Today: Hashing
    - review:
      - dictionaries
      - chaining
      - simple uniform
    - universal hashing
      - why (useful)
      - how
    - perfect hashing
      - how
      - why (it works)

Dictionary problem: Abstract Data Type (ADT)
maintain set of items, each with a key, subject to
    - insert(item): add item to set
    - delete(item): remove item from set
    - search(key): return item with key if it exists
    - assume items have distinct keys (or that inserting new one clobbers old)
    - easier than predecessor/successor problem
      solved by AVL/2-3 trees/skip lists
      & by van Emde Boas
Hashing [6.006]
- goal: $O(1)$ time per operation & $O(n)$ space
- $u =$ # keys over all possible items
- $n =$ # keys/items currently in table
- $m =$ # slots in table
- hashing with chaining achieves $\Theta(1+\alpha)$ time per op.

Assuming simple uniform hashing:
$$\Pr_{k_1 \neq k_2} \{ h(k_1) = h(k_2) \}^3 = \frac{1}{m}$$
which you'd expect if totally uniform.

- requires assuming input keys are random
- only works in average case
  (like Basic Quicksort)

We will remove this unreasonable assumption.

Etymology:
- English ‘hash’ (1650s) = cut into small pieces
- French ‘hacher’ = chop up
- Old French ‘hache’
  (cf. English ‘hatchet’)
- Vulcan ‘la’ash’ = axe
Universal hashing:
- choose a random hash function \( h \) from \( \mathcal{H} \)
- require \( \mathcal{H} \) to be a universal hash family:
  \[ \Pr_{h \in \mathcal{H}} \{ h(k) = h(k') \} \leq \frac{1}{m} \text{ for all } k \neq k' \]
- now just assuming \( h \) is random
- no assumption about input keys (like Randomized Quicksort)

**Theorem:** for \( n \) arbitrary distinct keys & for random \( h \in \mathcal{H} \) & \( \mathcal{H} \) universal
\[ E[\# \text{ keys colliding in a slot}] \leq 1 + \alpha \]

**Proof:** consider keys \( k_1, k_2, \ldots, k_n \)
- let \( I_{i,j} = \begin{cases} 1 & \text{if } h(k_i) = h(k_j) \\ 0 & \text{else} \end{cases} \)

\[ E[\# \text{ keys hashing to same slot as } k_i] = E \left[ \sum_{j=1}^{n} I_{i,j} \right] = \sum_{j=1}^{n} E[I_{i,j}] = \sum_{j=1}^{n} E[I_{i,j}] + E[I_{i,i}] = \Pr \{ I_{i,j} = 1 \} = \Pr \{ h(k_i) = h(k_j) \} \leq \frac{1}{m} \leq \frac{n}{m} + 1 \]

\[ \Rightarrow \text{Insert, Delete, Search cost } O(1 + \alpha) \text{ expected.} \]
Do universal hash families exist? YES:
\[ \mathcal{H} = \{ \text{all hash functions} \} \]
\[ h: \{0, 1, \ldots, u-1\} \rightarrow \{0, 1, \ldots, n-1\} \]
is universal
... but this is useless:
- storing \( h \) takes \( \lg(m) = \lg m \) bits \( \gg n \)
  \( \sim \) just like direct map table (big array)
- would need to precompute \( u \) values
  \( \Rightarrow \Omega(u) \) time, possibly \( \omega(# \text{ operations}) \)

\textbf{Dot-product hash family:}
- assume \( m \) is prime (find nearby prime)
- assume \( u = m^r \) for integer \( r \) (round up else)
- view keys in base \( m \): \( k = \langle k_0, k_1, \ldots, k_{r-1} \rangle \)
- for key \( a = \langle a_0, a_1, \ldots, a_{r-1} \rangle \)
  define \( h_a(k) = \sum_{i=0}^{r-1} a_i \cdot k_i \mod m \)
  dot product
  \( \sum_{i=0}^{r-1} a_i \cdot k_i \mod m \)
  \( \mathcal{H} = \{ h_a \mid a \in \{0, 1, \ldots, u-1\} \} \)

- storing \( h_a \in \mathcal{H} \) requires just storing 1 key, a
  word RAM model: manipulating \( O(1) \)
  machine words takes \( O(1) \) time,
  & “objects of interest” (here: keys)
  fit in a machine word
  \( \Rightarrow \) computing \( h_a(k) \) takes \( O(1) \) time
  \( O(\lg m \cdot u) \) using just + & * ~ can you do better?\]
Theorem: dot-product hash family $\mathcal{H}$ is universal

Proof: take any two keys $k \neq k'$
⇒ differ in some digit, say $k_d \neq k'_d$
- let $\not{d} = \{0, 1, \ldots, r-1\} \setminus \{d\}$

\[
\Pr_a \{ h_a(k) = h_a(k') \} \leq \frac{\sum_{i=0}^{r-1} a_i \cdot k_i}{\sum_{i=0}^{r-1} a_i \cdot k'_i \pmod{m}}
\]

\[
= \Pr_a \{ \sum_{i \neq d} a_i \cdot k_i + a_d \cdot k_d = \sum_{i \neq d} a_i \cdot k'_i + a_d \cdot k'_d \pmod{m} \}
\]

\[
= \Pr_a \{ \sum_{i \neq d} a_i (k_i - k'_i) + a_d (k_d - k'_d) = 0 \pmod{m} \}
\]

\[
= \Pr_a \{ a_d = - (k_d - k'_d)^{-1} \sum_{i \neq d} a_i (k_i - k'_i) \pmod{m} \}
\]

\[
= \mathbb{E}_{a_{\not{d}}} \left[ \Pr_a \{ a_d = f(k, k', a_{\not{d}}) \} \right]
\]

(by $a_d$ is independent from $a_{\not{d}}$)

\[
= \mathbb{E}_{a_{\not{d}}} \left[ \frac{1}{m} \right]
\]

\[
= \frac{1}{m}
\]

Another universal hash family: [CLRS]
- choose prime $p \geq u$ (once)
- $h_{ab}(k) = [(a \cdot k + b) \pmod{p}] \pmod{m}$
- $\mathcal{H} = \{ h_{ab} \mid a, b \in \{0, 1, \ldots, u-1\} \}$
Static dictionary problem: given n keys to store in table, support Search(k)
- no collisions

Perfect hashing: [Fredman, Komlós, Szemerédi 1984]
- polynomial build time w.h.p. (nearly linear)
- O(1) time for Search in worst case
- O(n) space in worst case

Idea: 2-level hashing

1. pick $h_1 : \mathbb{Z}_0, 1, \ldots, m-1 \rightarrow \mathbb{Z}_0, 1, \ldots, m-1$ from a universal hash family
   for $m = \Theta(n)$ (e.g. nearby prime)
   - hash all items with chaining using $h_1$

2. for each slot $j \in \mathbb{Z}_0, 1, \ldots, m-1$:
   - let $l_j = \# \text{ items in slot } j = \{ i \mid h(k_i) = j \}$
   - pick $h_{2,j} : \mathbb{Z}_0, 1, \ldots, m-1 \rightarrow \mathbb{Z}_0, 1, \ldots, m_j$ from a universal hash family
     for $l_j^2 \leq m_j \leq O(l_j^3)$ (e.g. nearby prime)
   - replace chain in 1 slot $j$ with hashing-with-chaining using $h_{2,j}$

Space = $O(n + \sum_{j=0}^{m-1} l_j^2)$
- to guarantee space = $O(n)$:

1.5 if $\sum_{j=0}^{m-1} l_j^2 > cn$ then redo step 1
\[ \text{Search time} = O(1) \text{ for first table } (h_1) \\
+ O(\text{max chain size in second table}) \]
\[ \rightarrow \text{to guarantee} = O(1): \]
\[ \text{⇒ ANY collision} \]
\[ \text{⇒ no collisions at second level!} \]

\underline{\text{Build time: } ①\&② \text{ are } O(n), \ ①5 \& ②5?} \]

\[ \text{②.5: } \Pr \left\{ h_{a, j}(k_i) = h_{a, j}(k_{i'}) \text{ for some } i \neq i' \right\} \]
\[ \leq \frac{\sum_{i \neq i'} \Pr \left\{ h_{a, j}(k_i) = h_{a, j}(k_{i'}) \right\}}{\binom{n}{2}} \]
\[ \leq \left( \frac{n}{2} \right) \cdot \frac{1}{l_j^2} \]
\[ \leq \frac{1}{2} \]
\[ \Rightarrow \text{each trial is like a coin flip, tails} \Rightarrow \text{OK} \]
\[ \Rightarrow E[\text{# trials}] \leq 2 \]
\[ \& \ \# \text{trials} = O(\log n) \text{ w.h.p. (by Lecture 7)} \]

- Chernoff bound \( \Rightarrow l_j = O(\log n) \text{ w.h.p.} \)
- each trial \( O(\log n) \text{ time} \) (also obviously \( O(n) \))
- must do this for each \( j \)
\[ \Rightarrow O(n \log^2 n) \text{ time w.h.p. (or obviously } O(n^2 \log n)) \]
\[ E\left[ \sum_{j=0}^{m-1} l_j^2 \right] = E\left[ \sum_{i=1}^{n} \sum_{i'=1}^{n} \text{ind}_{i, i'} \right] \]

indicator rand. var. = \[\begin{cases} 1 & \text{if } h_1(k_i) = h_2(k_i) \\ 0 & \text{else} \end{cases} \]

\[ = \sum_{i=1}^{n} \sum_{i'=1}^{n} E[\text{ind}_{i, i'}] \leq \text{linearity of expectation} \]

\[ = \sum_{i=1}^{n} E[\text{ind}_{i, i}] + 2 \sum_{i \neq i'} E[\text{ind}_{i, i'}] \]

\[ \leq n + 2 \binom{n}{2} \cdot \frac{1}{m} \leq \text{universality} \]

\[ = O(n) \quad \text{because} \quad m = \Theta(n) \]

\[ \Pr\left\{ \sum_{j=0}^{m-1} l_j^2 \geq c \cdot n^2 \right\} \leq \frac{E\left[ \sum_{j=0}^{m-1} l_j^2 \right]}{c \cdot n} \]

\[ \leq \frac{1}{2} \quad \text{for suff. large const. } c \]

\[ \Rightarrow E[\# \text{ trials}] \leq 2 \]

\& \# \text{ trials} = O(lg n) \quad \text{w.h.p.} \]

\Rightarrow \circ \& \circ \circ \text{ take } O(n lg n) \quad \text{w.h.p.}