Algorithmen und Datenstrukturen
Musterlösung 6

Martin Avanzini <martin.avanzini@uibk.ac.at>
Thomas Bauereiß <thomas.bauereiss@uibk.ac.at>
Herbert Jordan <herbert@dps.uibk.ac.at>
René Thiemann <rene.thiemann@uibk.ac.at>

3. Mai, zur Besprechung am 10. Mai

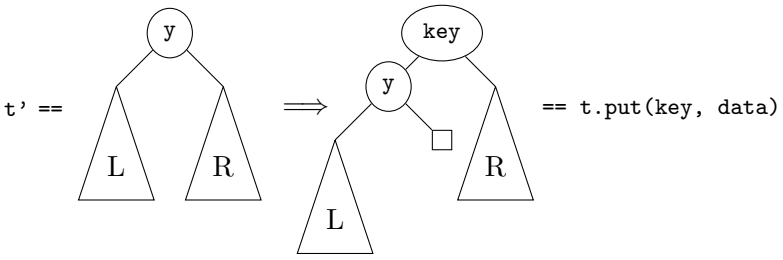
Aufgabe 1) Splay-Bäume

1. Siehe loesung1.1.pdf im Archiv.

2. (a) Da sich `splay(..)` nur aus Rotationsoperation welche die Suchbaumeigenschaft erhalten zusamensetzt gilt dies auch fuer `splay(..)`.

   (b) Sei `t` ein Splay-Baum und `t'` der Splay-Baum nach Anwendung von `splay(t.root,key)`. Aus Punkt 2a sehen wir dass `t'` die Suchbaumeigenschaft aufweist. Wir unterscheiden vier Fälle:

   - Fall 1 `t.root == t'.root == null`. Dann erfüllt `t.put(key,data)` (bestehend aus einem einzigen Knoten) die Suchbaumeigenschaft trivialerweise.
   - Fall 2 `t'.root.key == key`: In diesem Fall ist `t.put(key,data)` der Suchbaum `t'` mit Ausnahme des Attributes `t'.root.data`. Die Suchbaumeigenschaft bleibt also erhalten.
   - Fall 3 `t'.root.key < key`: In disem Fall führen wir folgende Umwandlung von `t'` durch:



   Sei L die Menge der Schlüssel vom linken und R die Menge der Schlüssel des rechten Teilbaumes in `t'`. Da `t'` die Suchbaumeigenschaft aufweist gilt `x < y` für alle `x` $\in L$. Wegen der Annahme `t'.root.key == y < key` gilt also `x < key` für alle `x` $\in L \cup \{$`y`$\}$.
   Nun nehmen wir an dass ein `x` $\in R$ existiert sodass `x < key` gilt und somit `t.put(key, data)` die Suchbaumeigenschaft nicht aufweist. Da `t'` die Suchbaumeigenschaft aufweist gilt also `y < x < key`. In diesem Fall hätte aber `splay(t.root,key)`

nicht den Knoten mit Schlüssel `y` an die Wurzel gebracht (Beachte dass die Operation `splay(t.root,key)` immer den unmittelbaren linken oder rechten Nachbarn von `key` an die Wurzel schiebt). Widerspruch! Somit gilt für alle `x` $\in R$, `key < x`.

- FALL 4 `t'.root.key > key`: Dieser Fall ist symmetrisch zu Fall 4.

Aufgabe 2) Hashing

1. Wenn gilt `h(key) == h(key')` für Hashing-Funktion `h`.
2. Wir unterscheiden zwischen Hashing mit Verkettung und Hashing mit offener Addressierung. Im ersteren werden Kollisionen durch Mehrfachbelegung von Feldern (Implementierung über verketteten Listen), im zweiteren werden Kollisionen durch Einfügen in Ausweichplätzen aufgelöst.
3. Siehe loesung2.3.pdf im Archiv.
4. Die Methoden `void put(K key, D data)` und `void remove(K key)` können wie folgt implementiert werden:

```
1  public void put(K key, D data) {
2     int hk = h(key);
3     for (int i=0; i<keys.length; i++) {
4        int pos = (hk + s(key, i)) % keys.length;
5        switch (status[pos]) {
6        case REMOVED:
7           status[pos] = OCCUPIED;
8           this.keys[pos] = key;
9           this.data[pos] = data;
10          // delete later occurrence of key (if any)
11          for (i=i+1; i<keys.length; i++) {
12             pos = (hk + s(key, i)) % keys.length;
13             switch (status[pos]) {
14             case FREE:
15                return;
16             case OCCUPIED:
17                if (this.keys[pos].equals(key)) {
18                   this.status[pos] = REMOVED;
19                   this.keys[pos] = null; // GC
20                   this.data[pos] = null; // GC
21                   return;
22                }
23             }
24          }
25          return;
26       case FREE:
27          status[pos] = OCCUPIED;
28          this.keys[pos] = key;
29          this.data[pos] = data;
30          return;
31       case OCCUPIED:
32          if (this.keys[pos].equals(key)) {
33             this.data[pos] = data;
34             return;
35          }
36       }
37    }
38    throw new RuntimeException("no space left");
39 }
```

```
40
41      public void remove(K key) {
42          int hk = h(key);
43          for (int i=0; i<keys.length; i++) {
44              int pos = (hk + s(key, i)) % keys.length;
45              switch (status[pos]) {
46              case FREE:
47                  return;
48              case OCCUPIED:
49                  if (this.keys[pos].equals(key)) {
50                      this.status[pos] = REMOVED;
51                      this.keys[pos] = null; // GC
52                      this.data[pos] = null; // GC
53                      return;
54                  }
55              }
56          }
57      }
```