Valid Anagram – Detailed Explanation

Problem Statement

Given two strings s and t, return true if t is an anagram of s, otherwise false. An anagram means both strings contain the same characters with the same frequencies.

Approach 1: Using HashMap

We count the frequency of each character in string s using a HashMap. Then we decrease the count using characters from string t. If any count becomes negative, it is not an anagram.

Example and Dry Run

s = listen, t = silent. After counting characters of s and reducing counts using t, all frequencies become zero. Hence, it is a valid anagram.

C++ Code (HashMap)

```
unordered_map<char,int> mp;
for(char c : s) mp[c]++;
for(char c : t){
    mp[c]--;
    if(mp[c] < 0) return false;
}
return true;
```

Java Code (HashMap)

```
HashMap<Character, Integer> map = new HashMap<>();
for(char c : s.toCharArray())
    map.put(c, map.getOrDefault(c, 0) + 1);

for(char c : t.toCharArray()){
    map.put(c, map.get(c) - 1);
    if(map.get(c) < 0) return false;
}
return true;
```

Python Code (HashMap)

```
mp = {}
for c in s:
    mp[c] = mp.get(c, 0) + 1

for c in t:
    mp[c] -= 1
    if mp[c] < 0:
        return False

return True
```

Time Complexity: O(n), Space Complexity: O(n)

Approach 2: Using Frequency Array

This approach is optimal when strings contain only lowercase letters. We use an array of size 26, incrementing for s and decrementing for t.

Example and Dry Run

s = rat, t = tar. After incrementing and decrementing character frequencies, all array values become zero. Hence, it is a valid anagram.

C++ Code (Frequency Array)

```
int freq[26] = {0};
for(int i = 0; i < s.length(); i++){
    freq[s[i] - 'a']++;
    freq[t[i] - 'a']--;
}
for(int i = 0; i < 26; i++){
    if(freq[i] != 0) return false;
}
return true;
```

Java Code (Frequency Array)

```
int[] freq = new int[26];
for(int i = 0; i < s.length(); i++){
    freq[s.charAt(i) - 'a']++;
    freq[t.charAt(i) - 'a']--;
}
for(int x : freq){
    if(x != 0) return false;
}
return true;
```

Python Code (Frequency Array)

```
freq = [0] * 26
for i in range(len(s)):
    freq[ord(s[i]) - ord('a')] += 1
    freq[ord(t[i]) - ord('a')] -= 1

for x in freq:
    if x != 0:
        return False
return True
```

Time Complexity: O(n), Space Complexity: O(1)

Final Note: Use frequency array for lowercase English letters for best performance. Use HashMap when the character set is large or unknown.