Introduction to Algorithms
6.046J/18.410J

Lecture 17
Prof. Piotr Indyk

Introduction to Algorithms
Menu

• Hashing recap
• Universal hashing
• Perfect hashing
Hash functions

A *hash function* $h$ maps the universe $U$ of all keys into $\{0, 1, \ldots, m-1\}$:

We want $h$ that maps $n$ keys $K$ to $T$, trying to minimize the number of collisions.
Search cost

Expected time to search for a record with a given key = $\Theta(1 + \alpha)$, where $\alpha = \frac{n}{m}$

- apply hash function and access slot
- search the list

Expected search time = $\Theta(1)$ if $\alpha = O(1)$, assuming uniform hashing
Uniform hashing assumption: Each key $k \in K$ of keys is equally likely to be hashed to any slot of table $T$, independent of where other keys are hashed.

- Ideal properties
- How do you compute/store?
- In practice, people often used deterministic (and really weird looking) hash functions
Universal hashing
Universal hashing

• Let $H$ be a finite collection of hash functions, each mapping $U$ to \{0, 1, \ldots, m-1\}

• We say $H$ is *universal* if for all $x, y \in U$, where $x \neq y$, we have

$$\Pr_{h \in H}\{h(x) = h(y)\} = \frac{1}{m}.$$
Universality is good

**Theorem.** Let $h$ be a hash function chosen (uniformly) at random from a universal set $H$ of hash functions. Suppose $h$ is used to hash $n$ arbitrary keys into the $m$ slots of a table $T$. Then, for a given key $x$, we have

$$E[\#\text{collisions with } x] < \frac{n}{m}.$$
Constructing a set of universal hash functions

• Let $m$ be prime.

• Decompose key $k$ into $r + 1$ digits, each with value in the set \{0, 1, \ldots, m–1\}.

• That is, let $k = \langle k_0, k_1, \ldots, k_r \rangle$, where $0 \leq k_i < m$.

**Randomized strategy:**

• Pick $a = \langle a_0, a_1, \ldots, a_r \rangle$ where each $a_i$ is chosen randomly from \{0, 1, \ldots, m–1\}.

• Define $h_a(k) = \sum_{i=0}^{r} a_i k_i \mod m$

• Denote $H = \{ h_a : a \text{ as above} \}$
Universality of dot-product hash functions

**Theorem.** The set $H = \{h_a\}$ is universal.

**Proof.** Suppose that

$$x = \langle x_0, x_1, \ldots, x_r \rangle$$

and

$$y = \langle y_0, y_1, \ldots, y_r \rangle$$

are distinct keys. Thus, they differ in at least one digit position, wlog position 0. What is the probability that $x$ and $y$ collide, that is $h_a(x) = h_a(y)$?
Proof (continued)

\[ h_a(x) = h_a(y) \iff \sum_{i=0}^{r} a_i x_i = \sum_{i=0}^{r} a_i y_i \pmod{m} \]

\[ \sum_{i=0}^{r} a_i (x_i - y_i) \equiv 0 \pmod{m} \]

\[ a_0(x_0 - y_0) + \sum_{i=1}^{r} a_i (x_i - y_i) \equiv 0 \pmod{m} \]

\[ a_0(x_0 - y_0) \equiv -\sum_{i=1}^{r} a_i (x_i - y_i) \pmod{m} \]
Proof (finish)

We have

\[ a_0(x_0 - y_0) \equiv -\sum_{i=1}^{r} a_i(x_i - y_i) \pmod{m}, \]

and since \( x_0 \neq y_0 \), an inverse \( (x_0 - y_0)^{-1} \) must exist, which implies that

\[ a_0 \equiv \left(-\sum_{i=1}^{r} a_i(x_i - y_i)\right) \cdot (x_0 - y_0)^{-1} \pmod{m}. \]

Thus, for any choices of \( a_1, a_2, \ldots, a_r \), exactly one choice of \( a_0 \) causes \( x \) and \( y \) to collide. The probability of making this choice is \( 1/m \).
Recap

- Showed how to implement dictionary so that SEARCH works in expected constant time under the universal hashing
- Constructed universal hashing for keys in \( \{0\ldots m^{r+1} - 1\}\) that can be evaluated in \(O(r)\) time
Perfect hashing
Perfect hashing

Given a set of $n$ keys, construct a static hash table of size $m = O(n)$ such that SEARCH takes $\Theta(1)$ time in the worst case.

Two-level scheme with universal hashing at both levels.

No collisions on level 2!
Collisions at level 2

**Theorem.** Let $\mathcal{H}$ be a class of universal hash functions for a table of size $m = n^2$. Then, if we use a random $h \in \mathcal{H}$ to hash $n$ keys into the table, the expected number of collisions is at most $1/2$.

**Proof.**

- By the definition of universality, the probability that 2 given keys in the table collide under $h$ is $1/m = 1/n^2$
- Since there are “$n$ choose 2” pairs of keys that can possibly collide, the expected number of collisions is

\[
\binom{n}{2} \cdot \frac{1}{n^2} = \frac{n(n-1)}{2} \cdot \frac{1}{n^2} < \frac{1}{2}.
\]
No collisions at level 2

Corollary. The probability of no collisions is at least $1/2$.

Proof. Markov’s inequality says that for any nonnegative random variable $X$, we have

$$\Pr\{X \geq t\} \leq \frac{E[X]}{t}.$$ 

Applying this inequality with $t = 1$, we find that the probability of 1 or more collisions is at most $1/2$. We can quickly find a hash function with no collisions while constructing the data structure and use it from now on.
Analysis of storage

- For the level-1 hash table $T$, choose $m = n$, and let $n_i$ be random variable for the number of keys that hash to slot $i$ in $T$.
- By using $n_i^2$ slots for the level-2 hash table $S_i$, the expected total storage required for the two-level scheme is

$$E \sum_{i=0}^{m-1} \left( n_i^2 \right)$$

© Charles Leiserson and Piotr Indyk

Introduction to Algorithms
How big is sum of squares?

- Observation: Let $C$ be the total number of collisions at the first level. Then
  \[ \sum_i n_i^2 = O(C) \]

- Suffice to estimate the expected number of collisions $E[C]$
Expected number of collisions

Proof.

• We have \( C = \sum_{x,y} c_{xy} \) where

\[
c_{xy} = \begin{cases} 
1 & \text{if } h(x) = h(y), \\
0 & \text{otherwise.} 
\end{cases}
\]

• Note that \( E[c_{xy}] = 1/m \). Therefore

\[
E[C] = E[\sum_{x,y} c_{xy}] = \sum_{x,y} E[c_{xy}] = O(n^2/m) = O(n)
\]

• We obtained perfect hashing in \( O(n) \) storage!