Recycled Number

locked

by abhi98khandelwal

Editorial by pkacprzak

In this problem, for a given array A consisting of n non-negative integers not exceeding 10^6, the goal is to find the number of **unique recycled pairs** that can be formed from elements of A.

A **recycled pair** is a pair of integers (x, y), where $x < y$, x and y have the same number of digits, and y can be transformed to x by moving some consecutive sequence of digits from the back of y to its front without changing their relative order.

For example, the pair $(x, y) = (1234, 3412)$ is a recycled pair, because we have $x < y$, both x and y have the same number of digits and we can transform y to x by moving the last 2 digits of y to its front. Notice that all 3 requirements have to be fulfilled, which means that none of the pairs $(312, 123), (12, 120), (56, 57)$ is a recycled pair.

Before solving the problem, we make a few observations.

First, we are asked to find the number of *unique* recycled pairs, which means that we can drop all duplicates from A.

Now, let's treat every number in A as a string.

Next, let's define $\mathbf{shift}(s, i)$, where s is a string and i is a non-negative integer, as the string which is the result of the operation of moving i consecutive characters from the back of s to its front without changing their relative order. For example, $\mathbf{shift}(123, 1) = 312$, $\mathbf{shift}(123, 0) = 123$, $\mathbf{shift}(123, 2) = 231$.

Next, let $c[s]$ be the *set* of all different strings that can be obtained by applying shift operation to s. Notice that this set is finite, because there are at most $|s| - 1$ different strings in it, where $|s|$ is the length of s.

Now, since all input integers are not greater than 10^6, we know that each string $s \in A$ has the length of at most 7, which means that the size of $c[s]$ is at most 6. This is the crucial observation, which leads to the following solution:

Let $f[w]$ be the number of strings $s \in A$ which has $w \in c[s]$. Notice that a *hash map* can be used to store the values of f efficiently. Now, we are ready to compute the number of recycled pairs in A counting each such pair twice, and the final result will be this result divided by 2.

Let's initialize the result to 0 and iterate over all strings $s \in A$. Remember that all the strings are unique because we dropped all duplicates at the beginning.

For each $s \in A$, we add to the result $f[s] - 1$, which is the number of strings in A, which can be shifted to s, and we subtract 1 because s can be shifted to itself and we don't want to count it.

Notice that by doing that we count each of the recycled pairs (x, y) twice: when we examine x and when we examine y, because if x can be shifted to y, then y can be also shifted to x, so we just divide the result by 2.

The total time complexity of this method is $O(n \cdot L)$ if a hash-map is used for storing values of f, where L is the length of strings in the input, and in our case $L \leq 7$.

Statistics

Difficulty: Medium
Time Complexity: $O(n)$
Required Knowledge: strings, hash map
Publish Date: May 11 2017

For an exact implementation of this method, please refer to my solution attached below (tester's solution).

Problem Setter's code :

```
from bisect import bisect_left

def rotate(x):
    return x[-1]+x[:-1]

l = int(raw_input())
arr = map(int,raw_input().split())
s = set([])
arr.sort()
for i in arr:
    temp = str(i)
    for j in xrange(len(str(i))-1):
        temp = rotate(str(temp))
        try:
            if temp[0]!="0" and arr[bisect_left(arr,int(temp))]==int(temp) and int(temp)!
=i:
                s.add(((min(i,int(temp))),max(i,int(temp))))
        except:
            pass
print s
```

 Tested by pkacprzak

Problem Tester's code :

```
N = 10**5
V = 10**6

n = int(raw_input())
assert (1 <= n <= N)

a = raw_input().split()
a = list(set(a))


for s in a:
    assert (0 <= int(s) <= V)

for s in a[1:]:
    assert (len(s) == len(s[0]))

a = filter(lambda s: len(set(list(s))) > 1, a)

mem = {}
for s in a:
    rotations = set()
    for i in xrange(1, len(s)):
        rotated = s[i:] + s[:i]
        rotations.add(rotated)
    for rotated in rotations:
        if rotated not in mem:
            mem[rotated] = 0
        mem[rotated] += 1

res = 0

for s in a:
    res += mem[s]-1

print res/2
```