Easy Arithmetic, Hard Arithmetic (Programming Club 4)

Tomasz Kosciuszko

11.02.2019

1 Introduction

Number theory is part of mathematics which speaks about **divisibility** of integers. Pizza with 12 pieces can be divided equally among 2,3 or 6 persons (we reject 1 and 12 as too trivial). On the other hand if you encounter a pizza with 13 pieces you can feel unlucky, there is no way to share it equally. That is because 13 is a **prime** number. While amateurs of pizza dislike primes mathematicians love them.

Prime numbers are source of many unsolved conjectures which are very easy to formulate and understand, yet modern mathematics struggles to make any progress. If you are curious you can look up: Goldbach conjecture, Twin Prime conjecture, Collatz conjecture.

Because of this mysterious nature of primes, number theory is very interesting to **cryptographers**. You can be sure that if your data is secured by a mathematical problems unsolved for centuries nobody will learn your secrets. However, the charm of number theory is that two problems might look similar, when one if simple and one very hard. We will look at GCD (Greatest Comon Divisor) and at factorization as examples of that.

2 GCD - Euclid's Algorithm

Let's assume we have been give two large numbers a, b (for example $a = 2^{1543} + 17$ and $b = 3^{954} + 4$). Are these two numbers co-prime (there is no integer $d > 1$ which can divide both a and b) or does there exist a common divisor? If yes, what is the Greatest Common Divisor? The algorithm which answers this question is almost trivial and very fast:

- If $b = 0$ then the answer is a.
- Otherwise assign $a' := b$ and $b' = a\%b$. % is the modulo operation.
- Repeat.

```
def gcd(a, b):
    if b == 0:
        return a
    else:
        return gcd(b, a%b)
```

We notice that if d divides both a, b then it divides a', b' and vice-versa, so the algorithm is correct. Moreover, after 2 steps a decreases at least two times, so the complexity is $O(\log a)$ (very fast, it would take less than 10000 steps to compute GCD of numbers given above, even though they are much bigger than any quantity we can ever imagine).

3 GCD - Think about this!

- How can we check whether two numbers are co-prime? (have no common divisors)
- Chess world championship happens every x years and golf world championship every y years. In 2019 they both take place, how many years do we have to wait for the next such year?
- Let $a = 5 \cdot 7 \cdot 13$, how many numbers smaller than a are co-prime with a?

4 Factorization

The problem of factorization is: given a number b, what are the prime numbers $p_1, p_2, ..., p_n$ such that their product is b, or is b prime itself? This turns out to be hard for big b.

We can make one reasonable observation: if b has no divisors in the range $2...\sqrt{b}$ then b has no divisors at all. That is because if, say, $b = xy$ then at least one from x, y is no bigger than $\sqrt{b}$. That can be used to come up with a simple algorithm:

- Check numbers $2...\sqrt{b}$ looking for d that divides b.
- If d is not found then return b as prime.
- Otherwise divide $b' = b/d$ and continue with the first step.

```
void Factorize(long long n) {
    if (n == 1) {
        cout << "1\n";
        return;
    }
    while (n > 1) {
        for (int d = 2; d * d <= n; d++) {
            if (n % d == 0) {
                cout << d << " ";
                n /= d;
                break;
            }
        }
        break;
    }
    cout << "\n";
}
```

The complexity of this algorithm $O(\sqrt{n})$, that means that for $n = 2^{282589933} - 1$ (prime!) you can wait until the last star disappears from the sky and you will not get the result.

5 Factorization - Think about this!

- Is $2^{100} + 4$ prime?
- Is $2^{282589934} - 1$ prime? (tricky!)
- What are the prime factors of 100? Are they unique?
- What are the prime factors of 100^{100}?

6 Further problems

- Easy: `http://codeforces.com/problemset/problem/776/B`
- Easy: `https://www.hackerrank.com/challenges/sherlock-and-gcd/problem`
- Medium: `https://www.hackerrank.com/challenges/constructing-a-number/problem`
- Hard: `http://codeforces.com/problemset/problem/980/D`