CS 373: Combinatorial Algorithms, Summer 2000
IMCS/Quantum

Final Exam (Saturday August 12, 2000)

Name:	
Net ID:	

This is a closed-book, no-calculator exam.

You have 2 hours to complete the exam.

You may use *two* $8\frac{1}{2}'' \times 11''$ sheets of notes (both sides). **Please turn this in with your exam**.

- Print your name and netid in the boxes above, and print your **netid** at the top of every page.
- Answer all the questions. They are ten (10) points each.
- Read the entire exam before writing anything. Make sure you understand what the questions are asking. If you give a beautiful answer to the wrong question, you'll get no credit.
- Make sure you justify your answers: a correct algorithm with no justification of its correctness, will only get partial credit; just pseudocode may get no points at all.
- If you need a subroutine or lemma already written/proved in CLR or the class notes, simply refer to it; you do not need to repeat the code/proof/analysis. You do, however, need to justify how you use that information.
- If you need more than the front of the page, please mark clearly that the answer is continued on the back of that page. If you feel like you want to erase something, don't. Just cross it out and plainly mark your official answer.

#	Score
1	
2	
3	
4	
5	
6	
7	
Total	

1. **Short Answer**

sorting	induction	Master theorem	divide and conquer
randomized algorithm	amortization	brute force	hashing
binary search	depth-first search	splay tree	Fibonacci heap
convex hull	sweep line	minimum spanning tree	longest path
shortest path	adversary argument	NP-hard	reduction
string matching	evasive graph property	dynamic programming	Fibonacci sequence

Choose from the list above the best method for solving each of the following problems. We do *not* want complete solutions, just the term above describing the best solution technique. Each item is worth 1 point. A method may be used more than once. You will not be penalized for wrong answers.

(a) Given a phone book and a phone number, find the name associated with the number.

(b) Given a collection of n segments in the plane, determine whether any two intersect in $O(n \log n)$ time.

(c) Given an undirected graph G and an integer k, determine if G has a complete subgraph with k edges.

(d) Given an undirected graph G, determine if G has a triangle — a complete subgraph with three vertices.

(e) Prove that any n-vertex graph with minimum degree at least $n/2$ has a spanning tree.

(f) Given a graph G and three distinguished vertices u, v, and w, determine whether G contains a path from u to v that passes through w.

(g) Given a graph G and two distinguished vertices u and v, determine whether G contains a path from u to v that passes through at most 17 edges.

(h) Solve the recurrence $T(n) = 5T(n/17) + O(n^{4/3})$.

(i) Solve the recurrence $T(n) = T(n-1) + \log n$, where $T(1) = 0$.

(j) Given an array of n integers, find the integer that appears most frequently in the array.

(a) ________________ (f) ________________

(b) ________________ (g) ________________

(c) ________________ (h) ________________

(d) ________________ (i) ________________

(e) ________________ (j) ________________

2. **Ghosts and Ghostbusters**

   A group n ghostbusters is battling n ghosts. Each ghostbuster has a special ghostbusting laser to eradicate a single ghost (the laser beam stops when it hits a ghost). The ghostbusters decide to shoot one ghost each, and everybody will shoot at the same time. Unfortunately, if two of these special laser beams cross, an anomaly in the space-time continuum will appear and destroy the universe. So they want to make sure that no beams cross.

   Assume that the position of each ghostbuster and ghost is a fixed point in the plane and that no three points are collinear.

   (a) **[6 pts]** Prove that there exists a line passing through one ghostbuster/ghost pair such that one one side of the pair, the number of ghosts and ghostbusters is the same (and therefore also on the other side)

   (b) **[4 pts]** Give an algorithm to find such a line in $O(n \log n)$ time.

3. **Adding to zero**

   Suppose you are given an array of n numbers, sorted in increasing order.

   (a) **[3 pts]** Describe an $O(n)$-time algorithm for the following problem:
   Find two numbers from the list that add up to zero, or report that there is no such pair. In other words, find two numbers a and b such that $a + b = 0$.

   (b) **[7 pts]** Describe an $O(n^2)$-time algorithm for the following problem:
   Find *three* numbers from the list that add up to zero, or report that there is no such triple. In other words, find three numbers a, b, and c, such that $a + b + c = 0$. [Hint: Use something similar to part (a) as a subroutine.]

4. **String Matching with "gap" characters**

   A "gap" in a pattern (to be matched with some text) matches anything. For example, suppose '*' is a gap character in the pattern 'aa*aa*aa'; this pattern will match the strings 'aaaaaa', 'aabaabbaa', and 'aaaaaaa', but not 'aababaa'. Describe (and justify) how to compute the failure function to handle a gap character.

5. **Sorting blocks**

   You are given k presorted arrays of total n items. Describe and analyze an algorithm that sorts all n items in $O(n \log k)$ time.

6. **PARTITION is NP-complete**

   PARTITION is the problem "Given a set of positive integers, is there a partition into two sets whose sums are equal?". For example, the set $\{1, 6, 7, 8, 10\}$ can be partitioned (into $\{1, 7, 8\}$, $\{6, 10\}$), but $\{2, 6, 7, 8, 10\}$ cannot.

   Another problem, SUBSET-SUM, asks "Given a set of positive integers and a target value k, is there a subset whose sum is k?". For example, the set $\{1, 6, 7, 8, 10\}$ has a subset whose sum is 16 (namely $\{1, 7, 8\}$).

   Prove that PARTITION is NP-hard by reduction from SUBSET-SUM, assuming that SUBSET-SUM is NP-complete.

7. **Finding the size of a max clique using CLIQUE**

   Suppose you are have a black box that magically solves CLIQUE (the graph clique problem) in constant time. That is, given an undirected graph and an integer k, the black box tells you, in constant time, whether or not the graph has clique of size k or more.

   (a) (3 pts) Using the black box, design and analyze a **polynomial-time** algorithm that computes the size of the largest clique in a graph.

   (b) (7 pts) Using the black box, design and analyze a **polynomial-time** algorithm that finds a subset of the vertices that is a clique of largest size in the graph.