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<=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

$1 \le T \le 100$
$1 \le len(A) \le 10^5$
$0 \le A[i] \le 9$

Output Format

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

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.


```java
import java.io.*;
import java.util.*;
import java.math.*;

class Trie{

    int ind;

    private class TrieNode{

        Map<Character,TrieNode> children;
        boolean isEndOfWord;

        public TrieNode(){
            children =  new HashMap<>();
            isEndOfWord = false;
        }
    }

    private final TrieNode root;

    public Trie(){
        root = new TrieNode();
    }

    public void insert(String word){
        TrieNode current = root;

        for(int i = 0 ; i < word.length() ; i++){

            char ch = word.charAt(i);
            TrieNode node = current.children.get(ch);

            if(node == null){
                node = new TrieNode();
                current.children.put(ch, node);
            }
            current = node;
```

```
        }
        current.isEndOfWord = true;
    }

    public int search(String word, int index){

        TrieNode current = root;
        int prevMatch = 0;

        int output = 0;

        for(int i = 0 ; i < word.length() ; i++){

            char ch = word.charAt(i);

            TrieNode node = current.children.get(ch);

            if(node == null){
                break;
            }

            if(node.isEndOfWord){
                if(index != ind || i == word.length() - 1)
                output++;
            }
            current = node;
        }

        ind = index;
        return output;
    }

}


public class Solution {

    static Set<String> powerOfTwo =  new HashSet<String>();
    static int largest;
    public static void main(String[] args) throws IOException {

        largest = computeValues();

        Trie trie = new Trie();

        Iterator<String> iterator = powerOfTwo.iterator();

        while(iterator.hasNext()){
            String str = iterator.next();
            trie.insert(str);
        }

        BufferedReader br = new BufferedReader(new InputStreamReader(System.in));
        int tst = Integer.parseInt(br.readLine());

        for(int i = 0 ; i < tst ; i++){
            System.out.println(solve(br.readLine(), 0, 0, trie));
        }
    }

    public static int solve(String str,int firstIndex, int lastIndex, Trie trie){

        int output = 0;

        if(firstIndex >= str.length()){
            return 0;
        }


        if(lastIndex >= str.length()){
            if(firstIndex != 0){
                output = trie.search(str.substring(firstIndex,lastIndex),firstIndex);
            }
```

```
            output += solve(str,firstIndex+1,lastIndex,trie);
            return output;
        }

        output = trie.search(str.substring(firstIndex,lastIndex+1),firstIndex);

        if(lastIndex < 241){
            output += solve(str,0,lastIndex+1,trie);
        }
        else{
            output += solve(str,lastIndex-240,lastIndex+1,trie);
        }

        return output;
    }

    public static int computeValues(){

        BigInteger bigInteger = new BigInteger("1");
        powerOfTwo.add(bigInteger.toString());


        for(int i = 1 ; i <= 800 ; i++){

            bigInteger = bigInteger.shiftLeft(1);
            powerOfTwo.add(bigInteger.toString());

        }

        return bigInteger.toString().length();

    }

}
```

Line: 1 Col: 1

Upload Code as File Test against custom input Run Code Submit Code