

## Find Anagram Mappings



● Easy

< Prev

> Next

1. You are given two integer arrays(A and B), where B is an anagram of A.
2. B is an anagram of A means B is made by randomizing the order of the elements in A.
3. For every element in A, you have to print the index of that element in B.

Note -> Both arrays(A and B) are of the same length.

map using 'B'

ans

A =	1	9	3	2	4	3	1
	0	1	2	3	4	5	6
B =	2	3	9	1	4	1	3

2 - 0      4 - 4

3 - 1, 6

9 - 2

1 - 3, 5

## Are Two Strings K Anagram

$s_1 =$     a4 b2 c4 d1 e1 f0 g1

$s_2 =$     a2 b4 c2 d1 e1 f1 g2

---

a2 - b2 c2 . . - f1 - g1

ans = 2 + 2 = 4

a-4

b-2

c-4

d-1

e-1

f-1

a-2

b-4

c-2

d-1

e-1

f-1

g-2

## 49. Group Anagrams

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

key: Hashmap <char, int> map

value: ArrayList <strings>

a-1 → eat, tea, ate,  
e-1  
t-1

→ tan, nat      a-1  
n-1  
t-1

a-1 → bat  
b-1  
t-1

Input: strs =

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

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

a-1  
e-1  
t-1 → eat, tea, ate

a-1  
n-1  
t-1 → tan, nat

a-1  
b-1  
t-1 → bat

```
class Pair {
```

```
    int s;
```

```
    int d;
```

```
    Pair() {
```

```
    }
```

41k -> 1

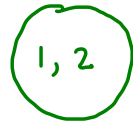
131k -> 1

101k -> 1

121k -> 1

151k -> 1

Hash map < Pair, Integer > map;



41k



91k



101k



121k



151k

```


for(int i=0; i < strs.length;i++) {
    String str = strs[i];
    HashMap<Character,Integer> fmap = new HashMap<>();


    //create fmap
    for(int j=0; j < str.length();j++) {
        char ch = str.charAt(j);
        int nf = fmap.getOrDefault(ch,0) + 1;
        fmap.put(ch,nf);
    }


    if(map.containsKey(fmap) == false) {
        ArrayList<String>list = new ArrayList<>();
        list.add(str);
        map.put(fmap,list);
    }
    else {
        ArrayList<String>list = map.get(fmap);
        list.add(str);
    }
}

```

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


 a-1  
 e-1  
 t-1 → [eat, tea, ate]


 a-1  
 n-1  
 t-1 → [tan, nat]


 a-1  
 b-1  
 t-1 → [bat]

input: baab, abcb, abba, bbaa, caaa  
cbba, abcd

$a_2b_2 \rightarrow$  baab, abba, bbaa

$a_1b_2c_1 \rightarrow$  abcb, cbba

$a_3c_1 \rightarrow$  caaa

$a_1b_1c_1d_1 \rightarrow$  abcd

baab



$a_2b_2$



baab, abcb, abba, bbaa, caaa, cbba

```
public List<List<String>> groupAnagrams(String[] strs) {  
    HashMap<String, ArrayList<String>> map = new HashMap<>();  
  
    for(int i=0; i < strs.length; i++) {  
        String str = strs[i];  
        String key = getKey(str);  
  
        ArrayList<String> list = map.getOrDefault(key, new ArrayList<>());  
        list.add(str);  
        map.put(key, list);  
    }  
  
    List<List<String>> ans = new ArrayList<>();  
    for(String key : map.keySet()) {  
        ans.add(map.get(key));  
    }  
  
    return ans;  
}
```

map

a2b2 → baab, abba, bbaa

a1b2c1 → abcb, cbba

a3c1 → caaa

[[baab, abba, bbaa], [abcb, cbba],  
[caaa]]



## Task Completion

15 6  
2 5 6 7 9 4

$N = 15$  (no. of tasks)  
 $M = 6$  (no. of completed tasks)  
arr = which tasks are complete

S1                      S2                                      S1                      S2                      S1                      S2                      S1

1   ~~2~~                      3                      ~~4~~   ~~5~~   ~~6~~   ~~7~~                      8                      ~~9~~                      10                      11                      12                      13                      14                      15

S1 : 1 , 8 , 11 , 13 , 15

S2 : 3 , 10 , 12 , 4

15 6

2 5 6 7 9 4

2, 5, 6, 7, 9, 4

1   2   3   4   5   6   7   8   9   10   11   12   13   14   15

$d_1 : 1, 8, 11, 13, 15$

$d_2 : 3, 10, 12, 14$