The Design and Analysis of Algorithms

Lecture 27 Randomized Algorithms II

Zhenbo Wang

Department of Mathematical Sciences, Tsinghua University

Content

Dictionary Data Type

- *Dictionary*. Given a universe U of possible elements, maintain a subset $S \subseteq U$ so that *inserting*, deleting, and *searching* in S is efficient.
- *Dictionary interface*.

  create(): initialize a dictionary with $S = \emptyset$.

  insert(u): add element $u \in U$ to S.

  delete(u): delete u from S (if u is currently in S).

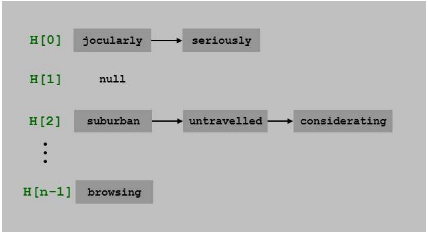
  lookup(u): is u in S?
- *Challenge*. Universe U can be extremely large so defining an array of size $|U|$ is infeasible.
- *Applications*. File systems, databases, Google, compilers, checksums P2P networks, associative arrays, cryptography, web caching, etc.

Hashing

- *Hash function*. $h : U \to \{0, 1, \cdots, n-1\}$.
- *Hashing*. Create an array H of size n. When processing element u, access array element $H[h(u)]$.
- *Collision*. When $h(u) = h(v)$ but $u \neq v$.

  A collision is expected after $\Theta(\sqrt{n})$ random insertions (Birthday Problem).

  Separate chaining: $H[i]$ stores linked list of elements u with $h(u) = i$.



Hashing Performance

- *Ideal hash function*. Maps m elements *uniformly at random* to n hash slots.

  Running time depends on length of chains.

  Average length of chain $= \alpha = m/n$.

  Choose $n \approx m \Rightarrow$ on average $O(1)$ per insert, lookup, or delete.

- *Challenge*. Achieve idealized randomized guarantees, but with a hash function where you can easily find items where you put them.

- *Approach*. Use randomization in the choice of h.

Universal Hashing

- *Universal family of hash functions*. [Carter-Wegman 1980s]

  For any pair of elements $u, v \in U$, $\Pr_{h \in H}[h(u) = h(v)] \leq 1/n$.

  Can select random h efficiently.

  Can compute $h(u)$ efficiently.

Ex. $U = \{a, b, c, d, e, f\}, n = 2$.

	a	b	c	d	e	f
$h_1(x)$	0	1	0	1	0	1
$h_2(x)$	0	0	0	1	1	1

$H = \{h_1, h_2\}$
$\Pr_{h \in H}[h(a) = h(b)] = 1/2$
$\Pr_{h \in H}[h(a) = h(c)] = 1$
$\Pr_{h \in H}[h(a) = h(d)] = 0$
...

not universal

	a	b	c	d	e	f
$h_1(x)$	0	1	0	1	0	1
$h_2(x)$	0	0	0	1	1	1
$h_3(x)$	0	0	1	0	1	1
$h_4(x)$	1	0	0	1	1	0

$H = \{h_1, h_2, h_3, h_4\}$
$\Pr_{h \in H}[h(a) = h(b)] = 1/2$
$\Pr_{h \in H}[h(a) = h(c)] = 1/2$
$\Pr_{h \in H}[h(a) = h(d)] = 1/2$
$\Pr_{h \in H}[h(a) = h(e)] = 1/2$
$\Pr_{h \in H}[h(a) = h(f)] = 0$
...

universal

Universal Hashing: Analysis

- *Proposition*. Let H be a universal family of hash functions; let $h \in H$ be chosen uniformly at random from H; and let $u \in U$. For any subset $S \subseteq U$ of size at most n, the expected number of items in S that collide with u is at most 1.

Pf. For any element $s \in S$, define indicator random variable $X_s = 1$ if $h(s) = h(u)$ and 0 otherwise.

Let X be a random variable counting the total number of collisions with u.

$$E_{h \in H}[X] = E[\sum_{s \in S} X_s] = \sum_{s \in S} \Pr[X_s = 1] \leq \sum_{s \in S} \frac{1}{n} = |S|\frac{1}{n} \leq 1. \square$$

Q. OK, but how do we design a universal class of hash functions?

Designing a Universal Family of Hash Functions

Theorem 1 (Chebyshev 1850)

There exists a prime between n and $2n$.

- *Modulus*. Choose a prime number $p \approx n$.
- *Integer encoding*. Identify each element $u \in U$ with a base-p integer of r digits: $x = (x_1, x_2, \cdots, x_r)$.
- *Hash function*. Let A = set of all r-digit, base-p integers. For each $a = (a_1, a_2, \cdots, a_r)$ where $0 \leq a_i < p$, define

$$h_a(x) = \sum_{i=1}^{r} a_i x_i \pmod{p}.$$

- *Hash function family*. $H = \{h_a : a \in A\}$.

Designing a Universal Family of Hash Functions

Theorem 2

$H = \{h_a : a \in A\}$ is a universal family of hash functions.

Pf. Choose a prime number $p \approx n$.

Let $x = (x_1, \cdots, x_r)$ and $y = (y_1, \cdots, y_r)$ be two distinct elements of U.

Since $x \neq y$, there exists an integer j such that $x_j \neq y_j$.

We have $h_a(x) = h_a(y)$ iff $a_j(y_j - x_j) = \sum_{i \neq j} a_i(x_i - y_i) \pmod{p}$.

Can assume a was chosen uniformly at random by first selecting all coordinates a_i where $i \neq j$, then selecting a_j at random. Thus, we can assume a_i is fixed for all coordinates $i \neq j$.

Since p is prime, $a_j z = m \pmod{p}$ has at most one solution among p possibilities.

Thus $\Pr[h_a(x) = h_a(y)] \leq 1/n$. $\square$

Number Theory Fact

Fact. Let p be prime, and let $z \neq 0 \pmod{p}$. Then $\alpha z = m \pmod{p}$ has at most one solution $0 \leq \alpha < p$.

Pf. Suppose α and β are two different solutions.

Then $(\alpha - \beta)z = 0 \pmod{p}$; hence $(\alpha - \beta)z$ is divisible by p.

Since $z \neq 0 \pmod{p}$, we know that z is not divisible by p; it follows that $(\alpha - \beta)$ is divisible by p.

This implies $\alpha = \beta$. $\square$

Chernoff Bounds

Theorem 3

Suppose $X_1, \cdots, X_n$ are independent $0-1$ random variables. Let $X = X_1 + \cdots + X_n$. Then for any $\mu \geq E[X]$ and for any $\delta > 0$, we have

$$Pr[X > (1+\delta)\mu] < \left(\frac{e^{\delta}}{(1+\delta)^{1+\delta}}\right)^{\mu};$$

for any $\mu \leq E[X]$ and for any $0 < \delta < 1$, we have

$$Pr[X < (1-\delta)\mu] < e^{-\delta^2\mu/2}.$$

Load Balancing

- *Load balancing*. System in which n jobs arrive in a stream and need to be processed immediately on n identical processors. Find an assignment that balances the workload across processors.
- *Centralized controller*. Each processor receives one job.
- *Decentralized controller*. Assign jobs to processors uniformly at random.

  How likely is it that some processor is assigned ”too many” jobs?

Load Balancing: Analysis

- Let X_i = number of jobs assigned to processor i.
- Let $Y_{ij} = 1$ if job j assigned to processor i, and 0 otherwise.
- We have $E[Y_{ij}] = 1/n$.
- Thus, $X_i = \sum_j Y_{ij}$, and $\mu = E[X_i] = 1$.
- Applying Chernoff bounds with $\delta = c - 1$ yields $\Pr[X_i > c] < \frac{e^{c-1}}{c^c}$.
- Let $\gamma(n)$ be number x such that $x^x = n$, and choose $c = e\gamma(n)$.

$$\Pr[X_i > c] < \frac{e^{c-1}}{c^c} < \left(\frac{e}{c}\right)^c = \left(\frac{1}{\gamma(n)}\right)^{e\gamma(n)} = (\frac{1}{n})^e < \frac{1}{n^2}$$

- Union bound $\Rightarrow$ with probability $\geq 1 - 1/n$ no processor receives more than $e\gamma(n) = \Theta(\log n / \log\log n)$ jobs.

BPP, RP and ZPP

BPP. Decision problems solvable with probability $\geq 2/3$ in poly-time.

RP. Decision problems solvable with one-sided error in poly-time.

If the correct answer is *no*, always return *no*.

If the correct answer is *yes*, return *yes* with probability $\geq 1/2$.

coRP. Complementary of *RP*.

ZPP. Decision problems solvable in *expected* poly-time.

BPP, RP and ZPP

Theorem 5

$$\begin{aligned}\mathbf{RP} &\subseteq \mathbf{BPP}\\ \mathbf{coRP} &\subseteq \mathbf{BPP}\\ \mathbf{ZPP} = \mathbf{RP} &\cap \mathbf{coRP}\end{aligned}$$

Theorem 6

$P \subseteq ZPP \subseteq BPP$, $RP \subseteq NP$.

- *Fundamental open questions*. To what extent does randomization help?
- Does $P = ZPP$? Does $ZPP = BPP$? Does $RP = NP$? Does $BPP = NP$?

Homework

- Read Chapter 13 of the textbook.
- Exercise 9 & 18 in Chapter 13.