

Week 10 q1

Given list=Input: strs = ["eat", "tea", "tan", "ate", "nat", "bat"]

Histogram for "eat"

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
1	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0

Key=10001000000000000001000000

Value="ear"

Histogram for "tea"

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
1	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0

Key=10001000000000000001000000

Value="tea"

Histogram for "tan"

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
1	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	1	0	0	0	0	0	0

Key=1000000000000010000001000000

Value="tan"

Histogram for "ate"

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
1	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0

Key=100010000000000000001000000

Value="ate"

Histogram for "nat"

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
1	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	1	0	0	0	0	0	0

Key=1000000000000010000001000000

Value="nat"

Histogram for "bat"

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
1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0

Key=110000000000000000001000000

Value="bat"

Now grouping all the words with same keys

Final answer is

Output= Output: [["bat"],["nat","tan"],["ate","eat","tea"]]

Chat gpt code

```
def group_anagrams(strs):  
    # Initialize a dictionary to store groups of anagrams  
    anagram_groups = {}  
  
    # Iterate through each string in the input list  
    for word in strs:  
        # Sort the characters of the word and use it as a key for the dictionary  
        sorted_word = ''.join(sorted(word))
```

```
        # If the sorted word is already in the dictionary, append the current
word to its value list
        if sorted_word in anagram_groups:
            anagram_groups[sorted_word].append(word)
        else:
            # If the sorted word is not in the dictionary, create a new list with
the current word as its first element
            anagram_groups[sorted_word] = [word]

    # Convert the dictionary values to a list to get the final result
    result = list(anagram_groups.values())
    return result

# Test data
strs = ["eat", "tea", "tan", "ate", "nat", "bat"]
print(group_anagrams(strs))
```

Test case

The screenshot shows the Visual Studio Code interface with a Python file named `week10q1.py` open. The file contains a function `group_anagrams(strs)` that finds groups of anagrams. The function initializes a dictionary `anagram_groups` and iterates through each string in the input list `strs`. For each word, it sorts its characters and uses the sorted string as a key. If the key is already in the dictionary, the word is appended to the corresponding list. If not, a new list is created. Finally, the dictionary values are converted to a list and returned.

```
1 def group_anagrams(strs):
2     # Initialize a dictionary to store groups of anagrams
3     anagram_groups = {}
4
5     # Iterate through each string in the input list
6     for word in strs:
7         # Sort the characters of the word and use it as a key for the dictionary
8         sorted_word = ''.join(sorted(word))
9
10        # If the sorted word is already in the dictionary, append the current word to its value list
11        if sorted_word in anagram_groups:
12            anagram_groups[sorted_word].append(word)
13        else:
14            # If the sorted word is not in the dictionary, create a new list with the current word as its first element
15            anagram_groups[sorted_word] = [word]
16
17    # Convert the dictionary values to a list to get the final result
18    result = list(anagram_groups.values())
19    return result
20
21 # Test data
22 strs = ["eat", "tea", "tan", "ate", "nat", "bat"]
23 print(group_anagrams(strs))
24 strs = [""]
25 print(group_anagrams(strs))
26 strs = ["a"]
27 print(group_anagrams(strs))
28
```

The output window shows the execution results:

```
[Running] python -u "c:\Users\cheth\OneDrive\Desktop\SFBUS\SEM3\ALGORITHMS\week10q1.py"
[['eat', 'tea', 'ate'], ['tan', 'nat'], ['bat']]
[['']]
[['a']]

[Done] exited with code=0 in 0.167 seconds
```

The status bar at the bottom indicates the file is at line 24, column 12, with 4 spaces, UTF-8 encoding, and CRLF line endings. The Python version is 3.9.13 64-bit (microsoft store). The system clock shows 3:03 PM on 7/25/2023.