

49. Group Anagrams

- <https://leetcode-cn.com/problems/group-anagrams/>

```
1 Given an array of strings strs, group the anagrams together. You can
  return the answer in any order.
2 An Anagram is a word or phrase formed by rearranging the letters of a
  different word or phrase, typically using all the original letters exactly
  once.
3
4
5
6 Example 1:
7 Input: strs = ["eat","tea","tan","ate","nat","bat"]
8 Output: [["bat"],["nat","tan"],["ate","eat","tea"]]
9
10 Example 2:
11 Input: strs = [""]
12 Output: [[""]]
13
14 Example 3:
15 Input: strs = ["a"]
16 Output: [["a"]]
```

```
1 给定一个字符串数组，将字母异位词组合在一起。字母异位词指字母相同，但排列不同的字符
  串。
2 示例：
3 输入：["eat", "tea", "tan", "ate", "nat", "bat"]
4 输出：
5 [
6   ["ate","eat","tea"],
7   ["nat","tan"],
8   ["bat"]
9 ]
10
11 说明：
12 所有输入均为小写字母。
13 不考虑答案输出的顺序。
```

Solutions

1.使用unordered_map作为临时容器1 O(nklongk) – 推荐

- 建立一个unordered_map<string, vector<string>>, key就是排序后的string, value时一个vector, 用来存放属于同一组异位词的原始字符串。
- 遍历原始字符串数组, 先将每个遍历的字符串排序, 然后根据排序后的字符串作为key在unordered_map里分组并保存原始的字符串。
- 最后遍历unordered_map将数据全部赋给结果vector即可

```

1 //using unordered_map as container O(nklongk)
2 class Solution {
3 public:
4     vector<vector<string>> groupAnagrams(vector<string>& strs) {
5         unordered_map<string, vector<string>> container;
6         vector<vector<string>> res;
7         for (auto &it : strs) {
8             string temp = it;
9             sort(temp.begin(), temp.end());
10            container[temp].push_back(it);
11        }
12        for (auto &it : container) {
13            res.push_back(it.second);
14        }
15        return res;
16    }
17 };

```

2.使用unordered_map作为临时容器2 O(nklongk) – 推荐(最优)

- 原理同第一个方法，但建立一个unordered_map<string, int>, key就是排序后的string, **value为index**, 用来表示在结果vector里的index。
- 遍历原始字符串数组，先将每个遍历的字符串排序，然后根据排序后的字符串作为key在unordered_map里分组并通过的得到的index，直接将原始字符串放入结果vector相应的组里。
- 最后返回结果vector即可。

```

1 //using unordered_map as container O(nklongk)
2 class Solution {
3 public:
4     vector<vector<string>> groupAnagrams(vector<string>& strs) {
5         unordered_map<string, int> container;
6         vector<vector<string>> res;
7         int groupID = 0;
8         for (auto &it : strs) {
9             auto item = it;
10            sort(item.begin(), item.end());
11            if(container.find(item) == container.end()) {
12                container[item] = groupID++;
13                vector<string> group = {it};
14                res.push_back(group);
15            }
16            else {
17                res[container[item]].push_back(it);
18            }
19        }
20        return res;
21    }
22 };

```

3.使用unordered_map作为临时容器3 O(nk)–不够简洁，效率也没有大幅提升

- 原理同第二个方法，只是不要使用sort函数，而是计算每组异位词上字符出现的个数，并将其转换位一个特定字符串，然后将这个字符串作为unordered_map的key来将原始字符串放入结果vector相应的组里。
- 最后返回结果vector即可。

```
1 //using unordered_map as container O(nk)
2 class Solution {
3 public:
4     string countStr(vector<int>& table) {
5         std::stringstream ss;
6         std::copy(table.begin(), table.end(), std::ostream_iterator<int>
7 (ss, "#"));
8         return ss.str();
9     }
10    vector<vector<string>> groupAnagrams(vector<string>& strs) {
11        unordered_map<string, int> container;
12        vector<vector<string>> res;
13        int groupID = 0;
14        for (auto &it : strs) {
15            vector<int> countTable(26);
16            for(auto &itChar : it) {
17                countTable[itChar - 'a']++;
18            }
19
20            string groupKey = countStr(countTable);
21            if(container.find(groupKey) == container.end()) {
22                container[groupKey] = groupID++;
23                vector<string> group = {it};
24                res.push_back(group);
25            }
26            else {
27                res[container[groupKey]].push_back(it);
28            }
29        }
30        return res;
31    }
32};
```

Code

```
1 #include <iostream>
2 #include <sstream>
3 #include <iterator>
4 #include <string>
5 #include <vector>
6 #include <set>
7 #include <unordered_map>
```

```

8  #include <algorithm>
10 using namespace std;
12 //using unordered_map as container O(nklongk)
15 // class Solution {
16 // public:
17 //     vector<vector<string>> groupAnagrams(vector<string>& strs) {
18 //         unordered_map<string, vector<string>> container;
19 //         vector<vector<string>> res;
20 //         for (auto &it : strs) {
23 //             string temp = it;
24 //             sort(temp.begin(), temp.end());
25 //             container[temp].push_back(it);
26 //         }
28 //         for (auto &it : container) {
30 //             res.push_back(it.second);
31 //         }
32 //         return res;
35 //     }
36 // };
38 //using unordered_map as container O(nklongk)
40 class Solution {
41 public:
42     vector<vector<string>> groupAnagrams(vector<string>& strs) {
43         unordered_map<string, int> container;
44         vector<vector<string>> res;
45         int groupID = 0;
48         for (auto &it : strs) {
49             string temp = it;
50             sort(temp.begin(), temp.end());
51             if (container.find(temp) == container.end()) {
52                 vector<string> group = {it};
53                 res.push_back(group);
54                 container[temp] = groupID++;
55             }
56             else {
57                 res[container[temp]].push_back(it);
58             }
59         }
60         return res;
61     }
62 };
64 int main() {
65     vector<string> array = {"eat", "tea", "tan", "ate", "nat", "bat"};
66     //vector<string> array = {"bddddddddd", "bbbbbbbbbbbc"};
67     Solution solution;
68     vector<vector<string>> result = solution.groupAnagrams(array);
69     //print
70     std::cout << "[";
71     for (int i = 0; i < (int)result.size(); i++) {

```

```
76     std::cout << "[";
77     for (int j = 0; j < (int)result[i].size(); j++) {
78         std::cout << result[i][j];
79         if (j != (int)result[i].size() - 1) {
80             std::cout << ",";
81         }
82     }
83     std::cout << "]";
84     if (i != (int)result.size() - 1) {
85         std::cout << ",";
86     }
87 }
88 std::cout << "]" <<std::endl;
89 return 0;
90 }
```