

Q7 ==> 49. Group Anagrams - LC - Medium

Anagrams using Hash Table

Example 1:

input = ["eat", "tea", "tan", "ate", "nat", "bat"]

Keys	list of words matching with key
eat	→ [tea] → [ate]
tan	→ [nat]
bat	→ []

output: [["eat", "tea", "ate"], ["tan", "nat"], ["bat"]]

```
from collections import defaultdict
from typing import List
class Solution:
    def groupAnagrams(self,str:List) -> List[List[str]]:
        result = defaultdict(list)
        for s in str:
            count= [0]*26
            for c in s:
                #ascii c= 101 ,a=97,101-97 = 4 , e is at index 4
                count[ord(c)-ord("a")] +=1
                # print("count",count[ord(c)-ord("a")])
            result[tuple(count)].append(s)
            # print(result[tuple(count)])
        return result.values()
```

```
if __name__ == "__main__":
    s = Solution()
    input = ["eat","tea","tan","ate","nat","bat"]
    input2 = [""]
    input3 = ["a"]

    print(s.groupAnagrams(input))
    print(s.groupAnagrams(input2))
    print(s.groupAnagrams(input3))
```

Output:

```
19
20     if __name__ == "__main__":
21         s = Solution()
22         input = ["eat", "tea", "tan", "ate", "nat", "bat"]
23         input2 = [""]
24         input3 = ["a"]
25
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

```
norinaakhtar@Norinas-Air python % /usr/bin/env /usr/bin/python3 /Users/norinaakhtar/Desktop/LeetCode/Python/171_Same_Palindrome.py
dict_values([['eat', 'tea', 'ate'], ['tan', 'nat'], ['bat']])
dict_values([['']])
dict_values([['a']])
```

Time Complexity:

$O(m*n)$ where m is number of strings and n average length of each string.