Solutions for APC Contest

Nguyễn Quản Bá Hồng*

Ngày 3 tháng 12 năm 2025

Link bài toán:

Bài toán 1 (H96's chicken paradise)**.** *Tóm tắt: Có n loại gà, gà loại thứ i có $a_i \in \mathbb{N}^\star$ chân, $\forall i \in [n]$, có tổng cộng M chân gà. Giả sử có $c_i \in \mathbb{N}$ con gà loại thứ i, thì $\{c_i\}_{i=1}^n \subset \mathbb{N}$ thỏa phương trình nghiệm nguyên*

$$\sum_{i=1}^n a_i c_i = a_1 c_1 + a_2 c_2 + \cdots + a_n c_n = M,\ c_i \in \mathbb{N},\ \forall i \in [n]. \tag{1}$$

Tìm GTNN & GTLN của tổng số gà $\sum_{i=1}^n c_i = c_1 + c_2 + \cdots + c_n$. Nếu không có bộ $\{c_i\}_{i=1}^n$ nào thỏa mãn (1) thì in ra -1.

Input. *Dòng 1: $n \in [100], m \in [10^5]$. Dòng 2: n số nguyên $\{a_i\}_{i=1}^n$, $a_i \in [10^5]$, $\forall i \in [n]$.*

Output. *Nếu (1) vô nghiệm thì in ra -1. Nếu (1) có nghiệm thì in ra 2 số: số con gà ít nhất & nhiều nhất có thể.*

Giải. Bản chất của bài toán này là 2 bài toán tối ưu nguyên (integer programming, see, e.g., Wikipedia/integer programming):

$$\min \sum_{i=1}^n c_i \ \&\ \max \sum_{i=1}^n c_i \text{ such that } (c_1, c_2, \ldots, c_n) \text{ satisfies } \sum_{i=1}^n a_i c_i = M.$$

Hiển nhiên trong trường hợp (1) vô nghiệm thì tập không gian tìm kiếm là tập rỗng, i.e., $\{\{c_i\}_{i=1}^n; \sum_{i=1}^n a_i c_i = M\} = \emptyset$ nên ta không thể tìm phần tử cực đại & phần tử cực tiểu của hàm tổng $S(c_1, c_2, \ldots, c_n) \coloneqq \sum_{i=1}^n c_i$ (đơn giản vì chẳng có gì để tìm trong tập rỗng cả). Dễ thấy (1) sẽ chắn chắc vô nghiệm khi $M \not\vdots \gcd(a_1, a_2, \ldots, a_n)$ (why?)

Nếu (1) có ít nhất 1 nghiệm, i.e., $\{\{c_i\}_{i=1}^n; \sum_{i=1}^n a_i c_i = M\} \neq \emptyset$, ta có thể tìm kiếm phần tử cực đại trên không gian tìm kiếm vì không gian này khác rỗng & hữu hạn (why?). Gọi $f_{\min}(x), f_{\max}(x)$ lần lượt là số con gà nhỏ nhất & lớn nhất thỏa tổng số chân của chúng bằng x, i.e.:

$$f_{\min}(x) \coloneqq \min\left\{\sum_{i=1}^n c_i; \sum_{i=1}^n a_i c_i = x\right\},\ f_{\max}(x) \coloneqq \max\left\{\sum_{i=1}^n c_i; \sum_{i=1}^n a_i c_i = x\right\}.$$

Hiển nhiên $f_{\min}(0) = f_{\max}(0) = 0$ (trường hợp cơ sở – base case) vì nghiệm tự nhiên duy nhất của phương trình $\sum_{i=1}^n a_i c_i = 0$ là $(c_1, c_2, \ldots, c_n) = (0, 0, \ldots, 0)$ (vì $a_i > 0, c_i \geq 0$, $\forall i \in [n]$). Chiến thuật: Khởi tạo $f_{\min}(x) = \infty$ hoặc 1 giá trị đủ lớn, e.g., 10^9, $\forall x \in [M]$ & khởi tạo $f_{\max}(x) = -1$, $\forall x \in [M]$. Với mỗi $i \in [n]$, ta xét loại gà thứ i, & giả sử có ít nhất 1 con gà loại i, i.e., $c_i \geq 1$, thì ta sẽ xét các giá trị từ a_i tới M & cập nhật giá trị của 2 hàm bằng công thức:

$$f_{\min}(x) = \min(f_{\min}(x), f_{\min}(x - a_i) + 1), f_{\max}(x) = \max(f_{\max}(x), f_{\max}(x - a_i) + 1).$$

2 công thức quy hoạch động này hoạt động đúng với yêu cầu đề bài vì với mỗi $i \in [n]$, ta cập nhật giá trị 2 hàm $f_{\min}, f_{\max}$ tại giá trị x bởi giá trị 2 hàm đó tại giá trị $x - a_i$, việc này sẽ ngầm tạo các tổ hợp tuyến tính hệ số nguyên có dạng $\sum_{i=1}^n \alpha_i a_i$, với $\alpha_i \in \mathbb{N}$, $\forall i \in [n]$, nên sẽ quét được hết các cấu hình thỏa mãn, những số x nào vẫn cho $f_{\min}(x) = \infty$ hoặc $f_{\max}(x) = -1$ thì tức không có tổ hợp tuyến tính hệ số nguyên nào của $\{a_i\}_{i=1}^n$ bằng x. □

```
#include <bits/stdc++.h>
using namespace std;

int main() {
    ios_base::sync_with_stdio(false);
    cin.tie(nullptr);
    int n, M;
    cin >> n >> M;
    vector<int> a(n), dpmin(M + 1, 1e9), dpmax(M + 1, -1);
    dpmin[0] = 0;
    dpmax[0] = 0;
```

*A scientist- & creative artist wannabe, a mathematics & computer science lecturer of Department of Artificial Intelligence & Data Science (AIDS), School of Technology (SOT), UMT Trường Đại học Quản lý & Công nghệ TP.HCM, Hồ Chí Minh City, Việt Nam.
E-mail: nguyenquanbahong@gmail.com & hong.nguyenquanba@umt.edu.vn. Website: https://nqbh.github.io/. GitHub: https://github.com/NQBH.

```
    for (int &x : a) cin >> x;
    int g = a[0];
    for (int i = 1; i < n; ++i) g = gcd(g, a[i]);
    if (M % g != 0) cout << "-1";
    else {
        for (int i = 0; i < n; ++i)
        for (int j = a[i]; j <= M; ++j) {
            if (dpmin[j - a[i]] != 1e9) dpmin[j] = min(dpmin[j], dpmin[j - a[i]] + 1);
            if (dpmax[j - a[i]] != -1) dpmax[j] = max(dpmax[j], dpmax[j - a[i]] + 1);
        }
        if (dpmin[M] == 1e9 || dpmax[M] == -1) cout << "-1\n";
        else cout << dpmin[M] << ' ' << dpmax[M] << '\n';
    }
}
```

Link bài toán:

Bài toán 2 (Cậu bé tinh ranh). *Tóm tắt: Đếm số chuỗi s độ dài n gồm 2 ký tự A, S sao cho s không chứa đoạn SS & có ít nhất k ký tự A theo modulo $m \coloneqq 10^9 + 7$.*

Input. *2 số $n \in [2000], 0 \le k \le n$.*

Output. *Số lượng chuỗi s thỏa mãn theo modulo m.*

Giải. Vì s có ít nhất k ký tự A, nên nếu gọi a là số ký tự A của 1 chuỗi s thỏa mãn thì $a \in \{k, k+1, \ldots, n\}$. Với mỗi $a \in \{k, k+1, \ldots, n\}$, dùng bài toán chia kẹo Euler (Euler's candy problem) bằng cách tạo $a+1$ "vách ngăn" $\star$ như sau: $\star A \star A \star \ldots \star A \star$. Vì $|s| = n$, nên số ký tự S bằng $n-a$, ta cần đặt $n-a$ ký tự S này vào các vách ngăn $\star$. Vì s không chứa 2 ký tự SS liên tiếp, nên mỗi vách ngăn chỉ chứa tối đa 1 ký tự S, i.e., 0 hoặc 1 ký tự S. Vì cần chọn ra $n-a$ trong $a+1$ vách ngăn, nên số chuỗi s hợp lệ & có đúng a ký tự A, $n-a$ ký tự S bằng $C_{a+1}^{n-a} = \binom{a+1}{n-a}$. Cho a chạy trong tập $\{k, k+1, \ldots, n\}$, suy ra tổng số chuỗi s thỏa mãn bằng

$$A \coloneqq \sum_{a=k}^{n} C_{a+1}^{n-a} = \sum_{a=k}^{n} \binom{a+1}{n-a},$$

nên cần xuất ra $A \mod m$. □

```
#include <bits/stdc++.h>
using namespace std;
using ll = long long;
const int m = 1000000007;

ll binpow(ll a, ll b, ll m) {
    a %= m;
    ll res = 1;
    while (b) {
        if (b & 1) res = res * a % m;
        a = a * a % m;
        b >>= 1;
    }
    return res;
}

const int MAXN = 2e5;
ll fac[MAXN + 1], inv[MAXN + 1];

/** @return x^n modulo m in O(log p) time. */
ll exp(ll x, ll n, ll m) {
    x %= m;  // note: m * m must be less than 2^63 to avoid ll overflow
    ll res = 1;
    while (n > 0) {
        if (n % 2 == 1) { res = res * x % m; }
        x = x * x % m;
        n /= 2;
    }
    return res;
}

```

```
/** Precomputes n! from 0 to MAXN. */
void factorial(ll p) {
    fac[0] = 1;
    for (int i = 1; i <= MAXN; i++) fac[i] = fac[i - 1] * i % p;
}

/**
* Precomputes all modular inverse factorials from 0 to MAXN in O(n + log p) time
*/
void inverses(ll p) {
    inv[MAXN] = exp(fac[MAXN], p - 2, p);
    for (int i = MAXN; i >= 1; i--) inv[i - 1] = inv[i] * i % p;
}

/** @return nCr mod p */
ll C(ll n, ll r, ll p) {
    if (r < 0 || r > n) return 0;
    return fac[n] * inv[r] % p * inv[n - r] % p;
}

int main() {
    ios_base::sync_with_stdio(false);
    cin.tie(nullptr);
    int n, k;
    cin >> n >> k;
    ll ans = 0;
    factorial(m);
    inverses(m);
    for (int a = k; a <= n; ++a) { // Euler's candy problem
        ans += C(a + 1, n - a, m);
        ans %= m;
    }
    cout << ans;
}
```

Có cách giải bằng quy hoạch động, nhưng có vẻ độ phức tạp sẽ cao hơn cách thuần tổ hợp này.