Group Anagrams (Medium)

Given an array of strings `strs`, group all *anagrams* together into sublists. You may return the output in any order.

An anagram is a string that contains the exact same characters as another string, but the order of the characters can be different.

Example 1:

```
Input: strs = ["act","pots","tops","cat","stop","hat"]

Output: [["hat"],["act", "cat"],["stop", "pots", "tops"]]
```

Example 2:

```
Input: strs = ["x"]

Output: [["x"]]
```

Example 3:

```
Input: strs = [""]

Output: [[""]]
```

Constraints:

- `1 <= strs.length <= 1000`.
- `0 <= strs[i].length <= 100`
- `strs[i]` is made up of lowercase English letters.

▼ 思路:

作法一：針對每字串sort後以hash比對

T:O(M*NlogN), S:O(M*N)

作法二：26字母陣列作為hash key比對

T:O(M*N), S:O(N) for extre space

作法一

```
class Solution {
public:
    vector<vector<string>> groupAnagrams(vector<string>& strs) {
        unordered_map<string, vector<string>> set;
        for(const auto &s : strs){
            string sort_s = s;
            sort(sort_s.begin(),sort_s.end());
            set[sort_s].push_back(s);
        }
        vector<vector<string>> ans;
        for(const auto &s: set){
            ans.push_back(s.second); //unordered_map value→.second
        }
        return ans;
    }
};
```

作法二：

```
class Solution {
public:
    vector<vector<string>> groupAnagrams(vector<string>& strs) {
        unordered_map<string, vector<string>> set;
        for(const auto &s : strs){
            int key[26] = {0};
            for(const auto c : s){
                key[c-'a'] += 1;
            }
            char kkey[26]={};
            for(int i=0;i<26;i++){
                kkey[i]='a'+key[i]; //notice int→char
```

```
            }
            set[kkey].push_back(s);
        }
        vector<vector<string>> ans;
        for(const auto &s: set){
            ans.push_back(s.second); //unordered_map value→.second
        }
        return ans;
    }
};
```