Binäre Suchbäume

Wolfgang Mulzer

1 Darstellung

Ein binärer Suchbaum besteht aus Knoten. Ein Knoten `n` hat folgende Felder:

- `k`, `v`: Schlüssel und Wert
- `parent`: Elternknoten
- `left`, `right`: linkes und rechtes Kind

2 Nachschlagen

```
get(k)
  n <- root
  while n != NULL do
    if n.k == k then
      return n.v
    if k < n.k then
      n <- n.left
    else
      n <- n.right
  throw NoSuchElementException
```

3 Einfügen

```
put(k,v)
  if root == NULL then
    root <- new node for (k,v)
    return
  n <- root
  while true do
    if n.k == k then
      n.v <- v
      return
    if k < n.k then
      if n.left != NULL then
        n <- n.left
      else
        n.left <- new node for (k,v)
```

```
      return
    else
      if n.right != NULL then
        n <- n.right
      else
        n.right <- new node for (k,v)
        return
```

4 Löschen

```
remove(k)
  // lokalisiere den Knoten fuer k
  n <- root
  while n != NULL && n.k != k do
    if k < n.k then
      n <- n.left
    else
      n <- n.right
  if n == NULL then
    throw NoSuchElementException
  // wenn der Knoten n kein linkes oder kein rechtes Kind hat, loesche ihn direkt
  if n.left == NULL then
    // ersetze n durch das rechte Kind. Wir zeigen die noetigen Zeigeroperationen
    // nur einmal
    if n.right != NULL then
      n.right.parent <- n.parent
    if n.parent != NULL then
      if n.parent.left == n then
        n.parent.left <- n.right
      else
        n.parent.right <- n.right
    return
  else if n.right == NULL then
    replace n by its left child
    return
  // n hat beide Kinder -> finde den Vorgaenger von n
  m <- n.left
  while m.right != NULL do
    m <- m.right
  // nun ist m der Vorgaenger von n, m hat kein rechtes Kind
  replace m by its left child
  replace n by m
```