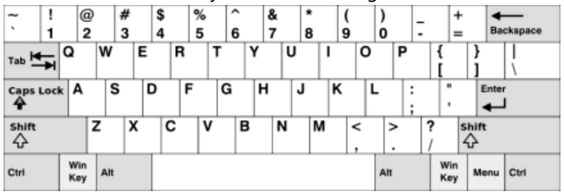
# 500 Keyboard Row

2018年4月13日 13:04

Given a List of words, return the words that can be typed using letters of **alphabet** on only one row's of American keyboard like the image below.



**Example 1:**

**Input:** ["Hello", "Alaska", "Dad", "Peace"]

**Output:** ["Alaska", "Dad"]

**Note:**

1. You may use one character in the keyboard more than once.
2. You may assume the input string will only contain letters of alphabet.

来自 <https://leetcode.com/problems/keyboard-row/description/>

给定一个单词列表，只返回可以使用在键盘同一行的字母打印出来的单词。键盘如下图所示。

**注意:**

1. 你可以重复使用键盘上同一字符。
2. 你可以假设输入的字符串将只包含字母。

## Solution for Python3:

```
class Solution1:
    def findWords(self, words):
        """
        :type words: List[str]
        :rtype: List[str]
        """
        d = [0]*26
        rows = ['qwertyuiop', 'asdfghjkl', 'zxcvbnm']
        for i in range(3):
            for c in rows[i]:
                d[ord(c) - ord('a')] = 1 << i
        res = []
        for word in words:
            r = 7
            for c in word:
                d[ord(c.lower()) - ord('a')]
                r &= d[ord(c.lower()) - ord('a')]
                if not r:
                    break
            if r:
                res.append(word)
        return res


class Solution2:
    def findWords(self, words):
        """
        :type words: List[str]
        :rtype: List[str]
        """
        import re
        return list(filter(re.compile('(?i)
([qwertyuiop]*|[asdfghjkl]*|[zxcvbnm]*)$').match, words))
```

## Solution for C++:

```
class Solution {
```

```
public:
    vector<string> findWords(vector<string>& words) {
        int dict[26];
        vector<string> rows = {"qwertyuiop", "asdfghjkl", "zxcvbnm"} ;
        for (int i = 0; i < rows.size(); i++) {
            for (char c : rows[i]) {
                dict[c - 'a'] = 1 << i;
            }
        }
        vector<string> res;
        for (string word : words) {
            int r = 7;
            for (char c : word) {
                r &= dict[tolower(c) - 'a'];
                if (r == 0)
                    break;
            }
            if (r)
                res.push_back(word);
        }
        return res;
    }
};
```

## Appendix:

**正则表达式分析:**

`return list(filter(re.compile('(?i)([qwertyuiop]*|[asdfghjkl]*|[zxcvbnm]*)$').match, words))`

1) filter(fun, iterable) :过滤掉可迭代对象中使得fun为空或者False的元素，这里就是过滤掉不匹配正则表达式的word。
2) import re 导入正则模块
3) re.compile('') 提前编译正则表达式，为了后面多次用到
4) re.compile('').match(str) 用提前编译好的正则表达式去匹配str
5) 正则表达式部分： `(?i)([qwertyuiop]*|[asdfghjkl]*|[zxcvbnm]*)$`
6) `(?i)`: '?'表示0个或1个 'i'表示忽略大小写
7) (a|b|c)$: '$'表示行的结束，'|'表示以a,b,c任意满足一个，该句表示行的结束只要满足a,b,c任意一个即可
8) a=[qwertyuiop]* 表示a可以有[]中的任意字符组成，'*'表示该字符可以为任意个（包括0个）