Lecture Notes on Skip Lists

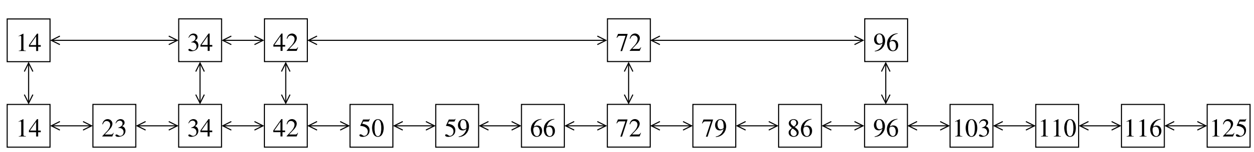
Lecture 12 — October 26, 2005

Erik Demaine

- Balanced tree structures we know at this point: red-black trees, B-trees, treaps.
- Could you implement them right now? Probably, with time… but without looking up any details?
- Skip lists are a simple randomized structure you'll never forget.

Starting from scratch

- Initial goal: *just searches* — ignore updates (Insert/Delete) for now
- Simplest data structure: linked list
- Sorted linked list: $\Theta(n)$ time
- 2 sorted linked lists:
  - Each element can appear in 1 or both lists
  - How to speed up search?
  - **Idea:** Express and local subway lines
  - **Example:** $\boxed{14}$, 23, $\boxed{34}$, $\boxed{42}$, 50, 59, 66, $\boxed{72}$, 79, 86, $\boxed{96}$, 103, 110, 116, 125
    (What is this sequence?)
  - $\boxed{\text{Boxed}}$ values are "express" stops; others are normal stops
  - Can quickly jump from express stop to next express stop, or from any stop to next normal stop
  - Represented as two linked lists, one for express stops and one for all stops:



  - Every element is in bottom linked list (L_2); some elements also in top linked list (L_1)
  - Link equal elements between the two levels
  - To search, first search in L_1 until about to go too far, then go down and search in L_2

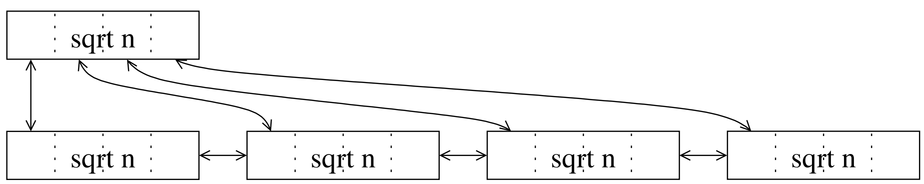
- Cost:

$$|L_1| + \frac{|L_2|}{|L_1|} = |L_1| + \frac{n}{|L_1|}$$

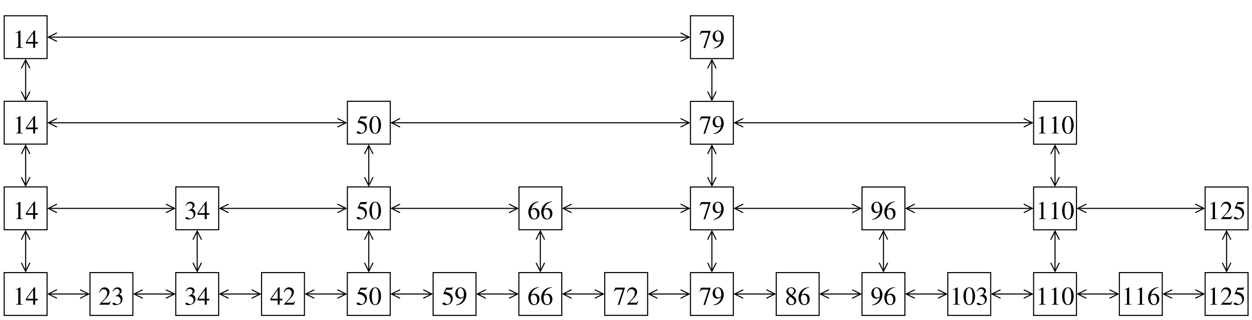
- Minimized when

$$\begin{aligned} & |L_1| = \frac{n}{|L_1|} \\ \Rightarrow\ & |L_1|^2 = n \\ \Rightarrow\ & |L_1| = \sqrt{n} \\ \Rightarrow\ & \text{search cost} = 2\sqrt{n} \end{aligned}$$

- Resulting 2-level structure:



- 3 linked lists: $3 \cdot \sqrt[3]{n}$
- k linked lists: $k \cdot \sqrt[k]{n}$
- $\lg n$ linked lists: $\lg n \cdot \sqrt[\lg n]{n} = \lg n \cdot \underbrace{n^{1/\lg n}}_{=2} = \Theta(\lg n)$
  - Becomes like a binary tree:



  - (In fact, a level-linked B$^+$-tree; see Problem Set 5.)
  - **Example:** Search for 72
    * Level 1: 14 too small, 79 too big; go down 14
    * Level 2: 14 too small, 50 too small, 79 too big; go down 50
    * Level 3: 50 too small, 66 too small, 79 too big; go down 66
    * Level 4: 66 too small, 72 spot on

Insert

- New element should certainly be added to bottommost level
  (Invariant: Bottommost list contains all elements)
- Which other lists should it be added to?
  (Is this the entire balance issue all over again?)
- **Idea:** Flip a coin
  - With what probability should it go to the next level?
  - To mimic a balanced binary tree, we'd like half of the elements to advance to the next-to-bottommost level
  - So, when you insert an element, flip a fair coin
  - If heads: add element to next level up, and flip another coin (repeat)
- Thus, on average:
  - $1/2$ the elements go up 1 level
  - $1/4$ the elements go up 2 levels
  - $1/8$ the elements go up 3 levels
  - Etc.
- Thus, "approximately even"

Example

- Get out a real coin and try an example
- You should put a special value $-\infty$ at the beginning of each list, and always promote this special value to the highest level of promotion
- This forces the leftmost element to be present in every list, which is necessary for searching

. . . many coins are flipped . . .
(Isn't this easy?)

- The result is a skip list.
- It probably isn't as balanced as the ideal configurations drawn above.
- It's clearly good on average.
- Claim it's really really good, almost always.

Analysis: Claim of With High Probability

- **Theorem:** *With high probability*, *every* search costs $O(\lg n)$ in a skip list with n elements
- What do we need to do to prove this? [Calculate the probability, and show that it's high!]
- We need to define the notion of "with high probability"; this is a powerful technical notion, used throughout randomized algorithms
- **Informal definition:** An event occurs ***with high probability*** if, for any $\alpha \geq 1$, there is an appropriate choice of constants for which E occurs with probability at least $1 - O(1/n^\alpha)$
- In reality, the constant hidden within $O(\lg n)$ in the theorem statement actually depends on c.
- **Precise definition:** A (parameterized) event E_α occurs ***with high probability*** if, for any $\alpha \geq 1$, E_α occurs with probability at least $1 - c_\alpha/n^\alpha$, where c_α is a "constant" depending only on α.
- The term $O(1/n^\alpha)$ or more precisely c_α/n^α is called the ***error probability***
- The idea is that the error probability can be made very very very small by setting α to something big, e.g., 100

Analysis: Warmup

- **Lemma:** With high probability, skip list with n elements has $O(\lg n)$ levels
- (In fact, the number of levels is $\Theta(\log n)$, but we only need an upper bound.)
- **Proof:**
  - Pr{element x is in more than $c \lg n$ levels} $= 1/2^{c \lg n} = 1/n^c$
  - Recall Boole's inequality / union bound:

$$\Pr\{E_1 \cup E_2 \cup \cdots \cup E_k\} \leq \Pr\{E_1\} + \Pr\{E_2\} + \cdots + \Pr\{E_k\}$$

  - Applying this inequality:
    Pr{any element is in more than $c \lg n$ levels} $\leq n \cdot 1/n^c = 1/n^{c-1}$
  - Thus, error probability is polynomially small and exponent ($\alpha = c - 1$) can be made arbitrarily large by appropriate choice of constant in level bound of $O(\lg n)$

Analysis: Proof of Theorem

- **Cool idea:** Analyze search backwards—from leaf to root
  - Search starts at leaf (element in bottommost level)
  - At each node visited:
    - If node wasn't promoted higher (got TAILS here), then we go [came from] left
    - If node was promoted higher (got HEADS here), then we go [came from] up
  - Search stops at root of tree
- Know height is $O(\lg n)$ with high probability; say it's $c \lg n$
- Thus, the number of "up" moves is at most $c \lg n$ with high probability
- Thus, search cost is at most the following quantity:

  How many times do we need to flip a coin to get $c \lg n$ heads?
- Intuitively, $\Theta(\lg n)$

Analysis: Coin Flipping

- **Claim:** Number of flips till $c \lg n$ heads is $\Theta(\lg n)$ with high probability
- Again, constant in $\Theta(\lg n)$ bound will depend on α
- **Proof of claim:**
  - Say we make $10c \lg n$ flips
  - When are there at least $c \lg n$ heads?
  - $\Pr\{\text{exactly } c \lg n \text{ heads}\} = \underbrace{\binom{10c \lg n}{c \lg n}}_{\substack{\text{orders} \\ \text{HHHTTT vs. HTHTHT}}} \cdot \underbrace{\left(\frac{1}{2}\right)^{c \lg n}}_{\text{heads}} \cdot \underbrace{\left(\frac{1}{2}\right)^{9c \lg n}}_{\text{tails}}$
  - $\Pr\{\text{at most } c \lg n \text{ heads}\} \le \underbrace{\binom{10c \lg n}{c \lg n}}_{\substack{\text{overestimate} \\ \text{on orders}}} \cdot \underbrace{\left(\frac{1}{2}\right)^{9c \lg n}}_{\text{tails}}$
  - Recall bounds on $\binom{y}{x}$:

$$\left(\frac{y}{x}\right)^x \le \binom{y}{x} \le \left(e\,\frac{y}{x}\right)^x$$

- Applying this formula to the previous equation:

$$
\begin{aligned}
\Pr\{\text{at most } c \lg n \text{ heads}\} &\le \binom{10c\lg n}{c\lg n}\left(\frac{1}{2}\right)^{9c\lg n} \\
&\le \left(\frac{e\cdot 10c\lg n}{c\lg n}\right)^{c\lg n}\cdot\left(\frac{1}{2}\right)^{9c\lg n} \\
&= (10e)^{c\lg n}\cdot\left(\frac{1}{2}\right)^{9c\lg n} \\
&= 2^{\lg(10e)\cdot c\lg n}\cdot\left(\frac{1}{2}\right)^{9c\lg n} \\
&= 2^{(\lg(10e)-9)c\lg n} \\
&= 2^{-\alpha\lg n} \\
&= 1/n^{\alpha}
\end{aligned}
$$

- The point here is that, as $10 \to \infty$, $\alpha = 9 - \lg(10e) \to \infty$, independent of (for all) c

• End of proof of claim and theorem

Acknowledgments

This lecture is based on discussions with Michael Bender at SUNY Stony Brook.