# Leet Code 242 [Valid anagram]

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

s: cat  t: tac ⇒ true

s: sam  t: sat ⇒ False

## Brute Force

s- cat  t: cat  We will check from the first string to each word char from other.

Time: $O(n^2)$

## Optimal

s = cart  t = ratc

s.length == t.length

c a r t

| | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

If we find from s we increase counter but if we find from t we decrease.
If we have different characters our array will have 1 or -1.

Time: $O(n)$
Space: $O(1)$

## Java Solution

```
public boolean isAnagram (String s, String t) {
    if (s.length() != t.length()){
        return False;
    }

    int [] count = new int [26];

    for (int i = 0; i < s.length(); i++){
        count [s.charAt(i) - 'a'] ++;
        count [s.charAt(i) - 'a'] --;
    }

    For (int i = 0; i < 26; i++){
        if (count [i] != 0) {
            return False;
        }
    }

    return true;
}
```

## Kotlin solution

```
Fun isAnagram (s: String, t: String): Boolean {
    if (s.length != t.length) return False

    val count = IntArray (26)

    For (i in s.indices) {
        count [s[i] - 'a'] ++
        count [t[i] - 'a'] --
    }

    For (i in count.indices) {
        if (count [i] != 0) return False
    }

    return true
}
```