

Algorithmen und Datenstrukturen

SoSe25

-Assignment 4-

Moritz Ruge

Matrikelnummer: 5600961

Lennard Wittenberg

Matrikelnummer: —

Mai 2025

Problem 1: Rot-Schwarz Bäume und (2,4)-Bäume

In Aufgabe 3 auf dem 3. Aufgabenblatt wurden rot-schwarz Bäume definiert.

a) Zeigen Sie: Rot-schwarz Bäume und $(2, 4)$ -Bäume sind äquivalent. Genauer: es gibt eine lokale Transformation, welche Gruppen von Knoten im rot-schwarz Baum in Knoten im $(2, 4)$ -Baum überführt, und umgekehrt. Geben Sie eine solche Transformation an, und begründen Sie, dass Ihre Transformation die Bedingungen an rot-schwarz Bäume und an $(2, 4)$ -Bäume erfüllt.

Problem 2: (2,3)-Bäume und (2,4)-Bäume

a) Fügen Sie die Schlüssel A, L, G, O, D, T, S, X, Y, Z in dieser Reihenfolge in einen anfangs leeren (2, 3)-Baum ein. Löschen Sie sodann die Schlüssel Z, A, L. Zeichnen Sie den Baum nach jedem Einfüge- und Löschvorgang, und zeigen Sie die Modifikation, welche durchgeführt werden.

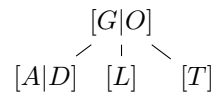
Ein (2,3)-Baum hat folgende Grenzen:

- min children: 2, max children: 3
- min entries: 1, max entries: 2
- A,B,C,D,E,F,G,H,I,J,K,L,M,N,O,P,Q,R,S,T,U,V,W,X,Y,Z

1. insert(A):

[A]

⇒ Das rechte Blatt [L|O|T] hat zu viele Einträge → Rebalance



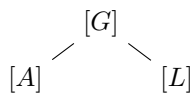
2. insert(L):

[A|L]

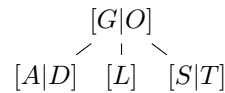
3. insert(G):

[A|G|L]

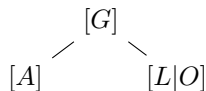
⇒ Die Wurzel hat zu viele Einträge → Split-



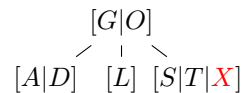
7. insert(S):



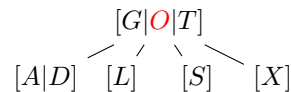
4. insert(O):



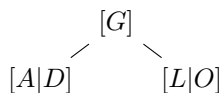
8. insert(X):



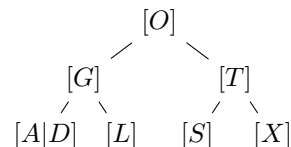
⇒ Das rechte Blatt [S|T|X] hat zu viele Einträge → Rebalance



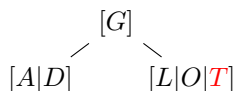
5. insert(D):



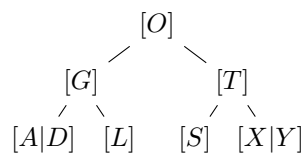
⇒ Die Wurzel hat zu viele Einträge & zu viele Kinder → Rebalance



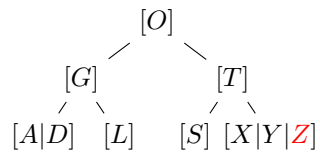
6. insert(T):



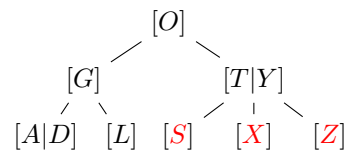
9. insert(Y):



10. insert(Z):



⇒ Der Knoten $[X|Y|Z]$ hat zu viele Einträge → Balancieren



11. remove(Z):

12. remove(A):

13. remove(L):

b) Wiederholen Sie die Teilaufgabe (a) mit einem (2, 4)-Baum.

min children: 2, max children: 4

min entries: 1, max entries: 3

A,B,C,D,E,F,G,H,I,J,K,L,M,N,O,P,Q,R,S,T,U,V,W,X,Y,Z

insert(A):

```

-----
| A |
-----

```

insert(L):

```

-----
| A | L |
-----

```

insert(G):

```

-----
| A | G | L |
-----

```

insert(O): first split

```

-----
| G |
-----
 /      \
| A |      | L | O |

```

insert(D):

```

-----
| G |
-----
 /      \
| A | D |  | L | O |

```

insert(T): second split

```

-----
| G | O |
-----
 /      |      \
| A | D | | L |  | T |

```

insert(S):

```

-----
| G | O |
-----
 /      |      \
| A | D | | L |  | S | T |

```

insert(X):

```

-----
| G | O |

```

```

      -----
     /      |      \
    /      |      \
   | A | D | | L |   | S | T | X |
insert(Y): rebasing the root -> third split

```

```

      -----
     | G      |      T |
      -----
    /      |      \
   /      |      \
  | A | D | | L | O | S | | X | Y |
insert(Z):

```

```

      -----
     | G      |      T |
      -----
    /      |      \
   /      |      \
  | A | D | | L | O | S | | X | Y | Z |
'''

```

Starting tree:

```

      -----
     | G      |      T |
      -----
    /      |      \
   /      |      \
  | A | D | | L | O | S | | X | Y | Z |

```

delete(Z): delete from leaf

```

      -----
     | G      |      T |
      -----
    /      |      \
   /      |      \
  | A | D | | L | O | S | | X | Y |

```

delete(A): delete from leaf -> min 1 key required -> condition holds true

```

      -----
     | G      |      T |
      -----
    /      |      \
   /      |      \
  | D |      | L | O | S | | X | Y |

```

delete(L): delete from Leaf -> 2 keys pressed -> node requirements satisfied -> $G < O$ and $S < T$ -> Order condition satisfied.

```

      -----
     | G      |      T |
      -----

```

D	/	
O	S	\ \
X	Y	

Problem 3: (a,b)-Bäume

a) Beschreiben Sie, wie man in einem (a, b)-Baum mit n Schlüsseln die Operation $\text{succ}(k)$ implementieren kann. Was ist die Laufzeit?

b) Beschreiben Sie, wie man in einem (a, b)-Baum mit n Schlüsseln die Operation $\text{findRange}(k_1, k_2)$ implementieren kann, die alle Schlüssel k liefert, für die $k_1 \leq k \leq k_2$ ist. Die Laufzeit soll $O(\log n + s)$ betragen. Dabei ist s die Anzahl der gelieferten Schlüssel.

c) Seien T_1 und T_2 zwei (a, b)-Bäume, und sei S_1 die Schlüsselmenge von T_1 und S_2 die Schlüsselmenge von T_2 . Sei x ein weiterer Schlüssel. Alle Schlüssel in S_1 sind kleiner als x , und alle Schlüssel in S_2 sind größer als x . Beschreiben Sie eine Operation join , die aus T_1 , T_2 und x einen (a, b)-Baum für die Schlüsselmenge $S_1 \cup x \cup S_2$ erzeugt. Die Laufzeit sollte $O(\log \max(|S_1|, |S_2|))$ betragen. Hinweis: Betrachten Sie zunächst den Fall, dass T_1 und T_2 die gleiche Höhe haben. Achten Sie darauf, dass hinterher die (a, b)-Baum Eigenschaften wieder hergestellt werden.