
Problem Statement:

Given a list of strings, group anagrams together. Two strings are anagrams if they have the same characters with the same frequencies, regardless of order.

Solution

```python
def group_anagrams(strs: List[str]) -> List[List[str]]:
    anagram_map = defaultdict(list)

    strs = [word.lower() for word in strs]

    for word in strs:
        char_count = [0] * 26
        for char in word:
            char_count[ord(char) - ord('a')] += 1
        anagram_map[tuple(char_count)].append(word)

    return list(anagram_map.values())
```

Step-by-Step Breakdown

1. **Input:** *["act", "pots", "tops", "cat", "stop", "hat"]*
2. **Character Counting:**
    - For each word in the input list, count the occurrences of each character and represent the count as a list of 26 integers (corresponding to each letter of the alphabet).

Word	Character Count Array
Ref	[a, b, c, d, e, f, g, h, i, j, k, l, m, n, o, p, q, r, s, t, u, v, w, x, y, z]
act	[1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0]
pots	[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0]
tops	[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0]
cat	[1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0]
stop	[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0]
hat	[1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0]

3. **Using the Character Count Array as a Key:**
    - Convert the list of character counts into an immutable tuple (mutable objects, such as arrays, cannot be used as keys).
    - Group all words with the same key together in the dictionary.

Key (Character Count Tuple)	Words
(1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0)	["act", "cat"]
(0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 1, 1, 0, 0, 0, 0, 0)	["pots", "tops", "stop"]
(1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0)	["hat"]

4. **Extract Values**: Take all the grouped anagrams (values in the dictionary) and return them as a list of lists.
5. **Output:** [["act", "cat"], ["pots", "tops", "stop"], ["hat"]]

Aspect	Solution One
Efficiency	O(N * M), N = number of words, M = average word length.
Memory Usage	
Ease of Implementation	Medium
Scalability	High
