Final Exam

- Do not open this exam booklet until directed to do so. Read all the instructions on this page.
- When the exam begins, write your name on every page of this exam booklet.
- You have 180 minutes to earn **180** points. 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.
- This exam booklet contains 20 pages, including this one. Two extra sheets of scratch paper are attached. Please detach them before turning in your exam at the end of the exam period.
- This exam is closed book. You may use **three** handwritten, $8\frac{1}{2}'' \times 11''$ or A4 crib sheets (both sides). No calculators or programmable devices are permitted. No cell phones or other communications 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. 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.
- 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	Parts	Points	Grade	Grader		Problem	Parts	Points	Grade	Grader
1	10	30				7	–	10		
2	8	40				8	–	10		
3	3	15				9	–	15		
4	–	10				10	–	10		
5	–	10				11	–	20		
6	–	10				Total		180		

Name: _____

Athena username: _____

Recitation:	Nick	Nick	Tianren	David	Joe	Joe	Michael
	WF10	**WF11**	**WF12**	**WF1**	**WF2**	**WF3a**	**WF3b**

Problem 1. True or false [30 points] (10 parts)

For each of the following questions, circle either T (True) or F (False). **Explain your choice.** (Your explanation is worth more than your choice of true or false.)

(a) T F For all positive $f(n)$, $f(n) + o(f(n)) = \Theta(f(n))$.

(b) T F For all positive $f(n)$, $g(n)$ and $h(n)$, if $f(n) = O(g(n))$ and $f(n) = \Omega(h(n))$, then $g(n) + h(n) = \Omega(f(n))$.

(c) T F Under the simple uniform hashing assumption, the probability that three specific data elements (say 1, 2 and 3) hash to the same slot (i.e., $h(1) = h(2) = h(3)$) is $1/m^3$, where m is a number of buckets.

(d) T F Given an array of n integers, each belonging to $\{-1, 0, 1\}$, we can sort the array in $O(n)$ time in the worst case.

(e) T F The following array is a max heap: $[10, 3, 5, 1, 4, 2]$.

(f) T F RADIX SORT does not work correctly (i.e., does not produce the correct output) if we sort each individual digit using INSERTION SORT instead of COUNTING SORT.

(g) T F Given a directed graph G, consider forming a graph G' as follows. Each vertex $u' \in G'$ represents a strongly connected component (SCC) of G. There is an edge (u', v') in G' if there is an edge in G from the SCC corresponding to u' to the SCC corresponding to v'.

Then G' is a directed acyclic graph.

(h) T F Consider two positively weighted graphs $G = (V, E, w)$ and $G' = (V, E, w')$ with the same vertices V and edges E such that, for any edge $e \in E$, we have $w'(e) = w(e)^2$.

For any two vertices $u, v \in V$, any shortest path between u and v in G' is also a shortest path in G.

(i) T F An optimal solution to a knapsack problem will always contain the object i with the greatest value-to-cost ratio v_i/c_i.

(j) T F Every problem in NP can be solved in exponential time.

Problem 2. Short answer [40 points] (8 parts)

(a) Rank the following functions by increasing order of growth; that is, find an arrangement g_1, g_2, g_3, g_4 of the functions satisfying $g_1 = O(g_2)$, $g_2 = O(g_3)$, $g_3 = O(g_4)$. (For example, the correct ordering of n^2, n^4, n, n^3 is n, n^2, n^3, n^4.)

$$f_1 = (n!)^{1/n} \qquad f_2 = \log n^n \qquad f_3 = n^{n^{1/2}} \qquad f_4 = n \log n \log\log n$$

(b) Solve the following recurrences by giving tight Θ-notation bounds. You do not need to justify your answers, but any justification that you provide will help when assigning partial credit.

i. $T(n) = 4T(n/2) + n^2 \log n$

ii. $T(n) = 8T(n/2) + n \log n$

iii. $T(n) = \sqrt{6006} \cdot T(n/2) + n^{\sqrt{6006}}$

(c) Give a recurrence $T(n) = \cdots$ for the running time of each of the following algorithms, along with the asymptotic solution to that recurrence:

i. Insertion sort

ii. Merge sort

iii. 2D peak finding (the fastest algorithm we've seen)

(d) Could a binary search tree be built using $o(n \lg n)$ comparisons in the comparison model? Explain why or why not.

(e) Given n integers in the range $0 \ldots k$, describe how to preprocess these integers into a data structure that can answer the following query in $O(1)$ time: given two integers a and b, how many integers fall within the range $a \ldots b$?

(f) Describe how any comparison-based sorting algorithm can be made stable, without affecting the running time by more than a constant factor.

(g) In dynamic programming, we derive a recurrence relation for the solution to one subproblem in terms of solutions to other subproblems. To turn this relation into a bottom-up dynamic programming algorithm, we need an order to fill in the solution cells in a table, such that all needed subproblems are solved before solving a subproblem. For each of the following relations, give such a valid traversal order, or if no traversal order is possible for the given relation, briefly justify why.

i. $A(i,j) = F(A(i,j-1), A(i-1,j-1), A(i-1,j+1))$

ii. $A(i,j) = F(A(\min\{i,j\}-1, \min\{i,j\}-1), A(\max\{i,j\}-1, \max\{i,j\}-1))$

iii. $A(i,j) = F(A(i-2,j-2), A(i+2,j+2))$

(h) Consider an array $A[1\ldots n]$ of integers in the range $1\ldots n^2$. A number a is a ***heavy hitter*** in A if a occurs in A at least $n/2$ times.

Give an efficient algorithm that finds all heavy hitters in a given array A.

Problem 3. You are the computer [15 points] (3 parts)

(a) Fill in the following grid with the correct subproblem solutions for this sequence alignment problem with these weights: 0 for mutation, 1 for insertion or deletion, and 3 for a match (the goal is to maximize the sum of the weights). Here "ATC" is the starting sequence and "TCAG" is the ending sequence.

–	–	A	T	C
–	0			
T				
C				
A				
G				

What is the optimal alignment?

(b) Draw a max-heap on the following set of integers: $\{2, 3, 5, 7, 11, 13, 17\}$. (You do not need to use the Build-Heap algorithm.)

(c) Compute $\sqrt[3]{6006}$ using two iterations of Newton's method, i.e., fill out the following table. Your entry for x_1 should be fully simplified. Your entry for x_2 can be left unsimplified.

i	x_i
0	1
1	
2	

Problem 4. Rotated array [10 points]

Consider an array $A[1 \cdots n]$ constructed by the following process: we start with n distinct elements, sort them, and then rotate the array k steps to the right. For example, we might start with the sorted array $[1, 4, 5, 9, 10]$, and rotate it right by $k = 3$ steps to get $[5, 9, 10, 1, 4]$. Give an $O(\log n)$-time algorithm that finds and returns the position of a given element x in array A, or returns None if x is not in A. Your algorithm is given the array $A[1 \cdots n]$ but does *not* know k.

Problem 5. Taxachusetts [10 points]

Suppose you are given a weighted graph $G = (V, E, w)$ of highways, and the state government has implemented a new tax rule whereby the cost of a path gets doubled as penalty if the number of edges in the path is greater than 10. Explain how to reduce finding the shortest-path weight between every pair of vertices (under this penalty) to the usual all-pairs shortest paths problem (as solved by Floyd-Warshall).

Problem 6. Does this path make me look fat? [10 points]

Consider a connected weighted directed graph $G = (V, E, w)$. Define the ***fatness*** of a path P to be the maximum weight of any edge in P. Give an efficient algorithm that, given such a graph and two vertices $u, v \in V$, finds the minimum possible fatness of a path from u to v in G.

Problem 7. Indiana Jones and the Temple of Algorithms [10 points]

While exploring an ancient temple, Prof. Jones comes across a locked door. In front of this door are two pedestals, and n blocks each labeled with its positive integer weight. The sum of the weights of the blocks is W. In order to open the door, Prof. Jones needs to put every block on one of the two pedestals. However, if the difference in the sum of the weights of the blocks on each pedestal is too large, the door will not open.

Devise an algorithm that Prof. Jones can use to divide the blocks into two piles whose total weights are as close as possible. To avoid a boulder rolling quickly toward him, Prof. Jones needs your algorithm to run in pseudopolynomial time.

Problem 8. Claustrophobic chickens [10 points]

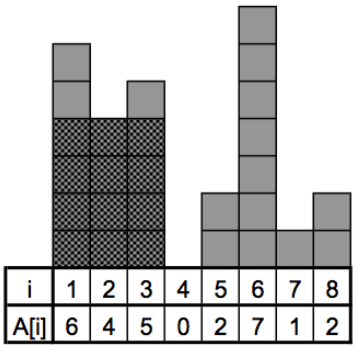
Prof. Tyson has noticed that the chickens in his farm frequently get claustrophobic. He wants to build a monitoring system that will alert him when this happens, allowing him to manually rearrange the chickens. After thorough research, Prof. Tyson determines that a chicken becomes claustrophobic if it is within 2 feet of at least 8 other chickens.

Prof. Tyson has installed a tracker on each chicken, which reports the (x, y) coordinates (measured in feet) of each of his chickens. Suppose that there are n chickens whose locations are represented as a sequence $(x_1, y_1), \ldots, (x_n, y_n)$. Prof. Tyson needs an efficient algorithm to determine whether there are any claustrophobic chickens (causing Prof. Tyson to go out and manually rearrange the chickens). Devise such an algorithm and save the chickens!

Problem 9. Architects 'R Us [15 points]

You are assisting Prof. Gehry with designing the shape of a new room in the Stata Center. The professor has given you n columns, each of the same unit thickness, but with different heights: $A[1], A[2], \ldots, A[n]$. He asks you to permute the columns in a line to define the shape of the room. To make matters difficult, MIT wants to be able to hang a large rectangular picture on the columns. If j consecutive columns in your order all have a height of at least k, then we can hang a rectangle of size $j \cdot k$.

The example below contains 3 consecutive columns with heights of at least 4, so we can hang a rectangle of area 12 on the first three columns.



i	1	2	3	4	5	6	7	8
A[i]	6	4	5	0	2	7	1	2

(a) Give an efficient algorithm to find the largest area of a hangable rectangle for the *initial* order $A[1], A[2], \ldots, A[n]$ of columns.

(b) Devise an efficient algorithm to permute the columns into an order that maximizes the area of a hangable rectangle.

Problem 10. Guess Who? [10 points]

Woody the woodcutter will cut a given log of wood, at any place you choose, for a price equal to the length of the given log. Suppose you have a log of length L, marked to be cut in n different locations labeled $1, 2, \ldots, n$. For simplicity, let indices 0 and $n + 1$ denote the left and right endpoints of the original log of length L. Let d_i denote the distance of mark i from the left end of the log, and assume that $0 = d_0 < d_1 < d_2 < \ldots < d_n < d_{n+1} = L$. The ***wood-cutting problem*** is the problem of determining the sequence of cuts to the log that will cut the log at all the marked places and minimize your total payment. Give an efficient algorithm to solve this problem.

Problem 11. Varying variance [20 points]

For a set S of numbers, define its ***average*** to be $\bar{S} = \frac{1}{|S|}\sum_{s\in S} s$, and its ***variance*** to be $V(S) = \frac{1}{|S|}\sum_{s\in S}(s-\bar{S})^2$.

A ***segment variance data structure*** supports the following operations on an array $A[1\ldots n]$:

- **Assignment:** given an index i and a value a, set $A[i] = a$.
- **Segment variance:** given a pair of indices i and j, compute $V(\{A[i], A[i+1], \ldots, A[j]\})$.

Initially all entries in A are set to 0.

Design a data structure that supports both operations in $O(\log n)$ time.

SCRATCH PAPER

SCRATCH PAPER