L2: Skip Lists

- Think TreeMap in Java
- Maps Key → Value. The keys are ordered. Support add / remove / update / lookup
- Goal: for most operations, take $O(\log n)$ time or faster.

Applications

- Table Mapping / DB indexes. Make a collection searchable quickly.
- Best-first Heuristics. ① Pack items into bins (max cap: C). A decent heuristics is to to find the best-fit bin (i.e, least remaining cap but enough for the item). Use ordered map to locate the best bin, reduce size, then put it back to the pool.
② malloc: find the best free block to use.

The Theme: approximating a perfectly balanced struct. in prob. / deterministically / on average.

Q: What comes to your mind when you want an ordered map?

* Balanced BST
* B-Tree

} not easy to implement

* Linked List ← any good thing?

  - the smallest & the small elts are quick to access
  - same for the largest elts

by contrast: the whole depth for trees.

Today's goal: Supercharging the linked list D/S.

Introduction to Algorithms — March 18, 2004
Massachusetts Institute of Technology — 6.046J/18.410J
Professors Erik Demaine and Shafi Goldwasser — Handout 14

Lecture Notes on Skip Lists

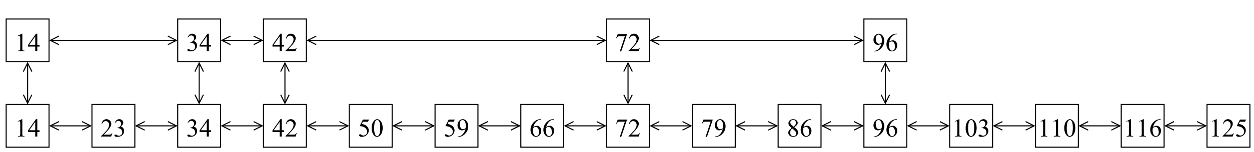
Lecture 12 — March 18, 2004

Erik Demaine

- ~~Balanced tree structures we know at this point: B-trees, red-black trees, treaps.~~
- Could you implement ~~them~~ balanced trees right now? Probably, with time... but without looking up any details in a book?
- 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 linked list 2 (LL2); some elements also in linked list 1 (LL1)
  - Link equal elements between the two levels
  - To search, first search in LL1 until about to go too far, then go down and search in LL2

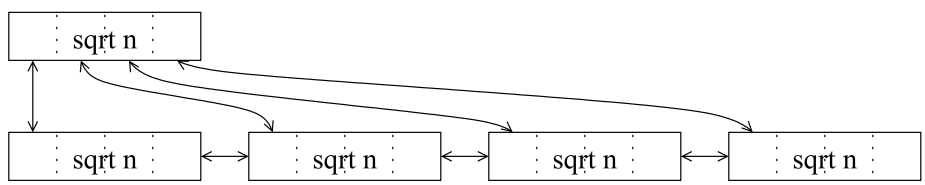
– Cost:

$$\text{len}(\text{LL1}) + \frac{\text{len}(\text{LL2})}{\text{len}(\text{LL1})} = \text{len}(\text{LL1}) + \frac{n}{\text{len}(\text{LL1})}$$

– Minimized when

$$\begin{aligned} & \text{len}(\text{LL1}) = \frac{n}{\text{len}(\text{LL1})} \\ \Rightarrow\ & \text{len}(\text{LL1})^2 = n \\ \Rightarrow\ & \text{len}(\text{LL1}) = \sqrt{n} \\ \Rightarrow\ & \text{search cost} = 2\sqrt{n} \end{aligned}$$

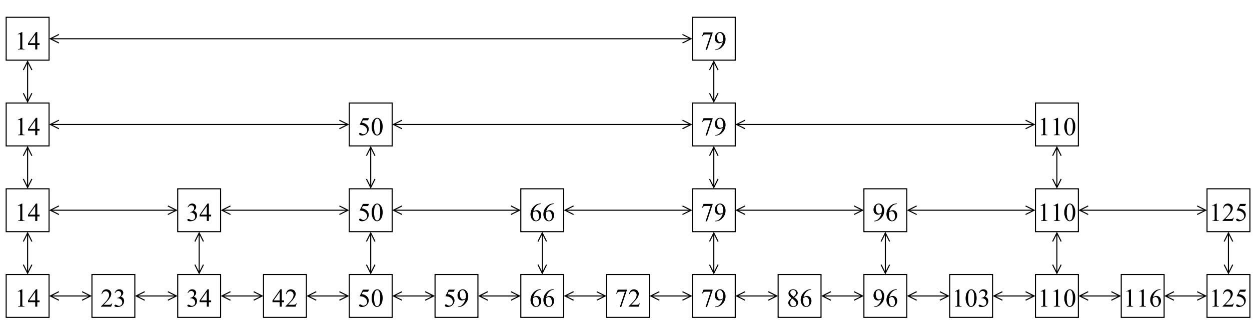
– Resulting 2-level structure:



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

✓ perfect.

– Becomes like a binary tree:



– **Example:** Search for 72

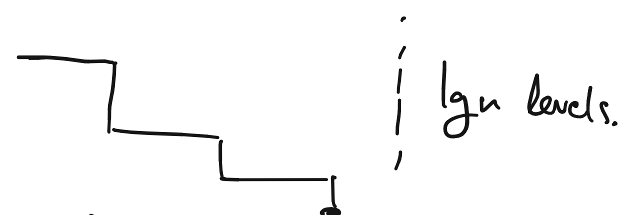
* Level 1: 14 too small, 79 too big; go down 14
* Level 2: 14 too small, 50 too small, 79 too big; go down 50
* Level 3: 50 too small, 66 too small, 79 too big; go down 66
* Level 4: 66 too small, 72 spot on

Maintaining this exactly is too rigid. Relax by flipping coins.

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"

Expected Search Cost.



- work backwards.
- $E[\text{horizontal motion/level}] = 1/\tfrac{1}{2} = 2$.
- $\#$ levels $= \lg n \Rightarrow E[\text{cost}] = 2\lg n$.

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 $\Theta(\lg n)$ 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 $\Theta(\lg n)$ in the theorem statement actually depends on ~~c~~ α
- **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:

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

  - 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 wasn't promoted higher (got HEADS here), then we go [came from] top
  - 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?

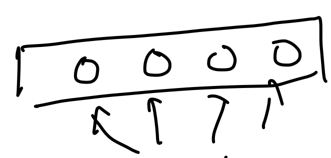
- Intuitively, $\Theta(\lg n)$

Analysis: Coin Flipping

$10c \cdot \log n$

- **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?

$c \log_2 n$ spots



  - $\Pr[\text{exactly } c \lg n \text{ heads}] = \underbrace{\binom{10c \lg n}{c \lg n}}_{\substack{\text{orders} \\ \text{HHHTTT vs. HTHTHT}}} \cdot \underbrace{\left(\frac{1}{2}\right)^{c \lg n}}_{\text{heads}} \cdot \underbrace{\left(\frac{1}{2}\right)^{9c \lg n}}_{\text{tails}}$
  - $\Pr[\text{at most } c \lg n \text{ heads}] \leq \underbrace{\binom{10c \lg n}{c \lg n}}_{\substack{\text{overestimate} \\ \text{on orders}}} \cdot \underbrace{\left(\frac{1}{2}\right)^{9c \lg n}}_{\text{tails}}$

heads or tails

must be heads

  - Recall bounds on $\binom{y}{x}$:

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

  [Michael's "deathbed" formula: even on your deathbed, if someone gives you a binomial and says "simplify", you should know this!]

$(\beta-1)c$

– Applying this formula to the previous equation:

$$\binom{\beta \cdot c\lg n}{c\lg n}$$

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

$\beta(\log_2(\beta e) - \beta - 1)c$

β

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

- End of proof of claim and theorem

Acknowledgments

The mysterious "Michael" is Michael Bender at SUNY Stony Brook. This lecture is based on discussions with him.

Activity: Thought experiment — for what kind of queries / inserts would the skip list be faster than, say, an RB tree?

- finger searching: position d — can you think of a scheme to speed this up?

into A1:
- implement the skip list with finger searching
- compare against built-in ordered maps.