

Valid Anagram

Check if two strings are rearrangements of each other

● String

■ EASY

What You Will Learn

- What an anagram is and how frequency maps detect them
- Using a 26-size int array as a frequency counter
- The sort-and-compare approach (simpler but slower)
- Why HashMap works for Unicode/general characters

Intuition First

'listen' and 'silent' — same letters, different order. To check, count each letter in both strings. If every letter has the same count in both, they're anagrams. Like weighing ingredients: same amounts = same recipe, different order.

```
s = 'anagram'    t = 'nagaram'
```

```
Count array (index = letter - 'a'):  
Process s: increment count for each char  
Process t: decrement count for each char
```

```
a: +3 then -3 = 0  ✓  
n: +1 then -1 = 0  ✓  
g: +1 then -1 = 0  ✓  
r: +1 then -1 = 0  ✓  
m: +1 then -1 = 0  ✓
```

```
All counts = 0  →  Valid Anagram! ✓
```

Java Code

```
1  class Solution {  
2      public boolean isAnagram(String s, String t) {  
3          if (s.length() != t.length()) return false;  
4  
5          int[] count = new int[26]; // 26 lowercase letters  
6  
7          for (int i = 0; i < s.length(); i++) {  
8              count[s.charAt(i) - 'a']++; // increment for s  
9              count[t.charAt(i) - 'a']--; // decrement for t  
10         }  
11  
12         // If any count != 0, frequencies don't match
```

```

13     for (int c : count) {
14         if (c != 0) return false;
15     }
16     return true;
17 }
18 }

```

■ Complexity

Type	Value	Meaning
Time	$O(n)$	Single pass through both strings simultaneously
Space	$O(1)$	Fixed size array of 26 — constant space

■ Common Mistakes

■ Mistake

Not checking length first. 'ab' and 'abc' can never be anagrams — early exit saves time.

■ Unicode Follow-up

If input has Unicode chars (not just a-z), use `HashMap<Character,Integer>` instead of `int[26]`. Interviewers often ask this follow-up!

■ Quick Revision

✓	Check length equality first — quick early exit
✓	Use <code>int[26]</code> count array: increment for s, decrement for t
✓	If all counts are 0 at end → valid anagram
✓	For Unicode: use <code>HashMap</code> instead of <code>int[26]</code>
✓	Time $O(n)$, Space $O(1)$