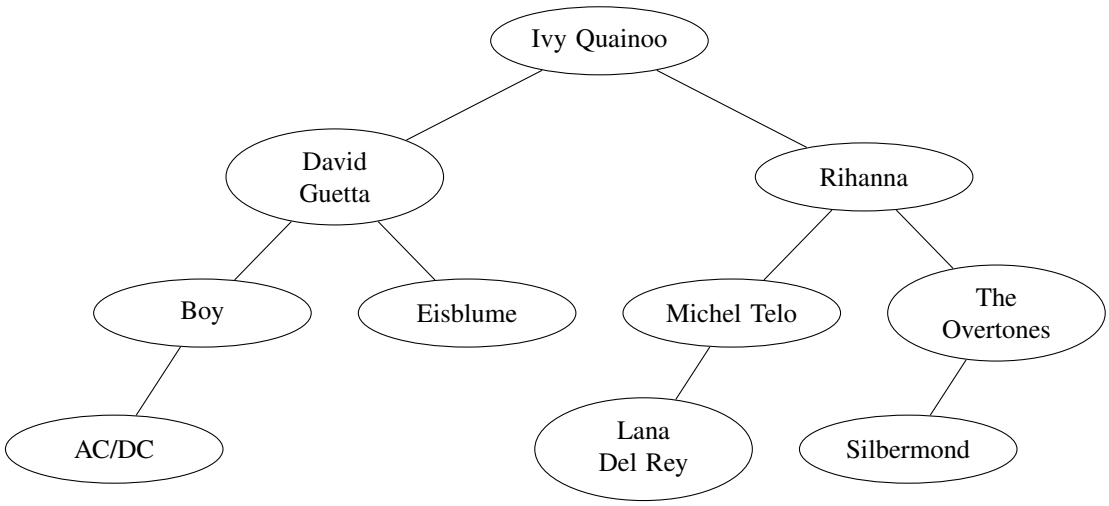
Aufgabe 3.2

(a) Entwerfen Sie eine Klasse für die Knoten des binären Suchbaums. Geben Sie lediglich an, über welche Attribute und Methoden die Klasse verfügen soll – es geht hierbei nicht um eine konkrete Implementierung!

(b) Geben Sie nun den Aufbau für eine Klasse BinaererSuchbaum an. Welche Methoden sollten für einen binären Suchbaum zur Verfügung stehen, damit er sinnvoll als Datenstruktur genutzt werden kann?

3.2 Neue Elemente braucht der Baum

Kommen wir nun zurück zu unserer CD-Sammlung:



Da wir hier nur einen kleinen Ausschnitt unserer Sammlung betrachten, möchten wir nachträglich weitere Elemente zu dem gegebenen Baum hinzufügen können. Weil Sie sich gerade das neue Album von »Die Ärzte« gekauft haben, soll uns dieses hier als Beispiel dienen.

Wenn wir neue Elemente zu unserem Baum hinzufügen, ist es natürlich wichtig, dass die grundlegende Eigenschaft aus Abschnitt 2.3 dabei erhalten bleibt. Bevor wir – Schritt für Schritt – einen Algorithmus zum Einfügen neuer Elemente entwickeln, überlegen wir uns, an welcher Stelle das Element eingefügt werden muss.

Aufgabe 3.3

Wir gehen davon aus, dass die bestehende Struktur des Baums so erhalten bleiben soll, wie sie ist. Überlegen Sie sich, wie das Einfügen eines neuen Knotens in einen binären Suchbaum abläuft. Greifen Sie im Zweifelsfalle auf die Fragen im Nothelfer zurück!

Nothelfer

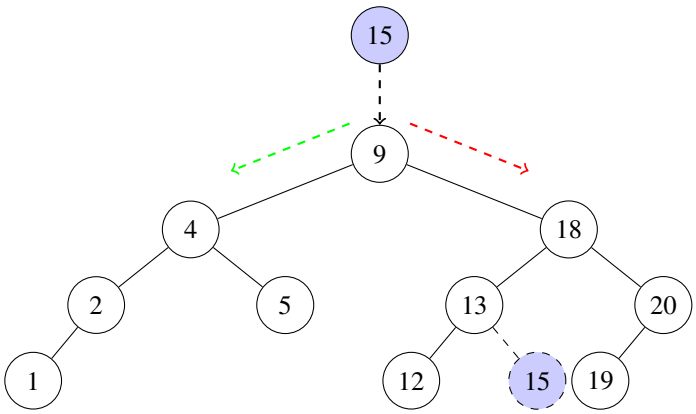
(a) Wie viele Knoten kommen überhaupt als Vorgänger unseres Knotens »Die Ärzte« in Betracht, wenn wir dabei die Eigenschaft aus 2.3 außer Acht lassen?

(b) Ordnen Sie den Knoten nun an der richtigen Stelle ein. (**Tipp**: Überlegen Sie sich beispielsweise, ob der Knoten »Die Ärzte« im rechten oder linken Teilbaum der Wurzel untergebracht werden muss. Gehen Sie so schrittweise jeden weiteren Knoten ab, bis Sie an die Stelle gelangen, die die Eigenschaft aus 2.3 erfüllt.)

(c) Gehen Sie nun erneut auf diese Weise vor und fügen Sie einen Knoten für »Nickelback« an der richtigen Stelle ein.

(d) Durchlaufen Sie die Knoten nach der »inorder«-Methode, um sicherzustellen, dass die richtige Reihenfolge bewahrt worden ist.

Um hieraus nun einen allgemeingültigen Algorithmus zu entwickeln, gehen wir vorerst davon aus, dass der Baum nicht leer ist, so dass wir diesen Sonderfall ausschließen können. Betrachten Sie dazu folgenden Baum:



Aufgabe 3.4

In obigen Baum soll die Zahl 15 eingefügt werden. Die dafür vorgesehene Stelle ist bereits markiert. Da auf die Knoten im Baum (wie bei der linearen Liste) nur über die Wurzel zugegriffen werden kann, sollte das Einfügen eines neues Elementes dort beginnen. Stellen Sie sich vor, der Knoten 15 wird »von oben« auf die Wurzel geworfen und rollt dann – je nach Knoten, die er währenddessen passiert – nach links oder rechts.

Zeichnen Sie anschließend ein Struktogramm (oder einen Programmablaufplan) für das Einfügen eines neuen Knotens. Es geht dabei nicht darum, dass Sie sich den konkreten Code überlegen.

(Falls möglich: Nutzen Sie zur Erzeugung des Diagramms eine Werkzeug wie dia, LibreOffice Draw o. ä.)

3.3 Neue Elemente und die Folgen

Nachdem wir nun in der Lage sind, neue Knoten zu einem binären Suchbaum hinzuzufügen, sollten wir uns Gedanken darum machen, welche Auswirkungen das Einfügen neuer Knoten auf einen binären Suchbaum haben kann.

Aufgabe 3.5

Experimentieren Sie mit dem Java-Applet unter `http://www.cs.jhu.edu/~goodrich/dsa/trees/btree.html`. Erzeugen Sie Bäume mit verschiedenem Aussehen und überlegen sich, welche Auswirkungen das Aussehen der Bäume auf die Sucheigenschaft des Baums haben kann.

Wie Sie festgestellt haben, lassen sich relativ leicht Bäume erzeugen, die nicht mehr viel von den Vorteilen bieten, die unserer eigentlichen Idee des Binären Suchbaums zugrunde liegen. Im letzten Kapitel werden wir uns hierzu einige Möglichkeiten anschauen, wie dieses Problem angegangen werden kann.

Lernkontrolle

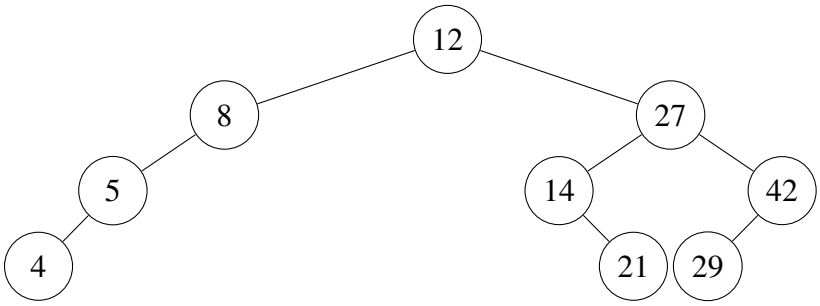
Aufgabe 1: Das Einfügen eines neuen Knotens kann die Eigenschaft des binären Suchbaums zerstören, dass alle Werte der Knoten im linken Teilbaum kleiner sind als der Wert der Wurzel.

- ☐ Richtig
- ☐ Falsch

Aufgabe 2: Ein binärer Suchbaum kann zu einer linearen Liste werden, je nachdem, in welcher Reihenfolge die Knoten eingefügt werden.

- ☐ Richtig
- ☐ Falsch

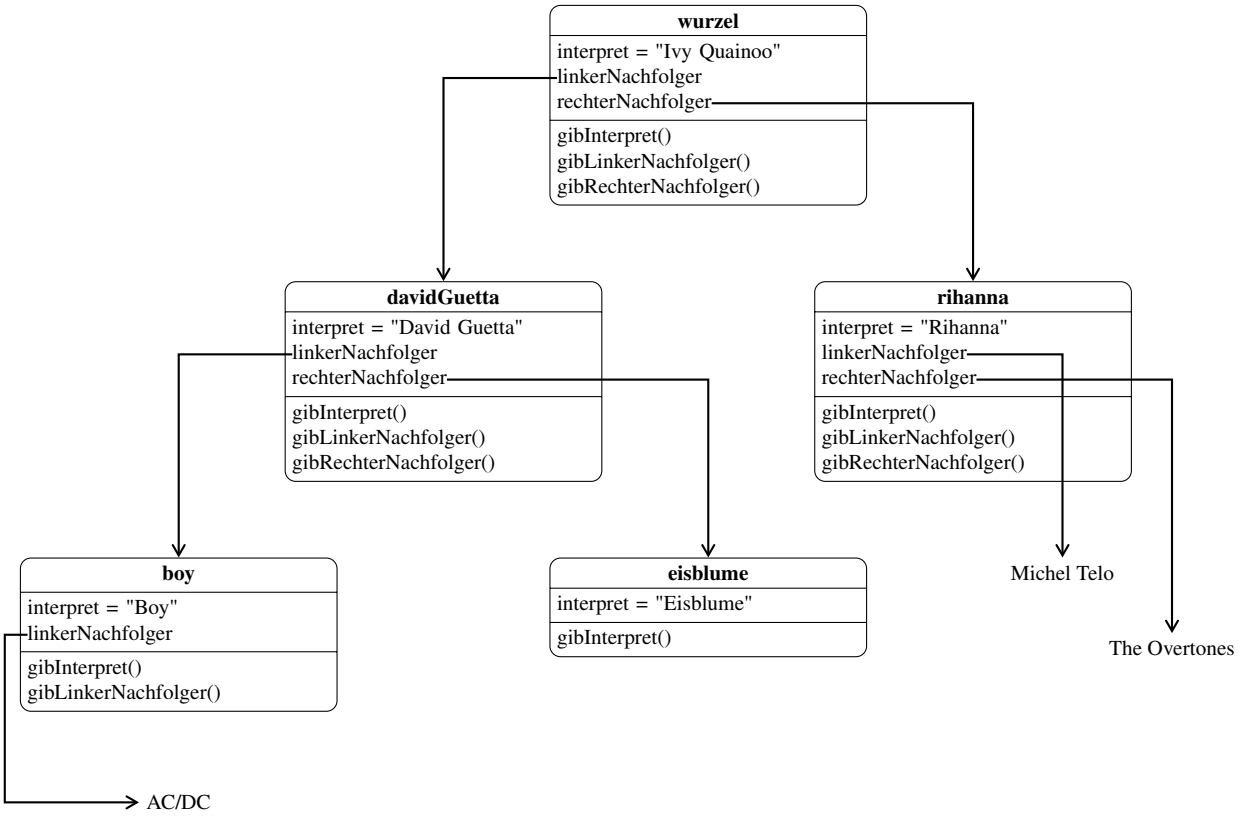
Aufgabe 3: Gegeben ist der folgende binäre Suchbaum:



Fügen Sie diesem Baum folgende Zahlen hinzu: 10; 6; 38; 49

Lösungen zu Kapitel 3

Aufgabe 3.1: Wir beschränken uns bei der Modellierung an dieser Stelle auf die Interpreten der einzelnen Alben. Natürlich bräuchten wir für eine vollständige Implementierung noch weitere Attribute wie z. B. den Albumtitel.



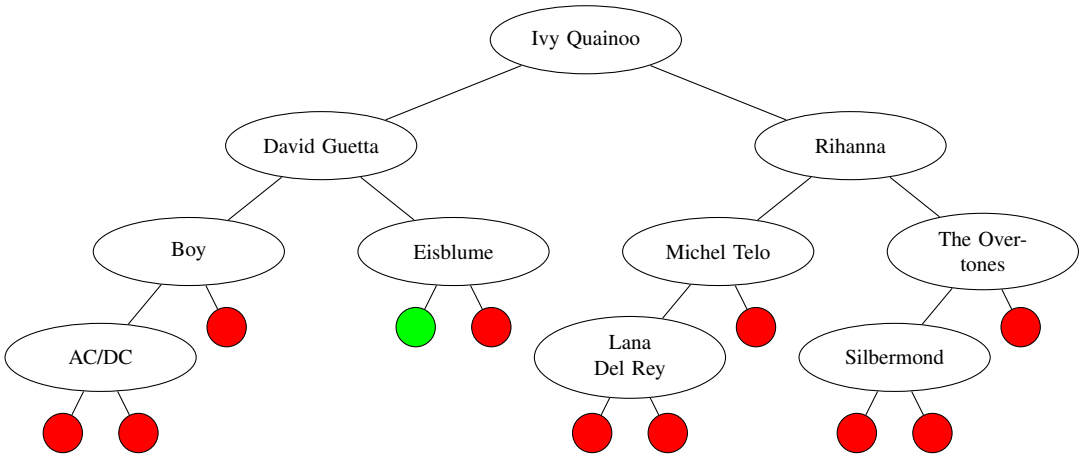
Aufgabe 3.2:

Knoten
interpret: Zeichenkette linkerNachfolger: Knoten rechterNachfolger: Knoten
Knoten(Zeichenkette interpret) gibInterpret(): Zeichenkette gibLinkenNachfolger(): Knoten gibRechtenNachfolger(): Knoten setzeLinkenNachfolger(Knoten k) setzeRechtenNachfolger(Knoten k)

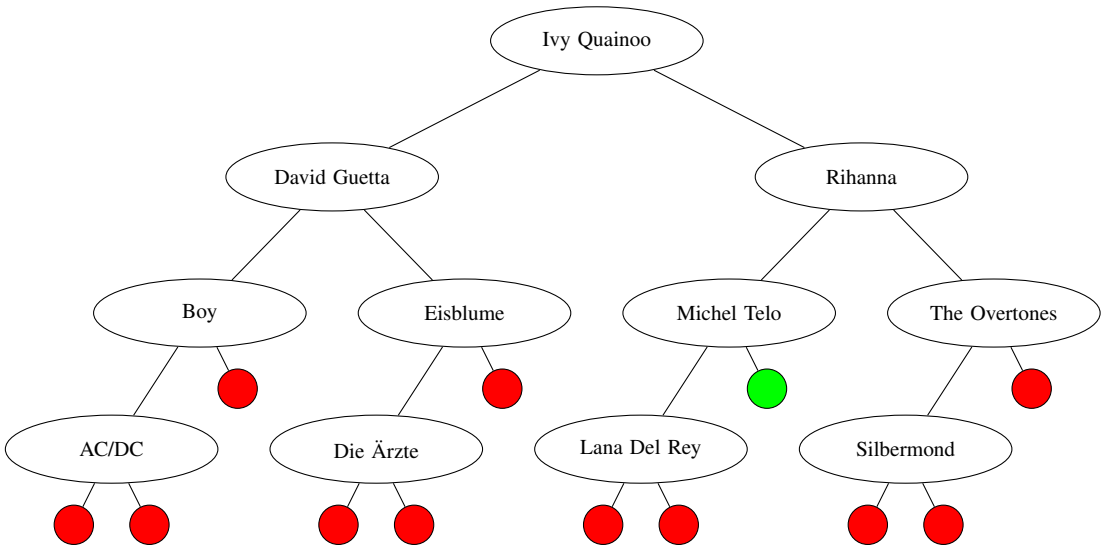
BinaererSuchbaum
wurzel: Knoten
fuegeKnotenEin(Knoten k) inOrderDurchlauf() gibWurzel()

Aufgabe 3.3: Die Antworten beziehen sich auf die Fragen des Nothelfers, damit Sie die Lösung Stück für Stück nachvollziehen können.

(a) + (b) Es gibt insgesamt 11 Möglichkeiten, den neuen Knoten einzufügen, ohne dabei die Eigenschaft aus 2.3 zu berücksichtigen. Der einzige passende Knoten ist jedoch der grün markierte.



(c) Die einzige richtige Möglichkeit ist als rechter Nachfolger von »Michel Telo«:



Aufgabe 3.4: Es ist möglich, die Methode rekursiv oder iterativ anzugehen.

Zuerst wird die **rekursive Variante** vorgestellt:

einfuegen(baum, eintrag)

<table>
<tr><td colspan="4">eintrag kleiner als Eintrag in baum
ja / nein</td></tr>
<tr><td colspan="2">hat baum einen linken Teilbaum
ja / nein</td><td colspan="2">hat baum einen rechten Teilbaum
ja / nein</td></tr>
<tr><td>einfuegen(linker Teilbaum von baum, eintrag)</td><td rowspan="2">fuege einen neuen Knoten mit eintrag als linker Nachfolger von baum ein</td><td>einfuegen(rechter Teilbaum von baum, eintrag)</td><td rowspan="2">fuege einen neuen Knoten mit eintrag als rechten Nachfolger von baum ein</td></tr>
<tr><td>∅</td><td>∅</td></tr>
</table>

Die **iterative Variante**:

einfuegen(baum, eintrag)

<table>
<tr><td colspan="4">setze tmp auf baum</td></tr>
<tr><td colspan="4">setze gefunden auf falsch</td></tr>
<tr><td colspan="4">solange nicht gefunden</td></tr>
<tr><td colspan="4">eintrag kleiner als Eintrag von tmp
ja / nein</td></tr>
<tr><td colspan="2">hat tmp einen linken Teilbaum
ja / nein</td><td colspan="2">hat tmp einen rechten Teilbaum
ja / nein</td></tr>
<tr><td>setze tmp auf linken Teilbaum von tmp</td><td rowspan="2">fuege einen neuen Knoten mit eintrag als linker Nachfolger von tmp ein</td><td>setze tmp auf rechten Teilbaum von tmp</td><td rowspan="2">fuege einen neuen Knoten mit eintrag als rechten Nachfolger von tmp ein</td></tr>
<tr><td>∅</td><td>∅</td></tr>
<tr><td colspan="2">setze gefunden auf wahr</td><td colspan="2">setze gefunden auf wahr</td></tr>
</table>

Aufgabe 3.5: Das Einfügen bereits vorsortierter Knoten sorgt dafür, dass aus einem binären Suchbaum schnell annähernd eine lineare Liste wird. Dadurch verliert der Baum seinen Geschwindigkeitsvorteil beim Durchsuchen.

Lernkontrolle

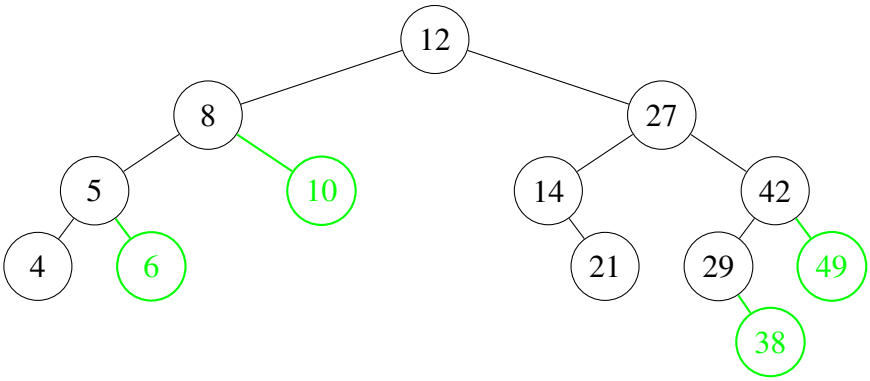
Aufgabe 1: Das Einfügen eines neuen Knotens kann die Eigenschaft des binären Suchbaums zerstören, dass alle Werte der Knoten im linken Teilbaum kleiner sind als der Wert der Wurzel.

- ☐ Richtig
- ☒ Falsch

Aufgabe 2: Ein binärer Suchbaum kann zu einer linearen Liste werden, je nachdem, in welcher Reihenfolge die Knoten eingefügt werden.

- ☒ Richtig
- ☐ Falsch

Aufgabe 3:



Kapitel 4

Balance ist alles

Übersicht

Am Ende des letzten Kapitels haben Sie gesehen, dass durch das Einfügen in einen binären Suchbaum auch Bäume entstehen können, die nicht mehr bei jedem Schritt die Anzahl der Elemente halbieren und somit nicht mehr sonderlich optimal sind. In diesem Kapitel werden Sie eine Vorgehensweise kennenlernen, die dieses Problem adressiert.

Lernziele

Nach der Bearbeitung dieses Kapitels,

- kennen Sie die Nachteile, die mit dem Einfügen in binäre Suchbäume verbunden sind.
- können Sie ein Verfahren anwenden, um einen binären Suchbaum wieder in eine »ideale Form« zu bringen.

4.1 Der Baum als Ideallösung?

Nachdem Sie sich in den letzten Kapiteln näher mit dem binären Suchbaum vertraut gemacht hast, werden Sie sicherlich auch schon einen Nachteil der Baumstruktur entdeckt haben: Durch Einfüge- und auch durch Lösch-Operationen kann es schnell passieren, dass der Baum zu einer Art linearen Liste »entartet«. Fügt man die Knoten beispielsweise in aufsteigender Reihenfolge hinzu, so entsteht ein Baum, in dem jeder Knoten nur Nachfolger auf der rechten Seite hat. Beim Durchsuchen dieser Liste würde dann der gleiche Aufwand entstehen, wie beim Durchsuchen einer linearen Liste – und alle Vorteile des binären Suchbaums wären dahin. Daher werden Lösungen für dieses Problem benötigt.

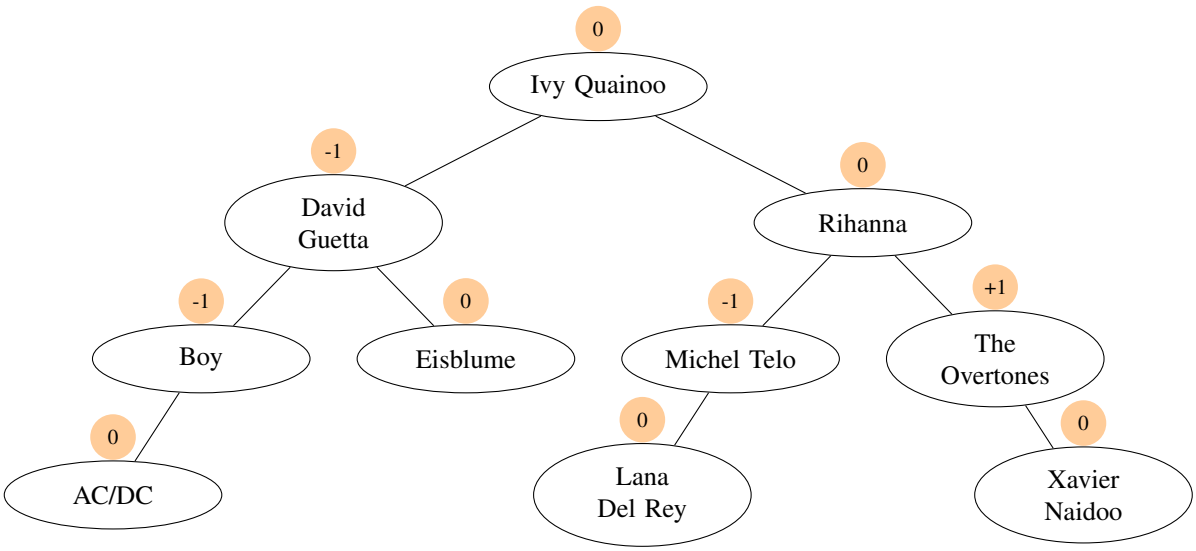
Aufgabe 4.1

(a) Formulieren Sie mit eigenen Worten: Wann bringt ein binärer Suchbaum die Vorteile mit sich, für die er von uns modelliert worden ist? Welche Eigenschaften muss der Baum erfüllen?

(b) Denken Sie an Kapitel 1 und 2 zurück – und daran, aus welchen Überlegungen der Suchbaum entstanden ist. Welche – zugegebenermaßen sehr aufwändige – Möglichkeit besteht, um einen Baum wieder in »Idealform« zu bringen?

4.2 Der AVL-Baum

Um das Problem des »unbalancierten« Baums zu lösen, benötigen wir zuerst eine Definition, was für uns ein »balancierter« Baum ist. Dazu betrachten wir den folgenden binären Suchbaum:



An jeden der Knoten k ist eine Zahl geschrieben worden, die sich folgendermaßen berechnet:

Höhe des rechten Teilbaums von k – Höhe des linken Teilbaums von k

Aufgabe 4.2

Beantworten Sie die folgenden Fragen:

(a) Blätter sind immer mit der Zahl 0 beschriftet. Warum ist das so?

(b) Welche Information erhalten wir durch das Vorzeichen der Zahl über die Teilbäume des Knotens?

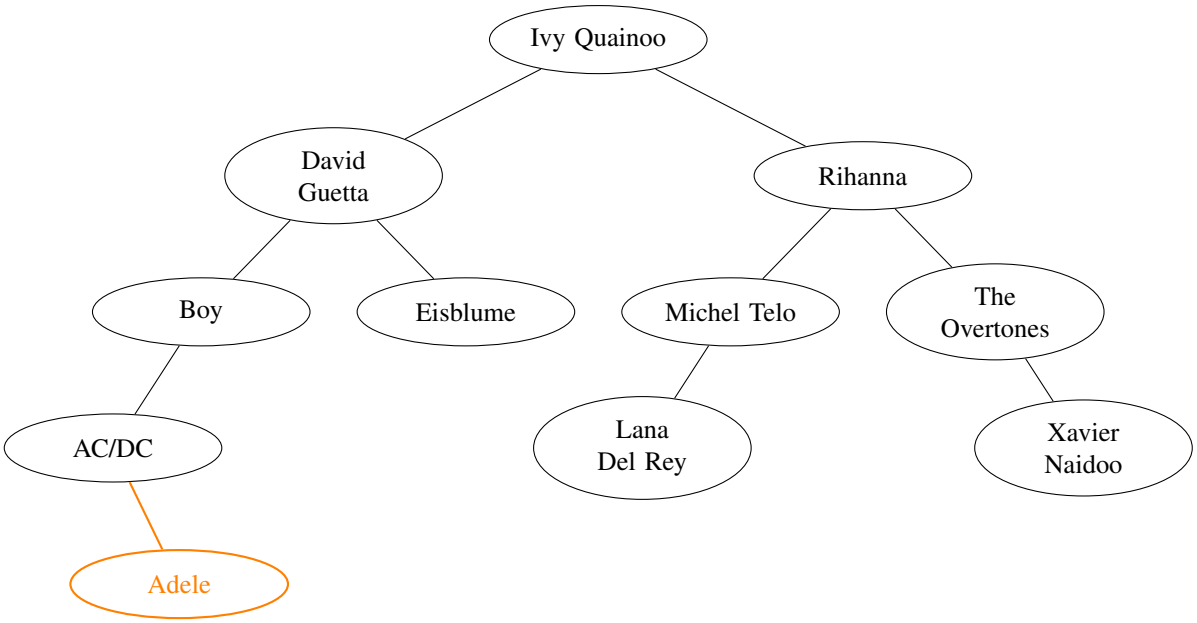
(c) Welche Eigenschaft muss für die Zahlen (und damit für die Knoten) gelten, damit die Knoten im Baum möglichst gleichmäßig verteilt sind?

Theorie

Ein binärer Baum heißt *ausgeglichener oder AVL-Baum*, wenn sich für jeden Knoten k die Höhen des linken und rechten Teilbaums höchstens um 1 unterscheiden. Der Baum im obigen Beispiel ist demnach ausgeglichen.

4.3 Rotation zum Ausgleich

Wir betrachten nun das Beispiel eines binären Suchbaums, der nicht ausgeglichen ist. Dazu fügen wir dem obigen Beispielbaum den neuen Knoten »Adele« nach den bekannten Regeln hinzu:



Aufgabe 4.3

Beschriften Sie die Knoten im obigen Baum mit den Werten der Höhendifferenz, wie sie im vorangehenden Abschnitt definiert wurde.

Betrachten wir den nun entstandenen Baum und die Werte für die Höhendifferenzen, so fällt auf, dass an zwei Stellen eine Differenz von »-2« eintritt und der Baum damit nicht mehr ausgeglichen ist. Um dieses Problem zu beheben, bedient man sich des Prinzips der »Rotation«. Man »dreht« Knoten so, dass einige Knoten auf- bzw. absteigen und erhält so schrittweise wieder einen AVL-Baum.

Aufgabe 4.4

Besuchen Sie die Internetseite:

`http://fbim.fh-regensburg.de/~saj39122/bruhi/index.html`

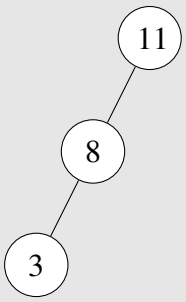
Wählen Sie links »Applet« und lassen Sie im erscheinenden Java-Applet anschließend durch die Aktivierung von »Random« 10 Zufallsknoten erzeugen. Beobachten Sie, was geschieht. Beschreiben Sie anschließend das Verfahren zum Ausgleich des Baums (grob) mit eigenen Worten!

Wie Sie in der vorherigen Aufgabe festgestellt haben, wird der Baum im Falle des Ungleichgewichts rotiert. Die Schwierigkeit liegt nun darin, zu bestimmen, in welche Richtung rotiert werden muss und ob es sich um eine einfache oder doppelte Rotation handelt. Damit kommen wir zu der folgenden Aufgabe:

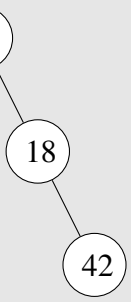
Aufgabe 4.5

Wir betrachten in dieser Aufgabe immer nur Teilbäume, da es genügt, diese beim Einfügen eines neuen Elementes auszubalancieren.

Baum 1:

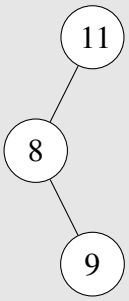


Baum 2:

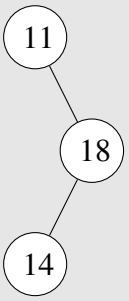


Beide gezeigten Teilbäume befinden sich im Ungleichgewicht. Wie müssen die Knoten rotiert werden, damit beide Bäume balanciert sind? Und welcher Zusammenhang besteht zu den Höhendifferenzen (achten Sie besonders auf die Vorzeichen)?

Baum 3:



Baum 4:



Bearbeiten Sie nun Baum 3 und 4 auf analoge Weise.

Beschreiben Sie abschließend, worin der Unterschied zwischen Baum 1 und 3 bzw. Baum 2 und 4 liegt. Achten Sie dabei besonders auf die Höhendifferenzen.

Nothelfer

Verwenden Sie das Applet der oben gezeigten Seite, um die Lösungen nachvollziehen zu können. Versuchen Sie anschließend, die Vorgehensweise zu begründen.

Aufgabe 4.6

Wenden Sie das Verfahren nun auf unseren Baum aus Aufgabe 4.3 an.

Lernkontrolle

Aufgabe 1: Ein binärer Suchbaum ist ausgeglichen, wenn die Höhendifferenz in allen Knoten 0 beträgt.

- ☐ Richtig
- ☐ Falsch

Aufgabe 2: Eine positive Höhendifferenz deutet darauf hin, dass der rechte Teilbaum eines Knotens größer ist als der linke.

- ☐ Richtig
- ☐ Falsch

Aufgabe 3: Bauen Sie einen neuen binären Suchbaum aus folgenden Zahlen auf:

83, 13, 25, 42, 19, 22, 49, 12

Achten Sie nach jedem Einfügen darauf, dass der Baum wieder ausgeglichen ist. Kontrollieren Sie Ihre Lösung anschließend mit dem Applet aus Aufgabe 4.4.

Lösungen zu Kapitel 4

Aufgabe 4.1:

(a) Der binäre Suchbaum muss auf beiden Seiten der Wurzel (und jedes einzelnen Knotens) möglichst die gleiche Anzahl von Knoten als Nachfolgern besitzen. Nur so kann gewährleistet werden, dass mit jedem »Abzweigen« auch etwa die Hälfte aller anderen Elemente ausgeschlossen werden.

(b) Eine Möglichkeit besteht darin, den bestehenden Baum »inorder« zu durchlaufen, so dass alle Elemente als sortierte Liste vorliegen. Anschließend führt man für jedes Element in dieser Liste die binäre Suche durch und notiert sich dabei »den Suchpfad«. So entsteht Schritt für Schritt ein Baum, bei dem gewährleistet ist, dass die Hälfte der Knoten pro Suchschritt ausgeschlossen werden kann.

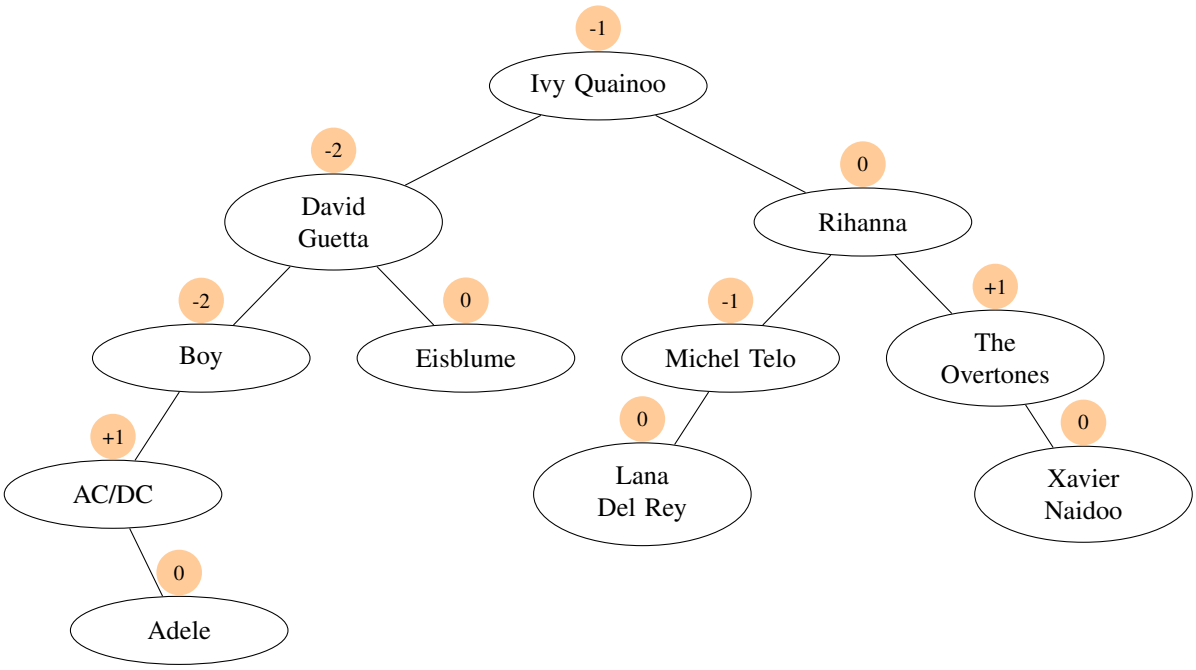
Aufgabe 4.2:

(a) Da bei Blättern beide Teilbäume leer sind (also die Höhe 0 haben), ergibt sich als Wert folgerichtig die 0.

(b) Betrachtet man die Formel, so lässt sich folgendes ablesen: Wenn der rechte Teilbaum eine größere Höhe hat als der linke, so wird die Zahl positiv. Der umgekehrte Fall gilt, falls der linke Teilbaum höher ist.

(c) Die Zahl oberhalb des Baums darf höchstens den Betrag 1 haben – auf diese Weise halten sich die Teilbäume gegenseitig relativ gut im Gleichgewicht.

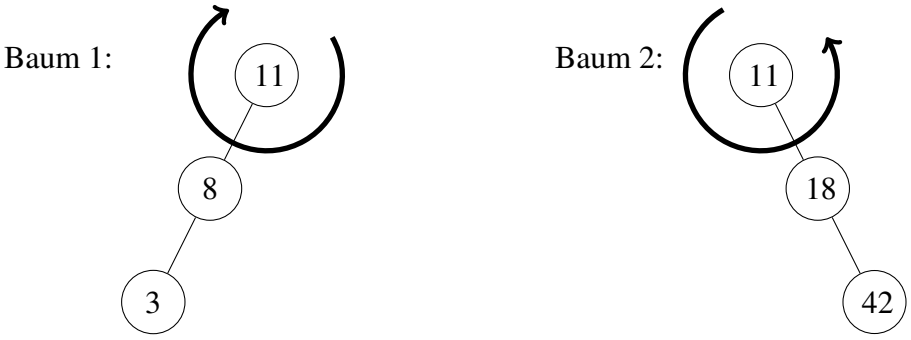
Aufgabe 4.3:



Aufgabe 4.4: Das Verfahren gliedert sich in folgende Schritte:

- Fügen Sie den neuen Knoten x an die passende Stelle im (bereits ausgeglichenen) Baum ein
- Wandern Sie von diesem Knoten den Baum nach oben entlang und aktualisieren Sie die Höhendifferenzen im »Vorbeigehen«
  - Fall 1: Eine Höhendifferenz wird auf 0 geändert. Beenden Sie die Operation, der Baum ist ausgeglichen.
  - Fall 2: Eine Höhendifferenz wird auf -2 oder +2 geändert – führen Sie eine Ausgleichsoperation (Rotation) durch

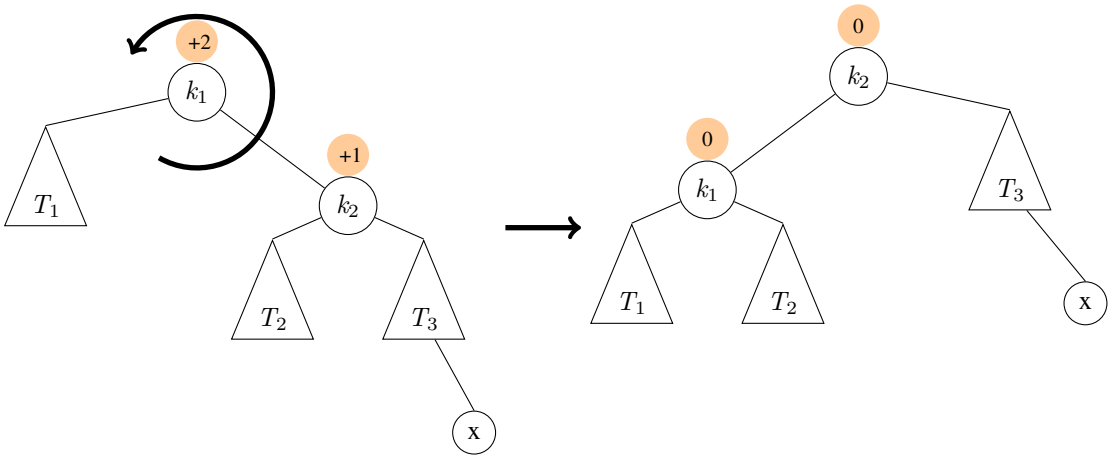
Aufgabe 4.5: Baum 1 hat ein Übergewicht auf der linken Seite (der Wurzelknoten hat eine negative Höhendifferenz). Daher drehen wir den Wurzelknoten nach rechts, um dem entgegenzuwirken. Dasselbe Verfahren gilt umgekehrt für Baum 2: eine positive Höhendifferenz von »+2« ergibt eine Drehung nach links.



Es ergeben sich daraus die folgenden neuen Bäume:

Berücksichtigt man dabei noch, dass die Knoten alle ebenfalls Wurzeln von Teilbäumen sein könnten, ergibt sich folgende Zeichnung für die Rotation nach links:

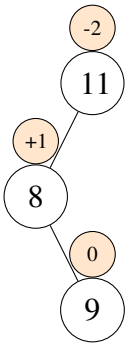
L-Rotation



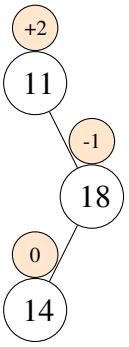
Die **R-Rotation** läuft auf symmetrische Art und Weise ab.

Betrachten wir nun die Bäume 3 und 4.

Baum 3:

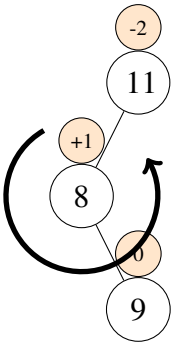


Baum 4:

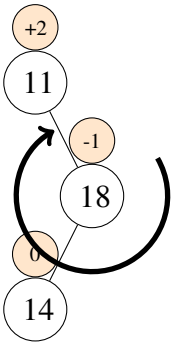


Eine einfache Rotation um die Wurzel würde in diesem Fall keinen Erfolg bringen, da hierdurch die Eigenschaft aus Abschnitt 2.3 verletzt werden würde bzw. die 8 beispielsweise sowohl die 9 als auch die 11 als Nachfolger hätte und damit wiederum einen unbalancierten Baum erzeugen würde. Aus diesem Grund führt man hier eine doppelte Rotation durch. Ein Hinweis darauf, dass eine doppelte Rotation nötig ist, sind die verschiedenen Vorzeichen der Höhendifferenzen: So hat die 11 in Baum 3 die Differenz »-2«, die 8 aber »+1«. Man führt in diesem Fall zuerst eine Rotation der 8 nach links durch, danach dann eine Rotation der Wurzel nach rechts:

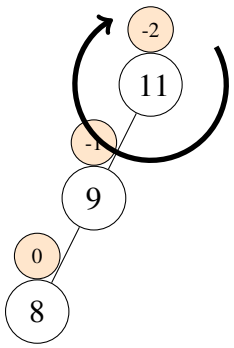
Baum 3:



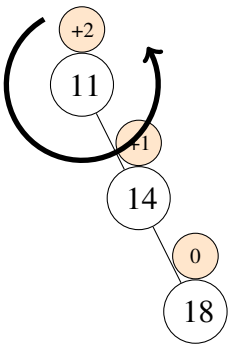
Baum 4:



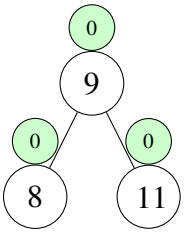
Baum 3:



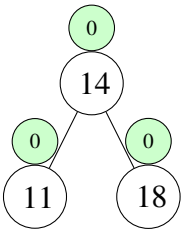
Baum 4:



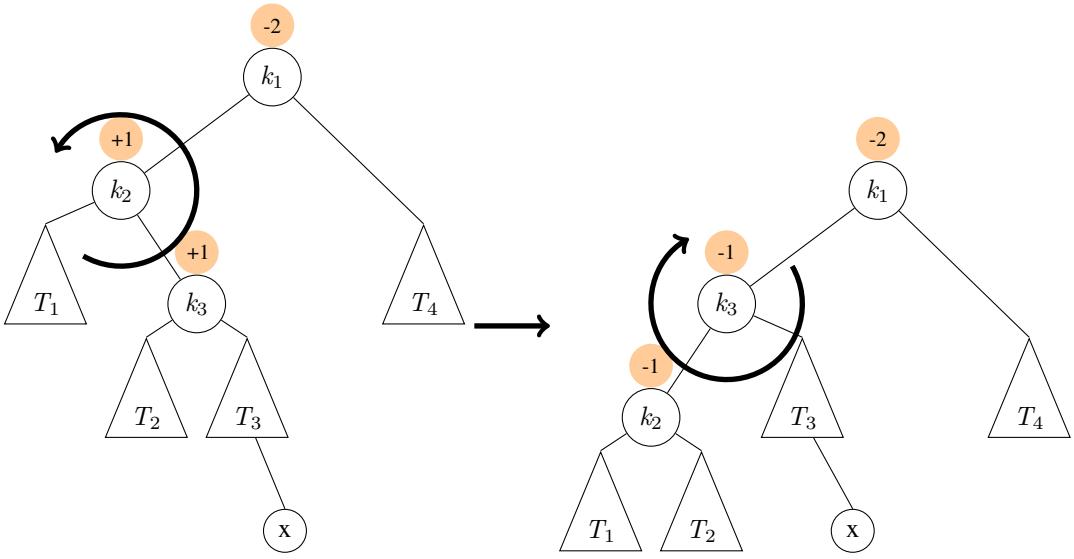
Baum 3:

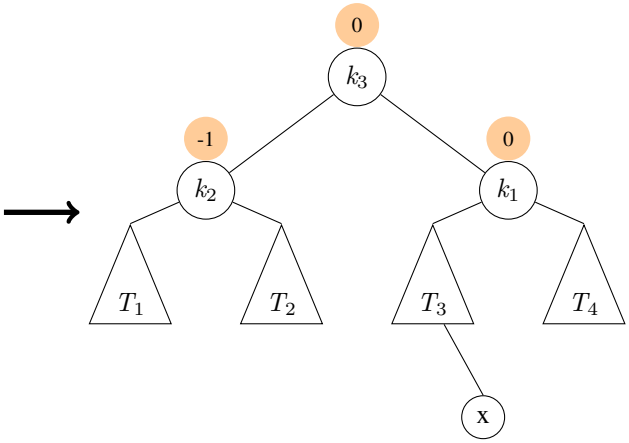


Baum 4:



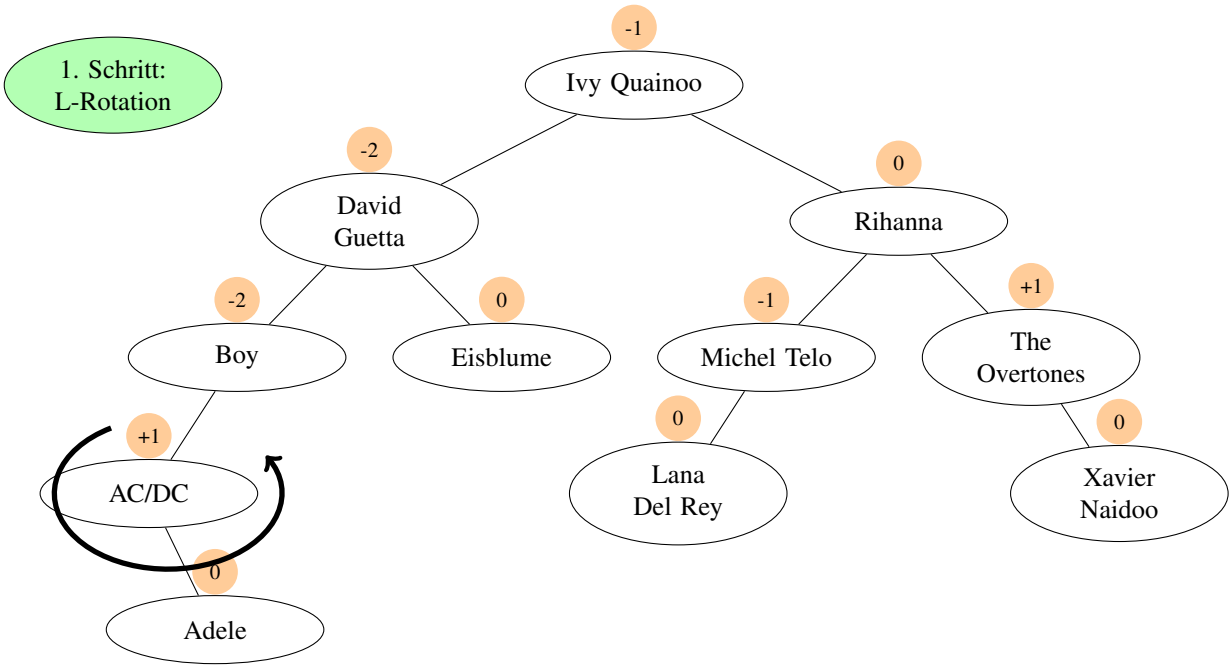
Das entsprechende Verfahren mit evtl. vorhandenen Teilbäume sieht folgendermaßen aus:
LR-Rotation:

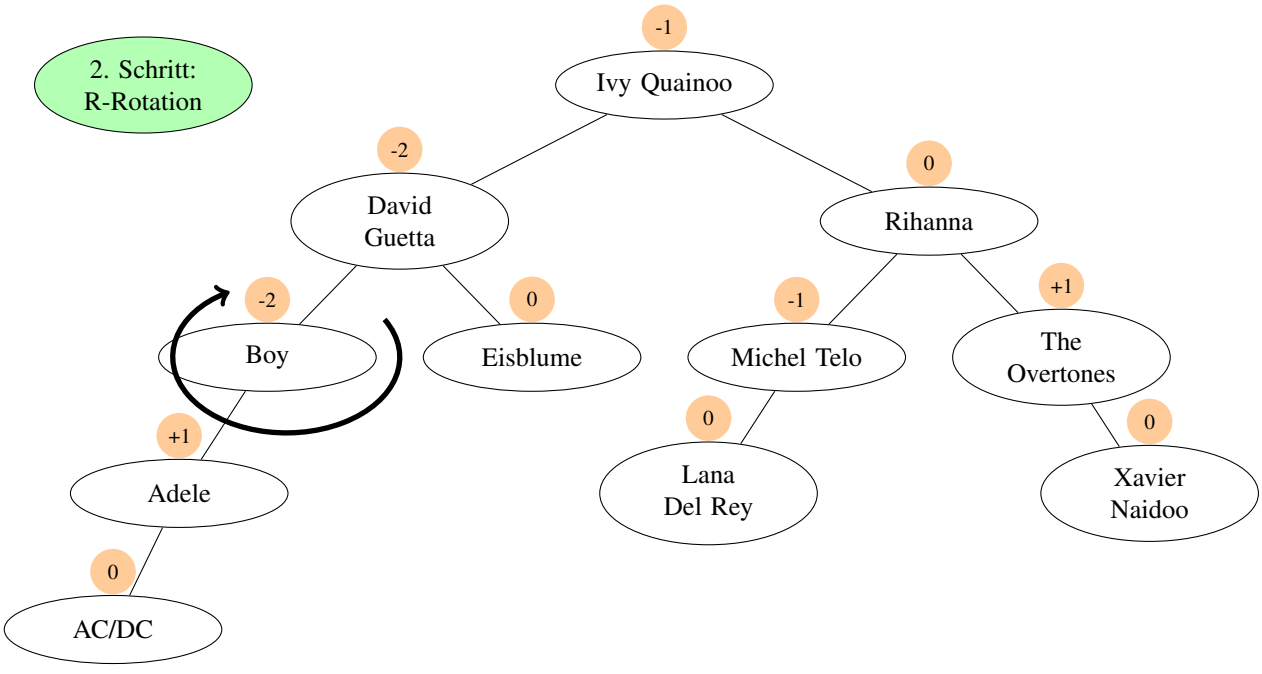


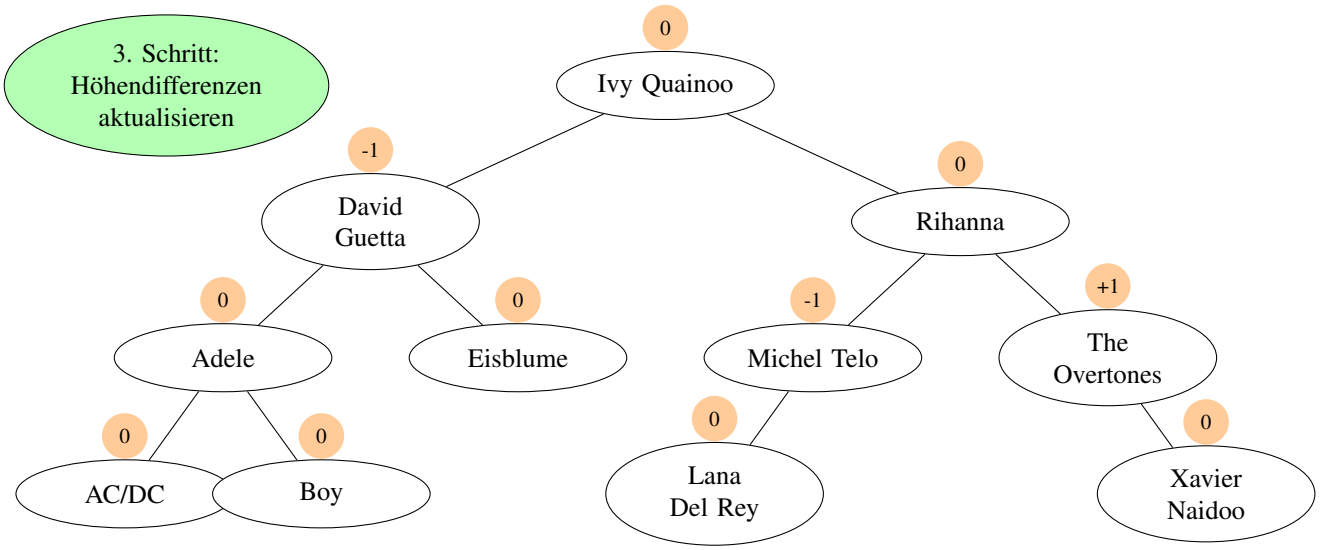


Auch hier existiert ein Gegenstück, das symmetrisch abläuft: Die **RL-Rotation**.

Aufgabe 4.6: Beim Eintragen der Höhendifferenzen fällt auf, dass zuerst die Differenz im Knoten »Boy« auf »-2« verändert wird. Dieser Knoten muss demnach rotiert werden. Da die Höhendifferenz des Nachfolgeknotens »AC/DC« ein anderes Vorzeichen hat als das »Boy«, muss eine Doppelrotation durchgeführt werden. Zudem handelt es sich um ein linksseitiges Übergewicht, demnach muss eine LR-Rotation durchgeführt werden:







Damit ist der Baum ausgeglichen.

Lernkontrolle

Aufgabe 1: Ein binärer Suchbaum ist ausgeglichen, wenn die Höhendifferenz in allen Knoten 0 beträgt.

- ☐ Richtig
- ☒ Falsch

Aufgabe 2: Eine positive Höhendifferenz deutet darauf hin, dass der rechte Teilbaum eines Knotens größer ist als der linke.

- ☒ Richtig
- ☐ Falsch

Kapitel 5

Additum – Ein paar Schritte weiter

5.1 Eine Effizienzanalyse

Als Informatiker ist man immer sehr daran interessiert zu erfahren, wie aufwändig ein neues Verfahren ist. Gerade im Zusammenhang mit großen Datenmengen ist die Effizienz eines Algorithmus überaus wichtig. Wir wollen in diesem Abschnitt eine Effizienzanalyse für die binäre Suche durchführen.

Vom intuitiven Verständnis her ist Ihnen sicher schon klar, dass die binäre gegenüber der linearen Suche schneller zu Ergebnissen führt. Generell kann man sagen, dass bei man bei der linearen Suche in n verschiedenen Elementen im Schnitt ca. $n/2$ Elemente betrachten muss, im schlimmsten Fall (*worst case*) sogar n Elemente.

Aufgabe 5.1

Wir wollen folgende Frage beantworten: Wie viele Versuche benötigt man im Schnitt bei der binären Suche in einer Menge mit n Elementen? Gehen Sie zur Beantwortung dieser Frage folgendermaßen vor:

- Wie viele Elemente können Sie mit einem einzigen Vergleich durchsuchen?
  Wir gehen ab jetzt übrigens immer davon aus, dass das gesuchte Element auch tatsächlich in der Menge vorhanden ist!
- Verdopplen nun die Anzahl der Vergleiche:
  Wie viele Elemente können Sie mit zwei Vergleichen durchsuchen?
- Dieses Spiel setzen Sie nun fort.
  Wie viele Elemente können mit drei, vier, fünf, ... Vergleichen durchsucht werden?
  Untersuchen Sie, ob Sie einen mathematischen Zusammenhang finden.
- Nun nehmen Sie, dass Sie eine Menge mit n Elementen durchsuchen möchten. Setzen Sie jetzt das Ergebnis aus der vorherigen Aufgabe in Zusammenhang mit der Zahl n. Beantworten Sie anschließend die folgende Frage:
  Wie viele Vergleiche benötigt man, um 10.000 CDs zu durchsuchen?

5.2 Rückwärts durch den Baum

Sie haben in Kapitel 2 den »in order«-Durchlauf durch einen Baum kennen gelernt. An dieser Stelle könnne Sie Ihr Wissen erneut überprüfen:

Aufgabe 5.2

- Entwickln Sie einen Algorithmus, um einen Baum in umgekehrter Reihenfolge ausgeben zu lassen.

Diesen Algorithmus nennt man den »reverse in-order«-Durchlauf.

5.3 Knoten löschen

Den Algorithmus zum Einfügen von Knoten haben Sie bereits in Kapitel 3 erarbeitet. Da der binäre Suchbaum als dynamische Datenstruktur fungieren soll, die sich regelmäßig verändert, wird natürlich ebenfalls eine Operation zum Löschen von Knoten benötigt. Diese erarbeiten Sie sich in der folgenden Aufgabe.

Aufgabe 5.3

Machen Sie sich Gedanken, wie das Löschen eines Knotens in einem binären Suchbaums zu bewerkstelligen ist. Unterscheiden Sie dabei zwischen verschiedenen Fällen, die auftreten können:

- Der Knoten hat keine Nachfolger.
- Der Knoten hat nur einen linken oder rechten Teilbaum.
- Der Knoten hat sowohl einen linken als auch einen rechten Teilbaum.

Es geht in dieser Aufgabe nicht darum, dass die Bäume nach den Löschoperationen wieder ausgeglichen werden (da es passieren kann, dass die Balance verletzt wird), sondern wirklich nur um den Löschvorgang an sich!

Lösungen

Aufgabe 5.1 Angenommen, unsere Liste besteht aus zwei Elementen, A und B. Mit einem einzigen Vergleich kann man nun eines dieser beiden Elemente finden: Betrachtet man zuerst Element B und vergleicht es mit dem gesuchten Element, so gibt es zwei Möglichkeiten:

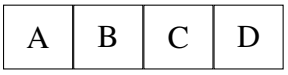
Fall 1: B ist das gesuchte Element, unsere Suche endet hier.

Fall 2: B ist nicht das gesucht Element. Da wir vorausgesetzt haben, dass das gesuchte Element in der Liste vorhanden sein muss, ist das andere Element der Liste das gesuchte.

Nun gehen wir den umgekehrten Weg und fragen uns: Wie viele Elemente kann ich mit x Vergleichen durchsuchen?

Nehmen wir nun zwei Vergleiche: Es ist bekannt, dass man mit einem Vergleich zwei Elemente durchsuchen kann. Da sich mit jedem Vergleich die Menge der noch zu durchsuchenden Elemente halbiert, muss man demnach mit zwei Vergleichen vier Elemente durchsuchen können.

Veranschaulichen wir dies:



Angenommen, wir suchen nach Element A. Wir beginnen unsere Suche bei Element C:

Wir wissen nun, dass das gesuchte Element im linken Teil der Liste liegen muss. Demnach schließen wir C und D aus und betrachten nun B:

Dieses schließen wir ebenfalls aus, da es nicht das gesuchte Element ist und wissen demnach, dass das gesuchte A links vom B stehen muss. Man sieht also: Mit zwei Vergleichen kann man vier Elemente durchsuchen – immer vorausgesetzt, dass das Element auch in der Liste vorhanden ist.

Dies kann man nun fortsetzen:

Vergleiche	**Anzahl Elemente**
1	2
2	4
3	8
4	16
$\vdots$	$\vdots$
x	2^x

Da sich die Anzahl der durchsuchbaren Elemente mit jedem Schritt verdoppelt, ergibt sich die Anzahl der durchsuchbaren Elemente für x Vergleiche somit als 2^x.

Kehren wir nun zu unserer Frage aus der Aufgabenstellung zurück: **Wie viele Vergleiche braucht man maximal für** 10.000 **Elemente?**

Wir können dies nun in Form einer mathematischen Gleichung ausdrücken:

$$2^x = 10.000$$

Um diese Gleichung lösen zu können, brauchen wir den *Logarithmus* als Hilfsmittel. Dann können wir die Gleichung schreiben als:

$$\log_2 10.000 = x$$

Oder, um es besser in den Taschenrechner eingeben zu können:

$$\frac{\log 10.000}{\log 2} = x,$$

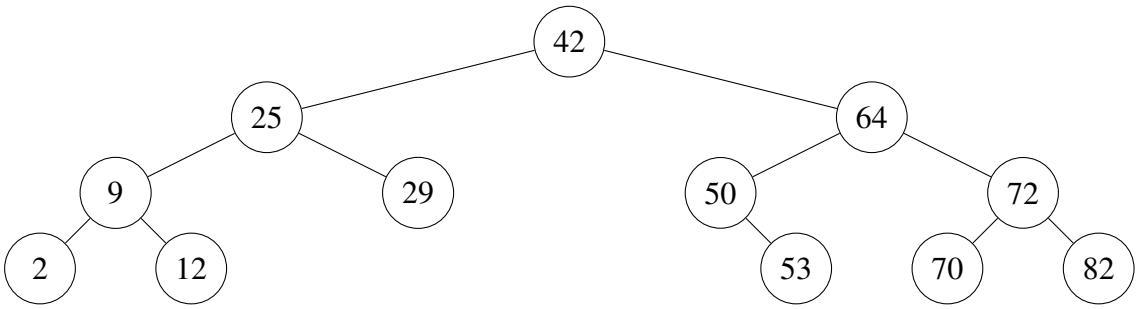
wobei $\log$ der Logarithmus zur Basis 10 ist. Damit ergibt sich $x \approx 13,288$. Da wir immer nur ganz Vergleiche betrachten können, runden wir das Ganze somit zu 14 Vergleichen auf.

Diese 14 Vergleiche müssen nun im Kontrast gesehen werden zur linearen Suche, die für 10.000 Elemente im Schnitt 5.000 Schritte benötigen würde.

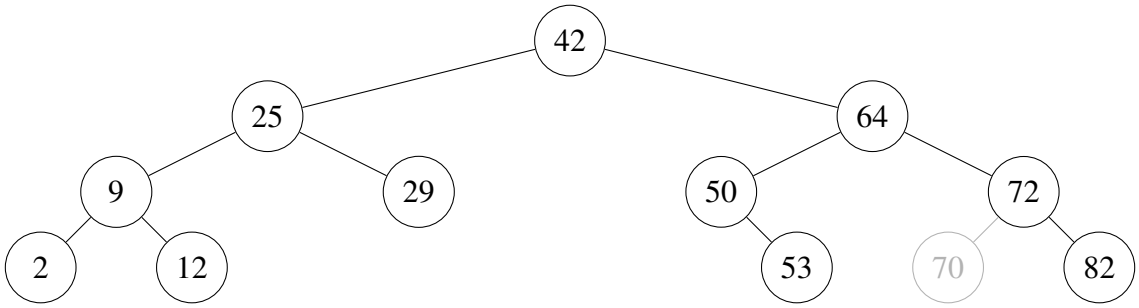
Aufgabe 5.2: Der Algorithmus basiert auf dem Wissen, dass das größte Element in einem binären Suchbaum immer ganz rechts außen platziert ist. Bedenkt man dies und die Vorgehensweise beim »in order«-Durchlauf, so ergibt sich der folgende Algorithmus:

- Steige an der Wurzel in den Baum ein.
- Wiederhole für jeden Knoten von der Wurzel aus folgende Schritte:
  - Steige in den rechten Teilbaum ein und beginne für jeden Knoten diese Abfolge von Schritten. Ist der rechte Teilbaum leer, gehe zum nächsten Schritt.
  - Gib den Knoten aus.
  - Steige in den linken Teilbaum ein und beginne für jeden Knoten diese Abfolge von Schritten erneut.

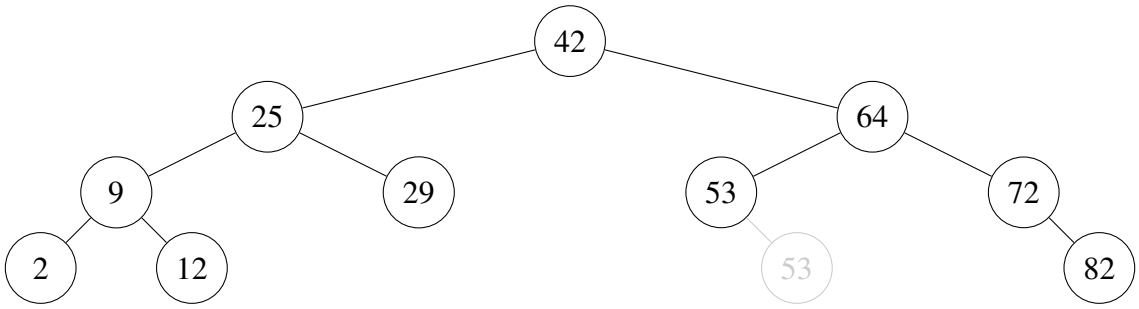
Aufgabe 5.3: Wir überlegen uns unser Vorgehen am Beispiel des folgenden Suchbaums:



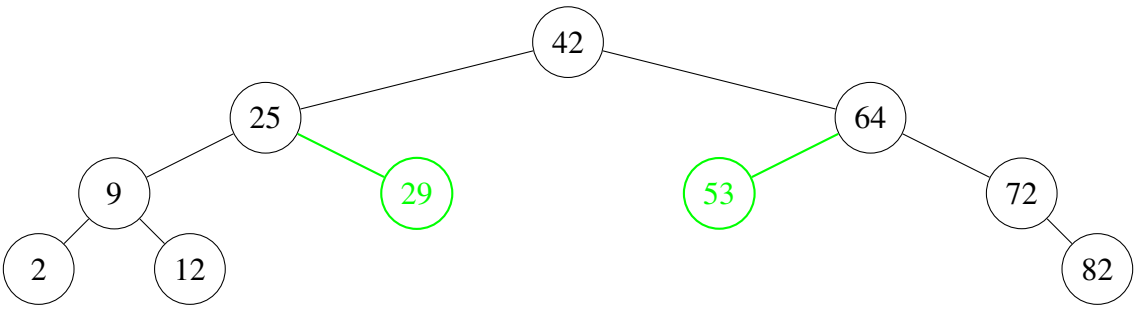
Fall 1: Wir löschen den Knoten mit dem Wert 70. Da dieser Knoten ein Blatt ist, müssen wir nichts weiter beachten und können den Knoten einfach aus dem Baum entfernen. Daraus entsteht der folgende Baum:



Fall 2: Wir löschen den Knoten mit dem Wert 50. Dieser Knoten hat nur einen rechten Teilbaum. Wir können den Teilbaum somit einfach »umhängen«, indem wir an die Stelle der 50 die Wurzel des rechten Teilbaums setzen (in diesem Fall nur ein Knoten). Es entsteht der folgende Baum:



Fall 3: Der komplexeste Fall tritt ein, wenn ein Knoten gelöscht werden soll, der sowohl einen linken als auch rechten Teilbaum hat. In diesem Fall stellt sich die Frage, welche Knoten aus den Teilbäumen den gelöschten ersetzen soll. Betrachten wir nochmals den Baum:



Es wurden die beiden Knoten markiert, die als potentielle Ersatzknoten in Frage kommen: Entweder der größte Knoten aus dem linken Teilbaum (da alle Knoten links der Wurzel kleiner sein müssen als diese) oder der kleinste Knoten rechts im Teilbaum (da alle Knoten rechts der Wurzel größer sein müssen als diese). Hätte man beispielsweise den Knoten 12 als Ersatz für die Wurzel genommen, wäre diese Eigenschaft unmittelbar verletzt worden, da der linke Nachfolger 25 größer ist als die 12. Es gibt somit zwei Möglichkeiten, den Wurzelknoten zu ersetzen.

Achtung: Die Knoten, die aus dem Baum entfernt werden, haben möglicherweise noch einen anhängenden Teilbaum. Dieser muss dann wiederum – entsprechend Fall 2 – gelöscht werden.

Literaturverzeichnis

[Brunninger und Hirsch 2001] BRUNNINGER, Martin ; HIRSCH, Martina: *OOP-Projekt: AVL Bäume*. 2001. – URL `http://fbim.fh-regensburg.de/~saj39122/bruhi/index.html`

[Claus und Schwill 2006] CLAUS, Volker ; SCHWILL, Andreas: *Duden Informatik A-Z. Fachlexikon für Studium, Ausbildung und Beruf*. 4. Auflage. Mannheim : Dudenverlag, 2006. – ISBN 3-411-91016-X

[Kloss 1997] KLOSS, John: *Animated Binary Tree*. 1997. – URL `http://www.cs.jhu.edu/~goodrich/dsa/trees/btree.html`

[Vöcking u. a. 2008] VÖCKING, Berthold ; ALT, Helmut ; DIETZFELBINGER, Martin ; REISCHUK, Rüdiger ; SCHEIDLER, Christian ; VOLLMER, Heribert ; WAGNER, Dorothea: *Taschenbuch der Algorithmen*. Berlin, Heidelberg : Springer, 2008. – ISBN 978-3-540-76393-2