INDIAN INSTITUTE OF TECHNOLOGY KHARAGPUR

Design and Analysis of Algorithms

MA21007/Assignment-3/Due: 11 Sep. 2015

Problem: 1

Suppose that a dynamic set S is represented by a direct-address table T of length m. Describe a procedure that f nds the maximum element of S. What is the worst-case performance of your procedure?

Problem: 2

Professor Marley hypothesizes that substantial performance gains can be obtained if we modify the chaining scheme so that each list is kept in sorted order. How does the professor's modification affect the running time for successful searches, unsuccessful searches, insertions, and deletions?

Problem: 3

Consider a version of the division method in which $h(k) = k \bmod m$, where $m = 2^p - 1$ and k is a character string interpreted in radix 2^p. Show that if string x can be derived from string y by permuting its characters, then x and y hash to the same value. Give an example of an application in which this property would be undesirable in a hash function.

Problem: 4 Slot-size bound for chaining

Suppose that we have a hash table with n slots, with collisions resolved by chain-ing, and suppose that n keys are inserted into the table. Each key is equally likely to be hashed to each slot. Let M be the maximum number of keys in any slot after all the keys have been inserted. Your mission is to prove an $O(\lg n / \lg \lg n)$ upper bound on $\mathrm{E}[M]$, the expected value of M.

a. Argue that the probability Q_k that exactly k keys hash to a particular slot is given by

$$Q_k = \left(\frac{1}{n}\right)^k \left(1 - \frac{1}{n}\right)^{n-k} \binom{n}{k}.$$

b. Let P_k be the probability that $M = k$, that is, the probability that the slot containing the most keys contains k keys. Show that $P_k \leq n Q_k$.

c. Use Stirling's approximation, to show that $Q_k < e^k / k^k$.

d. Show that there exists a constant $c > 1$ such that $Q_{k_0} < 1/n^3$ for $k_0 = c \lg n / \lg \lg n$. Conclude that $P_k < 1/n^2$ for $k \geq k_0 = c \lg n / \lg \lg n$.

e. Argue that

$$\mathrm{E}[M] \leq \Pr\left\{M > \frac{c \lg n}{\lg \lg n}\right\} \cdot n + \Pr\left\{M \leq \frac{c \lg n}{\lg \lg n}\right\} \cdot \frac{c \lg n}{\lg \lg n}.$$

Conclude that $\mathrm{E}[M] = O(\lg n / \lg \lg n)$.

Problem: 5 Pattern Matching

Principal Skinner has a problem: he is absolutely *sure* that Bart Simpson has plagiarized some text on a recent book report. One of Bart's sentences sounds oddly familiar, but Skinner can't quite figure out where it came from. Skinner decides to see if some smart-alec IIT student can help him out.

Skinner gives you a DVD containing the full text of the Springfield public library. The data is stored in a ***binary string*** $T[1], T[2], \ldots, T[n]$, which we view as an array $T[1\,..\,n]$, where each $T[i]$ is either 0 or 1. Skinner also gives you the quote from Bart Simpson's book report, a shorter binary string $P[1\,..\,m]$, again where each $P[i]$ is either 0 or 1, and where $m < n$. For a binary string $A[1\,..\,k]$ and for integers i, j with $1 \le i \le j \le k$, we use the notation $A[i\,..\,j]$ to refer to the binary string $A[i], A[i+1], \ldots, A[j]$, called a ***substring*** of A. The goal of this problem is to determine whether P is a substring of T, i.e., whether $P = A[i\,..\,j]$ for some i, j with $1 \le i \le j \le n$.

For the purpose of this problem, assume that you can manipulate $O(\log n)$-bit integers in constant time. For example, if $x \le n^7$ and $y \le n^5$, then you can calculate $x + y$ in constant time. On the other hand, you may not assume that m-bit integers can be manipulated in constant time, because m may be too large. For example, if $m = \Theta(\log^2 n)$ and x and y are each m-bit integers, you *cannot* calculate $x + y$ in constant time. (In general, it is reasonable to assume that you can manipulate integers of length logarithmic in the input size in constant time, but larger integers require proportionally more time.)

(a) Assume that you have a hash function $h(x)$ that computes a hash value of the m-bit binary string $x = A[i\,..\,(i+m-1)]$, for some binary string $A[1\,..\,k]$ and some $1 \le i \le k-m+1$. Moreover, assume that the hash function is perfect: if $x \ne y$, then $h(x) \ne h(y)$. Assume that you can calculate the hash function in $O(m)$ time. Show how to determine whether P is a substring of T in $O(mn)$ time.

(b) Consider the following family of hash functions h_p, parameterized by a prime number p in the range $[2, cn^4]$ for some constant $c > 0$:

$$h_p(x) = x \pmod{p} .$$

Assume that p is chosen uniformly at random among all prime numbers in the range $[2, cn^4]$. Fix some i with $1 \le i \le n-m+1$, and let $x = T[i\,..\,(i+m-1)]$. Show that, for an appropriate choice of c, and if $x \ne P$, then

$$\Pr_p \left\{ h_p(x) = h_p(P) \right\} \le \frac{1}{n} .$$

Hint: Recall the following two number-theoretic facts: (1) an integer x has at most $\lg x$ prime factors; (2) the Prime Number Theorem: there are $\Theta(x/\lg x)$ prime numbers in the range $[2, x]$.

(c) How long does it take to calculate $h_p(x)$, as defined in part (b)? *Hint:* Notice that x is an m-bit integer, and hence cannot be manipulated in constant time.

(d) For $1 \leq i \leq n - m$, show how to calculate $h_p(A[(i+1)\,..\,(i+m)])$ in constant time if you already know the value of $h_p(A[i\,..\,(i+m-1)])$, as defined in part (b)?

(e) Using the family of hash functions from part (b), devise an algorithm to determine whether P is a substring of T in $O(n)$ expected time.

Problem: 6 2-Universal Hashing

Let $\mathcal{H}$ be a class of hash functions in which each $h \in \mathcal{H}$ maps the universe U of keys to $\{0, 1, \ldots, m-1\}$. We say that $\mathcal{H}$ is **2-*universal*** if, for every fixed pair $\langle x, y\rangle$ of keys where $x \neq y$, and for any h chosen uniformly at random from $\mathcal{H}$, the pair $\langle h(x), h(y)\rangle$ is equally likely to be any of the m^2 pairs of elements from $\{0, 1, \ldots, m-1\}$. (The probability is taken *only* over the random choice of the hash function.)

(a) Show that, if $\mathcal{H}$ is 2-universal, then it is universal.

(b) Construct a specific family $\mathcal{H}$ that is universal, but not 2-universal, and justify your answer. Write down the family as a table, with one column per key, and one row per function. Try to make m, $|\mathcal{H}|$, and $|U|$ as small as possible.

Hint: There is an example with m, $|\mathcal{H}|$, and $|U|$ all less than 4.

(c) Suppose that an adversary knows the hash family $\mathcal{H}$ and controls the keys we hash, and the adversary wants to force a collision. In this problem part, suppose that $\mathcal{H}$ is universal. The following scenario takes place: we choose a hash function h randomly from $\mathcal{H}$, keeping it secret from the adversary, and then the adversary chooses a key x and learns the value $h(x)$. Can the adversary now force a collision? In other words, can it find a $y \neq x$ such that $h(x) = h(y)$ with probability greater than $1/m$?

If so, write down a particular universal hash family in the same format as in part (b), and describe how an adversary can force a collision in this scenario. **If not**, prove that the adversary cannot force a collision with probability greater than $1/m$.

(d) Answer the question from part (c), but supposing that $\mathcal{H}$ is 2-universal, not just universal.