Description | Editorial | Solutions | Submissions

2268. Minimum Number of Keypresses Premium

Solved

Medium | Topics | Companies | Hint

You have a keypad with `9` buttons, numbered from `1` to `9`, each mapped to lowercase English letters. You can choose which characters each button is matched to as long as:

- All 26 lowercase English letters are mapped to.
- Each character is mapped to by **exactly** `1` button.
- Each button maps to **at most** `3` characters.

To type the first character matched to a button, you press the button once. To type the second character, you press the button twice, and so on.

Given a string `s`, return *the* ***minimum*** *number of keypresses needed to type* `s` *using your keypad.*

Note that the characters mapped to by each button, and the order they are mapped in cannot be changed.

Example 1:


```
Input: s = "apple"
Output: 5
Explanation: One optimal way to setup your keypad is shown above.
Type 'a' by pressing button 1 once.
Type 'p' by pressing button 6 once.
Type 'p' by pressing button 6 once.
Type 'l' by pressing button 5 once.
Type 'e' by pressing button 3 once.
A total of 5 button presses are needed, so return 5.
```

Example 2:


```
Input: s = "abcdefghijkl"
Output: 15
Explanation: One optimal way to setup your keypad is shown above.
The letters 'a' to 'i' can each be typed by pressing a button once.
Type 'j' by pressing button 1 twice.
Type 'k' by pressing button 2 twice.
Type 'l' by pressing button 3 twice.
A total of 15 button presses are needed, so return 15.
```

Constraints:

- `1 <= s.length <= 10^5`
- `s` consists of lowercase English letters.

Seen this question in a real interview before? 1/5

Yes | No

Accepted 22.4K | Submissions 31.5K | Acceptance Rate 71.1%

Topics

Companies

Hint 1

Hint 2

Discussion (9)

223 | 9

Code

Python3 | Auto

```python
class Solution:
    def minimumKeypresses(self, s: str) -> int:

        counter = 1
        res = 0
        dictx = {}

        for each in s:
            if each not in dictx:
                dictx[each] = 1
            else:
                dictx[each] += 1

        dictx = sorted(dictx.items(), key = lambda x: x[1], reverse= True)

        for i in range(0, len(dictx)):
            if i < 9:
                res += dictx[i][1]
            elif i < 18:
                res += (dictx[i][1] * 2)
            else:
                res += (dictx[i][1] * 3)

        return res
```

Saved | Ln 16, Col 39

Testcase | Test Result

Accepted Runtime: 61 ms

Case 1 | Case 2 | Case 3

Input

s =
"apple"

Output