# Dry Run Table

## Step 1: Count characters from s = "anagram"

|  |  |  |  |
| --- | --- | --- | --- |
| Char | Index (ch - 'a') | freq Before | freq After |
| a | 0 | 0 | 1 |
| n | 13 | 0 | 1 |
| a | 0 | 1 | 2 |
| g | 6 | 0 | 1 |
| r | 17 | 0 | 1 |
| a | 0 | 2 | 3 |
| m | 12 | 0 | 1 |

Final freq[] after counting s:  
[3, 0, 0, 0, 0, 0, 1, 0, ..., 1 (m), ..., 1 (n), ..., 1 (r)]

## Step 2: Subtract characters from t = "nagaram"

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Char | Index | freq Before | freq After | Action |
| n | 13 | 1 | 0 | ok |
| a | 0 | 3 | 2 | ok |
| g | 6 | 1 | 0 | ok |
| a | 0 | 2 | 1 | ok |
| r | 17 | 1 | 0 | ok |
| a | 0 | 1 | 0 | ok |
| m | 12 | 1 | 0 | ok |

All frequencies now 0 → valid anagram ✅

## Example of Invalid Input:

s = "rat"

t = "car"

|  |  |  |
| --- | --- | --- |
| Char | freq[] Change | Result |
| r | +1 in s, -1 in t | 0 |
| a | +1 in s, -1 in t | 0 |
| t | +1 in s | 1 |
| c | -1 in t | ❌ -1 → return false |

# Time & Space Complexity

|  |  |
| --- | --- |
| Metric | Value |
| Time | O(n) |
| Space | O(1) (fixed array size 26) |

# Solution

public class Solution {

    public boolean isAnagram(String s, String t) {

        // If lengths differ, they can't be anagrams

        if (s.length() != t.length()) return false;

        int[] freq = new int[26]; // For 'a' to 'z'

        // Count frequency in s

        for (char ch : s.toCharArray()) {

            freq[ch - 'a']++;

        }

        // Subtract frequency using t

        for (char ch : t.toCharArray()) {

            freq[ch - 'a']--;

            if (freq[ch - 'a'] < 0) return false; // More chars in t than in s

        }

        return true; // All counts balanced

    }

}