LeetCode\_49\_GroupAnagrams\_同字符词语分组\_Medium

# LeetCode\_49\_GroupAnagrams\_同字符词语分组\_Medium

## 题目介绍

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*Leetcode49:GroupAnagrams\_同字符词语分组\_Medium\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*\*

\* 难度：**Medium**

\* https://leetcode.com/problems/group-anagrams/description/

\* Given an array of strings, group anagrams together.

\* Example:

\* Input: ["eat", "tea", "tan", "ate", "nat", "bat"],

\* Output:

\* [

\* ["ate","eat","tea"],

\* ["nat","tan"],

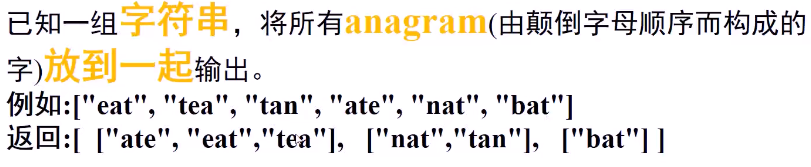
\* ["bat"]

\* ]

\* Note:

\* All inputs will be in lowercase.

\* The order of your output does not matter.



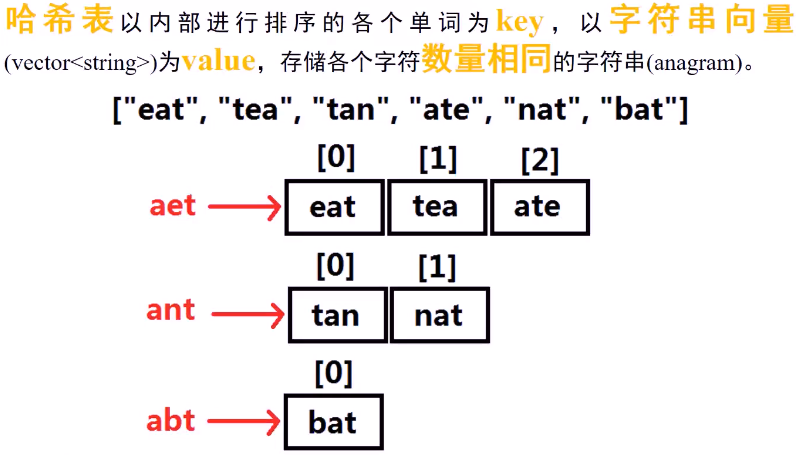
## 思路分析

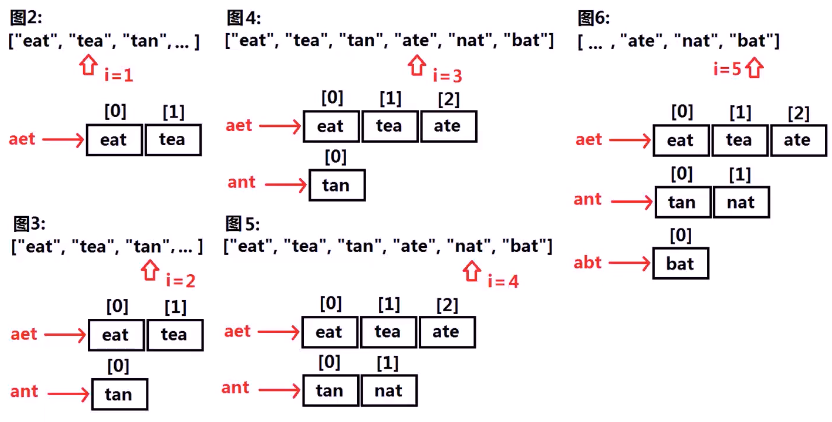
### 思路1：算法2

算法1-第一种映射：获取每个单词时，首先对其进行字典排序，如eat、tea、ate排序后都是aet；以aet做为key，存放到hashMap中，value为ArrayList的索引；

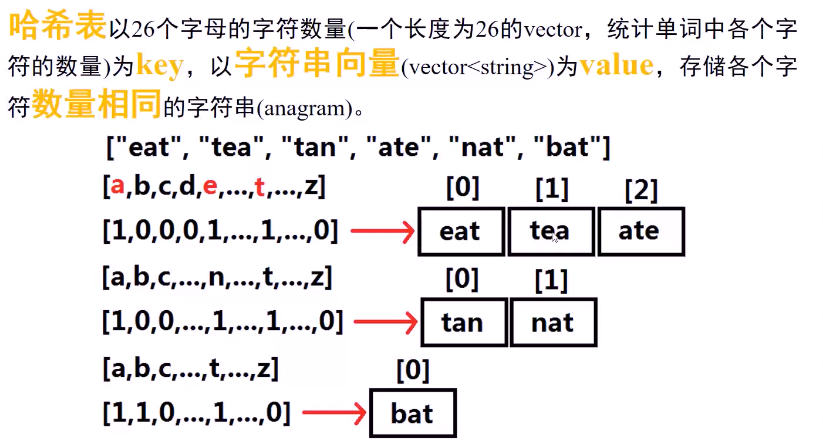
若hashMap存在该key，则获取索引，向List<List>中添加；

若hashMap中不存在该key，则将索引自动加1设置进去。(有个变量记录索引的增长)





### 思路2：算法2





### 两种算法的比较

/\*\*

\* 算法2：利用ArrayList统计单词中字符出现的次数，并作为HashMap的key。

\*/

public List<List<String>> **groupAnagrams2**(String[] strs) {

List<List<String>> result = new ArrayList<List<String>>();

if(strs == null||strs.length == 0) return result;

int listIndex = 0;

**HashMap<ArrayList<Integer>,Integer> hashMap = new HashMap<ArrayList<Integer>,Integer>();**

**ArrayList<Integer> hashTable = new ArrayList<>(26);//利用ArrayList作为HashMap的key**

for (int i = 0;i < 26;i++) hashTable.add(0);//初始容量26不起作用，需要手动初始化

for(String str : strs){

char[] chars = str.toCharArray();

for(char c:chars) hashTable.set(c-'a',hashTable.get(c-'a')+1);//一定是set修改，不是add添加

if(hashMap.containsKey(hashTable)){

result.get(hashMap.get(hashTable)).add(str);

}else{

ArrayList<String> list = new ArrayList<String>();

list.add(str);

result.add(list);

hashMap.put(hashTable,listIndex++);

}

for(int i = 0;i < 26;i++) hashTable.set(i,0);//将hashTable清空为0

}

return result;

}

## Java代码

### 算法1代码

public List<List<String>> **groupAnagrams**(String[] strs) {

List<List<String>> result = new ArrayList<List<String>>();

if(strs == null||strs.length == 0) return result;

int listIndex = 0;//result中索引

HashMap<String,Integer> hashMap = new HashMap<String,Integer>();//构建HashMap

for(String str:strs){

if(hashMap.containsKey(strSort(str))){//已经存在

result.get(hashMap.get(strSort(str))).add(str);

}else{//不存在

ArrayList<String> list = new ArrayList<String>();

list.add(str);

result.add(list);

hashMap.put(strSort(str),listIndex++);

}

}

return result;

}



### 算法2代码

/\*\*

\* 算法2：利用ArrayList统计单词中字符出现的次数，并作为HashMap的key。

\*/

public List<List<String>> groupAnagrams2(String[] strs) {

List<List<String>> result = new ArrayList<List<String>>();

if(strs == null||strs.length == 0) return result;

int listIndex = 0;

HashMap<ArrayList<Integer>,Integer> hashMap = new HashMap<ArrayList<Integer>,Integer>();

ArrayList<Integer> hashTable = new ArrayList<>(26);//利用ArrayList作为HashMap的key

for (int i = 0;i < 26;i++) hashTable.add(0);//初始容量26不起作用，需要手动初始化

for(String str : strs){

char[] chars = str.toCharArray();

for(char c:chars) hashTable.set(c-'a',hashTable.get(c-'a')+1);//一定是set修改，不是add添加

if(hashMap.containsKey(hashTable)){

result.get(hashMap.get(hashTable)).add(str);

}else{

ArrayList<String> list = new ArrayList<String>();

list.add(str);

result.add(list);

hashMap.put(hashTable,listIndex++);

}

for(int i = 0;i < 26;i++) hashTable.set(i,0);//将hashTable清空为0

}

return result;

}

