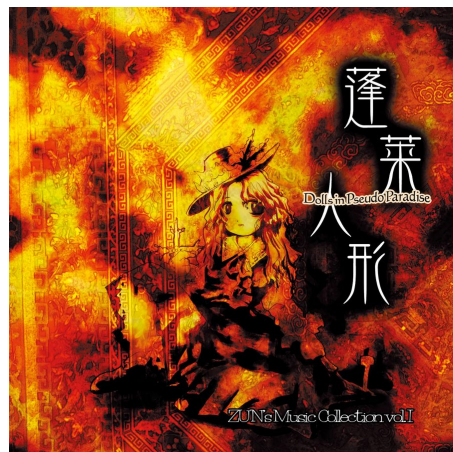


Our village of honest men originally consisted of only eight people.
We all picked up and moved to a mountain in the east. Two years of honest and boring daily life passed us by.
One day, one of us found a little hole by a peach tree.
Yes, after that we wandered into this paradise.
And right away, I quit being human.

— *Dolls in Pseudo Paradise*

蓬莱人形算法模板库

REFERENCE DOCUMENT for *Dolls in Pseudo Paradise*



2024-2025

Harbin Institute of Technology

| | | |
|--|----|--|
| 目录 | | |
| 1 动态规划 | 1 | |
| 1.1 斜率优化 | 1 | |
| 2 数据结构 | 2 | |
| 2.1 平衡树 | 2 | |
| 2.2 珂朵莉树 | 2 | |
| 2.3 可并堆 | 3 | |
| 2.4 Link Cut 树 | 3 | |
| 2.5 线段树 | 4 | |
| 2.6 根号数据结构 | 5 | |
| 3 树论 | 5 | |
| 3.1 点分树 | 5 | |
| 3.2 树哈希 | 7 | |
| 3.3 Prufer 序列 | 7 | |
| 3.4 虚树 | 7 | |
| 4 图论 | 7 | |
| 4.1 三元环计数 | 7 | |
| 4.2 四元环计数 | 8 | |
| 4.3 2-SAT | 8 | |
| 4.4 割点 | 9 | |
| 4.5 边双连通分量 | 9 | |
| 4.6 点双连通分量 | 9 | |
| 4.7 强连通分量 | 9 | |
| 5 网络流 | 10 | |
| 5.1 费用流 | 10 | |
| 5.2 最小割树 | 10 | |
| 5.3 最大流 | 11 | |
| 5.4 上下界费用流 | 12 | |
| 5.5 上下界最大流 | 12 | |
| 6 数学 | 12 | |
| 6.1 线性代数 | 12 | |
| 6.2 大步小步 | 14 | |
| 6.3 中国剩余定理 | 14 | |
| 6.4 狄利克雷前缀和 | 14 | |
| 6.5 万能欧几里得 | 14 | |
| 6.6 扩展欧几里得 | 15 | |
| 6.7 快速离散对数 | 15 | |
| 6.8 原根 | 15 | |
| 6.9 拉格朗日插值 | 16 | |
| 6.10 min-max 容斥 | 16 | |
| 6.11 Barrett 取模 | 16 | |
| 6.12 Pollard's Rho | 16 | |
| 6.13 polya 定理 | 16 | |
| 6.14 min25 筛 | 17 | |
| 6.15 杜教筛 | 18 | |
| 6.16 PN 筛 | 18 | |
| 6.17 常用数表 | 19 | |
| 6.18 二次剩余 | 19 | |
| 6.19 单位根反演 | 20 | |
| 7 多项式 | 20 | |
| 7.1 NTT 全家桶 | 20 | |
| 7.2 FWT 全家桶 | 21 | |
| 7.3 任意模数 NTT | 22 | |
| 8 字符串 | 22 | |
| 8.1 AC 自动机 | 22 | |
| 8.2 扩展 KMP | 22 | |
| 8.3 回文自动机 | 23 | |
| 8.4 后缀数组（倍增） | 23 | |
| 8.5 广义后缀自动机（离线） | 23 | |
| 8.6 广义后缀自动机（在线） | 23 | |
| 8.7 后缀自动机 | 24 | |
| 9 计算几何 | 24 | |
| 10 其他 | 24 | |
| 10.1 笛卡尔树 | 24 | |
| 10.2 CDQ 分治 | 24 | |
| 10.3 自适应辛普森 | 25 | |
| 10.4 模拟退火 | 25 | |
| 10.5 伪随机生成 | 25 | |
| 1 动态规划 | | |
| 1.1 斜率优化 | | |
| 1.1.1 形式 | | |
| 考虑一个经典的 dp 转移方程如下： | | |
| $f_i = \max_{j < i} \{f(j) + w(j, i)\}$ | | |
| 我们将式子拆成三个部分：只跟 i 有关或者与 i, j 均不相关的部分 $a(i)$ ，只跟 j 有关的部分 $b(j)$ ，跟 i, j 均有关的部分 $c(i, j)$ ： | | |
| $f_i = a(i) + \max_{j < i} \{b(j) + c(i, j)\}$ | | |
| 斜率优化可被用来解决这样一个情形： $c(i, j) = ic_j$ 。此时 $b(j) + c(i, j)$ 可视为关于 j 的一次函数。如果 c_j 随着 j 的增大而单调，那么可用单调栈维护；否则可以考虑 CDQ 分治或者在凸包上二分。在凸包上可以使用二分查询最高/最低点。 | | |
| 1.1.2 例题 | | |
| 玩具装箱。原始转移方程为： | | |
| $f_i = \max_{j < i} \{f_j + (s_i - s_j - L')^2\}$ | | |
| 其中 $s_i = i + \sum_{j \leq i} c_i, L' = L + 1$ 。将其分类得到： | | |
| $\begin{aligned} f_i &= \max_{j < i} \{f_j + s_i^2 + s_j^2 + L'^2 - 2s_i s_j + 2s_j L' - 2s_i L'\} \\ &= (s_i^2 - 2s_i L' + L'^2) + \max_{j < i} \{(f_j + s_j^2 + 2s_j L') - 2s_i s_j\} \end{aligned}$ | | |
| 在原始的玩具装箱中， s_j 单调增加，也就是斜率单调增加。因此可以直接使用单调栈维护凸包。同时 s_i 也单调增加，因此可以用指针维护。 | | |

```

1 #include "../header.cpp"
2 int n, L, p, e, C[MAXN], Q[MAXN];
3 f80 S[MAXN], F[MAXN];
4 f80 gtx(int x){ return S[x]; }
5 f80 gty(int x){ return F[x] + S[x] * S[x]; }
6 f80 gtw(int x){ return -2.0 * (L - S[x]); }
7 f80 gtk(int x,int y){ return (gty(y) - gty(x))
  / (gtx(y) - gtx(x)); }
8 int main(){
9     cin >> n >> L;
10    for(int i = 1;i ≤ n;++ i){
11        cin >> C[i];
12        S[i] = S[i - 1] + C[i];
13    }
14    for(int i = 1;i ≤ n;++ i){
15        S[i] += i;
16    }
17    e = p = 1, L ++, Q[p] = 0;
18    for(int i = 1;i ≤ n;++ i){
19        while(e < p && gtk(Q[e], Q[e + 1]) < gtw(i
20            ))
21            ++ e;
22        int j = Q[e];
23        F[i] = F[j] + pow(S[i] - S[j] - L, 2);
24        while(1 < p && gtk(Q[p - 1], Q[p]) > gtk(Q
25            [p], i))
26            e -= (e == p), -- p;
27        Q[++ p] = i;
28    }
29    printf("%.0Lf\n", F[n]);
30    return 0;
31 }

```

2 数据结构

2.1 平衡树

2.1.1 无旋 Treap

```

1 #include "../header.cpp"
2 mt19937_64 MT(114514);
3 namespace Treap{
4     const int SZ = 1e6 + 1e5 + 3;
5     int F[SZ], C[SZ], S[SZ], W[SZ], X[SZ
6         ][2], sz;
7     u64 H[SZ];
8     int newnode(int w){
9         W[++ sz] = w, C[sz] = S[sz] = 1; H[sz] =
10         MT();
11         return sz;
12     }
13 }

```

```

10 }
11 void pushup(int x){
12     S[x] = C[x] + S[X[x][0]] + S[X[x][1]];
13 }
14 pair<int, int> split(int u, int x){
15     if(u == 0)
16         return make_pair(0, 0);
17     if(W[u] > x){
18         auto [a, b] = split(X[u][0], x);
19         X[u][0] = b, pushup(u);
20         return make_pair(a, u);
21     } else {
22         auto [a, b] = split(X[u][1], x);
23         X[u][1] = a, pushup(u);
24         return make_pair(u, b);
25     }
26 }
27 int merge(int a, int b){
28     if(a == 0 || b == 0)
29         return a | b;
30     if(H[a] < H[b]){
31         X[a][1] = merge(X[a][1], b), pushup(a);
32         return a;
33     } else {
34         X[b][0] = merge(a, X[b][0]), pushup(b);
35         return b;
36     }
37 }
38 void insert(int &root, int w){
39     auto [p, q] = split(root, w );
40     auto [a, b] = split( p, w - 1);
41     if(b ≠ 0){
42         ++ S[b], ++ C[b];
43     } else b = newnode(w);
44     p = merge(a, b);
45     root = merge(p, q);
46 }
47 void erase(int &root, int w){
48     auto [p, q] = split(root, w );
49     auto [a, b] = split( p, w - 1);
50     -- C[b], -- S[b];
51     p = C[b] == 0 ? a : merge(a, b);
52     root = merge(p, q);
53 }
54 int find_rank(int &root, int w){
55     int x = root, o = x, a = 0;
56     for(;;x){
57         if(w < W[x])
58             o = x, x = X[x][0];
59         else {
60             a += S[X[x][0]];
61             if(w == W[x]){

```

```

62         o = x; break;
63     }
64     a += C[x];
65     o = x, x = X[x][1];
66 }
67 }
68 return a + 1;
69 }
70 int find_kth(int &root, int w){
71     int x = root, o = x, a = 0;
72     for(;;x){
73         if(w ≤ S[X[x][0]])
74             o = x, x = X[x][0];
75         else {
76             w -= S[X[x][0]];
77             if(w ≤ C[x]){
78                 o = x; break;
79             }
80             w -= C[x];
81             o = x, x = X[x][1];
82         }
83     }
84     return W[x];
85 }
86 int find_pre(int &root, int w){
87     return find_kth(root, find_rank(root, w) -
88         1);
89 }
90 int find_suc(int &root, int w){
91     return find_kth(root, find_rank(root, w +
92         1));
93 }
94 }

```

2.2 珂朵莉树

```

1 #include "../header.cpp"
2 namespace ODT {
3     // <pos_type, value_type>
4     map<int, long long> M;
5     // 分裂为 [1, p) 和 [p, +inf), 返回后者迭代
6     器
7     auto split(int p) {
8         auto it = prev(M.upper_bound(p));
9         return M.insert(
10             it,
11             make_pair(p, it → second)
12         );
13     }
14     // 区间赋值
15     void assign(int l, int r, int v) {
16         auto it = split(l);

```

```

16 split(r + 1);
17 while (it → first ≠ r + 1) {
18     it = M.erase(it);
19 }
20 M[l] = v;
21 }
22 // // 执行操作
23 // void perform(int l, int r) {
24 //     auto it = split(l);
25 //     split(r + 1);
26 //     while (it → first ≠ r + 1) {
27 //         // Do something...
28 //         it = next(it);
29 //     }
30 // }
31 };

```

2.3 可并堆

```

1 #include "../header.cpp"
2 namespace LeftHeap{
3     const int SIZ = 1e5 + 3;
4     int W[SIZ], D[SIZ], L[SIZ], R[SIZ], F[SIZ],
5         S;
6     bool E[SIZ];
7     int merge(int u, int v){
8         if(u == 0 || v == 0)
9             return u | v;
10        if(W[u] > W[v] || (W[u] == W[v] && u > v))
11            swap(u, v);
12        int &lc = L[u];
13        int &rc = R[u];
14        rc = merge(rc, v);
15        if(D[lc] < D[rc])
16            swap(lc, rc);
17        D[u] = min(D[lc], D[rc]) + 1;
18        if(lc ≠ 0) F[lc] = u;
19        if(rc ≠ 0) F[rc] = u;
20        return u;
21 }
22 void pop(int &root){
23     int root0 = merge(L[root], R[root]);
24     F[root0] = root0;
25     F[root] = root0;
26     E[root] = true;
27     root = root0;
28 }
29 int top(int &root){
30     return W[root];
31 }
32 int getfa(int u){
33     return u == F[u] ? u : F[u] = getfa(F[u]);

```

```

33 }
34 int newnode(int w){
35     ++ s;
36     W[s] = w;
37     F[s] = s;
38     D[s] = 1;
39     return s;
40 }
41 }

```

2.4 Link Cut 树

```

1 #include "../header.cpp"
2 namespace LinkCutTree{
3     const int SIZ = 1e5 + 3;
4     int F[SIZ], C[SIZ], S[SIZ], W[SIZ], A[SIZ],
5         X[SIZ][2], size;
6     bool T[SIZ];
7     bool is_root(int x){ return X[F[x]][0] ≠ x
8         && X[F[x]][1] ≠ x;}
9     bool is_rson(int x){ return X[F[x]][1] == x
10        ;}
11     int new_node(int w){
12         ++ size;
13         W[size] = w, C[size] = S[size] = 1;
14         A[size] = w, F[size] = 0;
15         X[size][0] = X[size][1] = 0;
16         return size;
17 }
18 void push_up(int x){
19     S[x] = C[x] + S[X[x][0]] + S[X[x][1]];
20     A[x] = W[x] ^ A[X[x][0]] ^ A[X[x][1]];
21 }
22 void push_down(int x){
23     if(!T[x]) return;
24     int lc = X[x][0], rc = X[x][1];
25     if(lc) T[lc] ^= 1, swap(X[lc][0], X[lc][1]);
26     if(rc) T[rc] ^= 1, swap(X[rc][0], X[rc][1]);
27     T[x] = false;
28 }
29 void update(int x){
30     if(!is_root(x)) update(F[x]); push_down(x);
31 }
32 void rotate(int x){
33     int y = F[x], z = F[y];
34     bool f = is_rson(x);
35     bool g = is_rson(y);
36     if(is_root(y)){
37         F[x] = z, F[y] = x;

```

```

35     X[y][ f] = X[x][!f], F[X[x][!f]] = y;
36     X[x][!f] = y;
37 } else {
38     F[x] = z, F[y] = x;
39     X[z][ g] = x;
40     X[y][ f] = X[x][!f], F[X[x][!f]] = y;
41     X[x][!f] = y;
42 }
43 push_up(y), push_up(x);
44 }
45 void splay(int x){
46     update(x);
47     for(int f = F[x]; f = F[x], !is_root(x);
48         rotate(x))
49         if(!is_root(f)) rotate(is_rson(x) ==
50             is_rson(f) ? f : x);
51 }
52 int access(int x){
53     int p;
54     for(p = 0; x; p = x, x = F[x]){
55         splay(x), X[x][1] = p, push_up(x);
56     }
57     return p;
58 }
59 void make_root(int x){
60     x = access(x);
61     T[x] ^= 1, swap(X[x][0], X[x][1]);
62 }
63 int find_root(int x){
64     access(x), splay(x), push_down(x);
65     while(X[x][0]) x = X[x][0], push_down(x);
66     splay(x);
67     return x;
68 }
69 void link(int x, int y){
70     make_root(x), splay(x), F[x] = y;
71 }
72 void cut(int x, int p){
73     make_root(x), access(p), splay(p), X[p][0]
74     = F[x] = 0;
75 }
76 void modify(int x, int w){
77     splay(x), W[x] = w, push_up(x);
78 }
79 const int MAXN = 1e5 + 3;
80 map<pair<int, int>, bool> M;
81 int n, m;
82 int main(){
83     cin >> n >> m;
84     for(int i = 1; i ≤ n; ++ i){
85         int a; cin >> a;

```

```

84 LinkCutTree :: new_node(a);
85 }
86 for(int i = 1; i ≤ m; ++ i){
87     int o; cin >> o;
88     if(o == 0){
89         int u, v; cin >> u >> v;
90         LinkCutTree :: make_root(u);
91         int p = LinkCutTree :: access(v);
92         printf("%d\n", LinkCutTree :: A[p]);
93     } else if(o == 1){
94         int u, v; cin >> u >> v;
95         int a = LinkCutTree :: find_root(u);
96         int b = LinkCutTree :: find_root(v);
97         if(a ≠ b){
98             LinkCutTree :: link(u, v);
99             M[make_pair(min(u, v), max(u, v))] =
100                 true;
101         } else if(o == 2){
102             int u, v; cin >> u >> v;
103             if(M.count(make_pair(min(u, v), max(u, v)))){
104                 M.erase(make_pair(min(u, v), max(u, v)));
105                 LinkCutTree :: cut(u, v);
106             }
107         } else {
108             int u, w; cin >> u >> w;
109             LinkCutTree :: modify(u, w);
110         }
111     }
112     return 0;
113 }

```

2.5 线段树

2.5.1 李超树

```

1 #include "../..//header.cpp"
2 struct Line{ int id; double k, b; Line() =
   default;};
3 namespace LSeg{
4     const int SIZ = 2e5 + 3;
5     struct Line T[SIZ];
6     #define lc(t) (t << 1)
7     #define rc(t) (t << 1 | 1)
8     bool cmp(int p, Line x, Line y){
9         double w1 = x.k * p + x.b;
10        double w2 = y.k * p + y.b;
11        double d = w1 - w2;
12        if(fabs(d) < 1e-8) return x.id > y.id;
13        return d < 0;

```

```

14 }
15 void merge(int t, int a, int b, Line x, Line
   y){
16     int c = a + b >> 1;
17     if(cmp(c, x, y)) swap(x, y);
18     if(cmp(a, y, x)){
19         T[t] = x; if(a ≠ b) merge(rc(t), c + 1,
   b, T[rc(t)], y);
20     } else {
21         T[t] = x; if(a ≠ b) merge(lc(t), a, c
   , T[lc(t)], y);
22     }
23 }
24 // 插入线段 (l, f(l)) -- (r, f(r))
25 void modify(int t, int a, int b, int l, int
   r, Line x){
26     if(l ≤ a && b ≤ r) merge(t, a, b, T[t],
   x);
27     else {
28         int c = a + b >> 1;
29         if(l ≤ c) modify(lc(t), a, c, l, r, x
   );
30         if(r > c) modify(rc(t), c + 1, b, l, r,
   x);
31     }
32 }
33 // 查询 x = p 位置最高的线段 (有多条取编号最
   小)
34 void query(int t, int a, int b, int p, Line
   &x){
35     if(cmp(p, x, T[t])) x = T[t];
36     if(a ≠ b){
37         int c = a + b >> 1;
38         if(p ≤ c) query(lc(t), a, c, p, x);
39         if(p > c) query(rc(t), c + 1, b, p, x);
40     }
41 }
42 }

```

2.5.2 线段树 3

```

1 #include "../..//header.cpp"
2 int A[MAXN];
3 struct Node{
4     i64 sum; int len, max1, max2, max_cnt,
   his_mx;
5     Node():
6         sum(0), max1(-INF), max2(-INF), max_cnt(0)
   , his_mx(-INF), len(0) {}
7     Node(int w):
8         sum(w), max1(w), max2(-INF), max_cnt(1)
   , his_mx(w), len(1) {}

```

```

9     bool update(int w1, int w2, int h1, int h2){
10         his_mx = max({his_mx, max1 + h1});
11         max1 += w1, max2 += w2;
12         sum += 1ll * w1 * max_cnt + 1ll * w2 * (
   len - max_cnt);
13         return max1 > max2;
14     }
15 };
16 struct Tag{
17     int max_add, max_his_add, umx_add,
   umx_his_add; bool have;
18     void update(int w1, int w2, int h1, int h2){
19         max_his_add = max(max_his_add, max_add +
   h1);
20         umx_his_add = max(umx_his_add, umx_add +
   h2);
21         max_add += w1, umx_add += w2, have = true;
22     }
23     void clear(){
24         max_add = max_his_add = umx_add =
   umx_his_add = have = 0;
25     }
26 };
27 struct Node operator +(Node a, Node b){
28     Node t;
29     t.max1 = max(a.max1, b.max1);
30     if(t.max1 ≠ a.max1){
31         if(a.max1 > t.max2) t.max2 = a.max1;
32     } else{
33         if(a.max2 > t.max2) t.max2 = a.max2;
34         t.max_cnt += a.max_cnt;
35     }
36     if(t.max1 ≠ b.max1){
37         if(b.max1 > t.max2) t.max2 = b.max1;
38     } else{
39         if(b.max2 > t.max2) t.max2 = b.max2;
40         t.max_cnt += b.max_cnt;
41     }
42     t.sum = a.sum + b.sum, t.len = a.len + b.len
   ;
43     t.his_mx = max(a.his_mx, b.his_mx);
44     return t;
45 };
46 namespace Seg{
47     const int SIZ = 2e6 + 3;
48     struct Node W[SIZ]; struct Tag T[SIZ];
49     #define lc(t) (t << 1)
50     #define rc(t) (t << 1 | 1)
51     void push_up(int t, int a, int b){
52         W[t] = W[lc(t)] + W[rc(t)];
53     }
54     void push_down(int t, int a, int b){

```



```

55 if(a == b) T[t].clear();
56 if(T[t].have){
57     int c = a + b >> 1, x = lc(t), y = rc(t);
58     ;
59     int w = max(W[x].max1, W[y].max1);
60     int w1 = T[t].max_add, w2 = T[t].umx_add
61     , w3 = T[t].max_his_add, w4 = T[t].
62     umx_his_add;
63     if(w == W[x].max1)
64         W[x].update(w1, w2, w3, w4),
65         T[x].update(w1, w2, w3, w4);
66     else
67         W[x].update(w2, w2, w4, w4),
68         T[x].update(w2, w2, w4, w4);
69     if(w == W[y].max1)
70         W[y].update(w1, w2, w3, w4),
71         T[y].update(w1, w2, w3, w4);
72     else
73         W[y].update(w2, w2, w4, w4),
74         T[y].update(w2, w2, w4, w4);
75     T[t].clear();
76 }
77 void build(int t, int a, int b){
78     if(a == b){W[t] = Node(A[a]), T[t].clear()
79     ;} else {
80         int c = a + b >> 1; T[t].clear();
81         build(lc(t), a, c);
82         build(rc(t), c + 1, b);
83         push_up(t, a, b);
84     }
85 }
86 void modiadd(int t, int a, int b, int l, int
87 r, int w){
88     if(l ≤ a && b ≤ r){
89         T[t].update(w, w, w, w);
90         W[t].update(w, w, w, w);
91     } else {
92         int c = a + b >> 1; push_down(t, a, b);
93         if(l ≤ c) modiadd(lc(t), a, c, l, r,
94         w);
95         if(r > c) modiadd(rc(t), c + 1, b, l, r
96         , w);
97         push_up(t, a, b);
98     }
99 }
100 void modimin(int t, int a, int b, int l, int
101 r, int w){
102     if(l ≤ a && b ≤ r){
103         if(w ≥ W[t].max1) return; else
104         if(w > W[t].max2){
105             int k = w - W[t].max1;
106             T[t].update(k, 0, k, 0);
107             W[t].update(k, 0, k, 0);
108         } else {
109             int c = a + b >> 1;
110             push_down(t, a, b);
111             modimin(lc(t), a, c, l, r, w);
112             modimin(rc(t), c + 1, b, l, r, w);
113             push_up(t, a, b);
114         }
115     }
116 }
117 Node query(int t, int a, int b, int l, int r
118 ){
119     if(l ≤ a && b ≤ r) return W[t];
120     int c = a + b >> 1; Node ret; push_down(t,
121     a, b);
122     if(l ≤ c) ret = ret + query(lc(t), a, c
123     , l, r);
124     if(r > c) ret = ret + query(rc(t), c + 1,
125     b, l, r);
126     return ret;
127 }
128 int qread();
129 int main(){
130     int n = qread(), m = qread();
131     for(int i = 1; i ≤ n; ++ i)
132         A[i] = qread();
133     Seg :: build(1, 1, n);
134     for(int i = 1; i ≤ m; ++ i){
135         int op = qread();
136         if(op == 1){
137             int l = qread(), r = qread(), w = qread
138             ();
139             Seg :: modiadd(1, 1, n, l, r, w);
140         } else if(op == 2){
141             int l = qread(), r = qread(), w = qread
142             ();
143             Seg :: modimin(1, 1, n, l, r, w);
144         } else if(op == 3){
145             int l = qread(), r = qread();
146             auto p = Seg :: query(1, 1, n, l, r);
147             printf("%lld\n", p.sum);
148         } else if(op == 4){
149             int l = qread(), r = qread();

```

```

143     auto p = Seg :: query(1, 1, n, l, r);
144     printf("%d\n", p.max1);
145     } else if(op == 5){
146         int l = qread(), r = qread();
147         auto p = Seg :: query(1, 1, n, l, r);
148         printf("%d\n", p.his_mx);
149     }
150     }
151     return 0;
152 }

```

```

143     auto p = Seg :: query(1, 1, n, l, r);
144     printf("%d\n", p.max1);
145     } else if(op == 5){
146         int l = qread(), r = qread();
147         auto p = Seg :: query(1, 1, n, l, r);
148         printf("%d\n", p.his_mx);
149     }
150     }
151     return 0;
152 }

```

2.6 根号数据结构

3 树论

3.1 点分树

3.1.1 例题

给定 n 个点组成的树，点有点权 v_i 。 m 个操作，分为两种：

- $0 \times k$ 查询距离 x 不超过 k 的所有点的点权之和；
- $0 \times y$ 将点 x 的点权修改为 y 。

```

1 #include "../header.cpp"
2 vector<int> E[MAXN];
3 namespace LCA{
4     const int SIZ = 1e5 + 3;
5     int D[SIZ], F[SIZ];
6     int P[SIZ], Q[SIZ], o;
7     void dfs(int u, int f){
8         P[u] = ++ o;
9         Q[o] = u;
10        F[u] = f;
11        D[u] = D[f] + 1;
12        for(auto &v : E[u]) if(v ≠ f){
13            dfs(v, u);
14        }
15    }
16    const int MAXH = 18 + 3;
17    int h = 18;
18    int ST[SIZ][MAXH];
19    int cmp(int a, int b){
20        return D[a] < D[b] ? a : b;
21    }
22    int T[SIZ], n;
23    void init(int _n){
24        n = _n;
25        dfs(1, 0);

```

```

26     for(int i = 1; i ≤ n; ++ i)
27         ST[i][0] = Q[i];
28     for(int i = 2; i ≤ n; ++ i)
29         T[i] = T[i >> 1] + 1;
30     for(int i = 1; i ≤ h; ++ i){
31         for(int j = 1; j ≤ n; ++ j) if(j + (1 <<
            i - 1) ≤ n){
32             ST[j][i] = cmp(ST[j][i - 1], ST[j + (1
                << i - 1)][i - 1]);
33         }
34     }
35 }
36 int lca(int a, int b){
37     if(a == b)
38         return a;
39     int l = P[a];
40     int r = P[b];
41     if(l > r)
42         swap(l, r);
43     ++ l;
44     int d = T[r - l + 1];
45     return F[cmp(ST[l][d], ST[r - (1 << d) +
        1][d])];
46 }
47 int dis(int a, int b){
48     return D[a] + D[b] - 2 * D[lca(a, b)];
49 }
50 }
51 namespace BIT{
52     void modify(int D[], int n, int p, int w){
53         ++ p;
54         while(p ≤ n)
55             D[p] += w, p += p & -p;
56     }
57     int query(int D[], int n, int p){
58         if(p < 0) return 0;
59         p = min(n, p + 1);
60         int r = 0;
61         while(p > 0)
62             r += D[p], p -= p & -p;
63         return r;
64     }
65 }
66 namespace PTree{
67     const int SIZ = 1e5 + 3;
68     bool V[SIZ];
69     int S[SIZ], L[SIZ];
70     vector<int> EE[MAXN];
71     int *D1[MAXN];
72     int *D2[MAXN];
73     void dfs1(int s, int &g, int u, int f){
74         S[u] = 1;
75         int maxsize = 0;
76         for(auto &v : E[u]) if(v ≠ f && !V[v]){
77             dfs1(s, g, v, u);
78             if(S[v] > maxsize)
79                 maxsize = S[v];
80             S[u] += S[v];
81         }
82         maxsize = max(maxsize, s - S[u]);
83         if(maxsize ≤ s / 2)
84             g = u;
85     }
86     int n;
87     void build(int s, int &g, int u, int f){
88         dfs1(s, g, u, f);
89         V[g] = true, L[g] = s;
90         for(auto &u : E[g]) if(!V[u]){
91             int h = 0;
92             if(S[u] < S[g]) build(S[u], h, u, 0);
93             else build(s - S[g], h, u, 0);
94             EE[g].push_back(h);
95             EE[h].push_back(g);
96         }
97     }
98     int F[SIZ];
99     void dfs2(int u, int f){
100         F[u] = f;
101         for(auto &v : EE[u]) if(v ≠ f){
102             dfs2(v, u);
103         }
104     }
105     void build(int _n){
106         n = _n;
107         int s = n, g = 0;
108         dfs1(s, g, 1, 0);
109         V[g] = true, L[g] = s;
110         for(auto &u : E[g]){
111             int h = 0;
112             if(S[u] < S[g]) build(S[u], h, u, 0);
113             else build(s - S[g], h, u, 0);
114             EE[g].push_back(h);
115             EE[h].push_back(g);
116         }
117         dfs2(g, 0);
118         for(int i = 1; i ≤ n; ++ i){
119             L[i] += 2;
120             D1[i] = new int[L[i] + 3];
121             D2[i] = new int[L[i] + 3];
122             for(int j = 0; j < L[i] + 3; ++ j)
123                 D1[i][j] = D2[i][j] = 0;
124         }
125     }
126     void modify(int x, int w){
127         int u = x;
128         while(1){
129             BIT :: modify(D1[x], L[x], LCA :: dis(u,
                x), w);
130             int y = F[x];
131             if(y ≠ 0){
132                 int e = LCA :: dis(x, y);
133                 BIT :: modify(D2[x], L[x], LCA :: dis(
                    u, y), w);
134                 x = y;
135             } else break;
136         }
137     }
138     int query(int x, int d){
139         int ans = 0, u = x;
140         while(1){
141             ans += BIT :: query(D1[x], L[x], d - LCA
                :: dis(u, x));
142             int y = F[x];
143             if(y ≠ 0){
144                 int e = LCA :: dis(x, y);
145                 ans -= BIT :: query(D2[x], L[x], d -
                    LCA :: dis(u, y));
146                 x = y;
147             } else break;
148         }
149         return ans;
150     }
151 }
152 int W[MAXN];
153 int main(){
154     ios :: sync_with_stdio(false);
155     int n, m;
156     cin >> n >> m;
157     for(int i = 1; i ≤ n; ++ i){
158         cin >> W[i];
159     }
160     for(int i = 2; i ≤ n; ++ i){
161         int u, v;
162         cin >> u >> v;
163         E[u].push_back(v);
164         E[v].push_back(u);
165     }
166     LCA :: init(n);
167     PTree :: build(n);
168     for(int i = 1; i ≤ n; ++ i)
169         PTree :: modify(i, W[i]);
170     int lastans = 0;
171     for(int i = 1; i ≤ m; ++ i){
172         int op; cin >> op;
173         if(op == 0){
174             int x, d;

```

```

175     cin >> x >> d;
176     x ^= lastans;
177     d ^= lastans;
178     cout << (lastans = PTree :: query(x, d))
        << endl;
179 } else {
180     int x, w;
181     cin >> x >> w;
182     x ^= lastans;
183     w ^= lastans;
184     PTree :: modify(x, -W[x] );
185     PTree :: modify(x, W[x] = w);
186 }
187 }
188 return 0;
189 }

```

3.2 树哈希

3.2.1 用法

给定大小为 n 的以 1 为根的树，计算 h_i 表示子树 i 的哈希值，计算有多少个本质不同的值。

```

1 #include "../header.cpp"
2 u64 xor_shift(u64 x);
3 u64 H[MAXN];
4 vector<int> E[MAXN];
5 void dfs(int u, int f){
6     H[u] = 1;
7     for(auto &v: E[u]) if(v != f){
8         dfs(v, u);
9         H[u] += H[v];
10    }
11    H[u] = xor_shift(H[u]); // !important
12 }
13 int main(){
14     int n;
15     cin >> n;
16     for(int i = 2; i <= n; ++ i){
17         int u, v;
18         cin >> u >> v;
19         E[u].push_back(v);
20         E[v].push_back(u);
21     }
22     dfs(1, 0);
23     sort(H + 1, H + 1 + n);
24     cout << (unique(H + 1, H + 1 + n) - H - 1)
        << endl;
25     return 0;
26 }

```

3.3 Prufer 序列

```

1 #include "../header.cpp"
2 int D[MAXN], F[MAXN], P[MAXN];
3 vector<int> tree2prufer(int n){
4     vector<int> P(n);
5     for(int i = 1, j = 1; i <= n - 2; ++ i, ++ j){
6         while(D[j]) ++ j;
7         P[i] = F[j];
8         while(i <= n - 2 && !--D[P[i]] && P[i] < j
9             )
10             P[i + 1] = F[P[i]], i ++;
11     }
12     return P;
13 }
14 vector<int> prufer2tree(int n){
15     vector<int> F(n);
16     for(int i = 1, j = 1; i <= n - 1; ++ i, ++ j){
17         while(D[j]) ++ j;
18         F[j] = P[i];
19         while(i <= n - 1 && !--D[P[i]] && P[i] < j
20             )
21             F[P[i]] = P[i + 1], i ++;
22     }
23     return F;
24 }

```

3.4 虚树

```

1 #include "../header.cpp"
2 vector<pair<int, int> > E[MAXN];
3 namespace LCA{
4     const int SIZ = 5e5 + 3;
5     int D[SIZ], H[SIZ], F[SIZ], P[SIZ], Q[SIZ],
        o;
6     void dfs(int u, int f){
7         P[u] = ++ o, Q[o] = u, F[u] = f, D[u] = D[
            f] + 1;
8         for(auto &[v, w] : E[u]) if(v != f){
9             H[v] = H[u] + w, dfs(v, u);
10        }
11    }
12    const int MAXH = 18 + 3;
13    int h = 18;
14    int ST[SIZ][MAXH];
15    int cmp(int a, int b){
16        return D[a] < D[b] ? a : b;
17    }
18    int T[SIZ], n;
19    void init(int _n, int root);
20    int lca(int a, int b);
21    int dis(int a, int b);

```

```

22 }
23 bool cmp(int a, int b){
24     return LCA :: P[a] < LCA :: P[b];
25 }
26 bool I[MAXN];
27 vector<int> E1[MAXN], V1;
28 void solve(vector<int> &V){
29     using LCA :: lca; using LCA :: D;
30     stack<int> S;
31     sort(V.begin(), V.end(), cmp);
32     S.push(1);
33     int v, l;
34     for(auto &u : V) I[u] = true;
35     for(auto &u : V) if(u != 1){
36         int f = lca(u, S.top());
37         l = -1;
38         while(D[v = S.top()] > D[f]){
39             if(l != -1)
40                 E1[v].push_back(l);
41             V1.push_back(l = v), S.pop();
42         }
43         if(l != -1)
44             E1[f].push_back(l);
45         if(f != S.top()) S.push(f);
46         S.push(u);
47     }
48     l = -1;
49     while(!S.empty()){
50         v = S.top();
51         if(l != -1) E1[v].push_back(l);
52         V1.push_back(l = v), S.pop();
53     }
54     // dfs(1, 0); // SOLVE HERE !!
55     for(auto &u : V1)
56         E1[u].clear(), I[u] = false;
57     V1.clear();
58 }

```

4 图论

4.1 三元环计数

4.1.1 三元环计数

无向图：考虑将所有点按度数从小往大排序，然后将每条边定向，由排在前面的指向排在后面的，得到一个有向图。然后考虑枚举一个点，再枚举一个点，暴力数，具体见代码。结论是，这样定向后，每个点的出度是 $O(\sqrt{m})$ 的。复杂度 $O(m\sqrt{m})$ 。有向图：不难发现，上述方法枚举

了三个点, 计算有向图三元环也就只需要处理下方向的事, 这个由于算法够暴力, 随便改改就能做了。

4.2 四元环计数

4.2.1 四元环计数

From zpk

- 无向图: 类似, 由于定向后出度结论过于强大, 可以暴力。讨论了三种情况。
- 有向图: 缺少题目, 但应当类似三元环计数有向形式记录定向边和原边的正反关系。因此法最强的结论是定向后出度 $O(\sqrt{m})$, 实际上方法很暴力, 应当不难数有向形式的。

```
1 ll n, m; cin >> n >> m;
2 vector<pair<ll, ll>> Edges(m);
3 vector<vector<ll>> G(n + 2), iG(n + 2);
4 vector<ll> deg(n + 2);
5 for (auto &i, j : Edges) cin >> i >> j, ++
  deg[i], ++deg[j];
6 for (auto [i, j] : Edges) {
7   if (deg[i] > deg[j] || (deg[i] == deg[j]
8     && i > j)) swap(i, j);
9   G[i].emplace_back(j);
10 }
11 vector<ll> val(n + 2);
12 ll ans = 0;
13 for (ll i = 1; i <= n; ++i) {
14   for (auto j : G[i]) ++val[j];
15   for (auto j : G[i]) for (auto k : G[j])
16     ans += val[k];
17   for (auto j : G[i]) val[j] = 0;
18 }
19 // 有向图
20 ll n, m; cin >> n >> m;
21 vector<pair<ll, ll>> Edges(m);
22 vector<vector<pll>> G(n + 2);
23 vector<ll> deg(n + 2);
24 for (auto &i, j : Edges) cin >> i >> j, ++
  deg[i], ++deg[j];
25 for (auto [i, j] : Edges) {
26   ll flg = 0;
27   if (deg[i] > deg[j] || (deg[i] == deg[j]
28     && i > j)) swap(i, j), flg = 1;
29   G[i].emplace_back(j, flg);
30 }
31 vector<ll> in(n + 2), out(n + 2);
32 ll ans = 0;
33 for (ll i = 1; i <= n; ++i) {
34   for (auto [j, w] : G[i]) w ? (++in[j]) : (
35     ++out[j]);
36   for (auto [j, w1] : G[i]) for (auto [k, w2]
37     : G[j]) {
38     if (w1 == w2) ans += w1 ? in[k] : out[
39       k];
40   }
41   for (auto [j, w] : G[i]) in[j] = out[j] =
42     0;
43 }
44 cout << ans << '\n';
```

4.3 2-SAT

4.3.1 例题

n 个变量 m 个条件, 形如若 $x_i = a$ 则 $y_j = b$, 找到任意一组可行解或者报告无解。

```
1 #include "../header.cpp"
2 namespace SCC{
3   const int MAXN = 2e6 + 3;
4   vector<int> V[MAXN];
5   stack<int> S;
6   int D[MAXN], L[MAXN], C[MAXN], o, s;
7   bool F[MAXN], I[MAXN];
8   void add(int u, int v){ V[u].push_back(v); }
9   void dfs(int u){
10     L[u] = D[u] = ++o, S.push(u), I[u] = F[u]
11     = true;
12     for(auto &v : V[u]){
13       if(F[v]){
14         if(I[v]) L[u] = min(L[u], D[v]);
15       } else {
16         dfs(v), L[u] = min(L[u], L[v]);
17       }
18     }
19     if(L[u] == D[u]){
20       int c = ++s;
21       while(S.top() != u){
22         int v = S.top(); S.pop();
23         I[v] = false;
24         C[v] = c;
25       }
26       S.pop(), I[u] = false, C[u] = c;
27     }
28   }
29   const int MAXN = 1e6 + 3;
30   int X[MAXN][2], o;
31   int main(){
32     ios :: sync_with_stdio(false);
33     int n, m;
34     cin >> n >> m;
35     for(int i = 1; i <= n; ++i)
36       X[i][0] = ++o;
37     for(int i = 1; i <= n; ++i)
38       X[i][1] = ++o;
39     for(int i = 1; i <= m; ++i){
40       int a, x, b, y;
41       cin >> a >> x >> b >> y;
42       SCC :: add(X[a][!x], X[b][y]);
43       SCC :: add(X[b][!y], X[a][x]);
44     }
45     for(int i = 1; i <= o; ++i)
46       if(!SCC :: F[i])
47         SCC :: dfs(i);
48     bool ok = true;
49     for(int i = 1; i <= n; ++i){
50       if(SCC :: C[X[i][0]] == SCC :: C[X[i][1]])
51         ok = false;
```

```

52 }
53 if(ok){
54     cout << "POSSIBLE" << endl;
55     for(int i = 1; i ≤ n; ++ i){
56         int a = SCC :: C[X[i][0]];
57         int b = SCC :: C[X[i][1]];
58         if(a < b)
59             cout << 0 << " ";
60         else
61             cout << 1 << " ";
62     }
63     cout << endl;
64 } else {
65     cout << "IMPOSSIBLE" << endl;
66 }
67 return 0;
68 }

```

4.4 割点

```

1 #include "../header.cpp"
2 vector<int> V[MAXN];
3 int n, m, o, D[MAXN], L[MAXN];
4 bool F[MAXN], C[MAXN];
5 void dfs(int u, int g){
6     L[u] = D[u] = ++ o, F[u] = true; int s = 0;
7     for(auto &v : V[u]){
8         if(!F[v]){
9             dfs(v, g), ++ s;
10            L[u] = min(L[u], L[v]);
11            if(u ≠ g && L[v] ≥ D[u]) C[u] = true;
12        } else {
13            L[u] = min(L[u], D[v]);
14        }
15    }
16    if(u == g && s > 1) C[u] = true;
17 }
18 int main(){
19     cin >> n >> m;
20     for(int i = 1; i ≤ m; ++ i){
21         int u, v;
22         cin >> u >> v;
23         V[u].push_back(v);
24         V[v].push_back(u);
25     }
26     for(int i = 1; i ≤ n; ++ i)
27         if(!F[i]) dfs(i, i);
28     vector<int> ANS;
29     for(int i = 1; i ≤ n; ++ i)
30         if(C[i]) ANS.push_back(i);
31     cout << ANS.size() << endl;
32     for(auto &u : ANS)

```

```

33     cout << u << " ";
34     return 0;
35 }

```

4.5 边双连通分量

```

1 #include "../header.cpp"
2 vector<vector<int>> A;
3 vector<pair<int, int>> V[MAXN];
4 stack<int> S;
5 int D[MAXN], L[MAXN], o;
6 bool I[MAXN];
7 void dfs(int u, int l){
8     D[u] = L[u] = ++ o; I[u] = true, S.push(u);
9     int s = 0;
10    for(auto &p : V[u]) {
11        int v = p.first, id = p.second;
12        if(id ≠ l){
13            if(D[v]){
14                if(I[v]) L[u] = min(L[u], D[v]);
15            } else {
16                dfs(v, id), L[u] = min(L[u], L[v]), ++ s;
17            }
18        }
19    }
20    if(D[u] == L[u]){
21        vector<int> T;
22        while(S.top() ≠ u){
23            int v = S.top(); S.pop();
24            T.push_back(v), I[v] = false;
25        }
26        T.push_back(u), S.pop(), I[u] = false;
27        A.push_back(T);
28    }

```

4.6 点双连通分量

```

1 #include "../header.cpp"
2 vector<vector<int>> A;
3 vector<int> V[MAXN];
4 stack<int> S;
5 int D[MAXN], L[MAXN], o; bool I[MAXN];
6 void dfs(int u, int f){
7     D[u] = L[u] = ++ o; I[u] = true, S.push(u);
8     int s = 0;
9     for(auto &v : V[u]) if(v ≠ f){
10        if(D[v]){
11            if(I[v]) L[u] = min(L[u], D[v]);

```

```

12        dfs(v, u), L[u] = min(L[u], L[v]), ++ s;
13    }
14    if(L[v] ≥ D[u]){
15        vector<int> T;
16        while(S.top() ≠ v){
17            int t = S.top(); S.pop();
18            T.push_back(t), I[t] = false;
19        }
20        T.push_back(v), S.pop(), I[v] = false;
21        T.push_back(u);
22        A.push_back(T);
23    }
24 }
25 if(f == 0 && s == 0){
26     A.push_back({u});
27 }
28 }

```

4.7 强连通分量

```

1 #include "../header.cpp"
2 vector<int> V[MAXN];
3 stack<int> S;
4 int D[MAXN], L[MAXN], C[MAXN], o, s;
5 bool F[MAXN], I[MAXN];
6 void add(int u, int v){ V[u].push_back(v); }
7 void dfs(int u){
8     L[u] = D[u] = ++ o, S.push(u), I[u] = F[u] = true;
9     for(auto &v : V[u]){
10        if(F[v]){
11            if(I[v]) L[u] = min(L[u], D[v]);
12        } else {
13            dfs(v), L[u] = min(L[u], L[v]);
14        }
15    }
16    if(L[u] == D[u]){
17        int c = ++ s;
18        while(S.top() ≠ u){
19            int v = S.top(); S.pop();
20            I[v] = false;
21            C[v] = c;
22        }
23        S.pop(), I[u] = false, C[u] = c;
24    }
25 }
26 vector<int> ANS[MAXN];
27 int main(){
28     int n, m;
29     cin >> n >> m;
30     for(int i = 1; i ≤ m; ++ i){
31         int u, v;

```

```

32     cin >> u >> v;
33     V[u].push_back(v);
34 }
35 for(int i = 1; i ≤ n; ++ i)
36     if(!F[i])
37         dfs(i);
38 for(int i = 1; i ≤ n; ++ i){
39     ANS[C[i]].push_back(i);
40 }
41 cout << s << endl;
42 for(int i = 1; i ≤ n; ++ i) if(F[i]){
43     int c = C[i];
44     sort(ANS[c].begin(), ANS[c].end());
45     for(auto &u : ANS[c])
46         cout << u << " ", F[u] = false;
47     cout << endl;
48 }
49 return 0;
50 }

```

```

24     int u = Q.front(); Q.pop(), I[u] = false
25 ;
26 for(int i = H[u]; i = N[i]){
27     const int &v = V[i];
28     const int &f = F[i];
29     const int &w = W[i];
30     if(f && D[u] + w < D[v]){
31         D[v] = D[u] + w;
32         if(!I[v]) Q.push(v), I[v] = true;
33     }
34 }
35 return D[t] ≠ INFL;
36 }
37 int C[MAXN]; bool T[MAXN];
38 pair<i64, i64> dfs(int s, int t, int u, i64
39     maxf){
40     if(u == t)
41         return make_pair(maxf, 0);
42     i64 totf = 0;
43     i64 totc = 0;
44     T[u] = true;
45     for(int &i = C[u]; i = N[i]){
46         const int &v = V[i];
47         const int &f = F[i];
48         const int &w = W[i];
49         if(f && D[v] = D[u] + w && !T[v]){
50             auto p = dfs(s, t, v, min(1ll * F[i],
51                 maxf));
52             i64 f = p.first;
53             i64 c = p.second;
54             F[i] -= f;
55             F[i ^ 1] += f;
56             totf += f;
57             totc += 1ll * f * W[i] + c;
58             maxf -= f;
59             if(maxf == 0){
60                 T[u] = false;
61                 return make_pair(totf, totc);
62             }
63         }
64     }
65     T[u] = false;
66     return make_pair(totf, totc);
67 }
68 pair<i64, i64> mcmf(int s, int t){
69     i64 ans1 = 0;
70     i64 ans2 = 0;
71     pair<i64, i64> r;
72     while(spfa(s, t)){
73         memcpy(C, H, sizeof(int) * (n + 3));
74         r = dfs(s, t, s, INFL);
75         ans1 += r.first;

```

```

74         ans2 += r.second;
75     }
76     return make_pair(ans1, ans2);
77 }
78 }
79 int qread();
80 int main(){
81     int n = qread(), m = qread(), s = qread(), t
82     = qread();
83     for(int i = 1; i ≤ m; ++ i){
84         int u = qread(), v = qread(), f = qread(),
85         c = qread();
86         MCMF :: add(u, v, f, c);
87     }
88     pair<long long, long long> ans = MCMF ::
89     mcmf(s, t);
90     printf("%lld %lld\n", ans.first, ans.second)
91 ;
92     return 0;
93 }

```

5 网络流

5.1 费用流

```

1 #include "../header.cpp"
2 namespace MCMF{
3     int H[MAXN], V[MAXM], N[MAXM], W[MAXM], F[
4     MAXM], o = 1, n;
5     void add(int u, int v, int f, int c){
6         V[++ o] = v, N[o] = H[u], H[u] = o, F[o] =
7         f, W[o] = c;
8         V[++ o] = u, N[o] = H[v], H[v] = o, F[o] =
9         0, W[o] = -c;
10         n = max(n, u);
11         n = max(n, v);
12     }
13 void clear(){
14     for(int i = 1; i ≤ n; ++ i)
15         H[i] = 0;
16     n = 0, o = 1;
17 }
18 bool I[MAXN];
19 i64 D[MAXN];
20 bool spfa(int s, int t){
21     queue<int> Q;
22     Q.push(s), I[s] = true;
23     for(int i = 1; i ≤ n; ++ i)
24         D[i] = INFL;
25     D[s] = 0;
26     while(!Q.empty()){

```

5.2 最小割树

5.2.1 用法

给定无向图求出最小割树，点 u 和 v 作为起点终点的
最小割为树上 u 到 v 路径上边权的最小值。

```

1 #include "../header.cpp"
2 namespace Dinic{
3     const long long INF = 1e18;
4     const int SZ = 1e5 + 3;
5     int n, m;
6     int H[SZ], V[SZ], N[SZ], F[SZ], t = 1;
7     int add(int u, int v, int f){
8         V[++ t] = v, N[t] = H[u], F[t] = f, H[u] =
9         t;
10         V[++ t] = u, N[t] = H[v], F[t] = 0, H[v] =
11         t;
12         n = max(n, u);
13         n = max(n, v);
14         return t - 1;
15 }
16 void clear(){
17     for(int i = 1; i ≤ n; ++ i)
18         H[i] = 0;
19     n = m = 0, t = 1;
20 }
21 int D[SZ];
22 bool bfs(int s, int t){
23     queue<int> Q;
24     for(int i = 1; i ≤ n; ++ i)

```

```

23     D[i] = 0;
24     Q.push(s), D[s] = 1;
25     while(!Q.empty()){
26         int u = Q.front(); Q.pop();
27         for(int i = H[u]; i; i = N[i]){
28             const int &v = V[i];
29             const int &f = F[i];
30             if(f != 0 && !D[v]){
31                 D[v] = D[u] + 1;
32                 Q.push(v);
33             }
34         }
35     }
36     return D[t] != 0;
37 }
38 int C[SIZ];
39 long long dfs(int s, int t, int u, long long
maxf){
40     if(u == t)
41         return maxf;
42     long long totf = 0;
43     for(int &i = C[u]; i; i = N[i]){
44         const int &v = V[i];
45         const int &f = F[i];
46         if(D[v] == D[u] + 1){
47             long long resf = dfs(s, t, v, min(maxf
, 1ll * f));
48             totf += resf;
49             maxf -= resf;
50             F[i] -= resf;
51             F[i ^ 1] += resf;
52             if(maxf == 0)
53                 return totf;
54         }
55     }
56     return totf;
57 }
58 long long dinic(int s, int t){
59     long long ans = 0;
60     while(bfs(s, t)){
61         memcpy(C, H, sizeof(int) * (n + 3));
62         ans += dfs(s, t, s, INF);
63     }
64     return ans;
65 }
66 }
67 namespace GHTree{
68     const int MAXN = 500 + 5;
69     const int MAXM = 1500 + 5;
70     const int INF = 1e9;
71     int n, m, U[MAXM], V[MAXM], W[MAXM], A[MAXM]
, B[MAXM];

```

```

72 void add(int u, int v, int w){
73     ++ m;
74     U[m] = u;
75     V[m] = v;
76     W[m] = w;
77     A[m] = Dinic :: add(u, v, w);
78     B[m] = Dinic :: add(v, u, w);
79     n = max(n, u);
80     n = max(n, v);
81 }
82 vector <pair<int, int> > E[MAXN];
83 void build(vector <int> N){
84     int s = N.front();
85     int t = N.back();
86     if(s == t) return;
87     for(int i = 1; i ≤ m; ++ i){
88         int a = A[i]; Dinic :: F[a] = W[i],
Dinic :: F[a ^ 1] = 0;
89         int b = B[i]; Dinic :: F[b] = W[i],
Dinic :: F[b ^ 1] = 0;
90     }
91     int w = Dinic :: dinic(s, t);
92     E[s].push_back(make_pair(t, w));
93     E[t].push_back(make_pair(s, w));
94     vector <int> P;
95     vector <int> Q;
96     for(auto &u : N){
97         if(Dinic :: D[u] != 0)
98             P.push_back(u);
99         else
100             Q.push_back(u);
101     }
102     build(P), build(Q);
103 }
104 int D[MAXN];
105 int cut(int s, int t){
106     queue <int> Q; Q.push(s);
107     for(int i = 1; i ≤ n; ++ i)
108         D[i] = -1;
109     D[s] = INF;
110     while(!Q.empty()){
111         int u = Q.front(); Q.pop();
112         for(auto &e : E[u]){
113             int v = e.first;
114             int w = e.second;
115             if(D[v] == -1){
116                 D[v] = min(D[u], w);
117                 Q.push(v);
118             }
119         }
120     }
121     return D[t];

```

```

122 }
123 }

```

5.3 最大流

```

1 #include "../header.cpp"
2 namespace Dinic{
3     const i64 INF = 1e18;
4     const int SIZ = 5e5 + 3;
5     int n;
6     int H[MAXN], V[MAXM], N[MAXM], F[MAXM], t =
1;
7     void add(int u, int v, int f){
8         V[++ t] = v, N[t] = H[u], F[t] = f, H[u] =
t;
9         V[++ t] = u, N[t] = H[v], F[t] = 0, H[v] =
t;
10        n = max(n, u);
11        n = max(n, v);
12    }
13    void clear(){
14        for(int i = 1; i ≤ n; ++ i)
15            H[i] = 0;
16        n = 0, t = 1;
17    }
18    i64 D[MAXN];
19    bool bfs(int s, int t){
20        queue <int> Q;
21        for(int i = 1; i ≤ n; ++ i)
22            D[i] = 0;
23        Q.push(s), D[s] = 1;
24        while(!Q.empty()){
25            int u = Q.front(); Q.pop();
26            for(int i = H[u]; i; i = N[i]){
27                const int &v = V[i];
28                const int &f = F[i];
29                if(f != 0 && !D[v]){
30                    D[v] = D[u] + 1;
31                    Q.push(v);
32                }
33            }
34        }
35        return D[t] != 0;
36    }
37    int C[MAXN];
38    i64 dfs(int s, int t, int u, i64 maxf){
39        if(u == t)
40            return maxf;
41        i64 totf = 0;
42        for(int &i = C[u]; i; i = N[i]){
43            const int &v = V[i];
44            const int &f = F[i];

```

```

45     if(f && D[v] == D[u] + 1){
46         i64 f = dfs(s, t, v, min(1ll * f, maxf
47             ));
48         F[i] -= f, F[i ^ 1] += f, totf += f,
49             maxf -= f;
50         if(maxf == 0)
51             return totf;
52     }
53     return totf;
54 }
55 i64 dinic(int s, int t){
56     i64 ans = 0;
57     while(bfs(s, t)){
58         memcpy(C, H, sizeof(int) * (n + 3));
59         ans += dfs(s, t, s, INFL);
60     }
61     return ans;
62 }

```

```

22     }
23     auto res = mcmf(s, t);
24     if(res.first != sum)
25         return -1;
26     return res.second + cost0;
27 }
28 i64 solve(int s0, int t0){
29     add0(t0, s0, INF, 0);
30     int s = ++n;
31     int t = ++n;
32     i64 sum = 0;
33     for(int i = 1; i ≤ n - 2; ++i){
34         if(G[i] < 0)
35             add0(i, t, -G[i], 0);
36         else
37             add0(s, i, G[i], 0), sum += G[i];
38     }
39     auto res = mcmf(s, t);
40     if(res.first != sum)
41         return -1;
42     return res.second + cost0;
43 }
44 }

```

```

19     }
20     bool solve(){
21         int s = ++n;
22         int t = ++n;
23         i64 sum = 0;
24         for(int i = 1; i ≤ n - 2; ++i){
25             if(G[i] < 0)
26                 add0(i, t, -G[i]);
27             else
28                 add0(s, i, G[i]), sum += G[i];
29         }
30         auto res = dinic(s, t);
31         if(res != sum)
32             return true;
33         return false;
34     }
35     i64 solve(int s0, int t0){
36         add0(t0, s0, INF);
37         int s = ++n;
38         int t = ++n;
39         i64 sum = 0;
40         for(int i = 1; i ≤ n - 2; ++i){
41             if(G[i] < 0)
42                 add0(i, t, -G[i]);
43             else
44                 add0(s, i, G[i]), sum += G[i];
45         }
46         auto res = dinic(s, t);
47         if(res != sum)
48             return -1;
49         return dinic(s0, t0);
50     }
51 }

```

5.4 上下界费用流

5.4.1 用法

- `add(u, v, l, r, c)`: 连一条容量在 $[l, r]$ 的从 u 到 v 的费用为 c 的边;
- `solve()`: 计算无源汇最小费用可行流;
- `solve(s, t)`: 计算有源汇最小费用最大流。

5.5 上下界最大流

5.5.1 用法

- `add(u, v, l, r, c)`: 连一条容量在 $[l, r]$ 的从 u 到 v 的边;
- `solve()`: 检查是否存在无源汇可行流;
- `solve(s, t)`: 计算有源汇最大流。

```

1 #define add add0
2 #include "flow-cost.cpp"
3 #undef add
4 namespace MCMF{
5     i64 cost0;
6     int G[MAXN];
7     void add(int u, int v, int l, int r, int c){
8         G[v] += l;
9         G[u] -= l;
10        cost0 += 1ll * l * c;
11        add0(u, v, r - l, c);
12    }
13    i64 solve(){
14        int s = ++n;
15        int t = ++n;
16        i64 sum = 0;
17        for(int i = 1; i ≤ n - 2; ++i){
18            if(G[i] < 0)
19                add0(i, t, -G[i], 0);
20            else
21                add0(s, i, G[i], 0), sum += G[i];

```

```

1 #define add add0
2 #include "flow-max.cpp"
3 #undef add
4 namespace Dinic{
5     int G[MAXN];
6     void add(int u, int v, int l, int r){
7         G[v] += l;
8         G[u] -= l;
9         add0(u, v, r - l);
10    }
11    void clear(){
12        for(int i = 1; i ≤ t; ++i){
13            N[i] = F[i] = V[i] = 0;
14        }
15        for(int i = 1; i ≤ n; ++i){
16            H[i] = G[i] = C[i] = 0;
17        }
18        t = 1, n = 0;

```

6 数学

6.1 线性代数

6.1.1 行列式

```

1 #include "../header.cpp"
2 struct Mat{
3     int n, m, W[MAXN][MAXN];
4     Mat(int _n = 0, int _m = 0){
5         n = _n, m = _m;
6         for(int i = 1; i ≤ n; ++i)
7             for(int j = 1; j ≤ m; ++j)
8                 W[i][j] = 0;
9     }
10 };
11 int mat_det(Mat a){

```



```

12 int ans = 1;
13 const int &n = a.n;
14 for(int i = 1; i ≤ n; ++ i){
15     int f = -1;
16     for(int j = i; j ≤ n; ++ j) if(a.W[j][i] ≠
17         0){
18         f = j; break;
19     }
20     if(f == -1) return 0;
21     if(f ≠ i){
22         for(int j = 1; j ≤ n; ++ j)
23             swap(a.W[i][j], a.W[f][j]);
24         ans = MOD - ans;
25     }
26     for(int j = i + 1; j ≤ n; ++ j) if(a.W[j][i]
27         ){
28         while(a.W[j][i]){
29             int u = a.W[i][i], v = a.W[j][i];
30             if(u > v){
31                 for(int k = 1; k ≤ n; ++ k)
32                     swap(a.W[i][k], a.W[j][k]);
33                 ans = MOD - ans, swap(u, v);
34             }
35             int rate = v / u;
36             for(int k = 1; k ≤ n; ++ k){
37                 a.W[j][k] = (a.W[j][k] - 1ll * rate
38                     * a.W[i][k] % MOD + MOD) % MOD;
39             }
40         }
41     }
42     ans = 1ll * ans * a.W[i][i] % MOD;
43     return ans;
44 }
45 int main(){
46     int n; cin >> n;
47     Mat A(n, n);
48     for(int i = 1; i ≤ n; ++ i)
49         for(int j = 1; j ≤ n; ++ j)
50             cin >> A.W[i][j], A.W[i][j] %= MOD;
51     cout << mat_det(A) << endl;
52     return 0;
}

```

6.1.2 矩阵树

LGV 定理叙述 设 G 是一张有向无环图，边带权，每个点的度数有限。给定起点集合 $A = \{a_1, a_2, \dots, a_n\}$ ，终点集合 $B = \{b_1, b_2, \dots, b_n\}$ 。

- 一段路径 $p: v_0 \xrightarrow{w_1} v_1 \xrightarrow{w_2} v_2 \rightarrow \dots \rightarrow^{w_k} v_k$ 的边权被定义为 $\omega(p) = \prod w_{i_0}$ 。
- 一对顶点 (a, b) 的权值定义为 $e(a, b) = \sum_{p: a \rightarrow b} \omega(p)$ 。

设矩阵 M 如下：

$$M = \begin{pmatrix} e(a_1, b_1) & e(a_1, b_2) & \cdots & e(a_1, b_n) \\ e(a_2, b_1) & e(a_2, b_2) & \cdots & e(a_2, b_n) \\ \vdots & \vdots & \ddots & \vdots \\ e(a_n, b_1) & e(a_n, b_2) & \cdots & e(a_n, b_n) \end{pmatrix}$$

从 A 到 B 得到一个不交的路径组 $p = (p_1, p_2, \dots, p_n)$ ，其中从 a_i 到达 b_{π_i} ， π 是一个排列。定义 $\sigma(\pi)$ 是 π 逆序对的数量。

给出 LGV 的叙述如下：

$$\det(M) = \sum_{p: A \rightarrow B} (-1)^{\sigma(\pi)} \prod_{i=1}^n \omega(p_i)$$

可以将边权视作边的重数，那么 $e(a, b)$ 就可以视为从 a 到 b 的不同路径方案数。

矩阵树定理 对于无向图，

- 定义度数矩阵 $D_{i,j} = [i = j] \deg(i)$ ；
- 定义邻接矩阵 $E_{i,j} = E_{j,i}$ 是从 i 到 j 的边数个数；
- 定义拉普拉斯矩阵 $L = D - E$ 。

对于无向图的矩阵树定理叙述如下：

$$t(G) = \det(L_i) = \frac{1}{n} \lambda_1 \lambda_2 \cdots \lambda_{n-1}$$

其中 L_i 是将 L 删去第 i 行和第 i 列得到的子式。

对于有向图，类似于无向图定义入度矩阵、出度矩阵、邻接矩阵 $D^{\text{in}}, D^{\text{out}}, E$ ，同时定义拉普拉斯矩阵 $L^{\text{in}} = D^{\text{in}} - E, L^{\text{out}} = E$ 。

$$t^{\text{leaf}}(G, k) = \det(L_k^{\text{in}})$$

$$t^{\text{root}}(G, k) = \det(L_k^{\text{out}})$$

其中 $t^{\text{leaf}}(G, k)$ 表示以 k 为根的叶向树， $t^{\text{root}}(G, k)$ 表示以 k 为根的根向树。

BEST 定理 对于一个有向欧拉图 G ，记点 i 的出度为 out_i ，同时 G 的根向生成树个数为 T 。 T 可以任意选取根。则 G 的本质不同的欧拉回路个数为：

$$T \prod_i (\text{out}_i - 1)!$$

```

1 #include "../header.cpp"
2 struct Mat{
3     int n, m;
4     int W[MAXN][MAXN];
5     Mat(int _n = 0, int _m = 0){
6         n = _n;
7         m = _m;
8         for(int i = 1; i ≤ n; ++ i)
9             for(int j = 1; j ≤ m; ++ j)
10                 W[i][j] = 0;
11     }
12 };
13 int mat_det(Mat a){
14     int ans = 1;
15     const int &n = a.n;
16     for(int i = 1; i ≤ n; ++ i){
17         int f = -1;
18         for(int j = i; j ≤ n; ++ j) if(a.W[j][i] ≠
19             0){
20             f = j;
21             break;
22         }
23         if(f == -1){
24             return 0;
25         }
26         if(f ≠ i){
27             for(int j = 1; j ≤ n; ++ j)
28                 swap(a.W[i][j], a.W[f][j]);
29             ans = MOD - ans;
30         }
31         for(int j = i + 1; j ≤ n; ++ j) if(a.W[j][i]
32             ){
33             while(a.W[j][i]){
34                 int u = a.W[i][i];
35                 int v = a.W[j][i];
36                 if(u > v){
37                     for(int k = 1; k ≤ n; ++ k)
38                         swap(a.W[i][k], a.W[j][k]);
39                     ans = MOD - ans;
40                     swap(u, v);
41                 }
42                 int rate = v / u;

```

```

41     for(int k = 1; k ≤ n; ++ k){
42         a.W[j][k] = (a.W[j][k] - 1ll * rate
43             * a.W[i][k] % MOD + MOD) % MOD;
44     }
45 }
46 }
47 for(int i = 1; i ≤ n; ++ i)
48     ans = 1ll * ans * a.W[i][i] % MOD;
49 return ans;
50 }
51 int D[MAXN];
52 int W[MAXN][MAXN];
53 int main(){
54     int n, m, t;
55     cin >> n >> m >> t;
56     for(int i = 1; i ≤ m; ++ i){
57         int u, v, w;
58         cin >> u >> v >> w;
59         if(u ≠ v){
60             if(t == 0){ // 无向图
61                 D[u] = (D[u] + w) % MOD;
62                 D[v] = (D[v] + w) % MOD;
63                 W[u][v] = (W[u][v] + w) % MOD;
64                 W[v][u] = (W[v][u] + w) % MOD;
65             } else { // 叶向树
66                 D[v] = (D[v] + w) % MOD;
67                 W[u][v] = (W[u][v] + w) % MOD;
68             }
69         }
70     }
71     Mat A(n - 1, n - 1);
72     for(int i = 2; i ≤ n; ++ i)
73         for(int j = 2; j ≤ n; ++ j) // 以 1 为根的
74             A.W[i - 1][j - 1] = MOD - W[i][j];
75     for(int i = 2; i ≤ n; ++ i)
76         A.W[i - 1][i - 1] = (D[i] + A.W[i - 1][i - 1]) % MOD;
77     cout << mat_det(A) << endl;
78     return 0;
79 }

```

6.2 大步小步

6.2.1 用法

给定 a, p 求出 x 使得 $a^x = y \pmod{p}$, 其中 p 为质数。

```

3 unordered_map <int, int> M;
4 int solve(int a, int y, int p){ // a ^ x =
5     y (mod p)
6     M.clear();
7     int B = sqrt(p);
8     int w1 = y, u1 = power(a, p - 2, p);
9     int w2 = 1, u2 = power(a, B, p);
10    for(int i = 0; i < B; ++ i){
11        M[w1] = i;
12        w1 = 1ll * w1 * u1 % p;
13    }
14    for(int i = 0; i < p / B; ++ i){
15        if(M.count(w2)){
16            return i * B + M[w2];
17        }
18        w2 = 1ll * w2 * u2 % p;
19    }
20    return -1;
21 }

```

6.3 中国剩余定理

6.3.1 定理

对于线性方程：

$$\begin{cases} x \equiv a_1 \pmod{m_1} \\ x \equiv a_2 \pmod{m_2} \\ \dots \\ x \equiv a_n \pmod{m_n} \end{cases}$$

如果 a_i 两两互质, 可以得到 x 的解 $x \equiv L \pmod{M}$,

其中 $M = \prod m_i$, 而 L 由下式给出：

$$L = \left(\sum a_i m_i \times ((M/m_i)^{-1} \bmod m_i) \right) \bmod M$$

```

1 #include "../header.cpp"
2 i64 A[MAXN], B[MAXN], M = 1;
3 i64 exgcd(i64 a, i64 b, i64 &x, i64 &y);
4 int main(){
5     int n; cin >> n;
6     for(int i = 1; i ≤ n; ++ i){
7         cin >> B[i] >> A[i];
8         M = M * B[i];
9     }
10    i64 L = 0;
11    for(int i = 1; i ≤ n; ++ i){

```

```

12        i64 m = M / B[i], b, k;
13        exgcd(m, B[i], b, k);
14        L = (L + (__int128)A[i] * m * b) % M;
15    }
16    L = (L % M + M) % M;
17    cout << L << endl;
18    return 0;
19 }

```

6.4 狄利克雷前缀和

6.4.1 用法

计算：

$$s(i) = \sum_{d|i} f_d$$

```

1 #include "../header.cpp"
2 unsigned A[MAXN];
3 int p, P[MAXN]; bool V[MAXN];
4 void solve(int n){
5     for(int i = 2; i ≤ n; ++ i){
6         if(!V[i]){
7             P[++ p] = i;
8             for(int j = 1; j ≤ n / i; ++ j){ // 前缀
9                 和
10                 A[j * i] += A[j];
11             }
12             for(int j = 1; j ≤ p && P[j] ≤ n / i; ++ j)
13                 V[i * P[j]] = true;
14             if(i % P[j] == 0) break;
15         }
16     }
17 }

```

6.5 万能欧几里得

6.5.1 类欧几里得（万能欧几里得）

From *zpk*

一种神奇递归, 对 $y = \left\lfloor \frac{Ax+B}{C} \right\rfloor$ 向右和向上走的每步进行压缩, 做到 $O(\log V)$ 复杂度。其中 $A \geq C$ 就是直接压缩, 向右之后必有至少 $\lfloor A/C \rfloor$ 步向上。 $A < C$ 实际上切换 x, y 轴后, 相当于压缩了一个上取整折线, 而上取整下取整可以互化, 便又可以递归。

```

1 #include "../header.cpp"
2 namespace BSGS {

```

代码中从 $(0, 0)$ 走到 $(n, \lfloor (An + B)/C \rfloor)$, 假设了 $A, B, C \geq 0, C \neq 0$ (类欧基本都作此假设), U, R 矩阵是从右往左乘的, 对列向量进行优化, 和实际操作顺序恰好相反。快速幂的 \log 据说可以被递归过程均摊掉, 实际上并不会导致变成两个 \log 。

```
1 Matrix solve(ll n, ll A, ll B, ll C, Matrix R,
2   Matrix U) { // (0, 0) 走到 (n, (An+B)/C)
3   if (A ≥ C) return solve(n, A % C, B, C, U
4     .qpow(A / C) * R, U);
5   ll l = B / C, r = (A * n + B) / C;
6   if (l == r) return R.qpow(n) * U.qpow(l);
7   // l = r → l = r or A = 0 or n = 0.
8   ll p = (C * r - B - 1) / A + 1;
9   return R.qpow(n - p) * U * solve(r - l -
10    1, C, C - B % C + A - 1, A, U, R) * U.
11    qpow(l);
12 }
```

6.6 扩展欧几里得

6.6.1 内容

给定 a, b , 求出 $ax + by = \gcd(a, b)$ 的一组 x, y 。

```
1 int exgcd(int a, int b, int &x, int &y){
2   if(a == 0){
3     x = 0, y = 1; return b;
4   } else {
5     int x0 = 0, y0 = 0;
6     int d = exgcd(b % a, a, x0, y0);
7     x = y0 - (b / a) * x0;
8     y = x0;
9     return d;
10  }
11 }
```

6.7 快速离散对数

6.7.1 用法

给定原根 g 以及模数 mod , T 次询问 x 的离散对数。

复杂度 $\mathcal{O}(\text{mod}^{2/3} + T \log \text{mod})$ 。

```
1 #include "../header.cpp"
2 namespace BSGS {
3   unordered_map<int, int> M;
4   int B, U, P, g;
5   void init(int g, int P0, int B0);
6   int solve(int y);
7 }
```

```
7 }
8 const int MAXN = 1e5 + 3;
9 int H[MAXN], P[MAXN], H0, p, h, g, mod;
10 bool V[MAXN];
11 int solve(int x){
12   if(x ≤ h) return H[x];
13   int v = mod / x, r = mod % x;
14   if(r < x - r) return ((H0 + solve(r)) % (mod
15     - 1) - H[v] + mod - 1) % (mod - 1);
16   else return (solve(x - r) - H[v +
17     1] + mod - 1) % (mod - 1);
18 }
19 int main(){
20   ios :: sync_with_stdio(false);
21   cin.tie(nullptr);
22   cin >> g >> mod;
23   h = sqrt(mod) + 1;
24   BSGS :: init(g, mod, sqrt(1ll * mod * sqrt(
25     mod) / log10(mod)));
26   H0 = BSGS :: solve(mod - 1);
27   H[1] = 0;
28   for(int i = 2; i ≤ h; ++ i){
29     if(!V[i]){
30       P[++ p] = i;
31       H[i] = BSGS :: solve(i);
32     }
33     for(int j = 1; j ≤ p && P[j] ≤ h / i; ++ j)
34       {
35         int &p = P[j];
36         H[i * p] = (H[i] + H[p]) % (mod - 1);
37         V[i * p] = true;
38         if(i % p == 0) break;
39       }
40   }
41   int T; cin >> T;
42   while(T --){
43     int x; cin >> x;
44     cout << solve(x) << "\n";
45   }
46   return 0;
47 }
```

6.8 原根

6.8.1 用法

计算 P 的最小原根。

原根表, 其中 $P = r \times 2^k$, 对应原根为 g_o 。

| Prime | g | Prime | g |
|------------|-----|---------------------|-----|
| 104857601 | 3 | 7881299347898369 | 6 |
| 167772161 | 3 | 31525197391593473 | 3 |
| 469762049 | 3 | 180143985094819841 | 6 |
| 998244353 | 3 | 1945555039024054273 | 5 |
| 1004535809 | 3 | 4179340454199820289 | 3 |

```
1 #include "../header.cpp"
2 int getphi(int x){
3   int t = x, r = x;
4   for(int i = 2; i ≤ x / i; ++ i){
5     if(t % i == 0){
6       r = r / i * (i - 1);
7       while(t % i == 0)
8         t /= i;
9     }
10  }
11  if(t ≠ 1){
12    r = r / t * (t - 1);
13  }
14  return r;
15 }
16 vector<int> getprime(int x){
17   vector<int> p;
18   int t = x;
19   for(int i = 2; i ≤ x / i; ++ i){
20     if(t % i == 0){
21       p.push_back(i);
22       while(t % i == 0)
23         t /= i;
24     }
25   }
26   if(t ≠ 1)
27     p.push_back(x);
28   return p;
29 }
30 bool test(int g, int m, int mm, vector<int> &p)
31 ){
32   for(auto &p: P){
33     if(power(g, mm / p, m) == 1)
34       return false;
35   }
36   return true;
37 }
38 int get_genshin(int m){
39   int mm = getphi(m);
40   vector<int> P = getprime(mm);
41 }
```

```

40  for(int i = 1; ++ i){
41      if(test(i, m, mm, P))
42          return i;
43  }
44  }

```

6.9 拉格朗日插值

6.9.1 定理

给定 n 个横坐标不同的点 (x_i, y_i) , 可以唯一确定一个 $n-1$ 阶多项式如下:

$$f(x) = \sum_{i=1}^n \frac{\prod_{j \neq i} (x - x_j)}{\prod_{j \neq i} (x_i - x_j)} \cdot y_i$$

6.10 min-max 容斥

6.10.1 定理

$$\max_{i \in S} \{x_i\} = \sum_{T \subseteq S} (-1)^{|T|-1} \min_{j \in T} \{x_j\}$$

$$\min_{i \in S} \{x_i\} = \sum_{T \subseteq S} (-1)^{|T|-1} \max_{j \in T} \{x_j\}$$

期望意义下上式依然成立。

另外设 \max^k 表示第 k 大的元素, 可以推广为如下形式:

$$\max_{i \in S}^k \{x_i\} = \sum_{T \subseteq S} (-1)^{|T|-k} \binom{|T|-1}{k-1} \min_{j \in T} \{x_j\}$$

此外在数论上可以得到:

$$\text{lcm}_{i \in S} \{x_i\} = \prod_{T \subseteq S} \left(\gcd_{j \in T} \{x_j\} \right)^{(-1)^{|T|-1}}$$

6.10.2 应用

对于计算 “ n 个属性都出现的期望时间” 问题, 设第 i 个属性第一次出现的时间是 t_i , 所求即为 $\max(t_i)$, 使用 min-max 容斥转为计算 $\min(t_i)$ 。

比如 n 个独立物品, 每次抽中物品 i 的概率是 p_i , 问期望抽多少次抽中所有物品。那么就可以计算 \min_S 表示第一次抽中物品集合 S 内物品的时间, 可以得到:

$$\max_U = \sum_{S \subseteq U} (-1)^{|S|-1} \min_S = \sum_{S \subseteq U} (-1)^{|S|-1} \cdot \frac{1}{\sum_{x \in S} p_x}$$

6.11 Barrett 取模

6.11.1 用法

调用 init 计算出 S 和 X , 得到计算 $\lfloor x/P \rfloor = (x \times X)/2^{60+S}$ 。从而计算 $x \bmod P = x - P \times \lfloor x/P \rfloor$ 。

```

1  #include "../header.cpp"
2  i64 S = 0, X = 0;
3  void init(int MOD){
4      while(((1 << (S + 1)) < MOD) S ++;
5      X = (((__int128)1 << 60 + S) / MOD + !(((__int128)1 << 60 + S) % MOD));
6      cerr << S << " " << X << endl;
7  }
8  int power(i64 x, int y, int MOD){
9      i64 r = 1;
10     while(y){
11         if(y & 1){
12             r = r * x;
13             r = r - MOD * ((__int128)r * X >> 60 + S);
14         }
15         x = x * x;
16         x = x - MOD * ((__int128)x * X >> 60 + S);
17         y >>= 1;
18     }
19     return r;
20 }

```

6.12 Pollard's Rho

6.12.1 用法

- 调用 test(n) 判断 n 是否是质数;
- 调用 rho(n) 计算 n 分解质因数后的结果, 不保证结果有序。

```

1  #include "../header.cpp"
2  i64 step(i64 a, i64 c, i64 m){
3      return (((__int128)a * a + c) % m);
4  }
5  i64 multi(i64 a, i64 b, i64 m){
6      return ((__int128) a * b % m);
7  }
8  i64 power(i64 a, i64 b, i64 m){
9      i64 r = 1;
10     while(b){
11         if(b & 1) r = multi(r, a, m);
12         b >>= 1, a = multi(a, a, m);
13     }

```

```

14     return r;
15 }
16 mt19937_64 MT;
17 bool test(i64 n){
18     if(n < 3 || n % 2 == 0) return n == 2;
19     i64 u = n - 1, t = 0;
20     while(u % 2 == 0) u /= 2, t += 1;
21     int test_time = 20;
22     for(int i = 1; i <= test_time; ++ i){
23         i64 a = MT() % (n - 2) + 2;
24         i64 v = power(a, u, n);
25         if(v == 1) continue;
26         int s;
27         for(s = 0; s < t; ++ s){
28             if(v == n - 1) break;
29             v = multi(v, v, n);
30         }
31         if(s == t) return false;
32     }
33     return true;
34 }
35 basic_string<i64> rho(i64 n){
36     if(n == 1) return { };
37     if(test(n)) return {n};
38     i64 a = MT() % (n - 1) + 1;
39     i64 x1 = MT() % (n - 1), x2 = x1;
40     for(int i = 1; i <= 1){
41         i64 tot = 1;
42         for(int j = 1; j <= i; ++ j){
43             x2 = step(x2, a, n);
44             tot = multi(tot, llabs(x1 - x2), n);
45             if(j % 127 == 0){
46                 i64 d = __gcd(tot, n);
47                 if(d > 1)
48                     return rho(d) + rho(n / d);
49             }
50         }
51         i64 d = __gcd(tot, n);
52         if(d > 1)
53             return rho(d) + rho(n / d);
54         x1 = x2;
55     }
56 }

```

6.13 polya 定理

6.13.1 Burnside 引理

记所有染色方案的集合为 X , 其中单个染色方案为 x 。一种对称操作 $g \in X$ 作用于染色方案 $x \in X$ 上可以得到另外一种染色 x' 。

将所有对称操作作为集合 G , 那么 $Gx = \{gx \mid g \in G\}$ 是与 x 本质相同的染色方案的集合, 形式化地称为 x 的轨道。统计本质不同染色方案数, 就是统计不同轨道个数。

Burnside 引理说明如下:

$$|X/G| = \frac{1}{|G|} \sum_{g \in G} |X^g|$$

其中 X^g 表示在 $g \in G$ 的作用下, 不动点的集合。不动点被定义为 $x = gx$ 的 x 。

6.13.2 Polya 定理

对于通常的染色问题, X 可以看作一个长度为 n 的序列, 每个元素是 1 到 m 的整数。可以将 n 看作面数、 m 看作颜色数。Polya 定理叙述如下:

$$|X/G| = \frac{1}{|G|} \sum_{g \in G} \sum_{m \in G} m^{c(g)}$$

其中 $c(g)$ 表示对一个序列做轮换操作 g 可以分解成多少个置换环。

然而, 增加了限制 (比如要求某种颜色必须要多少个), 就无法直接应用 Polya 定理, 需要利用 Burnside 引理进行具体问题具体分析。

6.13.3 应用

给定 n 个点 n 条边的环, 现在有 n 种颜色, 给每个顶点染色, 询问有多少种本质不同的染色方案。

显然 X 是全体元素在 1 到 n 之间长度为 n 的序列, G 是所有可能的单次旋转方案, 共有 n 种, 第 i 种方案会把 1 置换到 i 。于是:

$$\begin{aligned} \text{ans} &= \frac{1}{|G|} \sum_{i=1}^n m^{c(g_i)} \\ &= \frac{1}{n} \sum_{i=1}^n n^{\gcd(i, n)} \\ &= \frac{1}{n} \sum_{d|n} n^d \sum_{i=1}^n [\gcd(i, n) = d] \\ &= \frac{1}{n} \sum_{d|n} n^d \varphi(n/d) \end{aligned}$$

```

1 #include "../header.cpp"
2 vector<tuple<int, int>> P;
3 void solve(int step, int n, int d, int f, int
  &ans){
4     if(step == P.size()){
5         ans = (ans + 1ll * power(n, n / d) * f) %
          MOD;
6     } else {
7         auto [w, c] = P[step];
8         int dd = 1, ff = 1;
9         for(int i = 0; i <= c; ++i){
10             solve(step + 1, n, d * dd, f * ff, ans);
11             ff = ff * (w - (i == 0));
12             dd = dd * w;
13         }
14     }
15 }
16 int main(){
17     int T; cin >> T;
18     while(T--){
19         int n, t;
20         cin >> n;
21         t = n;
22         for(int i = 2; i * i <= n; ++i) if(n % i ==
          0){
23             int w = i, c = 0;
24             while(t % i == 0){
25                 t /= i, c++;
26             }
27             P.push_back({ w, c });
28         }
29         if(t != 1){
30             P.push_back({ t, 1 });
31         }
32         int ans = 0;
33         solve(0, n, 1, 1, ans);
34         ans = 1ll * ans * power(n, MOD - 2) % MOD;
35         cout << ans << endl;
36         P.clear();
37     }
38     return 0;
39 }

```

6.14 min25 筛

设有一个积性函数 $f(n)$, 满足 $f(p^k)$ 可以快速求, 考虑搞一个在质数位置和 $f(n)$ 相等的 $g(n)$, 满足它有完全积性, 并且单点和前缀和都可以快速求, 然后通过第一部分筛出 g 在质数位置的前缀和, 从而相当于得到 f 在质数

位置的前缀和, 然后利用它, 做第二部分, 求出 f 的前缀和。

第一部分: $G_k(n) = \sum_{i=1}^n [\text{mindiv}(i) > p_k \text{ or isprime}(i)] g(i)$ ($p_0 = 1$), 则有 $G_k(n) = G_{k-1}(n) - g(p_k)(G_{k-1}(n/p_k) - G_{k-1}(p_{k-1}))$, 复杂度 $O(n^{3/4}/\log n)$ 。

第二部分: $F_k(n) = \sum_{i=1}^n [\text{mindiv}(i) \geq p_k] f(i)$, $F_k(n) = \sum_{\substack{h \geq k \\ p_h^2 \leq n}} \sum_{\substack{c \geq 1 \\ p_h^{c+1} \leq n}} (f(p_h^c) F_{h+1}(n/p_h^c) + f(p_h^{c+1})) + F_{\text{prime}}(n) - F_{\text{prime}}(p_{k-1})$, 在 $n \leq 10^{13}$ 可以证明复杂度 $O(n^{3/4}/\log n)$ 。

常见细节问题:

- 由于 n 通常是 10^{10} 到 10^{11} 的数, 导致 n 会爆 int, n^2 会爆 long long, 而且往往会用自然数幂和, 更容易爆, 所以要小心。
- 记 $s = \lfloor \sqrt{n} \rfloor$, 由于 F 递归时会去找 F_{h+1} , 会访问到 s 以内最大的质数往后的一个质数, 而已经证明对于所有 $n \in \mathbb{N}^+$, $[n+1, 2n]$ 中有至少一个质数, 所以只需要筛到 $2s$ 即可。
- 注意补回 $f(1)$ 。

```

1 // 预处理, $1$ 所在的块也算进去了
2 namespace init {
3     ll init_n, sqrt_n;
4     vector<ll> np, p, id1, id2, val;
5     ll cnt;
6     void main(ll n) {
7         init_n = n, sqrt_n = sqrt(n);
8         ll M = sqrt_n * 2; // 筛出一个 > floor
          (sqrt(n)) 的质数, 避免后续讨论边界
9         np.resize(M + 1), p.resize(M + 1);
10        for (ll i = 2; i <= M; ++i) {
11            if (!np[i]) p[++p[0]] = i;
12            for (ll j = 1; j <= p[0]; ++j) {
13                if (i * p[j] > M) break;
14                np[i * p[j]] = 1;
15                if (i % p[j] == 0) break;
16            }
17        }
18        p[0] = 1;
19        id1.resize(sqrt_n + 1), id2.resize(
          sqrt_n + 1);
20        val.resize(1);
21        for (ll l = 1, r, v; l <= n; l = r +
          1) {

```



```

22     v = n / l, r = n / v;
23     if (v ≤ sqrt_n) id1[v] = ++cnt;
24     else id2[init_n / v] = ++cnt;
25     val.emplace_back(v);
26 }
27 }
28 ll id(ll n) {
29     if (n ≤ sqrt_n) return id1[n];
30     else return id2[init_n / n];
31 }
32 }
33 using namespace init;
34 // 计算 $G_k$, 两个参数分别是 $g$ 从 $2$ 开始
   的前缀和和 $g$
35 auto calcG = [&] (auto&& sum, auto&& g) →
   vector<ll> {
36     vector<ll> G(cnt + 1);
37     for (int i = 1; i ≤ cnt; ++i) G[i] = sum(
   val[i]);
38     ll pre = 0;
39     for (int i = 1; p[i] * p[i] ≤ n; ++i) {
40         for (int j = 1; j ≤ cnt; ++j) {
41             if (p[i] * p[i] > val[j]) break;
42             ll tmp = id(val[j] / p[i]);
43             G[j] = (G[j] - g(p[i]) * (G[tmp] -
   pre)) % MD;
44         }
45         pre = (pre + g(p[i])) % MD;
46     }
47     for (int i = 1; i ≤ cnt; ++i) G[i] = (G[i]
   % MD + MD) % MD;
48     return G;
49 };
50 // 计算 $F_k$, 直接搜, 不用记忆化。`fp` 是 $F_{\text{prime}}$, `pc` 是 $p^c$, 其中 `f(p[h]^c)` 要替换掉。
51 function<ll(ll, int)> calcF = [&] (ll m, int k) {
52     if (p[k] > m) return 0;
53     ll ans = (fp[id(m)] - fp[id(p[k - 1])]) %
   MD;
54     for (int h = k; p[h] * p[h] ≤ m; ++h) {
55         ll pc = p[h], c = 1;
56         while (pc * p[h] ≤ m) {
57             ans = (ans + calcF(m / pc, h + 1)
   * f(p[h] ^ c)) % MD;
58             ++c, pc = pc * p[h], ans = (ans +
   f(p[h] ^ c)) % MD;
59         }
60     }
61     return ans;
62 };

```

6.15 杜教筛

6.15.1 用法

对于积性函数 f , 找到易求前缀和的积性函数 g, h 使得 $h = f * g$, 根据递推式计算 $S(n) = \sum_{i=1}^n f(i)$:

$$S(n) = H(n) - \sum_{d=1}^n g(d) \times S\left(\left\lfloor \frac{n}{d} \right\rfloor\right)$$

6.15.2 例题

- 对于 $f = \varphi$, 寻找 $g = 1, h = \text{id}$;
- 对于 $f = \mu$, 寻找 $g = 1, h = \varepsilon$ 。

```

1  #include "../header.cpp"
2  const int H = 1e7;
3  int P[MAXN], p; bool V[MAXN];
4  i64 ph[MAXN], sph[MAXN];
5  i64 mu[MAXN], smu[MAXN];
6  i64 tp[MAXN];
7  i64 solve_ph(i64 N){
8      for(int d = N / H; d ≥ 1; -- d){
9          i64 n = N / d;
10         i64 wh = 1ll * n * (n + 1) / 2;
11         tp[d] = wh;
12         for(i64 l = 2, r; l ≤ n; l = r + 1){
13             r = n / (n / l);
14             i64 wg = r - l + 1;
15             i64 ws = n / l ≤ H ? sph[n / l] : tp[N
   / (n / l)];
16             tp[d] -= wg * ws;
17         }
18     }
19     return N ≤ H ? sph[N] : tp[1];
20 }
21 i64 solve_mu(i64 N){
22     for(int d = N / H; d ≥ 1; -- d){
23         i64 n = N / d;
24         i64 wh = 1;
25         tp[d] = wh;
26         for(i64 l = 2, r; l ≤ n; l = r + 1){
27             r = n / (n / l);
28             i64 wg = r - l + 1;
29             i64 ws = n / l ≤ H ? smu[n / l] : tp[N
   / (n / l)];
30             tp[d] -= wg * ws;
31         }
32     }
33     return N ≤ H ? smu[N] : tp[1];
34 }
35 int main(){

```

```

36     ios :: sync_with_stdio(false);
37     cin.tie(nullptr);
38     ph[1] = 1;
39     mu[1] = 1;
40     for(int i = 2; i ≤ H; ++ i){
41         if(!V[i]){
42             P[++ p] = i;
43             ph[i] = i - 1;
44             mu[i] = -1;
45         }
46         for(int j = 1; j ≤ p && P[j] ≤ H / i; ++ j)
   {
47             int &p = P[j];
48             V[i * p] = true;
49             if(i % p == 0){
50                 ph[i * p] = ph[i] * p;
51                 mu[i * p] = 0;
52                 break;
53             } else {
54                 ph[i * p] = ph[i] * (p - 1);
55                 mu[i * p] = -mu[i];
56             }
57         }
58     }
59     for(int i = 1; i ≤ H; ++ i){
60         sph[i] = sph[i - 1] + ph[i];
61         smu[i] = smu[i - 1] + mu[i];
62     }
63     int T; cin >> T;
64     while(T → 0){
65         int n; cin >> n;
66         cout << solve_ph(n) << " " << solve_mu(n)
   << "\n";
67     }
68     return 0;
69 }

```

6.16 PN 筛

6.16.1 用法

对于积性函数 $f(x)$, 寻找积性函数 $g(x)$ 使得 $g(p) = f(p)$, 且 g 易求前缀和 G 。

令 $h = f * g^{-1}$, 可以证明只有 PN 处 h 的函数值非 0, PN 指每个素因子幂次都不小于 2 的数。同时可以证明 n 以内的 PN 只有 $O(\sqrt{n})$ 个, 且可以暴力枚举质因子幂次得到所有 PN。

可利用下面公式计算 $h(p^c)$:

$$h(p^c) = f(p^c) - \sum_{i=1}^c g(p^i) \times h(p^{c-i})$$

6.16.2 例题

定义积性函数 $f(x)$ 满足 $f(p^k) = p^k(p^k - 1)$,

计算 $\sum f(i)$ 。

取 $g(p) = \text{id}(p)\varphi(p) = f(p)$, 根据 $g * \text{id} = \text{id}_2$ 利用杜教筛求解。 $h(p^c)$ 的值利用递推式进行计算。

```

37 for(i64 i = last + 1; x ≤ N / P[i] / P[i]; ++
38 i){
39     int c = 2;
40     for(i64 t = x * P[i] * P[i]; t ≤ N; t *= P[
41         i], c++){
42         int hh = 1ll * h * hc[i][c] % MOD;
43         sieve_pn(i, t, hh, N);
44     }
45 }
46 int main(){
47     ios :: sync_with_stdio(false);
48     cin.tie(nullptr);
49     g[1] = 1;
50     for(int i = 2; i ≤ H; ++ i){
51         if(!V[i]){
52             P[++ p] = i, g[i] = 1ll * i * (i - 1) %
53             MOD;
54         }
55         for(int j = 1; j ≤ p && P[j] ≤ H / i; ++ j)
56         {
57             int &p = P[j];
58             V[i * p] = true;
59             if(i % p == 0){
60                 g[i * p] = 1ll * g[i] * p % MOD * p %
61                 MOD;
62                 break;
63             } else {
64                 g[i * p] = 1ll * g[i] * p % MOD * (p -
65                 1) % MOD;
66             }
67         }
68     }
69     for(int i = 1; i ≤ H; ++ i){
70         le[i] = (le[i - 1] + g[i]) % MOD;
71     }
72     i64 N;
73     cin >> N;
74     for(int i = 1; i ≤ p && 1ll * P[i] * P[i] ≤
75         N; i++){
76         int &p = P[i];
77         hc[i].push_back(1);
78         gc[i].push_back(1);
79         for(i64 c = 1, t = p; t ≤ N; t = t * p, ++
80             c){
81             if(c == 1){
82                 gc[i].push_back(1ll * p * (p - 1) %
83                 MOD);
84             } else {
85                 gc[i].push_back(1ll * gc[i].back() * p
86                 % MOD * p % MOD);
87             }
88         }
89     }
90 }

```

```

79     int w = 1ll * (t % MOD) * ((t - 1) % MOD
80     ) % MOD;
81     int s = 0;
82     for(int j = 1; j ≤ c; ++ j){
83         s = (s + 1ll * gc[i][j] * hc[i][c - j
84         ]) % MOD;
85     }
86     hc[i].push_back((w - s + MOD) % MOD);
87 }
88 sieve_du(N);
89 sieve_pn(0, 1, 1, N);
90 cout << ANS << "\n";
91 return 0;
92 }

```

6.17 常用数表

6.17.1 大质数

10^{18} 级别:

- $P = 10^{18} + 3$, 好记。
- $P = 2924438830427668481$, 可以进行 NTT, $P = 174310137655 \times 2^24 + 1$, 原根为 3。

6.18 二次剩余

6.18.1 用法

多次询问, 每次询问给定奇素数 p 以及 y , 在 $\mathcal{O}(\log p)$ 复杂度计算 x 使得 $x^2 \equiv 0 \pmod{p}$ 或者无解。

```

1 #include "../header.cpp"
2 bool check(int x, int p){
3     return power(x, (p - 1) / 2, p) == 1;
4 }
5 struct Node {
6     int real, imag;
7 };
8 Node mul(const Node a, const Node b, int p,
9     int v){
10     int nreal = (1ll * a.real * b.real + 1ll * a
11     .imag * b.imag % p * v) % p;
12     int nimag = (1ll * a.real * b.imag + 1ll * a
13     .imag * b.real) % p;
14     return { nreal, nimag };
15 }
16 Node power(Node a, int b, int p, int v){
17     Node r = { 1, 0 };
18     while(b){

```

```

1 #include "../header.cpp"
2 const int H = 1e7;
3 const int MOD = 1e9 + 7;
4 const int DIV2 = 500000004;
5 const int DIV6 = 166666668;
6 int P[MAXN], p; bool V[MAXN];
7 int g[MAXN], le[MAXN], ge[MAXN];
8 int s1(i64 n){ // 1^1 + 2^1 + ... + n^1
9     n %= MOD;
10     return 1ll * n * (n + 1) % MOD * DIV2 % MOD;
11 }
12 int s2(i64 n){ // 1^2 + 2^2 + ... + n^2
13     n %= MOD;
14     return 1ll * n * (n + 1) % MOD * (2 * n + 1)
15     % MOD * DIV6 % MOD;
16 }
17 int sg(i64 n, i64 N){
18     return n ≤ H ? le[n] : ge[N / n];
19 }
20 int sieve_du(i64 N){
21     for(int d = N / H; d ≥ 1; -- d){
22         i64 n = N / d;
23         int wh = s2(n);
24         for(i64 l = 2, r; l ≤ n; l = r + 1){
25             r = n / (n / l);
26             int wg = (s1(r) - s1(l - 1) + MOD) % MOD
27             ;
28             int ws = sg(n / l, N);
29             ge[d] = (ge[d] + 1ll * wg * ws) % MOD;
30         }
31         ge[d] = (wh - ge[d] + MOD) % MOD;
32     }
33     return N ≤ H ? le[N] : ge[1];
34 }
35 vector<int> hc[MAXM], gc[MAXM];
36 int ANS;
37 void sieve_pn(int last, i64 x, int h, i64 N){
38     ANS = (ANS + 1ll * h * sg(N / x, N)) % MOD;

```

```

16     if(b & 1) r = mul(r, a, p, v);
17     b >>= 1, a = mul(a, a, p, v);
18 }
19 return r;
20 }
21 mt19937 MT;
22 void solve(int n, int p, int &x1, int &x2){
23     if(n == 0){
24         x1 = x2 = 0;
25         return;
26     }
27     if(!check(n, p)){
28         x1 = x2 = -1;
29         return;
30     }
31     int a, t;
32     do {
33         a = MT() % p;
34     } while(check(t = (1ll * a * a - n + p) % p,
35         p));
36     Node u = { a, 1 };
37     x1 = power(u, (p + 1) / 2, p, t).real;
38     x2 = (p - x1) % p;
39     if(x1 > x2) swap(x1, x2);
40 }
41 int main(){
42     ios :: sync_with_stdio(false);
43     cin.tie(nullptr);
44     int T; cin >> T;
45     while(T --){
46         int n, p, x1, x2;
47         cin >> n >> p;
48         solve(n, p, x1, x2);
49         if(x1 == -1){
50             cout << "Hola!\n";
51         } else {
52             if(x1 == x2){
53                 cout << x1 << "\n";
54             } else {
55                 cout << x1 << " " << x2 << "\n";
56             }
57         }
58     }
59     return 0;
60 }

```

6.19 单位根反演

6.19.1 定理

给出单位根反演如下：

$$[d \mid n] = \frac{1}{d} \sum_{i=0}^{d-1} \omega_d^{ni}$$

7 多项式

7.1 NTT 全家桶

7.1.1 用法

多项式全家桶。

- 包含基础多项式算法：快速傅里叶变换（FFT）及其逆变换（IFFT）、快速数论变换（NTT）及其逆变换（INTT）；
- 包含基于 NTT 的扩展多项式算法：多项式乘法（MUL）、多项式乘法逆元（INV）、多项式微分（DIF）、多项式积分（INT）、多项式对数（LN）、多项式指数（EXP）、多项式开根（SQT）、多项式平移（即计算 $G(x) = F(x + c)$ ，SHF）。

```

1 #include "../header.cpp"
2 int inv(int x);
3 const int MAX_ = (1 << 19) + 3;
4 using cplx = complex<double>;
5 const long double pi = acos(-1);
6 namespace Poly{
7     void FFT(int n, cplx Z[]){
8         static int W[MAX_];
9         int l = 1; W[0] = 0;
10        while (n >= 1)
11            up(0, l - 1, i)
12                W[l++] = W[i] << 1 | 1, W[i] <= 1;
13        up(0, l - 1, i)
14            if(W[i] > i) swap(Z[i], Z[W[i]]);
15        for (n = l >> 1, l = 1; n >= 1, l <= 1)
16            {
17                cplx* S = Z, o(cos(pi / l), sin(pi / l))
18                ;
19                up(0, n - 1, i){
20                    cplx s(1, 0);
21                    up(0, l - 1, j){
22                        cplx x = S[j] + s * S[j + l];

```

```

23                cplx y = S[j] - s * S[j + l];
24                S[j] = x, S[j + l] = y, s = s * o;
25            }
26        }
27    }
28    void IFFT(int n, cplx Z[]){
29        FFT(n, Z); reverse(Z + 1, Z + n);
30        up(0, n - 1, i) Z[i] /= n;
31    }
32    void NTT(int n, int Z[]){
33        static int W[MAX_];
34        int g = 3, l = 1; W[0] = 0;
35        while (n >= 1)
36            up(0, l - 1, i)
37                W[l++] = W[i] << 1 | 1, W[i] <= 1;
38        up(0, l - 1, i)
39            if (W[i] > i) swap(Z[i], Z[W[i]]);
40        for (n = l >> 1, l = 1; n >= 1, l <= 1)
41            {
42                int* S = Z, o = power(g, (MOD - 1) / l / 2);
43                up(0, n - 1, i){
44                    int s = 1;
45                    up(0, l - 1, j){
46                        int x = (S[j] + 1ll * s * S[j + l] % MOD) % MOD;
47                        int y = (S[j] - 1ll * s * S[j + l] % MOD + MOD) % MOD;
48                        S[j] = x, S[j + l] = y;
49                        s = 1ll * s * o % MOD;
50                    }
51                }
52            }
53    }
54    void INTT(int n, int Z[]){
55        NTT(n, Z); reverse(Z + 1, Z + n);
56        int o = inv(n);
57        up(0, n - 1, i)
58            Z[i] = 1ll * Z[i] * o % MOD;
59    }
60    void MUL(int n, int A[], int B[]) { // 乘法
61        NTT(n, A), NTT(n, B);
62        up(0, n - 1, i)
63            A[i] = 1ll * A[i] * B[i] % MOD;
64        INTT(n, A);
65    }
66    void INV(int n, int Z[], int T[]) { // 乘法逆

```

```

67 static int A[MAX_];
68 up(0, n - 1, i)
69 T[i] = 0;
70 T[0] = power(Z[0], MOD - 2);
71 for (int l = 1; l < n; l <= 1){
72 up(0, 2 * l - 1, i) A[i] = Z[i];
73 up(2 * l, 4 * l - 1, i) A[i] = 0;
74 NTT(4 * l, A), NTT(4 * l, T);
75 up(0, 4 * l - 1, i)
76 T[i] = (2ll * T[i] - 1ll * A[i] * T[i]
77 % MOD * T[i] % MOD + MOD) % MOD;
78 INTT(4 * l, T);
79 up(2 * l, 4 * l - 1, i)
80 T[i] = 0;
81 }
82 void DIF(int n, int Z[], int T[]) { // 微分
83 up(0, n - 2, i)
84 T[i] = 1ll * Z[i + 1] * (i + 1) % MOD;
85 T[n - 1] = 0;
86 }
87 void INT(int n, int c, int Z[], int T[]) { // 积分
88 up(1, n - 1, i)
89 T[i] = 1ll * Z[i - 1] * inv(i) % MOD;
90 T[0] = c;
91 }
92 void LN(int n, int* Z, int* T) { // 求对数
93 static int A[MAX_], B[MAX_];
94 up(0, 2 * n - 1, i)
95 A[i] = B[i] = 0;
96 DIF(n, Z, A), INV(n, Z, B), MUL(2 * n, A, B), INT(n, 0, A, T);
97 }
98 void EXP(int n, int* Z, int* T) { // 求指数
99 static int A[MAX_], B[MAX_];
100 up(1, 2 * n - 1, i) T[i] = 0;
101 T[0] = 1;
102 for (int l = 1; l < n; l <= 1){
103 LN(2 * l, T, A);
104 up(0, 2 * l - 1, i)
105 B[i] = (-A[i] + Z[i] + MOD) % MOD;
106 B[0] = (B[0] + 1) % MOD;
107 up(2 * l, 4 * l - 1, i)
108 T[i] = B[i] = 0;
109 MUL(4 * l, T, B);
110 }
111 }
112 void SQT(int n, int* Z, int* T) { // 开

```

```

113 根
114 static int A[MAX_], B[MAX_];
115 up(1, 2 * n - 1, i) T[i] = 0;
116 T[0] = 1;
117 int o = inv(2);
118 for (int l = 1; l < n; l <= 1){
119 INV(2 * l, T, A);
120 up(0, 2 * l - 1, i)
121 B[i] = Z[i];
122 up(2 * l, 4 * l - 1, i)
123 A[i] = B[i] = 0;
124 MUL(4 * l, A, B);
125 up(0, 2 * l - 1, i)
126 T[i] = 1ll * (T[i] + A[i]) * o % MOD;
127 }
128 void SHF(int n, int c, int* Z, int* T) { // 平移
129 static int A[MAX_], B[MAX_], F[MAX_], G[
130 MAX_];
131 int o = 1;
132 up(1, n - 1, i)
133 F[i] = 1ll * F[i - 1] * i % MOD,
134 G[i] = 1ll * G[i - 1] * inv(i) % MOD;
135 up(0, n - 1, i)
136 A[i] = 1ll * Z[n - 1 - i] * F[n - 1 - i]
137 % MOD;
138 up(0, n - 1, i){
139 B[i] = 1ll * G[i] * o % MOD;
140 o = 1ll * o * c % MOD;
141 }
142 int l = 1; while (l < 2 * n - 1) l <= 1;
143 up(n, l - 1, i)
144 A[i] = B[i] = 0;
145 MUL(l, A, B);
146 up(0, n - 1, i)
147 T[n - 1 - i] = 1ll * G[n - 1 - i] * A[i]
148 % MOD;
149 }
150 }

```

7.2 FWT 全家桶

7.2.1 用法

沃尔什全家桶。

包含与卷积、或卷积、异或卷积，定义分别为二进制与、或、异或带入下式：

$$b_k = \sum_{i \otimes j = k} a_i \times b_j$$

```

1 #include "../header.cpp"
2 namespace Solve1 { // or 卷积
3 void FWT(int n, int *A) {
4 for (int l = 1 << n, u = 2, v = 1; u <= l; u
5 <= 1, v <= 1)
6 for (int j = 0; j < l; j += u)
7 for (int k = 0; k < v; ++k)
8 A[j + v + k] = (A[j + v + k] + A[j +
9 k]) % MOD;
10 }
11 void IFWT(int n, int *A) {
12 for (int l = 1 << n, u = l, v = l / 2; u >
13 1; u >= 1, v >= 1)
14 for (int j = 0; j < l; j += u)
15 for (int k = 0; k < v; ++k)
16 A[j + v + k] = (A[j + v + k] - A[j +
17 k] + MOD) % MOD;
18 }
19 }
20 namespace Solve2 { // and 卷积
21 void FWT(int n, int *A) {
22 for (int l = 1 << n, u = 2, v = 1; u <= l; u
23 <= 1, v <= 1)
24 for (int j = 0; j < l; j += u)
25 for (int k = 0; k < v; ++k)
26 A[j + k] = (A[j + k] + A[j + v + k])
27 % MOD;
28 }
29 void IFWT(int n, int *A) {
30 for (int l = 1 << n, u = l, v = l / 2; u >
31 1; u >= 1, v >= 1)
32 for (int j = 0; j < l; j += u)
33 for (int k = 0; k < v; ++k)
34 A[j + k] = (A[j + k] - A[j + v + k]
35 + MOD) % MOD;
36 }
37 }
38 namespace Solve3 { // xor 卷积
39 void FWT(int n, int *A) {
40 for (int l = 1 << n, u = 2, v = 1; u <= l; u
41 <= 1, v <= 1)
42 for (int j = 0; j < l; j += u)
43 for (int k = 0; k < v; ++k) {
44 int a = A[j + k];
45 int b = A[j + v + k];
46 A[j + k] = (a + b + MOD) % MOD;
47 A[j + v + k] = (a - b + MOD) % MOD;
48 }
49 }
50 void IFWT(int n, int *A) {
51 int div2 = (MOD + 1) / 2;
52 }

```

```

43 for(int l = 1 << n, u = l, v = l / 2; u >
    1; u >= 1, v >= 1)
44 for(int j = 0; j < l; j += u)
45 for(int k = 0; k < v; ++ k){
46     int a = A[j + k];
47     int b = A[j + v + k];
48     A[j + k] = 1ll * (a + b + MOD) *
        div2 % MOD;
49     A[j + v + k] = 1ll * (a - b + MOD) *
        div2 % MOD;
50 }
51 }
52 }

```

7.3 任意模数 NTT

```

1 #include "poly-family.cpp"
2 const int BLOCK = 32768;
3 using cplx = complex<double>;
4 cplx A1[MAXN], A2[MAXN], B1[MAXN], B2[MAXN];
5 int n, m, L, mod;
6 cplx P[MAXN], Q[MAXN];
7 void FFTFFT(int L, cplx X[], cplx Y[]){
8     for(int i = 0; i < L; ++ i){
9         P[i] = { X[i].real(), Y[i].imag() };
10    }
11    Poly :: FFT(L, P);
12    for(int i = 0; i < L; ++ i){
13        Q[i] = (i == 0 ? P[0] : P[L - i]);
14        Q[i].imag(-Q[i].imag());
15    }
16    for(int i = 0; i < L; ++ i){
17        X[i] = (P[i] + Q[i]);
18        Y[i] = (Q[i] - P[i]) * cplx(0, 1);
19        X[i] /= 2, Y[i] /= 2;
20    }
21 }
22 int main(){
23     ios :: sync_with_stdio(false);
24     cin.tie(nullptr);
25     cin >> n >> m >> mod;
26     for(int i = 0; i <= n; ++ i){
27         int a; cin >> a; a %= mod;
28         A1[i].real(a / BLOCK);
29         A2[i].imag(a % BLOCK);
30     }
31     for(int i = 0; i <= m; ++ i){
32         int a; cin >> a; a %= mod;
33         B1[i].real(a / BLOCK);
34         B2[i].imag(a % BLOCK);
35     }
36     for(L = 1; L <= n + m; L <= 1);

```

```

37 FFTFFT(L, A1, A2), FFTFFT(L, B1, B2);
38 for(int i = 0; i < L; ++ i){
39     P[i] = A1[i] * B1[i] + cplx(0, 1) * A2[i]
        * B1[i];
40     Q[i] = A1[i] * B2[i] + cplx(0, 1) * A2[i]
        * B2[i];
41 }
42 Poly :: IFFT(L, P);
43 Poly :: IFFT(L, Q);
44 for(int i = 0; i < L; ++ i){
45     long long a1b1 = P[i].real() + 0.5;
46     long long a2b1 = P[i].imag() + 0.5;
47     long long a1b2 = Q[i].real() + 0.5;
48     long long a2b2 = Q[i].imag() + 0.5;
49     long long w = ((a1b1 % mod * (BLOCK *
        BLOCK % mod)) + ((a2b1 + a1b2) % mod) *
        BLOCK + a2b2) % mod;
50     if(i <= n + m) cout << w << " ";
51 }
52 return 0;
53 }

```

8 字符串

8.1 AC 自动机

```

1 #include "../header.cpp"
2 namespace ACAM{
3     int C[MAXN][MAXM], F[MAXN], o;
4     void insert(char *S){
5         int p = 0, len = 0;
6         for(int i = 0; S[i]; ++ i){
7             int e = S[i] - 'a';
8             if(C[p][e]) p = C[p][e];
9             else p = C[p][e] = ++ o;
10            ++ len;
11        }
12    }
13    void build(){
14        queue<int> Q; Q.push(0);
15        while(!Q.empty()){
16            int u = Q.front(); Q.pop();
17            for(int i = 0; i < 26; ++ i){
18                int v = C[u][i];
19                if(v == 0) continue;
20                int p = F[u];
21                while(!C[p][i] && p != 0) p = F[p];
22                if(C[p][i] && C[p][i] != v)
23                    F[v] = C[p][i];
24                Q.push(v);
25            }

```

```

26 }
27 }
28 }

```

8.2 扩展 KMP

8.2.1 定义

$$z_i^{(1)} = |\text{lcp}(b, \text{suffix}(b, i))|$$

$$z_i^{(2)} = |\text{lcp}(b, \text{suffix}(a, i))|$$

```

1 #include "../header.cpp"
2 char A[MAXN], B[MAXN * 2];
3 int n, m, l, r, Z[MAXN * 2];
4 i64 ans1, ans2;
5 int main(){
6     scanf("%s%s", A + 1, B + 1);
7     n = strlen(A + 1);
8     m = strlen(B + 1);
9     l = 0, r = 0; Z[1] = 0, ans1 = m + 1;
10    for(int i = 2; i <= m; ++ i){
11        if(i <= r) Z[i] = min(r - i + 1, Z[i - l +
            1]);
12        else Z[i] = 0;
13        while(B[Z[i] + 1] == B[i + Z[i]])
14            ++ Z[i];
15        if(i + Z[i] - 1 > r)
16            r = i + Z[i] - 1, l = i;
17        ans1 ^= 1ll * i * (Z[i] + 1);
18    }
19    l = 0, r = 0;
20    Z[1] = 0, B[m + 1] = '#', strcat(B + 1, A +
        1);
21    for(int i = 2; i <= n + m + 1; ++ i){
22        if(i <= r) Z[i] = min(r - i + 1, Z[i - l +
            1]);
23        else Z[i] = 0;
24        while(B[Z[i] + 1] == B[i + Z[i]])
25            ++ Z[i];
26        if(i + Z[i] - 1 > r)
27            r = i + Z[i] - 1, l = i;
28    }
29    for(int i = m + 2; i <= n + m + 1; ++ i){
30        ans2 ^= 1ll * (i - m - 1) * (Z[i] + 1);
31    }
32    printf("%lld\n%lld\n", ans1, ans2);
33    return 0;
34 }

```


8.3 回文自动机

```

1 #include "../header.cpp"
2 namespace PAM{
3     const int SIZ = 5e5 + 3;
4     int n, s, F[SIZ], L[SIZ], D[SIZ];
5     int M[SIZ][MAXM];
6     char S[SIZ];
7     void init(){
8         S[0] = '$', n = 1;
9         F[s = 0] = -1, L[0] = -1, D[0] = 0;
10        F[s = 1] = 0, L[1] = 0, D[1] = 0;
11    }
12    void extend(int &last, char c){
13        S[++ n] = c;
14        int e = c - 'a', a = last;
15        while(c != S[n - 1 - L[a]]) a = F[a];
16        if(M[a][e]){
17            last = M[a][e];
18        } else {
19            int cur = M[a][e] = ++ s;
20            L[cur] = L[a] + 2;
21            if(a == 0){
22                F[cur] = 1;
23            } else {
24                int b = F[a];
25                while(c != S[n - 1 - L[b]])
26                    b = F[b];
27                F[cur] = M[b][e];
28            }
29            D[cur] = D[F[cur]] + 1;
30            last = cur;
31        }
32    }
33 }

```

8.4 后缀数组 (倍增)

```

1 #include "../header.cpp"
2 int n, m, A[MAXN], B[MAXN];
3 int C[MAXN], R[MAXN], P[MAXN], Q[MAXN];
4 char S[MAXN];
5 int main(){
6     scanf("%s", S), n = strlen(S), m = 256;
7     for(int i = 0; i < n; ++ i) R[i] = S[i];
8     for (int k = 1; k ≤ n; k <= 1){
9         for(int i = 0; i < n; ++ i){
10            Q[i] = ((i + k > n - 1) ? 0 : R[i + k]);
11            P[i] = R[i];
12            m = max(m, R[i]);
13        }
14    }
15    #define fun(a, b, c) \

```

```

15    memset(C, 0, sizeof(int) * (m + 1));
16    for(int i = 0; i < n; ++ i) C[a] += 1;
17    for(int i = 1; i ≤ m; ++ i) C[i] += C[i - 1];
18    for(int i = n - 1; i ≥ 0; -- i) c[-- C[a]]
19        = b;
20    fun(Q[i], i, B)
21    fun(P[B[i]], B[i], A)
22    #undef fun
23    int p = 1; R[A[0]] = 1;
24    for(int i = 1; i ≤ n - 1; ++ i){
25        bool f1 = P[A[i]] == P[A[i - 1]];
26        bool f2 = Q[A[i]] == Q[A[i - 1]];
27        R[A[i]] = f1 && f2 ? R[A[i - 1]] : ++ p;
28    }
29    if (m == n) break;
30    for(int i = 0; i < n; ++ i)
31        printf("%u ", A[i] + 1);
32    return 0;
33 }

```

8.5 广义后缀自动机 (离线)

```

1 #include "../header.cpp"
2 namespace SAM{
3     const int SIZ = 2e6 + 3;
4     int M[SIZ][MAXM];
5     int L[SIZ], F[SIZ], S[SIZ];
6     int s = 0, h = 25;
7     void init(){
8         F[0] = -1, s = 0;
9     }
10    void extend(int &last, char c){
11        int e = c - 'a';
12        int cur = ++ s;
13        L[cur] = L[last] + 1;
14        int p = last;
15        while(p != -1 && !M[p][e])
16            M[p][e] = cur, p = F[p];
17        if(p == -1){
18            F[cur] = 0;
19        } else {
20            int q = M[p][e];
21            if(L[p] + 1 == L[q]){
22                F[cur] = q;
23            } else {
24                int clone = ++ s;
25                L[clone] = L[p] + 1;
26                F[clone] = F[q];

```

```

27        for(int i = 0; i ≤ h; ++ i)
28            M[clone][i] = M[q][i];
29        while(p != -1 && M[p][e] == q)
30            M[p][e] = clone, p = F[p];
31        F[cur] = F[q] = clone;
32    }
33    last = cur;
34 }
35 void solve(){
36     i64 ans = 0;
37     for(int i = 1; i ≤ s; ++ i)
38         ans += L[i] - L[F[i]];
39     cout << ans << endl;
40 }
41 }
42 namespace Trie{
43     const int SIZ = 1e6 + 3;
44     int M[SIZ][MAXM], s, h = 25;
45     void insert(char *S){
46         int p = 0;
47         for(int i = 0; S[i]; ++ i){
48             int e = S[i] - 'a';
49             if(M[p][e]){
50                 p = M[p][e];
51             } else
52                 p = M[p][e] = ++ s;
53         }
54     }
55     int O[SIZ];
56     void build_sam(){
57         queue<int> Q;
58         Q.push(0);
59         while(!Q.empty()){
60             int u = Q.front(); Q.pop();
61             for(int i = 0; i ≤ h; ++ i){
62                 char c = i + 'a';
63                 if(M[u][i]){
64                     int v = M[u][i];
65                     O[v] = O[u];
66                     SAM :: extend(O[v], c);
67                     Q.push(v);
68                 }
69             }
70         }
71     }
72 }
73 }

```

8.6 广义后缀自动机 (在线)

```

1 #include "../header.cpp"
2 namespace SAM{

```

```

3  const int SIZ = 2e6 + 3;
4  int M[SIZ][MAXM];
5  int L[SIZ], F[SIZ], S[SIZ];
6  int s = 0, h = 25;
7  void init(){
8      F[0] = -1, s = 0;
9  }
10 void extend(int &last, char c){
11     int e = c - 'a';
12     if(M[last][e]){
13         int p = last;
14         int q = M[last][e];
15         if(L[q] == L[last] + 1){
16             last = q;
17         } else {
18             int clone = ++s;
19             L[clone] = L[p] + 1;
20             F[clone] = F[q];
21             for(int i = 0; i ≤ h; ++i)
22                 M[clone][i] = M[q][i];
23             while(p ≠ -1 && M[p][e] == q)
24                 M[p][e] = clone, p = F[p];
25             F[q] = clone;
26             last = clone;
27         }
28     } else {
29         int cur = ++s;
30         L[cur] = L[last] + 1;
31         int p = last;
32         while(p ≠ -1 && !M[p][e])
33             M[p][e] = cur, p = F[p];
34         if(p == -1){
35             F[cur] = 0;
36         } else {
37             int q = M[p][e];
38             if(L[p] + 1 == L[q]){
39                 F[cur] = q;
40             } else {
41                 int clone = ++s;
42                 L[clone] = L[p] + 1;
43                 F[clone] = F[q];
44                 S[clone] = 0;
45                 for(int i = 0; i ≤ h; ++i)
46                     M[clone][i] = M[q][i];
47                 while(p ≠ -1 && M[p][e] == q)
48                     M[p][e] = clone, p = F[p];
49                 F[cur] = F[q] = clone;
50             }
51         }
52     }
53 }
54 void solve(){
55     i64 ans = 0;

```

```

56     for(int i = 1; i ≤ s; ++i)
57         ans += L[i] - L[F[i]];
58     cout << ans << endl;
59 }
60 }
61 // 每次插入新字符串前将 last 清零

```

8.7 后缀自动机

```

1  #include "../header.cpp"
2  namespace SAM{
3      const int SIZ = 2e6 + 3;
4      int M[SIZ][MAXM];
5      int L[SIZ], F[SIZ], S[SIZ];
6      int last = 0, s = 0, h = 25;
7      void init(){
8          F[0] = -1, last = s = 0;
9      }
10     void extend(char c){
11         int cur = ++s, e = c - 'a';
12         L[cur] = L[last] + 1;
13         S[cur] = 1;
14         int p = last;
15         while(p ≠ -1 && !M[p][e])
16             M[p][e] = cur, p = F[p];
17         if(p == -1){
18             F[cur] = 0;
19         } else {
20             int q = M[p][e];
21             if(L[p] + 1 == L[q]){
22                 F[cur] = q;
23             } else {
24                 int clone = ++s;
25                 L[clone] = L[p] + 1;
26                 F[clone] = F[q];
27                 S[clone] = 0;
28                 for(int i = 0; i ≤ h; ++i)
29                     M[clone][i] = M[q][i];
30                 while(p ≠ -1 && M[p][e] == q)
31                     M[p][e] = clone, p = F[p];
32                 F[cur] = F[q] = clone;
33             }
34         }
35         last = cur;
36     }
37     vector <int> E[SIZ];
38     void build(){
39         for(int i = 1; i ≤ s; ++i){
40             E[F[i]].push_back(i);
41         }
42     }
43     i64 ans = 0;

```

```

44     void dfs(int u){
45         for(auto &v : E[u]){
46             dfs(v), S[u] += S[v];
47         }
48         if(S[u] > 1)
49             ans = max(ans, 1ll * S[u] * L[u]);
50     }
51 }

```

9 计算几何

10 其他

10.1 笛卡尔树

```

1  #include "../header.cpp"
2  // Li: 左儿子; Ri: 右儿子
3  int n, L[MAXN], R[MAXN], A[MAXN];
4  void build(){
5      stack <int> S;
6      A[n + 1] = -1e9;
7      for(int i = 1; i ≤ n + 1; ++i){
8          int v = 0;
9          while(!S.empty() && A[S.top()] > A[i]){
10             auto u = S.top();
11             R[u] = v, v = u, S.pop();
12         }
13         L[i] = v, S.push(i);
14     }
15 }

```

10.2 CDQ 分治

10.2.1 例题

给定三元组序列 (a_i, b_i, c_i) , 求解 $f(i) = \sum_j [a_j \leq a_i \wedge b_j \leq b_i \wedge c_j \leq c_i]$.

```

1  #include "../header.cpp"
2  struct Node{
3      int id, a, b, c;
4  }A[MAXN], B[MAXN];
5  bool cmp(Node a, Node b){
6      if(a.a ≠ b.a) return a.a < b.a;
7      if(a.b ≠ b.b) return a.b < b.b;
8      if(a.c ≠ b.c) return a.c < b.c;
9      return a.id < b.id;
10 }
11 int K[MAXN], H[MAXN];

```

```

12 int qread();
13 int n, m, D[MAXM];
14 namespace BIT{
15     void increase(int x, int w){
16         while(x ≤ m) D[x] += w, x += x & -x;
17     }
18     void decrease(int x, int w){
19         while(x ≤ m) D[x] -= w, x += x & -x;
20     }
21     void query(int x, int &r){
22         while(x) r += D[x], x -= x & -x;
23     }
24 }
25 void cdq(int l, int r){
26     if(l ≠ r){
27         int t = l + r >> 1; cdq(l, t), cdq(t + 1,
28             r);
29         int p = l, q = t + 1, u = l;
30         while(p ≤ t && q ≤ r){
31             if(A[p].b ≤ A[q].b)
32                 BIT :: increase(A[p].c, 1), B[u ++] =
33                 A[p ++];
34             else
35                 BIT :: query(A[q].c, K[A[q].id]), B[u
36                 ++] = A[q ++];
37         }
38         while(p ≤ t) BIT :: increase(A[p].c, 1),
39         B[u ++] = A[p ++];
40         while(q ≤ r) BIT :: query(A[q].c, K[A[q].
41         id]), B[u ++] = A[q ++];
42         up(l, t, i) BIT :: decrease(A[i].c, 1);
43         up(l, r, i) A[i] = B[i];
44     }
45 }
46 int main(){
47     n = qread(), m = qread();
48     up(1, n, i) A[i].id = i, A[i].a = qread(), A
49     [i].b = qread(), A[i].c = qread();
50     sort(A + 1, A + 1 + n, cmp), cdq(1, n);
51     sort(A + 1, A + 1 + n, cmp);
52     dn(n, 1, i){
53         if(A[i].a = A[i + 1].a && A[i].b = A[i +
54         1].b && A[i].c = A[i + 1].c)
55             K[A[i].id] = K[A[i + 1].id];
56         H[K[A[i].id]] ++;
57     }
58     up(0, n - 1, i) printf("%d\n", H[i]);
59     return 0;
60 }

```

10.3 自适应辛普森

10.3.1 例题

计算

$$\int_0^{+\infty} x^{(a/x)-x}$$

```

1 #include "../header.cpp"
2 double simpson(double (*f)(double), double l,
3     double r){
4     double mid = (l + r) / 2;
5     return (r - l) * (f(l) + 4 * f(mid) + f(r))
6         / 6.0;
7 }
8 double adapt_simpson(double (*f)(double),
9     double l, double r, double EPS, int step){
10     double mid = (l + r) / 2;
11     double w0 = simpson(f, l, r);
12     double w1 = simpson(f, l, mid);
13     double w2 = simpson(f, mid, r);
14     if(fabs(w0 - w1 - w2) < EPS && step < 0)
15         return w1 + w2;
16     else
17         return adapt_simpson(f, l, mid, EPS, step
18             - 1) +
19             adapt_simpson(f, mid, r, EPS, step
20             - 1);
21 }
22 double a, l, r;
23 double fun(double x){
24     return pow(x, a / x - x);
25 }
26 int main(){
27     cin >> a;
28     if(a < 0)
29         cout << "orz" << endl;
30     else {
31         l = 1e-9, r = 150;
32         cout << fixed << setprecision(5) <<
33         adapt_simpson(fun, l, r, 1e-9, 15);
34     }
35 }

```

10.4 模拟退火

10.4.1 例题

给定 n 个物品挂在洞下，第 i 个物品坐标 (x_i, y_i) 重量为 w_i 。询问平衡点。

```

1 #include "../header.cpp"
2 const double T0 = 2e3, Tk = 1e-14, delta =
3     0.993, R = 1e-3;

```

```

3 mt19937 MT(114514);
4 double distance(double x, double y, double a,
5     double b){
6     return sqrt(pow(a - x, 2) + pow(b - y, 2));
7 }
8 const int MAXN = 1e3 + 3;
9 double X[MAXN], Y[MAXN], W[MAXN]; int n;
10 double calculate(double x, double y){
11     double gx, gy, a;
12     for(int i = 0; i < n; ++i){
13         a = atan2(y - Y[i], x - X[i]);
14         gx += cos(a) * W[i];
15         gy += sin(a) * W[i];
16     }
17     return pow(gx, 2) + pow(gy, 2);
18 }
19 double ex, ey, eans = 1e18;
20 void SA(){
21     double T = T0, x = 0, y = 0, ans = calculate
22     (x, y);
23     double ansx, ansy;
24     uniform_real_distribution<double> U;
25     while(T > Tk){
26         double nx, ny, nans;
27         nx = x + 2 * (U(MT) - .5) * T;
28         ny = y + 2 * (U(MT) - .5) * T;
29         if((nans = calculate(nx, ny)) < ans){
30             ans = nans;
31             ansx = x = nx;
32             ansy = y = ny;
33         } else if(exp(-distance(nx, ny, x, y) / T
34             / R) > U(MT)){
35             x = nx, y = ny;
36         }
37         T *= delta;
38     }
39     if(ans < eans) eans = ans, ex = ansx, ey =
40     ansy;
41 }

```

10.5 伪随机生成

```

1 #include "../header.cpp"
2 u32 xorshift32(u32 &x){
3     x ^= x << 13, x ^= x >> 17, x ^= x << 5;
4     return x;
5 }
6 u64 xorshift64(u64 &x){
7     x ^= x << 13, x ^= x >> 7, x ^= x << 17;
8     return x;
9 }

```