

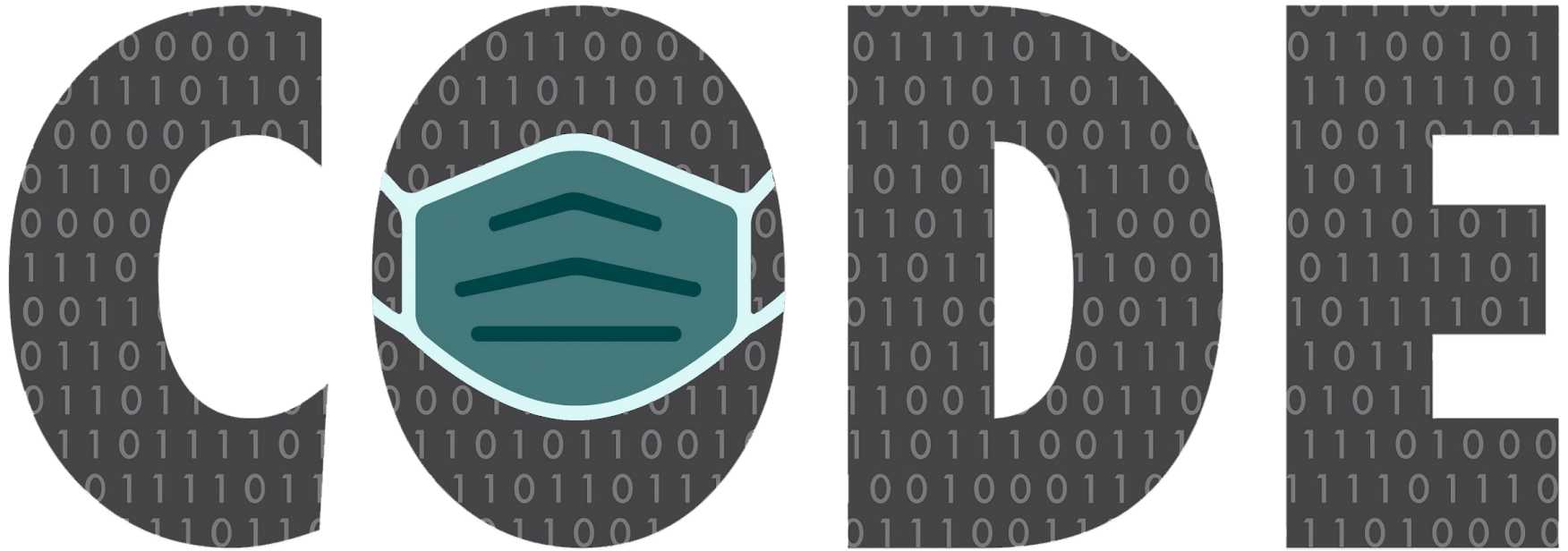
Welcome!

- We'll start in a moment 😊
- Get to know each other!
 - Share your name, where you are, and answer the ice breaker

Ice breakers

- What's your favorite emoji?
- What did you want to be when you grew up?

WOMEN WHO

CODE



WWCode Digital + **Seattle** **VIRTUAL NETWORKING EVENT**

May 3rd, 2022

WWC SEATTLE DIRECTORS



HEIDI TOUSSAINT
Staff UX Researcher @ Google



AKSHITA KORWAR
Software Engineer @ Hulu



RITIKA NEVATIA
Software Engineer @ Apple

VOLUNTEERS



ALEXANDRA RALEVSKI
Data Scientist @ Institute for
Systems Biology



SACHET VIJAY
Software Engineer @ Amazon



NINA KIM
Junior Developer @ AIM
Consulting Group



DEQING LI
Senior Data & Applied Scientist
@ Microsoft



RACHEL SALAZAR
Software Engineer Apprentice
@ Microsoft

OUR MISSION

Inspiring women to
excel in technology
careers.

WOMEN WHO
CODE



OUR VISION

A world where diverse women are better represented as engineers and leaders in technology.

WOMEN WHO
CODE



CODE OF CONDUCT

WWCode is an inclusive community, dedicated to providing an empowering experience for everyone who participates in or supports our community, regardless of gender, gender identity and expression, sexual orientation, ability, physical appearance, body size, race, ethnicity, age, religion, socioeconomic status, caste, creed, political affiliation, or preferred programming language(s).

Our events are intended to inspire women to excel in technology careers, and anyone who is there for this purpose is welcome. We do not tolerate harassment of members in any form. Our [Code of Conduct](#) applies to all WWCode events and online communities.

Read the full version and access our incident report form at

womenwhocode.com/codeofconduct



Agenda

- General Interview Tips
- Strings
- Problem-Solving

Online event best practices

- Mute yourself when you aren't talking
- Turn on your video if you feel comfortable!

General Interview Tips

- Firstly it is important to clarify exactly what the question is, what are the inputs, what are the assumptions, what is the expected output?
- Communicate with the interviewer, if you need time to think by yourself, let them know. Try to clearly communicate your thought process and decisions.
- Reach consensus with interviewer about your solution before implementing.
- In your implementation, focus on readability of code.
- Run through different test cases at the end if you have time.
- Don't forget edge cases!

Strings

- What are Strings?
 - a sequence of characters.
- String questions can be treated like array questions for the most part except we have access to string methods:
 - E.g. substring, equals, toLowerCase etc.

Types of String Problems

- Strings are used as a basis for a large variety of questions solved with many different techniques.
- We'll focus on questions that cover unique aspects of strings:
 - Properties of words/strings: e.g. anagrams, palindromes, lexicographical etc.
 - String manipulation.
 - Prefixes/substrings

Problem 1: Valid Anagram

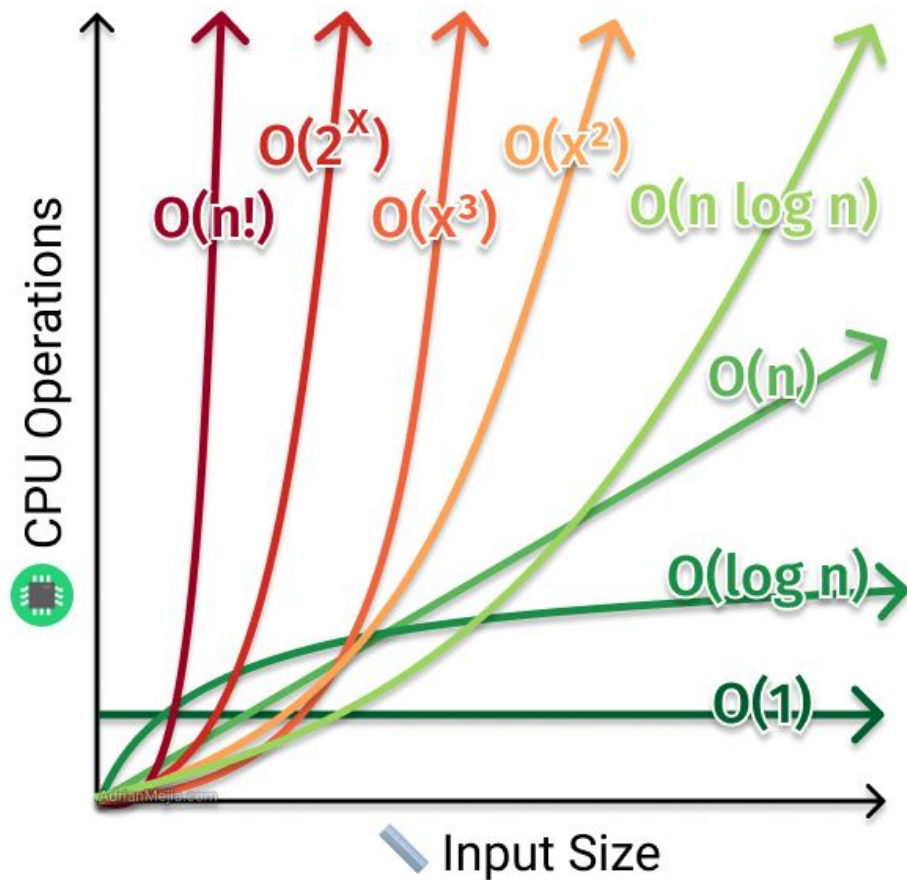
- <https://leetcode.com/problems/valid-anagram/>
- Determine whether one string is an anagram of another.
- If the letters in one word can be rearranged to form another word, then the two words are anagrams.
- Constraints: lowercase letters only.
- Example of anagrams: cat, tac, act.

Problem 1: Valid Anagram

- Naive approach: find all rearrangements of one word, and see if the second word is one of them.
- E.g. for cat, the rearrangements would be : cat, cta, act, atc, tac, tca.
- $N!$ Runtime.
- Can we make any improvements?



Time Complexity



Problem 1: Valid Anagram

- Approach 1:
- Don't need to generate all the rearrangements, rather we can try to generate the second word, by using the letters in the first word. If we can, the words are anagrams.
- For each letter in the first word, see if we can use an unused letter in the second word.
- Need to keep track of used indices because of duplicate letters.
- $O(n^2)$ time complexity, $O(n)$ space complexity.

Problem 1: Valid Anagram

- Approach 2:

- We are essentially just trying to see if the number of times a letter occurs in one word equals the number of times the same letter occurs in the other.
- Rather than for each letter, go through the second word, we can count frequency in first word first, then compare with second word.
- If the each letter in one word occur at the same frequency as in the other, then they are anagrams.
- Use a hashmap to store counts of each unique letter in one word and see if they match the second word.
- Since we are constrained with lower case letters, we can also use an array of size 26.
- $O(n)$ time complexity, $O(1)$ space complexity.

Problem 1: Valid Anagram

- Alternate Solution
- Another way of thinking about the problem: is there a way to identify a group of anagrams, that is all rearrangements of a word. A few different ways of doing so but the most straightforward way: sort the letters in alphabetical order.
- Two anagrams sorted in alphabetical order will be equal.
- $O(n \log n)$ solution.

Problem 1: Valid Anagram

```
public boolean isAnagram(String s, String t) {  
    if (s.length() != t.length()) {  
        return false;  
    }  
    int[] counter = new int[26];  
    for (int i = 0; i < s.length(); i++) {  
        counter[s.charAt(i) - 'a']++;  
        counter[t.charAt(i) - 'a']--;  
    }  
    for (int count : counter) {  
        if (count != 0) {  
            return false;  
        }  
    }  
    return true;  
}
```