

242. Valid Anagram

- <https://leetcode-cn.com/problems/valid-anagram/description/>

```
1 Given two strings s and t , write a function to determine if t is an
  anagram of s.
2 Example 1:
3 Input: s = "anagram", t = "nagaram"
4 Output: true
5 Example 2:
6 Input: s = "rat", t = "car"
7 Output: false
8 Note:
9 You may assume the string contains only lowercase alphabets.
10 Follow up:
11 What if the inputs contain unicode characters? How would you adapt your
  solution to such case?
```

```
1 给定两个字符串 s 和 t ，编写一个函数来判断 t 是否是 s 的字母异位词。
2 示例 1:
3 输入: s = "anagram", t = "nagaram"
4 输出: true
5 示例 2:
6 输入: s = "rat", t = "car"
7 输出: false
8 说明:
9 你可以假设字符串只包含小写字母。
10 进阶:
11 如果输入字符串包含 unicode 字符怎么办？你能否调整你的解法来应对这种情况？
```

Solutions

1. 两个字符串排序后比对是否一致 $O(n\log n)$

```
1 //sort and compare  $O(n\log n)$ 
2 class Solution {
3 public:
4     bool isAnagram(string s, string t) {
5         if (s.size() != t.size()) { return; }
6         sort(s.begin(), s.end());
7         sort(t.begin(), t.end());
8         return !s.compare(t);
9     }
```

```
10 };
```

2.使用Set去重，再判断字符个数 $O(n^2)$

- 首先利用set去重排序，然后遍历set里的值，比较这个值在两个字符串里的个数
- 遍历set的过程中，如果个数不一致，则返回false

```
1 //using set and compare count of characters  $O(n^2)$ 
2 class Solution {
3 public:
4     bool isAnagram(string s, string t) {
5         if (s.size() != t.size()) { return false; }
6         set<char> temp(s.begin(), s.end());
7         for (auto &it : temp) {
8             if (std::count(s.begin(), s.end(), it) != std::count(t.begin(),
9 , t.end(), it)) {
10                 return false;
11             }
12         }
13         return true;
14     }
15 };
```

3.使用unordered_map，最后判断是否有非0的Item $O(n)$ – 推荐

```
1 //using map and compare count of characters  $O(n)$ 
2 class Solution {
3 public:
4     bool isAnagram(string s, string t) {
5         if (s.size() != t.size()) { return false; }
6         unordered_map<char, int> countMap;
7         for(int i = 0; i < s.size(); i++) {
8             countMap[s[i]]++;
9             countMap[t[i]]--;
10         }
11         return !(std::any_of(countMap.begin(), countMap.end(), []
12 (const auto &it){ return it.second != 0; }));
13     }
14 }
15 };
```

4.使用计数Table $O(\text{string.size}())$ – 推荐

- 建立一个大小为26的计数Table(因为英文小写字母就是26个)，Table里的值全部初始化为0。
- 然后遍历这两个字符串，将位置为index(字符 - 'a')的计数加1或减1。
- 最后判断里面是否有非0的count，或者0的个数是否等于26。

```
1 //Traversal and using count table  $O(n)$ 
2 class Solution {
```

```

3 public:
4     bool isAnagram(string s, string t) {
5         if (s.size() != t.size()) { return false; }
6         int charCount = 26;
7         vector<int> countTable(charCount);
8         for(int i = 0; i < s.size(); i++) {
9             countTable[s[i] - 'a']++;
10            countTable[t[i] - 'a']--;
11        }
12        //return (std::count(countTable.begin(), countTable.end(), 0) == c
13        harCount);
14        return !(std::any_of(countTable.begin(), countTable.end(), []
15        (int x){ return x != 0; }));
16    }
17 };
18
19

```

5.使用计数Table，两次遍历 O(n) – 效率最高 – 推荐

- 建立一个大小为26的计数Table(因为英文小写字母就是26个)，Table里的值全部初始化为0。
- 先遍历第一个字符串，将位置为index(字符 - 'a')的计数加1。
- 然后再遍历第二个字符串，将位置为index(字符 - 'a')的计数减1，同时判断这个位置上的计数如果小于0了，则返回false
- 理论上效率会更高，有可能在第二次遍历时，如果出现计数小于0的，则提前结束遍历。

```

1 //Traversal and using count table O(n)
2 class Solution {
3 public:
4     bool isAnagram(string s, string t) {
5         if (s.size() != t.size()) { return false; }
6         vector<int> countTable(26);
7         for (auto &it : s) { countTable[it - 'a']++; }
8         for (auto &it : t) {
9             if(--countTable[it - 'a'] < 0) { return false; }
10        }
11        return true;
12    }
13 };
14
15
16

```

Code

```

1 #include <iostream>
2 #include <string>
3 #include <vector>
4 #include <set>
5 #include <unordered_map>
6 #include <algorithm>
7
8 using namespace std;
9
10

```

```

13 //sort and compare O(nlogn)
14 // class Solution {
15 // public:
16 //     bool isAnagram(string s, string t) {
17 //         if (s.size() != t.size()) { return; }
18 //         sort(s.begin(), s.end());
19 //         sort(t.begin(), t.end());
20 //         return !s.compare(t);
21 //     }
22 // };
23 //using set and compare count of characters O(n^2)
24 // class Solution {
25 // public:
26 //     bool isAnagram(string s, string t) {
27 //         if (s.size() != t.size()) { return false; }
28 //         set<char> temp(s.begin(), s.end());
29 //         for (auto &it : temp) {
30 //             if (std::count(s.begin(), s.end(), it) != std::count(t.begin(), t.end(), it)) {
31 //                 return false;
32 //             }
33 //         }
34 //         return true;
35 //     }
36 // };
37 //using map and compare count of characters O(n)
38 // class Solution {
39 // public:
40 //     bool isAnagram(string s, string t) {
41 //         if (s.size() != t.size()) { return false; }
42 //         unordered_map<char, int> countMap;
43 //         for(int i = 0; i < s.size(); i++) {
44 //             countMap[s[i]]++;
45 //             countMap[t[i]]--;
46 //         }
47 //         return !(std::any_of(countMap.begin(), countMap.end(), [](const auto &it){ return it.second != 0; }));
48 //     }
49 // };
50 //Traversal and using count table O(n)
51 // class Solution {
52 // public:
53 //     bool isAnagram(string s, string t) {
54 //         if (s.size() != t.size()) { return false; }
55 //         int countSize = 26;
56 //         vector<int> countTable(countSize);
57 //         for(int i = 0; i < s.size(); i++) {
58 //             countTable[s[i] - 'a']++;
59 //             countTable[t[i] - 'a']--;

```

```

74 //      }
75 //      //return (std::count(countTable.begin(), countTable.end(), 0) =
    = countSize);
78 //      return !(std::any_of(countTable.begin(), countTable.end(), []
    (int x){ return x != 0; }));
79 //      }
80 // };
81 //Traversal and using count table O(n)
82 class Solution {
83 public:
84     bool isAnagram(string s, string t) {
85         if (s.size() != t.size()) { return false; }
86         vector<int> countTable(26);
87         for (auto &it : s) { countTable[it - 'a']++; }
88         for (auto &it : t) {
89             if(--countTable[it - 'a'] < 0) { return false; }
90         }
91         return true;
92     }
93 };
94 int main() {
95     // std::string s = "anagram";
96     // std::string t = "nagaram";
97     std::string s = "rat";
98     std::string t = "car";
99     Solution solution;
100     string result = solution.isAnagram(s, t) ? "yes!" : "no!";
101     std::cout << result << std::endl;
102     return 0;
103 }

```