OJ 471

Magic Numbers

Prof. Edson Alves – UnB/FGA

Problema

Write a program that finds and displays all pairs of integers s_1 and s_2 such that:

1. neither s_1 nor s_2 have any digits repeated; and
2. $s_1/s_2 = N$, where N is a given integer.

Entrada e saída

Input

The input file consist a integer at the beginning indicating the number of test case followed by a blank line. Each test case consists of one line of input containing N.

Two input are separated by a blank line.

Output

For each input the output consists of a sequence of zero or more lines each containing '$s_1/s_2 = N$', where s_1, s_2 and N are the integers described above. When there are two or more solutions, sort them by increasing numerator values.

Two consecutive output set will separated by a blank line.

Exemplo de entradas e saídas

Sample Input

```
1

1234567890
```

Sample Output

```
1234567890 / 1 = 1234567890
2469135780 / 2 = 1234567890
4938271560 / 4 = 1234567890
6172839450 / 5 = 1234567890
8641975230 / 7 = 1234567890
9876543120 / 8 = 1234567890
```

Solução com complexidade $O(K)$

- Como o total de pares deve ser listado, a solução deve utilizar a busca completa
- Observe que não são informados os limites da entrada
- O pior caso aconteceria com $N = 1$, onde todos os pares (x, x), com $x \leq 10^{10}$, seriam válidos
- Da relação apresentada, s_1 é um múltiplo de N
- Assim, basta testar os valores $s_2 = 1, 2, 3, \ldots$, até que o produto $p = s_2 N \geq 10^{10}$
- Isto porque se p tem 11 ou mais dígitos, certamente ele terá duas ou mais repetições de um mesmo dígito
- De fato, para cada N existem $K = 10^{10}/N$ pares possíveis a serem verificados

Solução com complexidade $O(K)$

```cpp
#include <bits/stdc++.h>

using namespace std;
using ll = long long;
using ii = pair<ll, ll>;

int digits_count(ll x)
{
    int total = 0;

    do {
        x /= 10;

        ++total;
    } while (x);

    return total;
}
```

Solução com complexidade $O(K)$

```
20 bool has_repeated_digits(ll x)
21 {
22     bitset<10> used;
23     used.reset();
24
25     while (x)
26     {
27         int d = x % 10;
28         x /= 10;
29
30         if (used[d])
31             return true;
32
33         used[d] = true;
34     }
35
36     return false;
37 }
```

Solução com complexidade $O(K)$

```
39 vector<ii> solve(ll N)
40 {
41     vector<ii> ans;
42
43     for (ll d = 1; digits_count(d*N) <= 10; ++d)
44     {
45         if (not has_repeated_digits(d) and not has_repeated_digits(d*N))
46             ans.push_back(ii(d*N, d));
47     }
48
49     return ans;
50 }
51
52 int main()
53 {
54     ios::sync_with_stdio(false);
55
56     int T;
57     cin >> T;
```

Solução com complexidade $O(K)$

```
59      for (int test = 1; test <= T; ++test)
60      {
61          ll N;
62          cin >> N;
63
64          auto ans = solve(N);
65
66          if (test > 1)
67              cout << '\n';
68
69          for (auto p : ans)
70              cout << p.first << " / " << p.second << " = " << N << '\n';
71      }
72
73      return 0;
74  }
```