< Back | Python Solution with Explanation (44ms, 84%)

AnthonyChao ★ 43 April 27, 2019 7:32 AM 634 VIEWS

24

We can solve this problem by mapping each `string` in `strings` to a `key` in a `hashmap`. We then return `hashmap.values()`.

```
{
     (1, 1): ['abc', 'bcd', 'xyz'],
  (2, 2, 1): ['acef'],
     (25,): ['az', 'ba'],
        (): ['a', 'z']
}
```

- The key can be represented as a tuple of the "differences" between adjacent characters. Characters map to integers (e.g. `ord('a') = 97`). For example, 'abc' maps to `(1,1)` because `ord('b') - ord('a') = 1` and `ord('c') - ord('b') = 1`
- We need to watch out for the "wraparound" case - for example, `'az'` and `'ba'` should map to the same "shift group" as `a + 1 = b` and `z + 1 = a`. Given the above point, the respective tuples would be `(25,)` (122 - 97) and `(-1,)` (79 - 80) and `az` and `ba` would map to different groups. This is incorrect.
- To account for this case, we add 26 to the difference between letters (smallest difference possible is -25, `za`) and mod by 26. So, (**26** + 122 - 97) **% 26** and (**26** + 79 - 80) **% 26** both equal `(25,)`

```
def groupStrings(self, strings: List[str]) -> List[List[str]]:
    hashmap = {}
    for s in strings:
        key = ()
        for i in range(len(s) - 1):
            circular_difference = 26 + ord(s[i+1]) - ord(s[i])
            key += (circular_difference % 26,)
        hashmap[key] = hashmap.get(key, []) + [s]
    return list(hashmap.values())
```

- Time complexity would be `O(ab)` where `a` is the total number of strings and `b` is the length of the longest string in strings.
- Space complexity would be `O(n)`, as the most space we would use is the space required for `strings` and the keys of our hashmap.

23 / 23 test cases passed.
Status: Accepted
Runtime: 44 ms
Memory Usage: 13.1 MB

hashmap python dict tuple easy-to-understand python 3