RAUCODER 2025

EDITORIAL

24 mai 2025

Problema 1. Bomboane

Autor: Crișan Daniela Alexandra, conf. dr., Universitatea Româno-Americană

Cuvinte cheie: Principiul includerii și excluderii, Stars and Bars, Măști de biți, Aritmetica modulară, Exponențiere rapidă

Indicații

Pasul 1. Determinarea numărului de elevi

Pentru a afla numărul de elevi, trebuie să determinăm câte numere între 1 și N sunt coprime cu Q. Pentru aceasta vom folosi Principiul Includerii și Excluderii.

Vom afla câte numere între 1 și N nu au niciun divizor comun cu Q, în afară de 1.
Dacă $Q=p_1^{a1}\cdot p_2^{a2}\cdot\ldots\cdot p_q^{aq}$, atunci vom scădea din N toate numerele care **NU** sunt coprime cu Q, adică toate cele care sunt divizibile cu cel puțin unul din factorii primi ai lui Q: p_1, p_2, ..., p_q.

$$nr\_elevi = N - \sum|Ai| + \sum|Ai \cap Aj| - \sum|Ai \cap Aj \cap Aq| + \ldots$$

unde Ai este mulțimea numerelor între 1 și N divizibile cu p_i.

Vom calcula acest număr astfel: considerăm toate submulțimile de factori primi $S \subseteq \{p_1,p_2,\ldots,p_q\}$ și calculăm:

- d = produsul tuturor primilor din S
- partea întreagă [N/d] va fi numărul de multipli ai lui d în intervalul [1,N]
- adunăm sau scădem [N/d] în funcție de paritatea mărimii (cardinalului) lui S (un număr prim → scădem, două numere prime → adunăm etc.)

Pentru a genera toate submulțimile S se vor folosi măști de biți.

Pasul 2. Verificarea dacă bomboanele ajung

Numărul de colegi este k = nr_elevi -1, așadar bomboanele ajung dacă numărul lor, M, este cel puțin egal cu k.

Pasul 3. Împărțirea bomboanelor – Număr de moduri

Dacă bomboanele sunt suficiente, trebuie să aflăm în câte moduri putem împărți M bomboane către k = nr_colegi, cu cel puțin una pentru fiecare. Avem o problemă clasică de numărare a partițiilor cu restricții:
C(M−1, k-1) pentru care putem apela la metoda Stars and Bars.

Pentru a calcula eficient binomul modulo MOD = 10^9+7 se vor folosi aritmetica modulară și exponențierea rapidă.

Complexitate timp: $O(\sqrt{Q} + 2^{\wedge}q + M + log\ MOD)$, unde q = numărul de factori primi ai lui Q
Complexitate memorie: O(q)
(numerele Q ≤ 10^6 au cel mult 7 factori primi)

Soluție

```
#include <fstream>
using namespace std;

ifstream in("bomboane.in");
ofstream out("bomboane.out");

#define ll long long
const int MOD = 1e9 + 7;

static int count_coprime_with_Q(int N, int Q) {
    int primes[10], cnt=0;
    int q = Q;
    for (int p = 2; p * p <= q; ++p) {
        if (q % p == 0) {
            primes[cnt++] = p;
            while (q % p == 0)
                q /= p;
        }
    }
    if (q > 1)
        primes[cnt++] = q;

    int total = 0;
    int subsets = 1 << cnt;
    for (int mask = 1; mask < subsets; ++mask) {
        int mult = 1, bits = 0;
        for (int i = 0; i < cnt; ++i) {
            if (mask & (1 << i)) {
                mult *= primes[i];
                bits++;
            }
        }
        int x = N / mult;
        if (bits % 2 == 1) total += x;
        else total -= x;
    }
    return N - total;
}

static ll power(ll base, ll exp) {
    ll result = 1;
    while (exp > 0) {
        if (exp % 2 == 1) result = (result * base) % MOD;
        base = (base * base) % MOD;
        exp /= 2;
    }
    return result;
}

static ll comb(int n, int k) {
    if (k > n || k < 0) return 0;
    ll num = 1, den = 1;
```

```
    for (int i = 0; i < k; i++) {
        num = (num * (n - i)) % MOD;
        den = (den * (i + 1)) % MOD;
    }
    return (num * power(den, MOD - 2)) % MOD;
}

int main()
{
    int Q, N, M;
    in >> Q >> N >> M;
    int C = count_coprime_with_Q(N, Q);
    out << C << "\n";
    if (M >= C - 1)
        out << comb(M - 1, C - 2);
    else
        out << "NU";

    return 0;
}
```