

Platelet

Team Reference Material

凌皓煜 陈彤 顾逸

2018

Contents

1	Graph Theory	5
1.1	2-SAT	6
1.2	双连通分量	6
1.2.1	点双连通分量	6
1.2.2	边双连通分量	6
1.3	K 短路	6
1.4	最大团	6
1.5	一般图最大匹配	6
1.6	树	6
1.6.1	虚树	6
1.6.2	矩阵树定理	6
1.6.3	点分治	6
1.6.4	Prufer 编码	6
1.6.5	Link-Cut Tree	6
1.6.6	树上倍增	6
1.6.7	数链剖分	6
1.7	仙人掌	6
1.8	带花树	6
1.9	KM 算法	6
1.10	支配树	6
1.10.1	DAG	6
1.10.2	一般图	6
1.11	弦图	6
1.12	网络流	6
1.12.1	最小割	6
1.12.2	最大流	6
1.12.3	费用流	6
1.12.4	有上下界的网络流	6
1.12.5	zkw 费用流	6
1.13	差分约束	6
2	Math	7
2.1	int64 相乘取模 (Durandal)	7
2.2	扩展欧几里得 (gy)	7
2.3	中国剩余定理 (Durandal)	8
2.4	线性同余不等式 (Durandal)	8
2.5	组合数	8
2.5.1	Lucas 定理	8
2.5.2	组合数合数取模	8
2.6	高斯消元	8
2.7	Miller Rabin & Pollard Rho (gy)	8
2.8	$O(m^2 \log n)$ 线性递推	10
2.9	Polynomial	10
2.9.1	FFT	10

2.9.2	NTT & 多项式求逆	10
2.10	拉格朗日插值	10
2.11	杜教筛	10
2.12	BSGS	10
2.12.1	BSGS	10
2.12.2	扩展 BSGS	10
2.13	直线下整点个数 (gy)	10
2.14	单纯形	11
2.15	辛普森积分	11
2.16	常用数列定理	11
2.17	积分表	11
3	Geometry	12
3.1	点、直线、圆 (gy)	12
3.2	点到凸包切线	16
3.3	直线凸包交点	16
3.4	凸包游戏	16
3.5	半平面交	16
3.6	旋转卡壳	16
3.7	判断圆是否有交	16
3.8	最小圆覆盖	16
3.9	最小球覆盖	16
3.10	$O(n^2 \log n)$ 圆交面积和重心	16
3.11	圆与多边形交	16
3.12	$O(n \log n)$ 凸多边形内的最大圆	16
3.13	三角形的五心	16
3.14	三维凸包	16
3.15	三维绕轴旋转	16
3.16	几何公式	16
4	String	17
4.1	KMP	17
4.2	AC 自动机	17
4.3	后缀数组	17
4.4	后缀自动机	17
4.5	Manacher	17
4.6	回文自动机	17
4.7	最小表示法	17
5	Data Structure	18
5.1	莫队 (ct)	18
5.2	ST 表 (ct)	19
5.3	可并堆 (ct)	20
5.4	线段树 (ct)	21
5.4.1	ZKW 线段树	21
5.4.2	主席树	23
5.5	平衡树 (ct)	26
5.5.1	Splay	26
5.5.2	非旋转 Treap	28
5.5.3	可持久化平衡树	32
5.6	CDQ 分治 (ct)	36

6	Others	39
6.1	vimrc (gy)	39
6.2	Java Template (gy)	39
6.3	Big Fraction (gy)	42
6.4	模拟退火 (ct)	42
6.5	三分 (ct)	43
6.6	博弈论模型 (gy)	44

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)

```
1 int64_t mul(int64_t x, int64_t y, int64_t p) {
2     int64_t t = (x * y - (int64_t) ((long double) x / p * y + 1e-3) * p) % p;
3     return t < 0 ? t + p : t;
4 }
```

2.2 扩展欧几里得 (gy)

```
1 // return gcd(a, b)
2 // ax+by=gcd(a,b)
3 int extend_gcd(int a, int b, int &x, int &y) {
4     if (b == 0) {
5         x = 1, y = 0;
6         return a;
7     }
8     int res = extend_gcd(b, a % b, x, y);
9     int t = y;
10    y = x - a / b * y;
11    x = t;
12    return res;
13 }
14 // return minimal positive integer x so that ax+by=c
15 // or -1 if such x does not exist
16 int solve_equ(int a, int b, int c) {
17     int x, y, d;
18     d = extend_gcd(a, b, x, y);
19     if (c % d)
20         return -1;
21     int t = c / d;
22     x *= t;
23     y *= t;
24     int k = b / d;
25     x = (x % k + k) % k;
26     return x;
27 }
28 // return minimal positive integer x so that ax==b(mod p)
29 // or -1 if such x does not exist
30 int solve(int a, int b, int p) {
31     a = (a % p + p) % p;
32     b = (b % p + p) % p;
```

```

33     return solve_equ(a, p, b);
34 }

```

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

```

1 bool CRT(int &r1, int &m1, int r2, int m2) {
2     int x, y, g = extend_gcd(m1, m2, x, y);
3     if ((r2 - r1) % g != 0) return false;
4     x = 1ll * (r2 - r1) * x % m2;
5     if (x < 0) x += m2;
6     x /= g;
7     r1 += m1 * x;
8     m1 *= m2 / g;
9     return true;
10 }

```

2.4 线性同余不等式 (Durandal)

必须满足 $0 \leq d < m, 0 \leq l \leq r < m$, 返回 $\min\{x \geq 0 | l \leq x \cdot d \bmod m \leq r\}$, 无解返回 -1

```

1 int64_t calc(int64_t d, int64_t m, int64_t l, int64_t r) {
2     if (l == 0) return 0;
3     if (d == 0) return -1;
4     if (d * 2 > m) return calc(m - d, m, m - r, m - l);
5     if ((l - 1) / d < r / d) return (l - 1) / d + 1;
6     int64_t k = calc((-m % d + d) % d, d, l % d, r % d);
7     if (k == -1) return -1;
8     return (k * m + l - 1) / d + 1;
9 }

```

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)

```

1 /*
2  * In Java, use BigInteger.isProbablePrime(int certainty) to replace miller_rabin(BigInteger
  ↳ number)
3  * Test Set / First Wrong Answer
4  * 2 / 2,047
5  * 2, 3 / 1,373,653
6  * 31, 73 / 9,080,191
7  * 2, 3, 5 / 25,326,001
8  * 2, 3, 5, 7 / 3,215,031,751 (> Int.MAX_VALUE)
9  * 2, 7, 61 / 4,759,123,141
10 * 2, 13, 23, 1662803 / 1,122,004,669,633
11 * 2, 3, 5, 7, 11 / 2,152,302,898,747
12 * 2, 3, 5, 7, 11, 13 / 3,474,749,660,383

```



```

13  * 2, 3, 5, 7, 11, 13, 17 / 341,550,071,728,321
14  * 2, 3, 5, 7, 11, 13, 17, 19, 23 / 3,825,123,056,546,413,051
15  * 2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37 / 318,665,857,834,031,151,167,461 (> Long.MAX_VALUE)
16  * 2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37, 41 / 3,317,044,064,679,887,385,961,981
17  */
18  const int test_case_size = 12;
19  const int test_cases[test_case_size] = {2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37};

20  int64_t multiply_mod(int64_t x, int64_t y, int64_t p) {
21      int64_t t = (x * y - (int64_t) ((long double) x / p * y + 1e-3) * p) % p;
22      return t < 0 ? t + p : t;
23  }

24  int64_t add_mod(int64_t x, int64_t y, int64_t p) {
25      return (0ull + x + y) % p;
26  }

27  int64_t power_mod(int64_t x, int64_t exp, int64_t p) {
28      int64_t ans = 1;
29      while (exp) {
30          if (exp & 1)
31              ans = multiply_mod(ans, x, p);
32          x = multiply_mod(x, x, p);
33          exp >>= 1;
34      }
35      return ans;
36  }

37  bool miller_rabin_check(int64_t prime, int64_t base) {
38      int64_t number = prime - 1;
39      for (; ~number & 1; number >>= 1)
40          continue;
41      int64_t result = power_mod(base, number, prime);
42      for (; number != prime - 1 && result != 1 && result != prime - 1; number <<= 1)
43          result = multiply_mod(result, result, prime);
44      return result == prime - 1 || (number & 1) == 1;
45  }

46  bool miller_rabin(int64_t number) {
47      if (number < 2)
48          return false;
49      if (number < 4)
50          return true;
51      if (~number & 1)
52          return false;
53      for (int i = 0; i < test_case_size && test_cases[i] < number; i++)
54          if (!miller_rabin_check(number, test_cases[i]))
55              return false;
56      return true;
57  }

58  int64_t gcd(int64_t x, int64_t y) {
59      return y == 0 ? x : gcd(y, x % y);
60  }

61  int64_t pollard_rho_test(int64_t number, int64_t seed) {
62      int64_t x = rand() % (number - 1) + 1, y = x;
63      int head = 1, tail = 2;
64      while (true) {
65          x = multiply_mod(x, x, number);
66          x = add_mod(x, seed, number);

```

```

67     if (x == y)
68         return number;
69     int64_t answer = gcd(std::abs(x - y), number);
70     if (answer > 1 && answer < number)
71         return answer;
72     if (++head == tail) {
73         y = x;
74         tail <= 1;
75     }
76 }
77 }

78 void factorize(int64_t number, std::vector<int64_t> &divisor) {
79     if (number > 1) {
80         if (miller_rabin(number)) {
81             divisor.push_back(number);
82         } else {
83             int64_t factor = number;
84             while (factor >= number)
85                 factor = pollard_rho_test(number, rand() % (number - 1) + 1);
86             factorize(number / factor, divisor);
87             factorize(factor, divisor);
88         }
89     }
90 }

```

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 $\sum_{i=0}^{n-1} \frac{a+bi}{m}$

```

1 int64_t count(int64_t n, int64_t a, int64_t b, int64_t m) {
2     if (b == 0)
3         return n * (a / m);
4     if (a >= m)
5         return n * (a / m) + count(n, a % m, b, m);
6     if (b >= m)
7         return (n - 1) * n / 2 * (b / m) + count(n, a, b % m, m);

```

```
8 |     return count((a + b * n) / m, (a + b * n) % m, m, b);  
9 | }
```

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
 $c_1 = 1, c_{n+1} = \sum_{i=1}^n c_i \times c_{n+1-i}$
 $c_n = \frac{\binom{2n}{n}}{n+1}$
- Bell Number
- Bernoulli Number

2.17 积分表

Chapter 3

Geometry

3.1 点、直线、圆 (gy)

```
1  #include <cmath>
2  #include <algorithm>
3
4  using number = long double;
5  const number eps = 1e-8;
6
7  number _sqrt(number x) {
8      return std::sqrt(std::max(x, (number) 0));
9  }
10
11 number _asin(number x) {
12     x = std::min(x, (number) 1), x = std::max(x, (number) -1);
13     return std::asin(x);
14 }
15
16 number _acos(number x) {
17     x = std::min(x, (number) 1), x = std::max(x, (number) -1);
18     return std::acos(x);
19 }
20
21 int sgn(number x) {
22     return (x > eps) - (x < -eps);
23 }
24
25 int cmp(number x, number y) {
26     return sgn(x - y);
27 }
28
29 struct point {
30     number x, y;
31     point() {}
32     point(number x, number y) : x(x), y(y) {}
33
34     number len2() const {
35         return x * x + y * y;
36     }
37     number len() const {
38         return _sqrt(len2());
39     }
40     point unit() const {
41         return point(x / len(), y / len());
42     }
43     point rotate90() const {
44         return point(-y, x);
45     }
46 }
```

```

38     friend point operator+(const point &a, const point &b) {
39         return point(a.x + b.x, a.y + b.y);
40     }
41     friend point operator-(const point &a, const point &b) {
42         return point(a.x - b.x, a.y - b.y);
43     }
44     friend point operator*(const point &a, number b) {
45         return point(a.x * b, a.y * b);
46     }
47     friend point operator/(const point &a, number b) {
48         return point(a.x / b, a.y / b);
49     }
50     friend number dot(const point &a, const point &b) {
51         return a.x * b.x + a.y * b.y;
52     }
53     friend number det(const point &a, const point &b) {
54         return a.x * b.y - a.y * b.x;
55     }
56     friend number operator==(const point &a, const point &b) {
57         return cmp(a.x, b.x) == 0 && cmp(a.y, b.y) == 0;
58     }
59 };

60 number dis2(const point &a, const point &b) {
61     return (a - b).len2();
62 }
63 number dis(const point &a, const point &b) {
64     return (a - b).len();
65 }

66 struct line {
67     point a, b;
68     line() {}
69     line(point a, point b) : a(a), b(b) {}
70     point value() const {
71         return b - a;
72     }
73 };

74 bool point_on_line(const point &p, const line &l) {
75     return sgn(det(p - l.a, p - l.b)) == 0;
76 }
77 // including endpoint
78 bool point_on_ray(const point &p, const line &l) {
79     return sgn(det(p - l.a, p - l.b)) == 0 &&
80         sgn(dot(p - l.a, l.b - l.a)) >= 0;
81 }
82 // including endpoints
83 bool point_on_seg(const point &p, const line &l) {
84     return sgn(det(p - l.a, p - l.b)) == 0 &&
85         sgn(dot(p - l.a, l.b - l.a)) >= 0 &&
86         sgn(dot(p - l.b, l.a - l.b)) >= 0;
87 }
88 bool seg_has_intersection(const line &a, const line &b) {
89     if (point_on_seg(a.a, b) || point_on_seg(a.b, b) ||
90         point_on_seg(b.a, a) || point_on_seg(b.b, a))
91         return /* including endpoints */ true;
92     return sgn(det(a.a - b.a, b.b - b.a)) * sgn(det(a.b - b.a, b.b - b.a)) < 0
93         && sgn(det(b.a - a.a, a.b - a.a)) * sgn(det(b.b - a.a, a.b - a.a)) < 0;
94 }

```

```

95 point intersect(const line &a, const line &b) {
96     number s1 = det(a.b - a.a, b.a - a.a);
97     number s2 = det(a.b - a.a, b.b - a.a);
98     return (b.a * s2 - b.b * s1) / (s2 - s1);
99 }
100 point projection(const point &p, const line &l) {
101     return l.a + (l.b - l.a) * dot(p - l.a, l.b - l.a) / (l.b - l.a).len2();
102 }
103 number dis(const point &p, const line &l) {
104     return std::abs(dot(p - l.a, l.b - l.a)) / (l.b - l.a).len();
105 }
106 point symmetry_point(const point &a, const point &o) {
107     return o + o - a;
108 }
109 point reflection(const point &p, const line &l) {
110     return symmetry_point(p, projection(p, l));
111 }
112 struct circle {
113     point o;
114     number r;
115     circle() {}
116     circle(point o, number r) : o(o), r(r) {}
117 };
118 bool intersect(const line &l, const circle &a, point &p1, point &p2) {
119     number x = dot(l.a - a.o, l.b - l.a);
120     number y = (l.b - l.a).len2();
121     number d = x * x - y * ((l.a - a.o).len2() - a.r * a.r);
122     if (sgn(d) < 0) return false;
123     point p = l.a - (l.b - l.a) * (x / y), delta = (l.b - l.a) * (_sqrt(d) / y);
124     p1 = p + delta, p2 = p - delta;
125     return true;
126 }
127 bool intersect(const circle &a, const circle &b, point &p1, point &p2) {
128     if (a.o == b.o && cmp(a.r, b.r) == 0)
129         return /* value for coincident circles */ false;
130     number s1 = (b.o - a.o).len();
131     if (cmp(s1, a.r + b.r) > 0 || cmp(s1, std::abs(a.r - b.r)) < 0)
132         return false;
133     number s2 = (a.r * a.r - b.r * b.r) / s1;
134     number aa = (s1 + s2) / 2, bb = (s1 - s2) / 2;
135     point p = (b.o - a.o) * (aa / (aa + bb)) + a.o;
136     point delta = (b.o - a.o).unit().rotate90() * _sqrt(a.r * a.r - aa * aa);
137     p1 = p + delta, p2 = p - delta;
138     return true;
139 }
140 bool tangent(const point &p0, const circle &c, point &p1, point &p2) {
141     number x = (p0 - c.o).len2();
142     number d = x - c.r * c.r;
143     if (sgn(d) < 0) return false;
144     if (sgn(d) == 0)
145         return /* value for point_on_line */ false;
146     point p = (p0 - c.o) * (c.r * c.r / x);
147     point delta = ((p0 - c.o) * (-c.r * _sqrt(d) / x)).rotate90();
148     p1 = c.o + p + delta;
149     p2 = c.o + p - delta;
150     return true;
151 }
152 bool ex_tangent(const circle &a, const circle &b, line &l1, line &l2) {
153     if (cmp(std::abs(a.r - b.r), (b.o - a.o).len()) == 0) {

```

```

154     point p1, p2;
155     intersect(a, b, p1, p2);
156     l1 = l2 = line(p1, p1 + (a.o - p1).rotate90());
157     return true;
158 } else if (cmp(a.r, b.r) == 0) {
159     point dir = b.o - a.o;
160     dir = (dir * (a.r / dir.len())).rotate90();
161     l1 = line(a.o + dir, b.o + dir);
162     l2 = line(a.o - dir, b.o - dir);
163     return true;
164 } else {
165     point p = (b.o * a.r - a.o * b.r) / (a.r - b.r);
166     point p1, p2, q1, q2;
167     if (tangent(p, a, p1, p2) && tangent(p, b, q1, q2)) {
168         l1 = line(p1, q1);
169         l2 = line(p2, q2);
170         return true;
171     } else {
172         return false;
173     }
174 }
175 }
176 bool in_tangent(const circle &a, const circle &b, line &l1, line &l2) {
177     if (cmp(a.r + b.r, (b.o - a.o).len()) == 0) {
178         point p1, p2;
179         intersect(a, b, p1, p2);
180         l1 = l2 = line(p1, p1 + (a.o - p1).rotate90());
181         return true;
182     } else {
183         point p = (b.o * a.r + a.o * b.r) / (a.r + b.r);
184         point p1, p2, q1, q2;
185         if (tangent(p, a, p1, p2) && tangent(p, b, q1, q2)) {
186             l1 = line(p1, q1);
187             l2 = line(p2, q2);
188             return true;
189         } else {
190             return false;
191         }
192     }
193 }

```

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)

```
1 //
2 // Title: Modui
3 // Date: 26.02.2016
4 // Test: BZOJ-2038
5 // Complexity:  $O(n^{3/2})$ 
6 //
7 /*
8     莫队算法——将所有询问储存起来，然后分块暴力处理。
9     时间复杂度为  $O(n \times \sqrt{n})$ 。
10 */
11 #include <cstdio>
12 #include <cstring>
13 #include <algorithm>
14 #include <cmath>
15 #ifdef WIN32
16     #define LL "%I64d"
17 #else
18     #define LL "%lld"
19 #endif
20 #ifdef CT
21     #define debug(...) printf(__VA_ARGS__)
22 #else
23     #define debug(...)
24 #endif
25 #define R register
26 #define getc() (S==T&&(T=(S=B)+fread(B,1,1<<15,stdin),S==T)?EOF:*S++)
27 #define gmax(_a, _b) ((_a) > (_b) ? (_a) : (_b))
28 #define gmin(_a, _b) ((_a) < (_b) ? (_a) : (_b))
29 #define cmax(_a, _b) (_a < (_b) ? _a = (_b) : 0)
30 #define cmin(_a, _b) (_a > (_b) ? _a = (_b) : 0)
31 char B[1<<15], *S=B, *T=B;
32 inline int FastIn()
33 {
34     R char ch; R int cnt=0; R bool minus=0;
35     while (ch=getc(), (ch < '0' || ch > '9') && ch != '-') ;
36     ch == '-' ? minus=1: cnt=ch-'0';
37     while (ch=getc(), ch >= '0' && ch <= '9') cnt = cnt * 10 + ch - '0';
38     return minus?-cnt:cnt;
39 }
```

```

40 #define maxn 50010
41 int col[maxn], num[maxn], size, pos[maxn];
42 long long up[maxn], dw[maxn], ans;
43 struct Query{
44     int l, r, id;
45 }q[maxn];
46 inline bool cmp(const Query &i, const Query &j){
47     return pos[i.l] != pos[j.l] ? (i.l < j.l) : (pos[i.l] & 1 ? i.r < j.r : i.r > j.r);
48 }
49 inline long long gcd(R long long a, R long long b){
50     R long long tmp;
51     while (b){
52         tmp = b;
53         b = a % b;
54         a = tmp;
55     }
56     return a;
57 }
58 inline void update(R int x, R int d){
59     ans -= num[col[x]] * num[col[x]];
60     num[col[x]] += d;
61     ans += num[col[x]] * num[col[x]];
62 }
63 int main()
64 {
65     R int n = FastIn(), m = FastIn(); size = (int)sqrt(n * 1.0);
66     for (R int i = 1; i <= n; i++) col[i] = FastIn(), pos[i] = (i - 1) / size + 1;
67     for (R int i = 1; i <= m; i++){
68         q[i].l = FastIn(); q[i].r = FastIn(); q[i].id = i;
69     }
70     std::sort(q + 1, q + m + 1, cmp);
71     R int l = 1, r = 0;
72     for (R int i = 1; i <= m; i++){
73         R int id_now = q[i].id;
74         if (q[i].l == q[i].r){
75             up[id_now] = 0; dw[id_now] = 1; continue;
76         }
77         for (; r < q[i].r; r++) update(r + 1, 1);
78         for (; r > q[i].r; r--) update(r, -1);
79         for (; l < q[i].l; l++) update(l, -1);
80         for (; l > q[i].l; l--) update(l - 1, 1);
81         R long long aa, bb, cc;
82         aa = ans - q[i].r * q[i].l - 1;
83         bb = (long long)(q[i].r - q[i].l + 1) * (q[i].r - q[i].l);
84         cc = gcd(aa, bb); aa /= cc; bb /= cc;
85         up[id_now] = aa; dw[id_now] = bb;
86     }
87     for (R int i = 1; i <= m; i++) printf("%lld/%lld\n", up[i], dw[i]);
88     return 0;
89 }

```

5.2 ST 表 (ct)

```

1 #include <stdio>
2 #define dmax(_a, _b) ((_a) > (_b) ? (_a) : (_b))
3 #define maxn 200010
4 int a[maxn], f[20][maxn], n;

```

```

5 int Log[maxn];
6 void build()
7 {
8     for (int i = 1; i <= n; ++i) f[0][i] = a[i];
9
10    int lim = Log[n];
11    for (int j = 1; j <= lim; ++j)
12    {
13        int *fj = f[j], *fj1 = f[j - 1];
14        for (int i = 1; i <= n - (1 << j) + 1; ++i)
15            fj[i] = dmax(fj1[i], fj1[i + (1 << (j - 1))]);
16    }
17 int Query(int l, int r)
18 {
19     int k = Log[r - l + 1];
20     return dmax(f[k][l], f[k][r - (1 << k) + 1]);
21 }
22 int main()
23 {
24     scanf("%d", &n);
25     Log[0] = -1;
26     for (int i = 1; i <= n; ++i)
27     {
28         scanf("%d", &a[i]);
29         Log[i] = Log[i >> 1] + 1;
30     }
31     build();
32     int q;
33     scanf("%d", &q);
34     for (; q; --q)
35     {
36         int l, r; scanf("%d%d", &l, &r);
37         printf("%d\n", Query(l, r));
38     }
39 }

```

5.3 可并堆 (ct)

```

1 struct Node {
2     Node *ch[2];
3     ll val; int size;
4     inline void update()
5     {
6         size = ch[0] -> size + ch[1] -> size + 1;
7     }
8 } mem[maxn], *rt[maxn];
9 Node *merge(Node *a, Node *b)
10 {
11     if (a == mem) return b;
12     if (b == mem) return a;
13     if (a -> val < b -> val) std::swap(a, b);
14     std::swap(a -> ch[0], a -> ch[1]);
15     a -> ch[1] = merge(a -> ch[1], b);
16     a -> update();
17     return a;
18 }

```

5.4 线段树 (ct)

5.4.1 ZKW 线段树

```

1 <<<<<< HEAD
2 //
3 // Title:ZKW Segment Tree
4 // Date:19.11.2015
5 // Complexity:
6 // Build Tree: $O(N)$ 
7 // Query: $O(\log N)$ 
8 // Change: $O(\log N)$ 
9 #include<cstdio>
10 #include<cmath>
11 #define maxn 100000
12 #define R register
13 int T[1<<18|1],n,m,M;
14 inline int FastIn()
15 {
16     R char ch=getchar();R int cnt=0;R bool minus=0;
17     while ((ch<'0' || ch>'9')&&ch!='-') ch=getchar();
18     if (ch=='-') minus=1,ch=getchar();
19     while (ch>='0'&&ch<='9') cnt=cnt*10+ch-'0',ch=getchar();
20     return minus?-cnt:cnt;
21 }
22 inline void Build_Tree()
23 {
24     for (R int i=M-1;i>=1;i--)
25         T[i]=T[2*i]+T[2*i+1];
26 }
27 inline int Query(int s,int t)
28 {
29     R int Ans;
30     for (Ans=0,s=s+M-1,t=t+M+1;s^t^1;s>>=1,t>>=1)
31     {
32         if (~s&1) Ans+=T[s^1];
33         if (t&1) Ans+=T[t^1];
34     }
35     return Ans;
36 }
37 inline void Change(int x,int NewValue)
38 {
39     R int i=M+x;
40     for (T[i]=NewValue,i>>=1;i;i>>=1)
41         T[i]=T[2*i]+T[2*i+1];
42 }
43 int main()
44 {
45     n=FastIn();m=FastIn();
46     for (M=1;M<=n;M<<=1);
47     for (R int i=0;i<n;i++)
48         T[M+i]=FastIn();
49     Build_Tree();
50     for (R int i=1;i<=m;i++)

```

```

51     {
52         R char cmd=getchar();
53         if (cmd=='Q')
54         {
55             R int a=FastIn()-1,b=FastIn()-1;
56             printf("%d\n",Query(a,b));
57         }
58         if (cmd=='M')
59         {
60             R int a=FastIn()-1,b=FastIn();
61             Change(a,b);
62         }
63     }
64     return 0;
65 }
66 =====
67 //
68 // Title:ZKW Segment Tree
69 // Date:19.11.2015
70 // Complexity:
71 // Build Tree:O(N)
72 // Query:O(logN)
73 // Change:O(logN)
74 #include<cstdio>
75 #include<cmath>
76 #define maxn 100000
77 #define R register
78 int T[1<<18|1],n,m,M;
79 inline int FastIn()
80 {
81     R char ch=getchar();R int cnt=0;R bool minus=0;
82     while ((ch<'0' || ch>'9') && ch!='-') ch=getchar();
83     if (ch=='-') minus=1,ch=getchar();
84     while (ch>='0' && ch<='9') cnt=cnt*10+ch-'0',ch=getchar();
85     return minus?-cnt:cnt;
86 }
87 inline void Build_Tree()
88 {
89     for (R int i=M-1;i>=1;i--)
90         T[i]=T[2*i]+T[2*i+1];
91 }
92 inline int Query(int s,int t)
93 {
94     R int Ans;
95     for (Ans=0,s=s+M-1,t=t+M+1;s^t^1;s>>=1,t>>=1)
96     {
97         if (~s&1) Ans+=T[s^1];
98         if (t&1) Ans+=T[t^1];
99     }
100     return Ans;
101 }
102 inline void Change(int x,int NewValue)
103 {
104     R int i=M+x;
105     for (T[i]=NewValue,i>>=1;i>>=1)
106         T[i]=T[2*i]+T[2*i+1];

```

```

107 }
108 int main()
109 {
110     n=FastIn();m=FastIn();
111     for (M=1;M<=n;M<=1);
112     for (R int i=0;i<n;i++)
113         T[M+i]=FastIn();
114     Build_Tree();
115     for (R int i=1;i<=m;i++)
116     {
117         R char cmd=getchar();
118         if (cmd=='Q')
119         {
120             R int a=FastIn()-1,b=FastIn()-1;
121             printf("%d\n",Query(a,b));
122         }
123         if (cmd=='M')
124         {
125             R int a=FastIn()-1,b=FastIn();
126             Change(a,b);
127         }
128     }
129     return 0;
130 }
131 >>>>>> 49188fa6ef8b175c2f4a6388509d8dc5116ebccd

```

5.4.2 主席树

```

1 <<<<<<< HEAD
2 //
3 // Title: Functional Segment Tree
4 // Date:16.12.2015
5 // Complexity:O((n+m)logn)
6 // Test:YZ0J-1991
7 #include<cstdio>
8 #include<algorithm>
9 #define maxt 2000010
10 #define maxn 100010
11 #define R register
12 inline int FastIn(){
13     R char ch=getchar();R int cnt=0;
14     while (ch<'0' || ch>'9') ch=getchar();
15     while (ch>='0' && ch<='9') cnt=cnt*10+ch-'0',ch=getchar();
16     return cnt;
17 }
18 int ls[maxt],
19     rs[maxt],
20     count[maxt],
21     root[maxn],
22     tot;
23 int num[maxn],rank[maxn],n,m,r[maxn];
24 bool cmp(const int &i,const int &j){
25     return num[i]<num[j];
26 }
27 inline void Insert(int last,int left,int right,int pre)

```

```

28 {
29     count[++tot]=count[last]+1;
30     if (left==right) return;
31     R int mid=(left+right)>>1;
32     if (pre>mid){
33         rs[tot]=tot+1;
34         Insert(rs[last],mid+1,right,pre);
35     }
36     else{
37         ls[tot]=tot+1;
38         rs[tot]=rs[last];
39         Insert(ls[last],left,mid,pre);
40     }
41 }

42 inline int Query(int a,int b,int k)
43 {
44     R int l=1,r=n,mid,f1=a,f2=b,cnt,kk=k;
45     while (l<r){
46         mid=(l+r)>>1;cnt=count[ls[f2]]-count[ls[f1]];
47         if (cnt>=kk) f1=ls[f1],f2=ls[f2],r=mid;
48         else f1=rs[f1],f2=rs[f2],l=mid+1,kk-=cnt;
49     }
50     return l;
51 }

52 int main()
53 {
54     n=FastIn();m=FastIn();R int i,a,b,k;
55     for (i=1;i<=n;i++) num[i]=FastIn(),rank[i]=i;
56     std::sort(rank+1,rank+n+1,cmp);
57     std::sort(num+1,num+n+1);
58     for (i=1;i<=n;i++) r[rank[i]]=i;
59     for (i=1;i<=n;i++) {
60         root[i]=tot+1;
61         Insert(root[i-1],1,n,r[i]);
62     }
63     for (i=1;i<=m;i++){
64         a=FastIn();b=FastIn();k=FastIn();
65         printf("%d\n",num[Query(root[a-1],root[b],k)]);
66     }
67     return 0;
68 }
69 =====
70 //
71 // Title: Functional Segment Tree
72 // Date:16.12.2015
73 // Complexity:O((n+m)logn)
74 // Test:YZ0J-1991
75 #include<cstdio>
76 #include<algorithm>
77 #define maxt 2000010
78 #define maxn 100010
79 #define R register
80 inline int FastIn(){
81     R char ch=getchar();R int cnt=0;
82     while (ch<'0' || ch>'9') ch=getchar();
83     while (ch>='0' && ch<='9') cnt=cnt*10+ch-'0',ch=getchar();
84     return cnt;
85 }

```



```

86 int ls[maxt],
87    rs[maxt],
88    count[maxt],
89    root[maxn],
90    tot;

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

92 bool cmp(const int &i,const int &j){
93     return num[i]<num[j];
94 }

95 inline void Insert(int last,int left,int right,int pre)
96 {
97     count[++tot]=count[last]+1;
98     if (left==right) return;
99     R int mid=(left+right)>>1;
100    if (pre>mid){
101        rs[tot]=tot+1;
102        Insert(rs[last],mid+1,right,pre);
103    }
104    else{
105        ls[tot]=tot+1;
106        rs[tot]=rs[last];
107        Insert(ls[last],left,mid,pre);
108    }
109 }

110 inline int Query(int a,int b,int k)
111 {
112     R int l=1,r=n,mid,f1=a,f2=b,cnt,kk=k;
113     while (l<r){
114         mid=(l+r)>>1;cnt=count[ls[f2]]-count[ls[f1]];
115         if (cnt>=kk) f1=ls[f1],f2=ls[f2],r=mid;
116         else f1=rs[f1],f2=rs[f2],l=mid+1,kk-=cnt;
117     }
118     return l;
119 }

120 int main()
121 {
122     n=FastIn();m=FastIn();R int i,a,b,k;
123     for (i=1;i<=n;i++) num[i]=FastIn(),rank[i]=i;
124     std::sort(rank+1,rank+n+1,cmp);
125     std::sort(num+1,num+n+1);
126     for (i=1;i<=n;i++) r[rank[i]]=i;
127     for (i=1;i<=n;i++) {
128         root[i]=tot+1;
129         Insert(root[i-1],1,n,r[i]);
130     }
131     for (i=1;i<=m;i++){
132         a=FastIn();b=FastIn();k=FastIn();
133         printf("%d\n",num[Query(root[a-1],root[b],k)]);
134     }
135     return 0;
136 }
137 >>>>>> 49188fa6ef8b175c2f4a6388509d8dc5116ebccd

```

5.5 平衡树 (ct)

5.5.1 Splay

```

1 //
2 // Title : Splay Tree
3 // Date : 11.01.2016
4 // Complexity :  $O(n \log n)$  (期望)
5 // Test : BZOJ-1251
6 /*
7 */
8 #include <cstdio>
9 #include <cstring>
10 #include <algorithm>
11 #include <cmath>
12
13 #ifdef WIN32
14     #define LL "%I64d"
15 #else
16     #define LL "%lld"
17 #endif
18
19 #ifdef CT
20     #define debug(...) printf(__VA_ARGS__)
21 #else
22     #define debug(...)
23 #endif
24
25 #define R register
26 #define getc() (S==T&&(T=(S=B)+fread(B,1,1<<15,stdin),S==T)?EOF:*S++)
27 #define gmax(_a, _b) ((_a) > (_b) ? (_a) : (_b))
28 #define gmin(_a, _b) ((_a) < (_b) ? (_a) : (_b))
29 #define cmax(_a, _b) (_a < (_b) ? _a = (_b) : 0)
30 #define cmin(_a, _b) (_a > (_b) ? _a = (_b) : 0)
31
32 char B[1<<15], *S=B, *T=B;
33 inline int FastIn()
34 {
35     R char ch; R int cnt=0; R bool minus=0;
36     while (ch=getc(), (ch < '0' || ch > '9') && ch != '-') ;
37     ch == '-' ? minus=1 : cnt=ch-'0';
38     while (ch=getc(), ch >= '0' && ch <= '9') cnt = cnt * 10 + ch - '0';
39     return minus?-cnt:cnt;
40 }
41
42 #define maxn 50010
43 int n, Q, root;
44 int fa[maxn], ch[maxn][2], id[maxn], size[maxn];
45 int tag[maxn], mx[maxn], num[maxn];
46 bool rev[maxn];
47 inline void update(int x){
48     R int ls=ch[x][0], rs=ch[x][1];
49     mx[x]=num[x];
50     cmax(mx[x], mx[ls]); cmax(mx[x], mx[rs]);
51     size[x]=size[ls]+size[rs]+1;
52 } //更新
53 void build(int l, int r, int rt){
54     if (l>r) return ;
55     R int mid=l+r>>1;
56     fa[mid]=rt;
57     if (mid<rt) ch[rt][0]=mid;
58     else ch[rt][1]=mid;

```

```

54     build(l,mid-1,mid);
55     build(mid+1,r,mid);
56     update(mid);
57 }//建树
58 inline void pushdown(int x){
59     R int ls=ch[x][0],rs=ch[x][1];
60     if (tag[x]){
61         R int lazy=tag[x];
62         if (ls) tag[ls]+=lazy,num[ls]+=lazy,mx[ls]+=lazy;
63         if (rs) tag[rs]+=lazy,num[rs]+=lazy,mx[rs]+=lazy;
64         tag[x]=0;
65     }
66     if (rev[x]){
67         if (ls) rev[ls]^=1;
68         if (rs) rev[rs]^=1;
69         ch[x][1]=ls,ch[x][0]=rs;
70         rev[x]=0;
71     }
72 }//具体下传的过程
73 inline void rotate(int x){//把 x 向上旋转到 x 的父亲
74     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
75     if (f==root) root=x,ch[0][0]=x;
76     (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
77     (fa[x]=gf)>0 ? ch[gf][ch[gf][1]==f]=x : 0; //把 x 代替 f 的位置
78     fa[ch[x][d^1]=f]=x; //把 f 接到 x 的下面
79     update(f); //更新 f 节点
80 }
81 inline void splay(int x,int rt){//把 x 旋转到 rt
82     while (fa[x]!=rt){
83         R int f=fa[x],gf=fa[f];
84         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
85         rotate(x);
86     }
87     update(x);
88 }
89 int find(int x,int rank){
90     if (tag[x]||rev[x]) pushdown(x);
91     R int ls=ch[x][0],rs=ch[x][1],lsize=size[ls];
92     if (lsize+1==rank) return x;
93     if (lsize>=rank) return find(ls,rank);
94     else return find(rs,rank-lsize-1);
95 }//找第 k 小
96 inline int prepare(int l,int r){
97     R int x=find(root,l-1);
98     splay(x,0);
99     x=find(root,r+1);
100    splay(x,root);
101    return ch[x][0];
102 }//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
103 inline void add(int l,int r,int w){
104     R int x=prepare(l,r);
105     tag[x]+=w,num[x]+=w,mx[x]+=w;
106 }//区间加
107 inline void rever(int l,int r){
108     R int x=prepare(l,r);
109     rev[x]^=1;
110 }//区间翻转
111 inline void query(int l,int r){
112     R int x=prepare(l,r);
113     printf("%d\n",mx[x] );

```

```

114 }//区间查询最大值
115 inline int split(R int k){
116     R int ls;
117     if (k<size[root])
118     {
119         R int kth=find(root,k+1);
120         splay(kth);ls=ch[kth][0];
121         fa[ls]=0;ch[kth][0]=0;
122         size[kth]-=size[ls];
123     }
124     else{
125         ls=root;root=0;
126     }
127     return ls;
128 }//删除数列
129 inline void merge(R int nwrt){
130     if (!root) {root=nwrt;return;}
131     R int nw=find(root,1);
132     splay(nw);fa[nwrt]=nw;ch[nw][0]=nwrt;
133     size[nw]+=size[nwrt];
134 }//合并数列
135 int main()
136 {
137     n=FastIn()+2;Q=FastIn();R int i,l,r,v,cmd;mx[0]=-23333333;
138     build(1,n,0);root=(1+n)>>1;
139     for (;Q--;){
140         cmd=FastIn();l=FastIn()+1;r=FastIn()+1;
141         if (cmd==1) v=FastIn(),add(l,r,v);
142         else if (cmd==2) rever(l,r);
143         else query(l,r);
144     }
145     return 0;
146 }

```

5.5.2 非旋转 Treap

```

1  //
2  // Title : Treap (unrotated)
3  // Date : 13.04.2016
4  // Test : BZOJ-3224
5  // Complexity :  $O(n \log n)$  (期望)
6  //
7  /*
8      对于序列上的一些操作的问题——
9      解决办法：平衡树 Treap
10 */
11 #include <cstdio>
12 #include <cstring>
13 #include <algorithm>
14 #include <cmath>
15
16 #ifdef WIN32
17     #define LL "%I64d"
18 #else
19     #define LL "%lld"
20 #endif
21
22 #ifdef CT
23     #define debug(...) printf(__VA_ARGS__)
24     #define setfile()

```

```

23 #else
24     #define debug(...)
25     #define filename ""
26     #define setfile() freopen(filename".in", "r", stdin); freopen(filename".out", "w", stdout);
27 #endif
28 #define R register
29 #define getc() (S == T && (T = (S = B) + fread(B, 1, 1 << 15, stdin), S == T) ? EOF : *S++)
30 #define dmax(_a, _b) ((_a) > (_b) ? (_a) : (_b))
31 #define dmin(_a, _b) ((_a) < (_b) ? (_a) : (_b))
32 #define cmax(_a, _b) (_a < (_b) ? _a = (_b) : 0)
33 #define cmin(_a, _b) (_a > (_b) ? _a = (_b) : 0)
34 char B[1 << 15], *S = B, *T = B;
35 inline int FastIn()
36 {
37     R char ch; R int cnt = 0; R bool minus = 0;
38     while (ch = getc(), (ch < '0' || ch > '9') && ch != '-') ;
39     ch == '-' ? minus = 1 : cnt = ch - '0';
40     while (ch = getc(), ch >= '0' && ch <= '9') cnt = cnt * 10 + ch - '0';
41     return minus ? -cnt : cnt;
42 }
43 const int Ta = 1 << 16 | 3, Tb = 33333331;
44 int Tc;
45 inline int randint() {return Tc = Ta * Tc + Tb;}
46 struct Treap
47 {
48     int data, key, size;
49     Treap *ls, *rs;
50     Treap(int _val):data(_val), key(randint()), ls(NULL), rs(NULL), size(1){}
51     inline void update()
52     {
53         size = (ls ? ls -> size : 0) + (rs ? rs -> size : 0) + 1;
54     }
55 }*root;
56 inline int Size(Treap *x)
57 {
58     return x ? x -> size : 0;
59 }
60 //为了防止访问到空节点, 定义一个函数来访问 size
61 struct Pair
62 {
63     Treap *fir, *sec;
64 };
65 Treap *Merge(Treap *a, Treap *b)
66 {
67     if (!a) return b;
68     if (!b) return a;
69     if (a -> key < b -> key)
70     {
71         a -> rs = Merge(a -> rs, b);
72         a -> update();
73         return a;
74     }
75     else
76     {
77         b -> ls = Merge(a, b -> ls);
78         b -> update();
79         return b;
80     }
81 }
82 //按照 a, b 的顺序来合并两棵 Treap

```

```

83 Pair Split(Treap *x, int k)
84 {
85     if (!x) return (Pair){NULL, NULL};
86     Pair y; y.fir = NULL; y.sec = NULL;
87     if (Size(x -> ls) >= k)
88     {
89         y = Split(x -> ls, k);
90         x -> ls = y.sec;
91         x -> update();
92         y.sec = x;
93     }
94     else
95     {
96         y = Split(x -> rs, k - Size(x -> ls) - 1);
97         x -> rs = y.fir;
98         x -> update();
99         y.fir = x;
100    }
101    return y;
102 }
103 //将前 k 个的点分离出来
104 inline int Find(R int k)
105 {
106     Pair x = Split(root, k - 1);
107     Pair y = Split(x.sec, 1);
108     Treap *ans = y.fir;
109     root = Merge(Merge(x.fir, ans), y.sec);
110     return ans -> data;
111 }
112 //找到第 k 小的 data 值
113 int Get(Treap *x, R int val)
114 {
115     if (!x) return 0;
116     return val < x -> data ? Get(x -> ls, val) : Get(x -> rs, val) + Size(x -> ls) + 1;
117 }
118 //找到 val 的排名
119 inline void Insert(R int val)
120 {
121     R int k = Get(root, val);
122     Pair x = Split(root, k);
123     Treap *pre = new Treap(val);
124     root = Merge(Merge(x.fir, pre), x.sec);
125 }
126 //插入
127 inline void Delete(R int val)
128 {
129     R int k = Get(root, val);
130     Pair x = Split(root, k - 1);
131     Pair y = Split(x.sec, 1);
132     root = Merge(x.fir, y.sec);
133 }
134 //单点删除
135 inline int upper(R int val)
136 {
137     R int ans = 1e9;
138     Treap *tmp = root;
139     while (tmp)
140     {
141         if (tmp -> data > val)
142         {
143             cmin(ans, tmp -> data);

```

```

144         tmp = tmp -> ls;
145     }
146     else
147         tmp = tmp -> rs;
148 }
149 return ans;
150 }
151 inline int lower(R int val)
152 {
153     R int ans = -1e9;
154     Treap *tmp = root;
155     while (tmp)
156     {
157         if (tmp -> data < val)
158         {
159             cmax(ans, tmp -> data);
160             tmp = tmp -> rs;
161         }
162         else tmp = tmp -> ls;
163     }
164     return ans;
165 }
166 void print(Treap *x)
167 {
168     if (!x) return;
169     print(x -> ls);
170     printf("%d ", x -> data );
171     print(x -> rs);
172 }
173 int main()
174 {
175     root = NULL;
176     for (R int Q = FastIn(); Q; --Q)
177     {
178         R int opt = FastIn(), x = FastIn();
179         if (opt == 1) Insert(x);
180         else if (opt == 2) Delete(x);
181         else if (opt == 3)
182         {
183             R int ans = Get(root, x);
184             while (ans > 1 && Find(ans - 1) == x) ans--;
185             printf("%d\n", ans );
186         }
187         else if (opt == 4) printf("%d\n", Find(x) );
188         else if (opt == 5) printf("%d\n", lower(x) );
189         else printf("%d\n", upper(x) );
190     }
191     return 0;
192 }
193 /*
194 input:
195 10
196 1 106465
197 4 1
198 1 317721
199 1 460929
200 1 644985
201 1 84185
202 1 89851
203 6 81968
204 1 492737

```

```

205 5 493598
206 output:
207 106465
208 84185
209 492737
210 input2:
211 5
212 1 1
213 1 1
214 1 1
215 1 2
216 3 1
217 output2:
218 1
219 */

```

5.5.3 可持久化平衡树

```

1  //
2  // Title: Functional Treap
3  // Date: 16.04.2016
4  // Test: YZOJ-1620
5  // Complexity:  $O(n \log n)$  (期望)
6  //
7  /*
8   可持久化 Treap:
9   用来解决超级编辑器等问题。
10   优势: 好写好调好理解的平衡树
11   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
12  */
13 #include <cstdio>
14 #include <cstring>
15 #include <algorithm>
16 #include <cmath>
17 #ifdef WIN32
18     #define LL "%I64d"
19 #else
20     #define LL "%lld"
21 #endif
22 #ifdef CT
23     #define debug(...) printf(__VA_ARGS__)
24     #define setfile()
25 #else
26     #define debug(...)
27     #define filename ""
28     #define setfile() freopen(filename".in", 'r', stdin); freopen(filename".out", 'w', stdout)
29 #endif
30 #define R register
31 // #define getc() (S==EOF?(T=(S=B)+fread(B,1,1<<15,stdin),S==T)?EOF:*S++)
32 #define getc() getchar()
33 #define dmax(_a, _b) ((_a) > (_b) ? (_a) : (_b))
34 #define dmin(_a, _b) ((_a) < (_b) ? (_a) : (_b))
35 #define cmax(_a, _b) (_a < (_b) ? _a = (_b) : 0)
36 #define cmin(_a, _b) (_a > (_b) ? _a = (_b) : 0)
37 #define cabs(_x) ((_x)<0?(-_x):(_x))

```



```

38 char B[1<<15],*S=B,*T=B;
39 inline int FastIn()
40 {
41     R char ch;R int cnt=0;R bool minus=0;
42     while (ch=getc(),(ch < '0' || ch > '9') && ch != '-') ;
43     ch == '-' ? minus=1:cnt=ch-'0';
44     while (ch=getc(),ch >= '0' && ch <= '9') cnt = cnt * 10 + ch - '0';
45     return minus?-cnt:cnt;
46 }
47 #define maxn 100010
48 char str[maxn];
49 struct Treap
50 {
51     char data;
52     int size;
53     Treap *ls, *rs;
54     Treap(char _ch): data(_ch), size(1), ls(NULL), rs(NULL){}
55     inline void update()
56     {
57         size = (ls ? ls -> size : 0) + (rs ? rs -> size : 0) + 1;
58     }
59 }*root[maxn];
60 inline int Size(Treap *x)
61 {
62     return x ? x -> size : 0;
63 }
64 struct Pair
65 {
66     Treap *fir, *sec;
67 };
68 inline Treap *copy(Treap *x)
69 {
70     if (!x) return NULL;
71     Treap *nw = new Treap(x -> data);
72     nw -> ls = x -> ls;
73     nw -> rs = x -> rs;
74     nw -> size = x -> size;
75     return nw;
76 }
77 Pair Split(Treap *x, int k)
78 {
79     if (!x) return (Pair){NULL, NULL};
80     Pair y; y.fir = NULL; y.sec = NULL;
81     Treap *nw = copy(x);
82     if (Size(nw -> ls) >= k)
83     {
84         y = Split(nw -> ls, k);
85         nw -> ls = y.sec;
86         nw -> update();
87         y.sec = nw;
88     }
89     else
90     {
91         y = Split(nw -> rs, k - Size(nw -> ls) - 1);
92         nw -> rs = y.fir;
93         nw -> update();
94         y.fir = nw;
95     }
96     return y;
97 }
98 const int Ta = 1 << 16 | 3, Tb = 33333331;

```

```

99 unsigned int Tc;
100 inline unsigned int randint(){return Tc = Ta * Tc + Tb;}
101 Treap *Merge(Treap *a, Treap *b)
102 {
103     Treap *nw;
104     if (!a) return nw = copy(b);
105     if (!b) return nw = copy(a);
106     if (randint() % (Size(a) + Size(b)) < Size(a))
107     {
108         nw = copy(a);
109         nw -> rs = Merge(nw -> rs, b);
110     }
111     else
112     {
113         nw = copy(b);
114         nw -> ls = Merge(a, nw -> ls);
115     }
116     nw -> update();
117     return nw;
118 }
119 Treap *Build(int l, int r)
120 {
121     if (l > r) return NULL;
122     R int mid = l + r >> 1;
123     Treap *nw = new Treap(str[mid]);
124     nw -> ls = Build(l, mid - 1);
125     nw -> rs = Build(mid + 1, r);
126     nw -> update();
127     return nw;
128 }
129 int now;
130 inline void Insert(R int k, R char ch)
131 {
132     Pair x = Split(root[now], k);
133     Treap *nw = new Treap(ch);
134     root[++now] = Merge(Merge(x.fir, nw), x.sec);
135 }
136 inline void Del(R int l, R int r)
137 {
138     Pair x = Split(root[now], l - 1);
139     Pair y = Split(x.sec, r - l + 1);
140     root[++now] = Merge(x.fir, y.sec);
141 }
142 inline void Copy(R int l, R int r, R int ll)
143 {
144     Pair x = Split(root[now], l - 1);
145     Pair y = Split(x.sec, r - l + 1);
146     Pair z = Split(root[now], ll);
147     Treap *ans = y.fir;
148     root[++now] = Merge(Merge(z.fir, ans), z.sec);
149 }
150 inline void Print(Treap *x, R int l, R int r)
151 {
152     if (!x) return ;
153     if (l > r) return;
154     R int mid = Size(x -> ls) + 1;
155     if (r < mid)
156     {
157         Print(x -> ls, l, r);
158         return ;
159     }

```

```

160     if (l > mid)
161     {
162         Print(x -> rs, l - mid, r - mid);
163         return ;
164     }
165     Print(x -> ls, l, mid - 1);
166     printf("%c", x -> data );
167     Print(x -> rs, 1, r - mid);
168 }
169 inline void Printtree(Treap *x)
170 {
171     if (!x) return;
172     Printtree(x -> ls);
173     printf("%c", x -> data );
174     Printtree(x -> rs);
175 }
176 int main()
177 {
178     // setfile();
179     R int n = FastIn();
180     gets(str + 1);
181     R int len = strlen(str + 1);
182     root[0] = Build(1, len);
183     while (1)
184     {
185         R char opt = getc();
186         while (opt < 'A' || opt > 'Z')
187         {
188             if (opt == EOF) return 0;
189             opt = getc();
190         }
191         if (opt == 'I')
192         {
193             R int x = FastIn();
194             R char ch = getc();
195             Insert(x, ch);
196         }
197         else if (opt == 'D')
198         {
199             R int l = FastIn(), r = FastIn();
200             Del(l, r);
201         }
202         else if (opt == 'C')
203         {
204             R int x = FastIn(), y = FastIn(), z = FastIn();
205             Copy(x, y, z);
206         }
207         else if (opt == 'P')
208         {
209             R int x = FastIn(), y = FastIn(), z = FastIn();
210             // printf("%d %d %d\n", x, y, z );
211             Print(root[now - x], y, z);
212             puts("");
213         }
214         // Printtree(root[now]);
215         // puts("");
216     }
217     return 0;
218 }

```

5.6 CDQ 分治 (ct)

```

1  //
2  // Title : cdq 分治
3  // Date : 18.04.2016
4  // Test : BZOJ-1176
5  // Complexity :  $O(n \log^2 n)$ 
6  //
7  /*
8      对于三维偏序等问题——
9      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！
10 */
11 #include <cstdio>
12 #include <cstring>
13 #include <algorithm>
14 #include <cmath>
15
16 #ifdef WIN32
17     #define LL "%I64d"
18 #else
19     #define LL "%lld"
20 #endif
21
22 #ifdef CT
23     #define debug(...) printf(__VA_ARGS__)
24     #define setfile()
25 #else
26     #define debug(...)
27     #define filename ""
28     #define setfile() freopen(filename".in", "r", stdin); freopen(filename".out", "w", stdout);
29 #endif
30
31 #define R register
32 #define getc() (S == T && (T = (S = B) + fread(B, 1, 1 << 15, stdin), S == T) ? EOF : *S++)
33 #define dmax(_a, _b) ((_a) > (_b) ? (_a) : (_b))
34 #define dmin(_a, _b) ((_a) < (_b) ? (_a) : (_b))
35 #define cmax(_a, _b) (_a < (_b) ? _a = (_b) : 0)
36 #define cmin(_a, _b) (_a > (_b) ? _a = (_b) : 0)
37 char B[1 << 15], *S = B, *T = B;
38 inline int FastIn()
39 {
40     R char ch; R int cnt = 0; R bool minus = 0;
41     while (ch = getc(), (ch < '0' || ch > '9') && ch != '-') ;
42     ch == '-' ? minus = 1 : cnt = ch - '0';
43     while (ch = getc(), ch >= '0' && ch <= '9') cnt = cnt * 10 + ch - '0';
44     return minus ? -cnt : cnt;
45 }
46
47 #define maxn 200010
48 #define maxm 2000010
49 struct event
50 {
51     int x, y, pos, opet, ans;
52     inline bool operator < (const event &that) const {return pos < that.pos ;}
53 }t[maxn], q[maxn];
54 #define lowbit(_x) ((_x) & -(_x))
55 int bit[maxm], last[maxm], s, w, cnt, now;
56 inline void add(R int x, R int val)
57 {
58     for (; x <= w; x += lowbit(x))
59     {

```

```

56         if (last[x] != now)
57             bit[x] = 0;
58         bit[x] += val;
59         last[x] = now;
60     }
61 }
62 inline int query(R int x)
63 {
64     R int ans = 0;
65     for (; x ; x -= lowbit(x))
66     {
67         if (last[x] == now)
68             ans += bit[x];
69     }
70     return ans;
71 }
72 void cdq(R int left, R int right)
73 {
74     if (left == right) return ;
75     R int mid = left + right >> 1;
76     cdq(left, mid); cdq(mid + 1, right);
77     //分成若干个子问题
78     ++now;
79     for (R int i = left, j = mid + 1; j <= right; ++j)
80     {
81         for (; i <= mid && q[i].x <= q[j].x; ++i)
82             if (!q[i].opet)
83                 add(q[i].y, q[i].ans);
84         //考虑前面的修改操作对后面的询问的影响
85         if (q[j].opet)
86             q[j].ans += query(q[j].y);
87     }
88     R int i, j, k = 0;
89     //以下相当于归并排序
90     for (i = left, j = mid + 1; i <= mid && j <= right; )
91     {
92         if (q[i].x <= q[j].x)
93             t[k++] = q[i++];
94         else
95             t[k++] = q[j++];
96     }
97     for (; i <= mid; )
98         t[k++] = q[i++];
99     for (; j <= right; )
100         t[k++] = q[j++];
101     for (R int i = 0; i < k; ++i)
102         q[left + i] = t[i];
103 }
104 int main()
105 {
106     // setfile();
107     s = FastIn();
108     w = FastIn();
109     while (1)
110     {
111         R int opt = FastIn();
112         if (opt == 1)
113         {
114             R int x = FastIn(), y = FastIn(), a = FastIn();
115             q[++cnt] = (event){x, y, cnt, 0, a};
116         }

```

```

117         if (opt == 2)
118         {
119             R int x = FastIn() - 1, y = FastIn() - 1, a = FastIn(), b = FastIn();
120             q[++cnt] = (event) {x, y, cnt, 1, x * y * s};
121             q[++cnt] = (event) {a, b, cnt, 2, a * b * s};
122             q[++cnt] = (event) {x, b, cnt, 2, x * b * s};
123             q[++cnt] = (event) {a, y, cnt, 2, a * y * s};
124         }
125         if (opt == 3) break;
126     }
127     cdq(1, cnt);
128     std::sort(q + 1, q + cnt + 1);
129     for (R int i = 1; i <= cnt; ++i)
130         if (q[i].opet == 1)
131             printf("%d\n", q[i].ans + q[i + 1].ans - q[i + 2].ans - q[i + 3].ans ), i +=
132                 ↪ 3;
133     return 0;

```

Chapter 6

Others

6.1 vimrc (gy)

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

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

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

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

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

6.2 Java Template (gy)

```
1 import java.io.BufferedReader;
2 import java.io.IOException;
3 import java.io.InputStreamReader;
4 import java.math.BigDecimal;
5 import java.math.BigInteger;
6 import java.math.RoundingMode;
7 import java.util.ArrayDeque;
8 import java.util.ArrayList;
9 import java.util.Arrays;
10 import java.util.Comparator;
11 import java.util.Deque;
12 import java.util.LinkedList;
13 import java.util.List;
14 import java.util.Scanner;
15 import java.util.StringTokenizer;
```

```

16 public class Template {
17     // Input
18     private static BufferedReader reader;
19     private static StringTokenizer tokenizer;

20     private static String next() {
21         try {
22             while (tokenizer == null || !tokenizer.hasMoreTokens())
23                 tokenizer = new StringTokenizer(reader.readLine());
24         } catch (IOException e) {
25             // do nothing
26         }
27         return tokenizer.nextToken();
28     }

29     private static int nextInt() {
30         return Integer.parseInt(next());
31     }

32     private static double nextDouble() {
33         return Double.parseDouble(next());
34     }

35     private static BigInteger nextBigInteger() {
36         return new BigInteger(next());
37     }

38     public static void main(String[] args) {
39         reader = new BufferedReader(new InputStreamReader(System.in));
40         Scanner scanner = new Scanner(System.in);
41         while (scanner.hasNext())
42             scanner.next();
43     }

44     // BigInteger & BigDecimal
45     private static void bigDecimal() {
46         BigDecimal a = BigDecimal.valueOf(1.0);
47         BigDecimal b = a.setScale(50, RoundingMode.HALF_EVEN);
48         BigDecimal c = b.abs();
49         // if scale omitted, b.scale is used
50         BigDecimal d = c.divide(b, 50, RoundingMode.HALF_EVEN);
51         // since Java 9
52         BigDecimal e = d.sqrt(new MathContext(50, RoundingMode.HALF_EVEN));
53         BigDecimal x = new BigDecimal(BigInteger.ZERO);
54         BigInteger y = BigDecimal.ZERO.toBigInteger(); // RoundingMode.DOWN
55         y = BigDecimal.ZERO.setScale(0, RoundingMode.HALF_EVEN).unscaledValue();
56     }

57     // sqrt for Java 8
58     private static BigDecimal sqrt(BigDecimal a, int scale, RoundingMode mode) {
59         if (a.equals(BigDecimal.ZERO))
60             return BigDecimal.ZERO;
61         a = a.setScale(scale, mode);
62         BigDecimal ans = a;
63         BigDecimal TWO = BigDecimal.valueOf(2L);
64         for (int i = 1; i <= scale; i++)
65             ans = ans.add(a.divide(ans, scale, mode)).divide(TWO, scale, mode);
66         return ans;
67     }

68     private static BigInteger sqrt(BigInteger a) {

```



```

69     BigInteger about = BigInteger.ZERO.setBit(a.bitLength() / 2);
70     return sqrt(new BigDecimal(a.toString()), new BigDecimal(about.toString()).setScale(0,
    ↪ RoundingMode.FLOOR).unscaledValue());
71 }

72 private static BigDecimal sqrt(BigDecimal a, BigDecimal initial) {
73     if (a.equals(BigDecimal.ZERO))
74         return BigDecimal.ZERO;
75     a = a.setScale(50, RoundingMode.HALF_EVEN);
76     BigDecimal ans = initial;
77     for (int i = 1; i <= 10; i++)
78         ans = ans.add(a.divide(ans, RoundingMode.HALF_EVEN)).divide(BigDecimal.valueOf(2),
    ↪ RoundingMode.HALF_EVEN);
79     return ans;
80 }

81 // ArrayList
82 private static void arrayList() {
83     List<Integer> list = new ArrayList<>();
84     // Generic array is banned
85     List[] lists = new List[100];
86     lists[0] = new ArrayList<Integer>();
87     // for List<Integer>, remove(Integer) stands for element, while remove(int) stands for
    ↪ index
88     list.remove(list.get(1));
89     list.remove(list.size() - 1);
90     list.clear();
91 }

92 // Queue
93 private static void queue() {
94     LinkedList<Integer> queue = new LinkedList<>();
95     // return the value without popping
96     queue.peek();
97     // pop and return the value
98     queue.poll();
99     Deque<Integer> deque = new ArrayDeque<>();
100     deque.peekFirst();
101     deque.peekLast();
102     deque.pollFirst();
103 }

104 // Others
105 private static void others() {
106     Arrays.sort(new int[10]);
107     Arrays.sort(new Integer[10], (a, b) -> {
108         if (a.equals(b)) return 0;
109         if (a > b) return -1;
110         return 1;
111     });
112     Arrays.sort(new Integer[10], Comparator.comparingInt((a) -> (int) a).reversed());
113     long a = 1_000_000_000_000_000L;
114     int b = Integer.MAX_VALUE;
115     int c = 'a';
116 }
117 }

```

6.3 Big Fraction (gy)

```

1 fun gcd(a: Long, b: Long): Long = if (b == 0L) a else gcd(b, a % b)
2 class Fraction(val a: BigInteger, val b: BigInteger) {
3     constructor(a: Long, b: Long) : this(BigInteger.valueOf(a / gcd(a, b)), BigInteger.valueOf(b /
4         ↪ gcd(a, b)))
5
6     operator fun plus(o: Fraction): Fraction {
7         var gcd = b.gcd(o.b)
8         val tempProduct = (b / gcd) * (o.b / gcd)
9         var ansA = a * (o.b / gcd) + o.a * (b / gcd)
10        val gcd2 = ansA.gcd(gcd)
11        ansA /= gcd2
12        gcd /= gcd2
13        return Fraction(ansA, gcd * tempProduct)
14    }
15
16    operator fun minus(o: Fraction): Fraction {
17        var gcd = b.gcd(o.b)
18        val tempProduct = (b / gcd) * (o.b / gcd)
19        var ansA = a * (o.b / gcd) - o.a * (b / gcd)
20        val gcd2 = ansA.gcd(gcd)
21        ansA /= gcd2
22        gcd /= gcd2
23        return Fraction(ansA, gcd * tempProduct)
24    }
25
26    operator fun times(o: Fraction): Fraction {
27        val gcd1 = a.gcd(o.b)
28        val gcd2 = b.gcd(o.a)
29        return Fraction((a / gcd1) * (o.a / gcd2), (b / gcd2) * (o.b / gcd1))
30    }
31 }

```

6.4 模拟退火 (ct)

```

1 #include <stdio>
2 #include <cmath>
3 #include <stdlib>
4 #include <time>
5
6 #define R register
7 #define cmax(_a, _b) (_a < (_b) ? _a = (_b) : 0)
8 #define maxn 10010
9 struct Poi {
10     double x, y, m;
11 }p[maxn];
12 double ans_x, ans_y, fans;
13 int n;
14 inline double rand01() {return rand() / 2147483647.0;}
15 inline double randp() {return (rand() & 1 ? 1 : -1) * rand01();}
16 inline double sqr(R double x) {return x * x;}
17 inline double f(R double x, R double y)
18 {
19     R double maxx = 0;
20     for (R int i = 1; i <= n; ++i)
21         maxx += sqrt(sqr(x - p[i].x) + sqr(y - p[i].y)) * p[i].m;
22 }

```

```

21     if (maxx < fans) {fans = maxx; ans_x = x; ans_y = y;}
22     return maxx;
23 }
24 int main()
25 {
26     srand(time(NULL) + clock());
27     scanf("%d", &n);
28     R double x = 0, y = 0, tot = 0;
29     for (R int i = 1; i <= n; ++i)
30         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;
31     fans = 1e30; x /= tot; y /= tot;
32     R double fnow = f(x, y);
33     for (R double T = 1e4; T > 1e-4; T *= 0.997)
34     {
35         R double nx = x + randp() * T, ny = y + randp() * T, fnext = f(nx, ny);
36         R double delta = fnext - fnow;
37         if (delta < 1e-9 || exp(-delta / T) > rand01())
38         {
39             x = nx; y = ny; fnow = fnext;
40         }
41     }
42     printf("%.3lf %.3lf\n", ans_x, ans_y);
43     return 0;
44 }

```

6.5 三分 (ct)

```

1  #define maxn 200010
2  #define inf 1e9
3  int a[maxn], n;
4  inline double check(R double x)
5  {
6      R double tmp, tmp1 = 0, tmp2 = 0, maxx = -inf, minn = -inf;
7      for (R int i = 1; i <= n; ++i)
8      {
9          tmp = (double) a[i] - x;
10
11          tmp1 += tmp;
12          cmax(maxx, tmp1);
13          tmp1 < 0 ? tmp1 = 0 : 0;
14
15          tmp2 -= tmp;
16          cmax(minn, tmp2);
17          tmp2 < 0 ? tmp2 = 0 : 0;
18      }
19      return dmax(maxx, minn);
20 }
21 int main()
22 {
23     n = F();
24     for (R int i = 1; i <= n; ++i) a[i] = F();
25     R double l = -1e4, r = 1e4;
26     for (R int i = 1; i <= 100; ++i)
27     {
28         R double ll = (l + r) * 0.5;
29         R double rr = (ll + r) * 0.5;
30         if (check(ll) < check(rr)) r = rr;
31         else l = ll;

```

```

30     }
31     printf("%.6lf\n", check((1 + r) * 0.5));
32     return 0;
33 }

```

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