

Name \_\_\_\_\_

**Instructions** This exam consists of four problems worth a total of 100 points.

Do all of the short answer questions, Problems (1) through (3). From the algorithm design questions, Problems (4) and (5), *choose one*. If you do both Problems (4) and (5), only one of them will be graded.

On the algorithm design questions, pseudocode is not required, but it may help with analyzing your algorithm. When analyzing the time for a recursive algorithm, write down a recurrence and solve it.

You have 75 minutes. The exam is closed book, closed notes, and closed calculator. Good luck!

(1) **(Asymptotics)** (20 points)

- (a) (10 points) Suppose program  $A$  has time complexity  $\Theta(f(n))$  and program  $B$  has time complexity  $\Theta(g(n))$  where  $f = o(g)$ . What can you say about the *real-world running time* of program  $A$  versus program  $B$ ?

- (b) (10 points) Prove the following asymptotic relationship:

$$n \log_e n = o(n^2)$$

(2) **(Summations)** (15 points)

- (a) (5 points) Give the most accurate statement you can about the order of growth of the following summation, where  $k \geq 0$  is an integer.

$$\sum_{1 \leq i \leq n} i^k$$

- (b) (10 points) Derive a  $\Theta$ -bound on the order of growth of the following summation. Be sure to show your work.

$$\sum_{1 \leq i \leq n} i^2 n - \sum_{1 \leq i \leq n} i^3$$

(3) **(Recurrences)** (15 points) Determine the order of growth of the following recurrences.

(a) (5 points)  $T(n) = T(n/4) + \Theta(\sqrt{n} \log n)$

(a) (5 points)  $T(n) = 2T(n/4) + \Theta(\sqrt{n} \log n)$

(a) (5 points)  $T(n) = 3T(n/4) + \Theta(\sqrt{n} \log n)$

- (4) **(Algorithm design)** (50 points) For a set  $S$  of  $n$  real numbers, a pair of elements  $x, y \in S$ , where  $x < y$ , are said to be *close* if

$$y - x \leq \frac{1}{n-1} (\max S - \min S).$$

Suppose you are given an *unsorted* array  $A[1:n]$  of distinct real numbers. Design an algorithm that finds a pair of close numbers in  $A$  in  $\Theta(n)$  time.

(Note: Your algorithm should *not* use hashing. Also, keep in mind that you do not have to find all close pairs.)

In your solution, be sure to:

- (a) (35 points) present your algorithm using prose and pictures,
- (b) (10 points) argue why your algorithm is correct, and
- (c) (5 points) analyze the running time of your algorithm.

(This page is blank.)

(This page is blank.)

- (5) **(Algorithm design)** (50 points) Suppose you have two separate *sorted* arrays of numbers,  $A[1:m]$  and  $B[1:n]$ , where all the numbers in  $A$  and  $B$  are distinct. Given an integer  $k$  where  $1 \leq k \leq m+n$ , design an algorithm that finds the  $k$ th smallest element in the merge of arrays  $A$  and  $B$  in  $O(\log k)$  time.

(Note: Explicitly merging the sorted arrays  $A$  and  $B$  will take  $\Theta(m+n)$  time, which is too much. Your algorithm, from the point it receives the unmerged arrays  $A$  and  $B$ , to the point at which it outputs the  $k$ th smallest element, must run in  $O(\log k)$  time.)

In your solution, be sure to:

- (a) (35 points) present your algorithm using prose and pictures,
- (b) (10 points) argue why your algorithm is correct, and
- (c) (5 points) analyze the running time of your algorithm.

(This page is blank.)

(This page is blank.)