UNIVERSIDAD NACIONAL DE SAN AGUSTÍN DE AREQUIPA

FACULTAD DE PRODUCCION Y SERVICIOS

ESCUELA PROFESIONAL DE INGENIERÍA DE SISTEMAS

Curso: Laboratorio de Análisis y Diseño de Algoritmos

Aula 11

Presentado por:

Tacca Apaza, Nohelia Estefhania

Docente:

Alex Josue Florez Farfan

Grupo-"B"

Arequipa - Perú

Diciembre 2021

Ejercicio 01 - Group Anagrams

Enlace: https://leetcode.com/problems/group-anagrams/

Given an array of strings strs, group the anagrams together. You can return the answer in any order.

An Anagram is a word or phrase formed by rearranging the letters of a different word or phrase, typically using all the original letters exactly once.

Example 1:

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

Example 2:

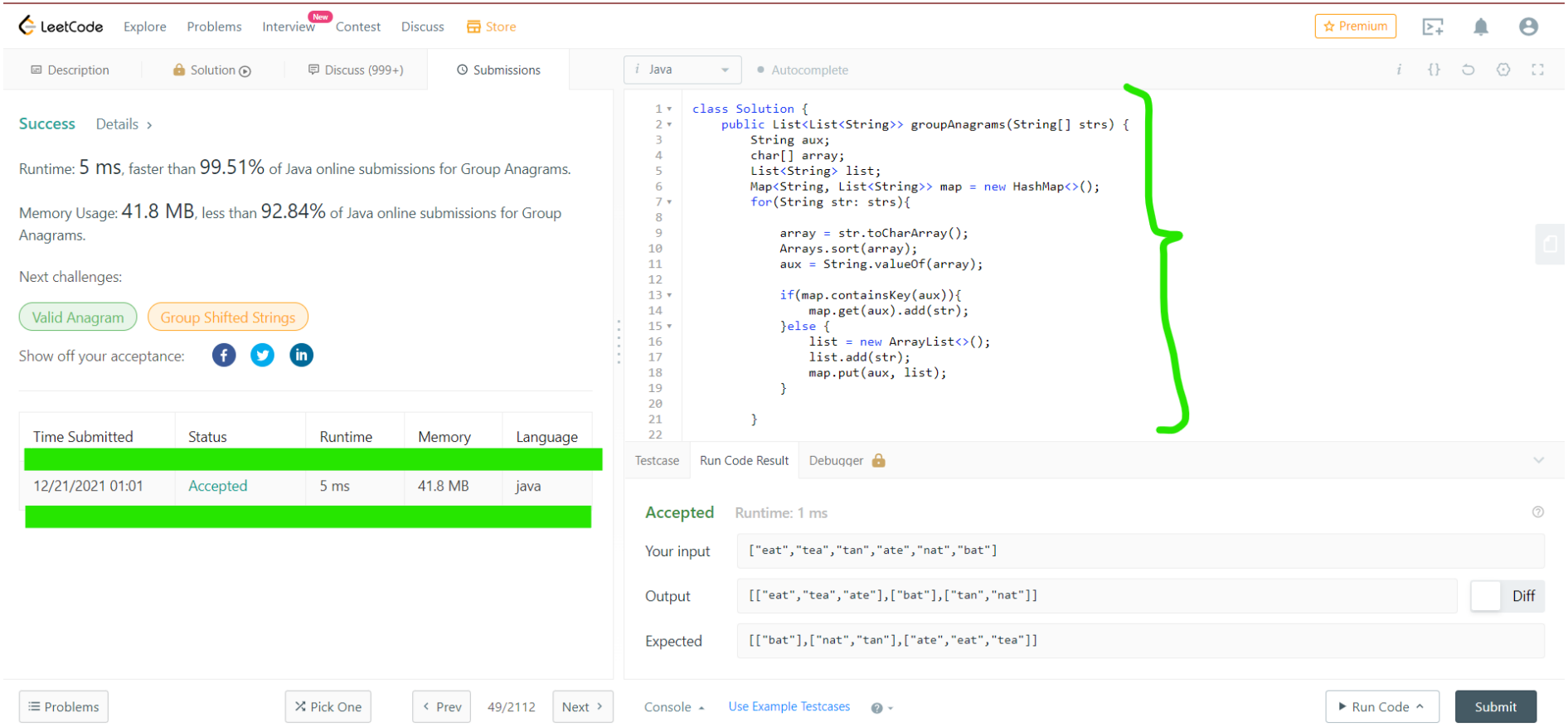
```
Input: strs = [""]
Output: [[""]]
```

Example 3:

```
Input: strs = ["a"]
Output: [["a"]]
```

Constraints:

- `1 <= strs.length <= 10^4`
- `0 <= strs[i].length <= 100`
- `strs[i]` consists of lowercase English letters.


LeetCode
Explore
Problems
Interview
Contest
Discuss
Store
Premium
Description
Solution
Discuss (999+)
Submissions
Java
Autocomplete
Success
Details ›
Runtime: 5 ms, faster than 99.51% of Java online submissions for Group Anagrams.
Memory Usage: 41.8 MB, less than 92.84% of Java online submissions for Group Anagrams.
Next challenges:
Valid Anagram
Group Shifted Strings
Show off your acceptance:
Time Submitted
Status
Runtime
Memory
Language
12/21/2021 01:01
Accepted
5 ms
41.8 MB
java
class Solution {
public List<List<String>> groupAnagrams(String[] strs) {
String aux;
char[] array;
List<String> list;
Map<String, List<String>> map = new HashMap<>();
for(String str: strs){
array = str.toCharArray();
Arrays.sort(array);
aux = String.valueOf(array);
if(map.containsKey(aux)){
map.get(aux).add(str);
}else {
list = new ArrayList<>();
list.add(str);
map.put(aux, list);
}
}
Testcase
Run Code Result
Debugger
Accepted
Runtime: 1 ms
Your input
["eat","tea","tan","ate","nat","bat"]
Output
[["eat","tea","ate"],["bat"],["tan","nat"]]
Diff
Expected
[["bat"],["nat","tan"],["ate","eat","tea"]]
Problems
Pick One
Prev
49/2112
Next
Console
Use Example Testcases
Run Code
Submit

Ejercicio 02 - Dyslectionary

Enlace: https://open.kattis.com/problems/dyslectionary

Have you ever been uncertain how to spell a word? If you ask someone, they will sometimes just tell you to look it up, and that's fine if you know how the word starts. However, it's not much help if you only feel certain about how the word ends. To help give a equal treatment to words that are easy to spell at the end, you have developed the DyslectionaryTM. The Dyslectionary is just like an ordinary dictionary except that it organizes words based on how they end rather than how they start. The first section covers words ending with 'a'. Within this section words are ordered by their second-to-last letter, and so on. If two words have the same suffix, the shorter word is sorted earlier. To help organize your Dyslectionary, you are developing a simple computer program to harvest words from ordinary dictionaries.

Input

Input contains up to 100 word groups. Each word group contains up to 100 words, one word per line, listed in regular dictionary order. All words contain only lowercase English letters a – z. A blank line separates each pair of consecutive groups.

Output

For each group, print out its words in Dyslectionary order, one word per line. Use a blank line to separate consecutive groups. To make words easy to lookup, right-justify them so that they all end at the same column and there is just enough space to fit all the words.

Sample Input 1

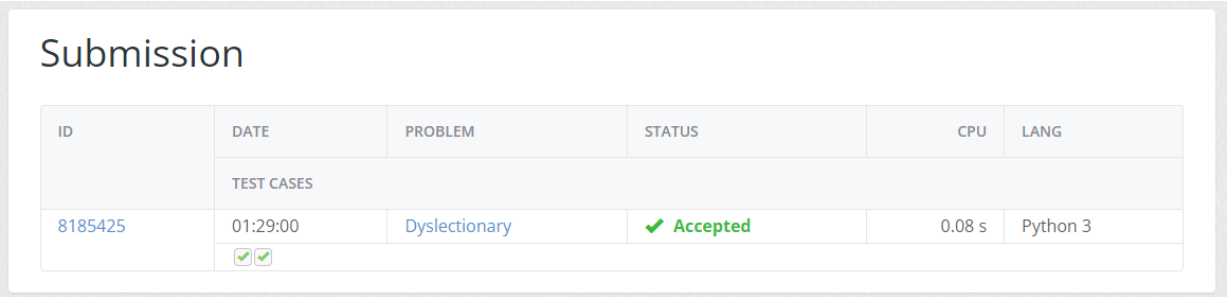
```
apple
banana
grape
kiwi
pear

airplane
bicycle
boat
car
```

Sample Output 1

```
banana
 apple
 grape
  kiwi
  pear

 bicycle
airplane
     car
    boat
```



Submission

ID	DATE	PROBLEM	STATUS	CPU	LANG
	TEST CASES				
8185425	01:29:00	Dyslectionary	✔ Accepted	0.08 s	Python 3
	✔✔				

Submission contains 1 file: download zip archive

FILENAME	FILESIZE	SHA-1 SUM	
dyslectionary.py	622 bytes	90ffab20f8049b27925a85b3add087f8eb535f29	download

Edit and resubmit this submission.

dyslectionary.py

```
1  def sort_and_print(longest, words):
2      for w in sorted(words, key=lambda w: w[::-1]):
3          padding = ' '*(longest-len(w))
4          print(f'{padding}{w}')
5
6  def main():
7      longest,words = -1,[]
8      while True:
9          try:
10             w = input()
11             if not w:
12                 sort_and_print(longest, words)
13                 print()
14                 longest,words = -1,[]
15             else:
16                 longest = max(longest, len(w))
17                 words.append(w)
18         except EOFError:
19             sort_and_print(longest,words)
20             break
21
22 if __name__ == "__main__":
23     main()
```

Ejercicio 04 -Finding Borders

Enlace: https://cses.fi/problemset/task/1732/

A border of a string is a prefix that is also a suffix of the string but not the whole string. For example, the borders of abcababcab are ab and abcab.

Your task is to find all border lengths of a given string.

Input

The only input line has a string of length n consisting of characters a–z.

Output

Print all border lengths of the string in increasing order. Constraints

$1 \leq n \leq 106$

Example

Input:

abcababcab

Output:

2 5

CSES Problem Set

Finding Borders

TASK | SUBMIT | RESULTS | STATISTICS | HACKING

Submission details

Task:	Finding Borders
Sender:	Nohelia
Submission time:	2021-12-21 04:16:34
Language:	C++11
Status:	READY
Result:	ACCEPTED

Test results

test	verdict	time	
#1	ACCEPTED	0.01 s	»
#2	ACCEPTED	0.01 s	»
#3	ACCEPTED	0.01 s	»
#4	ACCEPTED	0.01 s	»
#5	ACCEPTED	0.01 s	»
#6	ACCEPTED	0.12 s	»
#7	ACCEPTED	0.08 s	»
#8	ACCEPTED	0.03 s	»
#9	ACCEPTED	0.02 s	»
#10	ACCEPTED	0.01 s	»
#11	ACCEPTED	0.03 s	»

String Algorithms

- Word Combinations –
- String Matching –
- Finding Borders ✓
- Finding Periods –
- Minimal Rotation –
- Longest Palindrome –
- Required Substring –
- Palindrome Queries –
- ...

Your submissions

2021-12-21 04:16:34 ✓

CSES Problem Set

Finding Borders

TASK | SUBMIT | RESULTS | STATISTICS | HACKING

Number of submissions: 1

‹ 1 ›

time	lang	code time	code size	result	
2021-12-21 04:16:34	C++	0.12 s	501 ch.	✓	»

String Algorithms

- Word Combinations –
- String Matching –
- Finding Borders ✓
- Finding Periods –
- Minimal Rotation –
- Longest Palindrome –
- Required Substring –
- Palindrome Queries –
- ...

Your submissions

2021-12-21 04:16:34 ✓

Code

```cpp
#include <bits/stdc++.h>

using namespace std;
typedef long long ll;
const ll MOD = 1e9+7;
const ll p1 = 31;
const ll p2 = 37;
const int maxN = 1e6+5;

int N;
ll pow1[maxN], pow2[maxN], ph1, ph2, sh1, sh2;
char S[maxN];

int main(){
    scanf("%s", S);
    N = (int) strlen(S);

    pow1[0] = pow2[0] = 1;
    for(int i = 1; i < N; i++){
        pow1[i] = (pow1[i-1] * p1) % MOD;
        pow2[i] = (pow2[i-1] * p2) % MOD;
    }

    for(int i = 0; i < N-1; i++){
        int l = (S[i] - 'a' + 1);
        int r = (S[N-i-1] - 'a' + 1);

        ph1 = (ph1 + l * pow1[i]) % MOD;
        ph2 = (ph2 + l * pow2[i]) % MOD;
        sh1 = (sh1 * p1 + r) % MOD;
        sh2 = (sh2 * p2 + r) % MOD;

        if(ph1 == sh1 && ph2 == sh2)
            printf("%d ", i+1);
    }
}
```