(a, b)-Bäume

Wolfgang Mulzer

Wir schreiben einen Knoten v als $v = (w_1, k_1, w_2, k_2, \ldots, w_\ell, k_{\ell-1}, w_\ell)$. Dabei sind w_1, w_2, …, w_ℓ die Kindknoten von v, und k_1, …, $k_{\ell-1}$ die in v gespeicherten Schlüssel.

Es gilt $k_1 < k_2 < \cdots < k_{\ell-1}$. Alle Schlüssel im Teilbaum w_i sind kleiner als k_i und größer als k_{i-1}. Jeder innere Nichtwurzelknoten hat mindestens a und höchstens b Kinder. Die Wurzel hat mindestens 2 und höchstens b Kinder. Dementsprechend enthält jeder innere Nichtwurzelknoten mindestens $a - 1$ und höchstens $b - 1$ Schlüssel. Die Wurzel enthält mindestens einen und höchstens $b - 1$ Schlüssel.

Einfügen eines Schlüssels k in einen (a, b)-Baum (der Einfachheit halber lassen wir den Wert weg).

```
put(k)
  v <- root
  while v is not a leaf do
    write v as (w[1], k[1], w[2], k[2], ...., w[l-1], k[l-1], w[l])
    if k < k[1] then
      v <- w[1]
    else if k > k[l-1] then
      v <- w[l]
    else if there is an i such that k[i-1] < k < k[i] then
      v <- w[i]
    else // there is an i with k = k[i]
      return
  // now v is a leaf
  if v contains k
    return
  insert k into v
  // now we need to split the nodes that overflow
  while v contains b keys do
    split v into v', k', v'' // see below
    if v is not the root then
      let p be the parent of v
      insert v',k',v'' into p // see below
      v <- p
    else
      create a new root (v', k', v'')
      return
```

Die Splitoperation und das Einfügen in den Elternknoten sehen so aus: Sei $v = (w_1, k_1, \ldots, w_{b-1}, k_b, w_{b+1})$ und sei $m = (b + 1)/2$. Dann ist $v' = (w_1, k_1, \ldots, w_m)$, $v'' = (w_{m+1}, \ldots, w_{b+1})$ und $k' = k_m$. Das Einfügen in den Elternknoten geht so: $p = (\ldots, k_{r-1}, v, k_r, \ldots) \to (..., k_{r-1}, v', k', v'', k_r, \ldots)$

Löschen eines Schlüssels k aus einem (a, b)-Baum

```
remove(k)
  find the node v that contains k (as above)
  if v is not a leaf then
    find the successor or predecessor k' of k // it does not matter which one
                                              // we take
    replace k by k' in v
    let k <- k' and v <- the leaf that contains k'
  // now v is a leaf
  remove k from v
  // now we need to fix the underflow
  while v contains a-2 keys and v is not the root do
    if v has a sibling with >=a keys then
      borrow a key from the sibling // see below
      return
    else
      // both siblings contain a-1 keys
      merge v with a sibling // either left or right sibling is fine. See below
      v <- parent of v
  if v is the root and v contains no keys then
    remove v and let the new root be the child of v
```

Das Borgen von dem rechten Geschwisterknoten v' sieht so aus: Im Elternknoten stehe $(\dots, v, k'', v', \dots)$, wobei $v = (w_1, k_1, \dots, k_{a-2}, w_{a-1})$ und $v' = (w'_1, k'_1, \dots)$ mindestens a Schlüssel enthält. Dann $v \leftarrow (w_1, k_1, \dots, k_{a-2}, w_{a-1}, k'', w'_1)$ und $v' \leftarrow (w'_2, k'_2, \dots)$ und im Elternknoten ändert sich der Eintrag zu $(\dots, v, k'_1, v', \dots)$. In Worten: Der Schlüssel zwischen v und v' im Elternkoten wandert nach v, das linkeste Kind von v' wird das rechteste Kind von v und der linkeste Schlüssel von v' wandert in den Elternknoten zwischen v und v'. Das Borgen vom linken Geschwisterknoten geht analog.

Das Verschmelzen mit dem rechten Geschwisterknoten geht so: Im Elternknoten stehe $(\dots, v, k, v', \dots)$, wobei $v = (w_1, k_1, \dots, k_{a-2}, w_{a-1})$ und $v' = (w'_1, k'_1, \dots, k'_{a-1}, w'_a)$. Wir verschmelzen v und v' zu $v'' = (w_1, k_1, \dots, k_{a-2}, w_{a-1}, k, w'_1, k'_1, \dots, k'_{a-1}, w'_a)$. Der Elternknoten ändert sich folgendermaßen $(\dots, v, k, v', \dots) \rightarrow (\dots, v'', \dots)$. In Worten: v und v' werden aneinander gehängt, und der Schlüssel, der v und v' im Elternknoten trennt, wird nach unten gezogen. Das Verschmelzen mit dem linken Geschwisterknoten geht analog.

Beachte: Wenn der Elternknoten den Grad 2 hat, kann es durch Verschmelzen passieren, dass er hinterher keine Schlüssel mehr enthält. Handelt es sich beim Elternknoten um die Wurzel, so kann man sie danach löschen. Handelt es sich um einen inneren Knoten (wenn $a = 2$), verfährt man gemäß des obigen Algorithmus und füllt den Knoten vom Grad 1 durch Borgen oder Verschmelzen wieder auf.