

[Practice](#)[Compete](#)[Jobs](#)[Rank](#)[Leaderboard](#)[Dashboard](#) > [Algorithms](#) > [Strings](#) > Two Two

Badge Progress



Points: 4727.88 Rank: 491

# Two Two



by amititkgp

Problem

Submissions

Leaderboard

Discussions

Editorial

Russian

Prof. Twotwo as the name suggests is very fond powers of 2. Moreover he also has special affinity to number 800. He is known for carrying quirky experiments on powers of 2.

One day he played a game in his class. He brought some number plates on each of which a digit from 0 to 9 is written. He made students stand in a row and gave a number plate to each of the student. Now turn by turn, he called for some students who are standing continuously in the row say from index  $i$  to index  $j$  ( $i \leq j$ ) and asked them to find their strength.

The strength of the group of students from  $i$  to  $j$  is defined as:

```
strength(i , j)
{
    if a[i] = 0
        return 0; //If first child has value 0 in the group, strength of group is zero
    value = 0;
    for k from i to j
        value = value*10 + a[k]
    return value;
}
```

Prof called for all possible combinations of  $i$  and  $j$  and noted down the strength of each group. Now being interested in powers of 2, he wants to find out how many strengths are powers of two. Now its your responsibility to get the answer for prof.

## Input Format

First line contains number of test cases  $T$

Next  $T$  line contains the numbers of number plates the students were having when standing in the row in the form of a string  $A$ .

## Constraints

$1 \leq T \leq 100$   
 $1 \leq \text{len}(A) \leq 10^5$   
 $0 \leq A[i] \leq 9$

## Output Format

Output the total number of strengths of the form  $2^x$  such that  $0 \leq x \leq 800$ .

## Sample Input 0

```
5
2222222
24256
65536
023223
33579
```

## Sample Output 0

7  
4  
1  
4  
0

### Explanation 0

In following explanations group i-j is group of student from index i to index j (1-based indexing)

- In first case only 2 is of form power of two. It is present seven times for groups 1-1,2-2,3-3,4-4,5-5,6-6,7-7
- In second case 2,4 and 256 are of required form. 2 is strength of group 1-1 and 3-3, 4 is strength of group 2-2 and 256 is strength of group 3-5.
- In third case 65536 is only number in required form. It is strength of group 1-5
- In fourth case 2 and 32 are of forms power of 2. Group 1-2 has values 0,2 but its strength is 0, as first value is 0.
- In fifth case, None of the group has strength of required form.

f t in

Submissions: 1734

Max Score: 150

Difficulty: Advanced

Rate This Challenge:

☆☆☆☆☆

[More](#)

Current Buffer (saved locally, editable)  

Java 8



```

1 import java.io.*;
2 import java.util.*;
3 import java.math.*;
4
5 class Trie{
6
7     int ind;
8
9     private class TrieNode{
10
11         Map<Character,TrieNode> children;
12         boolean isEndOfWord;
13
14         public TrieNode(){
15             children = new HashMap<>();
16             isEndOfWord = false;
17         }
18     }
19
20     private final TrieNode root;
21
22     public Trie(){
23         root = new TrieNode();
24     }
25
26     public void insert(String word){
27         TrieNode current = root;
28
29         for(int i = 0 ; i < word.length() ; i++){
30
31             char ch = word.charAt(i);
32             TrieNode node = current.children.get(ch);
33
34             if(node == null){
35                 node = new TrieNode();
36                 current.children.put(ch, node);
37             }
38             current = node;

```

```

39     }
40     current.isEndOfWord = true;
41 }
42
43 public int search(String word, int index){
44     TrieNode current = root;
45     int prevMatch = 0;
46
47     int output = 0;
48
49     for(int i = 0 ; i < word.length() ; i++){
50
51         char ch = word.charAt(i);
52
53         TrieNode node = current.children.get(ch);
54
55         if(node == null){
56             break;
57         }
58
59         if(node.isEndOfWord){
60             if(index != ind || i == word.length() - 1)
61                 output++;
62         }
63         current = node;
64     }
65
66     ind = index;
67     return output;
68 }
69
70 }
71
72
73
74 public class Solution {
75
76     static Set<String> powerOfTwo = new HashSet<String>();
77     static int largest;
78     public static void main(String[] args) throws IOException {
79
80         largest = computeValues();
81
82         Trie trie = new Trie();
83
84         Iterator<String> iterator = powerOfTwo.iterator();
85
86         while(iterator.hasNext()){
87             String str = iterator.next();
88             trie.insert(str);
89         }
90
91         BufferedReader br = new BufferedReader(new InputStreamReader(System.in));
92         int tst = Integer.parseInt(br.readLine());
93
94         for(int i = 0 ; i < tst ; i++){
95             System.out.println(solve(br.readLine(), 0, 0, trie));
96         }
97     }
98
99     public static int solve(String str,int firstIndex, int lastIndex, Trie trie){
100
101         int output = 0;
102
103         if(firstIndex >= str.length()){
104             return 0;
105         }
106
107         if(lastIndex >= str.length()){
108             if(firstIndex != 0){
109                 output = trie.search(str.substring(firstIndex,lastIndex),firstIndex);
110             }
111

```

```
112         output += solve(str,firstIndex+1,lastIndex,trie);
113         return output;
114     }
115
116     output = trie.search(str.substring(firstIndex,lastIndex+1),firstIndex);
117
118     if(lastIndex < 241){
119         output += solve(str,0,lastIndex+1,trie);
120     }
121     else{
122         output += solve(str,lastIndex-240,lastIndex+1,trie);
123     }
124
125     return output;
126 }
127
128 public static int computeValues(){
129
130     BigInteger bigInteger = new BigInteger("1");
131     powerOfTwo.add(bigInteger.toString());
132
133
134     for(int i = 1 ; i <= 800 ; i++){
135
136         bigInteger = bigInteger.shiftLeft(1);
137         powerOfTwo.add(bigInteger.toString());
138
139     }
140
141     return bigInteger.toString().length();
142
143 }
144
145 }
```

Line: 1 Col: 1

 Upload Code as File☐ Test against custom input

Run Code

Submit Code

Copyright © 2017 HackerRank. All Rights Reserved

Join us on IRC at [#hackerrank](#) on freenode for hugs or bugs.[Contest Calendar](#) | [Interview Prep](#) | [Blog](#) | [Scoring](#) | [Environment](#) | [FAQ](#) | [About Us](#) | [Support](#) | [Careers](#) | [Terms Of Service](#) | [Privacy Policy](#) | [Request a Feature](#)