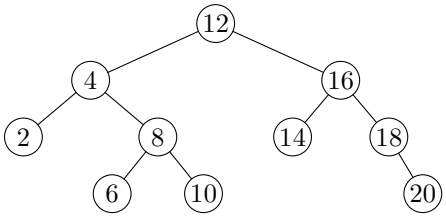
RWTH Aachen
Lehrgebiet Theoretische Informatik
Emmes—Langer—Reidl—Rossmanith

SS 2011
Übungsblatt 4
02.05.2011

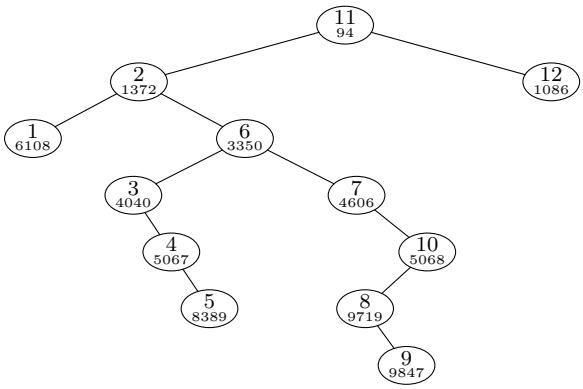
Übung zur Vorlesung Datenstrukturen und Algorithmen

Aufgabe T12 Gegeben ist der folgende AVL-Baum:



Was passiert, wenn wir erst 21 und dann 17 einfügen, danach erst 12 und dann 2 löschen? Schließlich wird die 12 wieder eingefügt. Wie sieht der Baum am Ende aus?

Aufgabe T13



Was passiert, wenn die 11 aus obigen Treap gelöscht und anschließend mit Priorität 3000 wieder eingefügt wird?

Aufgabe T14

Beantworten Sie diese Fragen:

a) Vorgegeben sei ein beliebiger binärer Baum, der die AVL-Eigenschaft erfüllt. Kann man durch die Einfüge- und Löschoperationen für AVL-Bäume einen AVL-Baum mit genau dieser Struktur erzeugen?

b) Können Sie eine Operation *Split* für Treaps angeben, die einen Treap für einen gegebenen Schlüssel s in zwei Treaps T_1, T_2 unterteilt, so daß alle Schlüssel in T_1 kleiner und alle Schlüssel in T_2 größer als s sind?

Aufgabe T15

Das Array a sei nur mit 0en und 1en gefüllt und repräsentiert auf diese Weise eine Binärzahl. Die Funktion *counterStep* erhöht diese Zahl um eins. Geben Sie die amortisierten Kosten der Funktion *counterStep* an, wenn das Array anfänglich nur mit 0en gefüllt ist.

```
void counterStep (int [ ] a) {
    int i = 0;
    while (a[i] ≡ 1) {
        a[i] = 0;
        i++;
    }
    a[i] = 1;
}
```

Aufgabe H10 (10 Punkte)

Einst schickte Frau Mutter, mit eiligen Worten,
klein Timmi zum Markt: „Es gibt neue Waren!
Strukturen für Daten! Verschiedenste Sorten!
Bring mir eine Queue—doch müssen wir sparen.
Drei Groschen für eine, das sollte schon passen."
So lief Timmi fort, den Marktplatz voraus
doch kaum war er dort, konnt' er sich nicht lassen
„Zwei Groschen reichen doch sicherlich aus!"

Von Gier besiegt und mit Eis in der Hand
erschrak Timmi heftig, als er sich besann
„Drei Groschen die Queue" las er dort am Stand—
er zerbrach sich den Kopf und das Eis zerann.

Mit zwei Stacks im Rucksack kehrt er schließlich heim
—wegen reichlicher Ernte bekam er sie beide—
Doch scholt' ihn die Mutter „Das kann doch nicht sein!
Stacks gehen verkehrt, weshalb ich sie meide!"

„Eine Queue, liebe Mutter, bau ich drumherum:
Enqueue-en werd' ich nur in den ersten der beid'
Und will ich dequeue-en, so füll ich sie um.
Amortisiert wird das klappen—in konstanter Zeit."

Hilf klein Timmi! Seine simulierte Queue funktioniert wie folgt:

```
int dequeue() {
  if(right.isEmpty()) {
    while(!left.isEmpty())
      right.push(left.pop());
  }
  return right.pop();
}

void enqueue(int x) {
  left.push(x);
}
```

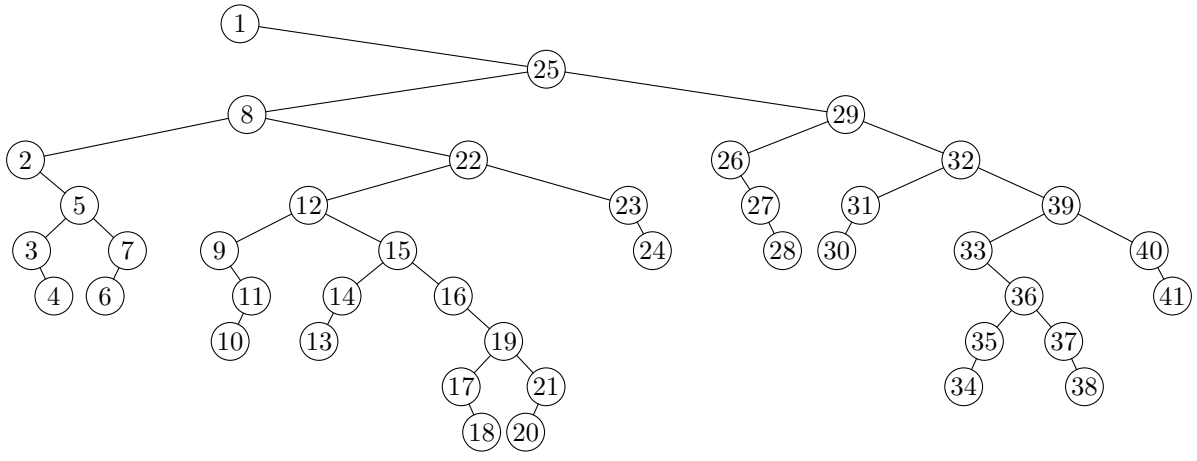
enqueue legt also nur Elemente auf den linken Stack, *dequeue* dreht dann bei Bedarf den Inhalt des linken Stacks um: Es entfernt sukzessive alle Element und legt sie auf den rechten Stack. Solange der rechte Stack jetzt noch Elemente enthält, nimmt *dequeue* sie schlicht von diesem.

Zeigen Sie, daß eine Operation auf dieser Queue amortisiert nur $O(1)$ Stack-Operationen benötigt. *Hinweis:* Benutzen Sie eine Potentialfunktion $\Phi\colon \mathbf{N} \to \mathbf{N}$, für die folgendes gilt:

1. $\Phi(i)$ ist das Potential *vor* der i-ten Operation, beginnend bei $i = 0$.
2. $\Phi(0) = 0$.
3. $\Phi(i+1) - \Phi(i) = O(1)$, wenn die i-te Operation *enqueue* ist.
4. Wenn die i-te Operation *dequeue* ist, werden nur $O(\Phi(i+1) - \Phi(i)) + O(1)$ Stack-Operationen benötigt.

Aufgabe H11 (10 Punkte)

Gegeben sei folgender Splay-Tree:



Es wird der Schlüssel 22 gelöscht. Wie sieht der Splay-Tree anschließend aus?