# 49. Group Anagrams

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

```
Given an array of strings strs, group the anagrams together. You can
return the answer in any order.
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.

Example 1:
Input: strs = ["eat","tea","tan","ate","nat","bat"]
Output: [["bat"],["nat","tan"],["ate","eat","tea"]]
Example 2:
Input: strs = [""]
Output: [[""]]
Example 3:
Input: strs = ["a"]
Output: [["a"]]
```

```
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。
示例:
输入: ["eat", "tea", "tan", "ate", "nat", "bat"]
输出:
[
  ["ate","eat","tea"],
  ["nat","tan"],
  ["bat"]
]
说明：
所有输入均为小写字母。
不考虑答案输出的顺序。
```

## Solutions

### 1.使用unorder_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

```
//using unordered_map as contianer O(nklongk)
class Solution {
public:
    vector<vector<string>> groupAnagrams(vector<string>& strs) {
        unordered_map<string, vector<string>> container;
        vector<vector<string>> res;
        for (auto &it : strs) {
            string temp = it;
            sort(temp.begin(), temp.end());
            container[temp].push_back(it);
        }
        for (auto &it : container) {
            res.push_back(it.second);
        }
        return res;
    }
};
```

### 2.使用unorder_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。

```
//using unordered_map as contianer O(nklongk)
class Solution {
public:
    vector<vector<string>> groupAnagrams(vector<string>& strs) {
        unordered_map<string, int> container;
        vector<vector<string>> res;
        int groupID = 0;
        for (auto &it : strs) {
            auto item = it;
            sort(item.begin(), item.end());
            if(container.find(item) == container.end()) {
                container[item] = groupID++;
                vector<string> group = {it};
                res.push_back(group);
            }
            else {
                res[container[item]].push_back(it);
            }
        }
        return res;
    }
};
```

### 3.使用unorder_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。

```
//using unordered_map as contianer O(nk)
class Solution {
public:
    string countStr(vector<int>& table) {
        std::stringstream ss;
        std::copy(table.begin(), table.end(), std::ostream_iterator<int>
(ss, "#"));
        return ss.str();
    }
    vector<vector<string>> groupAnagrams(vector<string>& strs) {
        unordered_map<string, int> container;
        vector<vector<string>> res;
        int groupID = 0;
        for (auto &it : strs) {
            vector<int> countTable(26);
            for(auto &itChar : it) {
                countTable[itChar - 'a']++;
            }

            string groupKey = countStr(countTable);
            if(container.find(groupKey) == container.end()) {
                container[groupKey] = groupID++;
                vector<string> group = {it};
                res.push_back(group);
            }
            else {
                res[container[groupKey]].push_back(it);
            }
        }
        return res;
    }
};
```

## Code

```
#include <iostream>
#include <sstream>
#include <iterator>
#include <string>
#include <vector>
#include <set>
#include <unordered_map>
```

```cpp
#include <algorithm>
using namespace std;
//using unordered_map as contianer O(nklongk)
// class Solution {
// public:
//     vector<vector<string>> groupAnagrams(vector<string>& strs) {
//         unordered_map<string, vector<string>> container;
//         vector<vector<string>> res;
//         for (auto &it : strs) {
//             string temp = it;
//             sort(temp.begin(), temp.end());
//             container[temp].push_back(it);
//         }
//         for (auto &it : container) {
//             res.push_back(it.second);
//         }
//         return res;
//     }
// };
//using unordered_map as contianer O(nklongk)
class Solution {
public:
    vector<vector<string>> groupAnagrams(vector<string>& strs) {
        unordered_map<string, int> container;
        vector<vector<string>> res;
        int groupID = 0;
        for (auto &it : strs) {
            string temp = it;
            sort(temp.begin(), temp.end());
            if (container.find(temp) == container.end()) {
                vector<string> group = {it};
                res.push_back(group);
                container[temp] = groupID++;
            }
            else {
                res[container[temp]].push_back(it);
            }
        }
        return res;
    }
};
int main() {
    vector<string> array = {"eat", "tea", "tan", "ate", "nat", "bat"};
    //vector<string> array = {"bddddddddddd", "bbbbbbbbbbc"};
    Solution solution;
    vector<vector<string>> result = solution.groupAnagrams(array);
 //print
    std::cout << "[";
    for (int i = 0; i < (int)result.size(); i++) {
```

```
        std::cout << "[";
        for (int j = 0; j < (int)result[i].size(); j++) {
            std::cout << result[i][j];
            if (j != (int)result[i].size() - 1) {
                std::cout << ",";
            }
        }
        std::cout << "]";
        if (i != (int)result.size() - 1) {
                std::cout << ",";
        }
    }
    std::cout << "]" <<std::endl;
    return 0;
}
```