49. Group Anagrams

Approach #1: Categorize by Sorted String [Accepted]

Intuition

Two strings are anagrams if and only if their sorted strings are equal.

Algorithm

Maintain a map `ans : {String -> List}` where each key K is a sorted string, and each value is the list of strings from the initial input that when sorted, are equal to K.

In Java, we will store the key as a string, eg. `code`. In Python, we will store the key as a hashable tuple, eg. `('c', 'o', 'd', 'e')`.

```
strs = ["are", "bat", "ear", "code", "tab", "era"]
```

Java

```
ans = {"aer": ["are", "ear", "era"],
       "abt": ["bat", "tab"],
       "ecdo": ["code"]}
```

Python

```
ans = {('a', 'e', 'r'): ["are", "ear", "era"],
       ('a', 'b', 't'): ["bat", "tab"],
       ('e', 'c', 'd', 'o'): ["code"]}
```

Java Python Copy

Complexity Analysis

- Time Complexity: $O(NK\log(K))$, where N is the length of `strs`, and K is the maximum length of a string in `strs`. The outer loop has complexity $O(N)$ as we iterate through each string. Then, we sort each string in $O(K \log K)$ time.
- Space Complexity: $O(N * K)$, the total information content stored in `ans`.

Approach #2: Categorize by Count [Accepted]

Intuition

Two strings are anagrams if and only if their character counts (respective number of occurrences of each character) are the same.

Algorithm

We can transform each string s into a character count, count, consisting of 26 non-negative integers representing the number of a's, b's, c's, etc. We use these counts as the basis for our hash map.

In Java, the hashable representation of our count will be a string delimited with '`#`' characters. For example, `abbccc` will be `#1#2#3#0#0#0...#0` where there are 26 entries total. In python, the representation will be a tuple of the counts. For example, `abbccc` will be `(1, 2, 3, 0, 0, ..., 0)`, where again there are 26 entries total.

```
strs = ["aab", "aba", "baa", "abbccc"]
```

Java

```
ans = {"#2#1#0#0#0...#0": ["aab", "aba", "baa"],
       "#1#2#3#0#0#0...#0": ["abbccc"]}
        └──── 26 total entries ────┘
```

Python

```
ans = {(2, 1, 0, 0, ..., 0): ["aab", "aba", "baa"],
       (1, 2, 3, 0, 0, ..., 0): ["abbccc"]}
       └──────── 26 total entries ────────┘
```

Java Python Copy

Complexity Analysis

- Time Complexity: $O(N * K)$, where N is the length of `strs`, and K is the maximum length of a string in `strs`. Counting each string is linear in the size of the string, and we count every string.
- Space Complexity: $O(N * K)$, the total information content stored in `ans`.

Analysis written by: @awice (https://leetcode.com/awice)

Join the conversation

Signed in as **williamfu4leetcode**.

Post a Reply

S (https://discuss.leetcode.com/user/santhosh-k)

santhosh-k commented last week

If the order of returned list is not important, the ruby solution is very trivial.
strs.group_by {|x| x.chars.sort }.values

A (https://discuss.leetcode.com/user/asphodelia)

asphodelia commented last month

Ruby solution with only one line
strs.map{|s| [s, s.split(//).sort.join]}.group_by{|a, b| b}.map{|k, v| v.map{|a, b| a}}

P (https://discuss.leetcode.com/user/piku)

piku commented last month

@sean1993519 (https://discuss.leetcode.com/uid/68589) a string with 21 a's and 6 b's And a string with 2 a's and 16 b's will both be represented as 216 without delimiters. So delimiters are required.

A (https://discuss.leetcode.com/user/amby_leet_code)

amby_leet_code commented last month

sean1993519, you need some delimiter to distinguish between indices