CS 373: Combinatorial Algorithms, Spring 1999

Final Exam (May 7, 1999)

Name:	
Net ID:	Alias:

This is a closed-book, closed-notes exam!

If you brought anything with you besides writing instruments and your two $8\frac{1}{2}'' \times 11''$ cheat sheets, please leave it at the front of the classroom.

- Print your name, netid, and alias in the boxes above, and print your name at the top of every page.
- **Answer six of the seven questions on the exam.** Each question is worth 10 points. If you answer every question, the one with the lowest score will be ignored. **1-unit graduate students must answer question #7.**
- Please write your answers on the front of the exam pages. Use the backs of the pages as scratch paper. Let us know if you need more paper.
- 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. If any question is unclear, please ask one of us for clarification.
- Don't spend too much time on any single problem. If you get stuck, move on to something else and come back later.
- Write *something* down for every problem. Don't panic and erase large chunks of work. Even if you think it's nonsense, it might be worth partial credit.

#	Score	Grader
1		
2		
3		
4		
5		
6		
7		

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	shortest paths
shortest path	adversary argument	NP-hard	reduction
string matching	evasive graph property	dynamic programming	H_n

Choose from the list above the best method for solving each of the following problems. We do *not* want complete solutions, just a short description of the proper solution technique! Each item is worth 1 point.

(a) Given a Champaign phone book, find your own phone number.

(b) Given a collection of n rectangles 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 Hamiltonian cycle.

(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 $T(n) = 1/n + T(n-1)$, where $T(0) = 0$.

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

(a) ______________________ (f) ______________________

(b) ______________________ (g) ______________________

(c) ______________________ (h) ______________________

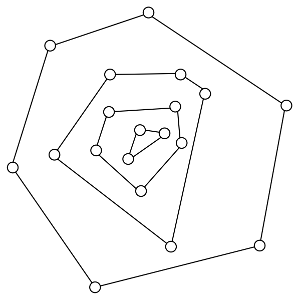
(d) ______________________ (i) ______________________

(e) ______________________ (j) ______________________

2. **Convex Layers**

   Given a set Q of points in the plane, define the *convex layers* of Q inductively as follows: The first convex layer of Q is just the convex hull of Q. For all $i > 1$, the ith convex layer is the convex hull of Q after the vertices of the first $i - 1$ layers have been removed.

   Give an $O(n^2)$-time algorithm to find all convex layers of a given set of n points. [Partial credit for a correct slower algorithm; extra credit for a correct faster algorithm.]



A set of points with four convex layers.

3. 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 $a + b = 0$.

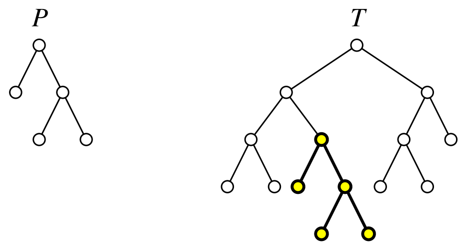
   (b) **[7 pts]** Describe an $O(n^2)$-time algorithm for the following problem:

   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. **Pattern Matching**

   (a) **[4 pts]** A *cyclic rotation* of a string is obtained by chopping off a prefix and gluing it at the end of the string. For example, ALGORITHM is a cyclic shift of RITHMALGO. Describe and analyze an algorithm that determines whether one string $P[1..m]$ is a cyclic rotation of another string $T[1..n]$.

   (b) **[6 pts]** Describe and analyze an algorithm that decides, given any two binary trees P and T, whether P equals a subtree of T. [Hint: First transform both trees into strings.]



P occurs exactly once as a subtree of T.

5. **Two-stage Sorting**

   (a) **[1 pt]** Suppose we are given an array $A[1 \,..\, n]$ of distinct integers. Describe an algorithm that splits A into n/k subarrays, each with k elements, such that the elements of each subarray $A[(i-1)k+1 \,..\, ik]$ are sorted. Your algorithm should run in $O(n \log k)$ time.

   (b) **[2 pts]** Given an array $A[1 \,..\, n]$ that is already split into n/k sorted subarrays as in part (a), describe an algorithm that sorts the entire array in $O(n \log(n/k))$ time.

   (c) **[3 pts]** Prove that your algorithm from part (a) is optimal.

   (d) **[4 pts]** Prove that your algorithm from part (b) is optimal.

4	14	7	3	1	20	11	9	5	13	12	19	10	16	17	2	8	6	18	15

$$\Big\Downarrow (a)$$

1	4	3	7	14 ‖	5	9	11	13	20 ‖	10	12	16	17	19 ‖	2	6	8	15	18

$$\Big\Downarrow (b)$$

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20

6. **SAT Reduction**

   Suppose you are have a black box that magically solves SAT (the formula satisfiability problem) in constant time. That is, given a boolean formula of variables and logical operators ($\wedge, \vee, \neg$), the black box tells you, in constant time, whether or not the formula can be satisfied. Using this black box, design and analyze a **polynomial-time** algorithm that computes an assignment to the variables that satisfies the formula.

7. **Knapsack**

You're hiking through the woods when you come upon a treasure chest filled with objects. Each object has a different size, and each object has a price tag on it, giving its value. There is no correlation between an object's size and its value. You want to take back as valuable a subset of the objects as possible (in one trip), but also making sure that you will be able to carry it in your knapsack which has a limited size.

In other words, you have an integer capacity K and a target value V, and you want to decide whether there is a subset of the objects whose total size is *at most* K and whose total value is *at least* V.

(a) **[5 pts]** Show that this problem is NP-hard. [Hint: Restate the problem more formally, then reduce from the NP-hard problem PARTITION: Given a set S of nonnegative integers, is there a partition of S into disjoint subsets A and B (where $A \cup B = S$) whose sums are equal, *i.e.*, $\sum_{a \in A} a = \sum_{b \in B} b$.]

(b) **[5 pts]** Describe and analyze a dynamic programming algorithm to solve the knapsack problem in $O(nK)$ time. Prove your algorithm is correct.

7. **Knapsack (continued)**