Lecture 5: Sets & Dictionaries via.... hashing (later) ①

Abstract Data Type: Set of strings [say]

[e.g. passwords, names, tweets; many possible, not all known in advance]

Ops: (• Initialize)

• Insert(s) // s a string

• IsMember(s) // true/false.
Lookup

[Do example, human alg., requesting ~~words~~ from class TV/movie/show names

Write on slips.

Store. First unordered. Then ordered....

Linked-list? BST...? Could get into these,...]

Binary Search Tree: $O(\log n)$ time ops if n elts.

[kind of complicated. Can we exploit $O(1)$-time array lookup?]

• If you know all elts s are 3-digit #'s...

alloc a Boolean array of len. 1000!

But what if they're 10-digit nums: 10B-size array :(

280-char tweets :(

Idea: Hashing: invent "hash" fcn: $h: \{\text{all strings}\} \to \{0, 1, \ldots, n-1\}$ $n =$ (out of space you can afford.)

• Initialize: $T[0 \ldots n-1] := \text{false}$

• Insert(s): $T[h(s)] := \text{true}$

• ~~IsMember~~(s): return $T[h(s)]$.
Lookup

Issues!

Issue 1: Where/how to get h?

Issue 2: Collisions: #strings $= \infty$. Even if len. bound, probably #strings $\gg n$.
$\therefore$ ~~[illegible]~~ (PHP) multiple strings s_1, s_2 have $h(s_1) = h(s_2)$ $\Rightarrow$ erroneous lookups, potentially

Solution 2: "Chaining" [other possibs...]
T[·] stores linked lists, not just True/False.

Init $\longrightarrow$ $T[i] := \text{null}$
Insert(s): insert "s" at end of list at $T[h(s)]$
~~IsMember~~ Lookup(s): look thru list at $T[h(s)]$ for 's'

Bonus: can store any data/val v together with "key" s, get a "dictionary" data type, like python's "MyDict[s] = v".

~~Worst-case Insert/Lookup for~~

Space (mem.) usage: $O(n$ ← len. of T[·], you choose
$+ m$ ← # items inserted
$(\cdot\, L)$ ← size of item, but say $O(1)$)

Worst-case time for insert/lookup(s): $O(\underbrace{\text{length of } T[h(s)]}_{\text{maybe } m})$ ☹ [BST has $O(\log m)$]

→ could set $n = 1$, $h(s) = 0$ $\forall s$ → just a linked list!

Idea: Suppose $h(s)$ was somehow a "random-looking" # from $0 \ldots n-1$ (but actually deterministic)

→ e.g. (ascii int. represent of s) mod n ??? [is that random?]

Hope: after inserting $s_1, \ldots, s_m$, the #'s $h(s_1), \ldots, h(s_m)$ are "random-ish", so "average" len. of a list is $\frac{m}{n}$.

⇒ "average-case" insert/lookup $= O(1 + \frac{m}{n})$ [curr # elts] ?

⇒ If you estimate set will have $\approx m$ elts., can choose $n = m$, have space $O(m)$ ☺ "avg-case time" $O(1)$ ☺.

[This is the key idea. But many q's to answer...]

- $h(s) = \text{int}(s) \bmod n$ is bad... // strings ending w/ same few chars map to same thing...
- how to make a "random-ish" h?
- what is "avg-case", formally? any worst-case guarantees?
- what if you can't est. m in advance — or, what if you <u>know</u> elts, in advance, just want to do lookups?
- methods other than chaining to handle collisions?

Random-ish hash fcn? [let's think on this first]

① Dream scenario: Some oracle does...

for s in all strings (!!?)

$h(s) := \text{RandInt}(0 \ldots n-1)$

And then you can "magically" use h in $O(1)$ time, ~~space~~ no space

② Reality attempt, python:

```
h = ord(s[0] << 7)
for char in s
    h = c_mul(1000003, h) ^ ord(char)
h = (h ^ len(s)) % n.
```

// maybe earlier python, now "SipHash" changed in 2012

// this is completely deterministic, but "seems sorta random".

[But knowing python does this, an evil person could keep inserting s's that hash to same #...]

"hash flooding" or "hash DoS"

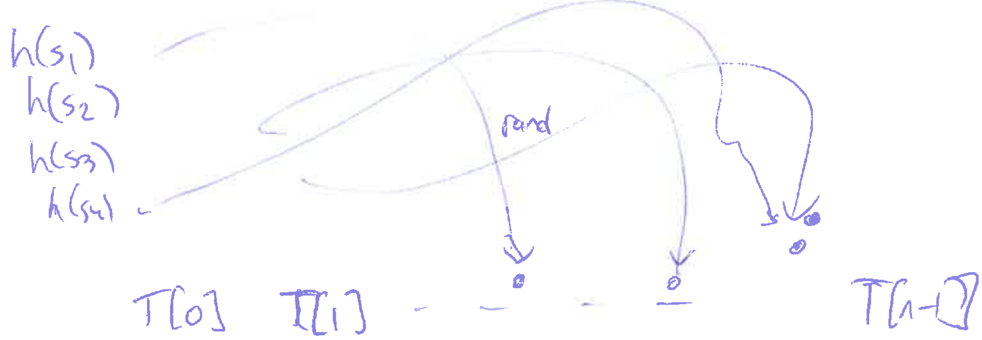
"The Power of Evil Choices in Bloom Filters."

Today: "SUHA": Simple Uniform Hashing Assumption: do ②, pretend it's ①

[Good for analyzing hash tables in ideal circs.]

Later: Principled ways to provably achieve things achievable with SUHA.

Under SUHA: if you insert m items into size-n table...



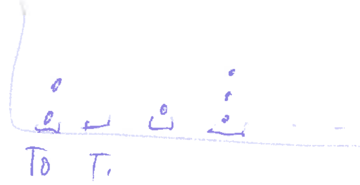
"Balls and Bins"

[a very common scenario]

"Average load/length": $\lambda = \frac{m}{n}$. (5)

Say you now do an unsuccessful lookup(s)

$\because$ s not in table, already $h(s) \overset{SUHA}{\sim} \text{RandInt}(0 \ldots n-1)$



$\therefore$ E[~~the~~ length of T[h(s)]]

$= \text{avg}\{\text{bin sizes}\} = \lambda$. $\rightarrow$ time $O(\lambda)$

Say "successful lookup": s in table. Say $h(s) = i$.

All $m-1$ other items act like rand. balls, so

E[length of T[h(s)]] = E[# balls in bin i after $m-1$ throws]

$= \frac{m-1}{n} \leq \frac{m}{n} = \lambda$. $\rightarrow$

$\therefore$ space $= O(n)$, op time $= O(1+\lambda) = O(1+\frac{m}{n})$ on average [$\leftarrow$ rigorously def'd, under SUHA]

$m=n$: popular choice. Space: $O(m)$, op time $O(1)$ on avg

Say $m=n$,

Q: What is worst-case lookup time ... on average?

load of most loaded bin.

$\lambda = 1$ but ... [very unlikely all balls in diff bins!!]

$\Pr[\text{first } 2 \text{ balls go into } T[0]] = \frac{1}{n^2}$

$\Pr[\text{exactly } 2 \text{ balls go into } T[0]] = \binom{n}{2} \cdot \left(\frac{1}{n}\right)^2 \left(1-\frac{1}{n}\right)^{n-2}$

$\binom{n}{2} = \frac{n(n-1)}{2} \leq \frac{n^2}{2}$, $\left(1-\frac{1}{n}\right)^{n-2} \leq 1$, so the whole is $\leq \frac{1}{2}$.

[??] what is this if $n = 1000$?

$\Pr[\text{exactly } 3 \text{ go into } T[0]]$

$= \binom{n}{3}\left(\frac{1}{n^3}\right)\left(1-\frac{1}{n}\right)^{n-3} \leq \frac{n(n-1)(n-2)}{3!} \cdot \frac{1}{n^3} \cdot 1 \leq \frac{1}{3!}$

(6)

$\therefore$ $\Pr[\text{ exactly } 10 \text{ go into } T[0]] \le \frac{1}{10!}$

$\therefore$ $\Pr[\ge 10 \text{ go into } T[0]]$

$= \Pr[\{=10 \text{ go in}\} \cup \{=11 \text{ go in}\} \cup \{=12 \text{ go in}\} \cup \cdots] \le \Pr[=10 \text{ in}] + \Pr[=11 \text{ in}] + \cdots$ "Union Bound"

$$\le \frac{1}{10!} + \frac{1}{11!} + \frac{1}{12!} + \frac{1}{13!} + \cdots$$

$$= \frac{1}{10!}\left(1 + \frac{1}{11} + \frac{1}{11\cdot 12} + \frac{1}{11\cdot 12\cdot 13} + \cdots\right)$$

$$\le \frac{1}{10!}\left(1 + \frac{1}{11} + \frac{1}{11^2} + \frac{1}{11^3} + \cdots\right)$$

$$= \frac{1}{10!}\left(\frac{1}{1-\frac{1}{11}}\right) = \frac{1}{10!}\cdot\frac{11}{10}$$

$\therefore$ $\Pr[\ge k \text{ go into } T[0]] \le \frac{1}{k!}\cdot\frac{k+1}{k} \le \frac{2}{k!}$ lazy/sloppy! $\le \frac{2}{2^{k-1}} = \frac{4}{2^k}$

$\therefore$ $\Pr[\{\ge k \text{ in } T[0]\} \cup \{\ge k \text{ in } T[1]\} \cup \cdots \cup \{\ge k \text{ in } T[n-1]\}]$

$\le \Pr[\ge k \text{ in } T[0]] + \cdots + \Pr[\ge k \text{ in } T[n-1]]$ U.B.

$\le \frac{2}{k!} + \frac{2}{k!} + \cdots + \frac{2}{k!} \overset{\text{sloppy}}{\le} \frac{4}{2^k}\cdot n \longrightarrow \le \frac{1}{2^{20}} = \frac{1}{\text{million}}$

if $2^{22}\cdot n \le 2^k$ $\Leftrightarrow$ $k \ge \lg n + 22$

How much is $k!$?

$100! = 100\cdot 99\cdot 98\cdot 97\cdot \cdots \cdot 2\cdot 1$

$\ge 2\cdot 2\cdot 2 \cdots \cdot 2 = 2^{99}$

or $\ge 3\cdot 3\cdot 3 \cdots \cdot 3 = 3^{98}$ $\leftarrow$ which bigger?? [way bigger]

or $\ge \ldots 4^{97}$ or $5^{96}\ldots$

or $\ldots \ge 50^{51} \ge 50^{50}$. [Not optimal, should do $\frac{100}{e}$, not $\frac{100}{2}$.]

$\log(3^{98}) = 98\cdot\log 3$
$= 98\cdot(1.6)$

$\log(2^{99}) = 99\lg 2 = 1$

So $k! \ge \left(\frac{k}{2}\right)^{k/2}$. $\lg(k!) \ge \frac{k}{2}\lg\left(\frac{k}{2}\right) = \Omega(k\log k)$.

$2^{22}\cdot n \le 2^{\Omega(k\log k)}$

$\Rightarrow k \le O\left(\frac{\log n}{\log\log n}\right)$.

max load $\le \lg n + 22$ except w.p. 1/million

Q: how strongly "[illegible]" used?