Standard Code Library

Boboge

Zhejiang University City College

November 12, 2021

Contents

一切的开始 **2**
- fread 快速读 . . . 2
- getchars() 快读 . . . 2
- dbg 宏 . . . 3
- mt19937 . . . 3
- 重载哈希用于 unordered_set . . . 3

数据结构 **3**
- 主席树 . . . 3

数学 **4**
- 倍增求矩阵幂和 . . . 4
- 二次剩余 . . . 5
- 高斯消元 . . . 6
- 矩阵快速幂 . . . 7
- 快速判断能否被一些数整除 . . . 8
- 欧拉函数 . . . 9
- 欧拉降幂 . . . 10
- 判断第二类斯特林数奇偶性 . . . 10
- 三分 . . . 10
  - 凸函数 . . . 10
  - 凹函数 . . . 11
- 线性基 . . . 11
- exBSGS . . . 11
- exCRT . . . 12
- Miller-Rabin & Pollard-Rho . . . 13
- FFT . . . 14
- NTT . . . 16

图论 **17**
- 点分治 . . . 17
- 轻重链剖分 . . . 18
- 虚树 . . . 21
- 网络流 . . . 23
- Tarjan . . . 23
- 倍增 LCA . . . 23
- dsu on tree . . . 24

几何板子 **25**
- pick 定理 . . . 25

字符串 **25**
- AC 自动机 . . . 25
- Z 函数（扩展 KMP） . . . 27
- 马拉车 . . . 28
- 双哈希方便写法 . . . 28

杂项 **29**
- 日期 . . . 29
- 子集枚举 . . . 29
- 数位 DP . . . 30
- 背包 . . . 30

一切的开始

fread 快速读

```
inline char nc() {
    static char buf[100000], *p1 = buf, *p2 = buf;
    return p1 == p2 && (p2 = (p1 = buf) + fread(buf, 1, 100000, stdin), p1 == p2) ? EOF : *p1++;
}
template <typename T>
bool rn(T& v) {
    static char ch;
    while (ch != EOF && !isdigit(ch)) ch = nc();
    if (ch == EOF) return false;
    for (v = 0; isdigit(ch); ch = nc())
        v = v * 10 + ch - '0';
    return true;
}

template <typename T>
void o(T p) {
    static int stk[70], tp;
    if (p == 0) { putchar('0'); return; }
    if (p < 0) { p = -p; putchar('-'); }
    while (p) stk[++tp] = p % 10, p /= 10;
    while (tp) putchar(stk[tp--] + '0');
}
```

- 需要初始化
- 需要一次读入
- 不支持负数

```
const int MAXS = 100 * 1024 * 1024;
char buf[MAXS];
template<typename T>
inline bool read(T& x) {
    static char* p = buf;
    x = 0;
    while (*p && !isdigit(*p)) ++p;
    if (!*p) return false;
    while (isdigit(*p)) x = x * 10 + *p++ - 48;
    return true;
}

fread(buf, 1, MAXS, stdin);
```

getchars() 快读

```
//quick read
template<typename T>
inline void read(T &x) {
    int s = 1;
    x = 0;
    char ch = getchar();
    while (ch < '0' || ch > '9') {
        if (ch == '-') s = -1;
        ch = getchar();
    }
    while (ch >= '0' && ch <= '9') {
        x = (x << 3) + (x << 1) + (ch ^ 48);
        ch = getchar();
    }
    x *= s;
}

template<typename T, typename... Args>
inline void read(T &x, Args &... args) {
    read(x);
    read(args...);
}
```

dbg 宏

```cpp
#define dbg(x...) \
    do { \
        cout << #x << " -> "; \
        err(x); \
    } while (0)

void err() {
    cout << endl;
}

template<class T, class... Ts>
void err(T arg, Ts &... args) {
    cout << arg << ' ';
    err(args...);
}
```

mt19937

```cpp
mt19937 mt(chrono::steady_clock::now().time_since_epoch().count());
ll rng(ll l, ll r) {
    uniform_int_distribution<ll> uni(l, r);
    return uni(mt);
}
```

重载哈希用于 unordered_set

```cpp
class my_hash {
public:
    ull operator()(const pair<ll, ll> &p) const {
        return (ull) p.first * P + (ull) p.second;
    }
};

//unorder_set<pair<ll, ll>, my_hash> s;
```

数据结构

主席树

```cpp
#include <bits/stdc++.h>

using namespace std;
const int maxn = 2e5 + 10;
int tot;
int sum[maxn << 5], id[maxn], ls[maxn << 5], rs[maxn << 5];
int a[maxn], len;


int build(int l, int r) {
    int root = ++tot;
    if (l == r) return root;
    int mid = l + r >> 1;
    ls[root] = build(l, mid);
    rs[root] = build(mid + 1, r);
    return root;
}

int update(int k, int l, int r, int root) {
    int dir = ++tot;
    ls[dir] = ls[root], rs[dir] = rs[root], sum[dir] = sum[root] + 1;
    if (l == r) return dir;
    int mid = l + r >> 1;
    if (k <= mid)
        ls[dir] = update(k, l, mid, ls[dir]);
    else
        rs[dir] = update(k, mid + 1, r, rs[dir]);
    return dir;
```

```
}

int query(int u, int v, int l, int r, int k) {
    int mid = l + r >> 1, x = sum[ls[v]] - sum[ls[u]];
    if (l == r) return l;
    if (k <= x)
        return query(ls[u], ls[v], l, mid, k);
    else
        return query(rs[u], rs[v], mid + 1, r, k - x);
}

vector<int> ind;

int getid(int val) {
    return lower_bound(ind.begin(), ind.end(), val) - ind.begin() + 1;
}

int main() {
    int n, m;
    scanf("%d%d", &n, &m);
    for (int i = 1; i <= n; ++i) scanf("%d", a + i), ind.push_back(a[i]);
    sort(ind.begin(), ind.end());
    ind.erase(unique(ind.begin(), ind.end()), ind.end());
    len = ind.size();
    id[0] = build(1, len);
    for (int i = 1; i <= n; ++i) id[i] = update(getid(a[i]), 1, len, id[i - 1]);
    while (m--) {
        int l, r, k;
        scanf("%d%d%d", &l, &r, &k);
        printf("%d\n", ind[query(id[l - 1], id[r], 1, len, k) - 1]);  // 回答询问
    }
    return 0;
}
```

数学

倍增求矩阵幂和

```
struct mat {
    int data[N][N] = {};
    int size{};

    int *operator[](int index) {
        return data[index];
    }
} G, e;

mat operator+(const mat &a, const mat &b) {
    mat ret;
    ret.size = a.size;
    for (int i = 1; i <= a.size; ++i) {
        for (int j = 1; j <= a.size; ++j) {
            ret.data[i][j] = a.data[i][j] + b.data[i][j];
        }
    }
    return ret;
}

mat mul(mat &A, mat &B) {
    mat C;
    C.size = A.size;
    for (int i = 1; i <= A.size; i++) {
        for (int k = 1; k <= A.size; k++) {
            for (int j = 1; j <= A.size; j++) {
                C[i][j] = (C[i][j] + A[i][k] * B[k][j]) % mod;
            }
        }
    }
    return C;
}
```

```
mat matpow(mat A, int n) {
    mat B;
    B.size = A.size;
    for (int i = 1; i <= A.size; i++) {
        B[i][i] = 1;
    }
    while (n) {
        if (n & 1) B = mul(B, A);
        A = mul(A, A);
        n >>= 1;
    }
    return B;
}

/* 倍增法求解 A^1 + A^2 + ... + A^n*/
mat pow_sum(const mat &a, int n) {
    if (n == 1) return a;
    mat tmp = pow_sum(a, n / 2);
    mat tt = matpow(a, n / 2);
    mat sum = tmp + mul(tmp, tt);
    ///若 n 为奇数, n/2 + n/2 = n-1, 因此 sum 需要加上 A^(n) 这一项
    if (n & 1) sum = sum + matpow(a, n);
    return sum;
}
```

二次剩余

```
/*
 * 二次剩余, mod 为奇素数时有解
 * 解最多两个, 为相反数 x, (mod-x)
 * n 为 0 特判
 * 算法返回-1 则无解
 * mod 为 2 返回 n
 */
#include<bits/stdc++.h>

using namespace std;
typedef long long ll;
ll w;
struct num {
    ll x, y;
};

num mul(num a, num b, ll p) {
    num ans = {0, 0};
    ans.x = ((a.x * b.x % p + a.y * b.y % p * w % p) % p + p) % p;
    ans.y = ((a.x * b.y % p + a.y * b.x % p) % p + p) % p;
    return ans;
}

ll qpow_real(ll a, ll b, ll p) {
    ll ans = 1;
    while (b) {
        if (b & 1)ans = 1ll * ans % p * a % p;
        a = a % p * a % p;
        b >>= 1;
    }
    return ans % p;
}

ll qpow_imag(num a, ll b, ll p) {
    num ans = {1, 0};
    while (b) {
        if (b & 1)ans = mul(ans, a, p);
        a = mul(a, a, p);
        b >>= 1;
    }
    return ans.x % p;
}
```

```
ll solve(ll n, ll p) {
    n %= p;
    if (p == 2)return n;
    if (qpow_real(n, (p - 1) / 2, p) == p - 1)return -1;//不存在
    ll a;
    while (1) {
        a = rand() % p;
        w = ((a * a % p - n) % p + p) % p;
        if (qpow_real(w, (p - 1) / 2, p) == p - 1)break;
    }
    num x = {a, 1};
    return qpow_imag(x, (p + 1) / 2, p);
}

int main() {
    srand(time(0));
    int t;
    scanf("%d", &t);
    while (t--) {
        ll n, p;
        scanf("%lld%lld", &n, &p);
        if (!n) {
            printf("0\n");
            continue;
        }
        ll ans1 = solve(n, p), ans2;
        if (ans1 == -1) {
            printf("Hola!\n");
        } else {
            ans2 = p - ans1;
            if (ans1 > ans2)swap(ans1, ans2);
            if (ans1 == ans2)printf("%lld\n", ans1);
            else printf("%lld %lld\n", ans1, ans2);
        }
    }
}
```

高斯消元

```
#include <iostream>
#include <iomanip>
#include <cmath>

using namespace std;

const int maxn = 10;
const int maxq = 100;

int n;
double a[maxn][maxn + 1];
double res[maxn] = {};
double query[maxq];

void solve() {
    int now;
    double temp;
    for (int j = 0; j < n; j++) {
        now = j;
        for (int i = j; i < n; i++)
            if (fabs(a[i][j]) > fabs(a[now][j]))
                now = i;
        if (now != j)
            for (int i = 0; i < n + 1; i++) {
                double t = a[j][i];
                a[j][i] = a[now][i];
                a[now][i] = t;
            }
        // œ° ‘™
        for (int i = j + 1; i < n; i++) {
            temp = a[i][j] / a[j][j];
            for (int k = j; k <= n; k++)
```

```
                a[i][k] -= a[j][k] * temp;
            }
        }
        for (int i = n - 1; i >= 0; i--) {
            if (a[i][i] == 0) {
                res[i] = 0;
                continue;
            }
            res[i] = a[i][n];
            for (int j = i + 1; j < n; j++) {
                res[i] -= a[i][j] * res[j];
            }
            res[i] /= a[i][i];
        }
    }

    int main() {
        cin >> n;
        if (n == 1) cin >> a[0][0] >> a[0][1];
        else {
            for (int i = 0; i < n; i++) {
                double x, y;
                cin >> x >> y;
                a[i][n - 1] = 1;
                a[i][n] = y;
                for (int j = 0; j < n - 1; j++) {
                    a[i][j] = pow(x, n - j - 1);
                }
            }
        }
        int q;
        cin >> q;
        for (int i = 0; i < q; i++) cin >> query[i];
        if (n == 1) {
            double k = a[0][1] / a[0][0];
            for (int i = 0; i < q; i++) {
                double ans = query[i] * k;
                if (ans >= -0.005 && ans < 0) ans = 0;
                cout << fixed << setprecision(2) << ans << '\n';
            }
        } else {
            solve();
            for (int i = 0; i < q; i++) {
                double ans = 0;
                for (int j = 0; j < n; j++) {
                    ans += pow(query[i], n - j - 1) * res[j];
                }
                if (ans >= -0.005 && ans < 0) ans = 0;
                cout << fixed << setprecision(2) << ans << '\n';
            }
        }
        return 0;
    }
```

矩阵快速幂

```
//矩阵快速幂
#include <vector>
#include <iostream>

using namespace std;
typedef long long ll;
const int maxn = 250;

struct mat {
    ll data[maxn][maxn] = {};
    int size;

    ll *operator[](int index) {
        return data[index];
    }
```

```
};

const ll mod = 1e9 + 7;

mat mul(mat &A, mat &B) {     //矩阵乘法
    mat C;
    C.size = A.size;
    for (int i = 0; i < A.size; i++) {
        for (int k = 0; k < A.size; k++) {
            for (int j = 0; j < A.size; j++) {
                C[i][j] = (C[i][j] + A[i][k] * B[k][j]) % mod;
            }
        }
    }
    return C;
}

mat matpow(mat A, ll n) {       //矩阵快速幂
    mat B;
    B.size = A.size;
    for (int i = 0; i < A.size; i++) {
        B[i][i] = 1;
    }
    while (n) {
        if (n & 1) B = mul(B, A);
        A = mul(A, A);
        n >>= 1;
    }
    return B;
}

int main() {
    int n;
    ll k;
    cin >> n >> k;
    mat A;
    A.size = n;
    for (int i = 0; i < n; i++) {
        for (int j = 0; j < n; j++) {
            cin >> A[i][j];
        }
    }
    A = matpow(A, k);
    for (int i = 0; i < n; i++) {
        for (int j = 0; j < n; j++) {
            cout << A[i][j] << " ";
        }
        cout << endl;
    }
    return 0;
}
```

快速判断能否被一些数整除

1. 被 **2** 整除的数的特征：一个整数的末位是偶数（0、2、4、6、8）的数能被 2 整除。
2. 被 **3** 整除的数的特征：一个整数的数字和能被 3 整除，则这个数能被 3 整除。
3. 被 **4** 整除的数的特征：**一个整数的末尾两位数能被 4 整除则这个数能被 4 整**除。可以这样快速判断：最后两位数，要是十位是单数，个位就是 2 或 6，要是十位是双数，个位就是 0、4、8。
4. 被 **5** 整除的数的特征：一个整数的末位是 0 或者 5 的数能被 5 整除。
5. 被 **6** 整除的数的特征：一个整数能被 2 和 3 整除，则这个数能被 6 整除。
6. 被 **7** 整除的数的特征：“割减法”。若一个整数的个位数字截去，再从余下的数中，减去个位数的 2 倍，这样，一次次下去，直到能清楚判断为止，如果差是 7 的倍数（包括 0），则这个数能被 7 整除。过程为：截尾、倍大、相减、验差。例如，判断 133 是否 7 的倍数的过程如下：13 − 3×2 = 7，所以 133 是 7 的倍数；又例如判断 6139 是否 7 的倍数的过程如下：613 − 9×2 = 595，59 − 5×2 = 49，所以 6139 是 7 的倍数，余类推。

7. 被 **8** 整除的数的特征：一个整数的末尾三位数能被 8 整除，则这个数能被 8 整除。
8. 被 **9** 整除的数的特征：一个整数的数字和能被 9 整除，则这个数能被 9 整除。
9. 被 **10** 整除的数的特征：一个整数的末位是 0，则这个数能被 10 整除。
10. 被 **11** 整除的数的特征：“奇偶位差法”。一个整数的奇位数字之和与偶位数字之和的差是 11 的倍数（包括 0），则这个数能被 11 整除。（隔位和相减）。例如，判断 491678 能不能被 11 整除的过程如下：奇位数字的和 9+6+8=23，偶位数位的和 4+1+7=12。23-12=11。因此 491678 能被 11 整除。
11. 被 **12** 整除的数的特征：一个整数能被 3 和 4 整除，则这个数能被 12 整除。
12. 被 **13** 整除的数的特征：若一个整数的个位数字截去，再从余下的数中，加上个位数的 4 倍，这样，一次次下去，直到能清楚判断为止，如果是 13 的倍数（包括 0），则这个数能被 13 整除。过程为：截尾、倍大、相加、验差。
13. 被 **17** 整除的数的特征：若一个整数的个位数字截去，再从余下的数中，减去个位数的 5 倍，这样，一次次下去，直到能清楚判断为止，如果差是 17 的倍数（包括 0），则这个数能被 17 整除。过程为：截尾、倍大、相减、验差。
14. 被 **19** 整除的数的特征：若一个整数的个位数字截去，再从余下的数中，加上个位数的 2 倍，这样，一次次下去，直到能清楚判断为止，如果是 19 的倍数（包括 0），则这个数能被 19 整除。过程为：截尾、倍大、相加、验差。
15. 被 **7、11、13** 整除的数的共同特征：若一个整数的末 3 位与末 3 位以前的数字所组成的数之差（以大减小）能被 7、11、13 整除，则这个数能被 7、11、13 整除。例如：128114，由于 128-114=14，14 是 7 的倍数，所以 128114 能被 7 整除。64152，由于 152-64=88，88 是 11 的倍数，所以 64152 能被 11 整除。94146，由于 146-94=52，52 是 13 的倍数，所以 94146 能被 13 整除。

欧拉函数

```cpp
//sqrt 求单个欧拉函数
ll ph(ll x) {
    ll ret = x, tmp = x;
    for (ll i = 2; i * i <= x; i++) {
        if (tmp % i == 0) {
            ret = ret / i * (i - 1);
            while (tmp % i == 0) tmp /= i;
        }
    }
    if (tmp > 1) ret = ret / tmp * (tmp - 1);
    return ret;
}


//线性筛欧拉函数
int phi[N];
vector<int> prime;
bool isprime[N];
void init() {
    memset(isprime, 1, sizeof(isprime));
    phi[1] = 1, isprime[1] = false;
    for (int i = 2; i < N; ++i) {
        if (isprime[i]) {
            prime.push_back(i);
            phi[i] = i - 1;
        }
        for (int p : prime) {
            if (1ll * i * p >= N) break;
            int now = i * p;
            isprime[now] = false;
            if (i % p == 0) {
                phi[now] = phi[i] * p;
                break;
            } else {
                phi[now] = phi[i] * (p - 1);
            }
        }
    }
}
```

$$a^b \equiv \begin{cases} a^{b\%\varphi(p)} & (gcd(a,p)=1) \\ a^b & (gcd(a,p)\neq 1, b<\varphi(p)) \\ a^{b\%\varphi(p)+\varphi(p)} & (gcd(a,p)\neq 1, b\geq\varphi(p)) \end{cases}$$

Figure 1: 欧拉降幂

欧拉降幂

判断第二类斯特林数奇偶性

```cpp
#include <bits/stdc++.h>

using namespace std;
typedef long long ll;

int solve(ll n) {
    int ret = 0;
    while (n) {
        n >>= 1;
        ret += n;
    }
    return ret;
}

int main() {
    ll n, m;
    cin >> n >> m;
    ll a = n - m;
    ll b = (m - 1) / 2;
    ll ans1 = solve(a + b);
    ll ans2 = solve(a);
    ll ans3 = solve(b);
    if (ans1 == ans2 + ans3) {
        printf("1\n");
    } else {
        printf("0\n");
    }
    return 0;
}
```

三分

凸函数

整数

```cpp
//L, R 保证在凸函数两端
while (l + 1 < r) {
    int lm = (l + r) >> 1, rm = (lm + r) >> 1;
    if (calc(lm) > calc(rm))
        r = rm;
    else
        l = lm;
}
//答案取 L
```

double

```cpp
while (l + eps < r) {
    double lm = (l + r) / 2, rm = (lm + r) / 2;
    if (calc(lm) > calc(rm))
```

```
        r = rm;
    else
        l = lm;
}
//答案取 (L + R) / 2
```

凹函数

只需要将 check 时的符号互换。

线性基

```
//线性基 build
typedef long long ll;
const int maxn = 50;
ll d[maxn + 1];

bool add(ll x) {
    for (int i = maxn; i >= 0; --i) {
        if ((x >> i)) {
            if (d[i]) x ^= d[i];
            else {
                d[i] = x;
                return true;
            }
        }
    }
    return false;
}
```

exBSGS

```
/* exBSGS 算法
 * N ^ x = M (mod P), 其中 N, P 互质
 * 返回的 x 为最小解
 */
#include<bits/stdc++.h>

using namespace std;
typedef long long ll;
unordered_map<ll, int> H;

ll gcd(ll a, ll b) {
    if (!b) return a;
    return gcd(b, a % b);
}

ll expow(ll a, ll b, ll mod) {
    ll res = 1;
    while (b) res = ((b & 1) ? res * a % mod : res), a = a * a % mod, b >>= 1;
    return res;
}

ll exgcd(ll &x, ll &y, ll a, ll b) {
    if (!b) {
        x = 1, y = 0;
        return a;
    }
    ll t = exgcd(y, x, b, a % b);
    y -= x * (a / b);
    return t;
}

ll BSGS(ll a, ll b, ll mod, ll q) {
    H.clear();
    ll Q, p = ceil(sqrt(mod)), x, y;
    exgcd(x, y, q, mod), b = (b * x % mod + mod) % mod,
                         Q = expow(a, p, mod), exgcd(x, y, Q, mod), Q = (x % mod + mod) % mod;
    for (ll i = 1, j = 0; j <= p; ++j, i = i * a % mod) if (!H.count(i)) H[i] = j;
```

```
    for (ll i = b, j = 0; j <= p; ++j, i = i * Q % mod) if (H[i]) return j * p + H[i];
    return -1;
}

ll exBSGS(ll N, ll M, ll P) {
    ll q = 1;
    ll k = 0, ret;
    if (M == 1) return 0;
    while ((ret = gcd(N, P)) > 1) {
        if (M % ret) return -1;
        ++k, M /= ret, P /= ret, q = q * (N / ret) % P;
        if (q == M) return k;
    }
    return (ret = BSGS(N, M, P, q)) == -1 ? -1 : ret + k;
}

int main() {
    while (true) {
        int N, M, P;
        scanf("%d%d%d", &N, &P, &M);
        if (!N && !M && !P) break;
        N %= P, M %= P;
        int ans = exBSGS(N, M, P);
        if (ans == -1)
            printf("No Solution\n");
        else
            printf("%d\n", ans);
    }
}
```

exCRT

```
//x % A[i] = B[i] O(nlogn) x 为最小解 x + k * lcm 都可行
#include <bits/stdc++.h>

using namespace std;
typedef long long ll;
const int N = 1e5 + 10;

ll mul(ll a, ll b, ll mod) {
    ll ret = 0;
    while (b) {
        if (b & 1) ret = (ret + a) % mod;
        a = (a + a) % mod;
        b >>= 1;
    }
    return ret;
}

ll exgcd(ll a, ll b, ll &x, ll &y) {
    ll ret, tmp;
    if (!b) {
        x = 1;
        y = 0;
        return a;
    }
    ret = exgcd(b, a % b, x, y);
    tmp = x;
    x = y;
    y = tmp - a / b * y;
    return ret;
}

ll A[N], B[N];

ll excrt(int n) {
    ll x, y;
    ll M = A[1], ans = B[1];
    for (int i = 2; i <= n; ++i) {
        ll a = M, b = A[i], c = (B[i] - ans % b + b) % b;
        ll g = exgcd(a, b, x, y), bg = b / g;
```

```
        if (c % g) return -1;
        x = mul(x, c / g, bg); //可能溢出
        ans += x * M;
        M *= bg;
        ans = (ans % M + M) % M;
    }
    return (ans % M + M) % M;
}

int main() {
    int n;
    cin >> n;
    for (int i = 1; i <= n; ++i) {
        cin >> A[i] >> B[i];
    }
    cout << excrt(n);
    return 0;
}
```

Miller-Rabin & Pollard-Rho

```
#include <bits/stdc++.h>

using namespace std;
typedef long long ll;

const int Times = 10;
const int N = 5500;

ll ct;
ll fac[N];

ll gcd(ll a, ll b) {
    return b ? gcd(b, a % b) : a;
}

ll multi(ll a, ll b, ll m) {
    ll ret = 0;
    a %= m;
    while (b) {
        if (b & 1) {
            ret = (ret + a) % m;
        }
        b >>= 1;
        a = (a + a) % m;
    }
    return ret;
}

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

bool Miller_Rabin(ll n) {
    if (n == 2) return true;
    if (n < 2 || !(n & 1)) return false;
    ll m = n - 1;
    int k = 0;
    while ((m & 1) == 0) {
        k++;
        m >>= 1;
    }
```

```
    for (int i = 0; i < Times; ++i) {
        ll a = rand() % (n - 1) + 1;
        ll x = qpow(a, m, n);
        ll y = 0;
        for (int j = 0; j < k; ++j) {
            y = multi(x, x, n);
            if (y == 1 && x != 1 && x != n - 1) return false;
            x = y;
        }
        if (y != 1) return false;
    }
    return true;
}

ll pollard_rho(ll n, ll c) {
    ll i = 1, k = 2;
    ll x = rand() % (n - 1) + 1;
    ll y = x;
    while (true) {
        i++;
        x = (multi(x, x, n) + c) % n;
        ll d = gcd((y - x + n) % n, n);
        if (1 < d && d < n) return d;
        if (y == x) return n;
        if (i == k) {
            y = x;
            k <<= 1;
        }
    }
}

void find(ll n, int c) {
    if (n == 1) return;
    if (Miller_Rabin(n)) {
        fac[ct++] = n;
        return;
    }
    ll p = n;
    ll k = c;
    while (p >= n) p = pollard_rho(p, c--);
    find(p, k);
    find(n / p, k);
}

int main() {
    ll n;
    cin >> n;
    find(n, 120);
    sort(fac, fac + ct);
    //排好序的所有质因子 例如 60 被拆解为 2 2 3 5
}
```

FFT

```
#include <bits/stdc++.h>

using namespace std;
typedef long long ll;
const int N = (1 << 21) + 10;
const double PI = acos(-1.0);

struct Complex {
    double x, y;

    Complex(double _x = 0.0, double _y = 0.0) {
        x = _x;
        y = _y;
    }

    Complex operator-(const Complex &b) const {
        return Complex(x - b.x, y - b.y);
```

```
    }

    Complex operator+(const Complex &b) const {
        return Complex(x + b.x, y + b.y);
    }

    Complex operator*(const Complex &b) const {
        return Complex(x * b.x - y * b.y, x * b.y + y * b.x);
    }
};

int rev[N];


void change(Complex x[], int len) {
    for (int i = 0; i < len; ++i) {
        if (i < rev[i]) {
            swap(x[i], x[rev[i]]);
        }
    }
}

void fft(Complex x[], int len, int opt) {
    change(x, len);
    for (int h = 2; h <= len; h <<= 1) {
        Complex wn(cos(2 * PI / h), sin(opt * 2 * PI / h));
        for (int j = 0; j < len; j += h) {
            Complex w(1, 0);
            for (int k = j; k < j + h / 2; k++) {
                Complex u = x[k];
                Complex t = w * x[k + h / 2];
                x[k] = u + t;
                x[k + h / 2] = u - t;
                w = w * wn;
            }
        }
    }
    if (opt == -1) {
        for (int i = 0; i < len; i++) {
            x[i].x /= len;
        }
    }
}

Complex A[N], B[N], C[N];

ll d[N];

int main() {
    string s, t;
    cin >> s >> t;
    reverse(s.begin(), s.end());
    reverse(t.begin(), t.end());
    int n = (int) s.length(), m = (int) t.length();
    for (int i = 0; i < n; ++i) A[i].x = s[i] - '0';
    for (int i = 0; i < m; ++i) B[i].x = t[i] - '0';
    int len = 1;
    while (len < (n << 1)) len <<= 1;
    while (len < (m << 1)) len <<= 1;
    for (int i = 0; i < len; ++i) {
        rev[i] = rev[i >> 1] >> 1;
        if (i & 1) rev[i] |= len >> 1;
    }
    fft(A, len, 1);
    fft(B, len, 1);
    for (int i = 0; i < len; ++i) C[i] = A[i] * B[i];
    fft(C, len, -1);
    for (int i = 0; i < len; ++i) d[i] = round(C[i].x);
    for (int i = 0; i < len; ++i) {
        d[i + 1] += d[i] / 10;
        d[i] %= 10;
```

```
    }
    string out;
    for (int i = len - 1, f = 0; i >= 0; --i) {
        if (d[i]) f = 1;
        if (f) out += (char) ('0' + d[i]);
    }
    cout << out << '\n';
    return 0;
}
```

NTT

```
#include <bits/stdc++.h>

using namespace std;
typedef long long ll;

const int N = (1 << 21) + 10;
const int mod = 998244353;

ll qpow(ll x, ll y) {
    ll ret = 1;
    x %= mod;
    while (y) {
        if (y & 1) ret = ret * x % mod;
        x = x * x % mod;
        y >>= 1;
    }
    return ret;
}

int r[N];

//opt == -1, point to num. otherwise num to point.
void ntt(ll *x, int lim, int opt) {
    for (int i = 0; i < lim; ++i)
        if (r[i] < i) swap(x[i], x[r[i]]);
    for (int m = 2; m <= lim; m <<= 1) {
        int k = m >> 1;
        ll gn = qpow(3, (mod - 1) / m);
        for (int i = 0; i < lim; i += m) {
            ll g = 1;
            for (int j = 0; j < k; ++j, g = g * gn % mod) {
                ll tmp = x[i + j + k] * g % mod;
                x[i + j + k] = (x[i + j] - tmp + mod) % mod;
                x[i + j] = (x[i + j] + tmp) % mod;
            }
        }
    }
    if (opt == -1) {
        reverse(x + 1, x + lim);
        ll inv = qpow(lim, mod - 2);
        for (int i = 0; i < lim; ++i) x[i] = x[i] * inv % mod;
    }
}

ll A[N], B[N], C[N];

int main() {
    string s, t;
    cin >> s >> t;
    int n, m;
    n = (int) s.length();
    m = (int) t.length();
    reverse(s.begin(), s.end());
    reverse(t.begin(), t.end());
    for (int i = 0; i < n; ++i) A[i] = s[i] - '0';
    for (int i = 0; i < m; ++i) B[i] = t[i] - '0';
    int lim = 1;
    while (lim < (n << 1)) lim <<= 1;
    while (lim < (m << 1)) lim <<= 1;
```

```
    for (int i = 0; i < lim; ++i) r[i] = (i & 1) * (lim >> 1) + (r[i >> 1] >> 1);
    ntt(A, lim, 1);
    ntt(B, lim, 1);
    for (int i = 0; i < lim; ++i) C[i] = A[i] * B[i] % mod;
    ntt(C, lim, -1);
    for (int i = 0; i < lim; ++i) {
        C[i + 1] += C[i] / 10;
        C[i] %= 10;
    }
    string out;
    for (int i = lim - 1, f = 0; i >= 0; --i) {
        if (C[i]) f = 1;
        if (f) out += (char) ('0' + C[i]);
    }
    cout << out << '\n';
    return 0;
}
```

图论

点分治

```
//点分治
#include <bits/stdc++.h>

using namespace std;
typedef long long ll;
const int maxn = 20005;
const int inf = 0x3f3f3f3f;
const int mod = 1e9 + 7;

struct edge {
    int to, val;
};
vector<edge> mp[maxn];

int mini, rt, totSZ;
int sz[maxn], dis[maxn];
bool vis[maxn];

int l, r, q[maxn];  //q 为每次得到的距离合集

int n, ans; //ans 记录合法点对

void getRT(int u, int pre) {    //每次调用 getRT() 前使 mini=inf, totSZ = sz[v];
    sz[u] = 1;
    int mxSub = 0;
    for (auto it: mp[u]) {
        int v = it.to;
        if (v == pre || vis[v]) continue;
        getRT(v, u);
        sz[u] += sz[v];
        mxSub = max(mxSub, sz[v]);
    }
    int mx = max(mxSub, totSZ - sz[u]);
    if (mx < mini) {
        mini = mx;
        rt = u;
    }
}

void getDIS(int u, int pre) {
    q[++r] = dis[u];
    for (auto it: mp[u]) {
        int v = it.to, val = it.val;
        if (v == pre || vis[v]) continue;
        dis[v] = dis[u] + val;
        getDIS(v, u);
    }
}
```

```

int calc(int u, int val) {
    l = 1, r = 0;
    dis[u] = val;
    getDIS(u, 0);
    //按照题意处理 q
    return sum;
}

void dfs(int u) {
    vis[u] = true;
    ans += calc(u, 0);
    for (auto it: mp[u]) {
        int v = it.to, val = it.val;
        if (vis[v]) continue;
        ans -= calc(v, val);
        mini = inf;
        totSZ = sz[v];
        getRT(v, 0);
        dfs(rt);
    }
}

int main() {
    while (~scanf("%d", &n)) {
        for (int i = 1; i <= n; i++) {
            mp[i].clear();
            vis[i] = false;
        }
        ans = 0;
        for (int i = 1; i < n; i++) {
            int u, v, val;
            scanf("%d%d%d", &u, &v, &val);
            mp[u].push_back(edge{v, val});
            mp[v].push_back(edge{u, val});
        }

        mini = inf;
        totSZ = n;
        getRT(1, 0);
        dfs(rt);
        printf("%d\n", ans);
    }
    return 0;
}
```

轻重链剖分

```
// Problem: P3384 【模板】轻重链剖分
// Contest: Luogu
// URL: https://www.luogu.com.cn/problem/P3384
// Memory Limit: 125 MB
// Time Limit: 1000 ms
// 已知一棵包含 N 个结点的树（连通且无环），每个节点上包含一个数值，需要支持以下操作：
// 操作 1：格式：  1 x y z 表示将树从 x 到 y 结点最短路径上所有节点的值都加上 z
// 操作 2：格式：  2 x y 表示求树从 x 到 y 结点最短路径上所有节点的值之和
// 操作 3：格式：  3 x z 表示将以 x 为根节点的子树内所有节点值都加上 z
// 操作 4：格式：  4 x 表示求以 x 为根节点的子树内所有节点值之和
// Powered by CP Editor (https://cpeditor.org)

#include <bits/stdc++.h>

using namespace std;
typedef long long ll;
const int N = 1e5 + 10;

int n, q, rt;
int dp[N], fa[N], son[N], ord[N], dep[N], a[N], top[N];
ll val[N], mod;
vector<int> G[N];

```

```
ll T[N << 2];
ll lazy[N << 2];

void push_down(int l, int r, int id) {
    if (lazy[id] == 0) return;
    int m = (l + r) >> 1;
    T[id << 1] += (m - l + 1) * lazy[id] % mod;
    T[id << 1] %= mod;
    T[id << 1 | 1] += (r - m) * lazy[id] % mod;
    T[id << 1 | 1] %= mod;
    lazy[id << 1] += lazy[id];
    lazy[id << 1] %= mod;
    lazy[id << 1 | 1] += lazy[id];
    lazy[id << 1 | 1] %= mod;
    lazy[id] = 0;
}

void push_up(int id) {
    T[id] = T[id << 1] + T[id << 1 | 1];
    T[id] %= mod;
}

void update(int ql, int qr, int l, int r, ll k, int id) {
    if (ql <= l && r <= qr) {
        lazy[id] += k;
        lazy[id] %= mod;
        T[id] += (r - l + 1) * k % mod;
        T[id] %= mod;
        return;
    }
    push_down(l, r, id);
    int m = (l + r) >> 1;
    if (ql <= m) update(ql, qr, l, m, k, id << 1);
    if (qr > m) update(ql, qr, m + 1, r, k, id << 1 | 1);
    push_up(id);
}

ll query(int ql, int qr, int l, int r, int id) {
    if (ql <= l && r <= qr) {
        return T[id] % mod;
    }
    push_down(l, r, id);
    ll ret = 0;
    int m = (l + r) >> 1;
    if (ql <= m) ret += query(ql, qr, l, m, id << 1) % mod, ret %= mod;
    if (qr > m) ret += query(ql, qr, m + 1, r, id << 1 | 1) % mod, ret %= mod;
    return ret;
}

void build(int l, int r, int id) {
    if (l == r) {
        T[id] = val[l] % mod;
        return;
    }
    int m = (l + r) >> 1;
    build(l, m, id << 1);
    build(m + 1, r, id << 1 | 1);
    push_up(id);
}

void init(int now, int father, int depth) {
    fa[now] = father;
    dp[now] = 1;
    dep[now] = depth;
    int mx = 0;
    for (int i : G[now]) {
        if (i == father) continue;
        init(i, now, depth + 1);
        dp[now] += dp[i];
        mx = max(mx, dp[i]);
    }
```

```
    for (int i : G[now]) {
        if (i == father) continue;
        if (dp[i] == mx) {
            son[now] = i;
            break;
        }
    }
}

int cnt;

void dfs(int now, int father, int st) {
    top[now] = st;
    ord[now] = ++cnt;
    val[ord[now]] = a[now];
    if (son[now]) dfs(son[now], now, st);
    for (int i : G[now]) {
        if (i == father) continue;
        if (i == son[now]) continue;
        dfs(i, now, i);
    }
}

int main() {
    ios_base::sync_with_stdio(false);
    cin >> n >> q >> rt >> mod;
    for (int i = 1; i <= n; ++i) cin >> a[i];
    for (int i = 1; i < n; ++i) {
        int u, v;
        cin >> u >> v;
        G[u].push_back(v);
        G[v].push_back(u);
    }
    init(rt, -1, 1);
    dfs(rt, -1, rt);
    build(1, n, 1);
    while (q--) {
        int op;
        cin >> op;
        if (op == 1) {
            int x, y, z;
            cin >> x >> y >> z;
            z %= mod;
            while (top[y] != top[x]) {
                if (dep[top[x]] > dep[top[y]]) swap(x, y);
                update(ord[top[y]], ord[y], 1, n, z, 1);
                y = fa[top[y]];
            }
            if (dep[x] > dep[y]) swap(x, y);
            update(ord[x], ord[y], 1, n, z, 1);
        } else if (op == 2) {
            int x, y;
            cin >> x >> y;
            ll ans = 0;
            while (top[y] != top[x]) {
                if (dep[top[x]] > dep[top[y]]) swap(x, y);
                ans += query(ord[top[y]], ord[y], 1, n, 1) % mod;
                ans %= mod;
                y = fa[top[y]];
            }
            if (dep[x] > dep[y]) swap(x, y);
            ans += query(ord[x], ord[y], 1, n, 1);
            ans %= mod;
            cout << ans << '\n';
        } else if (op == 3) {
            int x, z;
            cin >> x >> z;
            z %= mod;
            update(ord[x], ord[x] + dp[x] - 1, 1, n, z, 1);
        } else {
            int x;
```

```
            cin >> x;
            ll ans = query(ord[x], ord[x] + dp[x] - 1, 1, n, 1) % mod;
            cout << ans << '\n';
        }
    }
    return 0;
}
```

虚树

```
// Problem: P2495 [SDOI2011] 消耗战
// 1 为根，每次可以炸毁一个边，花费为边权。
// m 次询问，每次要求让 k 个关键点无法到达 1，问最小花费
// 对于每次询问单独建树，dfs 序 + 单调栈优化

#include <bits/stdc++.h>

using namespace std;
typedef long long ll;
const int N = 3e5 + 10;
vector<pair<int, ll>> G[N];
vector<int> g[N];
int ord[N]; //dfs 序
int stk[N];
int dep[N];
int parent[N][30];
bool inq[N]; //是否为关键点
ll minV[N]; //从 1 到 i 不联通的最小花费
ll dp[N]; //ans
int now;
int maxx = 0;

void dfs1(int u, int fa) {
    ++now;
    ord[u] = now;
    dep[u] = dep[fa] + 1;
    maxx = max(maxx, dep[u]);
    parent[u][0] = fa;
    for (auto pii : G[u]) {
        int v = pii.first;
        ll w = pii.second;
        if (v == fa) continue;
        minV[v] = min(minV[u], w);
        dfs1(v, u);
    }
}

void dfs2(int u) {
    ll sum = 0;
    for (int v : g[u]) {
        dfs2(v);
        sum += dp[v];
    }
    if (inq[u]) {
        dp[u] = minV[u];
    } else {
        dp[u] = min(minV[u], sum);
    }
    inq[u] = false;
    g[u].clear();
}

int n;

void init() {
    memset(minV, 0x3f, sizeof minV);
    dfs1(1, 0);
    for (int i = 1; (1 << i) <= n; ++i) {
        for (int j = 1; j <= n; ++j) {
            if (parent[j][i - 1] == 0) parent[j][i] = 0;
            else parent[j][i] = parent[parent[j][i - 1]][i - 1];
```

```
        }
    }
}

int LCA(int ta, int tb) {
    if (dep[ta] < dep[tb]) swap(ta, tb);
    for (int i = 0; dep[ta] != dep[tb]; ++i) {
        if ((dep[ta] - dep[tb]) >> i & 1)
            ta = parent[ta][i];
    }
    if (ta == tb) return ta;
    for (int i = log2(maxx); i >= 0; --i) {
        if (parent[ta][i] != parent[tb][i]) {
            ta = parent[ta][i];
            tb = parent[tb][i];
        }
    }
    return parent[ta][0];
}

void build() {
    vector<pair<int, int>> v;
    int k;
    cin >> k;
    for (int i = 0; i < k; ++i) {
        int id;
        cin >> id;
        inq[id] = true;
        v.emplace_back(ord[id], id);
    }
    sort(v.begin(), v.end());
    now = 0;
    stk[++now] = 1;
    g[1].clear();
    for (int i = 0; i < k; ++i) {
        int id = v[i].second;
        if (id != 1) {
            int lca = LCA(id, stk[now]);
            if (lca != stk[now]) {
                while (ord[lca] < ord[stk[now - 1]]) {
                    g[stk[now - 1]].push_back(stk[now]);
                    now--;
                }
                if (ord[lca] > ord[stk[now - 1]]) {
                    g[lca].clear();
                    g[lca].push_back(stk[now]);
                    now--;
                    stk[++now] = lca;
                } else {
                    g[lca].push_back(stk[now]);
                    now--;
                }
            }
        }
        g[id].clear();
        stk[++now] = id;
    }
    for (int i = 1; i < now; ++i) {
        g[stk[i]].push_back(stk[i + 1]);
    }
}

void solve() {
    cin >> n;
    for (int i = 1; i < n; ++i) {
        int u, v, w;
        cin >> u >> v >> w;
        G[u].emplace_back(v, w);
        G[v].emplace_back(u, w);
    }
    init();
```

```
    int m;
    cin >> m;
    while (m--) {
        build();
        dfs2(stk[1]);
        printf("%lld\n", dp[stk[1]]);
    }
}

int main() {
    ios_base::sync_with_stdio(false);
    int T = 1;
    // cin >> T;
    while (T--) {
        solve();
    }
    return 0;
}
```

网络流

感觉不如 awei template 。。。画质

Tarjan

```
const int N = 1e5 + 10;
vector<int> G[N];
int dfn[N], low[N];//dfn[u] -> u 被搜索的次序 low[u] -> u 子树中 dfn 最小值 (包括自身)
int index;//搜索序号
stack<int> S;
bool ins[N];//是否进栈
int col[N], num_color;//染色

void Tarjan(int u) {
    dfn[u] = low[u] = ++index;
    S.push(u);//进栈
    ins[u] = true;
    for (int i: G[u]) {
        int v = i;
        if (!dfn[v]) { //未被访问过
            Tarjan(v);
            low[u] = min(low[u], low[v]); //找爸爸（环开头）最小的
        } else if (ins[v]) { //已被访问过且在栈内，则需要处理；若不在栈内说明对应强连通分量处理完成
            low[u] = min(low[u], dfn[v]); //判断谁是爸爸
        }
    }
    if (dfn[u] == low[u]) { //发现更新完一轮自己是爸爸（某强连通分量中仅第一个被访问的结点满足 dfn[u] == low[u])
        num_color++;
        int tmp;
        do {
            tmp = S.top();
            col[tmp] = num_color; //出栈，染色
            ins[tmp] = false;
            S.pop();
        } while (tmp != u);
        col[u] = num_color;
        ins[u] = false;
    }
}
```

倍增 LCA

```
vector<int> G[N];
int st[N][20];
int dep[N];
int maxx = -1;

void dfs(int u, int fa = -1) {
    if (fa == -1) dep[u] = 0;
```

```
    else dep[u] = dep[fa] + 1;
    st[u][0] = fa;
    for (int v: G[u]) {
        if (v == fa) continue;
        dfs(v, u);
    }
}

void init(int n) {
    for (int i = 1; i <= n; ++i) memset(st[i], -1, sizeof st[i]);
    dfs(rt);
    for (int j = 1; j < 20; ++j) {
        for (int i = 1; i <= n; ++i) {
            if (st[i][j - 1] != -1) st[i][j] = st[st[i][j - 1]][j - 1];
        }
    }
}

int lca(int u, int v) {
    if (dep[u] < dep[v]) swap(u, v);
    for (int i = 0; dep[u] != dep[v]; ++i) {
        if ((dep[u] - dep[v]) >> i & 1) {
            u = st[u][i];
        }
    }
    if (u == v) return u;
    for (int i = 19; i >= 0; --i) {
        if (st[u][i] != st[v][i]) {
            u = st[u][i];
            v = st[v][i];
        }
    }
    return st[u][0];
}
```

dsu on tree

1. 先遍历轻儿子，并计算答案，但 **不保留遍历后它对答案的影响**；
2. 访问重儿子，**保留它对答案的影响**。
3. 再次遍历轻儿子，合并轻重儿子之间的答案。

```
void dfs(int u, int fa, bool keep) {
    // 计算轻儿子的答案
    for (int v: G[u])
        if (v != fa && v != big[u]) {
            dfs(v, u, false);
        }
    // 计算重儿子答案并保留计算过程中的数据（用于继承）
    if (big[u]) {
        dfs(big[u], u, true);
    }
    for (int v: G[u])
        if (v != fa && v != big[u]) {
            // 子树结点的 DFS 序构成一段连续区间，可以直接遍历
            for (int i = L[v]; i <= R[v]; i++) {
                add(Node[i]);
            }
        }
    add(u);
    ans[u] = getAns();
    if (!keep) {
        for (int i = L[u]; i <= R[u]; i++) {
            del(Node[i]);
        }
    }
}
```

几何板子

pick 定理

```cpp
//计算凸多边形格点数
#include <iostream>
#include <algorithm>

using namespace std;

typedef long long ll;

const int maxn = 1e5;

ll x[maxn + 1] = {};
ll y[maxn + 1] = {};

ll calc(int a, int b) {
    if (y[a] == y[b]) return abs(x[a] - x[b]) - 1;
    if (x[a] == x[b]) return abs(y[a] - y[b]) - 1;
    return __gcd(abs(y[a] - y[b]), abs(x[a] - x[b])) - 1;
}

int main() {
    int n;
    scanf("%d", &n);
    for (int i = 0; i < n; i++) scanf("%lld%lld", &x[i], &y[i]);
    for (int i = 1; i < n; i++) {
        x[i] -= x[0];
        y[i] -= y[0];
    }
    x[n] = y[n] = x[0] = y[0] = 0;
    //2c=2s-a-b+2
    ll ans = 0;
    ll sum = 0;
    for (int i = 1; i <= n; i++) {
        sum += x[i] * y[i + 1] - y[i] * x[i + 1];
    }
    ll b = 0;
    for (int i = 0; i < n; i++) {
        b += calc(i, i + 1);
    }
    cout << (abs(sum) - n - b + 2) / 2;
    return 0;
}
```

其他的。。。感觉不如 kuangbin hdu

字符串

AC 自动机

```cpp
// Problem: P5357 【模板】AC 自动机（二次加强版）
// Contest: Luogu
// URL: https://www.luogu.com.cn/problem/P5357
// Memory Limit: 256 MB
// Time Limit: 1000 ms
//
// Powered by CP Editor (https://cpeditor.org)

#include <bits/stdc++.h>

using namespace std;

#define dbg(x...) \
    do { \
        cout << #x << " -> "; \
        err(x); \
    } while (0)
```

```cpp

void err() {
    cout << endl;
}

template<class T, class... Ts>
void err(T arg, Ts &... args) {
    cout << arg << ' ';
    err(args...);
}

typedef long long ll;
typedef unsigned long long ull;
typedef long double ld;
typedef __int128 i128;
const int N = 2e5 + 10;
const ll mod = 1e9 + 7;

mt19937 mt(chrono::steady_clock::now().time_since_epoch().count());

ll rng(ll l, ll r) {
    uniform_int_distribution<ll> uni(l, r);
    return uni(mt);
}

struct AC {
    int t[N][26], tot, cnt;
    int ed[N], fail[N];
    vector<int> G[N];
    vector<int> top;

    void init() {
        memset(ed, -1, sizeof ed);
    }


    void insert(const string &s) {
        int u = 0;
        for (char ch: s) {
            int to = ch - 'a';
            if (!t[u][to]) {
                t[u][to] = ++tot;
            }
            u = t[u][to];
        }
        ed[cnt++] = u;
    }

    void build() {
        queue<int> q;
        for (int i = 0; i < 26; ++i) {
            if (t[0][i]) q.push(t[0][i]);
        }
        while (!q.empty()) {
            int u = q.front();
            q.pop();
            for (int i = 0; i < 26; ++i) {
                if (t[u][i]) {
                    fail[t[u][i]] = t[fail[u]][i];
                    q.push(t[u][i]);
                } else {
                    t[u][i] = t[fail[u]][i];
                }
            }
        }
        for (int i = 1; i <= tot; ++i) {
            G[fail[i]].push_back(i);
        }
        q.push(0);
        while (!q.empty()) {
            int u = q.front();
```

```
            q.pop();
            top.push_back(u);
            for (int v: G[u]) {
                q.push(v);
            }
        }
        reverse(top.begin(), top.end());
    }

    void query(const string &s) {
        int u = 0;
        vector<int> c(tot + 1, 0);
        vector<int> d(tot + 1, 0);
        for (char ch: s) {
            int to = ch - 'a';
            u = t[u][to];
            d[u]++;
        }
        for (int i: top) {
            c[i] += d[i];
            d[fail[i]] += d[i];
        }
        for (int i = 0; i < cnt; ++i) {
            cout << c[ed[i]] << '\n';
        }
    }
} ac;

void solve(int tCase) {
    int n;
    cin >> n;
    ac.init();
    for (int i = 0; i < n; ++i) {
        string s;
        cin >> s;
        ac.insert(s);
    }
    ac.build();
    string t;
    cin >> t;
    ac.query(t);
}

int main() {
    ios_base::sync_with_stdio(false);
    int T = 1;
//    cin >> T;
    for (int t = 1; t <= T; ++t) {
        solve(t);
    }
    return 0;
}
```

Z 函数（扩展 KMP）

```
//z[i] 为 s 本身和 s.substr(i) 的最长公共前缀长度, 但是 z[0] = 0
vector<int> z_function(string s) {
    int n = (int) s.length();
    vector<int> z(n);
    for (int i = 1, l = 0, r = 0; i < n; ++i) {
        if (i <= r && z[i - l] < r - i + 1) {
            z[i] = z[i - l];
        } else {
            z[i] = max(0, r - i + 1);
            while (i + z[i] < n && s[z[i]] == s[i + z[i]]) ++z[i];
        }
        if (i + z[i] - 1 > r) l = i, r = i + z[i] - 1;
    }
    return z;
}
```

马拉车

```cpp
//Manacher
string Manacher(string s)
{
    /* 改造字符串 */
    string res="$#";
    for(int i=0;i<s.size();++i)
    {
        res+=s[i];
        res+="#";
    }

    /* 数组 */
    vector<int> P(res.size(),0);
    int mi=0,right=0;   //mi 为最大回文串对应的中心点，right 为该回文串能达到的最右端的值
    int maxLen=0,maxPoint=0;    //maxLen 为最大回文串的长度，maxPoint 为记录中心点

    for(int i=1;i<res.size();++i)
    {
        P[i]=right>i ?min(P[2*mi-i],right-i):1;     //关键句，文中对这句以详细讲解

        while(res[i+P[i]]==res[i-P[i]])
            ++P[i];

        if(right<i+P[i])    //超过之前的最右端，则改变中心点和对应的最右端
        {
            right=i+P[i];
            mi=i;
        }

        if(maxLen<P[i])     //更新最大回文串的长度，并记下此时的点
        {
            maxLen=P[i];
            maxPoint=i;
        }
    }
    return s.substr((maxPoint-maxLen)/2,maxLen-1);
}
```

双哈希方便写法

```cpp
const pii mod = {1e9 + 7, 1e9 + 9};

const pii base = {131, 251};

pll pw[maxn * 2];

pll operator * (const pll &p1, const pll &p2) {
    return {p1.first * p2.first % mod.first, p1.second * p2.second % mod.second};
}

pll operator + (const pll &p1, const pll &p2) {
    return {(p1.first + p2.first) % mod.first, (p1.second + p2.second) % mod.second};
}

pll operator - (const pll &p1, const pll &p2) {
    return {(p1.first - p2.first + mod.first) % mod.first, (p1.second - p2.second + mod.second) % mod.second};
}

struct Hash {
    string s;
    vector<pll> f;
    int n;
    void init(char ss[]) {
        s = " ";
        s += string(ss);
        n = (int) s.length() - 1;
        f.resize(n + 1, {0, 0});
        for (int i = 1; i <= n; i++) {
            int ch = s[i] - 'a';
```

```
            f[i] = f[i - 1] * base + pll{ch, ch};
        }
    }
    pll ask(int l, int r) {//[l + 1, r]
        return f[r] - f[l] * pw[r - l];
    }
} s, t[10007];
```

杂项

日期

```
// Routines for performing computations on dates.  In these routines,
// months are exprsesed as integers from 1 to 12, days are expressed
// as integers from 1 to 31, and years are expressed as 4-digit
// integers.

string dayOfWeek[] = {"Mo", "Tu", "We", "Th", "Fr", "Sa", "Su"};

// converts Gregorian date to integer (Julian day number)

int DateToInt (int m, int d, int y){
  return
    1461 * (y + 4800 + (m - 14) / 12) / 4 +
    367 * (m - 2 - (m - 14) / 12 * 12) / 12 -
    3 * ((y + 4900 + (m - 14) / 12) / 100) / 4 +
    d - 32075;
}

// converts integer (Julian day number) to Gregorian date: month/day/year

void IntToDate (int jd, int &m, int &d, int &y){
  int x, n, i, j;

  x = jd + 68569;
  n = 4 * x / 146097;
  x -= (146097 * n + 3) / 4;
  i = (4000 * (x + 1)) / 1461001;
  x -= 1461 * i / 4 - 31;
  j = 80 * x / 2447;
  d = x - 2447 * j / 80;
  x = j / 11;
  m = j + 2 - 12 * x;
  y = 100 * (n - 49) + i + x;
}

// converts integer (Julian day number) to day of week

string IntToDay (int jd){
  return dayOfWeek[jd % 7];
}
```

子集枚举

- 枚举真子集

```
for (int s = (S - 1) & S; s; s = (s - 1) & S)
```

- 枚举大小为 k 的子集

```
template<typename T>
void subset(int k, int n, T&& f) {
    int t = (1 << k) - 1;
    while (t < 1 << n) {
        f(t);
        int x = t & -t, y = t + x;
        t = ((t & ~y) / x >> 1) | y;
    }
}
```

数位 DP

```
LL dfs(LL base, LL pos, LL len, LL s, bool limit) {
    if (pos == -1) return s ? base : 1;
    if (!limit && dp[base][pos][len][s] != -1) return dp[base][pos][len][s];
    LL ret = 0;
    LL ed = limit ? a[pos] : base - 1;
    FOR (i, 0, ed + 1) {
        tmp[pos] = i;
        if (len == pos)
            ret += dfs(base, pos - 1, len - (i == 0), s, limit && i == a[pos]);
        else if (s &&pos < (len + 1) / 2)
            ret += dfs(base, pos - 1, len, tmp[len - pos] == i, limit && i == a[pos]);
        else
            ret += dfs(base, pos - 1, len, s, limit && i == a[pos]);
    }
    if (!limit) dp[base][pos][len][s] = ret;
    return ret;
}

LL solve(LL x, LL base) {
    LL sz = 0;
    while (x) {
        a[sz++] = x % base;
        x /= base;
    }
    return dfs(base, sz - 1, sz - 1, 1, true);
}
```

背包

01 背包

有 N 件物品和一个容量为 V 的背包。第 i 件物品的费用是 c[i]，价值是 w[i]。求解将哪些物品装入背包可使价值总和最大。

```
for i=1..N
    for v=V..0
        f[v]=max{f[v],f[v-c[i]]+w[i]};
```

完全背包问题

有 N 种物品和一个容量为 V 的背包，每种物品都有无限件可用。第 i 种物品的费用是 c[i]，价值是 w[i]。求解将哪些物品装入背包可使这些物品的费用总和不超过背包容量，且价值总和最大。

```
for i=1..N
    for v=0..V
        f[v]=max{f[v],f[v-cost]+weight}
```

多重背包问题

有 N 种物品和一个容量为 V 的背包。第 i 种物品最多有 n[i] 件可用，每件费用是 c[i]，价值是 w[i]。求解将哪些物品装入背包可使这些物品的费用总和不超过背包容量，且价值总和最大。

$f[i][v] = max(f[i-1][v-k*c[i]] + k*w[i] | 0 <= k <= n[i])$

```
procedure MultiplePack(cost,weight,amount)
    if cost*amount>=V
        CompletePack(cost,weight)
        return
    integer k=1
    while k<amount
        ZeroOnePack(k*cost,k*weight)
        amount=amount-k
        k=k*2
    ZeroOnePack(amount*cost,amount*weight)
```

暴力 $O(V\sum n_i)$

单调队列可以优化到 $O(VN)$

混合三种背包问题

有的物品只可以取一次（01 背包），有的物品可以取无限次（完全背包），有的物品可以取的次数有一个上限（多重背包）

01 背包与完全背包的混合

如果只有两类物品：一类物品只能取一次，另一类物品可以取无限次，那么只需在对每个物品应用转移方程时，根据物品的类别选用顺序或逆序的循环即可，复杂度是 O(VN)

```
for i=1..N
    if 第i 件物品属于 01 背包
        for v=V..0
            f[v]=max{f[v],f[v-c[i]]+w[i]};
    else if 第i 件物品属于完全背包
        for v=0..V
            f[v]=max{f[v],f[v-c[i]]+w[i]};
```

加上多重背包

```
for i=1..N
    if 第i 件物品属于 01 背包
        ZeroOnePack(c[i],w[i])
    else if 第i 件物品属于完全背包
        CompletePack(c[i],w[i])
    else if 第i 件物品属于多重背包
        MultiplePack(c[i],w[i],n[i])
```

可以用单调队列优化

二维费用的背包问题

对于每件物品，具有两种不同的费用；选择这件物品必须同时付出这两种代价；对于每种代价都有一个可付出的最大值（背包容量）。问怎样选择物品可以得到最大的价值。设这两种代价分别为代价 1 和代价 2，第 i 件物品所需的两种代价分别为 a[i] 和 b[i]。两种代价可付出的最大值（两种背包容量）分别为 V 和 U。物品的价值为 w[i]。

费用加了一维，只需状态也加一维即可。设 f[i][v][u] 表示前 i 件物品付出两种代价分别为 v 和 u 时可获得的最大价值。状态转移方程就是：

$$f[i][v][u] = max(f[i-1][v][u], f[i-1][v-a[i]][u-b[i]] + w[i])$$

可以滚动优化

分组背包

有 N 件物品和一个容量为 V 的背包。第 i 件物品的费用是 c[i]，价值是 w[i]。这些物品被划分为若干组，每组中的物品互相冲突，最多选一件。求解将哪些物品装入背包可使这些物品的费用总和不超过背包容量，且价值总和最大。

```
for 所有的组k
    for v=V..0
        for 所有的i 属于组 k
            f[v]=max{f[v],f[v-c[i]]+w[i]}
```