Admin:

Today: □ Skip Lists
□ DHT
   (Distributed Hash Tables)
Some probability theory.

Fair coin has $P(H) = P(T) = \frac{1}{2}$

- Suppose we flip until we see a tail.
  Let $X = \#\text{flips}$ (RV with geometric distribution)

  $\begin{array}{c}
  \text{HHTT} \\
  X=3 \\
  P(X=k) = 2^{-k} \\
  P(X>k) = 2^{-k}
  \end{array}$

- Suppose we have $n$ coins. [Problem 1]
  Each round, we flip all coins that haven't had a T yet.
  Stop when all coins have had a T.
  Let $X = \#\text{rounds}$

  $\begin{array}{c}
  X=4 \\
  \text{T} \\
  \text{H} \\
  \text{T} \\
  \text{T}
  \end{array}$

  round 1

  round 2

  round 3

  round 4

Thm: $X \leq \log n$ w.h.p.

(Intuition: $\#\text{coins flipped in round } k \propto \frac{n}{2^{k-1}}$)
Proof:

Let \( f_i = \pm \) flips for \( i^{th} \) coin, \( 1 \leq i \leq n \).

Then \( X = \max_i f_i \)

\[
P(X > c \lg n) = P(\exists i : f_i > c \lg n)
\]

\[
\leq \sum_{1 \leq i \leq n} P(f_i > c \lg n) \quad (\text{union bound})
\]

\[
= \sum_{1 \leq i \leq n} 2^{-c \lg n} = n \cdot n^{-c} = n^{1-c}
\]

This is e.g. \( \Theta \left( \frac{1}{n^{2}} \right) \) for \( c \geq 3 \).

Thus, probability \( \# \) rounds is \( \leq 3 \lg n \) is \( \geq 1 - \frac{1}{n^{2}} \).
Problem 2

• Suppose now we flip a single coin until we see $n$ tails. Let $X =$ # flips.

• (≈ previous problem, but counting flips, not rounds.)

\[
\begin{array}{cccccc}
\text{H} & \text{T} & \text{H} & \text{H} & \text{T} & \text{T} & \ldots & \text{T} \\
1 & 2 & 3 & 4 & 5 & 6 & \ldots & n
\end{array}
\]

Split after each $T$ & group column-wise to get previous chart

• Known as "negative binomial distribution".

• Now $X = \sum f_i$ (sum, rather than max now)

$E(f_i) = 2$

$E(X) = 2n$

$X$ is very likely to be near expected value.

(Central limit theorem $\Rightarrow$ normal dist)

Let's argue it is very unlikely to be $> c n$ for some $c$. 

Chernoff Bounds:

Suppose we flip a fair coin $N$ times. Let $X = \# \text{ tails}$, $E(X) = N/2 = \mu$.

Then
\[ P(X > \mu + \alpha) \leq e^{-\frac{2\alpha^2}{N}} \quad \alpha > 0 \]
\[ P(X < \mu - \alpha) \leq e^{-\frac{2\alpha^2}{N}} \quad 0 < \alpha < \mu \]
\[ P(X \geq (1+\delta)\mu) \leq e^{-\delta^2\mu} \quad \delta > 0 \]
\[ P(X \leq (1-\delta)\mu) \leq e^{-\delta^2\mu} \quad 0 < \delta < 1 \]

(2 additive, 2 multiplicative bounds)
Probability that we need to flip a coin $N = cn$ times to get $n$ tails is

$$\leq P(n \text{ tails in } N \text{ flips})$$

$$\leq P(X \leq (1-\delta)\mu) \quad X = \# \text{ tails}$$

$$\mu = \frac{N}{n} = \frac{cn}{a}$$

$$\delta = (1-\frac{a}{c})$$

$$\leq e^{-\delta^2 \mu}$$

$$= e^{-(1-\frac{a}{c})^2 (cn/2)}$$

This $\to 0$ exponentially fast with $n$.

E.g. for $c = 8$, $\text{Prob} \leq e^{-9n/4}$

Thus, with high prob, $\# \text{ flips to get } n \text{ tails}$ is $\leq c \cdot n$.
Dictionary Problem
- create
- insert/delete
- search
- successor/predecessor (new?)

<table>
<thead>
<tr>
<th>Data Structure</th>
<th>Ins/Del</th>
<th>Search</th>
<th>Pred/Succ</th>
</tr>
</thead>
<tbody>
<tr>
<td>Unsorted Linked List</td>
<td>$O(1)$</td>
<td>$O(n)$</td>
<td>$O(n)$</td>
</tr>
<tr>
<td>Sorted Array</td>
<td>$O(1)$</td>
<td>$O(\log n)$</td>
<td>$O(\log n)$</td>
</tr>
<tr>
<td>Hash Table</td>
<td>$O(1)$</td>
<td>$O(1)$</td>
<td>$O(n)$</td>
</tr>
<tr>
<td>Binary Search Tree</td>
<td>$O(\log n)$</td>
<td>$O(\log n)$</td>
<td>$O(\log n)$</td>
</tr>
<tr>
<td>Balanced BST</td>
<td>$O(\log n)$</td>
<td>$O(\log n)$</td>
<td>$O(\log n)$</td>
</tr>
<tr>
<td>Skip Lists</td>
<td>$O(\log n)$</td>
<td>$O(\log n)$</td>
<td>$O(\log n)$</td>
</tr>
</tbody>
</table>

- very simple
- randomized
- $O(\log n)$ time/operation w.h.p.

[Pugh 1989]
Idea:

• Keep all els in doubly-linked sorted list $L_0$.
• Get "fast access" to $L_0$ via another list $L_1$ containing $1/2$ as many els.
• Get "fast access" to $L_1$ via another half-sized list $L_2$, etc.
• If $k \in L_i$, then $k \in L_{i+1}$ with prob. $1/2$.
• (Put "-oo" as header on all lists...)
• Lists are doubly-linked to each other.

For each elt: Flip coin until you get a tail.
That determines how many levels it is on.
Search: Start in top list
    move to last elt that is ≤ target.
    drop down a level
    keep moving right & dropping down until you
    are in list L_0, in some way at each level
    finish up as normal for linked list on L_0.

Example: Search(25)

Insert/Delete/Successor/Predecessor: Easy

(note that insert needs to flip more coins...)
(give examples)
**Thm:** Height of skip list (i.e. # levels) is $O(\log n)$ w.h.p.

**Pf:** This is problem 1 in our probability section earlier.

**Thm:** # nodes in skip list is $O(n)$ w.h.p.

**Pf:** This is problem 2 in our probability section earlier.
Thm: Time for search is $O(\lg n)$ w.h.p.

Pf: Let $h = \text{height of skip list}$. $h = O(\lg n)$ w.h.p.

Let $s = \# \text{nodes on search path}$.

We can follow search path backwards from ending place in $L_0$:

- go $\uparrow$ when you can, until you get to top-left node
- else go $\leftarrow$

Each $\uparrow$ corresponds to tossing an $\text{H}$ for corresponding elt.

Each $\leftarrow$ "" "" a $\text{T}"" "" "".

$\approx \text{sequence of} s \text{ coin tosses with only} h \text{ heads}.$

$(\exists c) s \geq c \lg n \Rightarrow P(s \text{ flips with } \leq h \text{ heads})$

$= O\left(\frac{1}{n^\alpha}\right)$ for some $\alpha > 0$.

(Problem 2)
Distributed Hash Table (e.g., Chord)

(Idea only)
Want to distribute files to servers.
File i has name $F_i$ & data $D_i$.
Each server $j$ should have $\approx$ same # of files.
Servers may come & go (new requirement!)
$\Rightarrow$ shouldn't be too costly.

Hash function $h : F_i \rightarrow [0,1]$ \hspace{1cm} $0 \leq h(F_i) < 1$

Also for servers: Each server $S_j$ has random $x_j \in [0,1)$

Work mod 1:

```
\text{these files stored on } S_1
\} \quad S_2 \text{ at position } x_2
\} \quad S_3
\} \quad S_4
\} \quad S_1
```

\text{\(S_3 \) at position x, \(S_4 \) at position y, \(S_1 \) at position z, \(S_2 \) at position x_2}
Define $\text{succ}(x) = \min \{x_j \text{ s.t. } x_j > x \pmod{1}\}^\dagger$

File $F_i$ stored at server $S_j$ where $x_j = \text{succ}(h(F_i))$

Each $S_j$ stores pointers to $\text{succ}(x_j + 2^{-k}) \quad k = 1, 2, \ldots, m$

(only logarithmic $n \leq n$ of servers)

Find right server for a file $F$ is very much like searching in a skip list: take "big steps" greedily, then smaller & smaller ones until you find relevant server.

Adding server $\Rightarrow$ split interval
Deleting server $\Rightarrow$ merge intervals