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 $O(\log n)$

Fixing the height balance property:

Support we insert/delete node $w$.

Walk from $w$ towards the root.

Let $t$ be the first node which
Complains.

hts if it children differ by > 1.

y: tall child if \( y \neq z \)

x: "" "" y.

Trinode restructuring.

1. Let \((a, b, c)\) be the inorder listing of \(a, y, z\).
   - Let \(T_0, T_1, T_2, T_3\) be the four subtrees of \(a, y, z\).

2. Replace \(z\) by \(b\).

3. a: left child of \(b\) & has subtree \(T_0\) and \(T_1\).

4. c: right child of \(b\) & has subtree \(T_2\) and \(T_3\).
Running time: $O(1)$ (Treasure restructuring)

Insert: $O(gn)$

Delete:

$y$: fallen child of $x$

$x$: """""" $y$

(tie for $x$, choose $x$ on the same side of $y$ as $y$ is to $x$).
Linked list.

Search time: $|L_2| + \frac{|L_1|}{|L_1|}$
For the search time to be minimized, we have

\[ L_2 = \frac{L_1}{L_2} \quad \therefore \quad L^2 = L_1 = n \]

\[ \therefore \quad L_2 = \sqrt{n} \]

Search time \( T_{\text{sum}} = |L_3| + \frac{|L_2|}{|L_3|} + \frac{|L_1|}{|L_2|} \).
\[ l_3 = \frac{l_2}{l_3} \quad \text{and} \quad \frac{l_2}{l_3} = \frac{l_1}{l_2} = \frac{n}{l_2} \]

\[ \therefore l_3 = l_2 \quad \text{and} \quad l_2^2 = n \sqrt[3]{l_3} \]

\[ l_2^2 = n \cdot \sqrt[3]{l_2} \]

\[ \therefore l_2 = n^{2/3} \quad \Rightarrow \quad l_1 = n^{1/3} \]

Search time: \( O(3 \sqrt[3]{n}) \)

K list

Search time: \( O(K \sqrt[3]{n}) \)

Support \( l_k = \lg n \)

\( O(\lg n \sqrt[3]{n}) \)

\( O(n^{1/3}) \)
= \mathcal{O}(\log^2 n) \\
= \mathcal{O}(\log n).

When \( k = \log n \)

\( L_1 \) has \( n \) nodes

\( L_2 \) \quad \frac{n}{2} \quad \ldots

\( L_k \) \quad \frac{n}{2^k} \quad \ldots

\ldots

\underbrace{\text{node in each list}}_{\infty}

\underbrace{\text{node alone in the last list}}_{L_k}.
Search \( C_n \)

\[ v_e \leftarrow \infty \text{ node } \leq L \]

for \( i \leftarrow L \) downto 1 do

- Follow downlink from \( v_i \) to \( v_{i-1} \)

- Follow the right links from \( v_{i-1} \) until we come to a node \( > x \)

\( v_{i-1} : \text{rightmost element} \leq x \)

return \( v_i \)

\[ \text{Search } (66) \]
```
Insert (x)
    if Search (x) = x then
        return
(\{v_1, v_{i-1}, \ldots, v_i\} : elements found in Search.
f : \# flips of a fair coin until we get Tails.
for i \leftarrow 1 \text{ to } \min \{f, b \}
    Insert x into L_i
```
if \( f > l \) then

Create new list \( L_1, L_2, \ldots, L_f \)

Insert \(( -\infty, n)\) in each of the above lists

Create \( L_{f+1} \) that contains \(-\infty\).

\( l \leftarrow \max \{ l, f+1 \} \).