A sárkány feladványa

Innen: Algowiki

Tartalomjegyzék

Feladat

Kapunk 3 sornyi számpárt és mindegyikhez egy N számot. Adjunk a számpárhoz egy N számjegyű közös többszöröst, ha nem létezik ilyen adjunk vissza -1-et.

Eredeti feladat

Mester/Kezdő/Elemi feladatok/30. A sárkány feladványa

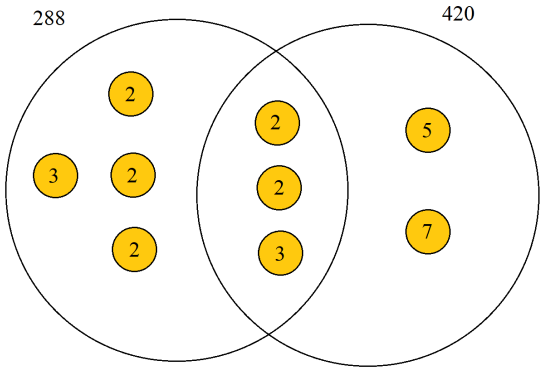
Megoldás

Ötlet

Keressük meg a két szám legkisebb közös többszörösét, majd vizsgáljuk meg a számjegyeit. Ha több, mint a kért N, akkor nem tudunk előállítani megfelelő többszöröst, ha kevesebb, akkor addig kell felszoroznunk egészekkel, amíg a kívánt számjegyet el nem érjük, ha egyenlő, akkor nincs semmi dolgunk.

Részletes megoldás

A **legkisebb közös többszörös** megkereséséhez az *Euklidesz algoritmust* fogjuk használni. Ha két szám prímtényezőit **Venn-diagramként** ábrázoljuk, ahol A az első szám tényezőit tartalmazza, B a második tényezőit, $A \cap B$ pedig a közös prímtényezőket, akkor a két szám legkisebb közös többszöröse $A \cup B$. A két szám szorzata tartalmazza az unió tényezőit, valamint a metszetét még egyszer, ebből adódóan ha elosztjuk a két szám szorzatát a metszetben lévő prímtényezőkkel, megkapjuk a várt többszöröst. A két halmaz metszete definíció szerint a két szám **legnagyobb közös osztója**, amit az algoritmus szerint megkaphatunk úgy, ha a két szám közül a nagyobból kivonjuk a kisebbet egészen addig, míg meg nem egyeznek, ekkor a hátramaradt szám a legnagyobb közös osztó.



2 szám prímtényezőinek Venn-diagramja

A számjegyek megszámlálásához a közös többszörös **tízes alapú logaritmusát** vennénk **felfelé kerekítve** a maximális gyorsaság érdekében (kivéve ha a logaritmus 0-t ad vissza, ekkor a többszörös a lehető legkisebb volt, azaz az 1, ami 0 helyett 1 számjegyből áll. Ez az egyetlen speciális eset), de a float értékek pontatlansága miatt ez egyetlen tesztesetre hibás választ ad vissza, ezért a második legjobb megoldást alkalmazzuk: egy while ciklusban addig osztjuk a számot tízzel, amíg 10-nél kisebb számot nem kapunk, miközben egy változót növelünk minden osztással.

A változókhoz **long long intet** használunk, aminek maximuma implementációfüggően, de optimális esetben kb. $9 \cdot 10^{18}$, ami 18 számjegyű. A feladatban az a, b számpárokra igaz, hogy $a, b \leq 100.000$, ebből adódóan a legnagyobb lehetséges legkisebb közös többszörös a két legnagyobb 100.000 alatti *relatív prímpáré* lesz, ami a 100.000 és 99.999, melyek legkisebb közös többszöröse $9.999.900.000$, ebből adódóan nem fog túlcsordulni semmink.

Ezekkel megadható minden eset, amiben vagy túl nagy a legkisebb közös többszörös, vagy pont megfelelő. Ha kevesebb számjegyből állna, mint a kívánt, akkor írjunk ki a szám mögé addig nullákat (mint 10-el való szorzás), amíg el nem érjük a megadott mennyiségű számjegyet.

Komplexitás

A legkisebb közös többszörös kiszámítása $O(log(a+b))$, a számjegyek számlálása $O(log(n))$ komplexitású. Az extra nullák kiíratása az std::string konstruktorával ciklus nélkül megoldható

Implementáció

C++

Kód [becsuk]

Kód:

```cpp
#include <iostream>
#include <math.h>
#include <string>

long long int least_common_multiple(long long int a, long long int b)
{
    long long int product = a * b;

    while (a != b) // Euklidesz algoritmus
    {
        if (a > b)
            a -= b;
        else
            b -= a;
    }
    return product / a;

}

int how_many_digits(long long int n)
{
    int counter = 1;
    while (n >= 10) // legrosszabb esetben 9-szer lép be
    {
        n = n / 10;
        counter++;
    }
    return counter;
}

int main()
{
    for (int round = 0; round < 3; round++)
    {
```

```
35          int a, b;
36          int digit;
37          std::cin >> a >> b >> digit;
38          long long int lcm = least_common_multiple(a, b);
39          int ans_digit = how_many_digits(lcm);
40          if (ans_digit == digit) // elsőre jó megoldás
41          {
42              std::cout << lcm << std::endl;
43          }
44          else if (ans_digit > digit) // nincs jó megoldás
45          {
46              std::cout << -1 << std::endl;
47          }
48          else // még "növelni" kell a számot
49          {
50              int power = digit - ans_digit;
51              std::cout << lcm;
52
53              std::string ans(power, '0'); // power db '0' karakter egymás mellett
54              std::cout << ans;
55              std::cout << std::endl;
56          }
57      }
58  }
```

A lap eredeti címe: „https://algowiki.miraheze.org/w/index.php?title=A_sárkány_feladványa&oldid=1370”