**Balanced Search Trees**

AVL trees
2-3 trees
weight-balanced trees

**Method 5:** Splay Trees (Sleator-Tarjan '85)

BST but without any color/size/height invariants!

"self-adjusting"

after inserting x,
do rotations along path from x to root

after deleting x,
similar

after query,
similar

querying changes DS!
Known results:
- $O((\log n))$ amortized time for insert/delete/query
- In static case, for any sequence of queries:
  Splay tree is optimal (up to constant factor) compared to any other BST methods

Open: "Dynamic Optimality Conjecture"
Is splay tree optimal for any sequence of queries & updates??

Is there any method that is within $O(1)$ factor of optimal?

Known: $O(\log \log n)$
Demaine et al., '04
Tango tree

Problem (Priority Queues)
build data structure for set $S$ of $n$ numbers to support
- find-min
- insert
- delete-min
- change-key \{ increase-key, decrease-key \}

Method 0: balanced search trees
Method 0 - balanced search trees

$O(\log n)$ time for all ops.

Method 1 - Standard binary heap

Heap property:

$\forall v$, $key(parent(v)) \leq key(v)$.

Perfectly balanced:

(no ptrs, $i \rightarrow 2i, 2i+1$)

find-min: $O(1)$

insert: $O(\log n)$

delete-min: $O(\log n)$

increase-key: $O(\log n)$

decrease-key: $O(\log n)$

Preprocess: (build-heap)

$O(n)$

Lower bound:

$n$ inserts/deletes require $\Omega(n \log n)$ time in comparison model by reduction from sorting.

$O\!$: faster insert? decrease-key?

Binomial heaps (Vuillemin '78):

find-min $O(1)$
Binomial Heaps (van Emde Boas '70):

- find-min $O(1)$
- insert $O(1)$
- delete-min $O(\log n)$

Fibonacci Heaps (Fredman-Tarjan '85):

- decrease-key $O(1)$ amortized

[apply: Dijkstra's algo, $n$ delete-mins, $m$ decrease-keys $\Rightarrow O(n \log n + m)$]

Other alternatives:

- Takata '03: "2-3 Heaps"
- C'09: "Quake Heaps"
- Hansen-Kaplan-Tarjan-Zwick '15: "Hollow Heaps"
- Brodal '96: worst case
- Pairing Heaps: simplified Fibonacci heaps
  but poorer guarantees

Tournament Tree Approach:
allow multiple tournament trees (forest)

\text{insert}(x): \quad // be lazy!
just create a new tree for \{x\}

\text{delete-min}():
X = \text{min of all the roots}
remove path of \text{x}'s nodes
whenever \exists 2 trees of same height
link them

\text{at end, } \leq log n \text{ trees}