6.851	Lecture 10	Mar. 20, 2012

TODAY: Hashing
- universal & k-wise independent
- simple tabulation hashing
- chaining & perfect hashing
- linear probing
- cuckoo hashing

Hash function: $h: \underbrace{\{0, 1, \dots, u-1\}}_{\text{universe } \mathcal{U} \text{ of keys}} \to \{0, 1, \dots, m-1\}$ table size

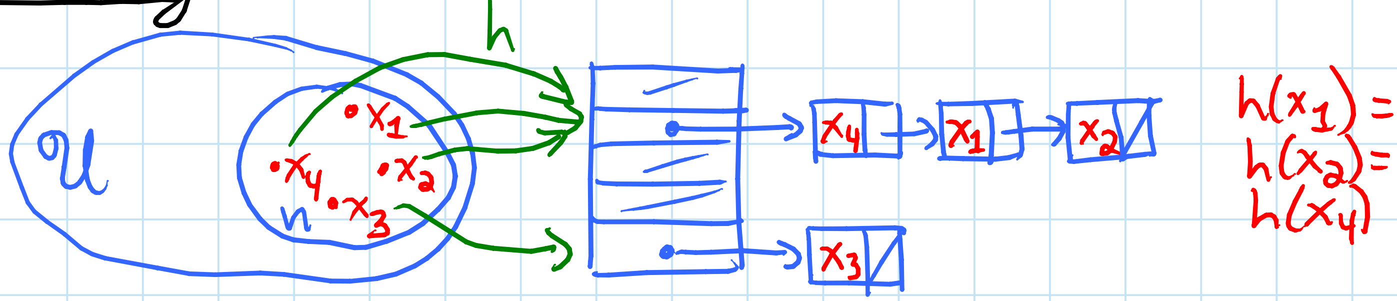
- **Totally random:** $\Pr_h\{h(x)=t\} = 1/m$, independent of $h(y)$ for all $x \neq y \in \mathcal{U}$
  - $\Theta(u \lg m)$ bits of information — TOO BIG
  - "simple uniform hashing" of [CLRS]

- **Universal:** choose h from family $\mathcal{H}$ with $\Pr_{h\in\mathcal{H}}\{h(x)=h(y)\} = O(1/m)$ for all $x \neq y \in \mathcal{U}$
  strong if $\leq 1/m$
  [Carter & Wegman — JCSS 1979]
  - $h(x) = [(ax) \bmod p] \bmod m$ for $0 < a < p$
    or: vector dot product
    prime $\geq u = |\mathcal{U}|$
  - $h(x) = [(ax) \bmod u] \,//\, 2^{\lg u - \lg m}$ for odd $a < 2^w$, m & u = powers of 2
    $= (a \cdot x) \gg (\lg u - \lg m)$
    integer division
    right shift
  [Dietzfelbinger, Hagerup, Katajainen, Penttonen — J.Alg. 1997]

- <u>k-wise independent</u>: family $\mathcal{H}$ of hash functions with $\Pr_{h \in \mathcal{H}}\{h(x_1)=t_1 \,\&\, \cdots \,\&\, h(x_k)=t_k\} = O(1/m^k)$ for all distinct $x_1, x_2, \ldots, x_k \in \mathcal{U}$ [Wegman & Carter – JCSS 1981]
  - pairwise ($k=2$) is stronger than universal
    - $h(x) = [(ax+b) \bmod \underbrace{p}_{\text{prime} \geq u}] \bmod m$ for $0 < a < p$ & $0 \leq b < p$
  - $h(x) = \left[\left(\sum_{i=0}^{k-1} a_i x^i\right) \bmod p\right] \bmod m$ for $0 < a_{k-1} < p$ & $0 \leq a_i < p$ [WC81]
  - $O(n^\varepsilon)$ space, $f(k)$ query, uniform, & practical [Thorup & Zhang – SODA 2004]
    - esp. $k=5$
  - $\underbrace{O(n^\varepsilon)\text{ space}}_{\text{necessary for}}$, $O(1)$ query for $k = \Theta(\lg n)$ [Siegel – SICOMP 2004]

- <u>Simple tabulation hashing</u>: [WC81]
  - view x as vector $x_1, x_2, \ldots, x_c$ of characters
  - totally random hash table T_i on <u>each</u> character
    $\Rightarrow O(c\,u^{1/c})$ words of space
  - $h(x) = T_1(x_1) \oplus T_2(x_2) \oplus \cdots \oplus T_c(x_c)$
    $\Rightarrow O(c)$ time to compute
  - 3-independent

Chaining:

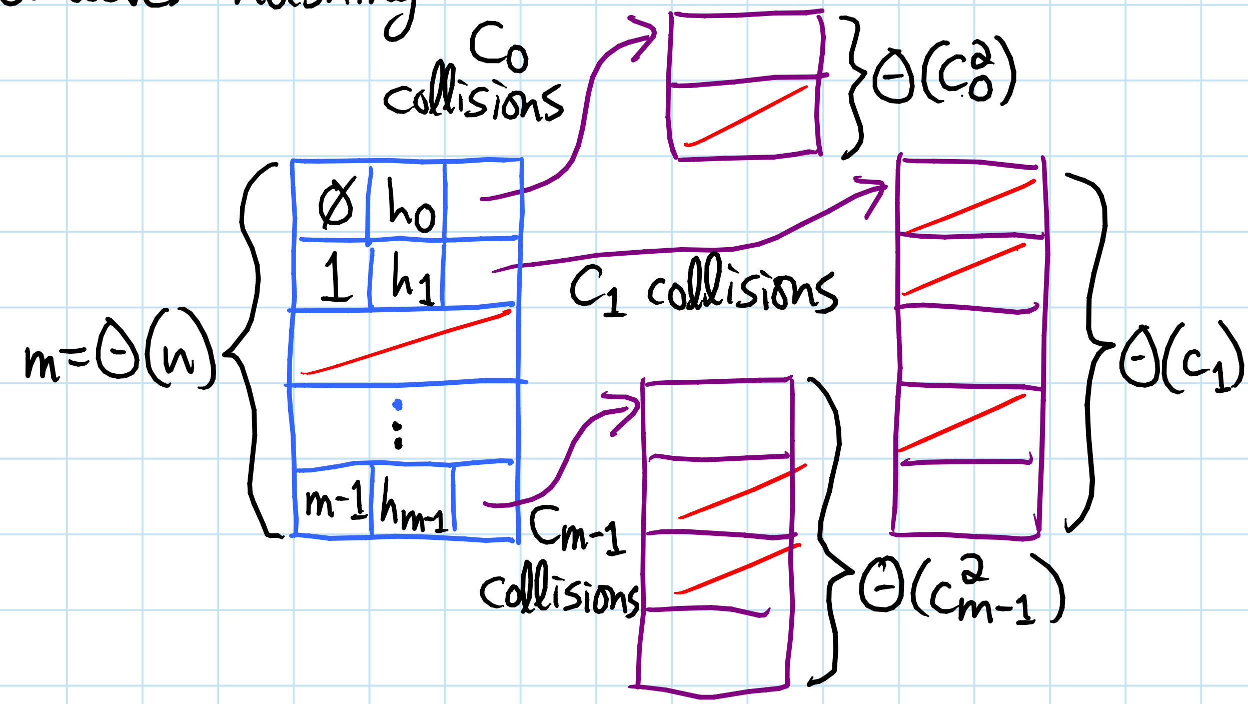


- $E[C_t = \text{length of chain } t] = \sum_i \Pr\{h(x_i) = t\}$
- universal $\Rightarrow O(n/m) = O(1)$ for $m = \Omega(n)$
- $\text{Var}[C_t] = E[C_t^2] - E[C_t]^2$
- assuming h is "symmetric":
  $E[C_t^2] = \frac{1}{m} \sum_s E[C_s^2] = \frac{1}{m} \sum_{i,j} \Pr\{h(x_i) = h(x_j)\}$
- universal $\Rightarrow = \frac{1}{m} \cdot n^2 \cdot O(1/m) = O(n^2/m^2)$
  $= O(1)$ for $m = \Omega(n)$
- totally random $\Rightarrow C_t = O\left(\frac{\lg n}{\lg \lg n}\right)$ with high probability
  (tight; $1 - 1/n^c$ for any c)
  - $\Pr\{C_t > c \cdot \mu\} \le e^{(c-1)\mu} / (c\mu)^{c\mu}$ [Chernoff]
  - $c = \frac{\lg n}{\lg \lg n} \Rightarrow \Pr \approx 1 / O\left(\frac{\lg n}{\lg \lg n}\right)^{O\left(\frac{\lg n}{\lg \lg n}\right)} = 1/n^{O(1)}$
- $\Theta\left(\frac{\lg n}{\lg \lg n}\right)$-wise independence $\Rightarrow$ same
  [Schmidt, Siegel, Srinivasan — SIDMA 1995]
- simple tabulation hashing $\Rightarrow$ same
  [Pătrașcu & Thorup — STOC 2011]
- with "cache" of $\Omega(\lg n)$ items, totally random $\Rightarrow$
  $O(1)$ amortized w.h.p.: [Pătrașcu — blog, Feb. 2, 2011]
  - $\Theta(\lg n)$ keys collide with "batch" of $\lg n$ ops. w.h.p.
  - $\mu = \lg n$, $c = 2 \Rightarrow \Pr \le e^{\lg n} / (2 \lg n)^{2 \lg n} < 1/n^c$

FKS perfect hashing: [Fredman, Komlós, Szemeredi – J. ACM 1984]

- store chain C_t as hash table of size $\Theta(C_t^2)$

$\Rightarrow$ 2-level hashing:



- $E[\#\text{ collisions in } C_t \text{ table}] = \sum_{i,j\in C_t} \Pr\{h(x_i) = h(x_j)\}$
- universal $\Rightarrow = C_t^2 \cdot O(1/C_t^2)$
- set Θ constant to make $\le 1/2$

$\Rightarrow \Pr\{\emptyset \text{ collisions in } C_t \text{ table}\} \ge 1/2$ $\quad \Pr\{X \ge x\} \le \frac{E[X]}{x}$ [Markov]

$\Rightarrow O(1)$ expected trials to build collision-free C_k

- $E[\text{space}] = \Theta(m + \sum_t C_t^2) = \Theta(m + n^2/m)$

$= \Theta(n)$ for $m = \Theta(n)$

$\Rightarrow O(1)$ deterministic query

$O(n)$ space $\Big\}$ one in expectation

$O(n)$ preprocessing

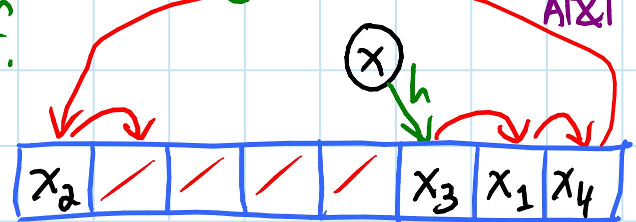
Dynamic: [Dietzfelbinger, Karlin, Mehlhorn, Meyer auf der Heide, Rohnert, Tarjan — SICOMP 1994]

- maintain C_t table size $\in [\frac{1}{4}, 4] \cdot c \cdot C_t^2$
  - double/halve table if C_t changes a lot
  - charge linear cost to linear # updates
- if space > $c \cdot n$: rebuild entire table
  - $\Pr\{\text{happening}\} = O(1/n)$ [Markov]
  - $\Rightarrow$ expected $O(1)$ cost per update

$\Rightarrow$ $O(1)$ deterministic query
$O(1)$ expected update → w.h.p. possible!
$O(n)$ space [Dietzfelbinger & Meyer auf der Heide - ICALP 1990]

→ 10% slower than memory access (Pătraşcu AT&T)

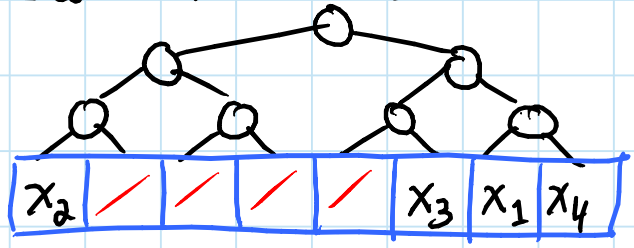
Linear probing: great cache perf.



- insert(x) puts x at first available slot $[h(x)+i] \bmod m$
- need $m \geq (1+\varepsilon) \cdot n$ (not just $m = \Omega(n)$)
- totally random $h \Rightarrow O(1/\varepsilon^2)$ expected time/op. (& $(1+\varepsilon)n$ space) [Knuth — TR 1962]
- $O(\lg n)$-wise independent $\Rightarrow$ constant expected [Schmidt & Siegel — STOC 1990]
- 5-wise independent $\Rightarrow$ constant expected [& pairwise $\Rightarrow \Omega(\lg n)$] [Pagh, Pagh, Ružić — STOC 2007]
- some 4-wise independent hashes $\Rightarrow \Omega(\lg n)$ expected [Pătraşcu & Thorup - ICALP 2010]
- simple tabulation hashing $\Rightarrow O(1/\varepsilon^2)$ expected [Pătraşcu & Thorup — STOC 2011]

Proof that totally random $h \Rightarrow O(1)$ expected
[Pagh, Pagh, Ružić – STOC 2007] (cf. Pătraşcu)

- totally random hash $h \Rightarrow$ balls in bins
- assume here $m = 3n$
- build perfect binary tree on leaves = slots



- call node of height h dangerous if $> \frac{2}{3} \cdot 2^h = 2\mu$ keys hash within node (via h)
  - $\Pr\{>2\mu\} \le e^{\mu}/2^{2\mu} = (e/4)^{2^h/3}$ [Chernoff]
- consider run in table of length $\in [2^\ell, 2^{\ell+1})$
  - look at nodes of height $h = \ell - 3$ spanning run
    ↳ between 8 & 17 of them
  - first 4 nodes span $> 3 \cdot 2^h$ slots of the run
  - these keys must hash within the 4 nodes (via h)
  - if nodes not dangerous: $\le 4 \cdot \frac{2}{3} \cdot 2^h = \frac{8}{3} \cdot 2^h$ keys hash within the nodes

  $\Rightarrow \ge 1$ node dangerous

$\Rightarrow \Pr\{\text{length of run} \ni x \text{ has length} \in [2^\ell, 2^{\ell+1})\}$
$\le 17 \cdot \Pr\{\text{node of height } \ell - 3 \text{ is dangerous}\}$
$\le 17 \cdot (e/4)^{2^\ell/24}$

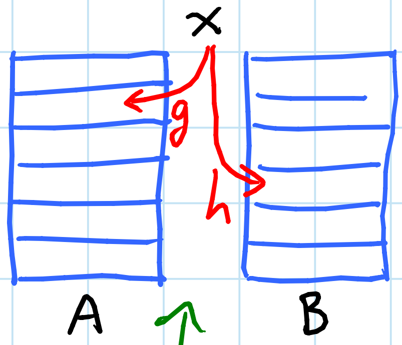
- $E[\text{length of run} \ni x] = \Theta\left(\sum_\ell 2^\ell \cdot \underbrace{\Pr\{\text{len.} \in [2^\ell, 2^{\ell+1})\}}_{1/\text{doubly exponential}}\right)$
$= \Theta(1)$

□

- cache of $\lg^{1+\varepsilon} n \Rightarrow O(1)$ amortized w.h.p. [Pat 11]
  - for batch of $\lg^{1+\varepsilon} n$, $E[\#\text{dangerous @ height } h] = \lg^{1+\varepsilon} n / c^{2^h}$

  $\Rightarrow$ for $h \le \lg \lg n$, $O(\text{that})$ w.h.p.; $h > \lg \lg n \Rightarrow$ not dang. w.h.p.

Cuckoo hashing: [Pagh & Rodler – J. Alg. 2004]



think of as bipartite graph

- 2 tables of size $m \geq (1+\varepsilon) \cdot n$
- 2 hash functions ($g \to A$, $h \to B$)
- query(x): check $A[g(x)]$ & $B[h(x)]$
- insert(x): put in $A[g(x)]$ or $B[h(x)]$
    - if kicked out y from $A[g(y)]$: move to $B[h(y)]$
    - etc.
    - if stuck: rehash entire structure

- $(2+\varepsilon)n$ space
- 2 deterministic probes for query
- fully random or $\Theta(\lg n)$-wise independence $\Rightarrow$ $O(1)$ expected update & $O(1/n)$ failure prob. [PR04]
  construction on n keys
- some 6-wise independent hash functions fail w.h.p. even if $m = n^{1+\varepsilon}$ [Cohen & Kane – manuscript 2009]
- simple tabulation hashing $\Rightarrow$ fail with prob. $\Theta(n^{-1/3})$ $\Rightarrow$ $\Theta(n^{4/3})$ inserts OK [Pătraşcu & Thorup – STOC 2011]

Proof that totally random hash functions ⇒
$\Pr\{\text{follow a path of length } k\} \le 1/2^k$
[PT11]   [Pătrașcu – blog, Feb. 2, 2010]

- assume $m = 2n$
- implied by existence of encoding of g & h in $2n \lg m - k$ bits:
  - does path start in A or B? 1 bit
  - slots of nodes along path: $(k+1) \lg m$ bits
  - keys of edges along path: $(k-1) \lg n$ bits
  - rest of g & h: $(n-k) 2 \lg m$ bits
  - total:

$$2n \lg m - k + O(\lg k) \text{ bits}$$

1 bit savings

□

similar proofs for cycles, ⊖, etc.