Valid anagram

part 2 using hash table - optimal

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

        if len(s) != len(t):
            return False

        count = [0] * 26
        for i in range(len(s)):
            count[ord(s[i]) - ord("a")] += 1
            count[ord(t[i]) - ord("a")] -= 1

        for val in count:
            if val != 0:
                return False
        return True

print(Solution().valid_anagram(s="racecar", t="carrace")
```

! Info : the ord() function returns the number representing the unicode.

- Step 1 :

self -> solution instance
s -> "racecar"
t -> "carrace"

- Step 2 : (count [0] * 26) : (26 letters alphabet)

count -> [0,0,0,0,0,0,0,0,.... 0] 26 times

- Step 3 : (for i in range (len (s)))

i -> 0

- Step 4 : (count[ord (s[i]) - ord ("a")] += 1)

ord(s[i]) -> ord ("r") -> 114
ord ("a") -> 97
count [17] = count[17] +1 =>
count -> $[\overset{0}{0},\overset{1}{0},\overset{2}{0},\overset{3}{0} \ldots, \overset{16}{0},\overset{17}{1},\overset{18}{0},\overset{19}{0} \ldots 0]$

- Step 5 : (count [ord (t[i]) - ord ("a")] -= 1)

ord (t[i]) -> ord ("c") -> 99
count [2] -= 1 => count [0,0,-1,0 ... ... 0]

- Step 6 : i increase by 1

  i -> 1

- Step 7 (count[ord(s[i]) - ord("a")] += 1)

  ord("a") - ord("a") => 0

  count [1, 0, -1, ..., 1, ... 0]

- Step 8 (count [ord(t[i]) - ord("a")] -= 1)

  ord("a") - ord("a") = 0 => -= 1

  count[0, 0, -1, ..., 1, ... 0]

everything is repeating -> -> ->

- Step 9 :

  i -> 2

- Step 10:

  ord("c") - ord("a") = 2 => += 1

  count [0, 0, 0, ..., 1, ... 0]

- Step 11:

  ord("r") - ord("a") = 17 => -= 1

  count [0, 0, 0, 0 ..... 0, ... 0]

eventually we end up with the list "count" having zero's every where

- Step x:
  i -> 6 (last number)

- Step y:
  ord("r") - ord("a") = 17 => += 1
  count at index 17 was -1 => becoming 0

- Step z:
  ord("e") - ord("a") => 101 - 97 => 4 => -= 1
  count at index 4 was 1 => becoming zero

we end up with count [0, 0, 0, 0 --- 0]

- Last step we check if val is different than zero, and we get to return True.