Midterm Examination on Algorithms

Teacher: Biing-Feng Wang **April 23, 2002**

Problem 1: (10%) We can extend our asymptotic notation to the case of two parameters n and m that can go to infinity independently at different rates. For a given function $g(n, m)$, we denote by $O(g(n, m))$ the set of functions

$$O(g(n, m))=\{f(n, m): \text{there exist positive constants } c, n_0, \text{ and } m_0 \text{ such that } 0 \le f(n, m) \le cg(n, m) \text{ for all } n \ge n_0 \text{ and } m \ge m_0\}.$$

(1) (5%) Give corresponding definition for $\Omega(g(n, m))$.

(2) (5%) Give corresponding definition for $\Theta(g(n, m))$.

Problem 2: (15%, 5% each)

(1) Find an upper bound on the recurrence $T(n)=2T(\lfloor n/2 \rfloor+17)+n$ by using the substitution method. (You may assume that $T(k)=1$ for $k \le$ [illegible].) 34

(2) Find an upper bound on the recurrence $T(n)=3T(\lfloor n/2 \rfloor)+n$ by appealing to a recursion tree. (You may assume that $T(1)=1$.)

(3) Let $T(n)=T(n/2)+1$. Use *master theorem* to give an upper bound for $T(n)$. Please explain.

Problem 3: (12%)

(1) (6%) What is the running time of heapsort on an array A of length n that is already sorted in increasing order? What about decreasing order?

(2) (6%) Give the procedure *Heapify*(A, i) for a *min-heap*. What is the time complexity?

Problem 4: (10%) Let G be an undirected graph with n vertices and m edges. The identifier of each vertex of G is a string of 'a'~'z' and is of length≤20. Let V be an array storing the identifiers of the n vertices and E be an array storing the m edges. Given n, m, V, and E, please design an efficient algorithm to construct the adjacency list $L(i)$ for each vertex $V[i]$ of G. For example, given n=4, m=4, V[1..4]=('tai', 'chu', 'yang', 'che') and E[1..4]=(('tai', 'che') , ('chu', 'che'), ('yang', 'tai'), ('yang', 'chu')), we have L(1)=('che', 'yang'), L(2)=('che', 'yang'), L(3)=('tai', 'chu'), and L(4)=('chu', 'tai'). Describe and analyze your algorithm in details.

Problem 5: (10%) Show that finding the k-th smallest element among a sequence A of n numbers can be done in $O(n)$ time. Describe and analyze your algorithm in details.

Problem 6: (15%) For any sequence of numbers $S=(s_1, s_2, ..., s_c)$, define $cost(S)=\Sigma_{1\le i\le c}\{|s_i-m(S)|\}$, where $m(S)=(\Sigma_{1\le i\le c}\{s_i\})/c$. Let $A=(a_1, a_2, ..., a_n)$ be a non-decreasing sequence of n positive numbers and $k\ge1$ be an integer. In this problem, you are asked to partition A into k sub-sequences $A_1, A_2, ..., A_k$ such that $\Sigma_{1\le i\le k}\{cost(A_i)\}$ is minimized.

(1) (10%) Give an efficient algorithm to compute the minimum value of $\Sigma_{1\le i\le k}\{cost(A_i)\}$. What is the time complexity of your algorithm?

(2) (5%) Modify your algorithm in (1) to output the best partition $A_1, A_2, ..., A_k$. What is the time complexity?

Problem 7: (10%)

(1) (5%) Describe an efficient algorithm that, given a set $\{x_1, x_2, ..., x_n\}$ of points on the real line, determines the smallest set of unit-length closed intervals that contains all of the given points. What is the time complexity?

(2) (5%) Argue that your algorithm in (1) is correct.

Problem 8: (10%) Write a non-recursive version of *FIND-SET*(x) for a disjoint-set forest with path compression. What is the time complexity?

Problem 9: (8%) Prove that the average case time complexity of quicksort is $O(n\log n)$.

Bonus: (5%) Define *double hashing* briefly.