Quiz 1

- Do not open this quiz booklet until you are directed to do so. Read all the instructions on this page.
- When the quiz begins, write your name on every page of this quiz booklet.
- This quiz contains 3 problems, some with multiple parts. You have 80 minutes to earn 80 points.
- This quiz booklet contains 9 pages, including this one. Two extra sheets of scratch paper are attached. Please detach them before turning in your quiz at the end of the examination period.
- This quiz is closed book. You may use one handwritten A4 or $8\frac{1}{2}'' \times 11''$ crib sheet. No calculators or programmable devices are permitted.
- Write your solutions in the space provided. If you need more space, write on the back of the sheet containing the problem. Do not put part of the answer to one problem on the back of the sheet for another problem, since the pages may be separated for grading.
- Do not waste time and paper rederiving facts that we have studied. It is sufficient to cite known results.
- Do not spend too much time on any one problem. Read them all through first, and attack them in the order that allows you to make the most progress.
- Show your work, as partial credit will be given. You will be graded not only on the correctness of your answer, but also on the clarity with which you express it. Be neat.
- Good luck!

Problem	Points	Grade	Initials
1	16		
2	19		
3	45		
Total	80		

Name: ______________________

MIT students: Circle the name of your recitation instructor:

Bob Chong George Jennifer Rachel

Problem 1. Match-up [16 points]

Fill in the following table by specifying, for each algorithm, the letter of the recurrence for its running time $T(n)$ and the numeral of the solution to that recurrence. Points will be deducted for wrong answers, so do not guess unless you are reasonably sure.

Algorithm	**Recurrence**	**Solution**
Merge sort (worst case)		
Binary search (worst case)		
Randomized quicksort (expected)		
Strassen's algorithm (worst case)		
Selection (worst case)		
Randomized selection (expected)		

Letter	**Recurrence**
a	$T(n) = 7\,T(n/2) + \Theta(n^2)$
b	$T(n) = T(n/2) + \Theta(1)$
c	$T(n) = T(\lceil n/5 \rceil) + T(7n/10 + 6) + \Theta(n)$
d	$T(n) = 2\,T(n/2) + \Theta(n)$
e	$T(n) = \frac{2}{n} \sum_{k=\lfloor n/2 \rfloor}^{n} T(k) + \Theta(n)$
f	$T(n) = 2\,T(n/2) + \Theta(1)$
g	$T(n) = \frac{2}{n} \sum_{k=0}^{n-1} T(k) + \Theta(n)$
h	$T(n) = T(n-1) + \Theta(n)$

Numeral	**Solution**
1	$T(n) = \Theta(n)$
2	$T(n) = \Theta(\lg n)$
3	$T(n) = \Theta(n^{\lg 7})$
4	$T(n) = \Theta(1)$
5	$T(n) = \Theta(n^2)$
6	$T(n) = \Theta(n \lg n)$

Problem 2. Merging several sorted lists [19 points]

Professor Fiorina uses the following algorithm for merging k sorted lists, each having n/k elements. She takes the first list and merges it with the second list using a linear-time algorithm for merging two sorted lists, such as the merging algorithm used in merge sort. Then, she merges the resulting list of $2n/k$ elements with the third list, merges the list of $3n/k$ elements that results with the fourth list, and so forth, until she ends up with a single sorted list of all n elements.

(a) Analyze the worst-case running time of the professor's algorithm in terms of n and k.

(b) Briefly describe an algorithm for merging k sorted lists, each of length n/k, whose worst-case running time is $O(n \lg k)$. Briefly justify the running time of your algorithm. (If you cannot achieve $O(n \lg k)$, do the best you can for partial credit.)

(c) Argue that there are

$$\frac{n!}{((n/k)!)^k}$$

different ways to interleave k lists, each of length n/k, into a single list of length n.

here are $n!$ ways of permutating the n elements. Any of the (n/k) elements of first list may be interleaved with any of the (n/k) elements of the second list. Given that there are k lists, the number of different way is therefore $(n!)/(((n/k)!)^k)$.

(d) Prove a lower bound of $\Omega(n \lg k)$ on the worst-case running time of any comparison algorithm for merging k sorted lists each of length n/k. (*Hint:* Use the fact that $(n/e)^n \le n! \le n^n$.)

Consider the decision tree used to make the comparisons. Height of tree is: $\lg \frac{(}{n}!)(((n/k)!)^k) = \lg n! - \lg((n/k)!)^k \le \lg n! - \lg(((n/k)^(n/k))^k) = \lg n! - n\lg(n/k) = \lg n! - n\lg n + n\lg k \le \lg((n/e)^n) - n\lg n + n\lg k = n\lg n - n\lg e - n\ lgn + n\lg k = n\lg k - n\lg e whichis \omega(n\lg k)$

Problem 3. True or False, and Justify [45 points]

Circle **T** or **F** for each of the following statements, and briefly explain why. The more content you provide in your justification, the higher your grade, but be brief. Your justification is worth more points than your true-or-false designation.

(a) T F A constant $n_0 \geq 1$ exists such that for any $n \geq n_0$, there is an array of n elements such that insertion sort runs faster than merge sort on that input.

(b) T F Consider the algorithm from the textbook for building a max-heap:

```
BUILD-MAX-HEAP$(A)$
1  heap-size$[A] \leftarrow$ length$[A]$
2  for $i \leftarrow \lfloor$length$[A]/2\rfloor$ downto $1$
3      do MAX-HEAPIFY$(A, i)$
```

On an array of n elements, this code runs in $\Theta(n \lg n)$ time in the worst case, because there are $\Theta(n)$ calls to MAX-HEAPIFY, and the worst-case time for any call is $\Theta(\lg n)$.

(c) **T F** Given a number a and a positive integer n, the value $a^{2^n} = a^{(2^n)}$ can be computed in $O(n \lg n)$ time by repeated squaring. (Assume that multiplying two numbers takes constant time.)

(d) **T F** An adversary can force randomized quicksort to run in $\Omega(n^2)$ time by providing as input an already sorted or reverse-sorted array of size n.

(e) **T F** For any integer-valued random variable $X \geq 0$, we have

$$\Pr\{X = 0\} \geq 1 - \mathrm{E}[X] \ .$$

(f) **T F** Bucket sort is a suitable auxiliary sort for radix sort.

(g) **T F** Consider two implementations of a hash table with m slots storing n keys, where $n < m$. Let $T_c(m, n)$ be the expected time for an unsuccessful search in the table if collisions are resolved by chaining, using the assumption of simple uniform hashing. Let $T_o(m, n)$ be the expected time for an unsuccessful search in the table if collisions are resolved by open addressing, using the assumption of uniform hashing. Then, we have $T_c(m, n) = \Theta(T_o(m, n))$.

(h) **T F** Let $\mathcal{H}$ be a class of universal hash functions for a hash table of size $m = n^3$. Then, if we use a random $h \in \mathcal{H}$ to hash n keys into the table, the expected number of collisions is at most $1/n$.

(i) **T F** Given a set $\{(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)\}$ of points in the plane, the $\sqrt{n}$ points closest to the origin $(0, 0)$ (using normal Euclidean distance) can be found in $O(n)$ time in the worst case.

SCRATCH PAPER — Please detach this page before handing in your exam.

SCRATCH PAPER — Please detach this page before handing in your exam.