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. All the characters are in lower case

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

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

Input: strs = ["a"]
Output: [[“a"]]

```
strs = ["are", "bat", "ear", "code", "tab", "era"]
```

Java

```
ans = {"aer": ["are", "ear", "era"],
       "abt": ["bat", "tab"],
       "ecdo": ["code"]}
```

Key in all the Anagram words have the same characters !!! Simply sort the words , have a key with sorted characters.

Ex : => “eat","ate","tea" here when we sort each word you get the same key : “aet”

Input:
String strings= ["eat","tea","tan","ate","nat","bat"]

"aet" = ["eat","ate","tea"]

"ant" = ["tan","nat"]

"bat" = ["bat"]

Apply hashing on sorted Key.
Time Complexity : O(nklog(k))
Space Complexity : O(n)

Do we really need Sorting ?

Lets apply math here , as we know that characters are in lower case and the max key we can make is 26 characters.
Have Character[] of size 26.
In each step before processing input String fill the Character[] with null values.
Gets the index of each character , make sure that current character placed into appropriate index.
int index = s.charAt(i) - ‘a’;
Just travers the Character array form a key where index value is not null.

Time Complexity : O(nk)
Space Complexity : O(n)

String strings= ["abc","cba","bac","dad","add"]

[["abc" , "cba" , "bac"], ["dad","add"]]

"cab"
'c' - 'a' = 98 - 96 = 2

'a' - 'a' = 96 - 96 = 0

'b' - 'a' = 97 - 96 = 1
{ a,b,c,null....null}

"abc"

'a' - 'a' = 96 - 96 = 0

'b' - 'a' = 97 - 96 = 1

'c' - 'a' = 98 - 96 = 2
{ a,b,c,null....null}

Words “cab” & “abc” formed a same array with {a, b,c, Null.... Null}
Just travers the array form a key where index value is not null.
So key is “abc” map the values .
Done => abc = {“cab”,”abc”}

"dad"
'd' - 'a' = 99 - 96 = 3
'a' - 'a' = 96 - 96 = 0
'd' - 'a' = 99 - 96 = 3
{ a, null , null ,d, ...null}

“add"
'a' - 'a' = 96 - 96 = 0
'd' - 'a' = 99 - 96 = 3
'd' - 'a' = 99 - 96 = 3

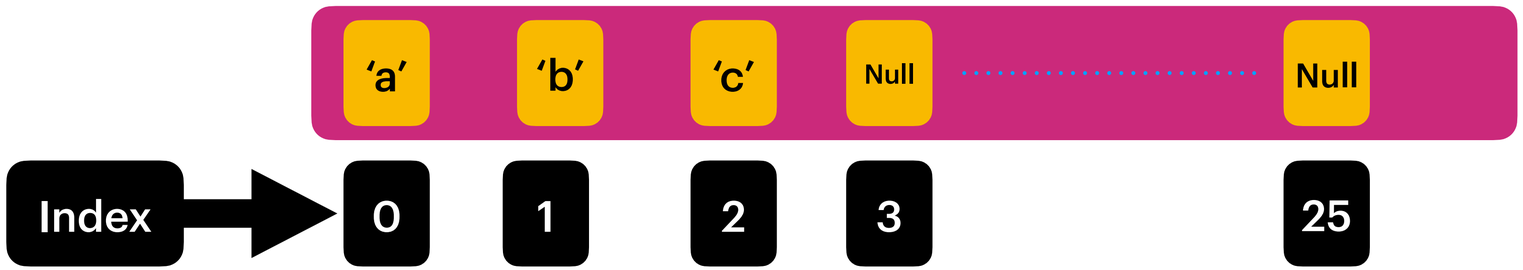
{ a, null , null ,d, ...null}

Words “dad” & “add” formed a same array with {a, null, null, d.... Null}
Just travers the array form a key where index value is not null.
So key is “ad” map the values .
Done => ad = {“dad”,”add”}

String strings= ["abc","cba","bac","dad","add"]

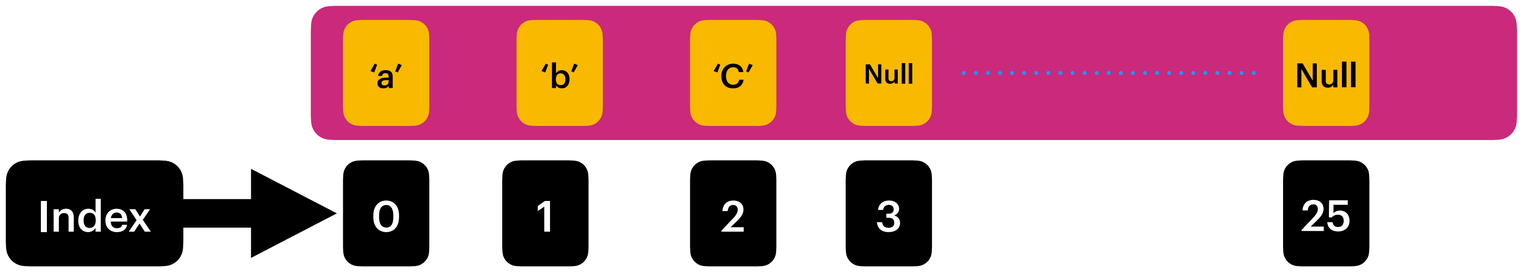
[["abc" , "cba" , "bac"], ["dad","add"]]

In each step before processing
input String fill the Character[] with null values.



"abc"
int index = s.charAt[i] - 'a'

"abc" => ["abc"]



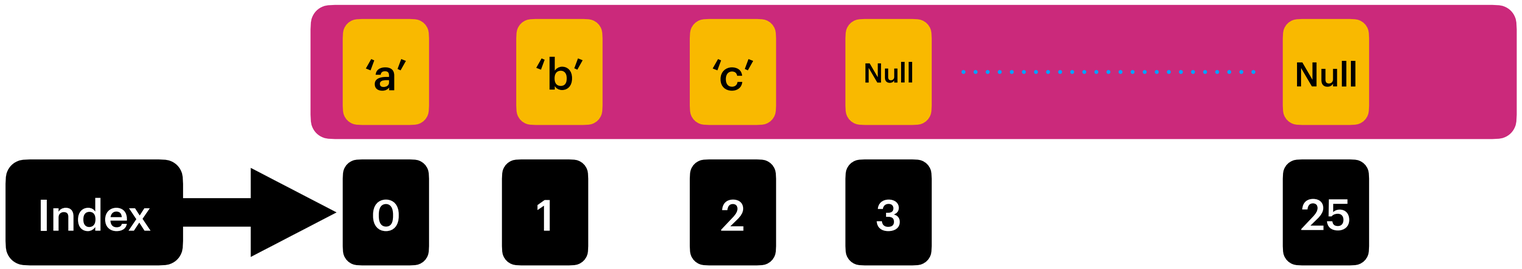
"cba"
int index = s.charAt[i] - 'a'

"abc" => ["abc", "cba"]

String strings= ["abc","cba","bac","dad","add"]

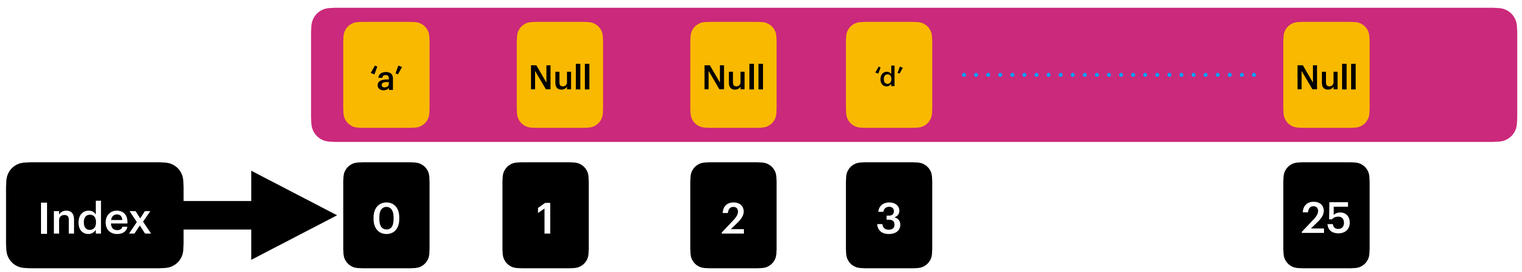
[["abc" , "cba" , "bac"], ["dad","add"]]

In each step before processing
input String fill the Character[] with null values.



"bac"
int index = s.charAt[i] - 'a'

"abc" => ["abc", "cba", "bac"]



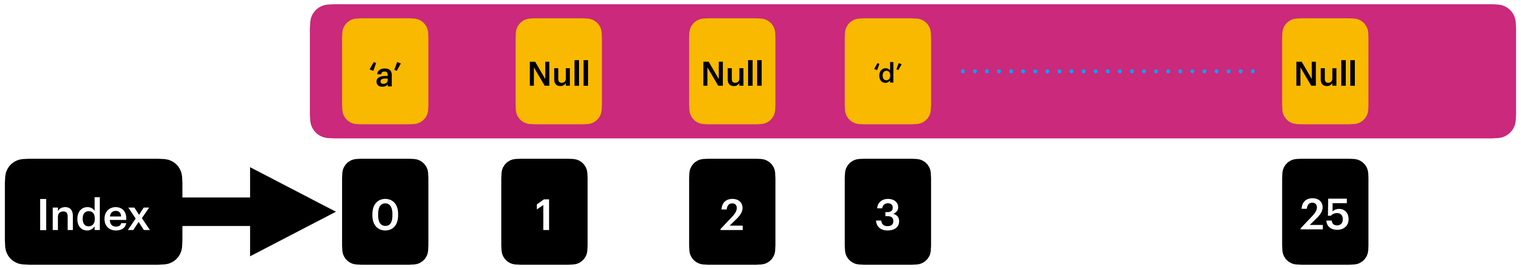
"dad"
int index = s.charAt[i] - 'a'

"ad " => ["dad"]

String strings= ["abc","cba","bac","dad","add"]

[["abc" , "cba" , "bac"], ["dad","add"]]

In each step before processing
input String fill the Character[] with null values.



"add"
int index = s.charAt[i] - 'a'

"ad " => ["dad" , "add"]