Lecture 12

Efficient dynamic search structures so far:
- 2-3-4 trees → Delete is tricky
- red-black trees → many cases
- AVL trees → some cases

Could you implement these?
Probably, with time...
But probably not without looking up some details
(a few years after this class)

Skip lists [Pugh 1989]
- a simple, efficient dynamic search structure
  that you'll never forget
- randomized: $O(\lg n)$ in expectation
  & with high probability
  $\implies$ very strong bound on tail of distribution
  $\implies$ “almost always” $O(\lg n)$

Experiment last year: implement skiplists
$\approx 10$ minutes for linked list $\{\text{not bad}\}$
$\approx 30$ minutes for skip list
$[\approx 60$ minutes debugging]
Starting from scratch: (from first principles)
  (initial focus: Search performance)
- simplest data structure:
  (sorted) linked list
- worst-case search: \( \Theta(n) \)
- how to improve?

Idea: 2 sorted linked lists
  - each element in one or both lists

**Example:** (New York City 7th Ave. subway line)

- \( 14, 23, 34, 42, 50, 59, 66, 72, 79, 86, 96, 103, 110, 116, 125 \)

- link between copies in both lists
- all elements in bottom list

**Search(\( x \)):**

- walk right in top list \( L_1 \)
  until going right would go too far
- walk down to bottom list \( L_2 \)
- walk right in \( L_2 \) until found (or not)
Two-lists data structure:
Which elements should go in top list $L_1$?
- in subway: "popular stations"
- here care about **worst-case performance**
- best to evenly space nodes in $L_1$
- but how many should be in $L_1$?

**Analysis**: search cost $\approx \frac{|L_1|}{n} + \frac{|L_2|/|L_1|}{n}$

- minimized (up to const. factors)
  - when $|L_1| = |L_2|/|L_1|$
  - i.e., $|L_1|^2 = |L_2| = n$ (everyone in $L_2$)
  - i.e., $|L_1| = \sqrt{n}$.
  - $\Rightarrow$ search cost $\approx \sqrt{n} + \frac{n}{\sqrt{n}} = 2\sqrt{n}$

**Structure:**

![Diagram of two-lists data structure]
More lists?
- 2 sorted lists $\Rightarrow 2 \sqrt{n}$
- 3 sorted lists $\Rightarrow 3 \sqrt[3]{n}$
- $k$ sorted lists $\Rightarrow k \sqrt[k]{n}$
- $\lg n$ sorted lists $\Rightarrow \lg n \cdot \sqrt{\frac{\lg n}{\lg n}} = \frac{2^{\frac{\lg n}{2}}}{2} = 2$

$\lg n$ lists: IDEAL SKIP LIST
- like a binary tree where key(x) = min in subtree
  (actually a level-linked $B^+$-tree)

Example: Search(72)

Skip list data structure maintains roughly this structure subject to updates
(insert/delete)
**Insert(x):**
- Search(x) to find where x fits in bottom list
- always insert x in bottom list

**INVARIENTS:**
1. bottom list contains all elements
2. each list contains subset of list below
- insert x into some list above. WHICH?

**IDEA:** Flip coin  
If heads: promote to next level up  
  flip again  
  with what probability should we promote?  
  \(1/2 \Rightarrow \text{fair coin} \)

\(\Rightarrow \) on average:
- \(1/2\) the elements promoted 0 levels  
- \(1/4\) the elements promoted 1 level  
- \(1/8\) the elements promoted 2 levels  
- etc.

\(\Rightarrow \) approximately balanced?

**EXERCISE:** Let's build a skip list with a real coin!

**Minor change:**  
Add special \(-\infty\) value to every list
  \(\Rightarrow\) can use same Search algorithm

**Delete(x):** just remove x from all lists
Intuitively, skip lists are pretty good on average. Claim they are really, really good, almost always.

**Theorem:** With high probability, every search in an n-element skip list costs $O(\log n)$.

With high probability (w.h.p.),

- informal definition: event $E$ occurs w.h.p. if, for any $\alpha > 1$, there is an appropriate choice of constants for which $E$ occurs with probability at least $1 - O(\frac{1}{n^\alpha})$.

- in fact, constant in $O(\log n)$ depends on $\alpha$.

- formal definition: parameterized event $E_\alpha$ occurs w.h.p. if, for any $\alpha > 1$, there is a $C_\alpha$ such that $E_\alpha$ occurs with probability $\geq 1 - C_\alpha / n^\alpha$.

- idea: can make error probability $O(1/n^\alpha)$ very small by setting $\alpha$ large, e.g., 100.

- almost certainly, bound remains true for entire execution of polynomial-time algorithm.

Boole's inequality/union bound: For events $E_1, E_2, \ldots, E_k$

$$\Pr[E_1 \cup E_2 \cup \cdots \cup E_k] \leq \Pr[E_1] + \Pr[E_2] + \cdots + \Pr[E_k]$$

$\Rightarrow$ if $k = n^{O(1)}$ & $E_1, E_2, \ldots, E_k$ occur w.h.p. then $E_1 \cap E_2 \cap \cdots \cap E_k$ occurs w.h.p.
Analysis: warmup

**Lemma:** With high probability, number of levels in an n-element skip list is $O(\log n)$

**Proof:** Error probability for $\leq c \log n$ levels

$$= \Pr \{ \text{levels} \geq c \log n \}$$

$$\leq n \cdot \Pr \{ \text{element } x \text{ promoted } \geq c \log n \text{ times} \}$$

by Boole’s inequality

$$= n \cdot \left(\frac{1}{2}\right)^{c \log n}$$

$$= n^{1/c}$$

$$= n^{c^{-1}}$$

$$= 1/n^\alpha$$ for $\alpha = c^{-1}$

polyonmially small

- can make $\alpha$ arbitrary large
  - by choosing $c$ in $O(\log n)$ large enough. □
Theorem: With high probability, any search in an \( n \)-element skip list costs \( O(\lg n) \).

Cool idea: analyze search backwards - leaf to root
- search starts [ends] at leaf
- at each node visited:
  - if node wasn't promoted higher (tails here) then we go [came from] left
  - if node was promoted higher (heads here) then we go [came from] up
- search stops [starts] at root (or -\( \infty \))

Proof of theorem:
- Search makes "up" & "left" moves until it reaches root (or -\( \infty \))
- number of "up" moves \( \leq \) number of levels
  \( \leq c \lg n \) w.h.p. (lemma)

\( \Rightarrow \) w.h.p., number of moves
  \( \leq \) number of coin flips to get \( c \lg n \) heads
Claim: Number of coin flips till $c \log n$ heads = $\Theta(c \log n)$ with high probability

Proof: Obviously $\geq c \log n = \Omega(c \log n)$
Let's say we flip $d \log n$ times.
When are there $\geq c \log n$ heads?
(C later generalize to arbitrary values of $0$)
\# configurations with $k$ heads = $\binom{d \log n}{k}$
\# configurations with $< k$ heads = $\sum_{i=0}^{k-1} \binom{d \log n}{i}$

For $i \leq \frac{1}{3}d \log n$, \( \binom{d \log n}{i} = \frac{d \log n - i + 1}{i} \binom{d \log n}{i-1} \)

\[ \geq 2 \]

\[ \sum_{i=0}^{\frac{1}{3}d \log n} \binom{d \log n}{i} \leq \binom{d \log n}{d \log n + 1} / 2 \]
\[ \leq \binom{d \log n}{d \log n + 1} / 2 (d/3 - 1) \log n \]
\[ \leq \frac{1}{2} \sum_{i=0}^{d \log n} \binom{d \log n}{i} / 2 (d/3 - 1) \log n \]
\[ \text{all configurations} = 2^{d \log n} \]

$\Rightarrow \Pr[\# \text{ heads } < c \log n] \leq \frac{1}{2} (d/3 - 1) \log n$

$= \frac{1}{n} (d/3 - 1) c$

$= \frac{1}{n^\alpha}$ where $\alpha = (d/3 - 1) c$

Key: $\alpha \to \infty$ as $d \to \infty$, for any $c > 0$

$\Rightarrow$ set $d$ (const. in $O(\log n)$ bound) large enough

for desired value of $\alpha$