technische universität dortmund

SoSe 2018

M. Buchin
A. Krivošija, D. Mezlaf
L. Pfahler, A. Rey

DAP2 – Heimübung 9

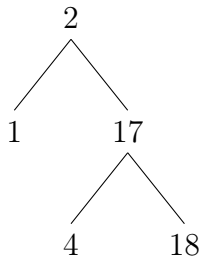
Ausgabedatum: 01.06.2018 — Abgabedatum: Mo. 11.06.2018 bis 12 Uhr

Abgabe:

Schreiben Sie unbedingt immer Ihren vollständigen Namen, Ihre Matrikelnummer und Ihre Gruppennummer auf Ihre Abgaben! Beweise sind nur dort notwendig, wo explizit danach gefragt wird. Eine Begründung der Antwort wird allerdings *immer* verlangt.

Aufgabe 9.1 (5 Punkte): (AVL-Bäume)

a) Gegeben sei folgender AVL-Baum T.



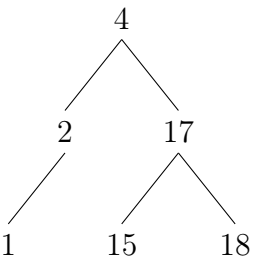
Pflegen Sie die folgenden Elemente in der angegebenen Reihenfolge in die Datenstruktur ein:

$$15, 6, 14, 8, 12, 11$$

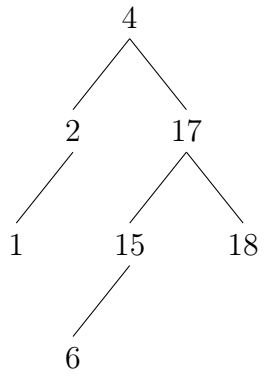
Geben Sie den Baum T nach jeder Operation an. Geben Sie außerdem an, an welchem Knoten und in welche Richtung rotiert wird.

Lösung:

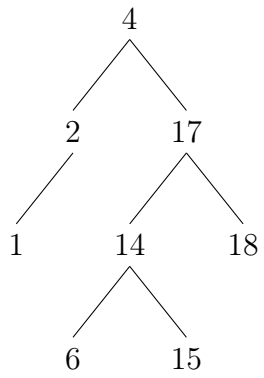
- Wir fügen 15 ein. Rechtsrotation in 17, Linksrotation in 2.



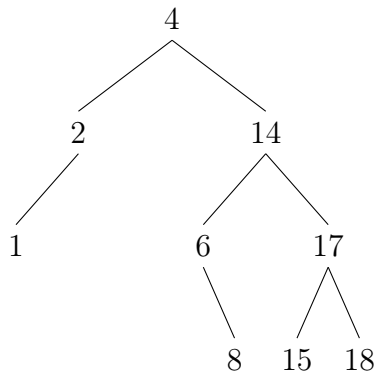
- Wir fügen 6 ein.



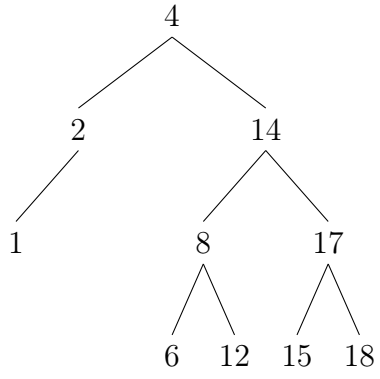
- Wir fügen 14 ein. Linksrotation in 6, Rechtsrotation 15.



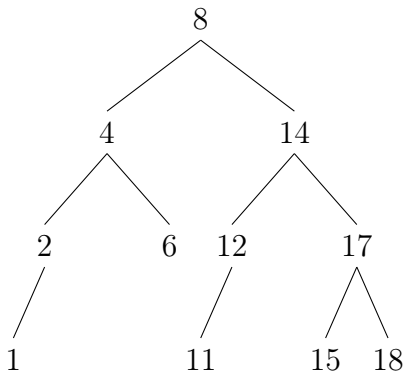
- Wir fügen 8 ein. Rechtsrotation in 17.



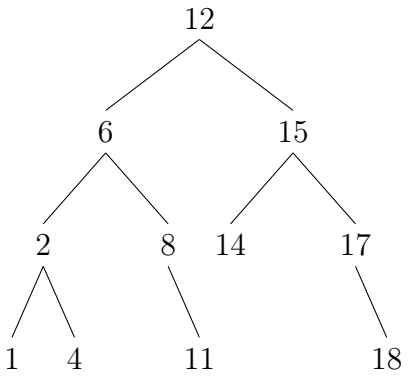
- Wir fügen 12 ein. Linksrotation in 6.



- Wir fügen 11 ein. Rechtsrotation in 14, Linksrotation in 4.



b) Gegeben sei folgender AVL-Baum T'.



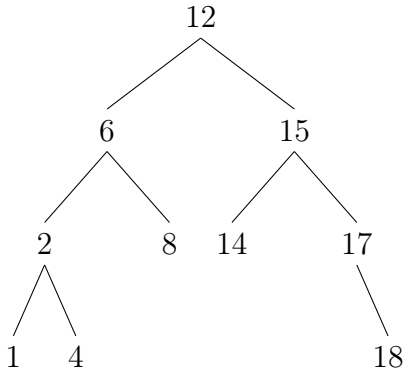
Löschen Sie die folgenden Elemente in der angegebenen Reihenfolge in die Datenstruktur ein:

$$11, 8, 12, 15$$

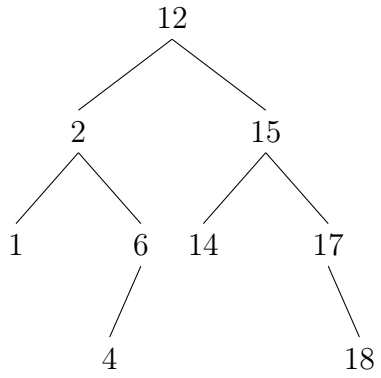
Geben Sie den Baum T' nach jeder Operation an. Geben Sie außerdem an, an welchem Knoten und in welche Richtung rotiert wird und durch welches Element das Gelöschte ersetzt wurde.

Lösung:

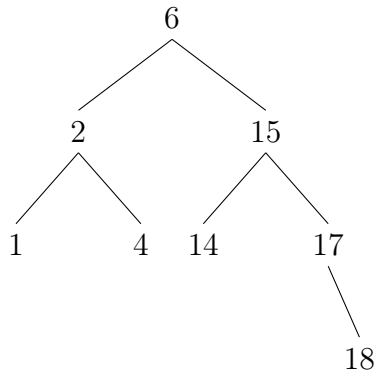
- Wir löschen 11.



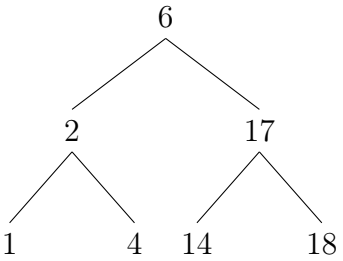
- Wir löschen 8. Rechtsrotation in 6.



- Wir löschen 12. Ersetzen durch 6.



- Wir löschen 15. Ersetzen durch 14. Linksrotation in 14.



Aufgabe 9.2 (5 Punkte): (AVL-Bäume)

Alice zieht schon wieder um. Sie hatte Streit mit Eve und möchte jetzt lieber bei Bob wohnen. In ihren n Umzugskartons gleicher Größe verstaut sie ihre Bücher, wobei Bücherkarton i Gewicht $w_i > 0$ hat. Wir nehmen an, dass alle Gewichte unterschiedlich sind. Für den Umzug hat sie einen Transporter gemietet, der Platz für $k < n$ Kisten bietet. Da die Umzugshelfer am Anfang die meiste Energie und Motivation haben, hat sie sich überlegt, bei der ersten Fahrt k Bücherkartons mit einem möglichst großen Gesamtgewicht zu transportieren.

a) Entwerfen Sie einen Algorithmus zur Berechnung des Gewichts der optimalen Beladung des Transporters $B \subseteq \{1, ..., n\}, |B| = k$ in Laufzeit von $\mathcal{O}(n \log k)$. Beschreiben Sie den Algorithmus in eigenen Worten sowie in Pseudocode.

b) Führen Sie eine Laufzeitanalyse für Ihren in a) gegebenen Algorithmus durch.

c) Beweisen Sie, dass Ihr Algorithmus korrekt das Gesamtgewicht der schwersten k Kisten berechnet.

Lösung:

a) Wir bearbeiten alle n Kisten der Reihe nach und verwenden einen AVL-Baum, in dem zu jedem Zeitpunkt die schwersten k Kisten, die bisher identifiziert wurden, gespeichert werden. Der Schlüssel für Kiste i ist w_i. Wir stellen sicher, dass der AVL-Baum nie mehr als k Elemente enthält, sodass die Operationen AVL-Einfügen, AVL-Löschen und AVL-Minimum immer in $\mathcal{O}(\log k)$ ausgeführt werden können. So ergibt sich eine Gesamtlaufzeit von $\mathcal{O}(n \log k)$.

Wir arbeiten in zwei Phasen: Die ersten k Kisten können unmittelbar im AVL-Baum abgelegt werden, bei allen anderen Kisten wird abgefragt, ob sie schwerer sind als die leichteste im AVL-Baum. Falls das zutrifft, wird das leichteste Element aus dem Baum entfernt und das Gewicht der aktuellen Kiste eingefügt.

`umzug`$(n, k, W[1, ..., n])$:

1. $t \leftarrow$AVL()
2. **for** $j = 1, ..., k$ **do**
3. AVL-Einfügen$(t, W[j])$ //Fügt die Kiste i mit Gewicht w_i ein
4. **for** $j = k + 1, ..., n$ **do**
5. $m \leftarrow$ AVL-Minimum(t) //Liefert das Gewicht der leichtesten Kiste in t
6. **if** $w_j > m$ **then**
7. AVL-Löschen(t, m) //Entfernt die Kiste mit Gewicht m
8. AVL-Einfügen$(t, W[j])$ //Fügt die Kiste j mit Gewicht w_j ein
9. **return** $\sum_{w \in t} w$

b) Die Zeilen 1-3 haben laut Vorlesung, Satz 37 Worst-Case Laufzeit $\mathcal{O}(k \log k)$. Wir analysieren die Laufzeit der zweiten For-Schleife (Zeilen 4-8).
Wir zeigen folgende Invariante: Vor der i-ten Iteration, $i = 1, ..., n-k+1$, enthält t genau k Elemente. (IA) Vor der ersten Iteration gilt $|t| = k$, da die erste For-Schleife genau k Elemente eingefügt hat. (IV) Wir nehmen an, dass für ein beliebiges $1 \leq i \leq n-k$ vor der $i$ten Iteration gilt, dass $|t| = k$. (IS) Falls die Bedingung in Zeile 6 nicht zutrifft, wird t nicht modifiziert. Falls die Bedingung zutrifft, wird ein Element gelöscht und ein Element eingefügt, die Größe $|t| - 1 + 1 = |t|$ verändert sich nicht. Nach IV gilt die Invariante also auch vor der $i + 1$ten Iteration. Somit gilt die Invariante nach Induktion für alle $i = 1, ..., n - k + 1$.

Nun kann die Laufzeit der zweiten Schleife analysiert werden: Der Schleifenrumpf wird $n-k$ mal ausgeführt. Im Worst-Case wird der If-Block jedes mal betreten. Nach Invariante ist $|t| = k$ vor jeder Iteration. Somit sind alle Operationen AVL-Einfügen, AVL-Löschen und AVL-Minimum laut Vorlesung in $\mathcal{O}(\log k)$. Genauer ist die Laufzeit von AVL-Einfügen sogar nur $\mathcal{O}(\log(k-1))$. Für die Schleife ergibt sich also insgesamt eine Laufzeit von $\mathcal{O}((n-k) \log k)$. Nach Invariante gilt $|t| = k$ nach der zweiten For-Schleife. Zeile 15 hat also Laufzeit $\mathcal{O}(k)$. Die Gesamtlaufzeit ist also $\mathcal{O}(k \log k + (n-k) \log k + k) = \mathcal{O}(n \log k)$.

c) Wir zeigen induktiv die folgende Invariante der zweiten For-Schleife: Vor der i-ten Iteration, $i = 1, ..., n-k+1$, enthält t die Gewichte der k schwersten Kisten aus $\{1, ..., k+i-1\}$.
(IA) Vor der ersten Iteration enthält t genau die ersten k Gewichte. Somit enthält es auch die Gewichte der schwersten k der ersten k Kisten.
(IV)Wir nehmen an, dass für ein beliebiges $1 \leq i \leq n-k$ vor der $i$ten Iteration gilt, dass t die Gewichte der schwersten k Kisten aus $\{1, ..., k+i-1\}$ enthält.
(IS)Wir führen die $i$te Iteration aus und betrachten also die $(k+i)$te Kiste. Falls diese Kiste weniger wiegt als die leichteste Kiste in t, gilt nach IV, dass Kiste j leichter ist als die kt-schwerste Kiste aus $\{1, ..., k+i-1\}$. Somit ist sie auch leichter als die k schwersten Kisten aus $\{1, ..., k+i\}$.
Falls die Kiste schwerer ist als die kt-schwerste Kiste aus $\{1, ..., k+i-1\}$, so ist die leichteste Kiste in t nicht mehr unter den schwersten k Kisten. Folgerichtig entfernen wir sie in Zeile 7 und ersetzen sie in Zeile 8 durch die Kiste $j = k+i$. In beiden Fällen gilt nun die Invariante nach der $i-ten$ bzw. vor der $(i+1)$ten Iteration.

Nach der $(n-k)$ten Iteration gilt die Invariante, somit enthält t in Zeile 15 die k-größten Elemente aus $1, ..., n$. Die Summe wird korrekt zurückgegeben.