⌨️ Keyboard Row Problem – Complete Documentation

📑 Table of Contents

1. Problem Statement

Given an array of strings words, return the words that can be typed using letters from only one row of an American QWERTY keyboard.

- The keyboard is divided into three rows:
    - Row 1: "qwertyuiop"
    - Row 2: "asdfghjkl"
    - Row 3: "zxcvbnm"
- The check is case-insensitive.

⚠️ Constraints:

- 1 <= words.length <= 20
- 1 <= words[i].length <= 100
- Each word consists of only English letters (uppercase/lowercase).

2. Intuition

Since each row can be represented as a set of characters, we can:

- Convert the word to lowercase.
- Check whether all characters of that word fall under any one of the sets.

3. Key Observations

- Using sets allows for fast membership testing.
- A word belongs to a row if its lowercase character set is a subset of the row set.
- Words like "Dad" and "Alaska" have letters only from one row, making them valid.

4. Approach

- Define three sets for the three keyboard rows.
- Loop through each word in the input list:
    - Convert the word to lowercase.
    - Convert the word to a set of unique characters.
    - Check if it is a subset of any row.
- If yes, append the original word to the result list.
- Return the result list.

5. Edge Cases

- Words with mixed cases (e.g., "Dad", "dAD") — should still work due to lowercase conversion.
- Words using characters from multiple rows — must be excluded.
- An empty string (not allowed per constraints).
- All valid words (e.g., ["sdf", "dfg", "jkl"]) — all should be returned.

6. Complexity Analysis

 Time Complexity

- Let n be the number of words and k be the average length of each word.
- Converting each word to lowercase and to a set: O(k)
- Checking subset: O(1) because the row sets are fixed size (≤ 10).
- Total Time: O(n × k)

 Space Complexity

- Three fixed-size sets (constant space): O(1)
- Result list: O(n) in worst case (all words are valid)
- Total Space: O(n)

7. Alternative Approaches

Approach 1: Character-to-Row Mapping

- Map each character to a row index.
- For each word, ensure all characters map to the same row.
- Slightly more complex but also efficient.

Approach 2: Regex (less recommended)

- Use regex patterns for each row.
- Match the word against each pattern.
- Less readable and less efficient for this simple logic.

8. Test Cases

✅ Example 1:

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

✅ Example 2:

Input: ["omk"]
Output: []

✅ Example 3:

Input: ["adsdf", "sfd"]
Output: ["adsdf", "sfd"]

✅ Custom Test Case:

Input: ["QWE", "zxc", "type", "Row"]
Output: ["QWE", "zxc"]

9. Final Thoughts

- This clean implementation problem combines string handling with set operations.
- Set-based logic offers a concise and readable solution.
- The technique of using set.issubset() is a valuable pattern for similar filtering problems.