Pre Interview Checklist:

1. Start recording Zoom meeting
2. Provide blank Google doc to candidate
3. Clarify coding language of choice (Java/Python)

Problem Statement:

[Paste this into the interview google doc when ready]

Given two strings `s` and `t`, return `true` *if* `t` *is an anagram of* `s`*, and* `false` *otherwise*.

An **Anagram** is a word or phrase formed by rearranging the letters of a different word or phrase, typically using all the original letters exactly once.

For Java:

```
class Solution {
    public boolean isAnagram(String s, String t) {

    }
}
```

For Python:

```
class Solution:
    def isAnagram(self, s: str, t: str) -> bool:
```

Hidden constraints:

[Ideally, the interviewee should ask clarifying questions to discover these. If they start working on a solution before discovering these hidden conditions, stop them and explain the below as well.]

Constraints:

- 1 <= s.length, t.length <= 5 * 104
- s and t consist of lowercase English letters.

Example Test Cases:

[Do not present these to the candidate before they come up with their own examples. You may wish to ask the candidate these tests as a follow-up question to test their code.]

Example 1:

Input: s = "anagram", t = "nagaram"
Output: true
Example 2:

Input: s = "rat", t = "car"
Output: false

Follow-up Questions

[Ask these after the candidate has completed their first solution]

1. Can you analyze the time and space complexity of your solution?
2. What if the inputs contain Unicode characters? How would you adapt your solution to such a case?

Potential Solutions

[Potential solutions candidates come up with. Remember, this list is NOT exhaustive, candidates may come up with a valid solution that is not listed here.]

It's Lab 2. Hopefully, you know how the solutions go!

Rubric

[Fill this up as the interview progresses. You may wish to update these values as new information is revealed.]

Criteria	Score (1-5)	Notes
Problem Understanding	5	He understood the problem perfectly, he even asked follow up questions about the special characters
Solution Efficiency	4	His solution was really good. It would work even if there are special characters. Could have had a better time complexity solution
Code Implementation	5	Did a great job typing out the code. It works - so 5/5
Algorithmic Complexity Analysis	5	Analyzed the time and space complexity correctly
Testing and Edge Cases	5	Works with all the cases, overall great solution
Problem-solving Approach	5	Loved his approach. He was quick to think that there has to be comparisons made and did a great job by coming up with a working solution soon enough.
Clarity and Communication	4	Clarified and communicated their ideas throughout.

Other Thoughts/ Notes

[Use this space to document other important information that you may discover about the candidate during the interview.]

The candidate did a great job. They solved the problem quickly and communicated well throughout. They even came up with their own test case. Their solution could handle any type of special characters as well. While they could have come up with a more efficient solution, they did a great job coming up with a working one in the short time frame.

Final Decision

[Conclude your findings with a verdict. In general, engineers are encouraged to use leaning hire/no hire unless something exceptionally good/ bad happens in the interview. You may wish to add a couple of lines to explain your final decision.]

Strong HIre | **Leaning Hire** | Leaning No HIre | Strong No Hire

Post Interview Checklist:

1. Finish Zoom meeting, submit recording when available
2. Submit interview Google doc that was shared with candidate
3. Submit completed rubric.

Criteria Explanation

Problem Understanding

- Does the candidate understand the problem statement clearly and can articulate it effectively?

Solution Efficiency

- Did the candidate propose an efficient solution that solves the problem within the given constraints?
- Considerations include time complexity, space complexity, and optimization techniques.

Code Implementation

- Does the candidate provide a clear, concise, and correct implementation of the proposed solution?

- Check for proper handling of edge cases, error checks, and variable naming conventions.

Algorithmic Complexity Analysis

- Does the candidate analyze the time and space complexity of their solution?
- Ensure the candidate demonstrates understanding of asymptotic notations and can justify their solution's efficiency.

Testing and Edge Cases

- Does the candidate test their solution with various inputs, including edge cases and boundary conditions?
- Evaluate the comprehensiveness of their test cases and their ability to identify potential issues.

Problem-solving Approach

- Assess the candidate's problem-solving approach, including their ability to break down the problem, devise a strategy, and communicate their thought process effectively.

Clarity and Communication

- Evaluate the clarity of the candidate's explanations and their ability to communicate their thought process and solution effectively.
- Consider their ability to respond to clarifying questions and engage in a constructive dialogue.

Scoring Guidelines

5: Exceptional - Demonstrates thorough understanding, proposes an optimal solution, implements it flawlessly, analyzes complexity accurately, tests comprehensively, and communicates effectively.

4: Proficient - Shows solid understanding, proposes an efficient solution, implements it correctly with minor flaws, analyzes complexity adequately, tests sufficiently, and communicates clearly.

3: Competent - Demonstrates basic understanding, proposes a functional solution with some inefficiencies or errors, implements it with noticeable flaws, analyzes complexity adequately, tests adequately, and communicates adequately.

2: Developing - Shows partial understanding, proposes a suboptimal solution with significant inefficiencies or errors, implements it with major flaws, struggles with complexity analysis and testing, and communicates with difficulty.

1: Inadequate - Demonstrates lack of understanding, proposes an incorrect or incomplete solution, struggles to implement it correctly, lacks understanding of complexity analysis and testing, and communicates poorly.