Homework 3

Fundamental Algorithms, Fall 2001, Professor Yap

- DUE: Mon Oct 22, in class.
- NOTICE: The solution for this homework will be published immediately after due date, in preparation for midterm.

1. Exercise 7.4-5, page 159. [**5+10+10+5 POINTS**]

   We will break down this question into subparts as follows. First, let us call the variant of Quicksort in this exercise the **k-Truncated Quicksort**, for any $k \geq 2$. In this variant, we stop the recurrence when the size of the subarray to be sorted is $< k$. The standard quicksort is the case $k = 2$. Insertion sort is a very simple algorithm described in chapter 2 (p.15).
   (i) Let the array A contain the output of the k-Truncated Quicksort (A is also going to be the input for our insertion sort). Assume the input numbers are distinct. What can you say about the ordering in A, without knowing anything about how Quicksort makes its random choices?
   (ii) Argue that the expected time of the k-Truncated Quicksort algorithm is $O(n \log(n/k))$.
   (iii) Argue that the worst case time of insertion sort on the output of the k-Truncated Quicksort algorithm is $O(kn)$.
   (iv) Suppose the time for (ii) is $C_1 n \log(n/k)$ and the time for (iii) is $C_2 kn$. Describe how you would chose k to minimize the sum

   $$T(n) = C_1 n \log(n/k) + C_2 kn.$$

2. Exercise 9.1-1, page 185. [**10 POINTS**]

   Find the second largest in at most $n + \lceil \lg n \rceil - 2$ comparisons. HINT: set up a binary tree T with n leaves and height $\lceil \lg n \rceil$. Place the numbers in leaves of T and use it as a "tournament" to compute the largest. Now, ask yourself what information do you already possess about the next largest. Then, find it with the help of T.

3. Exercise 9.3-9, page 193. [**10 POINTS**]

   Please help Professor Olay with his oil problem.

4. Median analysis. [**10+20 POINTS**]

   Suppose that we have a version of the median algorithm whose recurrence is

   $$T(n) = n + T(c_1 n) + T(c_2 n)$$

   for some $0 < c_1, c_2 < 1$. For instance, in the book, the version we saw was $c_1 = 1/5$ and $c_2 = 7/10$.

(i) Show that if $c_1 + c_2 < 1$, we will have the solution $T(n) = \Theta(n)$. NOTE: you must show two facts: $T(n) = O(n)$ and $T(n) = \Omega(n)$. But one of them is "trivial" (but you still need to note it in your answer).

(ii) Suppose $c_1 + c_2 = 1$. Guess what $T(n)$ should be, and try to prove the best upper and lower bound you can.

HINT: For both (i) and (ii), use an induction argument. To guess the correct value of $T(n)$, think of the Master theorem. For instance, suppose you guessed that $T(n) \leq Cn^3$ for $n \geq n_0$ large enough. Attempt an induction proof, keeping C, n_0 undetermined (the proof will suggest what C, n_0 should be).

5. Exercise 11.3-2, page 236. [**10 POINTS**]

   Implementing the division method for a radix-128 number.

6. Perfect Hashing (Section 11.5, page 245). [**20+10+15 POINTS**]

   This exercise is to help you understand perfect hashing. Let $U = \mathbb{Z}_p$ for some prime p, and we are given a set $K \subseteq U$ of n keys to be stored. We use a 2-level table structure: a primary table of size n and secondary tables whose *total* size is at most $4n$. Each hash function has the form

$$h(x) = (((ax + b) \bmod p) \bmod m) \tag{1}$$

   where m is the table size, and a, b are the parameters determining h. Here the prime number p and n are fixed throughout. Let us go into implementation details.

   Suppose the primary hash function $\widehat{h} : K \to \mathbb{Z}_n$ is fixed (in (1), it means you have chosen the constants a and b). For each $i \in \mathbb{Z}_n$, let $B_i = \widehat{h}^{-1}(i) = \{k \in K : \widehat{h}(k) = i\}$ be the ith bucket. Let $n_i = |B_i|$ and we choose $m_i = n_i^2$. Suppose also that the secondary hash functions,

$$h_i : B_j \to \mathbb{Z}_{m_i}$$

   have also been chosen for each i. In particular,

$$h_i(x) = (((a_i x + b_i) \bmod p) \bmod m_i)$$

   for for some a_i, b_i, m_i.

   (i) How would you implement this hashing scheme? There are two parts in answering this question: describe your data structure, and describe the "Hash Search" algorithm which uses this data structure. (In class, we call the Hash Search algorithm the "LookUp" algorithm in class.) ASSUMPTION: each $k \in U$ can be stored in one entry of your array. PLEASE make explicit any other reasonable assumptions you need.

   HINTS: Where would you store the constants a_i, b_i, m_i, and how would you organize the secondary tables. We suggest using two arrays: array T for the primary table, and an array S for *all* the secondary tables.

(ii) Suppose your keys are ASCII strings of arbitrary length. So we no longer assume that they fit into a single array entry. Continue to assume that $n = |K|$ can fit into an array entry. How would you modify the above scheme?

(iii) In part (i) we assume the hash functions were given to you. But how can we find a suitable $\widehat{h}$? Here is the method: pick randomly $a, b \in \mathbb{Z}_p$ where $a \neq 0$ as the parameters for the function $\widehat{h}$. Then check if

$$\sum_{i=0}^{n-1} \left|\widehat{h}^{-1}(i)\right|^2 < 2n.$$

If so, we have passed the "test" and we are done. If not, we repeat the test with another random choice of a, b.

Question A: how may times do you expect to do this test until it is passed? Question B: Give more details about how you would implement this test, and say how much time is needed for each test.

ADDITIONAL QUESTIONS (not graded, but we will sketch answers)

1. Exercise 7.4-2, page 159. Show that the best case time for Quicksort is $\Omega(n \log n)$.
2. Exercise 9.3-3, page 192. Show how to make Quicksort run in $O(n \log n)$ time in the WORST case. HINT: use a median algorithm somewhere.
3. Exercise 11.3-4, page 236. On the multiplication method.
4. Exercise 11-1, page 249. Longest probe for hashing.