Lecture Notes on Skip Lists
Lecture 11 — October 21, 2002
Erik Demaine

- Balanced tree structures we know at this point: B-trees, red-black trees, treaps.
- Could you implement them 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: 14, 23, 34, 42, 50, 59, 66, 72, 79, 86, 96, 103, 110, 116, 125
    (What is this sequence?)
  - 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:

```
14   34   42   72   96
  14  23   34   42   50   59   66   72   79   86   96   103   110   116   125
```
- 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:
\[
\frac{\text{len}(LL1)}{\text{len}(LL1)} + \frac{\text{len}(LL2)}{\text{len}(LL1)} = \frac{\text{len}(LL1)}{\text{len}(LL1)} + \frac{n}{\text{len}(LL1)}
\]

Minimized when
\[
\frac{\text{len}(LL1)}{\text{len}(LL1)} = \frac{n}{\text{len}(LL1)}
\Rightarrow \text{len}(LL1)^2 = n
\Rightarrow \text{len}(LL1) = \sqrt{n}
\Rightarrow \text{search cost} = 2\sqrt{n}
\]

Resulting 2-level structure:

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

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
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 $\Theta(\log n)$ 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 $\Theta(\log n)$ in the theorem statement actually depends on $c$.

- **Precise definition:** A (parameterized) event $E_\alpha$ occurs with high probability if, for any $\alpha \geq 1$, $E_\alpha$ occurs with probability at least $1 - c_\alpha/n^\alpha$, where $c_\alpha$ is a “constant” depending only on $\alpha$.

- The term $O(1/n^\alpha)$ or more precisely $c_\alpha/n^\alpha$ is called the **error probability**

- The idea is that the error probability can be made very very very small by setting $\alpha$ to something big, e.g., 100

Analysis: Warmup

- **Lemma:** With high probability, skip list with $n$ elements has $O(\log n)$ levels

- (In fact, the number of levels is $\Theta(\log n)$, but we only need an upper bound.)

- **Proof:**
  - $\Pr[\text{element } x \text{ is in more than } c \log n \text{ levels}] = 1/2^{c \log n} = 1/n^c$
  - 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:
    $\Pr[\text{any element is in more than } c \log n \text{ levels}] \leq n \cdot 1/n^c = 1/n^{c-1}$
  - 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 $O(\log n)$
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

- **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 $\alpha$
- **Proof of claim:**
  - Say we make $10c\lg n$ flips
  - When are there at least $c\lg n$ heads?
  - $Pr[\text{exactly } c\lg n \text{ heads}] = \binom{10c\lg n}{c\lg n} \cdot \left(\frac{1}{2}\right)^{c\lg n} \cdot \left(\frac{1}{2}\right)^{9c\lg n}$
  - $Pr[\text{at most } c\lg n \text{ heads}] = \binom{10c\lg n}{c\lg n} \cdot \left(\frac{1}{2}\right)^{c\lg n}$
  - Recall bounds on $\left(\frac{y}{x}\right)^x$:
    $\left(\frac{y}{x}\right)^x \leq \left(\frac{y}{x}\right) \leq \left(e \cdot \frac{y}{x}\right)^x$

[Michael’s “deathbed” formula: even on your deathbed, if someone gives you a binomial and says “simplify”, you should know this!]
– Applying this formula to the previous equation:

\[
\text{Pr[at most } c \lg n \text{ heads]} \leq \left( \frac{10c \lg n}{c \lg n} \right)^{9c \lg n} \cdot \left( \frac{1}{2} \right)^{9c \lg n}
\leq \left( \frac{e \cdot 10c \lg n}{c \lg n} \right)^{c \lg n} \cdot \left( \frac{1}{2} \right)^{9c \lg n}
= (10e)^{c \lg n} \cdot \left( \frac{1}{2} \right)^{9c \lg n}
= 2^{\lg(10e)c \lg n} \cdot \left( \frac{1}{2} \right)^{9c \lg n}
= 2^{(\lg(10e) - 9)c \lg n}
= 2^{-\alpha \lg n}
= 1/n^\alpha
\]

– 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**

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