Final

- 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. Skim 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 sheet with your own notes**. No calculators, cell phones, or other programmable or communication devices are permitted.
- Write your solutions in the space provided. Pages will be scanned and separated for grading. If you need more space, write "Continued on S1" (or S2, S3, S4, S5, S6, S7) and continue your solution on the referenced scratch page at the end of the exam.
- Do not waste time and paper rederiving facts that we have studied in lecture, recitation, or problem sets. Simply cite them.
- When writing an algorithm, a **clear** description in English will suffice. Pseudo-code is not required. Be sure to argue that your **algorithm is correct**, and analyze the **asymptotic running time of your algorithm**. Even if your algorithm does not meet a requested bound, you **may** receive partial credit for inefficient solutions that are correct.
- **Pay close attention to the instructions for each problem**. Depending on the problem, partial credit may be awarded for incomplete answers.

Problem	Parts	Points
1: Information	2	2
2: Decision Problems	10	40
3: Sorting Sorts	2	24
4: Pythagorean Quad	1	14
5: Animal Counting	1	20
6: Limited Connections	1	14
7: On the Street	1	14
8: RGB Graph	1	18
9: Separated Subsets	1	16
10: A Feast for Crowns	1	18
Total		180

Name: _____

School Email: _____

Problem 1. [2 points] **Information** (2 parts)

(a) [1 point] Write your name and email address on the cover page.

(b) [1 point] Write your name at the top of each page.

Problem 2. [40 points] **Decision Problems** (10 parts)

For each of the following questions, circle either **T** (True) or **F** (False), and **briefly** justify your answer in the box provided (a single sentence or picture should be sufficient). Each problem is worth 4 points: 2 points for your answer and 2 points for your justification. **If you leave both answer and justification blank, you will receive 1 point**.

(a) **T** **F** $2^{2n} \in \Theta(2^n)$.

(b) **T** **F** If $T(n) = \frac{9}{4}T\left(\frac{2}{3}n\right) + n^2$ and $T(1) = \Theta(1)$, then $T(n) = O(n^2)$.

(c) **T** **F** Performing an $O(1)$ amortized operation n times on an initially empty data structure takes worst-case $O(n)$ time.

(d) T F Given an array A containing n comparable items, sort A using merge sort. While sorting, each item in A is compared with $O(\log n)$ other items of A.

(e) T F Given a binary min-heap storing n items with comparable keys, one can build a Set AVL Tree containing the same items using $O(n)$ comparisons.

(f) T F Given a directed graph $G = (V, E)$, run breadth-first search from a vertex $s \in V$. While processing a vertex u, if some $v \in \text{Adj}^+(u)$ has already been processed, then G contains a directed cycle.

(g) T F Run Bellman-Ford on a weighted graph $G = (V, E, w)$ from a vertex $s \in V$. If there is a **witness** $v \in V$, i.e., $\delta_{|V|}(s, v) < \delta_{|V|-1}(s, v)$, then v is on a negative-weight cycle of G.

(h) T F Floyd–Warshall and Johnson's Algorithm solve all-pairs shortest paths in the same asymptotic running time when applied to weighted **complete** graphs, i.e., graphs where every vertex has an edge to every other vertex.

(i) T F If there is an algorithm to solve 0-1 Knapsack in polynomial time, then there is also an algorithm to solve Subset Sum in polynomial time.

(j) T F Suppose a decision problem A has a pseudopolynomial-time algorithm to solve A. If P $\neq$ NP, then A is not solvable in polynomial time.

Problem 3. [24 points] **Sorting Sorts**

(a) [12 points] An integer array A is ***k*-even-mixed** if there are exactly k even integers in A, and the odd integers in A appear in sorted order. Given a k-even-mixed array A containing n distinct integers for $k = \lceil n/\lg n \rceil$, describe an $O(n)$-time algorithm to sort A.

(b) [12 points] Let A be an array of n pairs of positive integers (x_i, y_i) with $x_i, y_i < n^2$ for all $i \in \{0, \ldots, n-1\}$. The **power** of pair (x, y) is the integer $x + n^y$. Describe an $O(n)$-time algorithm to sort the pairs in A increasing by power.

Problem 4. [14 points] **Pythagorean Quad**

A **Pythagorean Quad** consists of four integers (a, b, c, d) such that $d = \sqrt{a^2 + b^2 + c^2}$. Given an array A containing n distinct positive integers, describe an $O(n^2)$-time algorithm to determine whether four integers from A form a Pythagorean Quad, where integers from A may appear more than once in the Quad. State whether your running time is worst-case, expected, and/or amortized.

Problem 5. [20 points] **Animal Counting**

PurpleRock Park is a wildlife reserve, divided into zones, where each zone has a park ranger who records current **sightings** of animals of different **species** over time. Old animal sightings are periodically removed from the database. A species s is **common** if current park records contain at least 100 sightings of species s within any single zone of the park.

Describe a database to store animal sightings, supporting the following four operations, where n is the number of sightings stored in the database at the time of the operation. State whether your running times are worst-case, expected, and/or amortized.

`initialize()`	Initialize an empty database in $O(1)$ time
`add_sighting(s, i)`	Record a newest sighting of species `s` in zone `i` in $O(\log n)$ time
`remove_oldest()`	Remove the oldest sighting stored in the database in $O(\log n)$ time
`is_common(s)`	Return whether species `s` is common based on sightings that have not yet been removed from the database in $O(1)$ time

Problem 6. [14 points] **Limited Connections**

For any weighted graph $G = (V, E, w)$ and integer k, define $\boldsymbol{G_k}$ to be the graph that results from removing every edge in G having weight k or larger.

Given a connected undirected weighted graph $G = (V, E, w)$, where every edge has a unique integer weight, describe an $O(|E| \log |E|)$-time algorithm to determine the largest value of k such that G_k is not connected.

Problem 7. [14 points] **On the Street**

Friends Dal and Sean want to take a car trip across the country from Yew Nork to Fan Sancrisco by driving between cities during the day, and staying at a hotel in some city each night. There are n cities across the country. For each city c_i, Dal and Sean have compiled:

- the positive integer expense $h(c_i)$ of **staying at a hotel** in city c_i for one night; and
- a list L_i of the at most 10 other cities they could drive to **in a single day** starting from city c_i, along with the positive integer expense $g(c_i, c_j)$ required to drive directly from c_i to c_j for each $c_j \in L_i$.

Describe an $O(nd)$-time algorithm to determine whether it is possible for Dal and Sean to drive from Yew Nork to Fan Sancrisco in at most d days, spending at most b on expenses along the way.

Problem 8. [18 points] **RGB Graph**

Let $G = (V, E, w)$ be a weighted directed graph. Let $c : V \to \{r, g, b\}$ be an assignment of each vertex v to a **color** $c(v)$, representing red, green, or blue respectively. For $x \in \{r, g, b\}$,

- let V_x be the set of vertices with color x, i.e., $V_x = \{v \in V \mid c(v) = x\}$; and
- let E_x be the set of edges outgoing from vertices in V_x, i.e., $E_x = \{(u, v) \in E \mid u \in V_x\}$.

Suppose graph G and coloring c have the following properties:

1. Every edge in E either connects two vertices of the same color, goes from a red vertex to a green vertex, or goes from a green vertex to a blue vertex.
2. $|V_r| = |E_r| = O(|V|)$, and edges in E_r have identical positive integer weight w_r.
3. $|V_g| = |E_g| = O(|V|^{0.99})$, and edges in E_g have nonnegative integer weights.
4. $|V_b| = |E_b| = O(\sqrt{|V|})$, and edges in E_b can have positive or negative integer weights.

Given G, c, a red vertex $s \in V_r$, and a blue vertex $t \in V_b$, describe an $O(|V|)$-time algorithm to compute $\delta(s, t)$, the minimum weight of any path from s to t.

Problem 9. [16 points] **Separated Subsets**

For any set S of integers and for any positive integers m and k, an **(m, k)-separated subset** of S is any subset $S' \subseteq S$ such that S' sums to m and every pair of distinct integers $a, b \in S'$ satisfies $|a - b| \geq k$. Given positive integers m and k, and a set S containing n distinct positive integers, describe an $O(n^2 m)$-time algorithm to **count the number** of (m, k)-separated subsets of S.

(When solving this problem, you may assume that a single machine word is large enough to hold any integer computed during your algorithm.)

Problem 10. [18 points] **A Feast for Crowns**

Ted Snark is arranging a feast for the Queen of Southeros and her guests, and has been tasked with seating them along one side of a long banquet table.

- Ted has a list of the $2n$ guests, where each guest i has a known distinct positive integer f_i denoting the guest's **favor** with the Queen.
- Ted must seat the guests **respectfully**: the Queen must be seated in the center with n guests on either side so that guests' favor monotonically decreases away from the Queen, i.e., any guest seated between a guest i and the Queen must have favor larger than f_i.
- Additionally, every guest hates every other guest: for every two guests i, j, Ted knows the positive integer **mutual hatred** $d(i, j) = d(j, i)$ between them.

Given Ted's guest information, describe an $O(n^3)$-time algorithm to determine a respectful seating order that minimizes the sum of mutual hatred between pairs of guests seated next to each other. **Significant partial credit** will be awarded to correct $O(n^4)$-time 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 write "Continued on S1" on the problem statement's page.

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 write "Continued on S2" on the problem statement's page.

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 write "Continued on S3" on the problem statement's page.

SCRATCH PAPER 4. 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 write "Continued on S4" on the problem statement's page.

SCRATCH PAPER 5. 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 write "Continued on S5" on the problem statement's page.

SCRATCH PAPER 6. 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 write "Continued on S6" on the problem statement's page.

SCRATCH PAPER 7. 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 write "Continued on S7" on the problem statement's page.

6.006 Introduction to Algorithms

Spring 2020