Binäre Heaps

Wolfgang Mulzer

1 Datenstruktur

Ein binärer Heap ist ein vollstängdiger binärer Baum mit der *Heap-Eigenschaft*. Jeder Knoten hat bis zu 2 Kinder. Alle Ebenen bis auf die letzte sind voll besetzt. Die letzte Ebene ist von links besetzt. Die Elemente stehen in den Knoten. Das Element in einem Elternknoten ist kleiner oder gleich den Elementen in den Kindknoten (Min-Heap Eigenschaft). Binäre Heaps werden oft als Arrays gespeichert. Sei `elements` das entsprechende Array. Die Wurzel des Heaps steht bei Index 0. Die Indizes der Kind- und der Elternknoten werden folgendermaßen berechnet.

```
left(i) = 2i + 1
right(i) = 2i + 2
parent(i) = floor((i+1)/2) - 1
```

2 Implementierung der Operationen

`size()` und `isEmpty()` werden mit Hilfe eine eigenen Zählers `size` implementiert. Für `findMin()` muss man nur `elements[0]` inspizieren. Für `deleteMin()` ersetzen wir die Wurzel durch `elements[size-1]` und dekrementieren `size`. Dann führen wir `bubble-down(0)` aus. Für `insert(x)` schreiben wir `x` an die Stelle `elements[size]`, inkrementieren `size` und führen `bubble-down(size-1)` aus.

3 Absenken

```
bubble-down(i)
  while( (left(i) < size && elements[left(i)] < elements[i]) ||
         (right(i) < size && elements[right(i)] < elements[i]) )
    nextI <- left(i)
    if right(i) < size && elements[right(i)] < elements[left(i)] then
      nextI++
    elements[i] <-> elements[nextI]
    i <- nextI
```

4 Anheben

```
bubble-up(i)
  while (i > 0 && elements[parent(i)] > elements[i])
    elements[parent(i)] <-> elements[i]
    i <- parent(i)
```