INTRO TO ALGORITHMS MIDTERM

Total: 122 points

Hint: Do not try to earn 122 points. Try to earn 100 points instead.

1 True or false. You ***must*** justify your answers. ***No justifications, no credits.*** (16%)

a) $O(n) = \{f(n) : \exists c > 0 \text{ such that } 0 \le f(n) \le cn \text{ for all } n > 0\}$

Hint: Compare it with book's definition on big-O.

$O(n) = \{f(n) : \exists c > 0\ n_0 > 0 \text{ such that } 0 \le f(n) \le cn \text{ for all } n \ge n_0\}$

b) $O(n^k) = O(n)^k$ for any integer k

Hint: Consider $k < 0, k = 0,$ and $k > 0$

c)* Let $S(n) = S(n-1) + O(1)$

$T(n) = T\left(n/2 + \sqrt{n}\right) + n$

Then, $S(n) = \Theta(T(n))$

d)* Heapsort is a good choice for sorting a linked list.

2 For each algorithm below, give a recurrence that describes its running time and give a tight bound or upper bound of the running time. You need not justify your answers. Note that

- this problem asks for running time, rather than worst-case running time, and
- give a tight bound whenever possible. (12%)

a)* Strassen's algorithm

b) Randomized Quicksort

c) Binary search

3* Prove by the definition of big-O that $(2n+3) \times O(n) = O(n^2)$. (8%)

Hint: Be sure to specify the constants required by the big-O definition.

4* Given

$T(n) = 8T(n/2) + O(n^2)$

Use constructive induction to show that $T(n) = O(n^3)$. (8%)

Hint: Try $T(n) \le dn^3 - d'n^2$.

5* Consider the mergesort recurrence

$$T(n) = T\left(\left\lceil\frac{n}{2}\right\rceil\right) + T\left(\left\lfloor\frac{n}{2}\right\rfloor\right) + O(n)$$

Prove that $T(n) = O(n \lg n)$ by *domain transformation*.

That is, define $S(n) = T(n + \alpha)$, where α is a constant, chosen to make $S(n)$ satisfy a simpler recurrence. (8%)

Hint:

$$T(n) = T\left(\left\lfloor\frac{n}{2}\right\rfloor\right) + T\left(\left\lceil\frac{n}{2}\right\rceil\right) + O(n) \le 2T\left(\left\lceil\frac{n}{2}\right\rceil\right) + O(n) \le 2T\left(\frac{n}{2}+1\right) + O(n)$$

6 Consider a variant of mergesort that divides an array of n elements into $\sqrt{n}$ subarrays, each having $\sqrt{n}$ elements. The $\sqrt{n}$ sorted subarrays are then merged simultaneously with the help of a min-priority queue.

MERGESORT$(A[1..n])$

1 **for** $i = 1$ to $\sqrt{n}$ do // $\sqrt{n}$ subarrays

  MERGESORT$\left(A[(i-1)\sqrt{n}+1..i\sqrt{n}]\right)$ // sort the ith subarray

2 MERGE$(A[1..n])$

MERGE$(A[1..n])$ // $A[1..n]$ contains $\sqrt{n}$ sorted subarrays

1 Let $B[1..n]$ be a new array

2 Build a min-priority queue Q on the $\sqrt{n}$ smallest elements, one from each sorted subarray

3 **for** $k = 1$ to n do

  $B[k] =$ EXTRACT-MIN(Q)

  // Suppose the element just extracted comes from the ith subarray

  if the ith subarray is not empty **then**

    INSERT$(Q,$ the next element of the ith subarray$)$

4 Copy $B[1..n]$ back to $A[1..n]$

a) Show that the running time of MERGE$(A[1..n])$ is $O\left(n \lg \sqrt{n}\right)$. (4%)

b)* Let $T(n)$ be the running time of MERGESORT$(A[1..n])$, then

$T(n) = \sqrt{n}T\left(\sqrt{n}\right) + O\left(n \lg \sqrt{n}\right)$

Give an asymptotic upper bound for $T(n)$. (8%)

Hint: Range transformation $S(n) = T(n)/n$ and change of variable

7 Consider the insertion sort

INSERTION-SORT(A, n)

for $i = 2$ **to** n

  $key = A[i]$

  $j = i - 1$

  while $j > 0$ and $A[j] > key$ // key (i.e. $A[i]$) is compared to $A[j]$

    $A[j+1] = A[j]$

    $j = j - 1$

  $A[j+1] = key$

a)* Draw the decision tree for insertion sort on 3 elements. (8%)

b)* In terms of the number of comparisons, for what value of n is insertion sort an optimal comparison sort in the worst case? (6%)

Hint: First, find the number of comparisons taken by insertion sort in the worst case. Then, compare it with the theoretical lower bound.

8 (Continuing 7)

Assume that the n elements are distinct and each of the $n!$ possible inputs is equally likely.

a) Define the indicator random variable

$X_{ij} = \mathrm{I}\{A[i] \text{ is compared to } A[j]\}, \quad 2 \le i \le n, 1 \le j \le i-1$

where $A[i]$ and $A[j]$ denote the values held in variables key and $A[j]$, respectively, at the time $A[j] > key$ is evaluated.

What is the value of $\mathrm{E}[X_{ij}]$? (4%)

b) Let X be a random variable denoting the total number of times $A[j] > key$ is compared in the course of executing INSERTION-SORT(A, n).

Show that $\mathrm{E}[X] = n^2/2 + \Theta(n)$. (6%)

Hint: $X = \sum_{i=2}^{n} \sum_{j=1}^{i-1} X_{ij}$

9 Consider the following MSD radix sort on $A[1..n]$ in which each number $A[i]$ has d digits $x_d \ldots x_2 x_1$. The value of each digit x_i satisfies $0 \le x_i \le k$ for some constant k.

MSD_RADIX_SORT$(A[1..n])$

for $i = d$ **downto** 1

for each pile having the same digit x_{i+1}, sort digit x_i by counting sort

a) What is the running time of MSD_RADIX_SORT$(A[1..n])$? (6%)

Hint: Compare it with LSD radix sort

b)* Suppose each $A[i]$ is a 32-bit unsigned integer, what is the best value of d and the resulting running time? (4%)

Hint: Compare it with LSD radix sort

10* Prove the following theorem.

THEOREM Finding the minimum and maximum of n elements needs ***at least*** $\lceil 3n/2 \rceil - 2$ comparisons in the worst case. (10%)

11 Recall that, in the worst case, quicksort makes extremely unbalanced partitions ($0 : n-1$ split) that cause the depth of the recursion tree to be about n. But, in the best case, it makes balanced partitions ($(n-1)/2 : (n-1)/2$ split) that reduce the depth of the recursion tree to $\lg n$. Thus, to speed up quicksort, the depth of the recursion tree shall be limited.

The introspective sort (introsort) is a variant of quicksort that

- puts an $O(\lg n)$ limit on the depth of the recursion tree, and
- switches to heapsort when the depth limit is reached.

The following pseudocode also resorts to insertion sort when the array size is smaller than some threshold value.

INTROSORT($A[p..r], depth\_limit$)
if $(1 < r - p + 1 \leq threshold)$ INSERTION_SORT($A[p..r]$)
else if $(depth\_limit == 0)$ HEAPSORT($A[p..r]$)
else
    $q =$ PARTITION($A[p..r]$)
    INTROSORT($A[p..q-1], depth\_limit - 1$)
    INTROSORT($A[q+1..r], depth\_limit - 1$)

a)* What is the best value for the constant $threshold$? (2%)
1) 4   2) 16   3) 128   4) 256
Hint: HW#1

b) Consider the call
INTROSORT($A[1..n], c \lg n$)
What is the best value for the constant c? (2%)
1) 0.5   2) 1.0   3) 1.5   4) 5.0
Hint: Don't call heapsort and insertion sort too frequently.

c) Prove that the call in b) takes $O(n \lg n)$ time in the worst case. (6%)
Hint: Count the running time of all calls to HEAPSORT and the running time of all calls to PARTITION.

d)* Prove that the call in b) takes $\Omega(n \lg n)$ time in the worst case. (4%)
Hint: Part c) and theoretical lower bound