Platelet

Team Reference Material

凌皓煜 陈彤 顾逸

2018

Contents

Chapter 1

Graph Theory

1.1 2-SAT

1.2 双连通分量

1.2.1 点双连通分量

1.2.2 边双连通分量

1.3 K 短路

1.4 最大团

1.5 一般图最大匹配

1.6 树

1.6.1 虚树

1.6.2 矩阵树定理

1.6.3 点分治

1.6.4 Prufer 编码

1.6.5 Link-Cut Tree

1.6.6 树上倍增

1.6.7 数链剖分

1.7 仙人掌

1.8 带花树

1.9 KM 算法

1.10 支配树

1.10.1 DAG

1.10.2 一般图

1.11 弦图

1.12 网络流

1.12.1 最小割

1.12.2 最大流

Chapter 2

Math

2.1 int64 相乘取模 (Durandal)

```cpp
int64_t mul(int64_t x, int64_t y, int64_t p) {
    int64_t t = (x * y - (int64_t) ((long double) x / p * y + 1e-3) * p) % p;
    return t < 0 ? t + p : t;
}
```

2.2 扩展欧几里得 (gy)

```cpp
// return gcd(a, b)
// ax+by=gcd(a,b)
int extend_gcd(int a, int b, int &x, int &y) {
    if (b == 0) {
        x = 1, y = 0;
        return a;
    }
    int res = extend_gcd(b, a % b, x, y);
    int t = y;
    y = x - a / b * y;
    x = t;
    return res;
}

// return minimal positive integer x so that ax+by=c
// or -1 if such x does not exist
int solve_equ(int a, int b, int c) {
    int x, y, d;
    d = extend_gcd(a, b, x, y);
    if (c % d)
        return -1;
    int t = c / d;
    x *= t;
    y *= t;
    int k = b / d;
    x = (x % k + k) % k;
    return x;
}

// return minimal positive integer x so that ax==b(mod p)
// or -1 if such x does not exist
int solve(int a, int b, int p) {
    a = (a % p + p) % p;
    b = (b % p + p) % p;
    return solve_equ(a, p, b);
}
```

2.3 中国剩余定理 (Durandal)

返回是否可行，余数和模数结果为 r_1, m_1

```
bool CRT(int &r1, int &m1, int r2, int m2) {
    int x, y, g = extend_gcd(m1, m2, x, y);
    if ((r2 - r1) % g != 0) return false;
    x = 1ll * (r2 - r1) * x % m2;
    if (x < 0) x += m2;
    x /= g;
    r1 += m1 * x;
    m1 *= m2 / g;
    return true;
}
```

2.4 线性同余不等式 (Durandal)

必须满足 $0 \le d < m, 0 \le l \le r < m$，返回 $\min\{x \ge 0 | l \le x \cdot d \mod m \le r\}$，无解返回 -1

```
int64_t calc(int64_t d, int64_t m, int64_t l, int64_t r) {
    if (l == 0) return 0;
    if (d == 0) return -1;
    if (d * 2 > m) return calc(m - d, m, m - r, m - l);
    if ((l - 1) / d < r / d) return (l - 1) / d + 1;
    int64_t k = calc((-m % d + d) % d, d, l % d, r % d);
    if (k == -1) return -1;
    return (k * m + l - 1) / d + 1;
}
```

2.5 组合数

2.5.1 Lucas 定理

2.5.2 组合数合数取模

2.6 高斯消元

2.7 Miller Rabin & Pollard Rho (gy)

```
/*
 * In Java, use BigInteger.isProbablePrime(int certainty) to replace miller_rabin(BigInteger number)
 * Test Set / First Wrong Answer
 * 2 / 2,047
 * 2, 3 / 1,373,653
 * 31, 73 / 9,080,191
 * 2, 3, 5 / 25,326,001
 * 2, 3, 5, 7 / 3,215,031,751 (> Int.MAX_VALUE)
 * 2, 7, 61 / 4,759,123,141
 * 2, 13, 23, 1662803 / 1,122,004,669,633
 * 2, 3, 5, 7, 11 / 2,152,302,898,747
 * 2, 3, 5, 7, 11, 13 / 3,474,749,660,383
 * 2, 3, 5, 7, 11, 13, 17 / 341,550,071,728,321
 * 2, 3, 5, 7, 11, 13, 17, 19, 23 / 3,825,123,056,546,413,051
 * 2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37 / 318,665,857,834,031,151,167,461 (> Long.MAX_VALUE)
 * 2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37, 41 / 3,317,044,064,679,887,385,961,981
 */
const int test_case_size = 12;
const int test_cases[test_case_size] = {2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37};

int64_t multiply_mod(int64_t x, int64_t y, int64_t p) {
    int64_t t = (x * y - (int64_t) ((long double) x / p * y + 1e-3) * p) % p;
    return t < 0 ? t + p : t;
}

int64_t add_mod(int64_t x, int64_t y, int64_t p) {
    return (0ull + x + y) % p;
}

int64_t power_mod(int64_t x, int64_t exp, int64_t p) {
    int64_t ans = 1;
    while (exp) {
        if (exp & 1)
            ans = multiply_mod(ans, x, p);
        x = multiply_mod(x, x, p);
        exp >>= 1;
    }
    return ans;
}

bool miller_rabin_check(int64_t prime, int64_t base) {
    int64_t number = prime - 1;
    for (; ~number & 1; number >>= 1)
        continue;
    int64_t result = power_mod(base, number, prime);
    for (; number != prime - 1 && result != 1 && result != prime - 1; number <<= 1)
        result = multiply_mod(result, result, prime);
    return result == prime - 1 || (number & 1) == 1;
}

bool miller_rabin(int64_t number) {
    if (number < 2)
        return false;
    if (number < 4)
        return true;
    if (~number & 1)
        return false;
    for (int i = 0; i < test_case_size && test_cases[i] < number; i++)
        if (!miller_rabin_check(number, test_cases[i]))
            return false;
    return true;
}

int64_t gcd(int64_t x, int64_t y) {
    return y == 0 ? x : gcd(y, x % y);
}

int64_t pollard_rho_test(int64_t number, int64_t seed) {
    int64_t x = rand() % (number - 1) + 1, y = x;
    int head = 1, tail = 2;
    while (true) {
        x = multiply_mod(x, x, number);
        x = add_mod(x, seed, number);
        if (x == y)
            return number;
        int64_t answer = gcd(std::abs(x - y), number);
        if (answer > 1 && answer < number)
            return answer;
        if (++head == tail) {
            y = x;
            tail <<= 1;
        }
    }
}

void factorize(int64_t number, std::vector<int64_t> &divisor) {
    if (number > 1) {
        if (miller_rabin(number)) {
            divisor.push_back(number);
        } else {
            int64_t factor = number;
            while (factor >= number)
                factor = pollard_rho_test(number, rand() % (number - 1) + 1);
            factorize(number / factor, divisor);
            factorize(factor, divisor);
        }
    }
}
```

2.8 $O(m^2 \log n)$ 线性递推

2.9 Polynomial

2.9.1 FFT

2.9.2 NTT & 多项式求逆

2.10 拉格朗日插值

2.11 杜教筛

2.12 BSGS

2.12.1 BSGS

2.12.2 扩展 BSGS

2.13 直线下整点个数 (gy)

必须满足 $a \geq 0, b \geq 0, m > 0$，返回 $\sum_{i=0}^{n-1} \frac{a+bi}{m}$

```
int64_t count(int64_t n, int64_t a, int64_t b, int64_t m) {
    if (b == 0)
        return n * (a / m);
    if (a >= m)
        return n * (a / m) + count(n, a % m, b, m);
    if (b >= m)
        return (n - 1) * n / 2 * (b / m) + count(n, a, b % m, m);
    return count((a + b * n) / m, (a + b * n) % m, m, b);
}
```

2.14 单纯形

2.15 辛普森积分

2.16 常用数列定理

- 第一类 Stirling Number
- 第二类 Stirling Number
- Catalan Number
  c_n 表示长度为 $2n$ 的合法括号序的数量
  $c_1 = 1,\ c_{n+1} = \sum\limits_{i=1}^{n} c_i \times c_{n+1-i}$
  $c_n = \frac{\binom{2n}{n}}{n+1}$
- Bell Number
- Bernoulli Number

2.17 积分表

Chapter 3

Geometry

3.1 点、直线、圆 (gy)

```cpp
#include <cmath>
#include <algorithm>

using number = long double;
const number eps = 1e-8;

number _sqrt(number x) {
    return std::sqrt(std::max(x, (number) 0));
}
number _asin(number x) {
    x = std::min(x, (number) 1), x = std::max(x, (number) -1);
    return std::asin(x);
}
number _acos(number x) {
    x = std::min(x, (number) 1), x = std::max(x, (number) -1);
    return std::acos(x);
}

int sgn(number x) {
    return (x > eps) - (x < -eps);
}
int cmp(number x, number y) {
    return sgn(x - y);
}

struct point {
    number x, y;
    point() {}
    point(number x, number y) : x(x), y(y) {}

    number len2() const {
        return x * x + y * y;
    }
    number len() const {
        return _sqrt(len2());
    }
    point unit() const {
        return point(x / len(), y / len());
    }
    point rotate90() const {
        return point(-y, x);
    }
    friend point operator+(const point &a, const point &b) {
        return point(a.x + b.x, a.y + b.y);
    }
    friend point operator-(const point &a, const point &b) {
        return point(a.x - b.x, a.y - b.y);
    }
    friend point operator*(const point &a, number b) {
        return point(a.x * b, a.y * b);
    }
    friend point operator/(const point &a, number b) {
        return point(a.x / b, a.y / b);
    }
    friend number dot(const point &a, const point &b) {
        return a.x * b.x + a.y * b.y;
    }
    friend number det(const point &a, const point &b) {
        return a.x * b.y - a.y * b.x;
    }
    friend number operator==(const point &a, const point &b) {
        return cmp(a.x, b.x) == 0 && cmp(a.y, b.y) == 0;
    }
};

number dis2(const point &a, const point &b) {
    return (a - b).len2();
}
number dis(const point &a, const point &b) {
    return (a - b).len();
}

struct line {
    point a, b;
    line() {}
    line(point a, point b) : a(a), b(b) {}
    point value() const {
        return b - a;
    }
};

bool point_on_line(const point &p, const line &l) {
    return sgn(det(p - l.a, p - l.b)) == 0;
}
// including endpoint
bool point_on_ray(const point &p, const line &l) {
    return sgn(det(p - l.a, p - l.b)) == 0 &&
        sgn(dot(p - l.a, l.b - l.a)) >= 0;
}
// including endpoints
bool point_on_seg(const point &p, const line &l) {
    return sgn(det(p - l.a, p - l.b)) == 0 &&
        sgn(dot(p - l.a, l.b - l.a)) >= 0 &&
        sgn(dot(p - l.b, l.a - l.b)) >= 0;
}
bool seg_has_intersection(const line &a, const line &b) {
    if (point_on_seg(a.a, b) || point_on_seg(a.b, b) ||
            point_on_seg(b.a, a) || point_on_seg(b.b, a))
        return /* including endpoints */ true;
    return sgn(det(a.a - b.a, b.b - b.a)) * sgn(det(a.b - b.a, b.b - b.a)) < 0
        && sgn(det(b.a - a.a, a.b - a.a)) * sgn(det(b.b - a.a, a.b - a.a)) < 0;
}
```

```
point intersect(const line &a, const line &b) {
    number s1 = det(a.b - a.a, b.a - a.a);
    number s2 = det(a.b - a.a, b.b - a.a);
    return (b.a * s2 - b.b * s1) / (s2 - s1);
}
point projection(const point &p, const line &l) {
    return l.a + (l.b - l.a) * dot(p - l.a, l.b - l.a) / (l.b - l.a).len2();
}
number dis(const point &p, const line &l) {
    return std::abs(dot(p - l.a, l.b - l.a)) / (l.b - l.a).len();
}
point symmetry_point(const point &a, const point &o) {
    return o + o - a;
}
point reflection(const point &p, const line &l) {
    return symmetry_point(p, projection(p, l));
}

struct circle {
    point o;
    number r;
    circle() {}
    circle(point o, number r) : o(o), r(r) {}
};

bool intersect(const line &l, const circle &a, point &p1, point &p2) {
    number x = dot(l.a - a.o, l.b - l.a);
    number y = (l.b - l.a).len2();
    number d = x * x - y * ((l.a - a.o).len2() - a.r * a.r);
    if (sgn(d) < 0) return false;
    point p = l.a - (l.b - l.a) * (x / y), delta = (l.b - l.a) * (_sqrt(d) / y);
    p1 = p + delta, p2 = p - delta;
    return true;
}
bool intersect(const circle &a, const circle &b, point &p1, point &p2) {
    if (a.o == b.o && cmp(a.r, b.r) == 0)
        return /* value for coincident circles */ false;
    number s1 = (b.o - a.o).len();
    if (cmp(s1, a.r + b.r) > 0 || cmp(s1, std::abs(a.r - b.r)) < 0)
        return false;
    number s2 = (a.r * a.r - b.r * b.r) / s1;
    number aa = (s1 + s2) / 2, bb = (s1 - s2) / 2;
    point p = (b.o - a.o) * (aa / (aa + bb)) + a.o;
    point delta = (b.o - a.o).unit().rotate90() * _sqrt(a.r * a.r - aa * aa);
    p1 = p + delta, p2 = p - delta;
    return true;
}
bool tangent(const point &p0, const circle &c, point &p1, point &p2) {
    number x = (p0 - c.o).len2();
    number d = x - c.r * c.r;
    if (sgn(d) < 0) return false;
    if (sgn(d) == 0)
        return /* value for point_on_line */ false;
    point p = (p0 - c.o) * (c.r * c.r / x);
    point delta = ((p0 - c.o) * (-c.r * _sqrt(d) / x)).rotate90();
    p1 = c.o + p + delta;
    p2 = c.o + p - delta;
    return true;
}
bool ex_tangent(const circle &a, const circle &b, line &l1, line &l2) {
    if (cmp(std::abs(a.r - b.r), (b.o - a.o).len()) == 0) {
        point p1, p2;
        intersect(a, b, p1, p2);
        l1 = l2 = line(p1, p1 + (a.o - p1).rotate90());
        return true;
    } else if (cmp(a.r, b.r) == 0) {
        point dir = b.o - a.o;
        dir = (dir * (a.r / dir.len())).rotate90();
        l1 = line(a.o + dir, b.o + dir);
        l2 = line(a.o - dir, b.o - dir);
        return true;
    } else {
        point p = (b.o * a.r - a.o * b.r) / (a.r - b.r);
        point p1, p2, q1, q2;
        if (tangent(p, a, p1, p2) && tangent(p, b, q1, q2)) {
            l1 = line(p1, q1);
            l2 = line(p2, q2);
            return true;
        } else {
            return false;
        }
    }
}
bool in_tangent(const circle &a, const circle &b, line &l1, line &l2) {
    if (cmp(a.r + b.r, (b.o - a.o).len()) == 0) {
        point p1, p2;
        intersect(a, b, p1, p2);
        l1 = l2 = line(p1, p1 + (a.o - p1).rotate90());
        return true;
    } else {
        point p = (b.o * a.r + a.o * b.r) / (a.r + b.r);
        point p1, p2, q1, q2;
        if (tangent(p, a, p1, p2) && tangent(p, b, q1, q2)) {
            l1 = line(p1, q1);
            l2 = line(p2, q2);
            return true;
        } else {
            return false;
        }
    }
}
```

3.2 点到凸包切线

3.3 直线凸包交点

3.4 凸包游戏

3.5 半平面交

3.6 旋转卡壳

3.7 判断圆是否有交

3.8 最小圆覆盖

3.9 最小球覆盖

3.10 $O(n^2 \log n)$ 圆交面积和重心

3.11 圆与多边形交

3.12 $O(n \log n)$ 凸多边形内的最大圆

3.13 三角形的五心

3.14 三维凸包

3.15 三维绕轴旋转

3.16 几何公式

Chapter 4

String

4.1 KMP

4.2 AC 自动机

4.3 后缀数组

4.4 后缀自动机

4.5 Manacher

4.6 回文自动机

4.7 最小表示法

Chapter 5

Data Structure

5.1 莫队 (ct)

```
//
//  Title: Modui
//  Date: 26.02.2016
//  Test:BZOJ-2038
//  Complexity:O(n^3/2)
//
/*
        莫队算法——将所有询问储存起来，然后分块暴力处理。
        时间复杂度为 O (n× 根号 n)。
*/
#include <cstdio>
#include <cstring>
#include <algorithm>
#include <cmath>

#ifdef WIN32
        #define LL "%I64d"
#else
        #define LL "%lld"
#endif

#ifdef CT
        #define debug(...) printf(__VA_ARGS__)
#else
        #define debug(...)
#endif

#define R register
#define getc() (S==T&&(T=(S=B)+fread(B,1,1<<15,stdin),S==T)?EOF:*S++)
#define gmax(_a, _b) ((_a) > (_b) ? (_a) : (_b))
#define gmin(_a, _b) ((_a) < (_b) ? (_a) : (_b))
#define cmax(_a, _b) (_a < (_b) ? _a = (_b) : 0)
#define cmin(_a, _b) (_a > (_b) ? _a = (_b) : 0)
char B[1<<15],*S=B,*T=B;
inline int FastIn()
{
        R char ch;R int cnt=0;R bool minus=0;
        while (ch=getc(),(ch < '0' || ch > '9') && ch != '-') ;
        ch == '-' ?minus=1:cnt=ch-'0';
        while (ch=getc(),ch >= '0' && ch <= '9') cnt = cnt * 10 + ch - '0';
        return minus?-cnt:cnt;
}
```

```
#define maxn 50010
int col[maxn],num[maxn],size,pos[maxn];
long long up[maxn],dw[maxn],ans;
struct Query{
        int l,r,id;
}q[maxn];
inline bool cmp(const Query &i,const Query &j){
        return pos[i.l]!=pos[j.l] ? (i.l<j.l) : (pos[i.l]&1 ? i.r<j.r : i.r>j.r);
}
inline long long gcd(R long long a,R long long b){
        R long long tmp;
        while (b){
                tmp=b;
                b=a%b;
                a=tmp;
        }
        return a;
}
inline void update(R int x,R int d){
        ans-=num[col[x]]*num[col[x]];
        num[col[x]]+=d;
        ans+=num[col[x]]*num[col[x]];
}
int main()
{
        R int n=FastIn(),m=FastIn();size=(int)sqrt(n*1.0);
        for (R int i=1;i<=n;i++) col[i]=FastIn(),pos[i]=(i-1)/size+1;
        for (R int i=1;i<=m;i++){
                q[i].l=FastIn();q[i].r=FastIn();q[i].id=i;
        }
        std::sort(q+1,q+m+1,cmp);
        R int l=1,r=0;
        for (R int i=1;i<=m;i++){
                R int id_now=q[i].id;
                if (q[i].l==q[i].r){
                        up[id_now]=0;dw[id_now]=1;continue;
                }
                for (;r<q[i].r;r++) update(r+1,1);
                for (;r>q[i].r;r--) update(r,-1);
                for (;l<q[i].l;l++) update(l,-1);
                for (;l>q[i].l;l--) update(l-1,1);
                R long long aa,bb,cc;
                aa=ans-q[i].r+q[i].l-1;
                bb=(long long)(q[i].r-q[i].l+1)*(q[i].r-q[i].l);
                cc=gcd(aa,bb);aa/=cc;bb/=cc;
                up[id_now]=aa;dw[id_now]=bb;
        }
        for (R int i=1;i<=m;i++) printf("%lld/%lld\n",up[i],dw[i] );
        return 0;
}
```

5.2 ST 表 (ct)

```
#include <cstdio>

#define dmax(_a, _b) ((_a) > (_b) ? (_a) : (_b))

#define maxn 200010
int a[maxn], f[20][maxn], n;
int Log[maxn];

void build()
{
        for (int i = 1; i <= n; ++i) f[0][i] = a[i];

        int lim = Log[n];
        for (int j = 1; j <= lim; ++j)
        {
                int *fj = f[j], *fj1 = f[j - 1];
                for (int i = 1; i <= n - (1 << j) + 1; ++i)
                        fj[i] = dmax(fj1[i], fj1[i + (1 << (j - 1))]);
        }
}
int Query(int l, int r)
{
        int k = Log[r - l + 1];
        return dmax(f[k][l], f[k][r - (1 << k) + 1]);
}
int main()
{
        scanf("%d", &n);
        Log[0] = -1;
        for (int i = 1; i <= n; ++i)
        {
                scanf("%d", &a[i]);
                Log[i] = Log[i >> 1] + 1;
        }
        build();
        int q;
        scanf("%d", &q);
        for (; q; --q)
        {
                int l, r; scanf("%d%d", &l, &r);
                printf("%d\n", Query(l, r) );
        }
}
```

5.3 可并堆 (ct)

```
struct Node {
        Node *ch[2];
        ll val; int size;
        inline void update()
        {
                size = ch[0] -> size + ch[1] -> size + 1;
        }
} mem[maxn], *rt[maxn];
Node *merge(Node *a, Node *b)
{
        if (a == mem) return b;
        if (b == mem) return a;
        if (a -> val < b -> val) std::swap(a, b);
        std::swap(a -> ch[0], a -> ch[1]);
        a -> ch[1] = merge(a -> ch[1], b);
        a -> update();
        return a;
}
```

5.4 线段树 (ct)

5.4.1 ZKW 线段树

```
<<<<<<< HEAD
//
//  Title:ZKW Segment Tree
//  Date:19.11.2015
//  Complexity:
//    Build Tree:O(N)
//    Query:O(logN)
//    Change:O(logN)

#include<cstdio>
#include<cmath>
#define maxn 100000
#define R register
int T[1<<18|1],n,m,M;

inline int FastIn()
{
        R char ch=getchar();R int cnt=0;R bool minus=0;
        while ((ch<'0'||ch>'9')&&ch!='-') ch=getchar();
        if (ch=='-') minus=1,ch=getchar();
        while (ch>='0'&&ch<='9') cnt=cnt*10+ch-'0',ch=getchar();
        return minus?-cnt:cnt;
}

inline void Build_Tree()
{
        for (R int i=M-1;i>=1;i--)
          T[i]=T[2*i]+T[2*i+1];
}

inline int Query(int s,int t)
{
    R int Ans;
        for (Ans=0,s=s+M-1,t=t+M+1;s^t^1;s>>=1,t>>=1)
        {
                if (~s&1) Ans+=T[s^1];
                if (t&1) Ans+=T[t^1];
        }
        return Ans;
}

inline void Change(int x,int NewValue)
{
        R int i=M+x;
        for (T[i]=NewValue,i>>=1;i;i>>=1)
          T[i]=T[2*i]+T[2*i+1];
}

int main()
{
        n=FastIn();m=FastIn();
        for (M=1;M<=n;M<<=1);
        for (R int i=0;i<n;i++)
          T[M+i]=FastIn();
        Build_Tree();
        for (R int i=1;i<=m;i++)
        {
                R char cmd=getchar();
                if (cmd=='Q')
                {
                        R int a=FastIn()-1,b=FastIn()-1;
                        printf("%d\n",Query(a,b));
                }
                if (cmd=='M')
                {
                        R int a=FastIn()-1,b=FastIn();
                        Change(a,b);
                }
        }
        return 0;
}
=======
//
// Title:ZKW Segment Tree
// Date:19.11.2015
// Complexity:
//   Build Tree:O(N)
//   Query:O(logN)
//   Change:O(logN)

#include<cstdio>
#include<cmath>
#define maxn 100000
#define R register
int T[1<<18|1],n,m,M;

inline int FastIn()
{
        R char ch=getchar();R int cnt=0;R bool minus=0;
        while ((ch<'0'||ch>'9')&&ch!='-') ch=getchar();
        if (ch=='-') minus=1,ch=getchar();
        while (ch>='0'&&ch<='9') cnt=cnt*10+ch-'0',ch=getchar();
        return minus?-cnt:cnt;
}

inline void Build_Tree()
{
        for (R int i=M-1;i>=1;i--)
          T[i]=T[2*i]+T[2*i+1];
}

inline int Query(int s,int t)
{
    R int Ans;
        for (Ans=0,s=s+M-1,t=t+M+1;s^t^1;s>>=1,t>>=1)
        {
                if (~s&1) Ans+=T[s^1];
                if (t&1) Ans+=T[t^1];
        }
        return Ans;
}

inline void Change(int x,int NewValue)
{
        R int i=M+x;
        for (T[i]=NewValue,i>>=1;i;i>>=1)
          T[i]=T[2*i]+T[2*i+1];
```

```
}

int main()
{
        n=FastIn();m=FastIn();
        for (M=1;M<=n;M<<=1);
        for (R int i=0;i<n;i++)
          T[M+i]=FastIn();
        Build_Tree();
        for (R int i=1;i<=m;i++)
        {
                R char cmd=getchar();
                if (cmd=='Q')
                {
                        R int a=FastIn()-1,b=FastIn()-1;
                        printf("%d\n",Query(a,b));
                }
                if (cmd=='M')
                {
                        R int a=FastIn()-1,b=FastIn();
                        Change(a,b);
                }
        }
        return 0;
}
>>>>>>> 49188fa6ef8b175c2f4a6388509d8dc5116ebccd
```

5.4.2 主席树

```
<<<<<<< HEAD
//
//  Title: Functional Segment Tree
//  Date:16.12.2015
//  Complexity:O((n+m)logn)
//  Test:YZOJ-1991
#include<cstdio>
#include<algorithm>
#define maxt 2000010
#define maxn 100010
#define R register
inline int FastIn(){
        R char ch=getchar();R int cnt=0;
        while (ch<'0'||ch>'9') ch=getchar();
        while (ch>='0'&&ch<='9') cnt=cnt*10+ch-'0',ch=getchar();
        return cnt;
}

int ls[maxt],
    rs[maxt],
        count[maxt],
        root[maxn],
        tot;

int num[maxn],rank[maxn],n,m,r[maxn];

bool cmp(const int &i,const int &j){
        return num[i]<num[j];
}

inline void Insert(int last,int left,int right,int pre)
```

```
{
        count[++tot]=count[last]+1;
        if (left==right) return;
        R int mid=(left+right)>>1;
        if (pre>mid){
                rs[tot]=tot+1;
                Insert(rs[last],mid+1,right,pre);
        }
        else{
                ls[tot]=tot+1;
                rs[tot]=rs[last];
                Insert(ls[last],left,mid,pre);
        }
}

inline int Query(int a,int b,int k)
{
        R int l=1,r=n,mid,f1=a,f2=b,cnt,kk=k;
        while (l<r){
                mid=(l+r)>>1;cnt=count[ls[f2]]-count[ls[f1]];
                if (cnt>=kk) f1=ls[f1],f2=ls[f2],r=mid;
                else f1=rs[f1],f2=rs[f2],l=mid+1,kk-=cnt;
        }
        return l;
}

int main()
{
        n=FastIn();m=FastIn();R int i,a,b,k;
        for (i=1;i<=n;i++) num[i]=FastIn(),rank[i]=i;
        std::sort(rank+1,rank+n+1,cmp);
        std::sort(num+1,num+n+1);
        for (i=1;i<=n;i++) r[rank[i]]=i;
        for (i=1;i<=n;i++) {
                root[i]=tot+1;
                Insert(root[i-1],1,n,r[i]);
        }
        for (i=1;i<=m;i++){
                a=FastIn();b=FastIn();k=FastIn();
                printf("%d\n",num[Query(root[a-1],root[b],k)]);
        }
        return 0;
}
=======
//
//  Title: Functional Segment Tree
//  Date:16.12.2015
//  Complexity:O((n+m)logn)
//  Test:YZOJ-1991
#include<cstdio>
#include<algorithm>
#define maxt 2000010
#define maxn 100010
#define R register
inline int FastIn(){
        R char ch=getchar();R int cnt=0;
        while (ch<'0'||ch>'9') ch=getchar();
        while (ch>='0'&&ch<='9') cnt=cnt*10+ch-'0',ch=getchar();
        return cnt;
}
```

```
int ls[maxt],
    rs[maxt],
        count[maxt],
        root[maxn],
        tot;

int num[maxn],rank[maxn],n,m,r[maxn];

bool cmp(const int &i,const int &j){
        return num[i]<num[j];
}

inline void Insert(int last,int left,int right,int pre)
{
        count[++tot]=count[last]+1;
        if (left==right) return;
        R int mid=(left+right)>>1;
        if (pre>mid){
                rs[tot]=tot+1;
                Insert(rs[last],mid+1,right,pre);
        }
        else{
                ls[tot]=tot+1;
                rs[tot]=rs[last];
                Insert(ls[last],left,mid,pre);
        }
}

inline int Query(int a,int b,int k)
{
        R int l=1,r=n,mid,f1=a,f2=b,cnt,kk=k;
        while (l<r){
                mid=(l+r)>>1;cnt=count[ls[f2]]-count[ls[f1]];
                if (cnt>=kk) f1=ls[f1],f2=ls[f2],r=mid;
                else f1=rs[f1],f2=rs[f2],l=mid+1,kk-=cnt;
        }
        return l;
}

int main()
{
        n=FastIn();m=FastIn();R int i,a,b,k;
        for (i=1;i<=n;i++) num[i]=FastIn(),rank[i]=i;
        std::sort(rank+1,rank+n+1,cmp);
        std::sort(num+1,num+n+1);
        for (i=1;i<=n;i++) r[rank[i]]=i;
        for (i=1;i<=n;i++) {
                root[i]=tot+1;
                Insert(root[i-1],1,n,r[i]);
        }
        for (i=1;i<=m;i++){
                a=FastIn();b=FastIn();k=FastIn();
                printf("%d\n",num[Query(root[a-1],root[b],k)]);
        }
        return 0;
}
>>>>>>> 49188fa6ef8b175c2f4a6388509d8dc5116ebccd
```

5.5 平衡树 (ct)

5.5.1 Splay

```
//
// Title : Splay Tree
// Date : 11.01.2016
// Complexity : O(nlogn) (期望)
// Test : BZOJ-1251
/*
*/
#include <cstdio>
#include <cstring>
#include <algorithm>
#include <cmath>

#ifdef WIN32
        #define LL "%I64d"
#else
        #define LL "%lld"
#endif

#ifdef CT
        #define debug(...) printf(__VA_ARGS__)
#else
        #define debug(...)
#endif

#define R register
#define getc() (S==T&&(T=(S=B)+fread(B,1,1<<15,stdin),S==T)?EOF:*S++)
#define gmax(_a, _b) ((_a) > (_b) ? (_a) : (_b))
#define gmin(_a, _b) ((_a) < (_b) ? (_a) : (_b))
#define cmax(_a, _b) (_a < (_b) ? _a = (_b) : 0)
#define cmin(_a, _b) (_a > (_b) ? _a = (_b) : 0)
char B[1<<15],*S=B,*T=B;
inline int FastIn()
{
        R char ch;R int cnt=0;R bool minus=0;
        while (ch=getc(),(ch < '0' || ch > '9') && ch != '-') ;
        ch == '-' ?minus=1:cnt=ch-'0';
        while (ch=getc(),ch >= '0' && ch <= '9') cnt = cnt * 10 + ch - '0';
        return minus?-cnt:cnt;
}
#define maxn 50010
int n,Q,root;
int fa[maxn],ch[maxn][2],id[maxn],size[maxn];
int tag[maxn],mx[maxn],num[maxn];
bool rev[maxn];
inline void update(int x){
        R int ls=ch[x][0],rs=ch[x][1];
        mx[x]=num[x];
        cmax(mx[x],mx[ls]);cmax(mx[x],mx[rs]);
        size[x]=size[ls]+size[rs]+1;
}//更新
void build(int l,int r,int rt){
        if (l>r) return ;
        R int mid=l+r>>1;
        fa[mid]=rt;
        if (mid<rt) ch[rt][0]=mid;
        else ch[rt][1]=mid;
        build(l,mid-1,mid);
        build(mid+1,r,mid);
        update(mid);
}//建树
inline void pushdown(int x){
        R int ls=ch[x][0],rs=ch[x][1];
        if (tag[x]){
                R int lazy=tag[x];
                if (ls) tag[ls]+=lazy,num[ls]+=lazy,mx[ls]+=lazy;
                if (rs) tag[rs]+=lazy,num[rs]+=lazy,mx[rs]+=lazy;
                tag[x]=0;
        }
        if (rev[x]){
                if (ls) rev[ls]^=1;
                if (rs) rev[rs]^=1;
                ch[x][1]=ls;ch[x][0]=rs;
                rev[x]=0;
        }
}//具体下传的过程
inline void rotate(int x){//把x向上旋转到x的父亲
        R int f=fa[x],gf=fa[f],d=(ch[f][1]==x);//f表示x的父亲，gf是祖父，d是x在其父亲的位置
        if (f==root) root=x,ch[0][0]=x;
        (ch[f][d]=ch[x][d^1])>0 ? fa[ch[f][d]]=f : 0;//把x的儿子中与d相反的节点来代替x的位置
        (fa[x]=gf)>0 ? ch[gf][ch[gf][1]==f]=x : 0;//把x代替f的位置
        fa[ch[x][d^1]=f]=x;//把f接到x的下面
        update(f);//更新f节点
}
inline void splay(int x,int rt){//把x旋转到rt
        while (fa[x]!=rt){
                R int f=fa[x],gf=fa[f];
                if (gf!=rt) rotate((ch[gf][1]==f)^(ch[f][1]==x) ? x :f);//如果祖孙三代是相同方向就转父亲，不然转自己
                rotate(x);
        }
        update(x);
}
int find(int x,int rank){
        if (tag[x]||rev[x]) pushdown(x);
        R int ls=ch[x][0],rs=ch[x][1],lsize=size[ls];
        if (lsize+1==rank) return x;
        if (lsize>=rank) return find(ls,rank);
        else return find(rs,rank-lsize-1);
}//找第k小
inline int prepare(int l,int r){
        R int x=find(root,l-1);
        splay(x,0);
        x=find(root,r+1);
        splay(x,root);
        return ch[x][0];
}//把l-1旋到根，r+1旋到右儿子，然后返回r+1的左儿子，返回一个包含[l，r]的节点
inline void add(int l,int r,int w){
        R int x=prepare(l,r);
        tag[x]+=w,num[x]+=w,mx[x]+=w;
}//区间加
inline void rever(int l,int r){
        R int x=prepare(l,r);
        rev[x]^=1;
}//区间翻转
inline void query(int l,int r){
        R int x=prepare(l,r);
        printf("%d\n",mx[x] );
```

```
}//区间查询最大值
inline int split(R int k){
        R int ls;
        if (k<size[root])
        {
                R int kth=find(root,k+1);
                splay(kth);ls=ch[kth][0];
                fa[ls]=0;ch[kth][0]=0;
                size[kth]-=size[ls];
        }
        else{
                ls=root;root=0;
        }
        return ls;
}//删除数列
inline void merge(R int nwrt){
        if (!root) {root=nwrt;return;}
        R int nw=find(root,1);
        splay(nw);fa[nwrt]=nw;ch[nw][0]=nwrt;
        size[nw]+=size[nwrt];
}//合并数列
int main()
{
        n=FastIn()+2;Q=FastIn();R int i,l,r,v,cmd;mx[0]=-23333333;
        build(1,n,0);root=(1+n)>>1;
        for (;Q--;){
                cmd=FastIn();l=FastIn()+1;r=FastIn()+1;
                if (cmd==1) v=FastIn(),add(l,r,v);
                else if (cmd==2) rever(l,r);
                else query(l,r);
        }
        return 0;
}
```

5.5.2 非旋转 Treap

```
//
//  Title : Treap (unrotated)
//  Date : 13.04.2016
//  Test : BZOJ-3224
//  Complexity : O(nlogn)(期望)
//
/*
        对于序列上的一些操作的问题——
        解决办法: 平衡树 Treap
*/
#include <cstdio>
#include <cstring>
#include <algorithm>
#include <cmath>

#ifdef WIN32
        #define LL "%I64d"
#else
        #define LL "%lld"
#endif

#ifdef CT
        #define debug(...) printf(__VA_ARGS__)
        #define setfile()
```

```
#else
        #define debug(...)
        #define filename ""
        #define setfile() freopen(filename".in", "r", stdin); freopen(filename".out", "w", stdout);
#endif

#define R register
#define getc() (S == T && (T = (S = B) + fread(B, 1, 1 << 15, stdin), S == T) ? EOF : *S++)
#define dmax(_a, _b) ((_a) > (_b) ? (_a) : (_b))
#define dmin(_a, _b) ((_a) < (_b) ? (_a) : (_b))
#define cmax(_a, _b) (_a < (_b) ? _a = (_b) : 0)
#define cmin(_a, _b) (_a > (_b) ? _a = (_b) : 0)
char B[1 << 15], *S = B, *T = B;
inline int FastIn()
{
        R char ch; R int cnt = 0; R bool minus = 0;
        while (ch = getc(), (ch < '0' || ch > '9') && ch != '-') ;
        ch == '-' ? minus = 1 : cnt = ch - '0';
        while (ch = getc(), ch >= '0' && ch <= '9') cnt = cnt * 10 + ch - '0';
        return minus ? -cnt : cnt;
}
const int Ta = 1 << 16 | 3, Tb = 33333331;
int Tc;
inline int randint() {return Tc = Ta * Tc + Tb;}
struct Treap
{
        int data, key, size;
        Treap *ls, *rs;
        Treap(int _val):data(_val), key(randint()), ls(NULL), rs(NULL), size(1){}
        inline void update()
        {
                size = (ls ? ls -> size : 0) + (rs ? rs -> size : 0) + 1;
        }
}*root;
inline int Size(Treap *x)
{
        return x ? x -> size : 0;
}
//为了防止访问到空节点，定义一个函数来访问 size
struct Pair
{
        Treap *fir, *sec;
};
Treap *Merge(Treap *a, Treap *b)
{
        if (!a) return b;
        if (!b) return a;
        if (a -> key < b -> key)
        {
                a -> rs = Merge(a -> rs, b);
                a -> update();
                return a;
        }
        else
        {
                b -> ls = Merge(a, b -> ls);
                b -> update();
                return b;
        }
}
//按照 a, b 的顺序来合并两棵 Treap
```

```
Pair Split(Treap *x, int k)
{
        if (!x) return (Pair){NULL, NULL};
        Pair y; y.fir = NULL; y.sec = NULL;
        if (Size(x -> ls) >= k)
        {
                y = Split(x -> ls, k);
                x -> ls = y.sec;
                x -> update();
                y.sec = x;
        }
        else
        {
                y = Split(x -> rs, k - Size(x -> ls) - 1);
                x -> rs = y.fir;
                x -> update();
                y.fir = x;
        }
        return y;
}
//将前k个的点分离出来
inline int Find(R int k)
{
        Pair x = Split(root, k - 1);
        Pair y = Split(x.sec, 1);
        Treap *ans = y.fir;
        root = Merge(Merge(x.fir, ans), y.sec);
        return ans -> data;
}
//找到第k小的data值
int Get(Treap *x, R int val)
{
        if (!x) return 0;
        return val < x -> data ? Get(x -> ls, val) : Get(x -> rs, val) + Size(x -> ls) + 1;
}
//找到val的排名
inline void Insert(R int val)
{
        R int k = Get(root, val);
        Pair x = Split(root, k);
        Treap *pre = new Treap(val);
        root = Merge(Merge(x.fir, pre), x.sec);
}
//插入
inline void Delete(R int val)
{
        R int k = Get(root, val);
        Pair x = Split(root, k - 1);
        Pair y = Split(x.sec, 1);
        root = Merge(x.fir, y.sec);
}
//单点删除
inline int upper(R int val)
{
        R int ans = 1e9;
        Treap *tmp = root;
        while (tmp)
        {
                if (tmp -> data > val)
                {
                        cmin(ans, tmp -> data);
                        tmp = tmp -> ls;
                }
                else
                        tmp = tmp -> rs;
        }
        return ans;
}
inline int lower(R int val)
{
        R int ans = -1e9;
        Treap *tmp = root;
        while (tmp)
        {
                if (tmp -> data < val)
                {
                        cmax(ans, tmp -> data);
                        tmp = tmp -> rs;
                }
                else tmp = tmp -> ls;
        }
        return ans;
}
void print(Treap *x)
{
        if (!x) return;
        print(x -> ls);
        printf("%d ",x -> data );
        print(x -> rs);
}
int main()
{
        root = NULL;
        for (R int Q = FastIn(); Q; --Q)
        {
                R int opt = FastIn(), x = FastIn();
                if (opt == 1) Insert(x);
                else if (opt == 2) Delete(x);
                else if (opt == 3)
                {
                        R int ans = Get(root, x);
                        while (ans > 1 && Find(ans - 1) == x) ans--;
                        printf("%d\n", ans );
                }
                else if (opt == 4) printf("%d\n", Find(x) );
                else if (opt == 5) printf("%d\n",lower(x) );
                else printf("%d\n",upper(x) );
        }
        return 0;
}
/*
input:
10
1 106465
4 1
1 317721
1 460929
1 644985
1 84185
1 89851
6 81968
1 492737
```

```
5 493598

output:
106465
84185
492737

input2:
5
1 1
1 1
1 1
1 2
3 1
output2:
1
*/
```

5.5.3 可持久化平衡树

```
//
//  Title: Functional Treap
//  Date: 16.04.2016
//  Test:YZOJ-1620
//  Complexity:O(nlogn)(期望)
//
/*
    可持久化 Treap:
        用来解决超级编辑器等问题。
        优势：好写好调好理解的平衡树
        缺点：写不好看的话常数大。(相较于 SBT 来说，甚至有可能会比 splay 慢)，需手写 rand
*/
#include <cstdio>
#include <cstring>
#include <algorithm>
#include <cmath>

#ifdef WIN32
        #define LL "%I64d"
#else
        #define LL "%lld"
#endif

#ifdef CT
        #define debug(...) printf(__VA_ARGS__)
        #define setfile()
#else
        #define debug(...)
        #define filename ""
        #define setfile() freopen(filename".in", 'r', stdin); freopen(filename".out", 'w', stdout)
#endif

#define R register
//#define getc() (S==T&&(T=(S=B)+fread(B,1,1<<15,stdin),S==T)?EOF:*S++)
#define getc() getchar()
#define dmax(_a, _b) ((_a) > (_b) ? (_a) : (_b))
#define dmin(_a, _b) ((_a) < (_b) ? (_a) : (_b))
#define cmax(_a, _b) (_a < (_b) ? _a = (_b) : 0)
#define cmin(_a, _b) (_a > (_b) ? _a = (_b) : 0)
#define cabs(_x) ((_x)<0?(-_x):(_x))
```

```
char B[1<<15],*S=B,*T=B;
inline int FastIn()
{
        R char ch;R int cnt=0;R bool minus=0;
        while (ch=getc(),(ch < '0' || ch > '9') && ch != '-') ;
        ch == '-' ?minus=1:cnt=ch-'0';
        while (ch=getc(),ch >= '0' && ch <= '9') cnt = cnt * 10 + ch - '0';
        return minus?-cnt:cnt;
}
#define maxn 100010
char str[maxn];
struct Treap
{
        char data;
        int size;
        Treap *ls, *rs;
        Treap(char _ch): data(_ch), size(1), ls(NULL), rs(NULL){}
        inline void update()
        {
                size = (ls ? ls -> size : 0) + (rs ? rs -> size : 0) + 1;
        }
}*root[maxn];
inline int Size(Treap *x)
{
        return x ? x -> size : 0;
}
struct Pair
{
        Treap *fir, *sec;
};
inline Treap *copy(Treap *x)
{
        if (!x) return NULL;
        Treap *nw = new Treap(x -> data);
        nw -> ls = x -> ls;
        nw -> rs = x -> rs;
        nw -> size = x -> size;
        return nw;
}
Pair Split(Treap *x, int k)
{
        if (!x) return (Pair){NULL, NULL};
        Pair y; y.fir = NULL; y.sec = NULL;
        Treap *nw = copy(x);
        if (Size(nw -> ls) >= k)
        {
                y = Split(nw -> ls, k);
                nw -> ls = y.sec;
                nw -> update();
                y.sec = nw;
        }
        else
        {
                y = Split(nw -> rs, k - Size(nw -> ls) - 1);
                nw -> rs = y.fir;
                nw -> update();
                y.fir = nw;
        }
        return y;
}
const int Ta = 1 << 16 | 3, Tb = 33333331;
```

```
unsigned int Tc;
inline unsigned int randint(){return Tc = Ta * Tc + Tb;}
Treap *Merge(Treap *a, Treap *b)
{
        Treap *nw;
        if (!a) return nw = copy(b);
        if (!b) return nw = copy(a);
        if (randint() % (Size(a) + Size(b)) < Size(a))
        {
                nw = copy(a);
                nw -> rs = Merge(nw -> rs, b);
        }
        else
        {
                nw = copy(b);
                nw -> ls = Merge(a, nw -> ls);
        }
        nw -> update();
        return nw;
}
Treap *Build(int l, int r)
{
        if (l > r) return NULL;
        R int mid = l + r >> 1;
        Treap *nw = new Treap(str[mid]);
        nw -> ls = Build(l, mid - 1);
        nw -> rs = Build(mid + 1, r);
        nw -> update();
        return nw;
}
int now;
inline void Insert(R int k, R char ch)
{
        Pair x = Split(root[now], k);
        Treap *nw = new Treap(ch);
        root[++now] = Merge(Merge(x.fir, nw), x.sec);
}
inline void Del(R int l, R int r)
{
        Pair x = Split(root[now], l - 1);
        Pair y = Split(x.sec, r - l + 1);
        root[++now] = Merge(x.fir, y.sec);
}
inline void Copy(R int l, R int r, R int ll)
{
        Pair x = Split(root[now], l - 1);
        Pair y = Split(x.sec, r - l + 1);
        Pair z = Split(root[now], ll);
        Treap *ans = y.fir;
        root[++now] = Merge(Merge(z.fir, ans), z.sec);
}
inline void Print(Treap *x, R int l, R int r)
{
        if (!x) return ;
        if (l > r) return;
        R int mid = Size(x -> ls) + 1;
        if (r < mid)
        {
                Print(x -> ls, l, r);
                return ;
        }
        if (l > mid)
        {
                Print(x -> rs, l - mid, r - mid);
                return ;
        }
        Print(x -> ls, l, mid - 1);
        printf("%c",x -> data );
        Print(x -> rs, 1, r - mid);
}
inline void Printtree(Treap *x)
{
        if (!x) return;
        Printtree(x -> ls);
        printf("%c",x -> data );
        Printtree(x -> rs);
}
int main()
{
//        setfile();
        R int n = FastIn();
        gets(str + 1);
        R int len = strlen(str + 1);
        root[0] = Build(1, len);
        while (1)
        {
                R char opt = getc();
                while (opt < 'A' || opt > 'Z')
                {
                        if (opt == EOF) return 0;
                        opt = getc();
                }
                if (opt == 'I')
                {
                        R int x = FastIn();
                        R char ch = getc();
                        Insert(x, ch);
                }
                else if (opt == 'D')
                {
                        R int l = FastIn(), r = FastIn();
                        Del(l, r);
                }
                else if (opt == 'C')
                {
                        R int x = FastIn(), y = FastIn(), z = FastIn();
                        Copy(x, y, z);
                }
                else if (opt == 'P')
                {
                        R int x = FastIn(), y = FastIn(), z = FastIn();
//                        printf("%d %d %d\n",x, y, z );
                        Print(root[now - x], y, z);
                        puts("");
                }
//                Printtree(root[now]);
//                puts("");
        }
        return 0;
}
```

5.6 CDQ 分治 (ct)

```
//
//  Title : cdq 分治
//  Date : 18.04.2016
//  Test : BZOJ-1176
//  Complexity : O(nlog^2n)
//
/*
        对于三维偏序等问题——
        解决办法：离线询问，分治降维，剩下一维用随便什么树乱搞。这样就不用写树套树啦!
*/
#include <cstdio>
#include <cstring>
#include <algorithm>
#include <cmath>

#ifdef WIN32
        #define LL "%I64d"
#else
        #define LL "%lld"
#endif

#ifdef CT
        #define debug(...) printf(__VA_ARGS__)
        #define setfile()
#else
        #define debug(...)
        #define filename ""
        #define setfile() freopen(filename".in", "r", stdin); freopen(filename".out", "w", stdout);
#endif

#define R register
#define getc() (S == T && (T = (S = B) + fread(B, 1, 1 << 15, stdin), S == T) ? EOF : *S++)
#define dmax(_a, _b) ((_a) > (_b) ? (_a) : (_b))
#define dmin(_a, _b) ((_a) < (_b) ? (_a) : (_b))
#define cmax(_a, _b) (_a < (_b) ? _a = (_b) : 0)
#define cmin(_a, _b) (_a > (_b) ? _a = (_b) : 0)
char B[1 << 15], *S = B, *T = B;
inline int FastIn()
{
        R char ch; R int cnt = 0; R bool minus = 0;
        while (ch = getc(), (ch < '0' || ch > '9') && ch != '-') ;
        ch == '-' ? minus = 1 : cnt = ch - '0';
        while (ch = getc(), ch >= '0' && ch <= '9') cnt = cnt * 10 + ch - '0';
        return minus ? -cnt : cnt;
}
#define maxn 200010
#define maxm 2000010
struct event
{
        int x, y, pos, opet, ans;
        inline bool operator < (const event &that) const {return pos < that.pos ;}
}t[maxn], q[maxn];
#define lowbit(_x) ((_x) & -(_x))
int bit[maxm], last[maxm], s, w, cnt, now;
inline void add(R int x, R int val)
{
        for (; x <= w; x += lowbit(x))
        {
                if (last[x] != now)
                        bit[x] = 0;
                bit[x] += val;
                last[x] = now;
        }
}
inline int query(R int x)
{
        R int ans = 0;
        for (; x ; x -= lowbit(x))
        {
                if (last[x] == now)
                        ans += bit[x];
        }
        return ans;
}
void cdq(R int left, R int right)
{
        if (left == right) return ;
        R int mid = left + right >> 1;
        cdq(left, mid); cdq(mid + 1, right);
        //分成若干个子问题
        ++now;
        for (R int i = left, j = mid + 1; j <= right; ++j)
        {
                for (; i <= mid && q[i].x <= q[j].x; ++i)
                        if (!q[i].opet)
                                add(q[i].y, q[i].ans);
                //考虑前面的修改操作对后面的询问的影响
                if (q[j].opet)
                        q[j].ans += query(q[j].y);
        }
        R int i, j, k = 0;
        //以下相当于归并排序
        for (i = left, j = mid + 1; i <= mid && j <= right; )
        {
                if (q[i].x <= q[j].x)
                        t[k++] = q[i++];
                else
                        t[k++] = q[j++];
        }
        for (; i <= mid; )
                t[k++] = q[i++];
        for (; j <= right; )
                t[k++] = q[j++];
        for (R int i = 0; i < k; ++i)
                q[left + i] = t[i];
}
int main()
{
//        setfile();
        s = FastIn();
        w = FastIn();
        while (1)
        {
                R int opt = FastIn();
                if (opt == 1)
                {
                        R int x = FastIn(), y = FastIn(), a = FastIn();
                        q[++cnt] = (event){x, y, cnt, 0, a};
                }
                if (opt == 2)
                {
                        R int x = FastIn() - 1, y = FastIn() - 1, a = FastIn(), b = FastIn();
                        q[++cnt] = (event) {x, y, cnt, 1, x * y * s};
                        q[++cnt] = (event) {a, b, cnt, 2, a * b * s};
                        q[++cnt] = (event) {x, b, cnt, 2, x * b * s};
                        q[++cnt] = (event) {a, y, cnt, 2, a * y * s};
                }
                if (opt == 3) break;
        }
        cdq(1, cnt);
        std::sort(q + 1, q + cnt + 1);
        for (R int i = 1; i <= cnt; ++i)
                if (q[i].opet == 1)
                        printf("%d\n",q[i].ans + q[i + 1].ans - q[i + 2].ans - q[i + 3].ans ), i += 3;
        return 0;
}
```

Chapter 6

Others

6.1 vimrc (gy)

```
se et ts=4 sw=4 sts=4 nu sc sm lbr is hls mouse=a
sy on
ino <tab> <c-n>
ino <s-tab> <tab>
au winnew * winc L

nm <f6> ggVG"+y
nm <f7> :w<cr>:make<cr>
nm <f8> :!@@<cr>
nm <f9> :!@@ < in<cr>
nm <s-f9> :!(time @@ < in &>> out) &>> out<cr>:sp out<cr>

au filetype cpp cm @@ ./a.out | se cin fdm=syntax mp=g++\ %\ -std=c++11\ -Wall\ -Wextra\ -O2

map <c-p> :ha<cr>
se pheader=%n\ %f

au filetype java cm @@ java %< | se cin fdm=syntax mp=javac\ %
au filetype python cm @@ python % | se si fdm=indent
au bufenter *.kt setf kotlin
au filetype kotlin cm @@ kotlin _%<Kt | se si mp=kotlinc\ %
```

6.2 Java Template (gy)

```java
import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStreamReader;
import java.math.BigDecimal;
import java.math.BigInteger;
import java.math.RoundingMode;
import java.util.ArrayDeque;
import java.util.ArrayList;
import java.util.Arrays;
import java.util.Comparator;
import java.util.Deque;
import java.util.LinkedList;
import java.util.List;
import java.util.Scanner;
import java.util.StringTokenizer;
```

```java
public class Template {
    // Input
    private static BufferedReader reader;
    private static StringTokenizer tokenizer;

    private static String next() {
        try {
            while (tokenizer == null || !tokenizer.hasMoreTokens())
                tokenizer = new StringTokenizer(reader.readLine());
        } catch (IOException e) {
            // do nothing
        }
        return tokenizer.nextToken();
    }

    private static int nextInt() {
        return Integer.parseInt(next());
    }

    private static double nextDouble() {
        return Double.parseDouble(next());
    }

    private static BigInteger nextBigInteger() {
        return new BigInteger(next());
    }

    public static void main(String[] args) {
        reader = new BufferedReader(new InputStreamReader(System.in));
        Scanner scanner = new Scanner(System.in);
        while (scanner.hasNext())
            scanner.next();
    }

    // BigInteger & BigDecimal
    private static void bigDecimal() {
        BigDecimal a = BigDecimal.valueOf(1.0);
        BigDecimal b = a.setScale(50, RoundingMode.HALF_EVEN);
        BigDecimal c = b.abs();
        // if scale omitted, b.scale is used
        BigDecimal d = c.divide(b, 50, RoundingMode.HALF_EVEN);
        // since Java 9
        BigDecimal e = d.sqrt(new MathContext(50, RoundingMode.HALF_EVEN));
        BigDecimal x = new BigDecimal(BigInteger.ZERO);
        BigInteger y = BigDecimal.ZERO.toBigInteger(); // RoundingMode.DOWN
        y = BigDecimal.ZERO.setScale(0, RoundingMode.HALF_EVEN).unscaledValue();
    }

    // sqrt for Java 8
    private static BigDecimal sqrt(BigDecimal a, int scale, RoundingMode mode) {
        if (a.equals(BigDecimal.ZERO))
            return BigDecimal.ZERO;
        a = a.setScale(scale, mode);
        BigDecimal ans = a;
        BigDecimal TWO = BigDecimal.valueOf(2L);
        for (int i = 1; i <= scale; i++)
            ans = ans.add(a.divide(ans, scale, mode)).divide(TWO, scale, mode);
        return ans;
    }

    private static BigInteger sqrt(BigInteger a) {
        BigInteger about = BigInteger.ZERO.setBit(a.bitLength() / 2);
        return sqrt(new BigDecimal(a.toString()), new BigDecimal(about.toString())).setScale(0,
          ↪ RoundingMode.FLOOR).unscaledValue();
    }

    private static BigDecimal sqrt(BigDecimal a, BigDecimal initial) {
        if (a.equals(BigDecimal.ZERO))
            return BigDecimal.ZERO;
        a = a.setScale(50, RoundingMode.HALF_EVEN);
        BigDecimal ans = initial;
        for (int i = 1; i <= 10; i++)
            ans = ans.add(a.divide(ans, RoundingMode.HALF_EVEN)).divide(BigDecimal.valueOf(2),
              ↪ RoundingMode.HALF_EVEN);
        return ans;
    }

    // ArrayList
    private static void arrayList() {
        List<Integer> list = new ArrayList<>();
        // Generic array is banned
        List[] lists = new List[100];
        lists[0] = new ArrayList<Integer>();
        // for List<Integer>, remove(Integer) stands for element, while remove(int) stands for
          ↪ index
        list.remove(list.get(1));
        list.remove(list.size() - 1);
        list.clear();
    }

    // Queue
    private static void queue() {
        LinkedList<Integer> queue = new LinkedList<>();
        // return the value without popping
        queue.peek();
        // pop and return the value
        queue.poll();
        Deque<Integer> deque = new ArrayDeque<>();
        deque.peekFirst();
        deque.peekLast();
        deque.pollFirst();
    }

    // Others
    private static void others() {
        Arrays.sort(new int[10]);
        Arrays.sort(new Integer[10], (a, b) -> {
            if (a.equals(b)) return 0;
            if (a > b) return -1;
            return 1;
        });
        Arrays.sort(new Integer[10], Comparator.comparingInt((a) -> (int) a).reversed());
        long a = 1_000_000_000_000_000_000L;
        int b = Integer.MAX_VALUE;
        int c = 'a';
    }
}
```

6.3 Big Fraction (gy)

```
fun gcd(a: Long, b: Long): Long = if (b == 0L) a else gcd(b, a % b)

class Fraction(val a: BigInteger, val b: BigInteger) {
    constructor(a: Long, b: Long) : this(BigInteger.valueOf(a / gcd(a, b)), BigInteger.valueOf(b / gcd(a, b)))

    operator fun plus(o: Fraction): Fraction {
        var gcd = b.gcd(o.b)
        val tempProduct = (b / gcd) * (o.b / gcd)
        var ansA = a * (o.b / gcd) + o.a * (b / gcd)
        val gcd2 = ansA.gcd(gcd)
        ansA /= gcd2
        gcd /= gcd2
        return Fraction(ansA, gcd * tempProduct)
    }

    operator fun minus(o: Fraction): Fraction {
        var gcd = b.gcd(o.b)
        val tempProduct = (b / gcd) * (o.b / gcd)
        var ansA = a * (o.b / gcd) - o.a * (b / gcd)
        val gcd2 = ansA.gcd(gcd)
        ansA /= gcd2
        gcd /= gcd2
        return Fraction(ansA, gcd * tempProduct)
    }

    operator fun times(o: Fraction): Fraction {
        val gcd1 = a.gcd(o.b)
        val gcd2 = b.gcd(o.a)
        return Fraction((a / gcd1) * (o.a / gcd2), (b / gcd2) * (o.b / gcd1))
    }
}
```

6.4 模拟退火 (ct)

```
#include <cstdio>
#include <cmath>
#include <cstdlib>
#include <ctime>

#define R register
#define cmax(_a, _b) (_a < (_b) ? _a = (_b) : 0)
#define maxn 10010
struct Poi {
        double x, y, m;
}p[maxn];
double ans_x, ans_y, fans;
int n;
inline double rand01() {return rand() / 2147483647.0;}
inline double randp() {return (rand() & 1 ? 1 : -1) * rand01();}
inline double sqr(R double x) {return x * x;}
inline double f(R double x, R double y)
{
        R double maxx = 0;
        for (R int i = 1; i <= n; ++i)
                maxx += sqrt(sqr(x - p[i].x) + sqr(y - p[i].y)) * p[i].m;
```

```cpp
        if (maxx < fans) {fans = maxx; ans_x = x; ans_y = y;}
        return maxx;
}
int main()
{
        srand(time(NULL) + clock());
        scanf("%d", &n);
        R double x = 0, y = 0, tot = 0;
        for (R int i = 1; i <= n; ++i)
                scanf("%lf%lf%lf", &p[i].x, &p[i].y, &p[i].m), x += p[i].x * p[i].m, y += p[i].y * p[i].m, tot += p[i].m;
        fans = 1e30; x /= tot; y /= tot;
        R double fnow = f(x, y);
        for (R double T = 1e4; T > 1e-4; T *= 0.997)
        {
                R double nx = x + randp() * T, ny = y + randp() * T, fnext = f(nx, ny);
                R double delta = fnext - fnow;
                if (delta < 1e-9 || exp(-delta / T) > rand01())
                {
                        x = nx; y = ny; fnow = fnext;
                }
        }
        printf("%.3lf %.3lf\n", ans_x, ans_y);
        return 0;
}
```

6.5 三分 (ct)

```cpp
#define maxn 200010
#define inf 1e9
int a[maxn], n;
inline double check(R double x)
{
        R double tmp, tmp1 = 0, tmp2 = 0, maxx = -inf, minn = -inf;
        for (R int i = 1; i <= n; ++i)
        {
                tmp = (double) a[i] - x;

                tmp1 += tmp;
                cmax(maxx, tmp1);
                tmp1 < 0 ? tmp1 = 0 : 0;

                tmp2 -= tmp;
                cmax(minn, tmp2);
                tmp2 < 0 ? tmp2 = 0 : 0;
        }
        return dmax(maxx, minn);
}
int main()
{
        n = F();
        for (R int i = 1; i <= n; ++i) a[i] = F();
        R double l = -1e4, r = 1e4;
        for (R int i = 1; i <= 100; ++i)
        {
                R double ll = (l + r) * 0.5;
                R double rr = (ll + r) * 0.5;
                if (check(ll) < check(rr)) r = rr;
                else l = ll;
        }
        printf("%.6lf\n", check((l + r) * 0.5) );
        return 0;
}
```

6.6 博弈论模型 (gy)

- Wythoff’s game

  给定两堆石子，每次可以从任意一堆中取至少一个石子，或从两堆中取相同的至少一个石子，取走最后石子的胜

  先手胜当且仅当石子数满足：

  $\lfloor (b-a) \times \phi \rfloor = a, (a \leq b, \phi = \frac{\sqrt{5}+1}{2})$

  先手胜对应的石子数构成两个序列：

  Lower Wythoff sequence: $a_n = \lfloor n \times \phi \rfloor$

  Upper Wythoff sequence: $b_n = \lfloor n \times \phi^2 \rfloor$

- Fibonacci nim

  给定一堆石子，第一次可以取至少一个、少于石子总数数量的石子，之后每次可以取至少一个、不超过上次取石子数量两倍的石子，取走最后石子的胜

  先手胜当且仅当石子数为斐波那契数