OJ 11752

The Super Powers

Prof. Edson Alves - UnB/FGA

Problema

We all know the Super Powers of this world and how they manage to get advantages in political warfare or even in other sectors. But this is not a political platform and so we will talk about a different kind of super powers – "The Super Power Numbers". A positive number is said to be super power when it is the power of at least two different positive integers. For example 64 *is a super power as* $64 = 8^2$ *and* $64 = 4^3$*. You have to write a program that lists all super powers within* 1 *and* $2^{64} - 1$ *(inclusive).*

Entrada e saída

Input

This program has no input.

Output

Print all the Super Power Numbers within 1 *and* $2^{64} - 1$*. Each line contains a single super power number and the numbers are printed in ascending order.*

Note*: Remember that there are no input for this problem. The sample output is only a partial solution.*

Exemplo de entrada e saída

Entrada	Saída
	1 16 64 81 256 512 . . .

Solução

- O principal ponto a ser observado é que n tem que ser um número da forma m^c, onde c é um número composto
- Isto porque se c é composto, ele pode ser escrito como $c = rs$, com $r, s > 1$
- Daí

$$n = m^c = m^{rs} = (m^r)^s = (m^s)^r$$

- Como 4 é o menor número composto e $n^4 > 2^{64}$ para todos $n \geq 2^{16}$, a listagem dos números desejados pode ser obtida elevando-se todos os inteiros positivos no intervalo $[1, 2^{16})$ a todos os números compostos c no intervalo $[1, 64)$

Solução

- Os compostos menores ou iguais a n podem ser obtidos por meio de uma variante da função que determina se o número é ou não primo
- As possíveis repetições podem ser eliminadas se os resultados forem armazenados em um conjunto
- Para evitar o *overflow* no cálculo de n^c, é preciso saber se o resultado é ou não menor do que 2^{64}
- Isto pode ser verificado por meio de logaritmos, pois $n^c < 2^{64}$ se

$$c \log_2 n < 64 \log_2 2 = 64$$

Solução

```
5  vector<int> composite(int m)
6  {
7      vector<int> cs;
8
9      for (int n = 2; n < m; ++n)
10         for (int d = 2; d * d <= n; ++d)
11             if (n % d == 0)
12             {
13                 cs.push_back(n);
14                 break;
15             }
16
17     return cs;
18 }
```

Solução

```
20 unsigned long long power(int a, int n)
21 {
22     unsigned long long res = 1;
23
24     while (n--)
25         res *= a;
26
27     return res;
28 }
```

Solução

```
30 set<unsigned long long> solve()
31 {
32     auto cs = composite(64);
33     set<unsigned long long> ans;
34
35     for (int n = 1; n < (1 << 16); ++n)
36     {
37         for (auto c : cs)
38             if (c*log2(n) < 64)
39                 ans.insert(power(n, c));
40     }
41
42     return ans;
43 }
```