INTRO TO ALGORITHMS MIDTERM

1 True or False. You ***must*** justify your answers. ***No justifications, no credits.*** (30%)

a) $f(n) + o(f(n)) = \Theta(f(n))$

b) $T(n) \le O(n)$ iff $T(n) = O(n)$

c) $T_1(n) = \Theta(T_2(n))$, where $T_1(n)$ and $T_2(n)$ are defined below:

$$T_1(n) = \begin{cases} 1 & n = 1 \\ 2T_1(n/2) + n & n > 1 \end{cases}$$

$$T_2(n) = \begin{cases} n^2 & n \le 9999 \\ 2T_2(n/2) + 9999n & n > 9999 \end{cases}$$

d) The two statements

"The running time of **HEAPESORT** is $O(n \lg n)$. ", and

"The worst-case running time of **HEAPSORT** is $O(n \lg n)$."

are equivalent.

e) When sorting ***singed*** integers by radix sort, the signed bits have to be treated differently. Therefore, the ***best*** way to sort 65536 32-bit ***signed*** integers by radix sort is to partition each signed integer as a 3-digit number $d_2 d_1 d_0$, where d_2 is the signed bit, d_1 is a 15-bit digit, and d_0 is a 16-bit digit.

f) In the worst case, insertion sort takes a time $\le cn^2$ for ***each*** sufficiently large n. Thus, there are only a ***finite*** number of instances of size n on which insertion sort takes a time $> cn^2$.

Similarly, in the worst case, insertion sort takes a time $\ge dn^2$ for ***each*** sufficiently large n. Thus, there are only a ***finite*** number of instances of size n on which insertion sort takes a time $< dn^2$.

.

2 a) Prove that $2^n = \omega((3/2)^n)$ by the definition of ω. (4%)

b) Rank the following functions by order of growth. Justify your answers.

n^2 $\quad$ $\lg n!$ $\quad$ $(\lg n)!$ $\quad$ (6%)

3 We may partition an $n \times n$ matrix into nine $n/3 \times n/3$ matrices and use a divide-and-conquer approach to multiply two $n \times n$ matrices.

Let $T(n)$ be the running time of that divide-and-conquer algorithm. Then,

$T(n) = kT(n/3) + \Theta(n^2)$

where $k \ge 1$ is an integer.

For what value of k can the divide-and-conquer algorithm beat the remarkable Strassen's algorithm that runs in $\Theta(n^{\lg 7})$ or $O(n^{2.81})$ time? (8%)

Hint: Use the master theorem. Note: $\lg 7 \approx 2.8073549 < 2.81$

4 Given

$T(n) = 4T(n/3) + n$

Prove by the substitution method that $T(n) = O\left(n^{\log_3 4}\right)$ (6%)

5 Consider RANDOMIZED-QUICKSORT on n distinct elements.

Let X be the random variable that denotes the total number of comparisons performed in all calls to PARTITION.

Prove that $\mathrm{E}[X] = O(n \lg n)$. (8%)

6 Show that finding the minimum and maximum of n elements needs ***at most*** $\lceil 3n/2 \rceil - 2$ comparisons in the worst case. (8%)

7 Consider the following function that turns an array $A[1..n]$ into a max heap

BUILD-MAX-HEAP(A, n)
for $i = \lfloor n/2 \rfloor$ **downto** 1
  MAX-HEAPYFY(A, i, n)

Let $T(n) =$ the running time of BUILD-MAX-HEAP on an array of n elements

Clearly, $T(n) = \Omega(n)$, due to the **for** loop.

In this problem, we shall show that $T(n) = O(n)$ ***by recurrence***, even though BUILD-MAX-HEAP is an iterative function.

a) Let

$T'(k) =$ the time needed to build a heap of height k by BUILD-MAX-HEAP

Write down a recurrence for $T'(k)$ and show that $T'(k) = O(2^k)$ (6%)

Hint

The height of the left subheap is $k-1$. The height of the right subheap is $k-1$ or $k-2$. And, we are interested in the upper bound.

b) Use a) to show that $T(n) = O(n)$. (4%)

Note: Part b) is independent of part a), i.e. you may solve part b) without solving part a).

8 In this problem, we shall prove the following lower bound theorem by ***decision tree model***.

THEOREM

Any comparison-based algorithm for merging two n-element sorted lists needs ***at least*** $2n - o(n)$ comparisons in the worst case, assuming that all the $2n$ elements are distinct.

8 a) Prove that there are ***at least*** $\binom{2n}{n}$ leaves in the decision tree of a comparison-based merging algorithm working on two n-element sorted lists, assuming that all the $2n$ elements are distinct. (4%)

b) Use a) to prove the theorem. (6%)

Hint: $\binom{2n}{n}$ is the maximum term in the summation $\sum_{i=0}^{2n}\binom{2n}{i}$

Note: Part b) is independent of part a), i.e. you may solve part b) without solving part a).

9 a) Design a ***comparison-based*** sorting algorithm to sort n integers in $\Theta(n)$ time in the worst case, knowing that all integers are in the range 1 to 10.

Hint: Partition (6%)

b) Why is the lower bound $\Omega(n \lg n)$ for comparison-based sorting algorithms not satisfied in this case? (4%)

Note: Part b) is independent of part a), i.e. you may answer part b) without solving part a).