OH TODAY: 1:15pm - 2:15pm

Letters

AVL Trees

- Balanced Binary Search Trees
- for any internal node $u$, the heights of the trees rooted at $u$'s children differ by at most 1.
- Height of an AVL tree with $n$ nodes is $O(\log n)$.

**Trinode restructurings.**

Algorithm to fix the height balance properly.

Suppose we insert/delete node $w$.

Walk from $w$ towards the root & let $t$ be the first node which “complains”.

its height differs by $\geq 2$.
Insert: y: taller child of z
x: "" y.

1. Let (a, b, c) be the (inorder) listing (sorted order) of the nodes x, y, z. Let T₀, T₁, T₂, T₃ be the left-to-right ordering of the four trees of x, y, z.

2. Replace z by b

3. a: left child of b & has sub-tree T₀, T₁

4. c: right "" b "" T₂, T₃
Delet
\( z \): first node that complains.

\( y \): taller child of \( z \)

\( x \): taller child of \( y \)

\( \text{if tie for } x \text{ then } x \text{ should be on the same side of } y \text{ as } y \text{ is to } z \).
### Diagram 1

- **Nodes:**
  - **T₀:** 30
  - **T₁:** 50, 55
  - **T₂:** 55
  - **T₃:**

- **Edge Labels:**
  - (1) 30
  - (2) 45
  - (3) 59

- **Relationships:**
  - (1) 30 → T₀
  - (2) 45 → T₁
  - (3) 59 → T₃

### Diagram 2

- **Nodes:**
  - 7
  - 2
  - 3
  - 5
  - 10
  - 11
  - 13
  - 15
  - 17
  - 18

- **Edge Labels:**
  - (1) Deletion

- **Relationships:**
  - (1) Deletion → 7

- **Path:**
  - 7 → 2 → 10 → 15 → 17 → 18
Olgen) restructurings in delete.

Linked lists

Skip lists

```
10 -> 25 -> 38 -> 50 -> 71 -> 86
```

```
L2

L1: 10 -> 25 -> 38 -> 55 -> 71 -> 86 -> 98
```
Search time: \( |L_2| + \frac{|L_1|}{|L_2|} \)

Search time is minimized when

\[
|L_2| = \left( \frac{|L_1|}{|L_2|} \right)^n = \frac{n}{|L_2|}
\]

\[\therefore \quad L_2^2 = n\]

\[L_2 = \sqrt{n}\]
Search time: \( L_3 + \frac{L_2}{L_3} + \frac{L_1}{L_2} \)

Search time is minimized when

\( L_3 = \frac{L_2}{L_3} \)
\( L_3^2 = L_2 \)

and

\( \frac{L_2}{L_3} = \frac{L_1}{L_2} = \frac{n}{L_2} \)
\( L_2 = n \cdot L_3 \)

\( L_2 = n \cdot \sqrt{L_2} \)

\( \therefore \frac{2}{2} = n \)

\( L_3 = n^{\frac{1}{3}} \)

\( L_2 = n^{\frac{2}{3}} \)
Search time: \( 3. n \)  

\[ \vdots \]

\( k \) levels

Search time: \( O(k \cdot \sqrt{n}) \)

When \( k = \log n \):

Search time: \( O\left(\log n \cdot \frac{\log n}{\log n}\right) \)

\[ = O\left(\log n \cdot \left(\frac{\log n}{\log n}\right)^{\frac{1}{2}}\right) \]

\[ = O\left(\log n\right) \]

Skip list of \( \log n \) levels.
$n \mod n \in L_1$

$\frac{n}{2} \in L_2$

$\frac{n}{4} \in L_3$

\[ \vdots \]

Insert $x$

Toss a fair coin until we get Tail.

$f := \#\text{ flips}$

Create a tower of size $f$ from $n$.

Our Skip list has levels $L_0, L_1, \ldots, L_e$.

Search ($x$)

$V_e \leftarrow \text{element with key } -\infty \in L_e$. 
for $i = l$ down to $1$ do

Follow the downlink from $v_i$ to $v_{i-1}$.

Follow the right links starting from $v_{i-1}$ until we come to a key $> x$.

$v_{i-1} \leq$ largest element in $L_{i-1}$ that is $\leq x$.

return $v_l$.  

Search (66)
Input $(x)$

if Search$(x) = n$ then
  return

$(v_1, v_{e-1}, \ldots, v_i) \in \text{elements in } L_e, L_{e-1}, \ldots, L, \text{ that we stopped during search.}$

$f \leftarrow \#\text{flips of a fair coin until we get Tail.}$

for $i \leftarrow 1 \text{ to } \min \{f, e\}$

  Insert $x$ into $L_i$

if $f > e$ then
  Create new list $L_{e+1}, L_{e+2}, \ldots, L_f$
  Add $(-\infty, -\infty)_{-\infty}$ to each list.

  Create list $L_{f+1}$ with only $-\infty$.

  $l \leftarrow \max \{e, f+1\}.$