CSE 373: 24 Wi Final

Name:	UW Email:	@uw.edu

Instructions

- The allotted time is **50** minutes. Please do not turn the page until the staff says to do so.
- This is an open-book and open-notes exam. You are NOT permitted to access electronic devices including calculators.
- We can only give partial credit for work that you've written down.
- If you want to use an algorithm discussed in class, just cite the name of the algorithm, like "run BFS to…"
- If you use a data structure that requires data to be comparable (e.g. a search tree or heap), you should explain how the data is compared. For example, "I will use a binary min heap of (`name, count`) pairs where pairs are **compared by `count` in ascending order**"
- If you use a map, explain what the keys and values are.
- Unless otherwise noted, when we ask for algorithm runtime, it must be **simplified and tight**.
- **You may assume that all hash functions and operations (find, add, remove) are O(1).**
- If you run out of room on a page, indicate where the answer continues. Try to avoid writing on the very edges of the pages: we scan your exams and edges often get cropped off.

Advice

- If you feel like you're stuck on a problem, you may want to skip it and come back at the end if you have time.
- Relax and take a few deep breaths. You got this :-)

Questions
1. Heaps
2. Sorting
3. Graphs

Problem 1 (Heaps):

Yafqa is assigning each problem on the final to a TA to grade. Some TAs have already volunteered to grade certain problems, so Yafqa needs to assign the remaining problems. He wants the workload to be as evenly distributed as possible.

To be more specific, there are `k` problems and `n` TAs, where `k > n`. You are given an array `problemAssignments` of length `k`, where `problemAssignments[i]` is either the name of the TA who volunteered for the `ith` problem or `null` if no TA has been assigned to it yet. You are also given an array `tas` of all the TA names.

Let `min` be the fewest number of problems that any TA has been assigned, and let `max` be the greatest. Your task is to assign the remaining problems so that the difference between `max` and `min` is as small as possible. The method header would look something like this:

```
public void assignProblems(String[] problemAssignments, String[] tas)
```

Example

Suppose Yafqa has 7 problems to assign, and some TAs have volunteered as below:

`problemAssignments`

Problem 0	Problem 1	Problem 2	Problem 3	Problem 4	Problem 5	Problem 6
Azita	null	Simon	Simon	null	Josh	null

And suppose we have the following TAs:

`tas`

TA 0	TA 1	TA 2	TA 3
Azita	Simon	Josh	Emily

Then to make the workload as even as possible, we would assign the remaining problems like so:

`problemAssignments`

Problem 0	Problem 1	Problem 2	Problem 3	Problem 4	Problem 5	Problem 6
Azita	Emily	Simon	Simon	Azita	Josh	Josh

If there are multiple valid solutions, you can make any of them. For example, problem 6 could have been assigned to Emily instead.

1) Describe how you would compute and store the number of problems each TA in `tas` **initially** volunteered for, **before assigning any of the remaining problems**. If a TA didn't volunteer for any problems, you should record the fact that they volunteered for 0 problems. Whatever you store this data in, it should **enable constant time lookup of a TA's initial count**, in practice. For an **upper bound**, your algorithm should run in O(n + k) time. Your answer should contain a precise description of what data structure(s) you're using, **following the instructions on the front page**, as well as **an algorithm in either plain English or pseudocode**. No justification necessary. (≤ 5 sentences)

2) For this and the following questions, assume you have a data structure that enables constant-time lookup of a TA's initial counts, as in (1). Describe any additional data structures you would use to assign the remaining problems (as described in the problem statement above). Your response should contain a precise description of your additional data structure(s), **following the exam instructions on the front page**. You can use any data structure described in class. You **do NOT need to** justify your choices. (<= 4 sentences)

3) Now describe how you would assign the remaining problems in `problemAssignments` as described above. Your response should contain an algorithm in either plain English or pseudocode. For an **upper bound**, your algorithm should run in O(n + klog(n)) time in the worst-case. You can abbreviate variable names if you'd like, but they should either be explained or obvious. You **do NOT need to** justify the runtime. (<= 5 sentences).

Problem 2 (Sorting):

For each of the following situations, name **one sorting algorithm** from lecture that fits the constraints. Choose from among `insertion, selection, heap, quick, merge, radix,` and `bucket`. Explain how your choice meets all constraints. If you make any assumptions about any sorting algorithm, you should explain those too, e.g. if you assume that quicksort always picks good pivots. (≤ 4 sentences each)

1. As an employee at Suzzallo Library, you have been tasked with sorting a recent delivery of electronic books. These books are already arranged in alphabetical order by author. You must now sort them by genre, such that books within the same genre remain sorted by author. However, given the constraints of an older library computer, you must use a sorting algorithm that utilizes limited memory.

2. As an employee at Tema, you have been asked to sort the usernames of all Tema users. However, you realize that the data set is incredibly large, so you want to ensure an efficient worst-case runtime.

3. As a data analyst at the University of Washington, you have conducted a survey to gather information from students about their favorite courses. The collected data includes the names of students, their student IDs, and their preferred courses. Now, you want to sort the list of students by their ID numbers, as quickly as possible. (You can assume that student ID numbers are integers from 0 to 1,000,000)

Problem 3 (Graphs):

Mr. Meow Meow decided to reveal their secret identity. They tell some students, those students tell some other students, and so on. You want to figure out how long it takes until the entire school knows their true identity.

Suppose you are given a list of triplets of the form (`gossipGiver`, `gossipReceiver`, `time`), indicating that it takes `time` minutes for `gossipGiver` to spread gossip to `gossipReceiver` (once `gossipGiver` has something to say to `gossipReceiver`). Note that just because `gossipGiver` gossips to `gossipReceiver` doesn't mean that `gossipReceiver` spreads gossip to `gossipGiver`. You are also given a list of all student names at UW. At time 0, Mr. Meow Meow starts revealing the news. You need to find the earliest time **t** in which everyone on campus knows about it, or output -1 if this is not possible.

You will solve this problem using a graph:

In one word, what should each vertex represent in the graph?

In one short sentence, what should each edge represent in the graph?

Should the edges in the graph be directed or undirected?

Select the best answer:

○	Directed
○	Undirected

Should the edges in the graph be weighted or unweighted?

Select the best answer:

○	Weighted
○	Unweighted

If you answered "Weighted" to the previous question, in one short sentence, what should each edge weight represent? If you answered "Unweighted" to the previous question, leave it blank.

Explain how you can determine the earliest time `t` in which everyone knows Mr. Meow Meow's identity, or whether no such time exists. If you plan to use one of the graph algorithms you've learned in class, you do not need to explain how the algorithm is implemented. For example, if you want to use breadth-first search in your answer, simply say "use BFS to achieve…" but if you plan to adapt an algorithm, be sure to explain any changes you would make. (≤ 4 sentences).