Final Exam

- Do not open this quiz booklet until 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.
- You have 180 minutes to earn 180 points. Do not spend too much time on any one problem. Read them all first, and attack them in the order that allows you to make the most progress.
- This exam is closed book. You may use **three** $8\frac{1}{2}'' \times 11''$ 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.
- Don't waste time and paper rederiving facts that we have studied. It is sufficient to cite known results.
- When writing an algorithm, a **clear** description will suffice. Pseudo-code isn't required.
- When asked for an algorithm, your algorithm should have the time complexity specified in the problem with a correct analysis. If you cannot find such an algorithm, you will generally receive partial credit for a slower algorithm **if you analyze your algorithm correctly**.
- 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.

Problem	Parts	Points	Grade	Grader
1	2	2		
2	1	30		
3	2	30		
4	2	28		
5	3	30		
6	3	30		
7	2	30		
Total		180		

Name: ____________________

Friday Recitation:	Aleksander **11 AM**	Arnab **12 PM**	Alina **3 PM**	Matthew **4 PM**

Problem 1. What is Your Name? [2 points] (2 parts)

(a) [1 point] Flip back to the cover page. Write your name there.

(b) [1 point] Flip back to the cover page. Circle your recitation section.

Problem 2. Storing Partial Maxima [30 points] (1 part)

6.006 student, Mike Velli, wants to build a website where the user can input a time interval in history, and the website will return the most exciting sports event that occurred during this interval. Formally, suppose that Mike has a chronologically sorted list of n sports events with associated integer "excitement factors" $e_1, \ldots, e_n$. You can assume for simplicity that n is a power of 2. A user's query will consist of a pair (i, j) with $1 \leq i < j \leq n$, and the site is supposed to return $\max(e_i, e_{i+1}, \ldots, e_j)$.

Mike wishes to minimize the amount of computation per query, since there will be a lot of traffic to the website. If he precomputes and stores $\max(e_i, \ldots, e_j)$ for every possible input (i, j), he can respond to user queries quickly, but he needs storage $\Omega(n^2)$ which is too much.

In order to reduce storage requirements, Mike is willing to allow a small amount of computation per query. He wants to store a cleverer selection of precomputed values than just $\max(e_i, \ldots, e_j)$ for every (i, j), so that for any user query, the server can retrieve two precomputed values and take the maximum of the two to return the final answer. Show that now only $O(n \log n)$ values need to be precomputed.

Problem 3. Longest Simple Cycle [30 points] (2 parts)

Given an unweighted, directed graph $G = (V, E)$, a path $\langle v_1, v_2, ..., v_n \rangle$ is a set of vertices such that for all $0 < i < n$, there is an edge from v_i to v_{i+1}. A cycle is a path such that there is also an edge from v_n to v_1. A *simple path* is a path with no repeated vertices and, similarly, a *simple cycle* is a cycle with no repeated vertices. In this question we consider two problems:

- LONGESTSIMPLEPATH: Given a graph $G = (V, E)$ and two vertices $u, v \in V$, find a simple path of maximum length from u to v or output NONE if no path exists.
- LONGESTSIMPLECYCLE: Given a graph $G = (V, E)$, find a simple cycle of maximum length in G.

(a) [20 points] Reduce the problem of finding the longest simple path to the problem of finding the longest simple cycle. Prove the correctness of your reduction and show that it runs in polynomial time in $|V|$ and $|E|$.

(b) [10 points] As we discussed in class, finding a longest simple path is NP-Hard. Therefore, there is no known algorithm that, on input u, v, and G returns a longest simple path from u to v in polynomial time. Using this fact (which you do not need to prove) and Part (a), show that there is no known polynomial time algorithm that can find a longest simple cycle in a graph.

Note: If you were unable to solve Part (a), you may assume an algorithm SIMPLEPATHFROMCYCLE for finding a longest simple path from u to v that runs in time polynomial in L, $|V|$, and $|E|$ where L is the running time of a black-box algorithm for solving LONGESTSIMPLECYCLE.

Problem 4. Closest pair [28 points] (2 parts)

We are interested in finding the closest pair of points in the plane, closest in the sense of the rectilinear distance (also called the Manhattan or L_1 distance). The rectilinear distance between two points p_1 and p_2 on the plane is defined as $d(p_1, p_2) = |x_1 - x_2| + |y_1 - y_2|$, where the x's and y's are the first and second coordinates, respectively. In the first part we consider a one-dimensional version of the problem as a warm-up. In both cases, coordinates of the points are real numbers with k significant digits beyond the decimal point, k constant.

(a) [8 points] Warm up - Provide an efficient way to find a closest pair among n points in the interval $[0, 1]$ on the line. Full credit will be given to the most efficient algorithm with a correct analysis.

(b) [20 points] Case of the plane - A divide and conquer approach for n points in the square $[0, 1] \times [0, 1]$. Here is a possible strategy. Divide the set of points into two sets of about half size: those on the right of the x-coordinate median, and those on the left. Recursively, find the closest pair of points on the right, and the closest pair of points on the left and let δ_r and δ_l be the corresponding distances. The overall closest pair is either the minimum of these two options, or corresponds to a pair where one point is on the right of the median and the other is on the left. Given δ_r and δ_l, there should be an efficient way to find the latter. Explain how; then write the full recurrence for the running time of this approach; and conclude with its overall running time.

Problem 5. APSP Algorithm for Sparse Graphs [30 points] (3 parts)

Let $G = (V, E)$ be a weighted, directed graph that can have some of the weights negative. Let $n = |V|$ and $m = |E|$, and assume that G is strongly connected, i.e., for any vertex u and v there is a path from u to v in G.

We want to solve all-pairs-shortest-path problem (APSP) in G, i.e., we want to either find all the vertex-to-vertex distances $\{\delta(v, u)\}_{v,u \in V}$, or report existence of a negative-length cycle. We will design an algorithm for this task that runs in $O(mn + n^2 \log n)$ time. (Note that when G is not dense, i.e., when m is $o(n^2)$, the running time of this algorithm is asymptotically better than the one of the Floyd-Warshall algorithm.)

(a) [5 points] Fix some vertex $t \in V$ and consider the vertex potential $\lambda_t(u) = \delta(u, t)$ where $\delta(u, t)$ is the shortest path from $u \in V$ to t. Give an algorithm for calculating $\lambda_t(u)$ for all $u \in V$ and analyze the running time.

(b) [5 points] Show $\lambda_t(u)$, the potential from Part (a), is a feasible potential even if some of the original weights are negative.

(c) [20 points] Show how you use the vertex potential from Parts (a) and (b) to solve APSP in G in $O(mn + n^2 \log n)$ time, including the time it takes to calculate the vertex potential.

Problem 6. Airplane Scheduling [30 points] (3 parts)

Consider the runway reservation system from PS2. Recall that we kept track of requested landing times from airplanes by storing them in a balanced binary search tree. In that problem set, we required, for safety, that no landing time be within three minutes of any other landing time in the tree. We showed we could insert into the tree, delete from the tree, and check the validity of a landing time in $O(\log n)$ time.

Sometimes severe weather hits, and the 3-minute window between flights just isn't safe, so a new window size is determined. In these cases, an extra runway might be opened up at a nearby airport to take flights which don't fit within the new window. For all parts, you may use data structure augmentations provided that you explain the augmentation. Its maintenance may not increase the asymptotic running time of other operations, but you are not required to prove this.

(a) [5 points] Provide a very fast (constant-time) algorithm to determine if there are any flights which are not valid with the new window so that the extra runway can start opening immediately.

(b) [15 points] Also provide a slower algorithm which locates which specific flights are not valid within the new window so they can be rescheduled. For example, assuming a window of 3 minutes with flights at times 28, 31, 35, 40, 43, 48, 53, 57, 60, if the window size were expanded to 4 minutes, the algorithm should return that 28, 31, 40, 43, 57, 60 are invalid.

(c) [10 points] Can the running time of this slower algorithm be improved if we assume bounds on the maximum size the window could become?

Problem 7. Traveling on a Budget [30 points] (2 parts)

Arthur Dent has \$500 and 1000 hours to go from Cambridge, MA, to Berkeley, CA. He has a map of the States represented as a directed graph $G = (V, E)$. The vertices of the graph represent towns, and there is a directed edge $e = (A, B)$ from town A to town B if there is some means of public transportation connecting the two towns. Moreover, the edge is labeled with a pair (m_e, t_e), representing the cost $m_e \in \{0, 1, \ldots\}$ in dollars of transportation from A to B and the time $t_e \in \{0, 1, \ldots\}$ in hours that it takes to go from A to B.

Arthur is interested in finding a path from Cambridge to Berkeley that does not cost more than \$500 and does not take more than 1000 hours, while also minimizing the objective $5M^2 + 2T^2$, where M is the cost of the trip in dollars and T is the duration of the trip in hours. He was looking for an algorithm that runs in time polynomial in $|V|$ and $|E|$ . . .

(a) [10 points] Due to his lack of knowledge in algorithms, he gave up on the idea of respecting the budget and time constraints. At least he thought he could efficiently find the path minimizing the objective $5M^2 + 2T^2$. He tried modifying Dijkstra's algorithm as follows: If an edge e was labeled (m_e, t_e), he assigned it a weight $w_e = 5m_e^2 + 2t_e^2$, and ran Dijkstra's algorithm on the resulting weighted directed graph. Show that Arthur's algorithm may return incorrect results, i.e. return a path that does not minimize the objective $5M^2 + 2T^2$.

(b) [20 points] Now provide an algorithm that solves Arthur's original problem in time polynomial in $|E|$ and $|V|$. Your algorithm should find the path that minimizes the objective $5M^2 + 2T^2$, while at the same time respecting the constraints $M \leq 500$ and $T \leq 1000$. Please describe your algorithm precisely, and justify its correctness and running time. More credit will be given to faster algorithms, provided that the analysis of the algorithm is correct.

[*Hint 1: Use dynamic programming.*]

[*Hint 2: For each town A, integer values $m \leq 500$ and $t \leq 1000$, either there is a path from Cambridge to A that requires cost m and time t, or there is not.*]

SCRATCH PAPER

SCRATCH PAPER