Fun with Anagrams

50 points
27 minute(s)
Data Structures Strings Problem Solving Easy

Two strings are anagrams if they are permutations of each other. In other words, both strings have the same size and the same characters. For example, "aaagmnrs" is an anagram of "anagrams". Given an array of strings, remove each string that is an anagram of an earlier string, then return the remaining array in sorted order.

Example

str = ['code', 'doce', 'ecod', 'framer', 'frame']

- "*code*" and "*doce*" are anagrams. Remove "doce" from the array and keep the first occurrence *"code"* in the array.
- *"code"* and "*ecod*" are anagrams. Remove "ecod" from the array and keep the first occurrence *"code"* in the array.
- *"code"* and "*framer*" are not anagrams. Keep both strings in the array.
- "*framer*" and "*frame*" are not anagrams due to the extra *'r'* in *'framer'.* Keep both strings in the array.
- Order the remaining strings in ascending order: *["code","frame","framer"]*.

Function Description

Complete the function *funWithAnagrams* in the editor below.

funWithAnagrams has the following parameters:

string text[n]: an array of strings

Returns:

string[m]: an array of the remaining strings in ascending alphabetical order,.

Constraints

- *0 ≤ n ≤ 1000*
- *0 ≤ m ≤ n*
- *1 ≤* length of *text[i] ≤ 1000*
- Each string *text[i]* is made up of characters in the range ascii[a-z].

Input Format For Custom TestingSample Case 0

Sample Input For Custom Testing

```
STDIN       Function
-----       --------
4       →   n = 4
code    →   text = ["code","aaagmnrs","anagrams","doce"]
```

```
aaagmnrs
anagrams
doce
```

Sample Output

```
aaagmnrs
code
```

Explanation

- "*code"* and "*doce"* are anagrams. Remove "*doce*" and keep the first occurrence *"code"* in the array.
- *"aaagmnrs"* and "*anagrams"* are anagrams. Remove "a*nagrams*" and keep the first occurrence *"aaagmnrs"* in the array.
- Order the remaining strings in ascending order: *["aaagmnrs", "code"].*

Sample Case 1

Sample Input For Custom Testing

```
STDIN       Function
-----       --------
4       →   n = 4
poke    →   text = ["poke","pkoe","okpe","ekop"]
pkoe
okpe
ekop
```

Sample Output

```
poke
```

Explanation

- *"poke*" and *"pkoe"* are anagrams. Remove *"pkoe"* and keep the first occurrence *"poke"* in the array.
- *"poke"* and *"okpe"* are anagrams. Remove *"okpe"* and keep the first occurrence *"poke"* in the array.
- *"poke"* and *"ekop"* are anagrams. Remove *"ekop"* and keep the first occurrence *"poke"* in the array.
- Order the remaining strings in ascending order: *["poke"]*.