Theory & Practice of Linear Probing

Mihai Pătrașcu

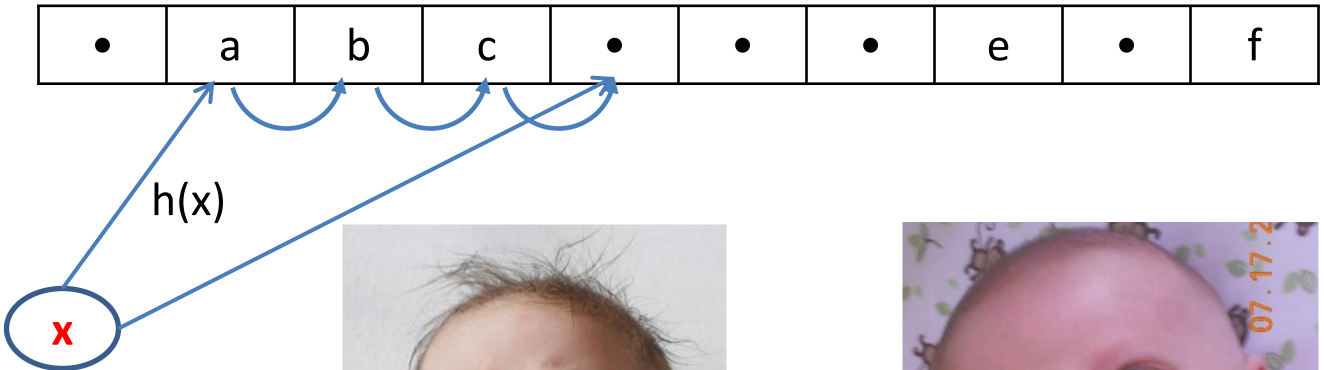
at&t

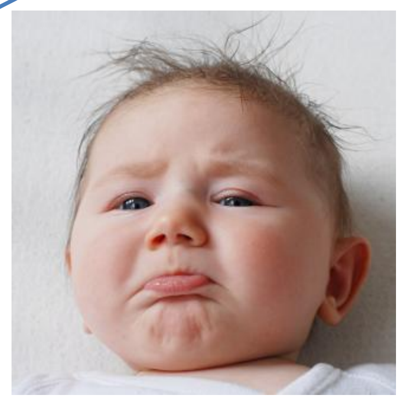
WADS 2011

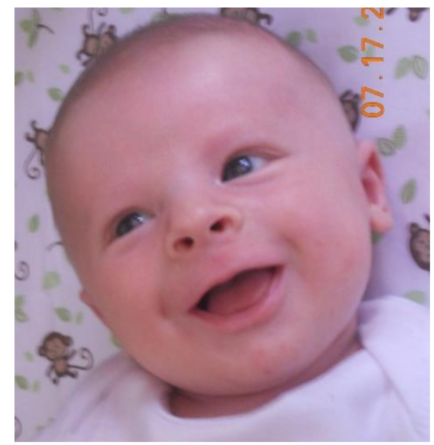
Hashtables

- Good target for theory:
  "10% of the code runs 90% of the time"
- Used in time-critical code
  (e.g. in routers)
- Understanding can only come from theory
  (nonobvious&counterintuitive Maths)
- Theory has had a great run.

Linear Probing







- interaction among close hashes
- positive feedback in growth of "runs"

- cache lines
- prefetcher

Bottom line: linear probing = the ★ of the moment [mem-access + 5%!]

A bit of history

[Samuel, Amdahl, Boehme'54] invented (IBM 701 asm)

Analysis 1: Assume ideal hashing...

h: U → [m]

Truly random function

Cryptographic hash functions

x

h(x)

uniform in [m],
independent of everything else

Analysis

[Samuel, Amdahl, Boehme'54] invented (IBM 701 asm)

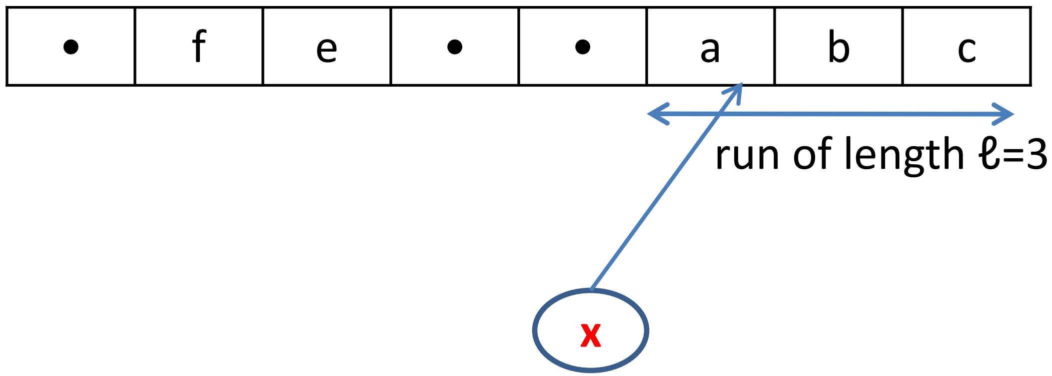
[Knuth ^{TR}62] $E[t] = O(1/\varepsilon^2)$ for load $1-\varepsilon$
[Pagh^2, Ružić ^{STOC}07] $E[time]=O(1)$ for load<1.
[P., Thorup'11] $E[t] = O(1/\varepsilon^2)$ for load $1-\varepsilon$ a la [PPR]

Following slides: sketch of [PPR'07]
- Assume load ⅓ : $m \geq 3n$
- Theorem: E[time / operation] = O(1)

Linear Probing: Analysis

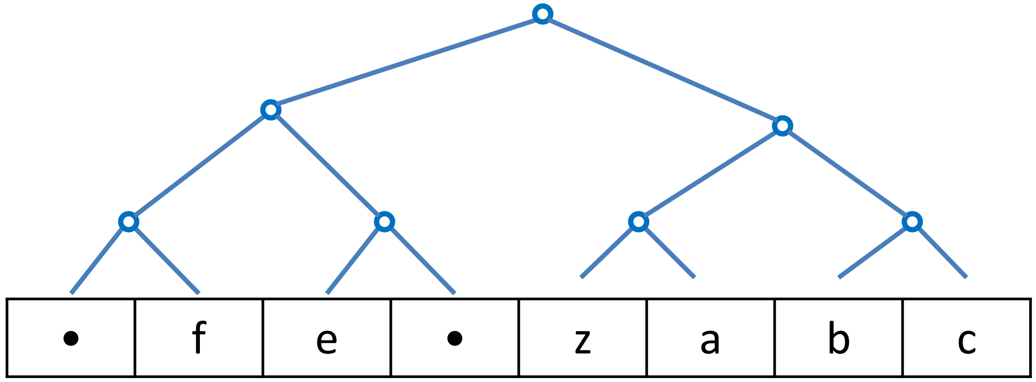
Cost of {query(x), insert(x), delete(x) }

≤ length of run containing h(x)

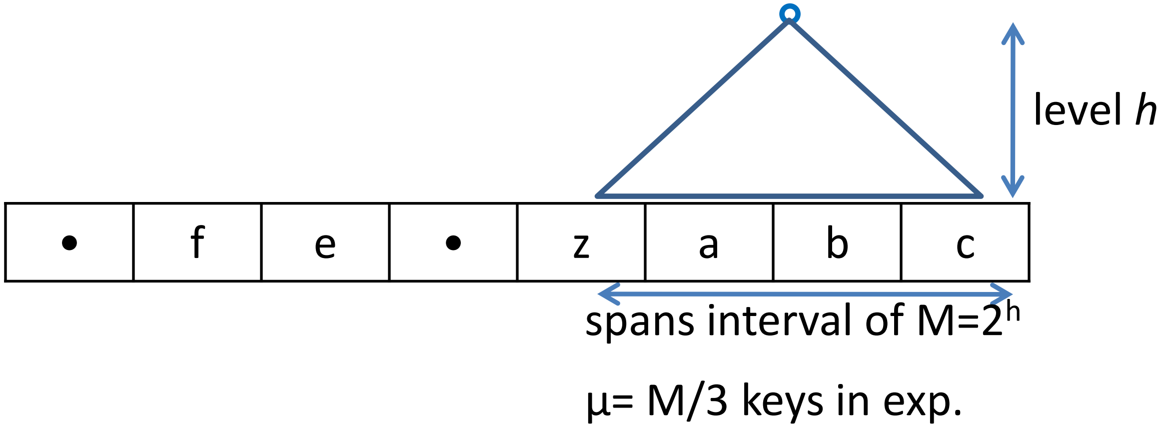


Linear Probing: Analysis

Consider binary tree on top of array.

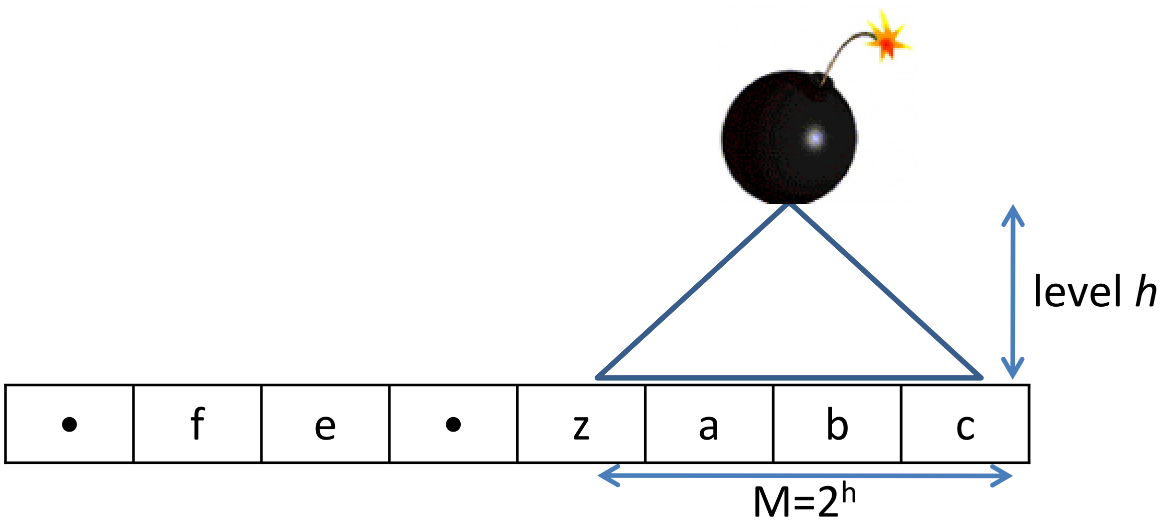


Linear Probing: Analysis



Linear Probing: Analysis

Dangerous node: > 2μ keys in subtree



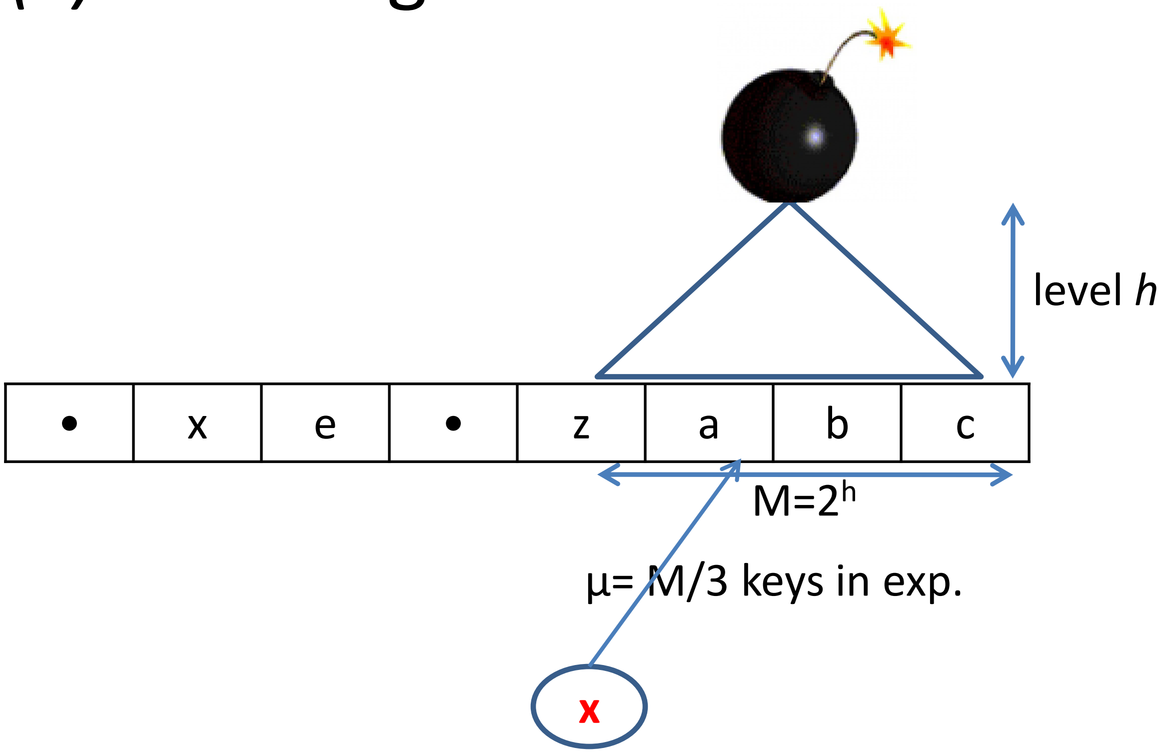
μ= M/3 keys in exp.

Linear Probing: Analysis

Almost, but not quite

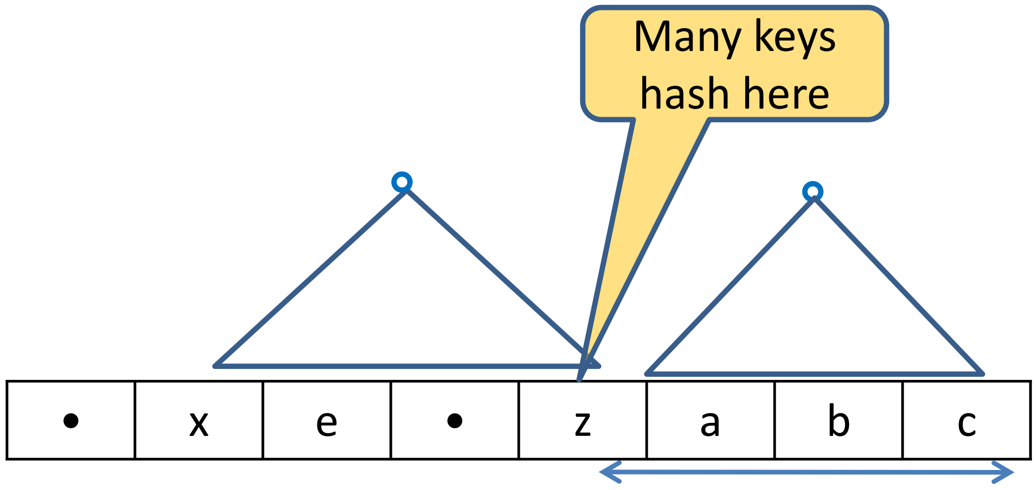
<u>Intuition</u> If $h(x) \in$ run of length $[2^{h-1}, 2^h)$

$\Rightarrow h(x)$ has dangerous ancestor on level $\approx h$



Linear Probing: Analysis

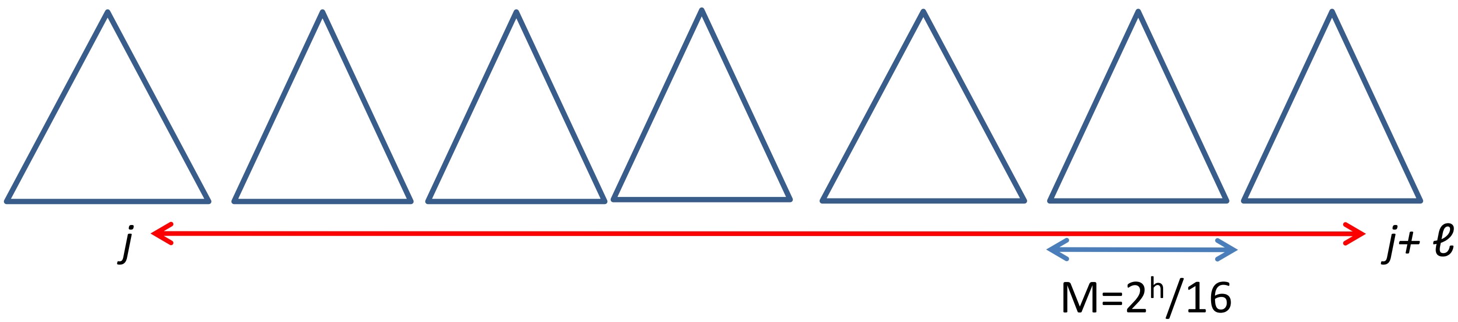
Reality: long run could also come from sibling.



Linear Probing: Analysis

Assume: h(x) ∈ run [$j..j+\ell$], $2^{h-1} \leq \ell < 2^h$

- run covered by 8–18 nodes on level h-4



<u>Claim</u> Not all these nodes can be safe.

Otherwise, first 4 subtrees:

- span ≥ 3M = 9μ table cells in the run
- have ≤ 8μ keys

Linear Probing: Analysis

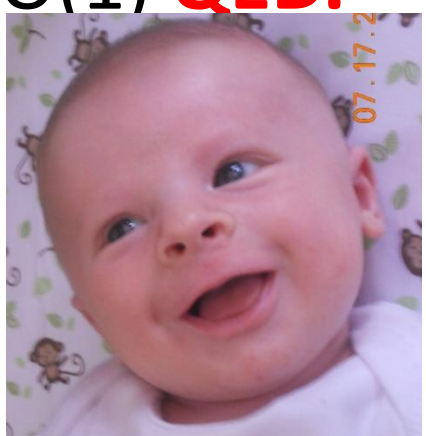
$E[\text{Cost}] \approx \Sigma_h\, 2^h \cdot \Pr[h(x) \in \text{run of length } 2^{h-1}..2^h]$

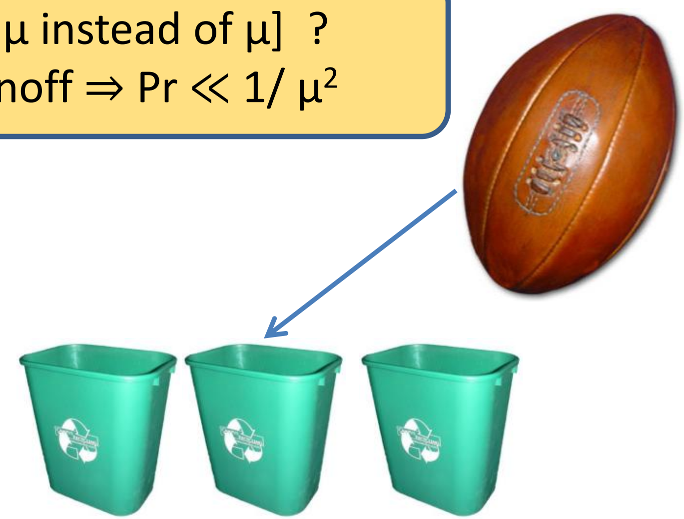
$\leq \Sigma_h\, 2^h \cdot (18 \cdot \Pr[\text{node on level } h\text{-}4 \text{ dangerous}])$

$\leq \Sigma_h\, 1/\mu$

$= \Sigma_h\, 2^{-h} = O(1)$ **QED.**

Pr[2 μ instead of μ] ?
Chernoff $\Rightarrow$ Pr $\ll 1/\mu^2$





The Holy Trinity of Hash Tables

Linear probing

Collision
-Chaining

Cuckoo
hashing

How to Implement Hash-Tables

You will need:

- a student
- coffee
- <u>a hash function</u>

Hashing ≈ (Integer) Hashing

•	a	x	c	•	•	•	e	•	f

integer hashing

vectors
strings
…

"universal" hashing [Wikipedia]

$n^{O(1)}$

Open: *subpolynomial* failure probability.

How to Hash Integers

’50s, ’60s   complicated functions

Make them “look hard”
… And assume they behave randomly

How to Hash Integers

’50s, ’60s complicated functions

[Carter & Wegman’79] simple functions

E.g. Let u=prime. Pick $a_k, \ldots, a_0$ randomly

$x \mapsto [a_k x^k + a_{k-1} x^{k-1} + \ldots + a_0]$ (mod u) mod m.

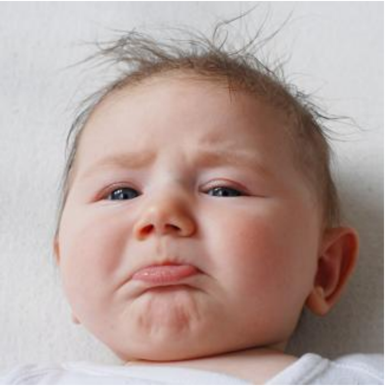
“*k*-independence”

Hash of any k inputs is independent, uniform on [m].

How to Hash Integers

’50s, ’60s   complicated functions

[Carter & Wegman’79] simple functions



For most things, we want independence $k \approx \lg n$ …

How to Hash Integers

’50s, ’60s complicated functions

[Carter & Wegman’79] simple functions

[Siegel’89] a complicated function

<u>Lower bound</u> With space $u^{1/c}$, any k-independent function needs evaluation time ≥ min{k, c}

How to Hash Integers

’50s, ’60s   complicated functions

[Carter & Wegman’79] simple functions

[Siegel’89] a complicated function

<u>Upper bound</u> With space $u^{1/c}$, a u^{1/c^2}-independent hash function with evaluation time $c^{O(c)}$

Theory summary:
Space n^{ε}, independence $n^{\Omega(1)} \gg \text{poly}(\log n)$,
O(1) evaluation. “Only issue” : non-explicit expander

How to Hash Integers

’50s, ’60s complicated functions

[Carter & Wegman’79] simple functions

[Siegel’89] a complicated function

[P. Thorup ^{STOC}11]

We don’t need no, complication

“Simple Tabulation” works for most things

How to Hash Integers

’50s, ’60s complicated functions

[Carter & Wegman’79] simple functions

[Siegel’89] a complicated function

[P. Thorup STOC11]

We don’t need no, complication

“Simple Tabulation” works for most things

$T_1, \ldots, T_q = u^{1/q}$ hashes picked randomly $\in [m]$

$x \mapsto (x_1, \ldots, x_q)$

$h(x) = T_1[x_1] \oplus \ldots \oplus T_q[x_q]$

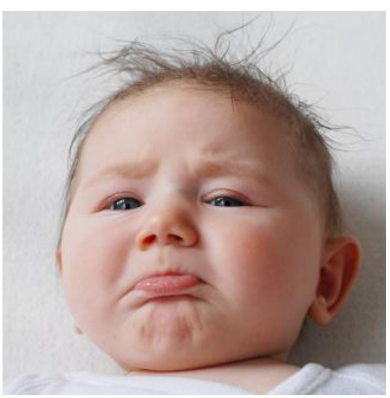
Simple Tabulation: "Uniting Theory and Practice"

- Simple & fast enough for practice.
- But with good mathematical guarantees:
- Chernoff bounds ⇒ chaining, linear probing
- Cuckoo Hashing

But who needs Proofs, Anyway?

Maybe we're fine with $x \mapsto (ax+b) \bmod p \bmod m$

[Mitzenmacher, Vadhan'08] It works for any distribution with high min-entropy

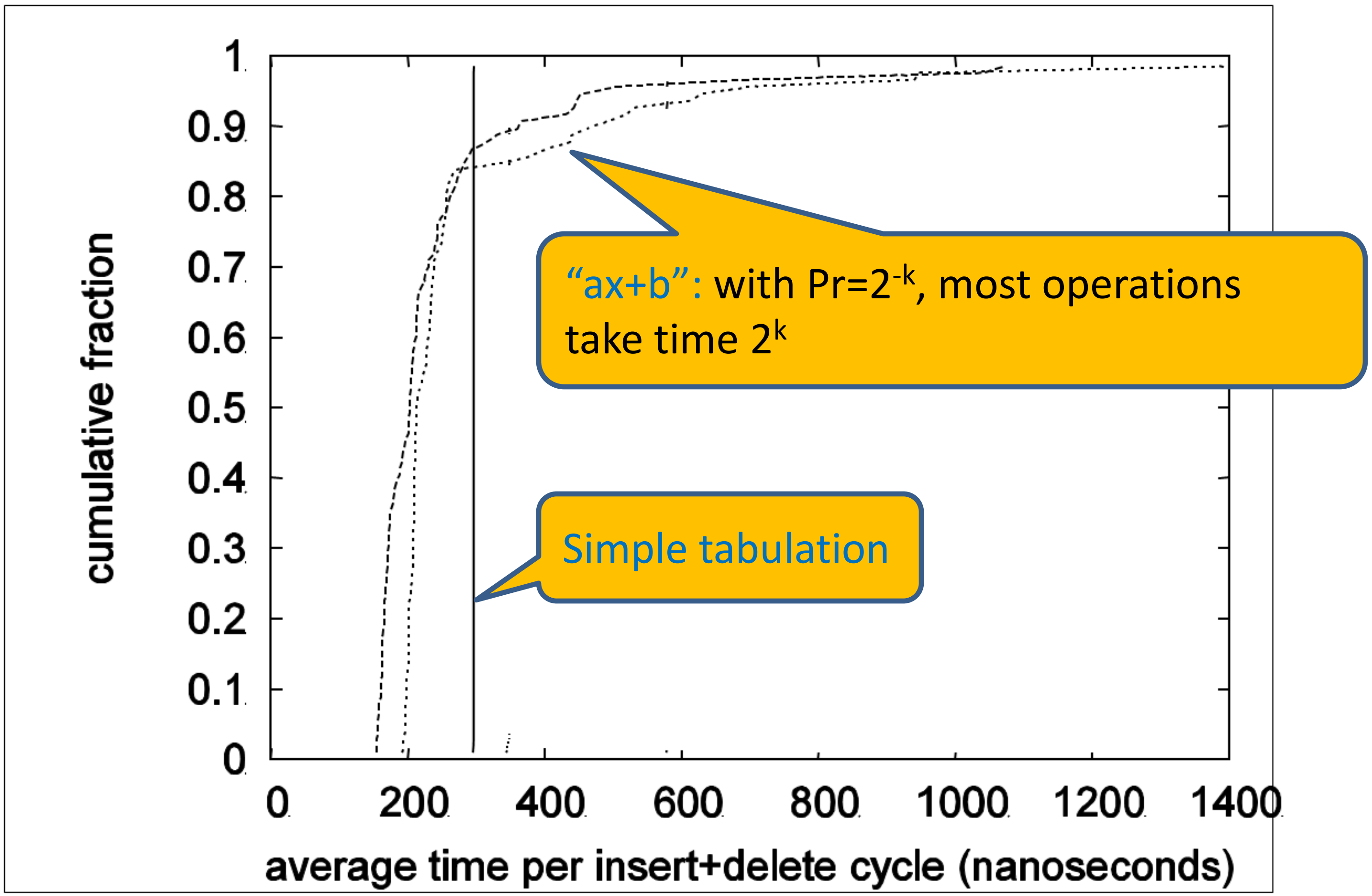


But E[time/operation] $=\Theta(\lg n)$ for the set $\{ x, x+1, x+2, \ldots, x+n \}$ [P. Thorup ICALP11]

[SQL Slammer Worm'03]

Failure may not be obvious ("Practice needs Theory")

x ↦ (ax+b)mod p mod m   on data set {1, 2, ..., n}



Theory needs Practice (to understand our targets)

Simple tabulation:

q probes into tables of size $u^{1/q}$

use $u^{1/q} = 256 \Rightarrow$ tables in cache

$\Rightarrow$ time close to a multiplication

Would it be "better" to use

$$x \mapsto \left[ax^2 + bx + c\right](\text{mod } p) \text{ mod } m \quad ?$$

Probably not...

Input = 32-bit	**32-bit machine**	**64-bit machine**
Universal (a*x)>>s	1.87ns	2.33ns
simple tabulation	4.98ns	4.61ns
Input = 64-bit		
Universal (a*x)>>s	7.05ns	3.14ns
Simple tabulation	15.54ns	11.40ns

Sample Result

Lemma

Say $m > n^{1+\varepsilon}$

Any bin has at most O(1) balls whp.

(with probability $1-n^{-\gamma}$, at most C_γ balls/bin)

Proof

$\Pr[\text{t independent keys in same bin}] \leq n^t/m^{t-1}$

Sample Result

Lemmata Among any set S of keys, ∃ T ⊆ S of $|T| \geq \log_2|S|$ keys that hash independently via simple tabulation.

- *i* = 1st coordinate where not all keys are same
- α = least common char on position *i*
- $S_{(\alpha,i)}$ = keys ∈S with α on position *i*
- pick some $x \in S_{(\alpha,i)}$
  Observe: h(x) independent of $S \setminus S_{(\alpha,i)}$
- remove $S_{(\alpha,i)}$ from S, repeat □

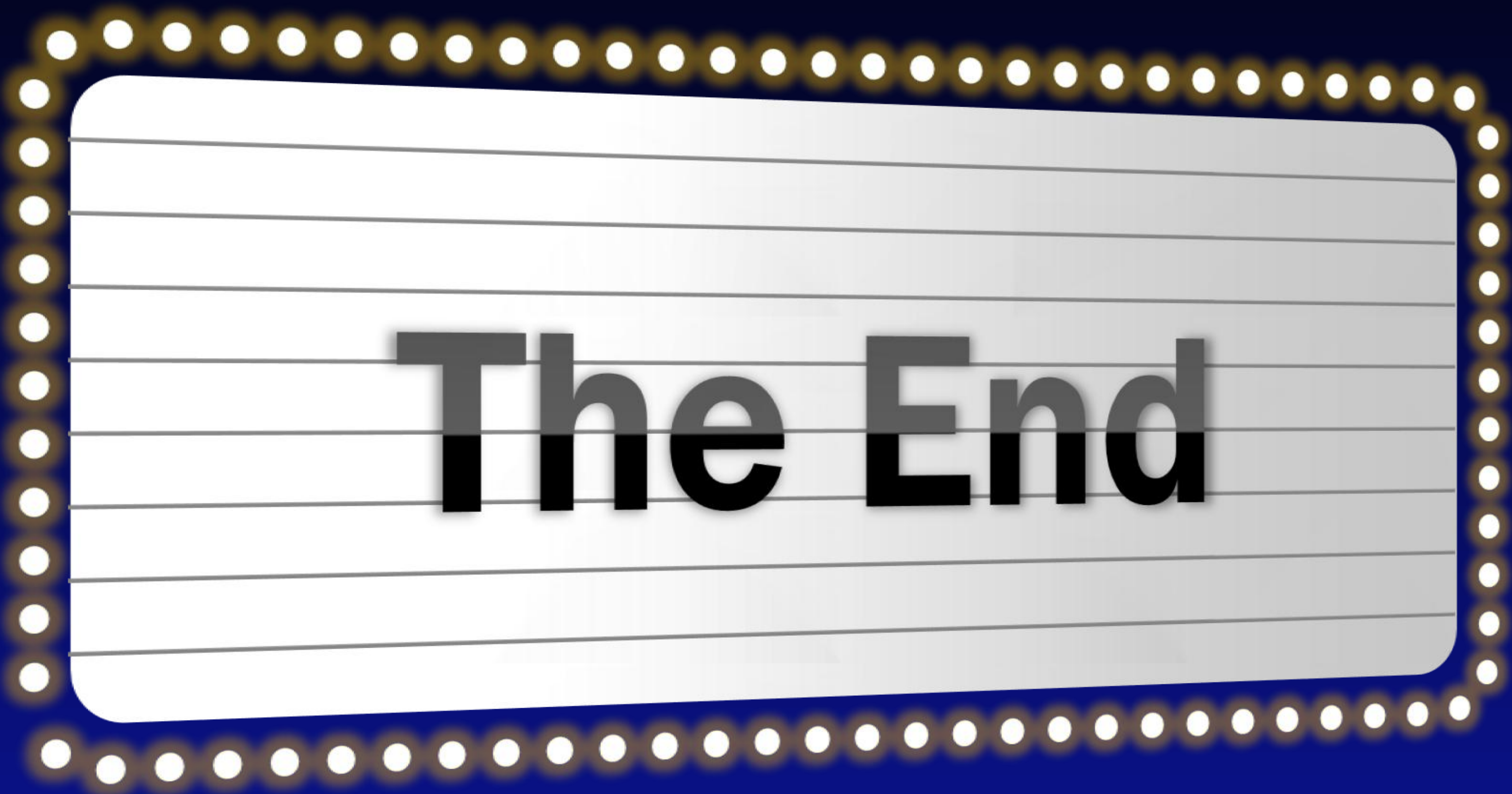
Sample Result

Theorem X = #balls in a "designated" bin

$\mu = E[X] = {}^{n}/_{m}\ \gamma$ = constant.

$$\Pr[X \notin (1 \pm \delta)\mu] < e^{-\Omega(\delta^2 \mu)} + n^{-\gamma}$$

- α on position 1 is *rare* iff $|S_{(\alpha,1)}| \le n^{1-1/c^2} = n^{1-\varepsilon}$
- each $S_{(\alpha,1)}$ is randomly shifted by $T_1[\alpha]$
  & contributes at most O(1) to any bin ⇒ Chernoff
- remove all such $S_{(\alpha,1)}$'s from S, repeat.
- at most n^{1/c^2} non-rare values remaining.
- so at the end, $|S| \le n^{1/c}$. Everything is rare. □


The End