Self-Adjusting Data Structures
Self-Adjusting Data Structures

Lists

Dictionaries
→ splay trees

Priority Queues
[C.A. Crane, Linear lists and priority queues as balanced binary trees, PhD thesis, Stanford University, 1972]
→ leftist heaps

→ skew heaps

[C. Okasaki, Alternatives to Two Classic Data Structures, Symposium on Computer Science Education, 162-165, 2005]
→ maxiphobix heaps

Okasaki: maxiphobix heaps are an alternative to leftist heaps ... but without the “magic”
Heaps (via Binary Heap-Ordered Trees)

MakeHeap, FindMin, Insert, **Meld**, DeleteMin

Leftist Heaps


Each node **distance to empty leaf**

**Inv.** Distance right child $\leq$ left child

$\Rightarrow$ rightmost path $\leq \lceil \log n+1 \rceil$ nodes

**Maxiphobic Heaps**

Meld $(x, y)$

$T_1 \ T_2 \ T_3$

$x < y$

$T_i$ largest size

$T_j$ two smallest

$T_k$

$\frac{2}{3}n$

Time O($\log_{3/2} n$)

Time O($\log n$)
Skew Heaps


- Heap ordered binary tree with *no* balance information
- MakeHeap, FindMin, Insert, Meld, DeleteMin
- **Meld** = merge rightmost paths + swap *all* siblings on merge path

\[ \text{Meld}(4, 2) = (4, 2) \]

\[ \nu \text{ heavy if } |T_v| > |T_{p(v)}|/2, \text{ otherwise light} \]
\[ \Rightarrow \text{ any path } \leq \log n \text{ light nodes} \]

\[ \Phi = \# \text{ heavy right children in tree} \]

\[ O(\log n) \text{ amortized Meld} \]

**Heavy** right child on merge path before meld \( \rightarrow \) replaced by **light** child
\[ \Rightarrow 1 \text{ potential released for heavy node} \]
\[ \Rightarrow \text{amortized cost } 2 \cdot \# \text{ light children on rightmost paths before meld} \]
Skew Heaps – O(1) time Meld


- **Meld** = Bottom-up merg of rightmost paths + swap *all* siblings on merge path

\[ \Phi = \# \text{ heavy right children in tree} + 2 \cdot \# \text{ light children on minor & major path} \]

**O(1) amortized Meld**

Heavy right child on merge path before meld $\rightarrow$ replaced by light child $\Rightarrow$ 1 potential released

Light nodes disappear from major paths (but might $\rightarrow$ heavy) $\Rightarrow$ $\geq 1$ potential released

4 and 5 become a heavy or light right children on major path $\Rightarrow$ potential increase by $\leq 4$

**O(log n) amortized DeleteMin**

Cutting root $\Rightarrow$ 2 new minor paths, i.e. $\leq 2 \cdot \log n$ new light children on minor & major paths
Splay Trees


- Binary search tree with **no** balance information
- \( \text{splay}(x) \) = rotate \( x \) to root (zig/zag, zig-zig/zag-zag, zig-zag/zag-zig)

Search (splay), Insert (splay predecessor+new root), Delete (splay+cut root+join), Join (splay max, link), Split (splay+unlink)
Splay Trees

- The access bounds of splay trees are amortized

  (1) $O(\log n)$
  (2) Static optimal
  (3) Static finger optimal
  (4) Working set optimal (proof requires dynamic change of weight)

- **Static optimality:** $\Phi = \sum_v \log |T_v|$