Today: Skip Lists
- one list, two lists, ...
- insert algorithm
- with-high-probability analysis

Skip lists [Pugh 1989]
- a simple, efficient dynamic search structure
  you'll never forget

Experiment in 2005: implement skip lists
≈ 10 minutes for linked list
≈ 30 minutes for skip list
[≈ 60 minutes debugging] not bad

- randomized: O(lg n) time/op. w.h.p.

With (polynomially) high probability: $\geq 1 - O\left(\frac{1}{n^a}\right)$
for desired constant $a > 0$

$\Rightarrow$ if we do polynomially many operations, w.h.p. they all take $O(lg n)$:
- $Pr[\exists \text{any fail}] \leq \sum Pr[\text{fail}]$ (Union Bound)
  - $\leq n^k \cdot O\left(\frac{1}{n^a}\right)$
  - $\leq O(n^{\alpha'})$
(set $\alpha = \alpha' + k$)
One sorted linked list: (starting from scratch)
- worst-case search: \(O(n)\)
- how to improve?

Example:

\[
\begin{align*}
L_0: &\quad 14 & 23 & 28 & 34 & 42 & 50 & 59 & 66 & 72 & 79 & 86 & 96 & 103 & 110 & \cdots \\
L_1: &\quad 14 & 34 & 42 & 72 & 96 & \cdots \\
L_1: &\quad 14 & 23 & 28 & 34 & 42 & 50 & 59 & 66 & 72 & 79 & 86 & 96 & \cdots 
\end{align*}
\]

Search(59):

\[\text{too far}\]

(New York City 7th Ave. Subway Line)

Two sorted lists:
- \(L_0\) stores all elements (as before)
- \(L_1\) stores some elements, including first
- link between copies of same element

Search(\(x\)):
- express: go right in \(L_1\) until going right would go too far
- transfer: go down to \(L_0\)
- local: go right in \(L_0\) until find \(x\) or successor
Which elements should go in \( L_1 \)?
- in subway: “popular stations”
- here want to minimize worst-case performance
- best to evenly space nodes in \( L_1 \)
- but how many should be in \( L_1 \)?

**Analysis:** search cost \( \approx \frac{|L_1| + |L_0|/|L_1|}{\text{express cost}} \)
- minimized (up to constant factors)
  when \(|L_1| = |L_0|/|L_1|\)
  i.e. \(|L_1|^2 = |L_0| = n\) (everyone is in \( L_2 \))
  i.e. \(|L_1| = \sqrt{n}\)
  \(\Rightarrow\) search cost \( \approx \sqrt{n} + \frac{n}{\sqrt{n}} = 2\sqrt{n} \)

**Structure:**

```
  3 \sqrt{n}   \rightarrow \rightarrow \rightarrow ...
  \sqrt{n} \quad \overrightarrow{\downarrow} \quad \overrightarrow{\uparrow}
```

**More lists:**
- 2 sorted lists \( \Rightarrow 2\sqrt{n} \)
- 3 sorted lists \( \Rightarrow 3 \cdot 3\sqrt{n} \)
- \( k \) sorted lists \( \Rightarrow k \cdot k\sqrt{n} \)
- \( \log n \) sorted lists \( \Rightarrow \log n \cdot \frac{\sqrt{n}}{\log n} = \log n \cdot \frac{\sqrt{n}}{2} = 2 \log n \)
**lg n lists**: IDEAL SKIP LIST
- like a binary tree where \( \text{key}(x) = \min(\text{subtree}) \)
  (actually a level-linked \( B^{+} \)-tree) (data in leaves)
- bottom list \( L_0 \) contains all elements
- each list \( L_i \) contains subset of list \( L_{i-1} \) below
- first element in all lists
- vertical linked list of all copies of an element

**Search(72):**
- start at top left (first node of top list)
- until we fall below bottom list:
  if going right would go too far: go down
else: go right
Skip list maintains roughly this idealized structure using randomization during inserts:
- **idea:** Insert(x) always adds x to bottom list L₀, then promotes x up some i levels
- when should i = 0 / i ≥ 1? \( \frac{1}{2} \)
- when should i = 1 / i ≥ 2? \( \frac{1}{4} \)
- when should i = 2 / i ≥ 3? \( \frac{1}{8} \)
etc.

**Insert(x):**
- Search(x) to find where x fits in L₀
- insert x in L₀
- flip fair coin
  - if heads: promote x to next level up
  - if tails: flip again (possibly newly created)

**Detail:** add special \(-\infty\) value to every list
  ⇒ first element is in every list (needed for Search)

**EXERCISE:** build a skip list with a real coin
- write H/T coin flip at each node

**Delete(x):** just remove x from all lists
Why are skip lists good?

**Warmup Lemma:** # levels in n-element skip list is $O(\lg n)$ w.h.p. $\frac{\lg n}{\lg \alpha} \sim \text{prob. } 1 - \frac{1}{n^\alpha}$

**Proof:** failure probability (not $\leq \alpha \lg n$ levels)

$= \Pr \{ \exists > c \lg n \text{ levels} \}$

$= \Pr \{ \exists \text{ some element got promoted } > c \lg n \text{ times} \}$

$\leq n \cdot \Pr \{ \text{element x got promoted } > c \lg n \text{ times} \}$ by Union Bound

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

$= \frac{n}{n^c}$

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

$= \frac{1}{n^\alpha}$ for $\alpha = c-1$ i.e. $c = \alpha + 1$. $\square$

**Theorem:** any search in an n-element skip list costs $O(\lg n)$ w.h.p.

**Cool idea:** analyze search backwards (up)

- search starts [ends] at node in bottom list
- 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
- stop [start] when we reach top level or $-\infty$
Proof of theorem:
- Search makes "up" & "left" moves, each with probability $1/2$
- number of "up" moves $< \#$ levels $\leq c \log n$ w.h.p.
  (by Warmup Lemma)
\[ \Rightarrow \text{w.h.p., number of moves} \leq \text{number of coin flips before we get $\geq c \log n$ tails} \]

Claim: $\geq c \log n$ tails in $O(\log n)$ coin flips w.h.p.

Proof: Chernoff bound
- $E[\# \text{tails in } d \log n \text{ coin flips}] = \frac{1}{2} d \log n$
- $\Pr\{ < c \log n \text{ tails in } c d \log n \text{ coin flips} \}$
- $X_i = \begin{cases} 1 & \text{if } i\text{th coin flip is heads} \\ 0 & \text{else} \end{cases}$
- $E[\sum X_i] = \frac{1}{2} d \log n$
- $\Pr\{ \sum X_i \geq (d-c) \log n \}$ (failure: too many heads)
- $\Pr\{ \sum X_i \geq (1+b)\frac{1}{2} d \log n \}$
  where $d-c = (1+b)\frac{1}{2} d$, i.e. $b = 1 - 2\frac{c}{d}$
\[ \leq e^{-b^2 d \log n / 3} \leq e^{-b^2 d / 6} < e \]
\[ = n^{-\alpha} \quad \text{for } \alpha = (d - 4c + 4c^2/d)/6 \]
  e.g. $d = 6(\alpha + 4c + 4c^2)$.
Space clearly $O(n \log n)$ by Warmup Lemma...

In fact:

**Theorem**: space used by $n$-element skip list is $O(n)$ with exponentially high prob.

$(\& \text{in expectation}) \quad \Rightarrow 1 - O(1/2^{\alpha n})$

specified constant

**Proof**: can think of skip-list construction as a big sequence of coin flips
- distance between consecutive heads = # times to promote next element
- space = # nodes = total # coin flips
- finished after $n$ heads

$\Rightarrow$ space = # coin flips until $\geq n$ heads

$= O(n)$ w. exp. h.p. (by same claim)

Also: $E[\text{space}] = E[\sum_x \# \text{ lists containing } x]$

$= \sum_x E[\# \text{ lists containing } x]$

(linearity of expectation)

$= \sum_x (1 + \frac{1}{2} + \frac{1}{4} + \cdots)$

$= \sum_x 2$

$= 2n \quad \square$