Algorithms First Examination

Oct. 26, 2010

(10:10 am ~ 12:30 am)

1. (10%) Please complete the following table. (10%)

	Insertion_sort	Selection_sort	Merge_sort	Heap_sort	Quick_sort
Average case		$O(N^2)$			
Worst case	$O(N^2)$				
Stable or not					

2. (5%) (a) What is the smallest possible depth of a leaf in a decision tree for a comparison sort?

   (5%) (b) Please explain in what situations using counting sort is better than comparison sort?

3. (5%) (a) Please explain what is the lower bound of a problem complexity?

   (5%) (b) Explain why the statement, "The running time of algorithm A is at least $O(n \lg n)$." is meaningless.

4. (10%) Give tight asymptotic complexity of $T(n)$, and show your steps:

   (a) $T(n) = 2T(\sqrt{n}) + \lg n$

   (b) $T(n) = 2T(n/2 + 20) + n$

5. (10%) What is the running time of quick sort algorithm when all elements of array A have the same value? How to improve the performance of quick sort algorithm when there are many elements have the same value?

6. (10%) Describe a method to compute the first kth smallest elements of a set of n distinct integers in $O(n + k \lg k)$ time.

7. (10%) Let $p(n) = \sum_{i=0}^{d} a_i n^i$, where $a_d > 0$, be a degree-d polynomial in n, and let k be a constant. Use the definitions of the asymptotic notation to prove the following property. If $k = d$, then $p(n) = \Theta(n^k)$.

8. (10%) Describe a $\Theta(n \lg n)$-time algorithm that, given a set S of n integers and

another integer x, determines whether or not there exist two elements in S whose sum is exactly x. (Hint: Sort the n integers of S.)

9. (10%) Consider the following balls-and-bin game. We start with one black ball and one white ball in a bin. We repeatedly do the following: choose one ball from the bin uniformly at random, and then put the ball back in the bin with another ball of the same color. We repeat until there are n balls in the bin. Show that the number of white balls is equally likely to be any number between 1 and n-1. (Hint: Prove by mathematical induction.)

10. Give n integers in the range 1 to k. If one of the n numbers appears more than $n/2$ times, please design algorithms to find the number and satisfy some subset of the following three criteria:
    (1) The algorithm runs in $O(n)$ time.
    (2) The algorithm uses no more than $O(k)$ of storage space in addition to the original array.
    (3) The algorithm uses no more than a constant amount of storage space in addition to the original array.

    (5%) (a) Give an algorithm that satisfies criteria 1 and 2 above.

    (5%) (b) Give an algorithm that satisfies criteria 1 and 3 above.

Algorithms Second Examination
Dec. 16, 2011

1. (10%) Describe an $O(n)$-time algorithm that, given a set S of n distinct numbers and a positive integer $k \leq n$, determines the k numbers in S that are closest to the median of S.

2. (15%) We have a bag which has a weight capacity of x kilograms. We also have n items, each of different weights. Our target is to pack as many items as possible in the bag, without exceeding the weight capacity. Note that, we want to maximize the number of items, not the total weights of the items. Your clever friend, Matthew, tells you that one way is to pack items according to increasing order of the weights, from smallest to the largest (until exceeding the weight capacity), and this strategy can pack the maximum number of items. You want to follow Matthew's idea. From the first glance, it seems that finding all the desired items to be packed will require sorting, thus taking $O(n \log n)$ time. However, this can be done much faster. Design an algorithm to determine the desired items in $O(n)$ time.

3. (10%) Give an $O(n^2)$-time algorithm to find the longest monotonically increasing subsequence of a sequence of n numbers.

4. (15%) Use the dynamic programming approach to write an algorithm to **find** the maximum sum in any contiguous sub-array of a given array of n real values. Show the time complexity of your algorithm.

5. (10%) What are the two key properties that an optimization problem must have in order for greedy algorithm to apply?

6. (15%) Supposed we have a set $S = \{a_1, a_2, \ldots, a_n\}$ of n proposed activities that wish to use a resource. Each activity a_i has a start time s_i and a finish time f_i, where $0 \leq s_i < f_i < \infty$. If selected, activity a_i takes place during the half-open time interval $[s_i, f_i)$. Activities a_i and a_j are compatible if the intervals $[s_i, f_i)$ and $[s_j, f_j)$ do not overlap. In the **activity-selection problem**, we wish to select a maximum-size subset of mutually compatible activities. We assume that the activities are sorted in monotonically increasing order

of start time: $s_1 \le s_2 \le s_3 \le \ldots s_{n-1} \le s_n$. An algorithm is designed to select the last activity to start that is compatible with all previously selected activities. Describe how this approach is a greedy algorithm, and prove that it yields an optimal solution.

7. (10%) Suppose that a counter begins at a number with b 1s in its binary representation, rather than at 0. Show that the cost of performing n INCREMENT operations is $O(n)$ if $n = \Omega(b)$. (Do not assume that b is constanat.)

8. (15%) In the dynamic table problem, if the table T is full before insertion of an item, we expand T by doubling its size and if table T is below (1/4)-full after deletion of an item, we contract table T by halving its size. we define a potential function Φ as follows:
   (a) If table T is at least half full $\Phi(T) = 2\ \mathrm{num}(T) - \mathrm{size}(T)$.
   (b) If table T is less than half full $\Phi(T) = \mathrm{size}(T)/2 - \mathrm{num}(T)$.
   Please find the amortized cost of insertion and deletion.