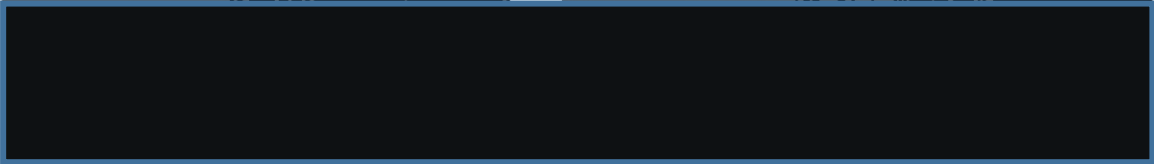


CSE 373: 23 Spring Final

@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.
- Read the directions carefully, especially for problems that require you to show work or provide an explanation.
- We can only give partial credit for work that you've written down.
- Unless otherwise noted, every time we ask for algorithm runtime, it must be **simplified and tight**.
- 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. Sorting
2. Graphs

Resubmission Details

- This exam will be graded out of 100 points. If you are not satisfied with your grade, you will be given the opportunity to resubmit it online and earn up to 50% of the missed points back.
- For example, a student scoring 80/100 points may receive up to 90/100 points on the resubmission.

1 Sorting

Given an String array of fruit names, you want to sort and print the fruits in increasing frequency order. You may assume there are at most k different types of fruits in the array, and the array is length N . If there are multiple valid orderings, print any of them.

Example:

- Array: [apple, pear, mango, mango, pear]
- Answer: [apple, pear, mango] or [apple, mango, pear]
- Explanation: Apple is the least frequent because it appears only 1 time. Both pear and mango appear more frequently than apple - each appearing 2 times. Thus either ordering is ok.

Please carefully review each of the following prompts. Please note that using data structures other than those specifically required by the prompt **may result in zero credit**. For example, if the problem only allows for the use of a single hashmap, using multiple auxiliary hashmaps may not earn any credit.

Design an algorithm that **only uses a hashmap and a binary min heap** as auxiliary data structures and has worst-case time complexity of $O(N + k \log k)$, assuming all hashmap operations are $O(1)$. Explain and justify both your algorithm's time and spatial complexity. Please limit your answer to 5 sentences.

We will loop through the input array once, which is n iterations, putting each element in a HashMap of type `Map<String, Integer>`. The first time we see an element, we will make a new key, value pair where the key is the fruit name & value is 1, but if it's not the first time, we will just increment the value by 1. After this, we will add our elements from the HashMap into the Binary min heap - each item has type `String` (the key) and `int` priority (the value). Then, we will call `removeMin()`, k times, printing out the string on each call. Thus, we have a total runtime of $O(n + k \log k)$ since `removeMin()` is $\log(k)$ runtime. Space complexity is $O(2k)$ b/c we used a HashMap & a minHeap of size k .

Assume the frequency of each fruit appearing in the input array is **unique**. Design an algorithm that **only uses a hashmap and an array** as auxiliary data structures and has worst-case time complexity of $O(N)$, assuming all hashmap operations are $O(1)$. Explain and justify both your algorithm's time and spatial complexity. Please limit your answer to 5 sentences.

First, we should put each element of the input array into a HashMap of `<String, Integer>`, which holds the fruit names as keys, and their frequencies as values (n iterations). Then we should create a new String array of size n . We should add the String keys from the HashMap into this new array, at index of their frequency value (k iterations). Finally, we should do a linear pass through the new array, printing out the array elements which have been initialized to a String value, (n iterations). This will result in fruits printed out from low to high frequencies, with runtime $O(2n + k) = O(n)$ and spatial complexity $O(k+n) = O(n)$.

2 Graphs

2.1 Roads

Consider a road construction log that documented roads built in Washington State. The log is in the form of a list of triples $[CityA, CityB, YearNum]$. The triple indicates that a road is built between $CityA$ and $CityB$ in year $YearNum$. The triples in the log follow an **arbitrary** ordering. You can assume that the roads in the log eventually connect together all cities in Washington State.

For example, $[Seattle, Bellevue, 1925]$ would mean a road was built between Seattle and Bellevue in the year 1925.

Your task is to find out the **first year** in which someone can travel from any city to any other city in Washington State via connected roads.

You decide to model this question using a graph.

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

Cities

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

roads between cities

Should the edges in the graph be directed or undirected?

<input type="radio"/>	Directed
<input checked="" type="radio"/>	Undirected

Select the best answer:

Should the edges in the graph be weighted or unweighted?

<input checked="" type="radio"/>	Weighted
<input type="radio"/>	Unweighted

Select the best answer:

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

years

Explain how you can determine the first year in which someone can travel from any city to any other city in Washington State via connected roads using the graph that you built. 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..."

Please limit your answer to 3 sentences.

~~We can traverse the graph starting at any node and perform BFS.~~ We can check whether the graph is strongly connected at a given year if there exists a spanning tree w/ edge weights less than that given year. To do so, we can apply Kruskal's algorithm and pick the minimum edge weights. The largest edge in this MST will be the first year we can travel to ANY city.

2.2 Alphabet

There is an ancient alphabetic system. Unlike English alphabets, where the orderings for each alphabet is known (letter a comes before b , and letter b comes before letter c , etc), the ordering for the ancient alphabetic system is **currently unknown**. You are given a list of words composed of the ancient alphabets. The list of words is sorted in **ascending language dictionary order** (refer to the example below). Your goal is to determine and return the **ascending order of each letter** in the ancient alphabet. You may assume there is always one and only one valid ordering of the letters from the given list.

Example:

- List of words: [ac , ab , c]
- Return: a, c, b

Explanation: You should **not** assume $a \leq b \leq c$, since the alphabetic system is unrelated to how the English alphabet is ordered. The letters a, b, c are used only for illustration purposes. Since ac is alphabetically less than ab , and both words share the same first letter a , we can reason that $c \leq b$ since the list is in ascending order. Similarly, since ab comes before c in the ancient alphabet system, then letter $a \leq c$ when comparing the first letter of the two words. Combining the two observations, the only valid ordering is $a \leq c \leq b$.

The code on the following page models the problem with a graph. You may assume each word in the list contains at least one letter, and the list contains at least two words.

```
public Map<Character, List<Character>> constructGraph(String[] words) {
```

```
    Map<Character, List<Character>> adjList = new HashMap<>();
```

```
    for (String word : words) {  $\}n$ 
```

```
        for (char c : word.toCharArray()) {
```

```
            adjList.put(c, new ArrayList<>());
```

```
        }
```

```
    }
```

$l = \text{avg word length}$

```
    for (int i = 0; i < words.length - 1; i++)  $\}n$ 
```

```
        String word1 = words[i];
```

```
        String word2 = words[i + 1];  $\}2$ 
```

```
        // a.startsWith(b) returns true if b is a prefix of a,
```

```
        // otherwise returns false
```

```
        if (word2.startsWith(word1) == false) { word1 not a prefix of word2
```

```
            for (int j = 0; j < Math.min(word1.length(), word2.length()); j++)
```

```
            {
```

```
                char charOne = word1.charAt(j);
```

```
                char charTwo = word2.charAt(j);
```

```
                if (charOne != charTwo) {
```

```
                    adjList.get(charOne).add(charTwo);
```

```
                    break;
```

```
                }
```

```
            }
```

```
        }
```

```
    }
```

```
    return adjList;
```

```
}
```

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

letters

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

they point to letters that come after them in the alphabet.

Are the edges in the graph directed or undirected?

Select the best answer:

<input checked="" type="radio"/>	Directed
<input type="radio"/>	Undirected

Are the edges in the graph weighted or unweighted?

Select the best answer:

<input type="radio"/>	Weighted
<input checked="" type="radio"/>	Unweighted

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

Explain how you can determine the correct ascending order of each letter in the system using the graph that is built. 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..."

Please limit your response to at most 3 sentences.

given our graph, we can perform a topological sort, since it should be directed & acyclic. This will give us an ordering of nodes s.t. for every edge, its origin comes before its destination. Thus, letters will be put in ascending order, if our edges point from letters that come earlier, to letters that come later in the alphabet.