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 the top of *every* page of this quiz booklet.
- You have 180 minutes to earn a maximum of 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.
- **You are allowed three double-sided letter-sized sheets with your own notes**. No calculators, cell phones, or other programmable or communication devices are permitted.
- Write your solutions in the space provided. If you need more space, write on the scratch pages at the end of the exam, and refer to the scratch pages in the solution space provided. Pages will be scanned and separated for grading.
- Do not waste time and paper rederiving facts that we have studied. Simply cite them.
- When writing an algorithm, a **clear** description in English will suffice. Pseudo-code is not required. But be sure to prove the required bound on **running time** and explain **correctness**. Even if your running time is slower than the requested bound, you will likely receive partial credit if your algorithm and analysis are correct.
- If you give a correct answer *and* an incorrect answer to the same problem (without labeling the latter as incorrect), you will receive partial but not full credit (approximately the average of the answers).
- **Pay close attention to the instructions for each problem**. Depending on the problem, partial credit may be awarded for incomplete answers.

Problem	Parts	Points
0 : What is Your Name?	2	2
1 : True or False, and Justify	10	50
2 : Fast & Furious Coding	4	32
3 : Shortcut through the Circle of Life	1	10
4 : I'm All About That Base	3	15
5 : Escape the Graph	1	10
6 : Groundhog Graphs	1	10
7 : Legal Need For Speed	2	18
8 : Squeezing into the Page Limit	3	18
9 : Love Triangles	3	15
Total		180

Name: _____

Problem 0. [2 points] **What is Your Name?** (2 parts)

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

(b) [1 point] Write your name on top of each page, including the scratch paper!

Problem 1. [50 points] **True or False, and Justify** (10 parts)

Circle T or F to indicate whether the statement is true or false, and explain your answer: if true, sketch a proof; and if false, give a counterexample. Your justification is worth more than your true or false designation.

(a) T F [5 points] The recurrence $T(n) = 2T(n/2) + O(n \lg n)$ solves to $O(n \lg n)$ by the Master Theorem.

(b) T F [5 points] As implemented in class, heapsort runs in $O(n)$ time on a sorted array of n elements.

(c) T F [5 points] Every algorithm that finds a peak (local maximum) in a 1D array A of n elements, in the comparison model, uses $\Omega(\lg n)$ comparisons in the worst case.

(d) T F [5 points] While running depth-first search, if the discover time of a node u is before the discover time of node v, then u must be an ancestor of v.

(e) T F [5 points] For all weighted directed graphs $G = (V, E, w)$, Dijkstra's algorithm is asymptotically faster than an $O(V^{1.3})$-time algorithm.

(f) T F [5 points] To compute $\sqrt[3]{6.006}$ using gradient descent, the method we learned in class would use the update equation $x_{i+1} := x_i^3 - 6.006$.

(g) T F [5 points] The number of intersection points among n lines is $O(n^2)$.

(h) T F [5 points] Given n line segments in the plane, we can determine whether any of them cross each other in $O(n \log n)$ time.

(i) T F [5 points] The following problem is in P: given a weighted directed graph $G = (V, E, w)$, vertices $s, t \in G$, and a number ℓ, decide whether the shortest-path weight in G from s to t is $\leq \ell$.

(j) T F [5 points] The following problem is in NP: given a weighted directed graph $G = (V, E, w)$, vertices $s, t \in G$, and a number ℓ, decide whether there is a **simple** path in G from s to t of weight $\geq \ell$.

Problem 2. [32 points] **Fast & Furious Coding** (4 parts)

Prof. Toretto is new to Python and its cost model, but needs to solve some important algorithmic problems for his upcoming fateful car heist. He has written **correct** code, but it's running too slow. For each of the following implementations, answer the following questions:

1. What is the asymptotic running time of the function right now? (Briefly justify your answer. If it's exponential, you don't need to give a precise bound—"exponential" will suffice. If it's polynomial, give a precise Θ bound.)
2. Find a small fix to make the function more efficient but still correct. (Your fix should give the optimal possible running time, and should involve changing only a few lines of code. **Just describe your changes to the algorithm in English; don't waste time by copying/writing code**. This part is worth the most points.)
3. What is the new asymptotic running time of the function, after your fix? (Briefly justify your answer. Give a precise Θ bound. It should be polynomial.)

Assume that input A is an array of n integers, and input k is a nonnegative integer.

(a) [8 points]

```
# Checks whether two (not necessarily distinct)
# elements in A sum to k.
def check_sum_to_k(A, k):
    for x in A:
        if k - x in A:
            return True
    return False
```

(b) [8 points]

```
# Extracts the k smallest elements of A
# (possibly destroying A).
import heapq
def get_min_k(A, k):
    ans = []
    for i in range(k):
        heapq.heapify(A)
        # A.pop(0) removes the element at index 0,
        # and returns that element. Because A is a heap,
        # the first element must be the minimum.
        x = A.pop(0)
        ans.append(x)
    return ans
```

(c) [8 points]

```
# Counts the number of size-k subarrays A[i:i+k]
# of A that have a sum of zero.  Assume k > 0.
def get_num_zero_subarrays(A, k):
    ans = 0
    for i in range(len(A) - k + 1):
        # sum(X) iterates through the elements of X and
        # calculates their sum.
        s = sum(A[i:i+k])
        if s == 0:
            ans += 1
    return ans
```

(d) [8 points]

```
# Computes the length of the longest increasing subsequence
# in the suffix A[k:], assuming the sequence starts with A[k].
def longest_increasing_subsequence(A, k):
    best = 1  # Sequence could be A[k] by itself.
    # Guess that the sequence goes from A[k] to A[m].
    for m in range(k+1, len(A)):
        if A[m] > A[k]:  # Sequence must be increasing.
            solution = 1 + longest_increasing_subsequence(A, m)
            if solution > best: best = solution
    return best
# You can assume that this function gets called only once
# by another function (i.e., ignoring recursions).
```

Problem 3. [10 points] **Shortcut through the Circle of Life** (1 part)

Prof. Timon has a weighted undirected graph $G = (V, E, w)$ that is just a cycle (i.e., G is connected and every vertex has degree exactly 2). It has two distinguished vertices $s, t \in V$.

Design an $O(V + E)$-time algorithm to find the shortest path in G from s to t.

Problem 4. [15 points] **I'm All About That Base** (3 parts)

Tired of regular old binary heaps, your colleagues at 6006LE decide to explore ***k-ary min-heaps*** which have the following properties:

1. Every node has exactly k children, except for the leaves which have zero children and are all at the same level, and except for the rightmost node in the level above the leaf level which has $\leq k$ children.
2. The key of every node is less than or equal to the keys of its children,

(a) [2 points] What is the height of an n-node k-ary min-heap? You can use Θ notation, but analyze in terms of both n and k.

(b) [5 points] Design an array representation $A[0\,..\,n-1]$ of an n-node k-ary min-heap that lets you find the parent and cth child ($1 \leq c \leq k$) of node $A[i]$ in constant time (without pointers). You do not need to prove your answer. *Hint:* Make sure your solution works for $k = 2$ (regular binary heaps).

(c) [8 points] Describe how to implement EXTRACT-MIN in an n-node k-ary min-heap. Analyze the asymptotic running time of your algorithm in terms of n and k. For full credit, your algorithm should run in $O(\log n)$ time whenever $k = O(1)$.

Problem 5. [10 points] **Escape the Graph** (1 part)

Prof. Houdini needs to find the quickest way out of any node in a graph. Design a data structure for representing a weighted directed graph $G = (V, E, w)$ supporting the following operations and time bounds:

FASTEST-ESCAPE(u): Find and delete the **minimum-weight** outgoing edge $(u, v) \in E$ from a given vertex $u \in V$, in $O(\log \textit{out-degree}(u))$ time.

INSERT(u, v): Insert an edge (u, v) into E in $O(\log \textit{out-degree}(u))$ time.

ADJ(u): List all outgoing edges $(u, v) \in E$ from a given vertex $u \in V$, in $O(\textit{out-degree}(u))$ time.

Problem 6. [10 points] **Groundhog Graphs** (1 part)

For every positive integer n, show how to construct

1. an unweighted undirected graph $G = (V, E)$ with $|V| = n$ vertices,
2. a source vertex $s \in V$, and
3. an ordering of the vertices in V

such that running DFS and BFS on G from start node s discovers the nodes in the same order. (Recall that the ***discovery time*** of a node v is the first time v is encountered during one of the search algorithms.)

Problem 7. [18 points] **Legal Need For Speed** (2 parts)

You're racing through Cambridge, which is described by a weighted directed graph $G = (V, E, w)$ where vertices represent intersections and edges represent roads. The weight $w(u, v)$ of road (u, v) is the positive integer of time it takes to traverse the road from u to v.

Unfortunately, the city has **traffic lights**: to leave an intersection $v \in V$, you must wait until the current time is an integer multiple of $f(v)$. Your goal is to find the shortest-time path from $s \in V$ to $t \in V$ in $O(E + V \lg V)$ time, assuming you start at s at time 0 (and thus can leave it immediately).

(a) [8 points] Assume that $f(v)$ is the same value F for all vertices $v \in V$. Show how to modify the weights w into w' such that running Dijkstra on $G' = (V, E, w')$ from s correctly solves the problem. You do not need to prove correctness.

(b) [10 points] Now consider the general case where $f(v)$ can be different for different vertices $v \in V$. Show how to change the RELAX(u, v) subroutine so that running Dijkstra on G from s correctly solves the problem. You do not need to prove correctness.

Problem 8. [18 points] **Squeezing into the Page Limit** (3 parts)

You've already written your solutions to Problem 8 of this final, but there is a requirement that you get it to fit on exactly one page, which is exactly k lines long. Your solutions consist of n words given by array $words[0\,..\,n-1]$. You've defined a function $badness(i, j)$ that measures how ugly it would be to squeeze words $words[i : j]$ onto one line.

You've defined the following **subproblems**: $DP[i, k']$ is the optimal justification (of minimum total badness) of suffix $words[i :]$ into exactly k' lines.

(a) [6 points] How many subproblems are there? Use Θ notation. You do not need to prove your answer.

(b) [6 points] Give a recurrence relating subproblem solutions. You do not need to prove correctness.

Hint: Recall from class the dynamic programming recurrence for justifying text into lines so as to minimize total badness of the lines:

$$DP[i] = \min\Big(badness(i, j) + DP[j] \text{ for } j \text{ in range}(i + 1, n + 1)\Big).$$

(c) [6 points] What is the running time of the resulting dynamic program? You do not need to prove your answer.

Problem 9. [15 points] **Love Triangles** (3 parts)

Suppose you are given a list of n distinct points $p_1, p_2, \ldots, p_n$ with coordinates (x_1, y_1), (x_2, y_2), $\ldots$, (x_n, y_n). Assume that all coordinates are $\Theta(\log n)$-bit integers, and operations on these integers take constant time.

(a) [5 points] Design an expected $O(n)$-time algorithm to list all distinct x coordinates (in no particular order) and, for each such x coordinate, the number of points with that x coordinate. (Even if you don't solve this part, assume you have a solution for future parts.)

(b) [5 points] Design an expected $O(n)$-time algorithm to count the number of unordered pairs of distinct points $\{p_i, p_j\}$ having the same x coordinate ($x_i = x_j$) or same y coordinate ($y_i = y_j$), i.e., the number of unordered pairs of distinct points $\{p_i, p_j\}$ that define a line parallel to the x or y axis. (*Hint:* Use the solution to part (a) on both x and y.)

(c) [5 points] Design an expected $O(n)$-time algorithm to count the number of (right) triangles $\triangle p_i p_j p_k$ having one side parallel to the x axis and another side parallel to the y axis. (*Hint:* Use the solution to part (a) on both x and y.) Partial credit will be awarded for slower but correct algorithms.

SCRATCH PAPER 1. DO NOT REMOVE FROM THE EXAM.

You can use this paper to write a longer solution if you run out of space, but be sure to refer to the scratch paper next to the question itself. Also be sure to write your name on the scratch paper!

SCRATCH PAPER 2. DO NOT REMOVE FROM THE EXAM.

You can use this paper to write a longer solution if you run out of space, but be sure to refer to the scratch paper next to the question itself. Also be sure to write your name on the scratch paper!

SCRATCH PAPER 3. DO NOT REMOVE FROM THE EXAM.

You can use this paper to write a longer solution if you run out of space, but be sure to refer to the scratch paper next to the question itself. Also be sure to write your name on the scratch paper!

JOKE QUESTIONS. <u>DO NOT ANSWER THESE.</u>

BUT ALSO DO NOT REMOVE FROM THE EXAM.

- [0 points] What are the names of the professors? (Do not look back at page 1.)
- [0 points] What is the name of your recitation instructor?
- [0 points] Write every misspelling of "Dÿkstra" that at least one student wrote on this exam.
- [0 points] For each problem on this exam, guess who wrote it.
- [0 points] What percentage of students will not get full credit on Problem 0 on this exam?
- [0 points] How many emails were sent to 6.006-staff@mit.edu and/or free-fruit@mit.edu during this semester?
- [0 points] How many questions were asked on Piazza? How many posts total?
- [0 points] If you were an algorithm, which algorithm would you be?
- [0 points] Which algorithm is the best algorithm?
- [0 points] Write the asymptotically slowest sorting algorithm, among all students' answers to this question on this final.