Ludwig-Maximilians-Universität München
Institut für Informatik
Prof. Dr. Thomas Seidl
Anna Beer, Florian Richter

München, 29.05.2018

Algorithmen und Datenstrukturen

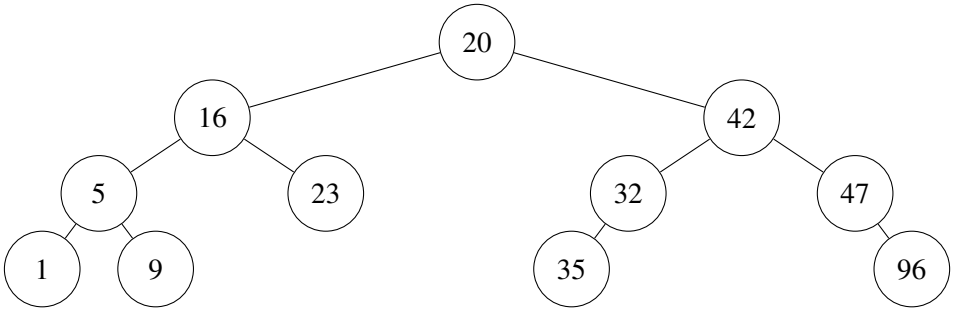
SS 2018

Übungsblatt 7: Suchen

Tutorien: 30.05-01.06.2018

Aufgabe 7-1 *Operationen in Binären Suchbäumen*

Gegeben ist folgender Baum:



Hinweis: Wenn nicht explizit gegeben, gilt hier: $\forall$ Knoten x $\in$ nachfolgender Bäumen: x.key == x.value.

(a) Handelt es sich um einen BST (Binary Search Tree)? Begründen Sie!

(b) Gegeben ist der **get(key)** Algorithmus aus der Vorlesung. Führen Sie den Algorithmus für die folgenden Elemente aus. Geben Sie das Ergebnis sowie die besuchten Knoten an:

(i) get(42)
(ii) get(1)
(iii) get(35)
(iv) get(9)
(v) get(23)

(c) Geben Sie einen binären Suchbaum mit den gleichen Einträgen und der gleichen Baumstruktur an, der obige Suchanfragen alle korrekt beantwortet, d.h. niemals **null** zurückgibt.

(d) Ausgehend von dem nun sortierten Baum aus Teilaufgabe (c). Führen Sie nacheinander folgende Operationen aus und zeichnen Sie nach jeder Operationen den veränderten Baum:

(i) remove(42)
(ii) remove(16)
(iii) insert(39, 39)
(iv) insert(17, 17)
(v) remove(23)

Lösungsvorschlag:

(a) Nein, 23 liegt im linken Teilbaum von 20 und 35 liegt im linken Teilbaum von 32. Es liegen also jeweils größere Elemente im linken Teilbaum, in welchem jedoch alle Elemente kleiner sein müssen, damit ein binärer Baum die BST Eigenschaften erfüllt.

(b) (i) get(42) = 42 ($20 \rightarrow 42$)

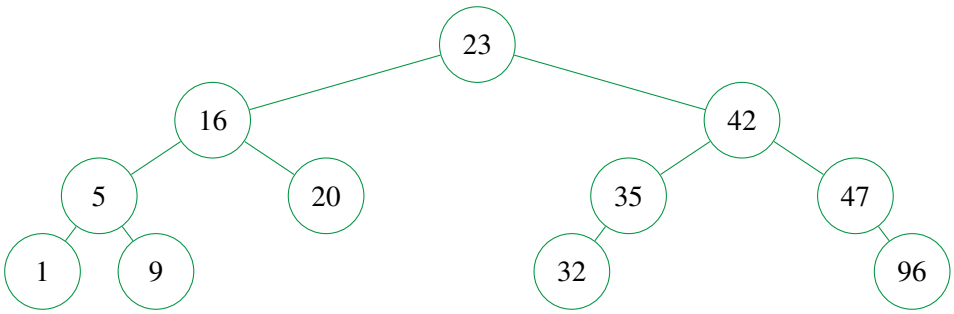
(ii) get(1) = 1 ($20 \rightarrow 16 \rightarrow 5 \rightarrow 1$)

(iii) get(35) = null ($20 \rightarrow 42 \rightarrow 32$)

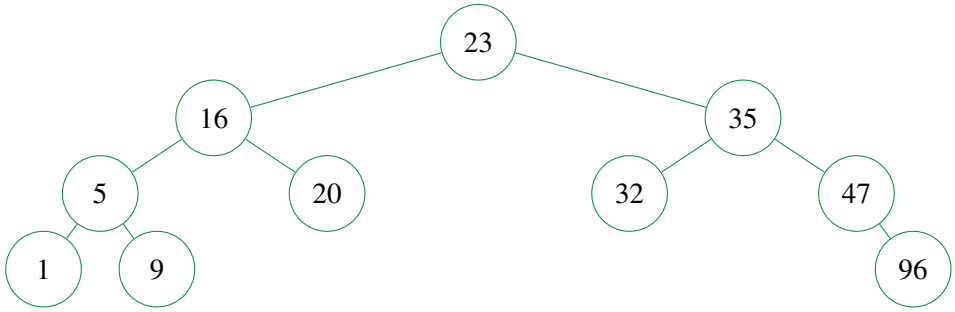
(iv) get(20) = 20 (20)

(v) get(23) = null ($20 \rightarrow 42 \rightarrow 32 \rightarrow 35$)

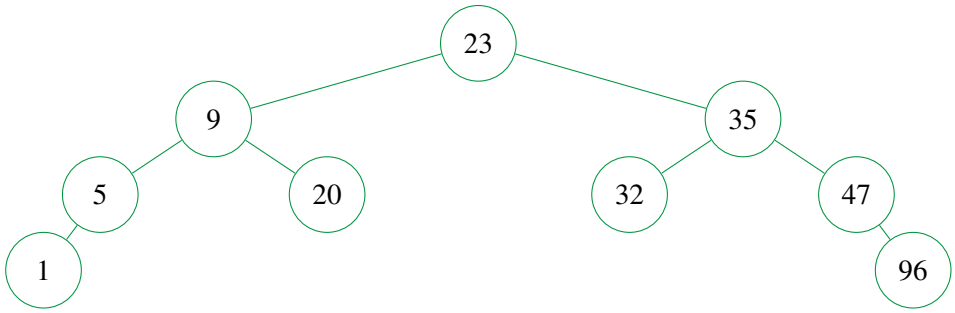
(c) Die eindeutige Lösung ist:



(d) (i) remove(42):

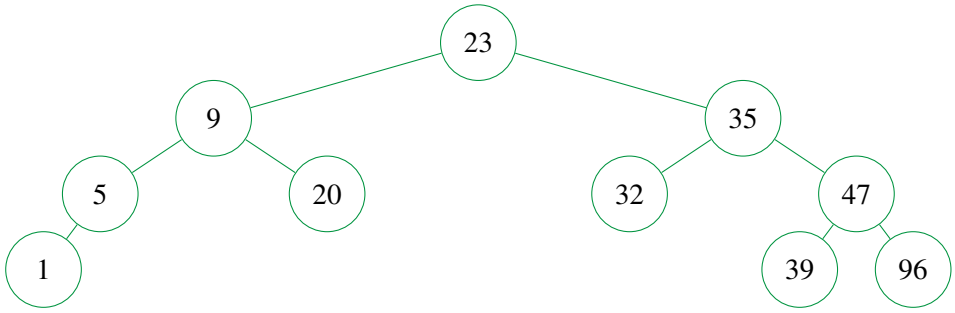


(ii) remove(16):

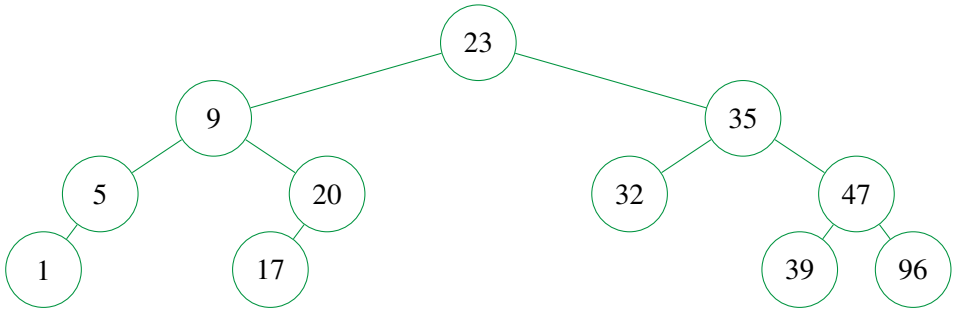


Lösungsvorschlag:

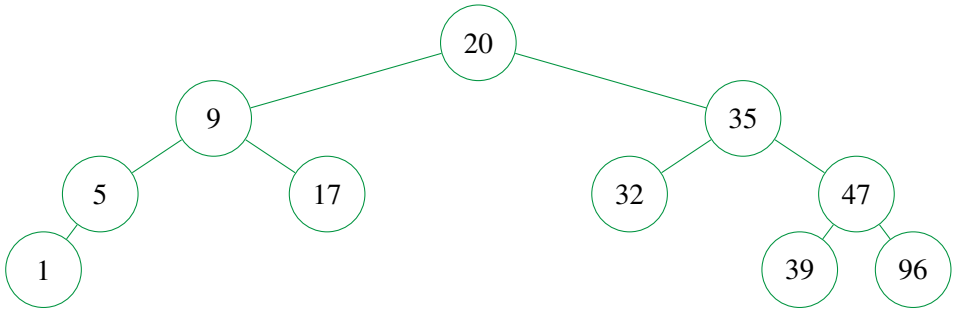
(iii) insert(39, 39):



(iv) insert(17, 17):



(v) remove(23):



Aufgabe 7-2 *AVL-Bäume*

Gegeben ist eine Sammlung von Informatik-Fachbüchern:

Nummer	Titel	Autor	Jahr	Kürzel
1	Design Patterns	E. Gamma, et al.	1994	DP
2	Clean Code	Robert C. Martin	2008	CC
3	Make Your Own Neural Network	Tariq Rashid	2016	MNN
4	Agile Software Development	Robert C. Martin	2002	AgSD
5	Introduction to Algorithms	T. H. Cormen, et al.	1989	IA
6	Functional Thinking: Paradigm Over Syntax	Neal Ford	2014	FuT
7	Extreme Programming Explained: Embrace Change	Kent Beck	1999	ExPE
8	Algorithms for Reinforcement Learning	Csaba Szepesvari	2010	AlRL
9	The Software Craftsman	Robert C. Martin	2014	TheSC
10	Test Driven Development: By Example	Kent Beck	2002	TDD
11	Programming Pearls	Jon Bentley	1986	PP
12	Building Your Own Compiler with C++	Jim Holmes	1994	BYOC

(a) Ihre Aufgabe ist es nun, diese Bücher in exakt der obigen Reihenfolge (auch ersichtlich anhand der Nummern) **nacheinander** in einen anfangs leeren AVL-Baum einzufügen. Gehen Sie hierzu beim Vergleich der Schlüssel von einer **lexikografischen** Ordnung aus. Falls es beim Einfügen zu Rebalancierungen kommt, zeichnen Sie bitte den AVL-Baum davor und danach und sagen Sie, ob es sich um eine einfache oder doppelte Rotation handelt. Zeichnen Sie auch den endgültigen AVL-Baum. Geben Sie ebenfalls immer die AVL-Labels (1; 0; +1) zu den Knoten mit an.
Hinweis: Es reicht die Kürzel anstelle der ganzen Namen in den Baum aufzunehmen.

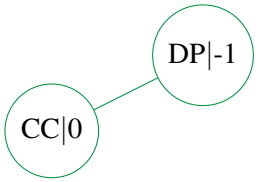
(b) Entfernen Sie aus obigem AVL-Baum nun nacheinander die **5 ältesten** Bücher (ausgehend vom **Erscheinungsjahr/Jahr**). Dabei sollen als erstes das älteste, dann das zweit-älteste usw. entfernt werden. Bei gleichem Erscheinungsjahr wird zunächst das Buch entfernt, dessen Titel lexikografisch kleiner ist.
Zeichnen Sie den Baum, wie er jeweils nach dem Entfernen aussieht. Unveränderte Teilbäume dürfen sie auch als Dreieck abkürzen, wie in der Vorlesung. Vergessen Sie nicht, die AVL-Labels (1; 0; +1) entsprechend den Regeln aus der Vorlesung zu aktualisieren. Geben Sie bei jedem Schritt die verwendeten Rotationen an!

Lösungsvorschlag:

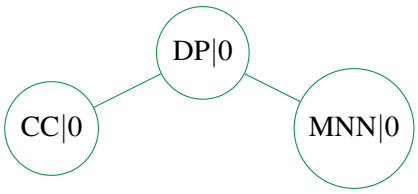
(a)

- Einfügen von Design Patterns/DP:

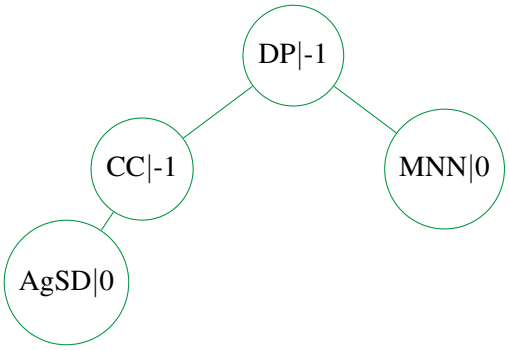
- Einfügen von Clean Code/CC:



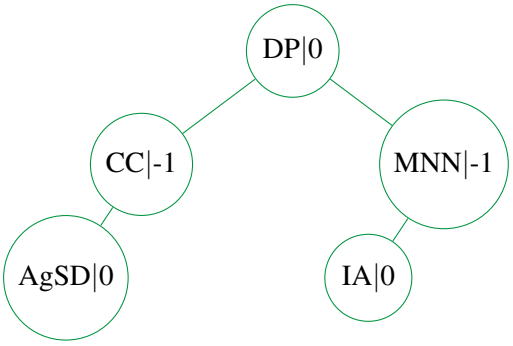
- Einfügen von Make Your Own Neural Network/MNN:



- Einfügen von Agile Software Development/AgSD:

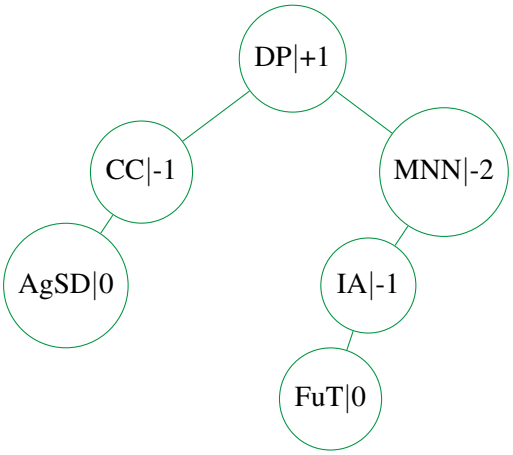


- Einfügen von Introduction to Algorithms/IA:

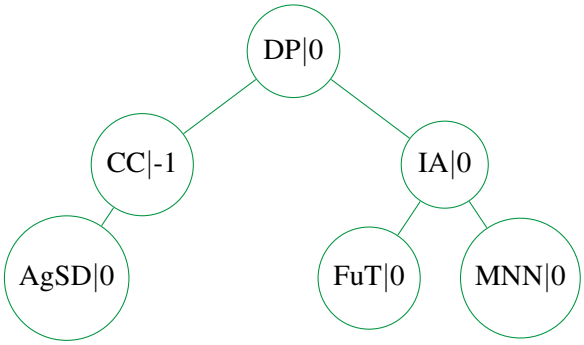


Lösungsvorschlag:

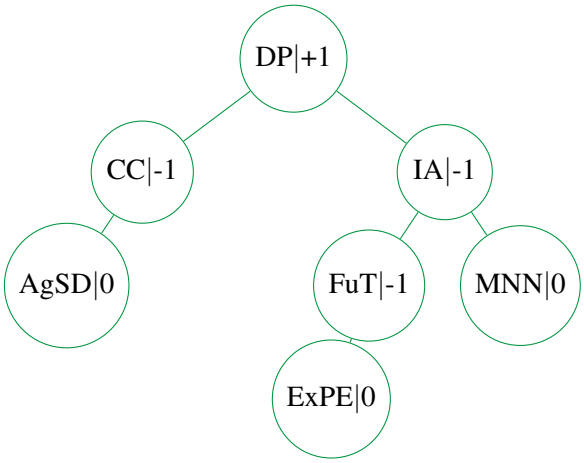
- Einfügen von Functional Thinking: Paradigm Over Syntax/FuT:



Einfache Rotation:

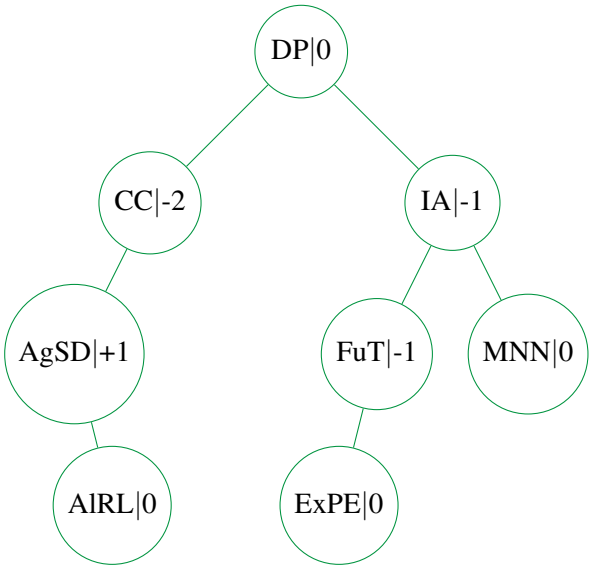


- Einfügen von Extreme Programming Explained: Embrace Change/ExPE:

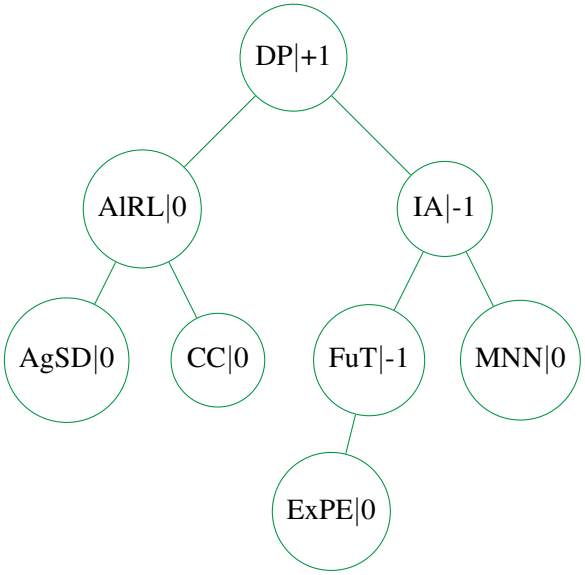


Lösungsvorschlag:

- Einfügen von Algorithms for Reinforcement Learning/AlRL:

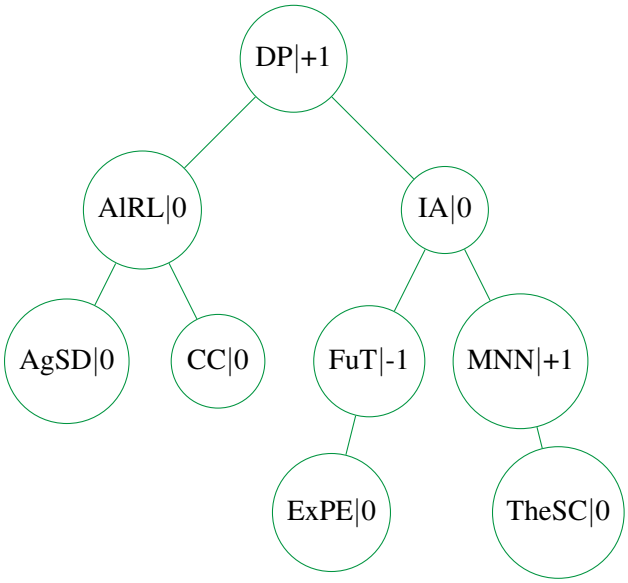


Doppelte Rotation:

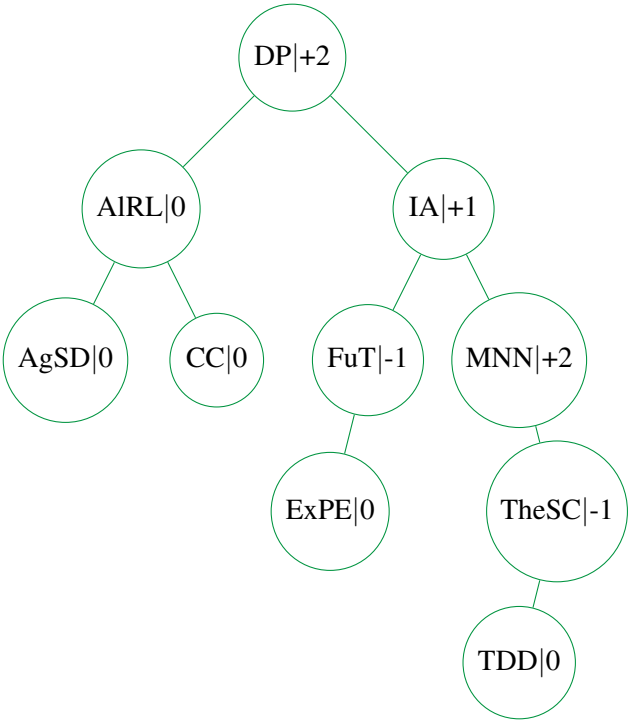


Lösungsvorschlag:

- Einfügen von The Software Craftsman/TheSC:

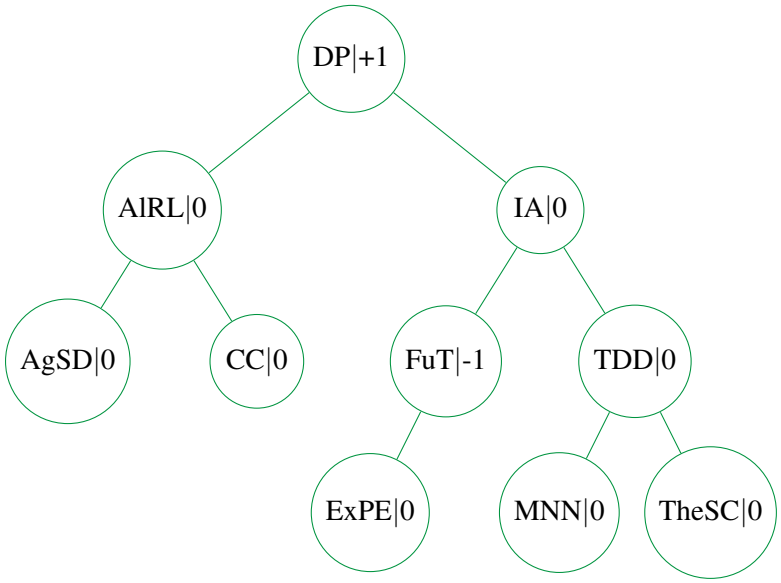


- Einfügen von Test Driven Development: By Example/TDD:

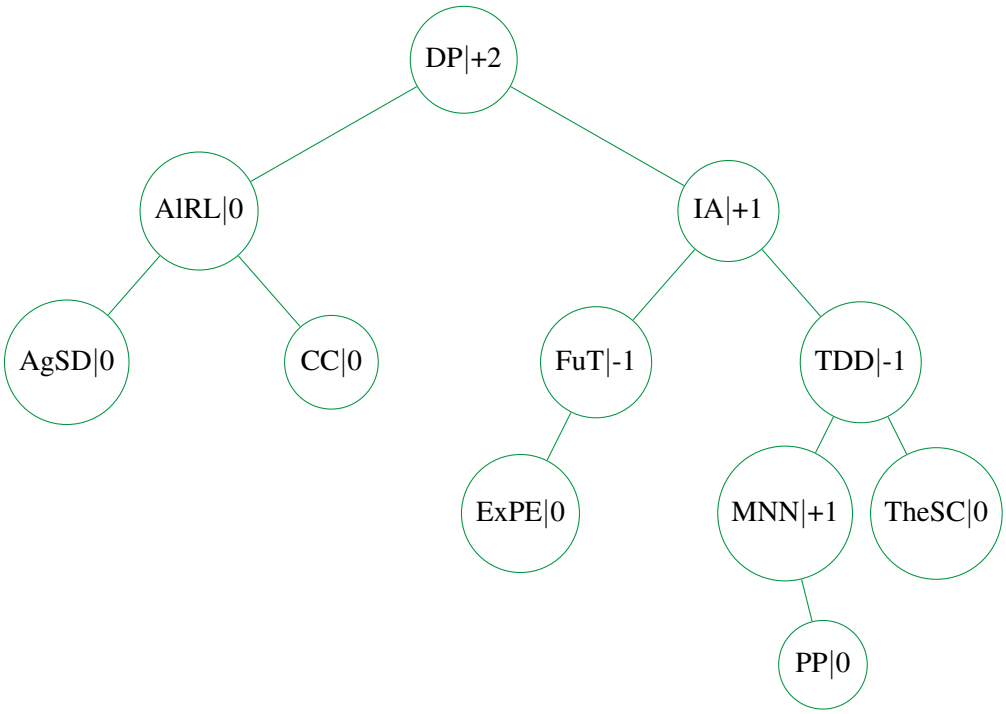


Lösungsvorschlag:

Doppelte Rotation:

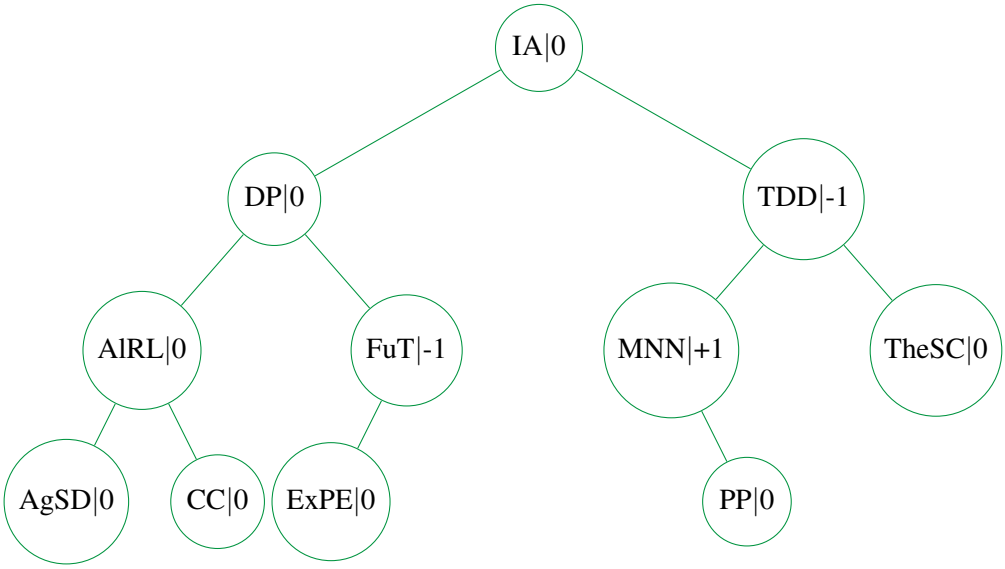


- Einfügen von Programming Pearls/PP:

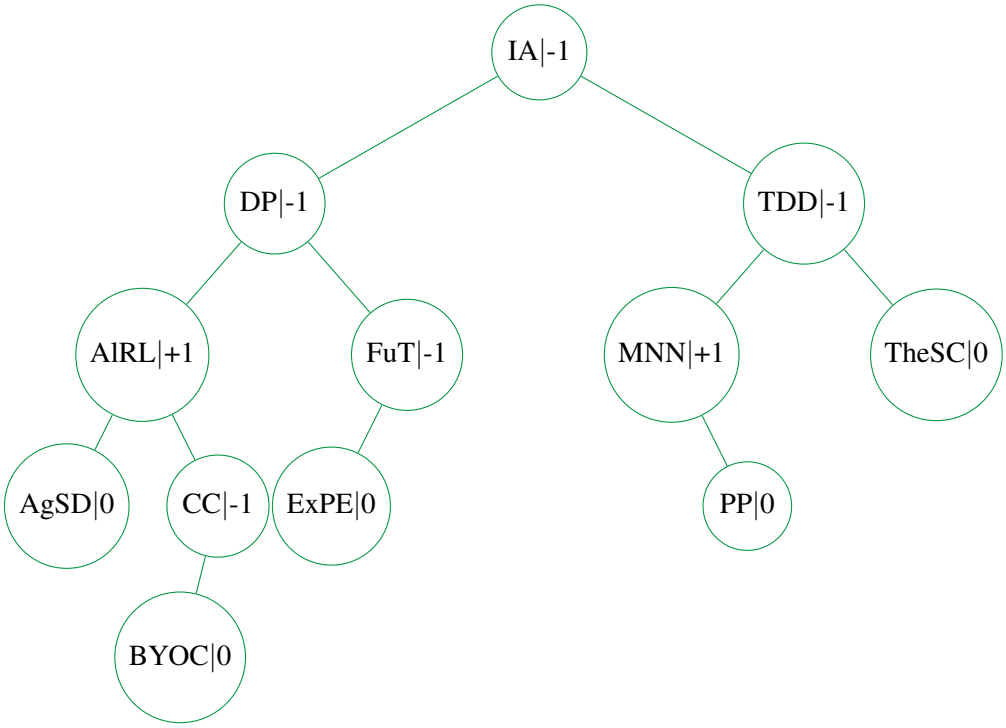


Lösungsvorschlag:

Einfache Rotation:



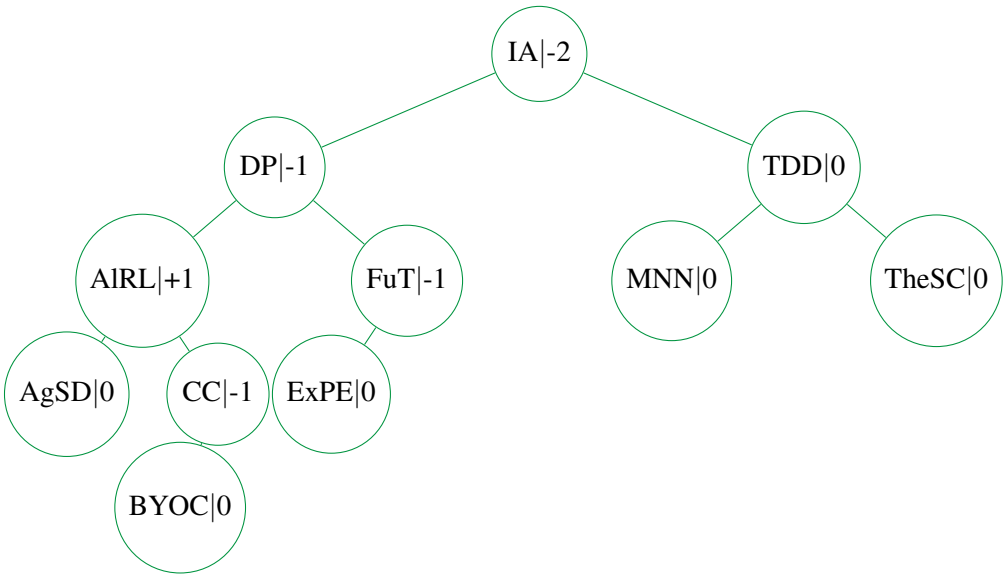
- Einfügen von Building Your Own Compiler with C++/BYOC:



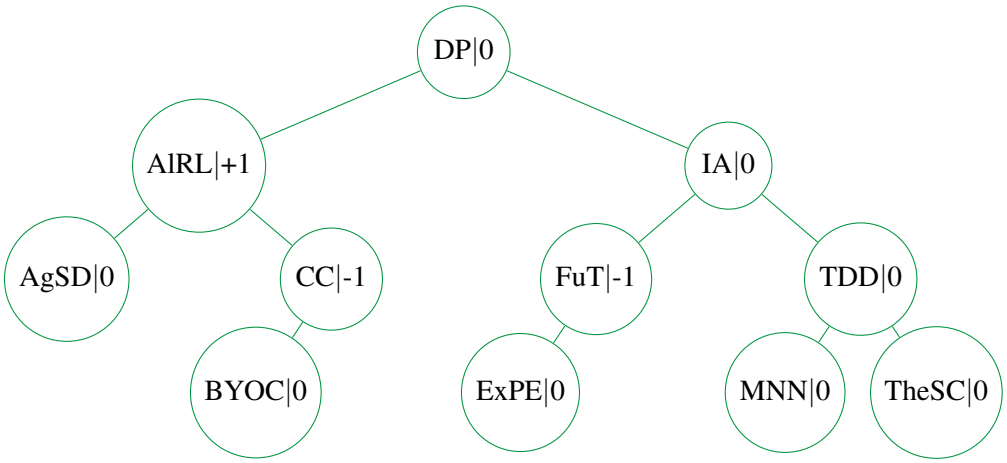
Lösungsvorschlag:

(b)

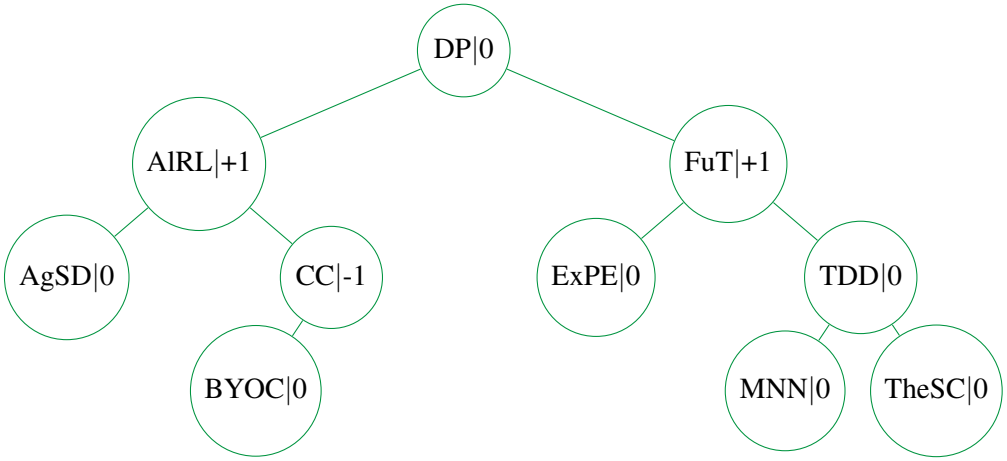
- Entfernen von Programming Pearls/PP (1986):



Einfache Rotation:

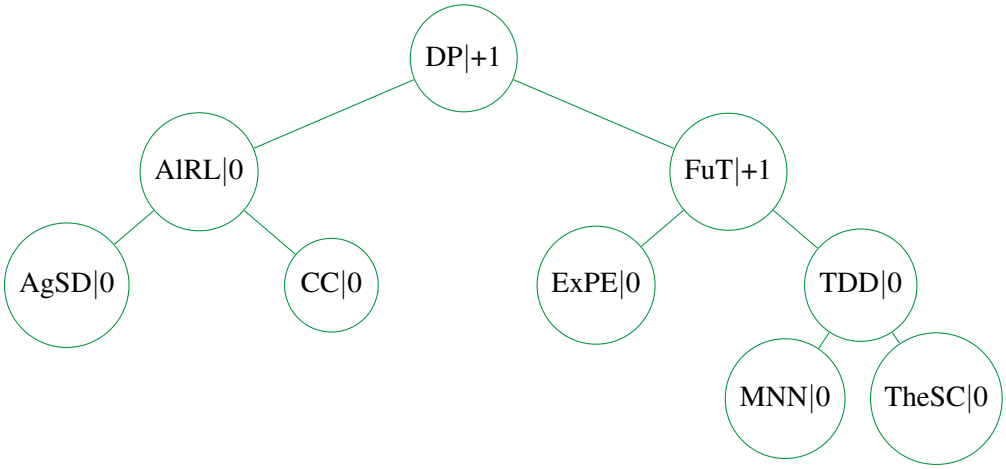


- Entfernen von Introduction to Algorithms/IA (1989):

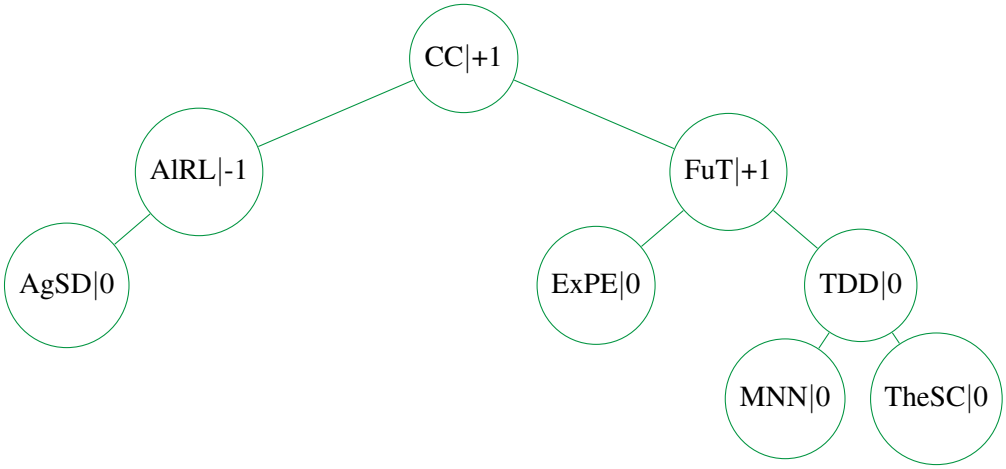


Lösungsvorschlag:

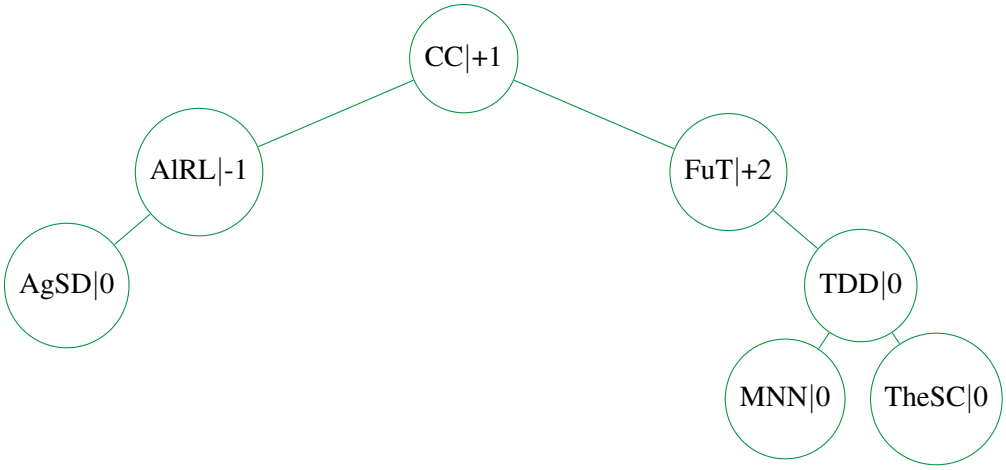
- Entfernen von Building Your Own Compiler with C++/BYOC (1994):



- Entfernen von Design Patterns/DP (1994):



- Entfernen von Extreme Programming Explained: Embrace Change/ExPE (1999)



Lösungsvorschlag:

Einfache Rotation: