Problem Set 3

MIT students: This problem set is due in recitation on **Friday, October 7, 2005.** The homework lab for this problem set will be held 7–9 P.M. on Wednesday, October 5, 2005.

SMA students:

Reading: Sections 11.1–11.3 and Section 11.5.

Both exercises and problems should be solved, but *only the problems* should be turned in. Exercises are intended to help you master the course material. Even though you should not turn in the exercise solutions, you are responsible for material covered in the exercises.

Mark the top of each sheet with your name, the course number, the problem number, your recitation section, the date and the names of any students with whom you collaborated.

You will often be called upon to "give an algorithm" to solve a certain problem. Your write-up should take the form of a short essay. A topic paragraph should summarize the problem you are solving and what your results are. The body of the essay should provide the following:

1. A description of the algorithm in English and, if helpful, pseudo-code.
2. At least one worked example or diagram to show more precisely how your algorithm works.
3. A proof (or indication) of the correctness of the algorithm.
4. An analysis of the running time of the algorithm.

Remember, your goal is to communicate. Full credit will be given only to correct solutions *which are described clearly*. Convoluted and obtuse descriptions will receive low marks.

Exercise 3-1. Do Exercise 11.1-1 on page 222 in CLRS.

Exercise 3-2. Do Exercise 11.2-3 on page 229 in CLRS.

Exercise 3-3. Do Exercise 11.3-3 on page 236 in CLRS.

Exercise 3-4. Do Problem 11-2 on page 250 in CLRS.

Problem 3-1. 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 MIT 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 $1 \le i \le n - m$, 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 3-2. 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 $x \ne y$, 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 $y \ne x$ 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.