Script generated by TTT

Title: TÃ¤ubig: GAD (11.06.2013)

Date: Tue Jun 11 14:50:10 CEST 2013

Duration: 56:16 min

Pages: 16

Übersicht

Übersicht

Adressierbare Prioritätswarteschlangen

Zusätzliche Operationen für adressierbare Priority Queues:

- Handle insert(Element e): wie zuvor, gibt aber ein Handle (Referenz / Zeiger) auf das eingefügte Element zurück
- remove(Handle h): lösche Element spezifiziert durch Handle h
- decreaseKey(Handle h, int k): reduziere Schlüssel / Priorität des Elements auf Wert k (je nach Implementation evt. auch um Differenz k)
- M.merge(Q): $M = M \cup Q$; $Q = \emptyset$;

Prioritätswarteschlangen mit Listen

Priority Queue mittels unsortierter Liste:

- build($\{e_1, \ldots, e_n\}$): Zeit $O(n)$
- insert(Element e): Zeit $O(1)$
- min(), deleteMin(): Zeit $O(n)$

Priority Queue mittels sortierter Liste:

- build($\{e_1, \ldots, e_n\}$): Zeit $O(n \log n)$
- insert(Element e): Zeit $O(n)$
- min(), deleteMin(): Zeit $O(1)$

$\Rightarrow$ Bessere Struktur als eine Liste notwendig!

Übersicht

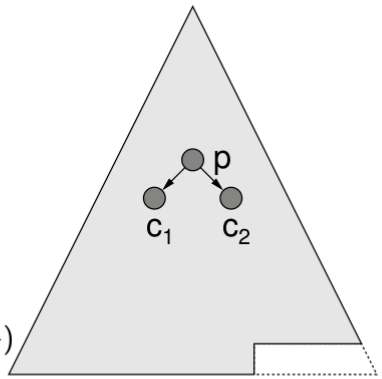
Binärer Heap

Idee: verwende Binärbaum

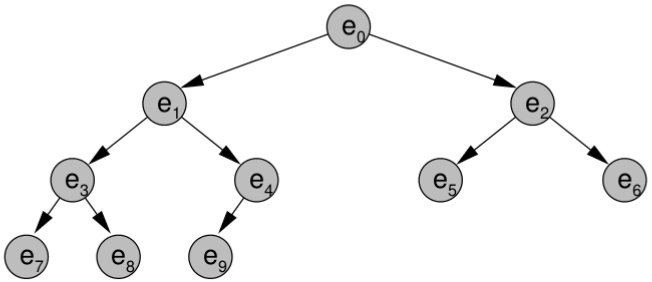
Bewahre zwei Invarianten:

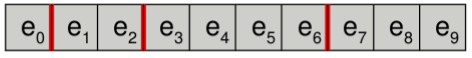
- Form-Invariante: fast vollständiger Binärbaum
- Heap-Invariante:

  $\text{prio}(p) \leq \min\{\text{prio}(c1), \text{prio}(c2)\}$



Binärer Heap als Feld





- Kinder von Knoten $H[i]$ in $H[2i+1]$ und $H[2i+2]$
- Form-Invariante: $H[0] \ldots H[n-1]$ besetzt
- Heap-Invariante: $H[i] \leq \min\{H[2i+1], H[2i+2]\}$

Binärer Heap als Feld

insert(e)

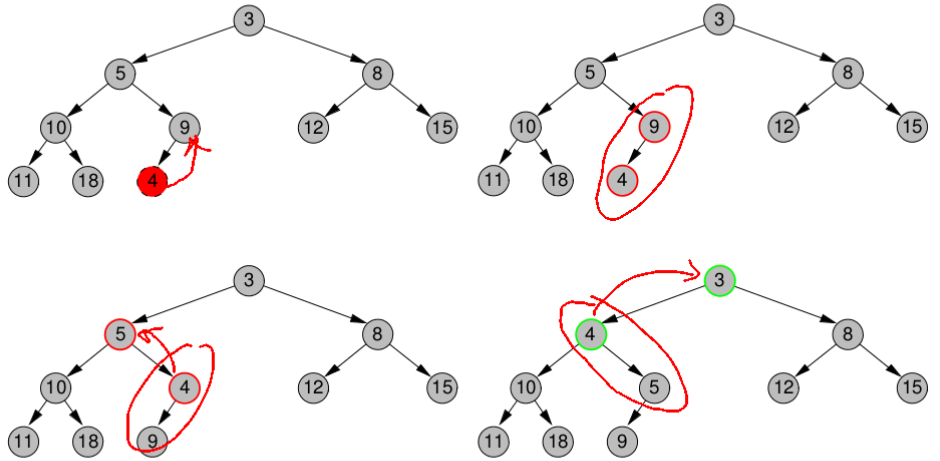
- Form-Invariante: $H[n] = e;$ siftUp(n); n++;
- Heap-Invariante:

  vertausche e mit seinem Vater bis
  $\text{prio}(H[\lfloor (k-1)/2 \rfloor]) \leq \text{prio}(e)$ für e in $H[k]$ (oder e in $H[0]$)

```
siftUp(i) {
  while (i > 0 ∧ prio(H[⌊(i − 1)/2⌋]) > prio(H[i])) {
    swap(H[i], H[⌊(i − 1)/2⌋]);
    i = (i − 1)/2;
  }
}
```

- Laufzeit: $O(\log n)$

Heap - siftUp()

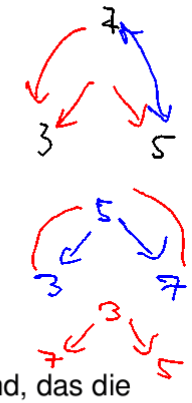


Binärer Heap als Feld

deleteMin()

- Form-Invariante:

```
e = H[0];
n − −;
H[0] = H[n];
siftDown(0);
return e;
```

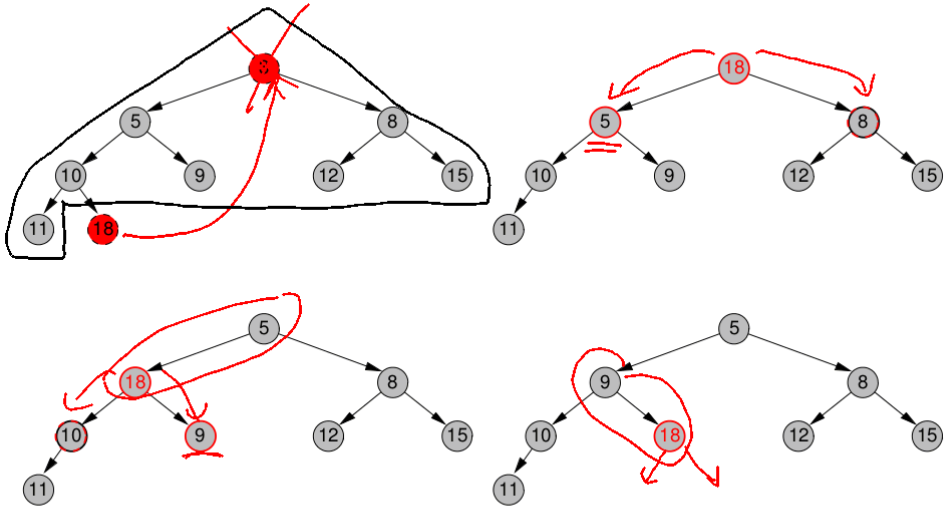


- Heap-Invariante: (siftDown)
  vertausche e (anfangs Element in $H[0]$) mit dem Kind, das die kleinere Priorität hat, bis e ein Blatt ist oder
  $\text{prio}(e) \leq \min\{\text{prio}(c_1(e)), \text{prio}(c_2(e))\}$.
- Laufzeit: $O(\log n)$

Binärer Heap als Feld

```
siftDown(i) {
  int m;
  while (2i + 1 < n) {
    if (2i + 2 ≥ n)
      m = 2i + 1;
    else
      if (prio(H[2i + 1]) < prio(H[2i + 2]))
        m = 2i + 1;
      else m = 2i + 2;
    if (prio(H[i]) ≤ prio(H[m]))
      return;
    swap(H[i], H[m]);
    i = m;
  }
}
```

Heap - siftDown()



Binärer Heap / Aufbau

build($\{e_0, \ldots, e_{n-1}\}$)

- naiv:

  Für alle $i \in \{0, \ldots, n-1\}$:
  insert(e_i)

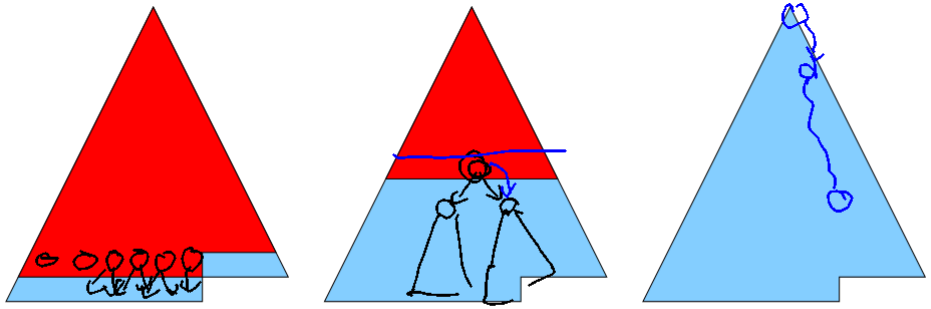
$\Rightarrow$ Laufzeit: $\Theta(n \log n)$

Binärer Heap / Aufbau

build($\{e_0, \ldots, e_{n-1}\}$)

effizient:

- Für alle $i \in \{0, \ldots, n-1\}$:
  $H[i] := e_i$.
- Für alle $i \in \left\{\left\lfloor \frac{n}{2} \right\rfloor - 1, \ldots, 0\right\}$:
  siftDown(i)



Binärer Heap / Aufbau

Laufzeit:

- $k = \lfloor \log n \rfloor$: Baumtiefe (gemessen in Kanten)
- siftDown-Kosten von Level ℓ aus proportional zur Resttiefe $(k - \ell)$
- Es gibt $\leq 2^\ell$ Knoten in Tiefe ℓ.

$$O\left(\sum_{0 \leq \ell < k} 2^\ell (k-\ell)\right) \subseteq O\left(2^k \sum_{0 \leq \ell < k} \frac{k-\ell}{2^{k-\ell}}\right) \subseteq O\left(2^k \sum_{j \geq 1} \frac{j}{2^j}\right) \subseteq O(n)$$

$$\begin{aligned} \sum_{j \geq 1} j \cdot 2^{-j} &= \sum_{j \geq 1} 2^{-j} + \sum_{j \geq 2} 2^{-j} + \sum_{j \geq 3} 2^{-j} + \ldots \\ &= 1 \cdot \sum_{j \geq 1} 2^{-j} + \frac{1}{2} \cdot \sum_{j \geq 1} 2^{-j} + \frac{1}{4} \cdot \sum_{j \geq 1} 2^{-j} + \ldots \\ &= (1 + 1/2 + 1/4 + \ldots) \sum_{j \geq 1} 2^{-j} = 2 \cdot 1 = 2 \end{aligned}$$

Laufzeiten des Binären Heaps

- min(): $O(1)$
- insert(e): $O(\log n)$
- deleteMin(): $O(\log n)$
- build($e_0, \ldots, e_{n-1}$): $O(n)$
- M.merge(Q): $\Theta(n)$

Adressen bzw. Feldindizes in array-basierten Binärheaps können nicht als Handles verwendet werden, da die Elemente bei den Operationen verschoben werden
$\Rightarrow$ ungeeignet als adressierbare PQs (kein remove bzw. decreaseKey)