

Platelet

Team Reference Material

凌皓煜 陈彤 顾逸

2018

Contents

1	Graph Theory	5
1.1	Dijkstra	6
1.2	2-SAT	6
1.3	双连通分量	6
1.3.1	点双连通分量	6
1.3.2	边双连通分量	6
1.4	K 短路	6
1.5	最大团	6
1.6	一般图最大匹配	6
1.7	树	6
1.7.1	虚树	6
1.7.2	矩阵树定理	6
1.7.3	点分治	6
1.7.4	Prufer 编码	6
1.7.5	Link-Cut Tree	6
1.7.6	树上倍增	6
1.7.7	数链剖分	6
1.8	仙人掌	6
1.9	带花树	6
1.10	KM 算法	6
1.11	支配树	6
1.11.1	DAG	6
1.11.2	一般图	6
1.12	弦图	6
1.13	网络流	6
1.13.1	最小割	6
1.13.2	最大流	6
1.13.3	费用流	6
1.13.4	有上下界的网络流	6
1.13.5	zkw 费用流	6
1.14	差分约束	6
2	Math	7
2.1	int64 相乘取模	7
2.2	扩展欧几里得 (gy)	7
2.3	中国剩余定理	8
2.4	组合数	8
2.4.1	Lucas 定理	8
2.4.2	组合数合数取模	8
2.5	高斯消元	8
2.6	Miller Rabin & Pollard Rho (gy)	8
2.7	$O(m^2 \log n)$ 线性递推	10
2.8	Polynomial	10
2.8.1	FFT	10

2.8.2	NTT & 多项式求逆	10
2.9	拉格朗日插值	10
2.10	杜教筛	10
2.11	BSGS	10
2.11.1	BSGS	10
2.11.2	扩展 BSGS	10
2.12	直线下整点个数	10
2.13	单纯形	10
2.14	辛普森积分	10
2.15	常用数列定理	10
2.16	积分表	10
3	Geometry	11
3.1	点、直线、圆	11
3.2	点到凸包切线	11
3.3	直线凸包交点	11
3.4	凸包游戏	11
3.5	半平面交	11
3.6	旋转卡壳	11
3.7	判断圆是否有交	11
3.8	最小圆覆盖	11
3.9	最小球覆盖	11
3.10	$O(n^2 \log n)$ 圆交面积和重心	11
3.11	圆与多边形交	11
3.12	$O(n \log n)$ 凸多边形内的最大圆	11
3.13	三角形的五心	11
3.14	三维凸包	11
3.15	三维绕轴旋转	11
3.16	几何公式	11
4	String	12
4.1	KMP	12
4.2	AC 自动机	12
4.3	后缀数组	12
4.4	后缀自动机	12
4.5	Manacher	12
4.6	回文自动机	12
4.7	最小表示法	12
5	Data Structure	13
5.1	莫队 (ct)	13
5.2	ST 表 (ct)	14
5.3	可并堆 (ct)	15
5.4	线段树 (ct)	16
5.4.1	ZKW 线段树	16
5.4.2	主席树	17
5.5	平衡树 (ct)	18
5.5.1	Splay	18
5.5.2	非旋转 Treap	21
5.5.3	可持久化平衡树	24
5.6	CDQ 分治 (ct)	28

6	Others	31
6.1	vimrc (gy)	31
6.2	Java Template (gy)	31
6.3	Big Fraction (gy)	34
6.4	模拟退火 (ct)	34
6.5	三分 (ct)	35
6.6	博弈论模型 (gy)	36

Chapter 1

Graph Theory

1.1 Dijkstra

1.2 2-SAT

1.3 双连通分量

1.3.1 点双连通分量

1.3.2 边双连通分量

1.4 K 短路

1.5 最大团

1.6 一般图最大匹配

1.7 树

1.7.1 虚树

1.7.2 矩阵树定理

1.7.3 点分治

1.7.4 Prufer 编码

1.7.5 Link-Cut Tree

1.7.6 树上倍增

1.7.7 数链剖分

1.8 仙人掌

1.9 带花树

1.10 KM 算法

1.11 支配树

1.11.1 DAG

1.11.2 一般图

1.12 弦图

1.13 网络流

Chapter 2

Math

2.1 int64 相乘取模

2.2 扩展欧几里得 (gy)

```
1 // return gcd(a, b)
2 // ax+by=gcd(a,b)
3 template <typename T>
4 T extend_gcd(T a, T b, T &x, T &y) {
5     if (b == 0) {
6         x = 1, y = 0;
7         return a;
8     }
9     T res = extend_gcd(b, a % b, x, y);
10    T t = y;
11    y = x - a / b * y;
12    x = t;
13    return res;
14 }
15
16 // return minimal positive integer x so that ax+by=c
17 // or -1 if such x does not exist
18 template <typename T>
19 inline T solve_equ(T a, T b, T c) {
20     T x, y, d;
21     d = extend_gcd(a, b, x, y);
22     if (c % d)
23         return -1;
24     T t = c / d;
25     x *= t;
26     y *= t;
27     T k = b / d;
28     x = (x % k + k) % k;
29     return x;
30 }
31 // return minimal positive integer x so that ax==b(mod p)
32 // or -1 if such x does not exist
33 template <typename T>
34 inline T solve(T a, T b, T p) {
35     a = (a % p + p) % p;
36     b = (b % p + p) % p;
37     return solve_equ(a, p, b);
38 }
```

2.3 中国剩余定理

2.4 组合数

2.4.1 Lucas 定理

2.4.2 组合数合数取模

2.5 高斯消元

2.6 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 #include <cstdint>
19 #include <cstdlib>
20 #include <vector>
21
22 const int test_case_size = 12;
23 const int test_cases[test_case_size] = {2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37};
24
25 int64_t multiply_mod(int64_t x, int64_t y, int64_t p) {
26     int64_t t = (x * y - (int64_t) ((long double) x / p * y + 1e-3) * p) % p;
27     return t < 0 ? t + p : t;
28 }
29
30 int64_t add_mod(int64_t x, int64_t y, int64_t p) {
31     return (0ull + x + y) % p;
32 }
33
34 int64_t power_mod(int64_t x, int64_t exp, int64_t p) {
35     int64_t ans = 1;
36     while (exp) {
37         if (exp & 1)
38             ans = multiply_mod(ans, x, p);
39         x = multiply_mod(x, x, p);
40         exp >>= 1;
41     }
42     return ans;
43 }
44
45 bool miller_rabin_check(int64_t prime, int64_t base) {
46     int64_t number = prime - 1;

```



```

47     for (; ~number & 1; number >>= 1)
48         continue;
49     int64_t result = power_mod(base, number, prime);
50     for (; number != prime - 1 && result != 1 && result != prime - 1; number <<= 1)
51         result = multiply_mod(result, result, prime);
52     return result == prime - 1 || (number & 1) == 1;
53 }
54
55 bool miller_rabin(int64_t number) {
56     if (number < 2)
57         return false;
58     if (number < 4)
59         return true;
60     if (~number & 1)
61         return false;
62     for (int i = 0; i < test_case_size && test_cases[i] < number; i++)
63         if (!miller_rabin_check(number, test_cases[i]))
64             return false;
65     return true;
66 }
67
68 int64_t gcd(int64_t x, int64_t y) {
69     return y == 0 ? x : gcd(y, x % y);
70 }
71
72 int64_t pollard_rho_test(int64_t number, int64_t seed) {
73     int64_t x = rand() % (number - 1) + 1, y = x;
74     int head = 1, tail = 2;
75     while (true) {
76         x = multiply_mod(x, x, number);
77         x = add_mod(x, seed, number);
78         if (x == y)
79             return number;
80         int64_t answer = gcd(std::abs(x - y), number);
81         if (answer > 1 && answer < number)
82             return answer;
83         if (++head == tail) {
84             y = x;
85             tail <<= 1;
86         }
87     }
88 }
89
90 void factorize(int64_t number, std::vector<int64_t> &divisor) {
91     if (number > 1) {
92         if (miller_rabin(number)) {
93             divisor.push_back(number);
94         } else {
95             int64_t factor = number;
96             while (factor >= number)
97                 factor = pollard_rho_test(number, rand() % (number - 1) + 1);
98             factorize(number / factor, divisor);
99             factorize(factor, divisor);
100         }
101     }
102 }
103
104 #include <cstdio>
105 int main() {
106     static int64_t tar;
107     static std::vector<int64_t> factors;

```

```

108     while (scanf("%lld", &tar)) {
109         printf("%d\n", miller_rabin(tar));
110         factors.clear();
111         factorize(tar, factors);
112         for (int64_t i : factors)
113             printf("%lld\n", i);
114     }
115     return 0;
116 }

```

2.7 $O(m^2 \log n)$ 线性递推

2.8 Polynomial

2.8.1 FFT

2.8.2 NTT & 多项式求逆

2.9 拉格朗日插值

2.10 杜教筛

2.11 BSGS

2.11.1 BSGS

2.11.2 扩展 BSGS

2.12 直线下整点个数

2.13 单纯形

2.14 辛普森积分

2.15 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

2.16 积分表

Chapter 3

Geometry

3.1 点、直线、圆

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
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 #else
25     #define debug(...)
26 #endif
27
28 #define R register
29 #define getc() (S==T&&(T=(S=B)+fread(B,1,1<<15,stdin),S==T)?EOF:*S++)
30 #define gmax(_a, _b) ((_a) > (_b) ? (_a) : (_b))
31 #define gmin(_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 #define maxn 50010
44 int col[maxn], num[maxn], size, pos[maxn];
45 long long up[maxn], dw[maxn], ans;
46 struct Query{
47     int l, r, id;
48 }q[maxn];
49 inline bool cmp(const Query &i, const Query &j){
50     return pos[i.l] != pos[j.l] ? (i.l < j.l) : (pos[i.l] & 1 ? i.r < j.r : i.r > j.r);
51 }
52 inline long long gcd(R long long a, R long long b){
53     R long long tmp;
54     while (b){
55         tmp = b;
56         b = a % b;
57         a = tmp;
58     }
59     return a;
60 }
61 inline void update(R int x, R int d){
62     ans -= num[col[x]] * num[col[x]];
63     num[col[x]] += d;
64     ans += num[col[x]] * num[col[x]];
65 }
66 int main()
67 {
68     R int n = FastIn(), m = FastIn(); size = (int)sqrt(n * 1.0);
69     for (R int i = 1; i <= n; i++) col[i] = FastIn(), pos[i] = (i - 1) / size + 1;
70     for (R int i = 1; i <= m; i++){
71         q[i].l = FastIn(); q[i].r = FastIn(); q[i].id = i;
72     }
73     std::sort(q + 1, q + m + 1, cmp);
74     R int l = 1, r = 0;
75     for (R int i = 1; i <= m; i++){
76         R int id_now = q[i].id;
77         if (q[i].l == q[i].r){
78             up[id_now] = 0; dw[id_now] = 1; continue;
79         }
80         for (; r < q[i].r; r++) update(r + 1, 1);
81         for (; r > q[i].r; r--) update(r, -1);
82         for (; l < q[i].l; l++) update(l, -1);
83         for (; l > q[i].l; l--) update(l - 1, 1);
84         R long long aa, bb, cc;
85         aa = ans - q[i].r * q[i].l - 1;
86         bb = (long long)(q[i].r - q[i].l + 1) * (q[i].r - q[i].l);
87         cc = gcd(aa, bb); aa /= cc; bb /= cc;
88         up[id_now] = aa; dw[id_now] = bb;
89     }
90     for (R int i = 1; i <= m; i++) printf("%lld/%lld\n", up[i], dw[i]);
91     return 0;
92 }

```

5.2 ST 表 (ct)

```

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

```

```

7  int Log[maxn];
8
9  void build()
10 {
11     for (int i = 1; i <= n; ++i) f[0][i] = a[i];
12
13     int lim = Log[n];
14     for (int j = 1; j <= lim; ++j)
15     {
16         int *fj = f[j], *fj1 = f[j - 1];
17         for (int i = 1; i <= n - (1 << j) + 1; ++i)
18             fj[i] = dmax(fj1[i], fj1[i + (1 << (j - 1))]);
19     }
20 }
21 int Query(int l, int r)
22 {
23     int k = Log[r - l + 1];
24     return dmax(f[k][l], f[k][r - (1 << k) + 1]);
25 }
26 int main()
27 {
28     scanf("%d", &n);
29     Log[0] = -1;
30     for (int i = 1; i <= n; ++i)
31     {
32         scanf("%d", &a[i]);
33         Log[i] = Log[i >> 1] + 1;
34     }
35     build();
36     int q;
37     scanf("%d", &q);
38     for (; q; --q)
39     {
40         int l, r; scanf("%d%d", &l, &r);
41         printf("%d\n", Query(l, r));
42     }
43 }

```

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 //
2 // Title:ZKW Segment Tree
3 // Date:19.11.2015
4 // Complexity:
5 // Build Tree: $O(N)$ 
6 // Query: $O(\log N)$ 
7 // Change: $O(\log N)$ 
8
9 #include<cstdio>
10 #include<cmath>
11 #define maxn 100000
12 #define R register
13 int T[1<<18|1],n,m,M;
14
15 inline int FastIn()
16 {
17     R char ch=getchar();R int cnt=0;R bool minus=0;
18     while ((ch<'0' || ch>'9') && ch!='-') ch=getchar();
19     if (ch=='-') minus=1,ch=getchar();
20     while (ch>='0' && ch<='9') cnt=cnt*10+ch-'0',ch=getchar();
21     return minus?-cnt:cnt;
22 }
23
24 inline void Build_Tree()
25 {
26     for (R int i=M-1;i>=1;i--)
27         T[i]=T[2*i]+T[2*i+1];
28 }
29
30 inline int Query(int s,int t)
31 {
32     R int Ans;
33     for (Ans=0,s=s+M-1,t=t+M+1;s^t^1;s>>=1,t>>=1)
34     {
35         if (~s&1) Ans+=T[s^1];
36         if (t&1) Ans+=T[t^1];
37     }
38     return Ans;
39 }
40
41 inline void Change(int x,int NewValue)
42 {
43     R int i=M+x;
44     for (T[i]=NewValue,i>>=1;i;i>>=1)
45         T[i]=T[2*i]+T[2*i+1];
46 }
47
48 int main()
49 {
50     n=FastIn();m=FastIn();
51     for (M=1;M<=n;M<<=1);
52     for (R int i=0;i<n;i++)
53         T[M+i]=FastIn();
54     Build_Tree();
55     for (R int i=1;i<=m;i++)
56     {

```



```

57         R char cmd=getchar();
58         if (cmd=='Q')
59         {
60             R int a=FastIn()-1,b=FastIn()-1;
61             printf("%d\n",Query(a,b));
62         }
63         if (cmd=='M')
64         {
65             R int a=FastIn()-1,b=FastIn();
66             Change(a,b);
67         }
68     }
69     return 0;
70 }

```

5.4.2 主席树

```

1 //
2 // Title: Functional Segment Tree
3 // Date:16.12.2015
4 // Complexity:O((n+m)logn)
5 // Test:YZOJ-1991
6 #include<cstdio>
7 #include<algorithm>
8 #define maxt 2000010
9 #define maxn 100010
10 #define R register
11 inline int FastIn(){
12     R char ch=getchar();R int cnt=0;
13     while (ch<'0' || ch>'9') ch=getchar();
14     while (ch>='0' && ch<='9') cnt=cnt*10+ch-'0',ch=getchar();
15     return cnt;
16 }
17
18 int ls[maxt],
19     rs[maxt],
20     count[maxt],
21     root[maxn],
22     tot;
23
24 int num[maxn],rank[maxn],n,m,r[maxn];
25
26 bool cmp(const int &i,const int &j){
27     return num[i]<num[j];
28 }
29
30 inline void Insert(int last,int left,int right,int pre)
31 {
32     count[++tot]=count[last]+1;
33     if (left==right) return;
34     R int mid=(left+right)>>1;
35     if (pre>mid){
36         rs[tot]=tot+1;
37         Insert(rs[last],mid+1,right,pre);
38     }
39     else{
40         ls[tot]=tot+1;
41         rs[tot]=rs[last];
42         Insert(ls[last],left,mid,pre);
43     }
44 }

```

```

44 }
45
46 inline int Query(int a,int b,int k)
47 {
48     R int l=1,r=n,mid,f1=a,f2=b,cnt,kk=k;
49     while (l<r){
50         mid=(l+r)>>1;cnt=count[ls[f2]]-count[ls[f1]];
51         if (cnt>=kk) f1=ls[f1],f2=ls[f2],r=mid;
52         else f1=rs[f1],f2=rs[f2],l=mid+1,kk-=cnt;
53     }
54     return l;
55 }
56
57 int main()
58 {
59     n=FastIn();m=FastIn();R int i,a,b,k;
60     for (i=1;i<=n;i++) num[i]=FastIn(),rank[i]=i;
61     std::sort(rank+1,rank+n+1,cmp);
62     std::sort(num+1,num+n+1);
63     for (i=1;i<=n;i++) r[rank[i]]=i;
64     for (i=1;i<=n;i++) {
65         root[i]=tot+1;
66         Insert(root[i-1],1,n,r[i]);
67     }
68     for (i=1;i<=m;i++){
69         a=FastIn();b=FastIn();k=FastIn();
70         printf("%d\n",num[Query(root[a-1],root[b],k)]);
71     }
72     return 0;
73 }

```

5.5 平衡树 (ct)

5.5.1 Splay

```

1  //
2  // Title : Splay Tree
3  // Date : 11.01.2016
4  // Complexity : O(nlogn) (期望)
5  // Test : BZOJ-1251
6  /*
7  */
8  #include <stdio>
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 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 n, Q, root;
42 int fa[maxn], ch[maxn][2], id[maxn], size[maxn];
43 int tag[maxn], mx[maxn], num[maxn];
44 bool rev[maxn];
45 inline void update(int x){
46     R int ls=ch[x][0], rs=ch[x][1];
47     mx[x]=num[x];
48     cmax(mx[x], mx[ls]); cmax(mx[x], mx[rs]);
49     size[x]=size[ls]+size[rs]+1;
50 }//更新
51 void build(int l, int r, int rt){
52     if (l>r) return ;
53     R int mid=l+r>>1;
54     fa[mid]=rt;
55     if (mid<rt) ch[rt][0]=mid;
56     else ch[rt][1]=mid;
57     build(l, mid-1, mid);
58     build(mid+1, r, mid);
59     update(mid);
60 }//建树
61 inline void pushdown(int x){
62     R int ls=ch[x][0], rs=ch[x][1];
63     if (tag[x]){
64         R int lazy=tag[x];
65         if (ls) tag[ls]+=lazy, num[ls]+=lazy, mx[ls]+=lazy;
66         if (rs) tag[rs]+=lazy, num[rs]+=lazy, mx[rs]+=lazy;
67         tag[x]=0;
68     }
69     if (rev[x]){
70         if (ls) rev[ls]^=1;
71         if (rs) rev[rs]^=1;
72         ch[x][1]=ls; ch[x][0]=rs;
73         rev[x]=0;
74     }
75 }//具体下传的过程
76 inline void rotate(int x){//把 x 向上旋转到 x 的父亲
77     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
78     if (f==root) root=x, ch[0][0]=x;
79     (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
80     (fa[x]=gf)>0 ? ch[gf][ch[gf][1]==f]=x : 0; //把 x 代替 f 的位置
81     fa[ch[x][d^1]]=x; //把 f 接到 x 的下面
82     update(f); //更新 f 节点
83 }
84 inline void splay(int x, int rt){//把 x 旋转到 rt
85     while (fa[x]!=rt){

```

```

86         R int f=fa[x],gf=fa[f];
87         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
88         rotate(x);
89     }
90     update(x);
91 }
92 int find(int x,int rank){
93     if (tag[x]||rev[x]) pushdown(x);
94     R int ls=ch[x][0],rs=ch[x][1],lsize=size[ls];
95     if (lsize+1==rank) return x;
96     if (lsize>=rank) return find(ls,rank);
97     else return find(rs,rank-lsize-1);
98 } //找第 k 小
99 inline int prepare(int l,int r){
100     R int x=find(root,l-1);
101     splay(x,0);
102     x=find(root,r+1);
103     splay(x,root);
104     return ch[x][0];
105 } //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
106 inline void add(int l,int r,int w){
107     R int x=prepare(l,r);
108     tag[x]+=w,num[x]+=w,mx[x]+=w;
109 } //区间加
110 inline void rever(int l,int r){
111     R int x=prepare(l,r);
112     rev[x]^=1;
113 } //区间翻转
114 inline void query(int l,int r){
115     R int x=prepare(l,r);
116     printf("%d\n",mx[x] );
117 } //区间查询最大值
118 inline int split(R int k){
119     R int ls;
120     if (k<size[root])
121     {
122         R int kth=find(root,k+1);
123         splay(kth);ls=ch[kth][0];
124         fa[ls]=0;ch[kth][0]=0;
125         size[kth]-=size[ls];
126     }
127     else{
128         ls=root;root=0;
129     }
130     return ls;
131 } //删除数列
132 inline void merge(R int nwrt){
133     if (!root) {root=nwrt;return;}
134     R int nw=find(root,1);
135     splay(nw);fa[nwrt]=nw;ch[nw][0]=nwrt;
136     size[nw]+=size[nwrt];
137 } //合并数列
138 int main()
139 {
140     n=FastIn()+2;Q=FastIn();R int i,l,r,v,cmd;mx[0]=-233333333;
141     build(1,n,0);root=(1+n)>>1;
142     for (;Q--){
143         cmd=FastIn();l=FastIn()+1;r=FastIn()+1;
144         if (cmd==1) v=FastIn(),add(l,r,v);
145         else if (cmd==2) rever(l,r);

```

```

146         else query(l,r);
147     }
148     return 0;
149 }

```

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()
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 const int Ta = 1 << 16 | 3, Tb = 33333331;
47 int Tc;
48 inline int randint() {return Tc = Ta * Tc + Tb;}
49 struct Treap
50 {
51     int data, key, size;
52     Treap *ls, *rs;
53     Treap(int _val):data(_val), key(randint()), ls(NULL), rs(NULL), size(1){}

```

```

54     inline void update()
55     {
56         size = (ls ? ls -> size : 0) + (rs ? rs -> size : 0) + 1;
57     }
58 }*root;
59 inline int Size(Treap *x)
60 {
61     return x ? x -> size : 0;
62 }
63 //为了防止访问到空节点, 定义一个函数来访问 size
64 struct Pair
65 {
66     Treap *fir, *sec;
67 };
68 Treap *Merge(Treap *a, Treap *b)
69 {
70     if (!a) return b;
71     if (!b) return a;
72     if (a -> key < b -> key)
73     {
74         a -> rs = Merge(a -> rs, b);
75         a -> update();
76         return a;
77     }
78     else
79     {
80         b -> ls = Merge(a, b -> ls);
81         b -> update();
82         return b;
83     }
84 }
85 //按照 a, b 的顺序来合并两棵 Treap
86 Pair Split(Treap *x, int k)
87 {
88     if (!x) return (Pair){NULL, NULL};
89     Pair y; y.fir = NULL; y.sec = NULL;
90     if (Size(x -> ls) >= k)
91     {
92         y = Split(x -> ls, k);
93         x -> ls = y.sec;
94         x -> update();
95         y.sec = x;
96     }
97     else
98     {
99         y = Split(x -> rs, k - Size(x -> ls) - 1);
100         x -> rs = y.fir;
101         x -> update();
102         y.fir = x;
103     }
104     return y;
105 }
106 //将前 k 个的点分离出来
107 inline int Find(R int k)
108 {
109     Pair x = Split(root, k - 1);
110     Pair y = Split(x.sec, 1);
111     Treap *ans = y.fir;
112     root = Merge(Merge(x.fir, ans), y.sec);
113     return ans -> data;
114 }

```

```

115 //找到第 k 小的 data 值
116 int Get(Treap *x, R int val)
117 {
118     if (!x) return 0;
119     return val < x->data ? Get(x->ls, val) : Get(x->rs, val) + Size(x->ls) + 1;
120 }
121 //找到 val 的排名
122 inline void Insert(R int val)
123 {
124     R int k = Get(root, val);
125     Pair x = Split(root, k);
126     Treap *pre = new Treap(val);
127     root = Merge(Merge(x.fir, pre), x.sec);
128 }
129 //插入
130 inline void Delete(R int val)
131 {
132     R int k = Get(root, val);
133     Pair x = Split(root, k - 1);
134     Pair y = Split(x.sec, 1);
135     root = Merge(x.fir, y.sec);
136 }
137 //单点删除
138 inline int upper(R int val)
139 {
140     R int ans = 1e9;
141     Treap *tmp = root;
142     while (tmp)
143     {
144         if (tmp->data > val)
145         {
146             cmin(ans, tmp->data);
147             tmp = tmp->ls;
148         }
149         else
150             tmp = tmp->rs;
151     }
152     return ans;
153 }
154 inline int lower(R int val)
155 {
156     R int ans = -1e9;
157     Treap *tmp = root;
158     while (tmp)
159     {
160         if (tmp->data < val)
161         {
162             cmax(ans, tmp->data);
163             tmp = tmp->rs;
164         }
165         else tmp = tmp->ls;
166     }
167     return ans;
168 }
169 void print(Treap *x)
170 {
171     if (!x) return;
172     print(x->ls);
173     printf("%d ", x->data);
174     print(x->rs);
175 }

```

```

176 int main()
177 {
178     root = NULL;
179     for (R int Q = FastIn(); Q; --Q)
180     {
181         R int opt = FastIn(), x = FastIn();
182         if (opt == 1) Insert(x);
183         else if (opt == 2) Delete(x);
184         else if (opt == 3)
185         {
186             R int ans = Get(root, x);
187             while (ans > 1 && Find(ans - 1) == x) ans--;
188             printf("%d\n", ans );
189         }
190         else if (opt == 4) printf("%d\n", Find(x) );
191         else if (opt == 5) printf("%d\n", lower(x) );
192         else printf("%d\n", upper(x) );
193     }
194     return 0;
195 }
196 /*
197 input:
198 10
199 1 106465
200 4 1
201 1 317721
202 1 460929
203 1 644985
204 1 84185
205 1 89851
206 6 81968
207 1 492737
208 5 493598
209
210 output:
211 106465
212 84185
213 492737
214
215 input2:
216 5
217 1 1
218 1 1
219 1 1
220 1 2
221 3 1
222 output2:
223 1
224 */

```

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:

```


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*

```

9
10
11
12 */
13 #include <cstdio>
14 #include <cstring>
15 #include <algorithm>
16 #include <cmath>
17
18 #ifdef WIN32
19     #define LL "%I64d"
20 #else
21     #define LL "%lld"
22 #endif
23
24 #ifdef CT
25     #define debug(...) printf(__VA_ARGS__)
26     #define setfile()
27 #else
28     #define debug(...)
29     #define filename ""
30     #define setfile() freopen(filename".in", 'r', stdin); freopen(filename".out", 'w', stdout)
31 #endif
32
33 #define R register
34 // #define getc() (S==EOF?(T=(S=B)+fread(B,1,1<<15,stdin),S==T)?EOF:*S++)
35 #define getc() getchar()
36 #define dmax(_a, _b) ((_a) > (_b) ? (_a) : (_b))
37 #define dmin(_a, _b) ((_a) < (_b) ? (_a) : (_b))
38 #define cmax(_a, _b) (_a < (_b) ? _a = (_b) : 0)
39 #define cmin(_a, _b) (_a > (_b) ? _a = (_b) : 0)
40 #define cabs(_x) ((_x)<0?(-_x):(_x))
41 char B[1<<15],*S=B,*T=B;
42 inline int FastIn()
43 {
44     R char ch;R int cnt=0;R bool minus=0;
45     while (ch=getc(),(ch < '0' || ch > '9') && ch != '-') ;
46     ch == '-' ? minus=1:cnt=ch-'0';
47     while (ch=getc(),ch >= '0' && ch <= '9') cnt = cnt * 10 + ch - '0';
48     return minus?-cnt:cnt;
49 }
50 #define maxn 100010
51 char str[maxn];
52 struct Treap
53 {
54     char data;
55     int size;
56     Treap *ls, *rs;
57     Treap(char _ch): data(_ch), size(1), ls(NULL), rs(NULL){}
58     inline void update()
59     {
60         size = (ls ? ls -> size : 0) + (rs ? rs -> size : 0) + 1;
61     }
62 }*root[maxn];
63 inline int Size(Treap *x)
64 {
65     return x ? x -> size : 0;
66 }
67 struct Pair
68 {
69     Treap *fir, *sec;

```

```

70 };
71 inline Treap *copy(Treap *x)
72 {
73     if (!x) return NULL;
74     Treap *nw = new Treap(x -> data);
75     nw -> ls = x -> ls;
76     nw -> rs = x -> rs;
77     nw -> size = x -> size;
78     return nw;
79 }
80 Pair Split(Treap *x, int k)
81 {
82     if (!x) return (Pair){NULL, NULL};
83     Pair y; y.fir = NULL; y.sec = NULL;
84     Treap *nw = copy(x);
85     if (Size(nw -> ls) >= k)
86     {
87         y = Split(nw -> ls, k);
88         nw -> ls = y.sec;
89         nw -> update();
90         y.sec = nw;
91     }
92     else
93     {
94         y = Split(nw -> rs, k - Size(nw -> ls) - 1);
95         nw -> rs = y.fir;
96         nw -> update();
97         y.fir = nw;
98     }
99     return y;
100 }
101 const int Ta = 1 << 16 | 3, Tb = 33333331;
102 unsigned int Tc;
103 inline unsigned int randint(){return Tc = Ta * Tc + Tb;}
104 Treap *Merge(Treap *a, Treap *b)
105 {
106     Treap *nw;
107     if (!a) return nw = copy(b);
108     if (!b) return nw = copy(a);
109     if (randint() % (Size(a) + Size(b)) < Size(a))
110     {
111         nw = copy(a);
112         nw -> rs = Merge(nw -> rs, b);
113     }
114     else
115     {
116         nw = copy(b);
117         nw -> ls = Merge(a, nw -> ls);
118     }
119     nw -> update();
120     return nw;
121 }
122 Treap *Build(int l, int r)
123 {
124     if (l > r) return NULL;
125     R int mid = l + r >> 1;
126     Treap *nw = new Treap(str[mid]);
127     nw -> ls = Build(l, mid - 1);
128     nw -> rs = Build(mid + 1, r);
129     nw -> update();
130     return nw;

```

```

131 }
132 int now;
133 inline void Insert(R int k, R char ch)
134 {
135     Pair x = Split(root[now], k);
136     Treap *nw = new Treap(ch);
137     root[++now] = Merge(Merge(x.fir, nw), x.sec);
138 }
139 inline void Del(R int l, R int r)
140 {
141     Pair x = Split(root[now], l - 1);
142     Pair y = Split(x.sec, r - l + 1);
143     root[++now] = Merge(x.fir, y.sec);
144 }
145 inline void Copy(R int l, R int r, R int ll)
146 {
147     Pair x = Split(root[now], l - 1);
148     Pair y = Split(x.sec, r - l + 1);
149     Pair z = Split(root[now], ll);
150     Treap *ans = y.fir;
151     root[++now] = Merge(Merge(z.fir, ans), z.sec);
152 }
153 inline void Print(Treap *x, R int l, R int r)
154 {
155     if (!x) return ;
156     if (l > r) return;
157     R int mid = Size(x -> ls) + 1;
158     if (r < mid)
159     {
160         Print(x -> ls, l, r);
161         return ;
162     }
163     if (l > mid)
164     {
165         Print(x -> rs, l - mid, r - mid);
166         return ;
167     }
168     Print(x -> ls, l, mid - 1);
169     printf("%c", x -> data );
170     Print(x -> rs, 1, r - mid);
171 }
172 inline void Printtree(Treap *x)
173 {
174     if (!x) return;
175     Printtree(x -> ls);
176     printf("%c", x -> data );
177     Printtree(x -> rs);
178 }
179 int main()
180 {
181     // setfile();
182     R int n = FastIn();
183     gets(str + 1);
184     R int len = strlen(str + 1);
185     root[0] = Build(1, len);
186     while (1)
187     {
188         R char opt = getc();
189         while (opt < 'A' || opt > 'Z')
190         {
191             if (opt == EOF) return 0;

```

```

192         opt = getc();
193     }
194     if (opt == 'I')
195     {
196         R int x = FastIn();
197         R char ch = getc();
198         Insert(x, ch);
199     }
200     else if (opt == 'D')
201     {
202         R int l = FastIn(), r = FastIn();
203         Del(l, r);
204     }
205     else if (opt == 'C')
206     {
207         R int x = FastIn(), y = FastIn(), z = FastIn();
208         Copy(x, y, z);
209     }
210     else if (opt == 'P')
211     {
212         R int x = FastIn(), y = FastIn(), z = FastIn();
213         // printf("%d %d %d\n", x, y, z );
214         Print(root[now - x], y, z);
215         puts("");
216     }
217     // Printtree(root[now]);
218     // puts("");
219 }
220 return 0;
221 }

```

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 #define maxn 200010
47 #define maxm 2000010
48 struct event
49 {
50     int x, y, pos, opet, ans;
51     inline bool operator < (const event &that) const {return pos < that.pos ;}
52 }t[maxn], q[maxn];
53 #define lowbit(_x) ((_x) & -(_x))
54 int bit[maxm], last[maxm], s, w, cnt, now;
55 inline void add(R int x, R int val)
56 {
57     for (; x <= w; x += lowbit(x))
58     {
59         if (last[x] != now)
60             bit[x] = 0;
61         bit[x] += val;
62         last[x] = now;
63     }
64 }
65 inline int query(R int x)
66 {
67     R int ans = 0;
68     for (; x ; x -= lowbit(x))
69     {
70         if (last[x] == now)
71             ans += bit[x];
72     }
73     return ans;
74 }
75 void cdq(R int left, R int right)
76 {
77     if (left == right) return ;
78     R int mid = left + right >> 1;
79     cdq(left, mid); cdq(mid + 1, right);
80     //分成若干个子问题
81     ++now;
82     for (R int i = left, j = mid + 1; j <= right; ++j)
83     {
84         for (; i <= mid && q[i].x <= q[j].x; ++i)
85             if (!q[i].opet)
86                 add(q[i].y, q[i].ans);
87         //考虑前面的修改操作对后面的询问的影响

```

```

88         if (q[j].opet)
89             q[j].ans += query(q[j].y);
90     }
91     R int i, j, k = 0;
92     //以下相当于归并排序
93     for (i = left, j = mid + 1; i <= mid && j <= right; )
94     {
95         if (q[i].x <= q[j].x)
96             t[k++] = q[i++];
97         else
98             t[k++] = q[j++];
99     }
100     for (; i <= mid; )
101         t[k++] = q[i++];
102     for (; j <= right; )
103         t[k++] = q[j++];
104     for (R int i = 0; i < k; ++i)
105         q[left + i] = t[i];
106 }
107 int main()
108 {
109     // setfile();
110     s = FastIn();
111     w = FastIn();
112     while (1)
113     {
114         R int opt = FastIn();
115         if (opt == 1)
116         {
117             R int x = FastIn(), y = FastIn(), a = FastIn();
118             q[++cnt] = (event){x, y, cnt, 0, a};
119         }
120         if (opt == 2)
121         {
122             R int x = FastIn() - 1, y = FastIn() - 1, a = FastIn(), b = FastIn();
123             q[++cnt] = (event){x, y, cnt, 1, x * y * s};
124             q[++cnt] = (event){a, b, cnt, 2, a * b * s};
125             q[++cnt] = (event){x, b, cnt, 2, x * b * s};
126             q[++cnt] = (event){a, y, cnt, 2, a * y * s};
127         }
128         if (opt == 3) break;
129     }
130     cdq(1, cnt);
131     std::sort(q + 1, q + cnt + 1);
132     for (R int i = 1; i <= cnt; ++i)
133         if (q[i].opet == 1)
134             printf("%d\n", q[i].ans + q[i + 1].ans - q[i + 2].ans - q[i + 3].ans ), i +=
135                 ↵ 3;
136     return 0;
137 }

```

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
7 nm <f6> ggVG"+y
8 nm <f7> :w<cr>:make<cr>
9 nm <f8> :!@@<cr>
10 nm <f9> :!@@ < in<cr>
11 nm <s-f9> :!(time @@ < in &>> out) &>> out<cr>:sp out<cr>
12
13 au filetype cpp cm @@ ./a.out | se cin fdm=syntax mp=g++\ %\ -std=c++11\ -Wall\ -Wextra\ -O2
14
15 map <c-p> :ha<cr>
16 se pheader=%n\ %f
17
18 au filetype java cm @@ java %< | se cin fdm=syntax mp=javac\ %
19 au filetype python cm @@ python % | se si fdm=indent
20 au bufenter *.kt setf kotlin
21 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
```

```

17 public class Template {
18     // Input
19     private static BufferedReader reader;
20     private static StringTokenizer tokenizer;
21
22     private static String next() {
23         try {
24             while (tokenizer == null || !tokenizer.hasMoreTokens())
25                 tokenizer = new StringTokenizer(reader.readLine());
26         } catch (IOException e) {
27             // do nothing
28         }
29         return tokenizer.nextToken();
30     }
31
32     private static int nextInt() {
33         return Integer.parseInt(next());
34     }
35
36     private static double nextDouble() {
37         return Double.parseDouble(next());
38     }
39
40     private static BigInteger nextBigInteger() {
41         return new BigInteger(next());
42     }
43
44     public static void main(String[] args) {
45         reader = new BufferedReader(new InputStreamReader(System.in));
46         Scanner scanner = new Scanner(System.in);
47         while (scanner.hasNext())
48             scanner.next();
49     }
50
51     // BigInteger & BigDecimal
52     private static void bigDecimal() {
53         BigDecimal a = BigDecimal.valueOf(1.0);
54         BigDecimal b = a.setScale(50, RoundingMode.HALF_EVEN);
55         BigDecimal c = b.abs();
56         // if scale omitted, b.scale is used
57         BigDecimal d = c.divide(b, 50, RoundingMode.HALF_EVEN);
58         // since Java 9
59         BigDecimal e = d.sqrt(new MathContext(50, RoundingMode.HALF_EVEN));
60         BigDecimal x = new BigDecimal(BigInteger.ZERO);
61         BigInteger y = BigDecimal.ZERO.toBigInteger(); // RoundingMode.DOWN
62         y = BigDecimal.ZERO.setScale(0, RoundingMode.HALF_EVEN).unscaledValue();
63     }
64
65     // sqrt for Java 8
66     private static BigDecimal sqrt(BigDecimal a, int scale, RoundingMode mode) {
67         if (a.equals(BigDecimal.ZERO))
68             return BigDecimal.ZERO;
69         a = a.setScale(scale, mode);
70         BigDecimal ans = a;
71         BigDecimal TWO = BigDecimal.valueOf(2L);
72         for (int i = 1; i <= scale; i++)
73             ans = ans.add(a.divide(ans, scale, mode)).divide(TWO, scale, mode);
74         return ans;
75     }
76
77     private static BigInteger sqrt(BigInteger a) {

```



```

78     BigInteger about = BigInteger.ZERO.setBit(a.bitLength() / 2);
79     return sqrt(new BigDecimal(a.toString()), new BigDecimal(about.toString()).setScale(0,
    ↪ RoundingMode.FLOOR).unscaledValue());
80 }
81
82 private static BigDecimal sqrt(BigDecimal a, BigDecimal initial) {
83     if (a.equals(BigDecimal.ZERO))
84         return BigDecimal.ZERO;
85     a = a.setScale(50, RoundingMode.HALF_EVEN);
86     BigDecimal ans = initial;
87     for (int i = 1; i <= 10; i++)
88         ans = ans.add(a.divide(ans, RoundingMode.HALF_EVEN)).divide(BigDecimal.valueOf(2),
    ↪ RoundingMode.HALF_EVEN);
89     return ans;
90 }
91
92 // ArrayList
93 private static void arrayList() {
94     List<Integer> list = new ArrayList<>();
95     // Generic array is banned
96     List[] lists = new List[100];
97     lists[0] = new ArrayList<Integer>();
98     // for List<Integer>, remove(Integer) stands for element, while remove(int) stands for
    ↪ index
99     list.remove(list.get(1));
100    list.remove(list.size() - 1);
101    list.clear();
102 }
103
104 // Queue
105 private static void queue() {
106     LinkedList<Integer> queue = new LinkedList<>();
107     // return the value without popping
108     queue.peek();
109     // pop and return the value
110     queue.poll();
111     Deque<Integer> deque = new ArrayDeque<>();
112     deque.peekFirst();
113     deque.peekLast();
114     deque.pollFirst();
115 }
116
117 // Others
118 private static void others() {
119     Arrays.sort(new int[10]);
120     Arrays.sort(new Integer[10], (a, b) -> {
121         if (a.equals(b)) return 0;
122         if (a > b) return -1;
123         return 1;
124     });
125     Arrays.sort(new Integer[10], Comparator.comparingInt((a) -> (int) a).reversed());
126     long a = 1_000_000_000_000_000L;
127     int b = Integer.MAX_VALUE;
128     int c = 'a';
129 }
130 }

```

6.3 Big Fraction (gy)

```

1 fun gcd(a: Long, b: Long): Long = if (b == 0L) a else gcd(b, a % b)
2
3 class Fraction(val a: BigInteger, val b: BigInteger) {
4     constructor(a: Long, b: Long) : this(BigInteger.valueOf(a / gcd(a, b)), BigInteger.valueOf(b /
5         ↪ gcd(a, b)))
6
7     operator fun plus(o: Fraction): Fraction {
8         var gcd = b.gcd(o.b)
9         val tempProduct = (b / gcd) * (o.b / gcd)
10        var ansA = a * (o.b / gcd) + o.a * (b / gcd)
11        val gcd2 = ansA.gcd(gcd)
12        ansA /= gcd2
13        gcd /= gcd2
14        return Fraction(ansA, gcd * tempProduct)
15    }
16
17    operator fun minus(o: Fraction): Fraction {
18        var gcd = b.gcd(o.b)
19        val tempProduct = (b / gcd) * (o.b / gcd)
20        var ansA = a * (o.b / gcd) - o.a * (b / gcd)
21        val gcd2 = ansA.gcd(gcd)
22        ansA /= gcd2
23        gcd /= gcd2
24        return Fraction(ansA, gcd * tempProduct)
25    }
26
27    operator fun times(o: Fraction): Fraction {
28        val gcd1 = a.gcd(o.b)
29        val gcd2 = b.gcd(o.a)
30        return Fraction((a / gcd1) * (o.a / gcd2), (b / gcd2) * (o.b / gcd1))
31    }
32 }

```

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     if (maxx < fans) {fans = maxx; ans_x = x; ans_y = y;}
23     return maxx;
24 }
25 int main()
26 {
27     srand(time(NULL) + clock());
28     scanf("%d", &n);
29     R double x = 0, y = 0, tot = 0;
30     for (R int i = 1; i <= n; ++i)
31         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;
32     fans = 1e30; x /= tot; y /= tot;
33     R double fnow = f(x, y);
34     for (R double T = 1e4; T > 1e-4; T *= 0.997)
35     {
36         R double nx = x + randp() * T, ny = y + randp() * T, fnext = f(nx, ny);
37         R double delta = fnext - fnow;
38         if (delta < 1e-9 || exp(-delta / T) > rand01())
39         {
40             x = nx; y = ny; fnow = fnext;
41         }
42     }
43     printf("%.3lf %.3lf\n", ans_x, ans_y);
44     return 0;
45 }

```

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;

```

```

32     }
33     printf("%.6lf\n", check((l + r) * 0.5));
34     return 0;
35 }

```

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