

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*

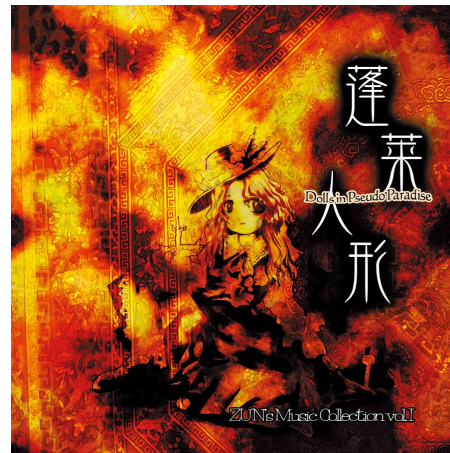
蓬莱人形算法模板库

REFERENCE DOCUMENT for *Dolls in Pseudo Paradise*



Coach

赵文博 邱思宇
Wenbo Zhao Siyu Qiu



Contestant

丁文涛 张彭凯 陈煜航
Wentao Ding Pengkai Zhang Yuhang Chen



Harbin Institute of Technology

2024-2025

10 其他	42
10.1 笛卡尔树	42
10.2 CDQ 分治	42
10.3 日期公式	42
10.4 misc	42
10.5 pbds 库	42
10.6 Python 技巧	42
10.7 快速测试	43
10.8 自适应辛普森	43
10.9 模拟退火	43
10.10 对拍脚本	43
10.11 伪随机生成	43

11 公共头文件	43
----------	----

附录	44
----	----

1 动态规划

1.1 多重背包

1.1.1 用法

n 个物品, m 容量背包, 第 i 个物品重量为 w_i 价值为 v_i 共有 c_i 个, 计算不超过容量的情况下最多拿多少价值的物品。

```
1 #include "../header.cpp"
2 int F[MAXN];
3 int main(){
4     int n, m; cin >> n >> m;
5     for(int i = 1; i <= n; ++ i){
6         int w, v, c; cin >> w >> v >> c;
7         // w: value, v: volume, c: count
8         for(int j = 0; j < v; ++ j){
9             deque <tuple<int, int> > Q;
10            for(int k = 0; j + k * v <= m; ++ k){
11                int x = j + k * v;
12                int f = F[x] - (x / v) * w;
13                while(!Q.empty() && get<0>(Q.back()) <= f)
14                    Q.pop_back();
15                Q.push_back({f, x});
16                while(!Q.empty() && get<1>(Q.front()) < x - c * v)
```

```
17            Q.pop_front();
18            F[x] = get<0>(Q.front()) + (x / v) * w;
19        }
20    }
21    cout << F[m] << endl;
22    return 0;
23 }
24 }
```

1.2 树形背包

```
1 #include <bits/stdc++.h>
2 using namespace std;
3 typedef long long i64;
4 const int MAXN = 2e3 + 3;
5 vector<int> E[MAXN];
6 int W[MAXN];
7 int F[MAXN][MAXN], S[MAXN];
8 void dfs(int u, int f){
9     F[u][1] = W[u], S[u] = 1;
10    for(auto &v : E[u]) if(v != f){
11        dfs(v, u);
12        for(int i = S[u]; i >= 1; -- i)
13            for(int j = S[v]; j >= 1; -- j)
14                F[u][i + j] = max(F[u][i + j], F[u][i] + F[v][j]);
15        S[u] += S[v];
16    }
17 }
18 int main(){
19     int n, m;
20     cin >> n >> m;
21     for(int i = 1; i <= n; ++ i){
22         int f;
23         cin >> f >> W[i];
24         E[f].push_back(i);
25     }
26     dfs(0, 0);
27     cout << F[0][m + 1] << endl;
28     return 0;
29 }
```

1.3 动态动态规划 1

1.3.1 例题

给定一棵 n 个点的树, 点有点权, 求最大独立集。 m 次修改, 每次把 x 的权值修改成 y 。

```
1 #include "../header.cpp"
2 int W[MAXN];
```

```
3 struct Mat{ int M[2][2]; };
4 struct Vec{ int V[2]; };
5 Mat operator *(const Mat &a, const Mat &b){
6     Mat c;
7     c.M[0][0] = max(a.M[0][0] + b.M[0][0], a.M[0][1] + b.M[1][0]);
8     c.M[0][1] = max(a.M[0][0] + b.M[0][1], a.M[0][1] + b.M[1][1]);
9     c.M[1][0] = max(a.M[1][0] + b.M[0][0], a.M[1][1] + b.M[1][0]);
10    c.M[1][1] = max(a.M[1][0] + b.M[0][1], a.M[1][1] + b.M[1][1]);
11    return c;
12 }
13 Vec operator *(const Mat &a, const Vec &v){
14     Vec r;
15     r.V[0] = max(a.M[0][0] + v.V[0], a.M[0][1] + v.V[1]);
16     r.V[1] = max(a.M[1][0] + v.V[0], a.M[1][1] + v.V[1]);
17     return r;
18 }
19 namespace Gra{
20     vector<int> E[MAXN];
21     int G[MAXN], S[MAXN], D[MAXN], T[MAXN], F[
22         MAXN];
23     int X[MAXN], Y[MAXN];
24     int H[MAXN][2];
25     int K[MAXN][2];
26     struct Mat M[MAXN];
27     void dfs1(int u, int f){
28         S[u] = 1;
29         F[u] = f;
30         for(auto &v : E[u]) if(v != f){
31             dfs1(v, u);
32             S[u] += S[v];
33             if(S[v] > S[G[u]]) G[u] = v;
34         }
35     }
36     int o;
37     void dfs2(int u, int f){
38         if(u == G[f])
39             X[u] = X[f];
40         else
41             X[u] = u;
42         H[u][0] = H[u][1] = 0;
43         K[u][0] = K[u][1] = 0;
44         const int &g = G[u];
45         D[u] = ++ o;
46         T[o] = u;
47         if(g){
```

```

47     dfs2(g, u);
48     Y[u] = Y[g];
49     K[u][0] += max(K[g][0], K[g][1]);
50     K[u][1] += K[g][0];
51 } else {
52     Y[u] = u;
53 }
54 for(auto &v : E[u]) if(v != f && v != g){
55     dfs2(v, u);
56     H[u][0] += max(K[v][0], K[v][1]);
57     H[u][1] += K[v][0];
58 }
59 M[u].M[0][0] = H[u][0];
60 M[u].M[0][1] = H[u][0];
61 M[u].M[1][0] = H[u][1] + W[u];
62 M[u].M[1][1] = -INF;
63 K[u][0] += H[u][0];
64 K[u][1] += H[u][1] + W[u];
65 }
66 }
67 namespace Seg{
68     const int SIZ = 4e5 + 3;
69     struct Mat M[SIZ];
70     #define lc(t) (t << 1)
71     #define rc(t) (t << 1 | 1)
72     void pushup(int t, int a, int b){
73         M[t] = M[lc(t)] * M[rc(t)];
74     }
75     void build(int t, int a, int b){
76         if(a == b){
77             M[t] = Gra :: M[Gra :: T[a]];
78         } else {
79             int c = a + b >> 1;
80             build(lc(t), a, c);
81             build(rc(t), c + 1, b);
82             pushup(t, a, b);
83         }
84     }
85     void modify(int t, int a, int b, int p,
86         const Mat &w){
87         if(a == b){
88             M[t] = w;
89         } else {
90             int c = a + b >> 1;
91             if(p <= c) modify(lc(t), a, c, p, w);
92             else modify(rc(t), c + 1, b, p, w);
93             pushup(t, a, b);
94         }
95     }
96     Mat query(int t, int a, int b, int l, int r)
97     {

```

```

96     if(l <= a && b <= r){
97         return M[t];
98     } else {
99         int c = a + b >> 1;
100        if(r <= c) return query(lc(t), a, c , l
101            , r); else
102            if(l > c) return query(rc(t), c + 1, b,
103                l, r); else
104                return query(lc(t), a, c , l, r) *
105                    query(rc(t), c + 1, b, l, r);
106    }
107 }
108 int qread();
109 int main(){
110     int n = qread(), m = qread();
111     up(1, n, i)
112     W[i] = qread();
113     up(2, n, i){
114         int u = qread(), v = qread();
115         Gra :: E[u].push_back(v);
116         Gra :: E[v].push_back(u);
117     }
118     Gra :: dfs1(1, 0);
119     Gra :: dfs2(1, 0);
120     Seg :: build(1, 1, n);
121     Vec v0;
122     v0.V[0] = v0.V[1] = 0;
123     up(1, m, i){
124         using namespace Gra;
125         int x = qread(), y = qread();
126         W[x] = y;
127         int u = x;
128         while(u != 0){
129             const int &v = X[u];
130             const int &f = F[v];
131             M[u].M[0][0] = H[u][0];
132             M[u].M[0][1] = H[u][0];
133             M[u].M[1][0] = H[u][1] + W[u];
134             M[u].M[1][1] = -INF;
135             const Vec p = Seg :: query(1, 1, n, D[v
136                 ], D[Y[u]]) * v0;
137             Seg :: modify(1, 1, n, D[u], M[u]);
138             const Vec q = Seg :: query(1, 1, n, D[v
139                 ], D[Y[u]]) * v0;
140             if(f != 0){
141                 H[f][0] = H[f][0] - max(p.V[0], p.V
142                     [1]) + max(q.V[0], q.V[1]);
143                 H[f][1] = H[f][1] - p.V[0] + q.V[0];
144             }
145             u = f;

```

```

142     }
143     Vec v1 = Seg :: query(1, 1, n, D[1], D[Y
144         [1]]) * v0;
145     printf("%d\n", max(v1.V[0], v1.V[1]));
146 }
147 return 0;

```

1.4 插头 dp

1.4.1 例题

给出 $n \times m$ 的方格，有些格子不能铺线，其它格子必须铺，形成一个闭合回路。问有多少种铺法？

```

1 #include "../header.cpp"
2 namespace HashT{
3     const int SIZ = 19999997;
4     int H[SIZ], V[SIZ], N[SIZ], t;
5     bool F[SIZ];
6     i64 W[SIZ];
7     void add(int u, int v, bool f, i64 w){
8         V[++ t] = v, N[t] = H[u], F[t] = f, W[t] =
9             w, H[u] = t;
10 }
11 i64 find(int u, bool f){
12     for(int p = H[u % SIZ]; p; p = N[p]){
13         if(V[p] == u && F[p] == f)
14             return W[p];
15     }
16     add(u % SIZ, u, f, 0);
17     return W[t];
18 }
19 char S[MAXN][MAXN];
20 int qread();
21 int n, m;
22 vector <pair<pair<int, bool>, i64> > M[2];
23 int getp(int s, int p){
24     return (s >> (2 * p - 2)) & 3;
25 }
26 int setw(int s, int p, int w){
27     return (s & ~(3 << (2 * p - 2))) | (w << (2
28         * p - 2));
29 }
30 int findr(int s, int p){
31     int c = 0;
32     for(int q = p; q <= m + 1; ++ q){
33         if(((s >> (2 * q - 2)) & 3) == 1) ++ c;
34         if(((s >> (2 * q - 2)) & 3) == 2) -- c;
35         if(c == 0)
36             return q;
37     }

```

```

36 return -1;
37 }
38 int findl(int s, int p){
39     int c = 0;
40     for(int q = p; q ≥ 1; -- q){
41         if(((s >> (2 * q - 2)) & 3) == 2) ++ c;
42         if(((s >> (2 * q - 2)) & 3) == 1) -- c;
43         if(c == 0)
44             return q;
45     }
46     return -1;
47 }
48 void state(int s){
49     return ;
50     up(1, m + 1, i){
51         switch(getp(s, i)){
52             case 0 : putchar('#'); break;
53             case 1 : putchar('('); break;
54             case 2 : putchar(')'); break;
55             case 3 : putchar('E');
56         }
57     }
58     puts("");
59 }
60 int main(){
61     n = qread(), m = qread();
62     up(1, n, i)
63         scanf("%s", S[i] + 1);
64     int o = 0;
65     #define X M[o]
66     #define Y M[!o]
67     vector <pair<int, bool>> T;
68     X.push_back({0, 0}, {1});
69     up(1, n, i){
70         Y.clear();
71         for(auto &u : X){
72             auto [s0, c] = u;
73             auto [s, f] = s0;
74             if(getp(s, m + 1) == 0)
75                 Y.push_back({s << 2, f}, c});
76         }
77         o ^= 1;
78         up(1, m, j){
79             int x = j, y = j + 1;
80             for(auto &u : X){
81                 auto [s0, c] = u;
82                 auto [s, f] = s0;
83                 int a = getp(s, x);
84                 int b = getp(s, y);
85                 int t = setw(setw(s, x, 0), y, 0);
86                 #define update(t, c) HashT :: find(t,

```

```

87         f) += c, T.push_back({t, f})
88     if(S[i][j] == '.'){ // 经过该格
89         if(a == 1 && b == 1){
90             t = setw(t, findr(s, y), 1),
91             update(t, c);
92         } else
93         if(a == 2 && b == 2){
94             t = setw(t, findl(s, x), 2),
95             update(t, c);
96         } else
97         if(a == 1 && b == 2){
98             if(f == false) // 还没有闭合回路
99                 f = true, update(t, c);
100         } else
101         if(a == 2 && b == 1){
102             update(t, c);
103         } else
104         if(a == 0 && b == 0){
105             t = setw(t, x, 1);
106             t = setw(t, y, 2);
107             update(t, c);
108         } else { // a = 0 || b = 0
109             int t1 = setw(t, x, a | b);
110             int t2 = setw(t, y, a | b);
111             update(t1, c);
112             update(t2, c);
113         }
114     }
115     if(S[i][j] == '*'){ // 不经过该格
116         if(a == 0 && b == 0)
117             update(t, c);
118     }
119 }
120 Y.clear();
121 for(auto &u : T){
122     auto [s, f] = u;
123     if(HashT :: find(s, f) ≠ 0){
124         Y.push_back({s, f}, HashT :: find(s,
125             , f));
126         HashT :: find(s, f) = 0;
127     }
128 }
129 T.clear(), o ^= 1;
130 }
131 i64 ans = 0;
132 for(auto &u : X){
133     auto [s0, c] = u;
134     auto [s, f] = s0;
135     bool g = true;
136     up(1, m + 1, i)

```

```

136     g &= getp(s, i) == 0;
137     f &= g;
138     if(f)
139         ans = c;
140 }
141 printf("%lld\n", ans);
142 return 0;
143 }

```

1.5 dp-quad-inequality

```

1 #include "../header.cpp"
2 // 要求对于  $a \leq b \leq c \leq d$  有  $w(b, c) \leq w(a, d)$  且
    $w(a, c) + w(b, d) \leq w(a, d) + w(b, c)$ 
3 int w(int l, int r);
4 int f[MAXN][MAXN], m[MAXN][MAXN], n;
5 int solve(){
6     for(int len = 2; len ≤ n; ++len)
7         for(int l = 1, r = len; r ≤ n; ++l, ++r){
8             f[l][r] = INF;
9             for(int k = m[l][r - 1]; k ≤ m[l + 1][r
10                ]; ++k){
11                 int u = f[l][k] + f[k + 1][r] + w(l, r);
12                 if(f[l][r] > u)
13                     f[l][r] = u, m[l][r] = k;
14             }
15 }

```

1.6 斜率优化

1.6.1 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。

1.6.2 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 $s_i = i + \sum_{j \leq i} c_j, L' = L + 1$ 。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      ++ e;
21    int j = Q[e];
22    F[i] = F[j] + pow(S[i] - S[j] - L, 2);
23    while(1 < p && gtk(Q[p - 1], Q[p]) > gtk(Q[p], i))
24      e -= (e == p), -- p;
25    Q[++ p] = i;
26  }
```

```
27 printf("%.0Lf\n", F[n]);
28 return 0;
29 }
```

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 SIZ = 1e6 + 1e5 + 3;
5   int F[SIZ], C[SIZ], S[SIZ], W[SIZ], X[SIZ][2], sz;
6   u64 H[SIZ];
7   int newnode(int w){
8     W[++ sz] = w, C[sz] = S[sz] = 1, H[sz] = MT();
9     return sz;
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), root = merge(p, q);
45  }
46  void erase(int &root, int w){
47    auto [p, q] = split(root, w);
48    auto [a, b] = split(p, w - 1);
49    -- C[b], -- S[b];
50    p = C[b] == 0 ? a : merge(a, b);
51    root = merge(p, q);
52  }
53  int find_rank(int &root, int w){
54    int x = root, o = x, a = 0;
55    for(; x;){
56      if(w < W[x])
57        o = x, x = X[x][0];
58      else {
59        a += S[X[x][0]];
60        if(w == W[x]){ o = x; break; }
61        a += C[x];
62        o = x, x = X[x][1];
63      }
64    }
65    return a + 1;
66  }
67  int find_kth(int &root, int w){
68    int x = root, o = x, a = 0;
69    for(; x;){
70      if(w <= S[X[x][0]])
71        o = x, x = X[x][0];
72      else {
73        w -= S[X[x][0]];
74        if(w <= C[x]){ o = x; break; }
75        w -= C[x];
76        o = x, x = X[x][1];
77      }
78    }
79    return W[o];
80  }
81  int find_pre(int &root, int w){
82    return find_kth(root, find_rank(root, w) - 1);
83  }
84  int find_suc(int &root, int w){
85    return find_kth(root, find_rank(root, w + 1));
86  }
87 }
```

2.1.2 Splay

```

1 #include "../header.cpp"
2 namespace Splay{
3     const int SIZ = 1e6 + 1e5 + 3;
4     int F[SIZ], C[SIZ], S[SIZ], X[SIZ][2], size;
5     bool T[SIZ];
6     bool is_root(int x){ return F[x] == 0; }
7     bool is_rson(int x){ return X[F[x]][1] == x; }
8     void push_down(int x){
9         if(!T[x]) return;
10        int lc = X[x][0], rc = X[x][1];
11        if(lc) T[lc] ^= 1, swap(X[lc][0], X[lc][1]);
12        if(rc) T[rc] ^= 1, swap(X[rc][0], X[rc][1]);
13        T[x] = 0;
14    }
15    void pushup(int x){
16        S[x] = C[x] + S[X[x][0]] + S[X[x][1]];
17    }
18    void rotate(int x){
19        int y = F[x], z = F[y];
20        bool f = is_rson(x), g = is_rson(y);
21        int &t = X[x][!f];
22        if(z){ X[z][g] = x; }
23        if(t){ F[t] = y; }
24        X[y][f] = t, t = y;
25        F[y] = x, pushup(y);
26        F[x] = z, pushup(x);
27    }
28    void splay(int &r, int x, int g = 0){
29        for(int f; f = F[x], f != g; rotate(x))
30            if(F[f] != g) rotate(is_rson(x) == is_rson(f) ? f : x);
31        if(is_root(x)) r = x;
32    }
33    int get_kth(int &r, int w){
34        int x = r, o = x;
35        for(;;){
36            push_down(x);
37            if(w <= S[X[x][0]]) o = x, x = X[x][0];
38            else {
39                w -= S[X[x][0]];
40                if(C[x] && w <= C[x]){ o = x; break; }
41                w -= C[x], o = x, x = X[x][1];
42            }
43        }
44        splay(r, o); return o;
45    }
46    int build(int l, int r){

```

```

46    if(l == r){
47        C[l] = S[l] = 1; return l;
48    }
49    int c = l + r >> 1, a = 0, b = 0;
50    if(l <= c - 1) a = build(l, c - 1), F[a] = c, X[c][0] = a;
51    if(c + 1 <= r) b = build(c + 1, r), F[b] = c, X[c][1] = b;
52    C[c] = 1, pushup(c); return c;
53 }
54 }

```

2.1.3 Treap

```

1 #include "../header.cpp"
2 mt19937_64 MT(114514);
3 namespace Treap{
4     const int SIZ = 1e6 + 1e5 + 3;
5     int F[SIZ], C[SIZ], S[SIZ], W[SIZ], X[SIZ][2], sz;
6     u64 H[SIZ];
7     bool is_root(int x){ return F[x] == 0; }
8     bool is_rson(int x){ return X[F[x]][1] == x; }
9     int newnode(int w){
10        W[++sz] = w, C[sz] = S[sz] = 1; H[sz] = MT();
11        return sz;
12    }
13    void pushup(int x){
14        S[x] = C[x] + S[X[x][0]] + S[X[x][1]];
15    }
16    void rotate(int &root, int x){
17        int y = F[x], z = F[y];
18        bool f = is_rson(x);
19        bool g = is_rson(y);
20        int &t = X[x][!f];
21        if(z){ X[z][g] = x; } else root = x;
22        if(t){ F[t] = y; }
23        X[y][f] = t, t = y;
24        F[y] = x, pushup(y);
25        F[x] = z, pushup(x);
26    }
27    void insert(int &root, int w){
28        if(root == 0) {root = newnode(w); return;}
29        int x = root, o = x;
30        for(;;x;o = x, x = X[x][w > W[x]]){
31            ++S[x]; if(w == W[x]){ ++C[x], o = x; break; }
32        }
33        if(W[o] != w){

```

```

34        if(w < W[o]) X[o][0] = newnode(w), F[sz] = 0, o = sz;
35        else X[o][1] = newnode(w), F[sz] = o, o = sz;
36    }
37    while(!is_root(o) && H[o] < H[F[o]])
38        rotate(root, o);
39 }
40 void erase(int &root, int w){
41     int x = root, o = x;
42     for(;;x;o = x, x = X[x][w > W[x]]){
43         --S[x]; if(w == W[x]){ --C[x], o = x; break; }
44     }
45     if(C[o] == 0){
46         while(X[o][0] || X[o][1]){
47             u64 wl = X[o][0] ? H[X[o][0]] : ULLONG_MAX;
48             u64 wr = X[o][1] ? H[X[o][1]] : ULLONG_MAX;
49             if(wl < wr){
50                 int p = X[o][0]; rotate(root, p);
51             } else {
52                 int p = X[o][1]; rotate(root, p);
53             }
54         }
55         if(is_root(o)){
56             root = 0;
57         } else {
58             X[F[o]][is_rson(o)] = 0;
59         }
60     }
61 }
62 int find_rank(int &root, int w){
63     int x = root, o = x, a = 0;
64     for(;;x){
65         if(w < W[x])
66             o = x, x = X[x][0];
67         else {
68             a += S[X[x][0]];
69             if(w == W[x]){
70                 o = x; break;
71             }
72             a += C[x];
73             o = x, x = X[x][1];
74         }
75     }
76     return a + 1;
77 }
78 int find_kth(int &root, int w){
79     int x = root, o = x, a = 0;

```

```

80 for(;x;){
81     if(w ≤ S[X[x][0]])
82         o = x, x = X[x][0];
83     else {
84         w -= S[X[x][0]];
85         if(w ≤ C[x]){
86             o = x; break;
87         }
88         w -= C[x];
89         o = x, x = X[x][1];
90     }
91 }
92 return W[x];
93 }
94 int find_pre(int &root, int w){
95     return find_kth(root, find_rank(root, w) - 1);
96 }
97 int find_suc(int &root, int w){
98     return find_kth(root, find_rank(root, w + 1));
99 }
100 }

```

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) [p, +inf), 返回后者迭代器
6     auto split(int p) {
7         auto it = prev(M.upper_bound(p));
8         return M.insert(
9             it,
10             make_pair(p, it → second)
11         );
12     }
13     // 区间赋值
14     void assign(int l, int r, int v) {
15         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], s;
5     bool E[SIZ];
6     int merge(int u, int v){
7         if(u = 0 || v = 0)
8             return u | v;
9         if(W[u] > W[v] || (W[u] == W[v] && u > v))
10             swap(u, v);
11         int &lc = L[u], &rc = R[u];
12         rc = merge(rc, v);
13         if(D[lc] < D[rc]) swap(lc, rc);
14         D[u] = min(D[lc], D[rc]) + 1;
15         if(lc ≠ 0) F[lc] = u;
16         if(rc ≠ 0) F[rc] = u;
17         return u;
18     }
19     void pop(int &root){
20         int root0 = merge(L[root], R[root]);
21         F[root0] = F[root] = root0;
22         E[root] = true, root = root0;
23     }
24     int top(int &root){ return W[root]; }
25     int getfa(int u){
26         return u = F[u] ? u : F[u] = getfa(F[u]);
27     }
28     int newnode(int w){
29         ++ s, W[s] = w, F[s] = s, D[s] = 1;
30         return s;
31     }
32 }

```

2.4 KD 树

```

1 #include "../header.cpp"
2 const int LG = 18; // 二进制分组合并
3 struct pt { // x: 坐标; v: 权值; l, r: 儿子
4     int x[2], v, sum, l, r, L[2], R[2];
5 } t[MAXN], l, h; // l, h: 查询的左上/右下角
6 int rt[LG], b[MAXN], cnt;
7 void update(int p) {

```

```

8     t[p].sum = t[t[p].l].sum + t[t[p].r].sum + t[p].v;
9     for (int k: {0, 1}) {
10         t[p].L[k] = t[p].R[k] = t[p].x[k];
11         if(t[p].l)
12             t[p].L[k] = min(t[p].L[k], t[t[p].l].L[k]),
13             t[p].R[k] = max(t[p].R[k], t[t[p].l].R[k]);
14         if(t[p].r)
15             t[p].L[k] = min(t[p].L[k], t[t[p].r].L[k]),
16             t[p].R[k] = max(t[p].R[k], t[t[p].r].R[k]);
17     }
18 }
19 int build(int l, int r, int dep = 0){ // 重构
20     int p{(l + r) >> 1};
21     nth_element(b + l, b + p, b + r + 1, [dep](
22         int x, int y) { return t[x].x[dep] < t[y].x[dep]; });
23     int x{b[p]};
24     if(l < p) t[x].l = build(l, p - 1, dep ^ 1);
25     if(p < r) t[x].r = build(p + 1, r, dep ^ 1);
26     update(x);
27     return x;
28 }
29 void append(int &p) { // 收集 p 子树等待重构
30     if(!p) return;
31     b[++cnt] = p;
32     append(t[p].l), append(t[p].r), p = 0;
33 }
34 int query(int p) { // 查询 p 子树矩阵内和
35     if (!p) return 0;
36     bool flag = false;
37     for (int k: {0, 1}) flag |= (!(l.x[k] ≤ t[p].L[k] && t[p].R[k] ≤ h.x[k]));
38     if (!flag) return t[p].sum;
39     for (int k: {0, 1})
40         if (t[p].R[k] < l.x[k] || h.x[k] < t[p].L[k]) return 0;
41     int ans = 0;
42     flag = false;
43     for (int k: {0, 1}) flag |= (!(l.x[k] ≤ t[p].x[k] && t[p].x[k] ≤ h.x[k]));
44     if (!flag) ans = t[p].v;
45     return ans += query(t[p].l) + query(t[p].r);
46 }
47 int main() {
48     int n; cin >> n; n = 0; // n: 当前 KDT 大小
49     while (true) {
50         int op; cin >> op;
51         if (op == 1) {
52             int x, y, A; cin >> x >> y >> A;
53             t[++ n] = {{x, y}, A};

```

```

53     b[cnt = 1] = n;
54     for (int sz = 0;; ++sz) // 二进制合并
55         if (!rt[sz]) {
56             rt[sz] = build(1, cnt); break;
57         } else append(rt[sz]);
58     } else if (op == 2) {
59         cin >> l.x[0] >> l.x[1] >> h.x[0] >> h.x
60             [1];
61         int ans = 0;
62         for (int i = 0; i < LG; ++i) ans += query
63             (rt[i]);
64         cout << ans << "\n";
65     } else break;
66     return 0;

```

2.5 线性基

```

1 #include "../header.cpp"
2 namespace LB{
3     const int SIZ = 60 + 3;
4     i64 W[SIZ], h = 60;
5     void insert(i64 w){
6         for (int i = h; i ≥ 0; -- i){
7             if (w & (1ll << i)){
8                 if (!W[i]){
9                     W[i] = w;
10                    break;
11                } else {
12                    w ^= W[i];
13                }
14            }
15        }
16    }
17    i64 query(i64 x){
18        for (int i = h; i ≥ 0; -- i){
19            if (W[i]){
20                x = max(x, x ^ W[i]);
21            }
22        }
23        return x;
24    }
25 }
26 namespace realLB{
27     const int SIZ = 500 + 3;
28     long double W[SIZ][SIZ];
29     int n = 0;
30     void init(int n0){
31         n = n0;
32     }

```

```

33     bool zero(long double w){
34         return fabs(w) < 1e-9;
35     }
36     bool insert(long double X[]){
37         for (int i = 1; i ≤ n; ++ i){
38             if (!zero(X[i])){
39                 if (zero(W[i][i])){
40                     for (int j = 1; j ≤ n; ++ j)
41                         W[i][j] = X[j];
42                     return true;
43                 } else {
44                     long double t = X[i] / W[i][i];
45                     for (int j = 1; j ≤ n; ++ j)
46                         X[j] -= t * W[i][j];
47                 }
48             }
49         }
50         return false;
51     }
52 }
53 // === TEST ===
54 int qread();
55 const int MAXN = 500 + 3;
56 long double X[MAXN][MAXN], C[MAXN];
57 int I[MAXN];
58 bool cmp(int a, int b){
59     return C[a] < C[b];
60 }
61 int main(){
62     int n, m;
63     cin >> n >> m;
64     realLB :: init(m);
65     for (int i = 1; i ≤ n; ++ i){
66         for (int j = 1; j ≤ m; ++ j){
67             cin >> X[i][j];
68         }
69     }
70     for (int i = 1; i ≤ n; ++ i){
71         cin >> C[i];
72         I[i] = i;
73     }
74     sort(I + 1, I + 1 + n, cmp);
75     int ans = 0, cnt = 0;
76     for (int i = 1; i ≤ n; ++ i){
77         int x = I[i];
78         if (realLB :: insert(X[x]))
79             ans += C[x],
80             cnt += 1;
81     }
82     cout << cnt << "␣" << ans << endl;
83     return 0;

```

```

84 }

```

2.6 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){ // 创建节点, 返回编号
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;
38             X[y][f] = X[x][!f], F[X[x][!f]] = y;
39             X[x][!f] = y;
40         } else {
41             F[x] = z, F[y] = x;
42             X[z][g] = x;
43             X[y][f] = X[x][!f], F[X[x][!f]] = y;
44             X[x][!f] = y;
45         }
46         push_up(y), push_up(x);
47     }

```

```

45 void splay(int x){ // 旋到树根
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){ // 将根到 x 变成实链
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){ // 使 x 为所在树的根
60     x = access(x);
61     T[x] ^= 1, swap(X[x][0], X[x][1]);
62 }
63 int find_root(int x){ // 查找 x 所在树的根
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){ // 连边
70     make_root(x), splay(x), F[x] = y;
71 }
72 void cut(int x, int p){ // 删边
73     make_root(x), access(p), splay(p), X[p][0]
74     = F[x] = 0;
75 }
76 void modify(int x, int w){ // 修改点权
77     splay(x), W[x] = w, push_up(x);
78 }

```

2.7 线段树

2.7.1 李超树

```

1 #include "../header.cpp"
2 struct Line{ int id; double k, b; Line() =
3     default; };
4 namespace LCSeg{
5     const int SIZ = 2e5 + 3;
6     struct Line T[SIZ];
7     #define lc(t) (t << 1)
8     #define rc(t) (t << 1 | 1)
9     bool cmp(int p, Line x, Line y){
10         double w1 = x.k * p + x.b;
11         double w2 = y.k * p + y.b;
12         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
16     y){
17     int c = a + b >> 1;
18     if(cmp(c, x, y)) swap(x, y);
19     if(cmp(a, y, x)){
20         T[t] = x; if(a != b) merge(rc(t), c + 1,
21             b, T[rc(t)], y);
22     } else {
23         T[t] = x; if(a != b) merge(lc(t), a, c,
24             T[lc(t)], y);
25     }
26 }
27 // 插入线段 (l, f(l)) -- (r, f(r))
28 void modify(int t, int a, int b, int l, int
29     r, Line x){
30     if(l <= a && b <= r) merge(t, a, b, T[t],
31         x);
32     else {
33         int c = a + b >> 1;
34         if(l <= c) modify(lc(t), a, c, l, r, x);
35         if(r > c) modify(rc(t), c + 1, b, l, r,
36             x);
37     }
38 }
39 // 查询 x = p 位置最高的线段 (有多条取编号最
40     小)
41 void query(int t, int a, int b, int p, Line
42     &x){
43     if(cmp(p, x, T[t])) x = T[t];
44     if(a != b){
45         int c = a + b >> 1;
46         if(p <= c) query(lc(t), a, c, p, x);
47         if(p > c) query(rc(t), c + 1, b, p, x);
48     }
49 }

```

2.7.2 线段树 3

例题 给出一个长度为 n 的数列 A , 同时定义一个辅助数组 B , B 开始与 A 完全相同。接下来进行了 m 次操作, 操作有五种类型, 按以下格式给出:

- 1 l r k: 对于所有的 $i \in [l, r]$, 将 A_i 加上 k (k 可以为负数)。
- 2 l r v: 对于所有的 $i \in [l, r]$, 将 A_i 变成

$\min(A_i, v)$ 。

- 3 l r: 求 $\sum_{i=l}^r A_i$ 。
- 4 l r: 对于所有的 $i \in [l, r]$, 求 A_i 的最大值。
- 5 l r: 对于所有的 $i \in [l, r]$, 求 B_i 的最大值。

在每一次操作后, 我们都进行一次更新, 让 $B_i \leftarrow \max(B_i, A_i)$ 。

```

1 #include "../header.cpp"
2 int A[MAXN];
3 struct Node{
4     i64 sum; int len, max1, max2, max_cnt,
5     his_mx;
6     Node():
7         sum(0), max1(-INF), max2(-INF), max_cnt(0),
8         his_mx(-INF), len(0) {}
9     Node(int w):
10         sum(w), max1(w), max2(-INF), max_cnt(1),
11         his_mx(w), len(1) {}
12     bool update(int w1, int w2, int h1, int h2){
13         his_mx = max(his_mx, max1 + h1);
14         max1 += w1, max2 += w2;
15         sum += 1ll * w1 * max_cnt + 1ll * w2 * (
16             len - max_cnt);
17         return max1 > max2;
18     }
19 };
20 struct Tag{
21     int max_add, max_his_add, umx_add,
22     umx_his_add; bool have;
23     void update(int w1, int w2, int h1, int h2){
24         max_his_add = max(max_his_add, max_add +
25             h1);
26         umx_his_add = max(umx_his_add, umx_add +
27             h2);
28         max_add += w1, umx_add += w2, have = true;
29     }
30     void clear(){
31         max_add = max_his_add = umx_add =
32         umx_his_add = have = 0;
33     }
34 };
35 struct Node operator +(Node a, Node b){
36     Node t;
37     t.max1 = max(a.max1, b.max1);
38     if(t.max1 != a.max1){
39         if(a.max1 > t.max2) t.max2 = a.max1;
40     } else {
41         if(a.max2 > t.max2) t.max2 = a.max2;
42         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 ;
44 t.his_mx = max(a.his_mx, b.his_mx);
45 return t;
46 }
47 namespace Seg{
48     const int SIZ = 2e6 + 3;
49     struct Node W[SIZ]; struct Tag T[SIZ];
50     #define lc(t) (t << 1)
51     #define rc(t) (t << 1 | 1)
52     void push_up(int t, int a, int b){
53         W[t] = W[lc(t)] + W[rc(t)];
54     }
55     void push_down(int t, int a, int b){
56         if(a == b) T[t].clear();
57         if(T[t].have){
58             int c = a + b >> 1, x = lc(t), y = rc(t)
59             ;
60             int w = max(W[x].max1, W[y].max1);
61             int w1 = T[t].max_add, w2 = T[t].umx_add
62             , w3 = T[t].max_his_add, w4 = T[t].
63             umx_his_add;
64             if(w == W[x].max1)
65                 W[x].update(w1, w2, w3, w4),
66                 T[x].update(w1, w2, w3, w4);
67             else
68                 W[x].update(w2, w2, w4, w4),
69                 T[x].update(w2, w2, w4, w4);
70             if(w == W[y].max1)
71                 W[y].update(w1, w2, w3, w4),
72                 T[y].update(w1, w2, w3, w4);
73             else
74                 W[y].update(w2, w2, w4, w4),
75                 T[y].update(w2, w2, w4, w4);
76             T[t].clear();
77         }
78     }
79     void build(int t, int a, int b){
80         if(a == b){W[t] = Node(A[a]), T[t].clear()

```

```

81     }
82     }
83     void modiadd(int t, int a, int b, int l, int
84     r, int w){
85         if(l ≤ a && b ≤ r){
86             T[t].update(w, w, w, w);
87             W[t].update(w, w, w, w);
88         } else {
89             int c = a + b >> 1; push_down(t, a, b);
90             if(l ≤ c) modiadd(lc(t), a, c, l, r,
91             w);
92             if(r > c) modiadd(rc(t), c + 1, b, l, r
93             , w);
94             push_up(t, a, b);
95         }
96     }
97     void modimin(int t, int a, int b, int l, int
98     r, int w){
99         if(l ≤ a && b ≤ r){
100             if(w ≥ W[t].max1) return; else
101             if(w > W[t].max2){
102                 int k = w - W[t].max1;
103                 T[t].update(k, 0, k, 0);
104                 W[t].update(k, 0, k, 0);
105             } else {
106                 int c = a + b >> 1;
107                 push_down(t, a, b);
108                 modimin(lc(t), a, c, l, r, w);
109                 modimin(rc(t), c + 1, b, l, r, w);
110                 push_up(t, a, b);
111             }
112         } else {
113             int c = a + b >> 1; push_down(t, a, b);
114             if(l ≤ c) modimin(lc(t), a, c, l, r,
115             w);
116             if(r > c) modimin(rc(t), c + 1, b, l, r
117             , w);
118             push_up(t, a, b);
119         }
120     }
121     Node query(int t, int a, int b, int l, int r

```

```

122 }
123 int qread();
124 int main(){
125     int n = qread(), m = qread();
126     for(int i = 1; i ≤ n; ++ i)
127         A[i] = qread();
128     Seg :: build(1, 1, n);
129     for(int i = 1; i ≤ m; ++ i){
130         int op = qread();
131         if(op == 1){
132             int l = qread(), r = qread(), w = qread
133             ();
134             Seg :: modiadd(1, 1, n, l, r, w);
135         } else if(op == 2){
136             int l = qread(), r = qread(), w = qread
137             ();
138             Seg :: modimin(1, 1, n, l, r, w);
139         } else if(op == 3){
140             int l = qread(), r = qread();
141             auto p = Seg :: query(1, 1, n, l, r);
142             printf("%lld\n", p.sum);
143         } else if(op == 4){
144             int l = qread(), r = qread();
145             auto p = Seg :: query(1, 1, n, l, r);
146             printf("%d\n", p.max1);
147         } else if(op == 5){
148             int l = qread(), r = qread();
149             auto p = Seg :: query(1, 1, n, l, r);
150             printf("%d\n", p.his_mx);
151         }
152     }
153     return 0;
154 }

```

2.7.3 扫描线

```

1 #include "../header.cpp"
2 const int MAXN = 1e5 + 3;
3 int X1[MAXN], Y1[MAXN];
4 int X2[MAXN], Y2[MAXN];
5 int n, h, H[MAXN * 2];
6 namespace Seg{
7     #define lc(t) (t << 1)
8     #define rc(t) (t << 1 | 1)
9     const int SIZ = 8e5 + 3;
10     int T[SIZ], S[SIZ], L[SIZ];
11     void pushup(int t, int a, int b){
12         S[t] = 0;
13         if(a != b){
14             S[t] = S[lc(t)] + S[rc(t)];
15             L[t] = L[lc(t)] + L[rc(t)];

```

```

16     }
17     if(T[t]) S[t] = L[t];
18 }
19 void modify(int t, int a, int b, int l, int
    r, int w){
20     if(l ≤ a && b ≤ r){
21         T[t] += w, pushup(t, a, b);
22     } else {
23         int c = a + b >> 1;
24         if(l ≤ c) modify(lc(t), a, c, l, r, w);
25         if(r > c) modify(rc(t), c + 1, b, l, r,
            w);
26         pushup(t, a, b);
27     }
28 }
29 void build(int t, int a, int b){
30     if(a == b){
31         L[t] = H[a] - H[a - 1];
32     } else {
33         int c = a + b >> 1;
34         build(lc(t), a, c);
35         build(rc(t), c + 1, b);
36         pushup(t, a, b);
37     }
38 }
39 int query(int t){
40     return S[t];
41 }
42 }
43 tuple <int, int, int> P[MAXN], Q[MAXN];
44 int main(){
45     n = qread();
46     for(int i = 1; i ≤ n; ++ i){
47         X1[i] = qread(), Y1[i] = qread();
48         X2[i] = qread(), Y2[i] = qread();
49         if(X1[i] > X2[i]) swap(X1[i], X2[i]);
50         if(Y1[i] > Y2[i]) swap(Y1[i], Y2[i]);
51         H[++ h] = Y1[i];
52         H[++ h] = Y2[i];
53         P[i] = make_tuple(X1[i], Y1[i], Y2[i]);
54         Q[i] = make_tuple(X2[i], Y1[i], Y2[i]);
55     }
56     sort(H + 1, H + 1 + h);
57     sort(P + 1, P + 1 + n);
58     sort(Q + 1, Q + 1 + n);
59     int o = unique(H + 1, H + 1 + h) - H - 1;
60     Seg :: build(1, 1, o);
61     i64 ans = 0, last = -1;
62     int p = 1, q = 1;
63     while(p ≤ n || q ≤ n){
64         int x = INF;

```

```

65         if(p ≤ n) x = min(x, get<0>(P[p]));
66         if(q ≤ n) x = min(x, get<0>(Q[q]));
67         if(last ≠ -1){
68             ans += 1ll * Seg :: query(1) * (x - last
                );
69         }
70         last = x;
71         while(q ≤ n && get<0>(Q[q]) == x){
72             auto [x, l, r] = Q[q]; ++ q;
73             l = lower_bound(H + 1, H + 1 + o, l) - H
                + 1;
74             r = lower_bound(H + 1, H + 1 + o, r) - H
                ;
75             Seg :: modify(1, 1, o, l, r, 1);
76         }
77         while(p ≤ n && get<0>(P[p]) == x){
78             auto [x, l, r] = P[p]; ++ p;
79             l = lower_bound(H + 1, H + 1 + o, l) - H
                + 1;
80             r = lower_bound(H + 1, H + 1 + o, r) - H
                ;
81             Seg :: modify(1, 1, o, l, r, -1);
82         }
83     }
84     printf("%lld\n", ans);
85     return 0;
86 }

```

2.8 根号数据结构

2.8.1 块状链表

```

1 #include "../header.cpp"
2 namespace BLOCK{
3     const int SIZ = 1e6 + 1e5 + 3;
4     const int BSZ = 2000;
5     list <vector<int>> block;
6     void build(int n, const int A[]){
7         for(int l = 0, r = 0; r ≠ n; ){
8             l = r;
9             r = min(l + BSZ / 2, n);
10            vector <int> V0(A + l, A + r);
11            block.emplace_back(V0);
12        }
13    }
14    int get_kth(int k){
15        for(auto it = block.begin(); it ≠ block.
            end(); ++ it){
16            if(it → size() < k)
17                k -= it → size();
18            else return it → at(k - 1);
19        }

```

```

20        return -1;
21    }
22    int get_rank(int w){
23        int ans = 0;
24        for(auto it = block.begin(); it ≠ block.
            end(); ++ it){
25            if(it → back() < w)
26                ans += it → size();
27            else {
28                ans += lower_bound(it → begin(), it
                    → end(), w) - it → begin();
29                break;
30            }
31        }
32        return ans + 1;
33    }
34    // 插入到第 k 个位置
35    void insert(int k, int w){
36        for(auto it = block.begin(); it ≠ block.
            end(); ++ it){
37            if(it → size() < k)
38                k -= it → size();
39            else{
40                it → insert(it → begin() + k - 1, w)
                    ;
41                if(it → size() > BSZ){
42                    vector <int> V1(it → begin(), it →
                        begin() + BSZ / 2);
43                    vector <int> V2(it → begin() + BSZ
                        / 2, it → end());
44                    *it = V2;
45                    block.insert(it, V1);
46                }
47                return;
48            }
49        }
50    }
51    // 删除第 k 个数
52    void erase(int k){
53        for(auto it = block.begin(); it ≠ block.
            end(); ++ it){
54            if(it → size() < k)
55                k -= it → size();
56            else{
57                it → erase(it → begin() + k - 1);
58                if(it → empty())
59                    block.erase(it);
60                return;
61            }
62        }
63    }

```

```

64 }
65 int A[MAXN];
66 // == TEST ==
67 int main(){
68     ios :: sync_with_stdio(false);
69     cin.tie(nullptr);
70     int n, m;
71     cin >> n >> m;
72     for(int i = 1; i ≤ n; ++ i)
73         cin >> A[i];
74     sort(A + 1, A + 1 + n);
75     A[n + 1] = INT_MAX;
76     BLOCK :: build(n + 1, A + 1);
77     int last = 0;
78     int ans = 0;
79     // Do some op...
80     cout << ans << endl;
81     return 0;
82 }

```

2.8.2 莫队二次离线

```

1 #include "../header.cpp"
2 int n, m, k, maxt = 16383, X[MAXN], C[MAXN], t
;
3 int A[MAXN], bsize; i64 B[MAXN], R[MAXN];
4 struct Qry1{ int l, r, id; }O[MAXN];
5 struct Qry2{ int id, l, r; };
6 struct Qry3{ int id, l, r; };
7 bool cmp(Qry1 a, Qry1 b){
8     return a.l / bsize == b.l / bsize ? a.r < b.
        r : a.l < b.l;
9 }
10 vector <Qry2> P[MAXN];
11 vector <Qry3> Q[MAXN];
12 int main(){
13     n = qread(), m = qread(), k = qread(), bsize
        = sqrt(m + 1);
14     up(1, n, i) A[i] = qread();
15     up(1, m, i){
16         int l = qread(), r = qread(); O[i] = {l, r
            , i};
17     }
18     sort(O + 1, O + 1 + m, cmp);
19     int l = 1, r = 0;
20     up(1, m, i){
21         int p = O[i].l, q = O[i].r;
22         if(r < q){
23             P[r].push_back({ i, r + 1, q});
24             Q[l - 1].push_back({-i, r + 1, q});
25         }
26         if(r > q){

```

```

27     P[q].push_back({-i, q + 1, r});
28     Q[l - 1].push_back({ i, q + 1, r});
29     }
30     r = q;
31     if(l > p){
32         P[p].push_back({-i, p, l - 1});
33         Q[r].push_back({ i, p, l - 1});
34     }
35     if(l < p){
36         P[l].push_back({ i, l, p - 1});
37         Q[r].push_back({-i, l, p - 1});
38     }
39     l = p;
40 }
41 up(0, maxt, i) if(__builtin_popcount(i) == k
    ) X[++ t] = i;
42 up(0, n, i){
43     up(1, t, j) ++ C[A[i] ^ X[j]];
44     for(auto &o : P[i]){
45         if(o.id > 0) R[ o.id] += C[A[o.l]];
46         else R[-o.id] -= C[A[o.l]];
47         if(o.l < o.r)
48             P[i + 1].push_back({o.id, o.l + 1, o.r
                });
49     }
50     for(auto &o : Q[i]){
51         up(o.l, o.r, j){
52             if(o.id > 0) R[ o.id] += C[A[j]];
53             else R[-o.id] -= C[A[j]];
54         }
55     }
56     P[i].clear(), Q[i].clear();
57     P[i].shrink_to_fit();
58     Q[i].shrink_to_fit();
59 }
60 i64 ans = 0;
61 up(1, m, i){ ans += R[i], B[O[i].id] = ans;
    }
62 up(1, m, i) printf("%lld\n", B[i]);
63 return 0;
64 }

```

- $0 \times k$ 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 MAXH = 18 + 3;
5     int D[MAXN], F[MAXN];
6     int P[MAXN], Q[MAXN], o, h = 18;
7     void dfs(int u, int f){
8         ++ o;
9         P[u] = o, Q[o] = u;
10        F[u] = f, D[u] = D[f] + 1;
11        for(auto &v : E[u]) if(v ≠ f)
12            dfs(v, u);
13    }
14    int ST[MAXN][MAXH];
15    int cmp(int a, int b){
16        return D[a] < D[b] ? a : b;
17    }
18    int T[MAXN], n;
19    void init(int _n); // 初始化 ST 表
20    int lca(int a, int b){
21        if(a == b) return a;
22        int l = P[a], r = P[b];
23        if(l > r) swap(l, r);
24        ++ l;
25        int d = T[r - l + 1];
26        return F[cmp(ST[l][d], ST[r - (1 << d) +
            1][d])];
27    }
28    int dis(int a, int b);
29 }
30 namespace BIT{
31     void add(int D[], int n, int p, int w){
32         ++ p;
33         while(p ≤ n) D[p] += w, p += p & -p;
34     }
35     int pre(int D[], int n, int p){
36         if(p < 0) return 0;
37         p = min(n, p + 1);
38         int r = 0;
39         while(p > 0) r += D[p], p -= p & -p;
40         return r;
41     }
42 }
43 namespace PTree{
44     vector<int> EE[MAXN];
45     bool V[MAXN];
46     int S[MAXN], L[MAXN], *D1[MAXN], *D2[MAXN];
47     using LCA :: dis, BIT :: add, BIT :: pre;

```

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：

```

48 void dfs1(int s, int &g, int u, int f){
49     S[u] = 1;
50     int maxsize = 0;
51     for(auto &v : E[u]) if(v != f && !V[v]){
52         dfs1(s, g, v, u);
53         maxsize = max(maxsize, S[v]);
54         S[u] += S[v];
55     }
56     maxsize = max(maxsize, s - S[u]);
57     if(maxsize ≤ s / 2) g = u;
58 }
59 int n;
60 void build(int s, int &g, int u, int f){
61     dfs1(s, g, u, f);
62     V[g] = true, L[g] = s;
63     for(auto &u : E[g]) if(!V[u]){
64         int h = 0;
65         if(S[u] < S[g]) build(S[u], h, u, 0);
66         else build(s - S[g], h, u, 0);
67         EE[g].push_back(h);
68         EE[h].push_back(g);
69     }
70 }
71 int F[MAXN];
72 void dfs2(int u, int f){
73     F[u] = f;
74     for(auto &v : EE[u]) if(v != f){
75         dfs2(v, u);
76     }
77 }
78 void build(int _n){ // 建树 (需初始化 LCA)
79     n = _n;
80     int s = n, g = 0;
81     dfs1(s, g, 1, 0);
82     V[g] = true, L[g] = s;
83     for(auto &u : E[g]){
84         int h = 0;
85         if(S[u] < S[g]) build(S[u], h, u, 0);
86         else build(s - S[g], h, u, 0);
87         EE[g].push_back(h);
88         EE[h].push_back(g);
89     }
90     dfs2(g, 0);
91     for(int i = 1; i ≤ n; ++ i){
92         L[i] += 2;
93         D1[i] = new int[L[i] + 3];
94         D2[i] = new int[L[i] + 3];
95         for(int j = 0; j < L[i] + 3; ++ j)
96             D1[i][j] = D2[i][j] = 0;
97     }

```

```

98 }
99 void modify(int x, int w){ // 修改点权
100     int u = x;
101     while(1){
102         add(D1[x], L[x], dis(u, x), w);
103         int y = F[x];
104         if(y != 0){
105             int e = LCA :: dis(x, y);
106             add(D2[x], L[x], dis(u, y), w);
107             x = y;
108         } else break;
109     }
110 }
111 int query(int x, int d){
112     int ans = 0, u = x;
113     while(1){
114         ans += pre(D1[x], L[x], d - dis(u, x));
115         int y = F[x];
116         if(y != 0){
117             int e = dis(x, y);
118             ans -= pre(D2[x], L[x], d - dis(u, y));
119             x = y;
120         } else break;
121     }
122     return ans;
123 }
124 }

```

3.2 长链剖分

```

1 #include<bits/stdc++.h>
2 using namespace std;
3 using i64 = long long;
4 const int INF = 1e9;
5 const i64 INFL = 1e18;
6 const int MAXN = 5e5 + 3;
7 const int MAXM = 19 + 3;
8 vector<int> P[MAXN];
9 vector<int> Q[MAXN];
10 vector<int> E[MAXN];
11 int h = 19;
12 int L[MAXN], F[MAXN], G[MAXN], D[MAXN], S[MAXN]
    ][MAXN];
13 void dfs1(int u, int f){
14     L[u] = 1, S[0][u] = f;
15     F[u] = f, D[u] = D[f] + 1;
16     for(int i = 1; i ≤ h; ++ i)
17         S[i][u] = S[i - 1][S[i - 1][u]];
18     for(auto &v : E[u]) if(v != f){
19         dfs1(v, u);
20         if(L[v] > L[G[u]])

```

```

21             G[u] = v;
22             L[u] = max(L[u], L[v] + 1);
23     }
24 }
25 int T[MAXN];
26 void dfs2(int u, int f){
27     if(u == G[f]){
28         T[u] = T[f];
29         P[T[u]].push_back(u);
30         Q[T[u]].push_back(F[Q[T[u]].back()]);
31     } else {
32         T[u] = u;
33         P[u].push_back(u);
34         Q[u].push_back(u);
35     }
36     if(G[u]) dfs2(G[u], u);
37     for(auto &v : E[u]) if(v != f && v != G[u])
38         dfs2(v, u);
39 }
40 typedef unsigned int u32;
41 typedef unsigned long long u64;
42 int n, q; u32 s;
43 u32 get(u32 x) {
44     x ^= x << 13;
45     x ^= x >> 17;
46     x ^= x << 5;
47     return s = x;
48 }
49 int qread();
50 int H[MAXN];
51 int main(){
52     scanf("%d%d%u", &n, &q, &s);
53     int root = 0; H[0] = -1;
54     for(int i = 1; i ≤ n; ++ i){
55         int f = qread();
56         if(f == 0)
57             root = i;
58         else {
59             E[f].push_back(i);
60             E[i].push_back(f);
61         }
62         H[i] = H[i >> 1] + 1;
63     }
64     dfs1(root, 0);
65     dfs2(root, 0);
66     int lastans = 0;
67     i64 realans = 0;
68     for(int i = 1; i ≤ q; ++ i){
69         int x = (get(s) ^ lastans) % n + 1;
70         int k = (get(s) ^ lastans) % D[x];

```

```

71     if(k == 0){
72         lastans = x;
73     } else {
74         int h = H[k];
75         k -= 1 << h;
76         x = S[h][x];
77         int t = T[x];
78         k -= D[x] - D[t];
79         if(k > 0){
80             x = Q[t][k];
81         } else {
82             x = P[t][-k];
83         }
84         lastans = x;
85     }
86     realans ^= 1ll * i * lastans;
87 }
88 printf("%lld\n", realans);
89 return 0;
90 }

```

3.3 重链剖分

```

1 #include "../header.cpp"
2 int n, m, root, MOD, A[MAXN];
3 int qread();
4 vector<int> E[MAXN];
5 int S[MAXN], G[MAXN], D[MAXN], F[MAXN];
6 void dfs1(int u, int f){
7     S[u] = 1, G[u] = 0, D[u] = D[f] + 1, F[u] = f;
8     for(auto &v : E[u]) if(v != f){
9         dfs1(v, u);
10        S[u] += S[v];
11        if(S[v] > S[G[u]])
12            G[u] = v;
13    }
14 }
15 int B[MAXN];
16 int P[MAXN], Q[MAXN], T[MAXN], L[MAXN], R[MAXN], cnt;
17 void dfs2(int u, int f){
18     P[++cnt] = u, B[cnt] = A[u], Q[u] = cnt;
19     L[u] = cnt;
20     if(u != G[f]) T[u] = u;
21     else T[u] = T[f];
22     if(G[u]) dfs2(G[u], u);
23     for(auto &v : E[u]) if(v != f && v != G[u]){
24         dfs2(v, u);
25     }
26     R[u] = cnt;
27 }

```

```

28 namespace Seg{
29     const int SIZ = 4e5 + 3;
30     i64 S[SIZ], T[SIZ];
31     void pushup(int t, int a, int b);
32     void pushdown(int t, int a, int b);
33     void modify(int t, int a, int b, int l, int r, int w);
34     i64 query(int t, int a, int b, int l, int r);
35     void build(int t, int a, int b);
36 }
37 int main(){
38     n = qread(), m = qread(), root = qread(),
39     MOD = qread();
40     for(int i = 1; i <= n; ++i)
41         A[i] = qread();
42     for(int i = 2; i <= n; ++i){
43         int u = qread(), v = qread();
44         E[u].push_back(v);
45         E[v].push_back(u);
46     }
47     dfs1(root, 0);
48     Seg :: build(1, 1, n);
49     for(int i = 1; i <= m; ++i){
50         int op = qread();
51         if(op == 1){
52             int u = qread(), v = qread(), k = qread();
53             while(T[u] != T[v]){
54                 if(D[T[u]] < D[T[v]])
55                     swap(u, v);
56                 Seg :: modify(1, 1, n, Q[T[u]], Q[u], k);
57                 u = F[T[u]];
58             }
59             if(D[u] < D[v]) swap(u, v);
60             Seg :: modify(1, 1, n, Q[v], Q[u], k);
61         } else if(op == 2){
62             int u = qread(), v = qread();
63             i64 ans = 0;
64             while(T[u] != T[v]){
65                 if(D[T[u]] < D[T[v]])
66                     swap(u, v);
67                 ans = (ans + Seg :: query(1, 1, n, Q[T[u]], Q[u])) % MOD;
68                 u = F[T[u]];
69             }
70             if(D[u] < D[v]) swap(u, v);
71             ans = (ans + Seg :: query(1, 1, n, Q[v], Q[u])) % MOD;

```

```

72         printf("%lld\n", ans);
73     } else if(op == 3){
74         int x = qread(), w = qread();
75         Seg :: modify(1, 1, n, L[x], R[x], w);
76     } else {
77         int x = qread();
78         printf("%lld\n", Seg :: query(1, 1, n, L[x], R[x]));
79     }
80 }
81 return 0;
82 }

```

3.4 树哈希

3.4.1 用法

有根树求出每个子树的哈希值，儿子间顺序可交换。

```

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), H[u] += H[v];
9     H[u] = xor_shift(H[u]); // !important
10 }

```

3.5 tree-intersect

假设两条链 $(a_1, b_1), (a_2, b_2)$ 的 LCA 分别为 c_1, c_2 。再求出 $(a_1, a_2), (a_1, b_2), (b_1, a_2), (b_1, b_2)$ 的 LCA，记为 d_1, d_2, d_3, d_4 。将 (d_1, d_2, d_3, d_4) 按照深度从小到大排序， (c_1, c_2) 也从小到大排序。两条链有交当且仅当 $\text{dep}[c_1] \leq \text{dep}[d_1]$ 且 $\text{dep}[c_2] \leq \text{dep}[d_3]$ ，则 (d_3, d_4) 是链交的两个端点。

3.6 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             P[i + 1] = F[P[i]], i++;

```

```

10 }
11 return P;
12 }
13 vector<int> prufer2tree(int n){
14     vector<int> F(n);
15     for(int i = 1, j = 1; i ≤ n - 1; ++ i, ++ j){
16         while(D[j]) ++ j;
17         F[j] = P[i];
18         while(i ≤ n - 1 && !--D[P[i]] && P[i] < j)
19             F[P[i]] = P[i + 1], i ++;
20     }
21     return F;
22 }

```

3.7 虚树

```

1 #include "../header.cpp"
2 vector<pair<int, int> > E[MAXN];
3 namespace LCA{
4     const int SZ = 5e5 + 3;
5     int D[SZ], H[SZ], F[SZ], P[SZ], Q[SZ],
6         o;
7     void dfs(int u, int f){
8         P[u] = ++ o, Q[o] = u, F[u] = f, D[u] = D[
9             f] + 1;
10         for(auto &[v, w] : E[u]) if(v ≠ f){
11             H[v] = H[u] + w, dfs(v, u);
12         }
13     }
14     const int MAXH = 18 + 3;
15     int h = 18;
16     int ST[SZ][MAXH];
17     int cmp(int a, int b){
18         return D[a] < D[b] ? a : b;
19     }
20     int T[SZ], n;
21     void init(int _n, int root);
22     int lca(int a, int b);
23     int dis(int a, int b);
24 }
25 bool cmp(int a, int b){
26     return LCA :: P[a] < LCA :: P[b];
27 }
28 bool I[MAXN];
29 vector<int> E1[MAXN], V1;
30 void solve(vector<int> &V){
31     using LCA :: lca; using LCA :: D;
32     stack<int> S;
33     sort(V.begin(), V.end(), cmp);
34     S.push(1);
35     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 仙人掌

4.1.1 例题

给定一个仙人掌，多组询问 u, v 之间最短路长度。

```

1 #include "../header.cpp"
2 const int MAXD = 18 + 3;
3 struct edge{int u, v, w;};
4 vector<edge> V1[MAXN];
5 vector<edge> V2[MAXN];
6 vector<int> H[MAXN];
7 int n, D[MAXN], W[MAXN], F[MAXD][MAXN];
8 int o, X[MAXN], L[MAXN];
9 bool E[MAXN];
10 void dfs1(int u, int f){
11     D[u] = D[f] + 1, F[0][u] = f;
12     for(auto &e : V1[u]) if(e.v ≠ f){
13         if(D[e.v] && D[e.v] < D[u]){
14             int a = e.u;
15             int b = e.v;
16             int c = ++ o, t = c + n;
17             H[c].push_back(a);
18             L[c] = W[a] - W[b] + e.w;

```

```

19 while(a ≠ b)
20     E[a] = true, a = F[0][a], H[c].
21     push_back(a);
22     for(auto &x : H[c]){
23         int w = min(W[x] - W[b], L[c] - W[x] +
24             W[b]);
25         V2[x].push_back(edge{x, t, w});
26         V2[t].push_back(edge{t, x, w});
27     }
28     } else if(!D[e.v]){
29         W[e.v] = W[u] + e.w, dfs1(e.v, u);
30     }
31     for(auto &e : V1[u]) if(D[e.v] > D[u]){
32         if(!E[e.v]){
33             V2[e.u].push_back({e.u, e.v, e.w});
34             V2[e.v].push_back({e.v, e.u, e.w});
35         }
36     }
37     int d = 18;
38     void dfs2(int u, int f){
39         D[u] = D[f] + 1, F[0][u] = f;
40         up(1, d, i) F[i][u] = F[i - 1][F[i - 1][u]];
41         for(auto &e : V2[u]) if(e.v ≠ f){
42             X[e.v] = X[e.u] + e.w;
43             dfs2(e.v, u);
44         }
45     }
46     int lca(int u, int v){
47         if(D[u] < D[v]) swap(u, v);
48         dn(d, 0, i) if(D[F[i][u]] ≥ D[v]) u = F[i][
49             u];
50         if(u = v) return u;
51         dn(d, 0, i) if(F[i][u] ≠ F[i][v]) u = F[i][
52             u], v = F[i][v];
53         return F[0][u];
54     }
55     int jump(int u, int v){
56         dn(d, 0, i) if(D[F[i][v]] > D[u]) v = F[i][
57             v];
58         return v;
59     }
60     int dis(int x, int y){
61         int t = lca(x, y);
62         if(t > n){
63             int u = jump(t, x);
64             int v = jump(t, y);
65             int w = abs(W[u] - W[v]);
66             int l = min(w, L[t - n] - w);
67             return X[x] - X[u] + X[y] - X[v] + l;

```

```

65 } else {
66     return X[x] + X[y] - 2 * X[t];
67 }
68 }
69 int m, q;
70 int qread();
71 int main(){
72     n = qread(), m = qread(), q = qread();
73     up(1, m, i){
74         int u = qread(), v = qread(), w = qread();
75         V1[u].push_back(edge{u, v, w});
76         V1[v].push_back(edge{v, u, w});
77     }
78     dfs1(1, 0);
79     dfs2(1, 0);
80     up(1, q, i){
81         int u = qread(), v = qread();
82         printf("%d\n", dis(u, v));
83     }
84     return 0;
85 }

```

4.2 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。

```

1 #include "../header.cpp"
2 // 无向图
3 ll solve1(){
4     ll n, m; cin >> n >> m;
5     vector<pair<ll, ll>> Edges(m);
6     vector<vector<ll>> G(n + 2);
7     vector<ll> deg(n + 2);
8     for (auto &[i, j] : Edges)
9         cin >> i >> j, ++deg[i], ++deg[j];
10    for (auto [i, j] : Edges) {
11        if (deg[i] > deg[j] || (deg[i] == deg[j]
12            && i > j)) swap(i, j);
13        G[i].emplace_back(j);
14    }
15    vector<ll> val(n + 2);
16    ll ans = 0;
17    for (ll i = 1; i <= n; ++i) {

```

```

17         for (auto j : G[i]) ++val[j];
18         for (auto j : G[i])
19             for (auto k : G[j]) ans += val[k];
20             for (auto j : G[i]) val[j] = 0;
21     }
22     return ans;
23 }
24 // 有向图
25 ll solve2(){
26     ll n, m; cin >> n >> m;
27     vector<pair<ll, ll>> Edges(m);
28     vector<vector<pair<ll, ll>>> G(n + 2);
29     vector<ll> deg(n + 2);
30     for (auto &[i, j] : Edges)
31         cin >> i >> j, ++deg[i], ++deg[j];
32     for (auto [i, j] : Edges) {
33         ll flg = 0;
34         if (deg[i] > deg[j] || (deg[i] == deg[j]
35             && i > j)) swap(i, j), flg = 1;
36         G[i].emplace_back(j, flg);
37     }
38     vector<ll> in(n + 2), out(n + 2);
39     ll ans = 0;
40     for (ll i = 1; i <= n; ++i) {
41         for(auto [j, w] : G[i])
42             w ? (++in[j]) : (++out[j]);
43         for(auto [j, w1] : G[i])
44             for (auto [k, w2] : G[j]) {
45                 if (w1 == w2) ans += w1 ? in[k] :
46                     out[k];
47             }
48         for(auto [j, w] : G[i]) in[j] = out[j]
49             = 0;
50     }
51     return ans;
52 }

```

4.3 四元环计数

- 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。因为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 #include "../header.cpp"
2 ll solve(){
3     ll n, m; cin >> n >> m;

```

```

4     vector<pair<ll, ll>> Edges(m);
5     vector<vector<ll>> G(n + 2), iG(n + 2);
6     vector<ll> deg(n + 2);
7     for (auto &[i, j] : Edges)
8         cin >> i >> j, ++deg[i], ++deg[j];
9     for (auto [i, j] : Edges) {
10        if (deg[i] > deg[j] || (deg[i] == deg[j]
11            && i > j)) swap(i, j);
12        G[i].emplace_back(j), iG[j].emplace_back(i);
13    }
14    ll ans = 0;
15    vector<ll> v1(n + 2), v2(n + 2);
16    for (ll i = 1; i <= n; ++i) {
17        for (auto j : G[i])
18            for (auto k : G[j]) ++v1[k];
19        for (auto j : iG[i])
20            for (auto k : G[j])
21                ans += v1[k], ++v2[k];
22        for (auto j : G[i])
23            for (auto k : G[j])
24                ans += v1[k] * (v1[k] - 1) / 2, v1[k]
25                = 0;
26        for (auto j : iG[i]) for (auto k : G[j]) {
27            if (deg[k] > deg[i] || (deg[k] == deg[i]
28                && k > i)) ans += v2[k] * (v2[k] - 1)
29                / 2;
30            v2[k] = 0;
31        }
32    }
33    return ans;
34 }

```

4.4 支配树

```

1 // From Alex_Wei
2 #include "../header.cpp"
3 int n, m, dn, F[MAXN], ind[MAXN], dfn[MAXN];
4 int sdom[MAXN], idom[MAXN], sz[MAXN];
5 vector<int> E[MAXN], rE[MAXN], buc[MAXN];
6 void dfs(int u, int f) {
7     ind[dfn[u] = ++dn] = u, F[u] = f;
8     for(auto &v: E[u]) if(!dfn[v]) dfs(v, u);
9 }
10 struct dsu {
11     // M 维护 sdom 最小的点的编号
12     int F[MAXN], M[MAXN];
13     int find(int x) {
14         if(F[x] == x) return F[x];
15         int f = F[x];
16         F[x] = find(f);

```

```

17     if(sdom[M[f]] < sdom[M[x]]) M[x] = M[f];
18     return F[x];
19 }
20 int get(int x) { return find(x), M[x]; }
21 } tr;
22 int main() {
23     cin >> n >> m;
24     for(int i = 1; i ≤ m; i++) {
25         int u, v; cin >> u >> v;
26         E[u].push_back(v), rE[v].push_back(u);
27     }
28     dfs(1, 0), sdom[0] = n + 1;
29     for(int i = 1; i ≤ n; i++) tr.F[i] = i;
30     for(int i = n; i; -- i){
31         int u = ind[i];
32         for(auto &v: buc[i]) idom[v] = tr.get(v);
33         if(i == 1) break;
34         sdom[u] = i;
35         for(auto &v: rE[u]) {
36             sdom[u] = min(sdom[u], dfn[v] ≤ i ? dfn
37                           [v] : sdom[tr.get(v)]);
38         }
39         tr.M[u] = u, tr.F[u] = F[u];
40         buc[sdom[u]].push_back(u);
41     }
42     for(int i = 2; i ≤ n; i++) {
43         int u = ind[i];
44         if(sdom[idom[u]] ≠ sdom[u])
45             idom[u] = idom[idom[u]];
46         else idom[u] = sdom[u];
47     }
48     for(int i = n; i; -- i){
49         sz[i] += 1;
50         if(i > 1) sz[ind[idom[i]]] += sz[i];
51     }
52     for(int i = 1; i ≤ n; ++ i){
53         cout << sz[i] << "\n"[i == n];
54     }
55     return 0;

```

4.5 基环树

```

1 #include "../header.cpp"
2 using edge = tuple<int, int, int>;
3 vector<edge> E[MAXN];
4 vector<edge> W;
5 vector<int> C;
6 edge F[MAXN];
7 bool V[MAXN];
8 int I[MAXN], o;

```

```

9 void dfs0(int u, int e){
10     V[u] = true;
11     I[u] = ++ o;
12     for(auto &[i, v, w] : E[u]) if(i ≠ e){
13         if(V[v]){
14             if(I[v] < I[u]){
15                 for(int p = u; p ≠ v;){
16                     auto &[j, f, x] = F[p];
17                     C.push_back(p);
18                     W.push_back({j, p, x});
19                     p = f;
20                 }
21                 C.push_back(v);
22                 W.push_back({i, v, w});
23             }
24         } else {
25             F[v] = {i, u, w};
26             dfs0(v, i);
27         }
28     }
29 }
30 namespace Problem2{
31 // === 删除环上第 i 条边, 求直径 ===
32 i64 H[MAXN], A1[MAXN], B1[MAXN], A2[MAXN],
33     B2[MAXN], A3[MAXN], B3[MAXN];
34 i64 L[MAXN];
35 i64 dis = 0;
36 void dfs1(int u, int e){
37     for(auto &[i, v, w] : E[u]) if(i ≠ e){
38         if(!V[v]){
39             dfs1(v, i);
40             dis = max(dis, L[u] + w + L[v]);
41             L[u] = max(L[u], L[v] + w);
42         }
43     }
44 }
45 int main(){
46     int n;
47     cin >> n;
48     for(int i = 1; i ≤ n; ++ i){
49         int u, v, w;
50         cin >> u >> v >> w;
51         E[u].push_back({i, v, w});
52         E[v].push_back({i, u, w});
53     }
54     dfs0(1, 0);
55     memset(V, 0, sizeof(V));
56     for(auto &u : C)
57         V[u] = true;
58     for(auto &u : C){
59         dfs1(u, 0);

```

```

59     }
60     int l = 0, r = C.size() - 1;
61     for(int i = l; i ≤ r; ++ i){
62         int x = C[i];
63         if(i > 0)
64             H[i] = H[i - 1] + get<2>(W[i - 1]);
65         A1[i] = L[x] + H[i];
66         B1[i] = L[x] - H[i];
67         A2[i] = L[x] - H[i];
68         B2[i] = L[x] + H[i];
69     }
70     i64 h = H[r] + get<2>(W.back());
71     for(int i = l; i ≤ r; ++ i)
72         A1[i] = max(i == l ? -INFL : A1[i - 1],
73                     L[C[i]] + H[i]);
74     A2[i] = max(i == l ? -INFL : A2[i - 1],
75                 L[C[i]] - H[i]);
76     for(int i = r; i ≥ l; -- i)
77         B1[i] = max(i == r ? -INFL : B1[i + 1],
78                     L[C[i]] - H[i]);
79     B2[i] = max(i == r ? -INFL : B2[i + 1],
80                 L[C[i]] + H[i]);
81     A3[l] = -INFL, B3[r] = -INFL;
82     for(int i = l + 1; i ≤ r; ++ i){
83         int x = C[i];
84         i64 w = A2[i - 1] + L[x] + H[i];
85         A3[i] = max(A3[i - 1], w);
86     }
87     for(int i = r - 1; i ≥ l; -- i){
88         int x = C[i];
89         i64 w = B2[i + 1] + L[x] - H[i];
90         B3[i] = max(B3[i + 1], w);
91     }
92     i64 t = -INFL;
93     for(int i = l; i < r; ++ i){
94         i64 d = A1[i] + B1[i + 1] + h;
95         i64 g = A2[i] + B2[i + 1] + 0;
96         d = max({d, dis, A3[i], B3[i + 1]});
97         t = min(t, d);
98     }
99     t = min(t, max(A3[r], dis));
100     if(t % 2 == 0)
101         cout << t / 2 << ".0" << endl;
102     if(t % 2 == 1)
103         cout << t / 2 << ".5" << endl;
104     return 0;
105 }
106 namespace Problem3{
107 // === 求最大点权独立集 ===
108 int A[MAXN];

```

```

106 i64 X[MAXN], Y[MAXN];
107 i64 P[MAXN][2], Q[MAXN][2];
108 void dfs1(int u, int e){
109     for(auto &i, v, w : E[u]) if(i != e){
110         if(!V[v]){
111             dfs1(v, i);
112             Y[u] += max(X[v], Y[v]);
113             X[u] += Y[v];
114         }
115     }
116     X[u] += A[u];
117 }
118 int main(){
119     int n;
120     cin >> n;
121     for(int i = 1; i ≤ n; ++ i){
122         cin >> A[i];
123     }
124     for(int i = 1; i ≤ n; ++ i){
125         int u, v;
126         cin >> u >> v;
127         ++ u, ++ v;
128         E[u].push_back({i, v, 0});
129         E[v].push_back({i, u, 0});
130     }
131     double p;
132     cin >> p;
133     dfs0(1, 0);
134     memset(V, 0, sizeof(V));
135     for(auto &u : C)
136         V[u] = true;
137     for(auto &u : C){
138         dfs1(u, 0);
139     }
140     int l = 0, r = C.size() - 1;
141     P[0][1] = X[C[0]];
142     P[0][0] = -INFL;
143     Q[0][0] = Y[C[0]];
144     Q[0][1] = -INFL;
145     for(int i = l + 1; i ≤ r; ++ i){
146         int x = C[i];
147         P[i][1] = X[x] + P[i - 1][0];
148         P[i][0] = Y[x] + max(P[i - 1][0], P[i - 1][1]);
149         Q[i][1] = X[x] + Q[i - 1][0];
150         Q[i][0] = Y[x] + max(Q[i - 1][0], Q[i - 1][1]);
151     }
152     i64 ans = max({P[r][0], Q[r][0], Q[r][1]});
153     cout << fixed << setprecision(1) << ans *

```

```

        p << endl;
        return 0;
    }
}
157 int main(){
158     return Problem3 :: main();
159 }

```

4.6 二分图最大匹配

```

1 #include "../header.cpp"
2 vector <int> G[MAXN];
3 bool V[MAXN];
4 int ML[MAXN], MR[MAXN];
5 bool kuhn(int u){
6     V[u] = true;
7     for(auto &v: G[u]) if(MR[v] == 0){
8         ML[u] = v, MR[v] = u;
9         return true;
10    }
11    for(auto &v: G[u]) if(!V[MR[v]] && kuhn(MR[v])){
12        ML[u] = v, MR[v] = u;
13        return true;
14    }
15    return false;
16 }
17 void solve(int L, int R){
18     for(int i = 1; i ≤ L; ++ i){
19         shuffle(G[i].begin(), G[i].end(), MT);
20     } // 需要打乱避免构造
21     while(1){
22         bool ok = false;
23         memset(V, false, sizeof(V));
24         for(int i = 1; i ≤ L; ++ i)
25             ok |= !ML[i] && kuhn(i);
26         if(!ok) break;
27     }
28 }

```

4.7 一般图最大匹配

```

1 #include "../header.cpp"
2 int n;
3 vector <int> E[MAXN];
4 queue <int> Q;
5 int vis[MAXN], F[MAXN], col[MAXN], pre[MAXN],
6   mat[MAXN], tmp;
7 int getfa(int x){
8     return x == F[x] ? x : F[x] = getfa(F[x]);
9 }

```

```

9 int lca(int x, int y){
10     for(++ tmp; x = pre[mat[x]], swap(x, y))
11         if(vis[x = getfa(x)] == tmp) return x;
12         else vis[x] = x ? tmp : 0;
13 }
14 void flower(int x, int y, int z){
15     while(getfa(x) != z){
16         pre[x] = y, y = mat[x], F[x] = F[y] = z;
17         x = pre[y];
18         if(col[y] == 2)
19             Q.push(y), col[y] = 1;
20     }
21 }
22 bool aug(int u){
23     for(int i = 1; i ≤ n; ++ i)
24         col[i] = pre[i] = 0, F[i] = i;
25     Q = queue<int>({u}), col[u] = 1;
26     while(!Q.empty()){
27         auto x = Q.front(); Q.pop();
28         for(auto &v: E[x]){
29             int y = v, z;
30             if(col[y] == 2) continue;
31             if(col[y] == 1){
32                 z = lca(x, y);
33                 flower(x, y, z), flower(y, x, z);
34             } else
35                 if(!mat[y]){
36                     for(pre[y] = x; y;){
37                         mat[y] = x = pre[y], swap(y, mat[x]);
38                     }
39                     return true;
40                 } else {
41                     pre[y] = x, col[y] = 2;
42                     Q.push(mat[y]), col[mat[y]] = 1;
43                 }
44             }
45     }
46     return false;
47 }

```

4.8 2-SAT

4.8.1 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 "tarjan-scc.cpp"
2 const int MAXN = 1e6 + 3;
3 int X[MAXN][2], o;
4 int main(){
5     ios :: sync_with_stdio(false);

```

```

6  int n, m;
7  cin >> n >> m;
8  for(int i = 1; i ≤ n; ++ i)
9      X[i][0] = ++ o, X[i][1] = ++ o;
10 for(int i = 1; i ≤ m; ++ i){
11     int a, x, b, y;
12     cin >> a >> x >> b >> y;
13     SCC :: add(X[a][!x], X[b][y]);
14     SCC :: add(X[b][!y], X[a][x]);
15 }
16 for(int i = 1; i ≤ o; ++ i)
17     if(!SCC :: F[i])
18         SCC :: dfs(i);
19 bool ok = true;
20 for(int i = 1; i ≤ n; ++ i){
21     if(SCC :: C[X[i][0]] = SCC :: C[X[i][1]])
22         ok = false;
23 }
24 if(ok){
25     cout << "POSSIBLE" << endl;
26     for(int i = 1; i ≤ n; ++ i){
27         int a = SCC :: C[X[i][0]];
28         int b = SCC :: C[X[i][1]];
29         cout << (a ≥ b) << "u";
30     }
31     cout << endl;
32 } else {
33     cout << "IMPOSSIBLE" << endl;
34 }
35 return 0;
36 }

```

4.9 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  // 对每个连通块调用 dfs(i, i)
6  void dfs(int u, int g){
7      L[u] = D[u] = ++ o, F[u] = true; int s = 0;
8      for(auto &v : V[u]){
9          if(!F[v]){
10             dfs(v, g), ++ s;
11             L[u] = min(L[u], L[v]);
12             if(u ≠ g && L[v] ≥ D[u]) C[u] = true;
13             } else {
14                 L[u] = min(L[u], D[v]);
15             }
16     }
17     // C[u] 为真表示该点是割点

```

```

18     if(u = g && s > 1) C[u] = true;
19 }

```

4.10 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; bool I[MAXN];
6  void dfs(int u, int l){
7      D[u] = L[u] = ++ o; I[u] = true, S.push(u);
8      int s = 0;
9      for(auto &v, g : V[u]) if(g ≠ l) {
10         if(D[v]){
11             if(I[v]) L[u] = min(L[u], D[v]);
12             } else {
13                 dfs(v, g), L[u] = min(L[u], L[v]), ++ s;
14             }
15     }
16     if(D[u] = L[u]){
17         vector<int> T;
18         while(S.top() ≠ u){
19             int v = S.top(); S.pop();
20             T.push_back(v), I[v] = false;
21         }
22         T.push_back(u), S.pop(), I[u] = false;
23         A.push_back(T);
24     }
25 }

```

4.11 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             } else {
13                 dfs(v, u), L[u] = min(L[u], L[v]), ++ s;
14             }
15         if(L[v] ≥ D[u]){
16             vector<int> T;
17             while(S.top() ≠ v){
18                 int t = S.top(); S.pop();
19                 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 }
26 if(f = 0 && s = 0)
27     A.push_back({u}); // 孤立点特判
28 }

```

4.12 强连通分量

```

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

```

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){

```

```

5   V[++ o] = v, N[o] = H[u], H[u] = o, F[o] =
6   f, W[o] = c;
7   V[++ o] = u, N[o] = H[v], H[v] = o, F[o] =
8   0, W[o] = -c;
9   n = max({n, u, v});
10  }
11  void clear(){
12      for(int i = 1; i ≤ n; ++ i) H[i] = 0;
13      n = 0, o = 1;
14  }
15  bool I[MAXN]; i64 D[MAXN];
16  bool spfa(int s, int t){
17      queue<int> Q;
18      Q.push(s), I[s] = true;
19      for(int i = 1; i ≤ n; ++ i)
20          D[i] = INFL;
21      D[s] = 0;
22      while(!Q.empty()){
23          int u = Q.front(); Q.pop(), I[u] = false;
24          for(int i = H[u]; i; i = N[i]){
25              int &v = V[i], &f = F[i], &w = W[i];
26              if(f && D[u] + w < D[v]){
27                  D[v] = D[u] + w;
28                  if(!I[v]) Q.push(v), I[v] = true;
29              }
30          }
31      }
32      return D[t] ≠ INFL;
33  }
34  int C[MAXN]; bool T[MAXN];
35  pair<i64, i64> dfs(int s, int t, int u, i64
36  maxf){
37      if(u = t)
38          return make_pair(maxf, 0);
39      i64 totf = 0, totc = 0;
40      T[u] = true;
41      for(int &i = C[u]; i; i = N[i]){
42          int &v = V[i], &f = F[i], &w = W[i];
43          if(f && D[v] = D[u] + w && !T[v]){
44              auto [f, c] = dfs(s, t, v, min(1ll * F
45              [i], maxf));
46              F[i] -= f, F[i ^ 1] += f;
47              totf += f, maxf -= f;
48              totc += 1ll * f * W[i] + c;
49              if(maxf == 0){
50                  T[u] = false;
51                  return make_pair(totf, totc);
52              }
53          }
54      }
55      T[u] = false;

```

```

52      return make_pair(totf, totc);
53  }
54  pair<i64, i64> mcmf(int s, int t){
55      i64 ans1 = 0, ans2 = 0;
56      while(spfa(s, t)){
57          memcpy(C, H, sizeof(int) * (n + 3));
58          auto [f, c] = dfs(s, t, s, INFL);
59          ans1 += f, ans2 += c;
60      }
61      return make_pair(ans1, ans2);
62  }
63  }

```

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 i64 INF = 1e18;
4      const int SIZ = 1e5 + 3;
5      int n, m;
6      int H[SIZ], V[SIZ], N[SIZ], F[SIZ], 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) H[i] = 0;
18         n = m = 0, t = 1;
19     }
20     int D[SIZ];
21     bool bfs(int s, int t){
22         queue<int> Q;
23         for(int i = 1; i ≤ n; ++ i) 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], &f = F[i];
29                 if(f ≠ 0 && !D[v]){
30                     D[v] = D[u] + 1, Q.push(v);
31                 }
32             }
33         }
34     }

```

```

31     return D[t] ≠ 0;
32 }
33 int C[SIZ];
34 i64 dfs(int s, int t, int u, i64 maxf){
35     if(u = t)
36         return maxf;
37     i64 totf = 0;
38     for(int &i = C[u]; i; i = N[i]){
39         const int &v = V[i];
40         const int &f = F[i];
41         if(D[v] = D[u] + 1){
42             i64 ff = dfs(s, t, v, min(maxf, 1ll *
43             f));
44             totf += ff, maxf -= ff;
45             F[i] -= ff, F[i ^ 1] += ff;
46             if(maxf == 0) return totf;
47         }
48     }
49     return totf;
50 }
51 i64 dinic(int s, int t){
52     i64 ans = 0;
53     while(bfs(s, t)){
54         memcpy(C, H, sizeof(int) * (n + 3));
55         ans += dfs(s, t, s, INF);
56     }
57     return ans;
58 }
59 namespace GHTree{
60     const int INF = 1e9;
61     int n, m, U[MAXM], V[MAXM], W[MAXM], A[MAXM],
62     B[MAXM];
63     void add(int u, int v, int w){
64         ++ m;
65         U[m] = u, V[m] = v, W[m] = w;
66         A[m] = Dinic :: add(u, v, w);
67         B[m] = Dinic :: add(v, u, w);
68         n = max({n, u, v});
69     }
70     vector<pair<int, int>> E[MAXN];
71     void build(vector<int> N){
72         int s = N.front(), t = N.back();
73         if(s = t) return;
74         for(int i = 1; i ≤ m; ++ i){
75             int a = A[i]; Dinic :: F[a] = W[i],
76             Dinic :: F[a ^ 1] = 0;
77             int b = B[i]; Dinic :: F[b] = W[i],
78             Dinic :: F[b ^ 1] = 0;
79         }
80         int w = Dinic :: dinic(s, t);

```

```

78 E[s].push_back(make_pair(t, w));
79 E[t].push_back(make_pair(s, w));
80 vector<int> P, Q;
81 for(auto &u : N){
82     if(Dinic :: D[u] ≠ 0)
83         P.push_back(u);
84     else
85         Q.push_back(u);
86 }
87 build(P), build(Q);
88 }
89 int D[MAXN];
90 int cut(int s, int t){
91     queue<int> Q; Q.push(s);
92     for(int i = 1; i ≤ n; ++ i)
93         D[i] = -1;
94     D[s] = INF;
95     while(!Q.empty()){
96         int u = Q.front(); Q.pop();
97         for(auto &[v, w] : E[u]){
98             if(D[v] = -1){
99                 D[v] = min(D[u], w);
100                 Q.push(v);
101             }
102         }
103     }
104     return D[t];
105 }
106 }

```

5.3 最大流

```

1 #include "../header.cpp"
2 namespace Dinic{
3     const i64 INF = 1e18;
4     int n, H[MAXN], V[MAXM], N[MAXM], F[MAXM], t = 1;
5     void add(int u, int v, int f){
6         V[++ t] = v, N[t] = H[u], F[t] = f, H[u] = t;
7         V[++ t] = u, N[t] = H[v], F[t] = 0, H[v] = t;
8         n = max({n, u, v});
9     }
10    void clear(){
11        for(int i = 1; i ≤ n; ++ i)
12            H[i] = 0;
13        n = 0, t = 1;
14    }
15    i64 D[MAXN];
16    bool bfs(int s, int t){

```

```

17    queue<int> Q;
18    for(int i = 1; i ≤ n; ++ i) D[i] = 0;
19    Q.push(s), D[s] = 1;
20    while(!Q.empty()){
21        int u = Q.front(); Q.pop();
22        for(int i = H[u]; i ≤ N[i]; ++ i){
23            const int &v = V[i], &f = F[i];
24            if(f ≠ 0 && !D[v]){
25                D[v] = D[u] + 1, Q.push(v);
26            }
27        }
28        return D[t] ≠ 0;
29    }
30    int C[MAXN];
31    i64 dfs(int s, int t, int u, i64 maxf){
32        if(u = t) return maxf;
33        i64 totf = 0;
34        for(int &i = C[u]; i ≤ N[i]; ++ i){
35            const int &v = V[i], &f = F[i];
36            if(f && D[v] = D[u] + 1){
37                i64 f = dfs(s, t, v, min(1ll * f, maxf));
38                F[i] -= f, F[i ^ 1] += f, totf += f, maxf -= f;
39                if(maxf = 0) return totf;
40            }
41        }
42        return totf;
43    }
44    i64 dinic(int s, int t){
45        i64 ans = 0;
46        while(bfs(s, t)){
47            memcpy(C, H, sizeof(int) * (n + 3));
48            ans += dfs(s, t, s, INFL);
49        }
50        return ans;
51    }
52 }
53 }

```

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。

```
1 #define add add0
```

```

2 #include "flow-cost.cpp"
3 #undef add
4 namespace MCMF{
5     i64 cost0; int G[MAXN];
6     void add(int u, int v, int l, int r, int c){
7         G[v] += l, G[u] -= l;
8         cost0 += 1ll * l * c;
9         add0(u, v, r - l, c);
10    }
11    i64 solve(){
12        int s = ++ n, t = ++ n;
13        i64 sum = 0;
14        for(int i = 1; i ≤ n - 2; ++ i){
15            if(G[i] < 0)
16                add0(i, t, -G[i], 0);
17            else
18                add0(s, i, G[i], 0), sum += G[i];
19        }
20        auto res = mcmf(s, t);
21        if(res.first ≠ sum)
22            return -1;
23        return res.second + cost0;
24    }
25    i64 solve(int s0, int t0){
26        add0(t0, s0, INF, 0);
27        int s = ++ n;
28        int t = ++ n;
29        i64 sum = 0;
30        for(int i = 1; i ≤ n - 2; ++ i){
31            if(G[i] < 0)
32                add0(i, t, -G[i], 0);
33            else
34                add0(s, i, G[i], 0), sum += G[i];
35        }
36        auto res = mcmf(s, t);
37        if(res.first ≠ sum)
38            return -1;
39        return res.second + cost0;
40    }
41 }

```

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-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, G[u] -= l;
8         add0(u, v, r - l);
9     }
10    void clear(){
11        for(int i = 1; i ≤ t; ++ i){
12            N[i] = F[i] = V[i] = 0;
13        }
14        for(int i = 1; i ≤ n; ++ i){
15            H[i] = G[i] = C[i] = 0;
16        }
17        t = 1, n = 0;
18    }
19    bool solve(){ // 为真表示有解
20        int s = ++ n, t = ++ n;
21        i64 sum = 0;
22        for(int i = 1; i ≤ n - 2; ++ i){
23            if(G[i] < 0)
24                add0(i, t, -G[i]);
25            else
26                add0(s, i, G[i]), sum += G[i];
27        }
28        return sum ≠ dinic(s, t);
29    }
30    i64 solve(int s0, int t0){
31        add0(t0, s0, INF);
32        int s = ++ n, t = ++ n;
33        i64 sum = 0;
34        for(int i = 1; i ≤ n - 2; ++ i){
35            if(G[i] < 0)
36                add0(i, t, -G[i]);
37            else
38                add0(s, i, G[i]), sum += G[i];
39        }
40        return dinic(s, t) = sum ? dinic(s0, t0)
41            : -1;
42    }
}

```

```

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             f = j; break;
18         }
19         if(f == -1) return 0;
20         if(f ≠ i){
21             for(int j = 1; j ≤ n; ++ j)
22                 swap(a.W[i][j], a.W[f][j]);
23             ans = MOD - ans;
24         }
25         for(int j = i + 1; j ≤ n; ++ j) if(a.W[j][i]
26             ){
27             while(a.W[j][i]){
28                 int u = a.W[i][i], v = a.W[j][i];
29                 if(u > v){
30                     for(int k = 1; k ≤ n; ++ k)
31                         swap(a.W[i][k], a.W[j][k]);
32                     ans = MOD - ans, swap(u, v);
33                 }
34                 int rate = v / u;
35                 for(int k = 1; k ≤ n; ++ k){
36                     a.W[j][k] = (a.W[j][k] - 1ll * rate
37                         * a.W[i][k] % MOD + MOD) % MOD;
38                 }
39             }
40         }
41     }
42     for(int i = 1; i ≤ n; ++ i)
43         ans = 1ll * ans * a.W[i][i] % MOD;
44     return ans;
45 }

```

```

7     n = _n;
8     m = _m;
9     for(int i = 1; i ≤ n; ++ i)
10        for(int j = 1; j ≤ m; ++ j)
11            W[i][j] = 0;
12 }
13 };
14 bool zero(double f){
15     return fabs(f) < EPS;
16 }
17 int mat_rank(Mat &a){
18     const int &n = a.n;
19     const int &m = a.m;
20     int cnt = 0;
21     for(int i = 1; i ≤ m; ++ i){
22         int p = cnt + 1;
23         int f = -1;
24         for(int j = p; j ≤ n; ++ j){
25             if(!zero(a.W[j][i])){
26                 f = j;
27                 break;
28             }
29         }
30         if(f == -1)
31             continue;
32         if(f ≠ p){
33             for(int j = 1; j ≤ m; ++ j)
34                 swap(a.W[p][j], a.W[f][j]);
35         }
36         ++ cnt;
37         for(int j = p + 1; j ≤ n; ++ j){
38             double rate = a.W[j][i] / a.W[p][i];
39             for(int k = 1; k ≤ m; ++ k){
40                 a.W[j][k] -= rate * a.W[p][k];
41             }
42         }
43     }
44     return cnt;
45 }
46 double X[MAXN];
47 int main(){
48     int n;
49     cin >> n;
50     Mat A(n, n);
51     Mat T(n, n + 1);
52     for(int i = 1; i ≤ n; ++ i){
53         for(int j = 1; j ≤ n; ++ j)
54             cin >> A.W[i][j];
55         for(int j = 1; j ≤ n; ++ j)
56             T.W[i][j] = A.W[i][j];
57         cin >> T.W[i][n + 1];

```

6 数学

6.1 线性代数

6.1.1 行列式

```

1 #include "../..header.cpp"
2 struct Mat{

```

6.1.2 高斯消元与求秩 (实数)

```

1 #include "../..header.cpp"
2 const double EPS = 1e-9;
3 struct Mat{
4     int n, m;
5     double W[MAXN][MAXN];
6     Mat(int _n = 0, int _m = 0){

```

```

58 }
59 int res1 = mat_rank(A);
60 int res2 = mat_rank(T);
61 if(res1 != res2)
62     cout << -1 << endl;
63 else
64 if(res2 < n)
65     cout << 0 << endl;
66 else {
67     for(int i = n; i ≥ 1; -- i){
68         X[i] = T.W[i][n + 1] / T.W[i][i];
69         for(int j = i - 1; j ≥ 1; -- j){
70             double rate = T.W[j][i] / T.W[i][i];
71             T.W[j][i] -= rate * T.W[i][i];
72             T.W[j][n + 1] -= rate * T.W[i][n + 1];
73         }
74     }
75     for(int i = 1; i ≤ n; ++ i)
76         cout << "x" << i << "=" << fixed <<
77         setprecision(2) << X[i] << endl;
78     return 0;
79 }

```

6.1.3 高斯消元与求秩 (整数)

```

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 power(int a, int b){
14     int r = 1;
15     while(b){
16         if(b & 1) r = 1ll * r * a % MOD;
17         b >>= 1, a = 1ll * a * a % MOD;
18     }
19     return r;
20 }
21 int inv(int x){
22     return power(x, MOD - 2);
23 }
24 int mat_rank(Mat &a){
25     const int &n = a.n;
26     const int &m = a.m;

```

```

27 int cnt = 0;
28 for(int i = 1; i ≤ m; ++ i){
29     int p = cnt + 1;
30     int f = -1;
31     for(int j = p; j ≤ n; ++ j){
32         if(a.W[j][i] != 0){
33             f = j;
34             break;
35         }
36     }
37     if(f == -1)
38         continue;
39     if(f != p){
40         for(int j = 1; j ≤ m; ++ j)
41             swap(a.W[p][j], a.W[f][j]);
42     }
43     ++ cnt;
44     int invp = inv(a.W[p][i]);
45     for(int j = p + 1; j ≤ n; ++ j){
46         int rate = 1ll * a.W[j][i] * invp % MOD;
47         for(int k = 1; k ≤ m; ++ k){
48             a.W[j][k] = (a.W[j][k] - 1ll * rate *
49                 a.W[p][k] % MOD + MOD) % MOD;
50         }
51     }
52     return cnt;
53 }
54 int X[MAXN];
55 int main(){
56     int n;
57     cin >> n;
58     Mat A(n, n);
59     Mat T(n, n + 1);
60     for(int i = 1; i ≤ n; ++ i){
61         for(int j = 1; j ≤ n; ++ j)
62             cin >> A.W[i][j];
63         for(int j = 1; j ≤ n; ++ j)
64             T.W[i][j] = A.W[i][j];
65         cin >> T.W[i][n + 1];
66     }
67     int res1 = mat_rank(A);
68     int res2 = mat_rank(T);
69     if(res1 != res2)
70         cout << -1 << endl;
71     else
72     if(res2 < n)
73         cout << 0 << endl;
74     else {
75         for(int i = n; i ≥ 1; -- i){
76             int invp = inv(T.W[i][i]);

```

```

77         X[i] = 1ll * T.W[i][n + 1] * invp % MOD;
78         for(int j = i - 1; j ≥ 1; -- j){
79             int rate = 1ll * T.W[j][i] * invp %
80                 MOD;
81             T.W[j][i] = (T.W[j][i] - 1ll *
82                 rate * T.W[i][i] % MOD + MOD) %
83                 MOD;
84             T.W[j][n + 1] = (T.W[j][n + 1] - 1ll *
85                 rate * T.W[i][n + 1] % MOD + MOD) %
86                 MOD;
87         }
88     }
89     for(int i = 1; i ≤ n; ++ i)
90         cout << "x" << i << "=" << X[i] << endl;
91     return 0;
92 }

```

6.1.4 矩阵求逆

```

1 #include <bits/stdc++.h>
2 using namespace std;
3 using i64 = long long;
4 const int INF = 1e9;
5 const i64 INFL = 1e18;
6 const int MAXN = 400 + 3;
7 const int MOD = 1e9 + 7;
8 struct Mat{
9     int n, m;
10     int W[MAXN][MAXN];
11     Mat(int _n = 0, int _m = 0){
12         n = _n, m = _m;
13         for(int i = 1; i ≤ n; ++ i)
14             for(int j = 1; j ≤ m; ++ j)
15                 W[i][j] = 0;
16     }
17 };
18 int power(int a, int b){
19     int r = 1;
20     while(b){
21         if(b & 1) r = 1ll * r * a % MOD;
22         b >>= 1, a = 1ll * a * a % MOD;
23     }
24     return r;
25 }
26 int inv(int x){
27     return power(x, MOD - 2);
28 }
29 bool mat_inv(Mat &a){
30     const int &n = a.n;
31     Mat b(n, n);
32     for(int i = 1; i ≤ n; ++ i)

```

```

33     b.W[i][i] = 1;
34     for(int i = 1; i ≤ n; ++ i){
35         int f = -1;
36         for(int j = i; j ≤ n; ++ j) if(a.W[j][i]
37             ] ≠ 0){
38             f = j;
39             break;
40         }
41         if(f == -1){
42             return false;
43         }
44         if(f ≠ i){
45             for(int j = 1; j ≤ n; ++ j)
46                 swap(a.W[i][j], a.W[f][j]),
47                 swap(b.W[i][j], b.W[f][j]);
48         }
49         int invp = inv(a.W[i][i]);
50         for(int j = i + 1; j ≤ n; ++ j){
51             int rate = 1ll * a.W[j][i] * invp
52                 % MOD;
53             for(int k = 1; k ≤ n; ++ k){
54                 a.W[j][k] = (a.W[j][k] - 1ll *
55                     rate * a.W[i][k] % MOD +
56                     MOD) % MOD;
57                 b.W[j][k] = (b.W[j][k] - 1ll *
58                     rate * b.W[i][k] % MOD +
59                     MOD) % MOD;
60             }
61         }
62     }
63     for(int i = n; i ≥ 1; -- i){
64         int invp = inv(a.W[i][i]);
65         for(int j = 1; j ≤ n; ++ j){
66             a.W[i][j] = 1ll * a.W[i][j] * invp
67                 % MOD;
68             b.W[i][j] = 1ll * b.W[i][j] * invp
69                 % MOD;
70         }
71     }
72     for(int i = 1; i ≤ n; ++ i)

```

```

72         for(int j = 1; j ≤ n; ++ j)
73             a.W[i][j] = b.W[i][j];
74         return true;
75     }
76     int X[MAXN];
77     int main(){
78         int n;
79         cin >> n;
80         Mat A(n, n);
81         for(int i = 1; i ≤ n; ++ i)
82             for(int j = 1; j ≤ n; ++ j)
83                 cin >> A.W[i][j];
84         bool res = mat_inv(A);
85         if(res == false){
86             cout << "No Solution" << endl;
87         } else {
88             for(int i = 1; i ≤ n; ++ i)
89                 for(int j = 1; j ≤ n; ++ j)
90                     cout << A.W[i][j] << " " <<
91                         n;
92         }
93         return 0;
94     }

```

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。

```

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

```

6.3 树图计数

6.3.1 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 $p: v_0 \xrightarrow{w_1} v_1 \xrightarrow{w_2} v_2 \rightarrow \dots \rightarrow^{w_k} v_k$ 的权值: $\omega(p) = \prod w_{i_0}$
- 一对顶点 (a, b) 的权值: $e(a, b) = \sum_{p: a \rightarrow b} \omega(p)$

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 得到一个不相交的路径组 $p = (p_1, p_2, \dots, p_n)$, 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。

6.3.2 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。

6.3.3 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:

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

6.3.4 图连通方案数

n 个点的图, k 个连通块, 第 i 个大小为 s_i , 添加 $k-1$ 条边使得它们连通的方案数:

$$\text{Ans} = n^{k-2} \cdot \prod_{i=1}^k s_i$$

6.4 中国剩余定理

6.4.1 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:

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

```
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], M *= B[i];
8     }
9     i64 L = 0;
10    for(int i = 1; i <= n; ++i){
11        i64 m = M / B[i], b, k;
12        exgcd(m, B[i], b, k);
13        L = (L + (__int128)A[i] * m * b) % M;
14    }
15    L = (L % M + M) % M;
16    cout << L << endl;
17    return 0;
18 }
```

6.5 狄利克雷前缀和

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

```
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){
9                 A[j * i] += A[j];
10            } // 前缀和
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 }
```

6.6 万能欧几里得

6.6.1 类欧几里得 (万能欧几里得)

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。

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, R);
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 - 1, C, C - B % C + A - 1, A, U, R) * U.qpow(l);
10 }
```

6.7 扩展欧几里得

6.7.1 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, y = x0;
8         return d;
9     }
10 }
```

6.8 快速离散对数

6.8.1 用法

给定原根 g 以及模数 M , T 次询问 x 的离散对数。复杂度 $O(M^{2/3} + T \log M)$ 。

```
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 }
```

```

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

```

6.9 快速最大公约数

6.9.1 用法

已知小值域 m 以及 n 次询问, $\mathcal{O}(m)$ 预处理, $\mathcal{O}(1)$ 单次查询 x, y 的最大公约数。

```

1 #include "../header.cpp"
2 const int MAXT = 1e6 + 3;
3 int G[MAXM][MAXM], T[MAXT][3];

```

```

4 int A[MAXN], B[MAXN], o = 1e6, h = 1e3, V[MAXT];
5 int tgcd(int a, int b){
6     if(a ≤ h && b ≤ h) return G[a][b];
7     return a == b ? a : 1;
8 }
9 int qgcd(int a, int b){
10    int ans = 1;
11    up(0, 2, i){
12        if(T[b][i] > h){
13            if(a % T[b][i] == 0) a /= T[b][i], ans *= T[b][i];
14        } else {
15            int d = G[a % T[b][i]][T[b][i]];
16            a /= d, ans *= d;
17        }
18    }
19    return ans;
20 }
21 int main(){
22     ios :: sync_with_stdio(false);
23     cin.tie(nullptr);
24     up(1, h, i) G[0][i] = G[i][0] = i;
25     up(1, h, i) up(1, h, j){
26         if(i ≥ j) G[i][j] = G[i - j][j];
27         else G[i][j] = G[i][j - i];
28     }
29     up(2, o, i) if(!V[i]){
30         V[i] = i;
31         for(int j = 2; i * j ≤ o; ++ j)
32             if(!V[i * j]) V[i * j] = i;
33     }
34     T[1][0] = T[1][1] = T[1][2] = 1;
35     up(2, o, i){
36         int p = V[i];
37         int a = T[i / p][0];
38         int b = T[i / p][1];
39         int c = T[i / p][2];
40         int x, y, z;
41         if(p ≥ h){
42             x = 1, y = i / p, z = p;
43         } else {
44             if(c * p ≤ h){
45                 x = a, y = b, z = c * p;
46             }
47             else if(b * p ≤ h){
48                 x = a, y = b * p, z = c;
49                 if(y > z) swap(y, z);
50             }
51             else if(a * p ≤ h){
52                 x = a * p, y = b, z = c;

```

```

53         if(x > y) swap(x, y);
54         if(y > z) swap(y, z);
55     } else {
56         x = a * b, y = c, z = p;
57         if(x > y) swap(x, y);
58         if(y > z) swap(y, z);
59         if(x > z) swap(x, z);
60     }
61 }
62 T[i][0] = x;
63 T[i][1] = y;
64 T[i][2] = z;
65 }
66 int n;
67 cin >> n;
68 up(1, n, i) cin >> A[i];
69 up(1, n, i) cin >> B[i];
70 up(1, n, i){
71     int s = 0, u = 1;
72     up(1, n, j){
73         int d = qgcd(A[i], B[j]);
74         u = 1ll * u * i % MOD;
75         s = (s + 1ll * d * u) % MOD;
76     }
77     printf("%d\n", s);
78 }
79 return 0;
80 }

```

6.10 原根

```

1 #include "../header.cpp"
2 int getphi(int x); // 求解 phi
3 vector<int> getprime(int x); // 求解质因数
4 bool test(int g, int m, int mm, vector<int>&P){
5     for(auto &p: P)
6         if(power(g, mm / p, m) == 1)
7             return false;
8     return true;
9 }
10 int get_genshin(int m){
11     int mm = getphi(m);
12     vector<int> P = getprime(mm);
13     for(int i = 1; ++ i)
14         if(test(i, m, mm, P)) return i;
15 }

```

6.11 快速乘法逆元 (离线)

6.11.1 用法

离线计算 $x = [x_1, x_2, \dots, x_n]$ 在模 p 意义下的逆元。

```

1 #include "../header.cpp"
2 int A[MAXN], B[MAXN];
3 int P[MAXN], Q[MAXN];
4 int main(){
5     ios :: sync_with_stdio(false);
6     cin.tie(nullptr);
7     int n, p, K, S = 1;
8     cin >> n >> p >> K;
9     P[0] = 1;
10    for(int i = 1; i <= n; ++ i){
11        cin >> A[i];
12        P[i] = 1ll * P[i - 1] * A[i] % p;
13    }
14    Q[n] = power(P[n], p - 2, p);
15    for(int i = n; i >= 1; -- i){
16        Q[i - 1] = 1ll * Q[i] * A[i] % p;
17        B[i] = 1ll * Q[i] * P[i - 1] % p;
18    }
19    int ans = 0;
20    for(int i = 1; i <= n; ++ i){
21        S = 1ll * S * K % p;
22        ans = (ans + 1ll * S * B[i]) % p;
23    }
24    cout << ans << "\n";
25    return 0;
26 }

```

6.12 快速乘法逆元（在线）

6.12.1 用法

在线计算 $x = [x_1, x_2, \dots, x_n]$ 在模 p 意义下的逆元。

```

1 #include "../header.cpp"
2 pair<int, int> F[MAXN], G[MAXN];
3 int I[MAXN];
4 using u32 = uint32_t;
5 u32 read(u32 &seed);
6 int main(){
7     ios :: sync_with_stdio(false);
8     cin.tie(nullptr);
9     u32 seed;
10    int n, p;
11    cin >> n >> p >> seed;
12    int m = pow(p, 1.0 / 3.0);
13    I[1] = 1;
14    for(int i = 2; i <= p / m; ++ i){
15        I[i] = 1ll * (p / i) * (p - I[p % i]) % p;
16    }
17    for(int i = 1; i < m; ++ i){
18        for(int j = i + 1; j <= m; ++ j){
19            if(!F[i * m * m / j].second){

```

```

20                F[i * m * m / j] = { i, j };
21                G[i * m * m / j] = { i, j };
22            }
23        }
24    }
25    F[0] = G[0] = { 0, 1 };
26    F[m * m] = G[m * m] = { 1, 1 };
27    for(int i = 1; i < m * m; ++ i) if(!F[i].
        second)
28        F[i] = F[i - 1];
29    for(int i = m * m - 1; i >= 1; -- i) if(!G[i].
        second)
30        G[i] = G[i + 1];
31    int lastans = 0;
32    for(int i = 1; i <= n; ++ i){
33        int a, inv;
34        a = (read(seed) ^ lastans) % (p - 1) + 1;
35        int w = 1ll * a * m * m / p;
36        auto &yy1 = F[w].second; // *avoid y1 in
        <cmath>
37        if(1ll * a * yy1 % p <= p / m){
38            inv = 1ll * I[1ll * a * yy1 % p] * yy1 %
                p;
39        } else {
40            auto &yy2 = G[w].second;
41            inv = 1ll * I[1ll * a * (p - yy2) % p] *
                (p - yy2) % p;
42        }
43        lastans = inv;
44    }
45    cout << lastans << "\n";
46    return 0;
47 }

```

6.13 拉格朗日插值

6.13.1 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：

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

6.14 min-max 容斥

6.14.1 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，可以推广为如下式子：

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

此外在数论上可以得到：

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

6.14.2 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：

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

6.15 Barrett 取模

6.15.1 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 = (((i80)1 << 60 + S) / MOD + !(((i80)1 <<
        60 + S) % MOD));
6     cerr << S << "\n" << 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 * ((i80)r * X >> 60 + S);
14        }
15        x = x * x;
16        x = x - MOD * ((i80)x * X >> 60 + S);
17        y >>= 1;
18    }
19    return r;
20 }

```

6.16 Pollard's Rho

6.16.1 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 ((i80)a * a + c) % m;
4 }
5 i64 multi(i64 a, i64 b, i64 m){
6     return (i80) 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 }

```

6.17 polya 定理

6.17.1 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_0 . 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 .

6.17.2 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:

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

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。

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。

6.17.3 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)), dd = dd * w;
12 }
13 }
14 }
15 int main(){
16     int T; cin >> T;
17     while(T --){
18         int n, t; cin >> n; t = n;
19         for(int i = 2; i ≤ n / i; ++i) if(n % i == 0){
20             int w = i, c = 0;
21             while(t % i == 0) t /= i, c ++;
22             P.push_back({ w, c });
23         }
24         if(t ≠ 1) P.push_back({ t, 1 });
25         int ans = 0;
26         solve(0, n, 1, 1, ans);
27         ans = 1ll * ans * power(n, MOD - 2) % MOD;
28         cout << ans << endl;
29         P.clear();
30     }
31     return 0;
32 }

```

6.18 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。

1. $G_k(n) = \sum_{i=1}^n [\text{mindiv}(i) > p_k \text{ or isprime}(i)] g(i)$ ($p_0 = 1$), 则有 $G_k(n) = G_{k-1}(n) - g(p_k) \times (G_{k-1}(n/p_k) - G_{k-1}(p_{k-1}))$, 复杂度 $O(n^{3/4}/\log n)$ 。
2. $F_k(n) = \sum_{i=1}^n [\text{mindiv}(i) \geq p_k] f(i) = \sum [h \geq k, p_h^2 \leq n] \sum [c \geq 1, p_h^{c+1} \leq n] (f(p_h^c) F_{h+1}(n/p_h^c) + f(p_h^{c+1})) + F_{\text{prime}}(n) - F_{\text{prime}}(p_{k-1})$, 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(

```

```

38     val[i]);
39     ll pre = 0;
40     for (int i = 1; p[i] * p[i] ≤ n; ++i) {
41         for (int j = 1; j ≤ cnt; ++j) {
42             if (p[i] * p[j] > val[j]) break;
43             ll tmp = id(val[j] / p[i]);
44             G[j] = (G[j] - g(p[i]) * (G[tmp] -
          pre)) % MD;
45         }
46         pre = (pre + g(p[i])) % MD;
47     }
48     for (int i = 1; i ≤ cnt; ++i) G[i] = (G[i]
          % MD + MD) % MD;
49     return G;
50 }
51 // 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。
52 function<ll(ll, int)> calcF = [&] (ll m, int k) {
53     if (p[k] > m) return 0;
54     ll ans = (fp[id(m)] - fp[id(p[k - 1])]) %
          MD;
55     for (int h = k; p[h] * p[h] ≤ m; ++h) {
56         ll pc = p[h], c = 1;
57         while (pc * p[h] ≤ m) {
58             ans = (ans + calcF(m / pc, h + 1)
          * f(p[h] ^ c)) % MD;
59             ++c, pc = pc * p[h], ans = (ans +
          f(p[h] ^ c)) % MD;
60         }
61     }
62     return ans;
63 };

```

6.19 杜教筛

6.19.1 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 $S(n) = \sum_{i=1}^n f(i)$:

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

6.19.2 例题

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

6.20 PN 筛

6.20.1 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 $h(p^c)$:

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

6.20.2 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。

```
1 #include "../header.cpp"
2 const int H = 1e7;
3 const int MOD = 1e9 + 7;
4 const int DIV2 = 5000000004;
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) % MOD * DIV6 % MOD;
15 }
16 int sg(i64 n, i64 N){
17     return n ≤ H ? le[n] : ge[N / n];
18 }
19 int sieve_du(i64 N){
20     for(int d = N / H; d ≥ 1; -- d){
21         i64 n = N / d;
22         int wh = s2(n);
23         for(i64 l = 2, r; l ≤ n; l = r + 1){
24             r = n / (n / l);
```

```
25         int wg = (s1(r) - s1(l - 1) + MOD) % MOD;
26         int ws = sg(n / l, N);
27         ge[d] = (ge[d] + 1ll * wg * ws) % MOD;
28     }
29     ge[d] = (wh - ge[d] + MOD) % MOD;
30 }
31 return N ≤ H ? le[N] : ge[1];
32 }
33 vector<int> hc[MAXM], gc[MAXM];
34 int ANS;
35 void sieve_pn(int last, i64 x, int h, i64 N){
36     ANS = (ANS + 1ll * h * sg(N / x, N)) % MOD;
37     for(i64 i = last + 1; x ≤ N / P[i] / P[i]; ++ i){
38         int c = 2;
39         for(i64 t = x * P[i] * P[i]; t ≤ N; t *= P[i], c++){
40             int hh = 1ll * h * hc[i][c] % MOD;
41             sieve_pn(i, t, hh, N);
42         }
43     }
44 }
45 int main(){
46     ios :: sync_with_stdio(false);
47     cin.tie(nullptr);
48     g[1] = 1;
49     for(int i = 2; i ≤ H; ++ i){
50         if(!V[i]){
51             P[++ p] = i, g[i] = 1ll * i * (i - 1) % MOD;
52         }
53         for(int j = 1; j ≤ p && P[j] ≤ H / i; ++ j){
54             int &p = P[j];
55             V[i * p] = true;
56             if(i % p == 0){
57                 g[i * p] = 1ll * g[i] * p % MOD * p % MOD;
58                 break;
59             } else {
60                 g[i * p] = 1ll * g[i] * p % MOD * (p - 1) % MOD;
61             }
62         }
63     }
64     for(int i = 1; i ≤ H; ++ i){
65         le[i] = (le[i - 1] + g[i]) % MOD;
66     }
67     i64 N;
68     cin >> N;
```

```
69     for(int i = 1; i ≤ p && 1ll * P[i] * P[i] ≤ N; i++){
70         int &p = P[i];
71         hc[i].push_back(1), gc[i].push_back(1);
72         for(i64 c = 1, t = p; t ≤ N; t = t * p, ++ c){
73             if(c == 1){
74                 gc[i].push_back(1ll * p * (p - 1) % MOD);
75             } else {
76                 gc[i].push_back(1ll * gc[i].back() * p % MOD * p % MOD);
77             }
78             int w = 1ll * (t % MOD) * ((t - 1) % MOD) % MOD;
79             int s = 0;
80             for(int j = 1; j ≤ c; ++ j){
81                 s = (s + 1ll * gc[i][j] * hc[i][c - j]) % MOD;
82             }
83             hc[i].push_back((w - s + MOD) % MOD);
84         }
85     }
86     sieve_du(N);
87     sieve_pn(0, 1, 1, N);
88     cout << ANS << "\n";
89     return 0;
90 }
```

6.21 二次剩余

6.21.1 用法

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

```
1 #include "../header.cpp"
2 // 检查 x 在模 p 意义下是否有二次剩余
3 bool check(int x, int p){
4     return power(x, (p - 1) / 2, p) == 1;
5 }
6 struct Node { int real, imag; };
7 Node mul(const Node a, const Node b, int p, int v){
8     int nreal = (1ll * a.real * b.real
9         + 1ll * a.imag * b.imag % p * v) % p;
10    int nimag = (1ll * a.real * b.imag
11        + 1ll * a.imag * b.real % p * v) % p;
12    return { nreal, nimag };
13 }
14 Node power(Node a, int b, int p, int v){
15     Node r = { 1, 0 };
```

```

16 while(b){
17     if(b & 1) r = mul(r, a, p, v);
18     b >>= 1, a = mul(a, a, p, v);
19 }
20 return r;
21 }
22 mt19937 MT;
23 // 无解 x1 = x2 = -1, 唯一解 x1 = x2
24 void solve(int n, int p, int &x1, int &x2){
25     if(n == 0){ x1 = x2 = 0; return; }
26     if(!check(n, p)){ x1 = x2 = -1; return; }
27     i64 a, t;
28     do {
29         a = MT() % p;
30     }while(check(t = (a * a - n + p) % p, p));
31     Node u = { a, 1 };
32     x1 = power(u, (p + 1) / 2, p, t).real;
33     x2 = (p - x1) % p;
34     if(x1 > x2) swap(x1, x2);
35 }

```

6.22 单位根反演

6.22.1 定理

给出单位根反演如下:

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

7 多项式

7.1 最小多项式

```

1 #include "../header.cpp"
2 vector<int> bm(const vector<int> &a) {
3     vector<int> v, ls;
4     int k = -1, delta = 0;
5     for (int i = 0; i < a.size(); i++) {
6         int tmp = 0;
7         for (int j = 0; j < v.size(); j++)
8             tmp = (tmp + (ll)a[i - j - 1] * v[j]) %
9                 MD;
10        if (a[i] == tmp) continue;
11        if (k < 0) { k = i; delta = (a[i] - tmp +
12            MD) % MD; v = vector<int>(i + 1);
13            continue; }
14        vector<int> u = v;
15        int val = (ll)(a[i] - tmp + MD) * qpow(
16            delta, MD - 2) % MD;
17        v.resize(max(v.size(), ls.size() + i - k));

```

```

14 (v[i - k - 1] += val) %= MD;
15 for (int j = 0; j < (int)ls.size(); j++){
16     v[i - k + j] = (v[i - k + j] - (ll)val *
17         ls[j]) % MD;
18     if(v[i - k + j] < 0) v[i - k + j] += MD;
19 }
20 if (u.size() + k < ls.size() + i){
21     ls = u; k = i, delta = a[i] - tmp;
22     if (delta < 0) delta += MD;
23 }
24 for (auto &x : v) x = (MD - x) % MD;
25 v.insert(v.begin(), 1);
26 return v;
27 } // ∀i, ∑_{j=0}^m a_{i-j} v_j = 0

```

7.2 NTT 全家桶

```

1 #include "../header.cpp"
2 using Poly = vector<ll>;
3 #define lg(x) ((x) == 0 ? -1 : __lg(x))
4 #define Size(x) (int)(x.size())
5 namespace NTT_ns {
6     const long long G = 3, invG = inv(G);
7     vector<int> rev;
8     void NTT(ll* F, int len, int sgn) {
9         rev.resize(len);
10        for (int i = 1; i < len; ++i) {
11            rev[i] = (rev[i] >> 1] >> 1) | ((i & 1) *
12                (len >> 1));
13            if (i < rev[i]) swap(F[i], F[rev[i]]);
14        }
15        for (int tmp = 1; tmp < len; tmp <= 1) {
16            ll w1 = qpow(sgn ? G : invG,
17                (MD - 1) / (tmp <= 1));
18            for (int i = 0; i < len; i += tmp <= 1){
19                for (ll j = 0, w = 1; j < tmp; ++j,
20                    w = w * w1 % MD) {
21                    ll x = F[i + j];
22                    ll y = F[i + j + tmp] * w % MD;
23                    F[i + j] = (x + y) % MD;
24                    F[i + j + tmp] = (x - y + MD) % MD;
25                }
26            }
27        }
28        if (sgn == 0) {
29            ll inv_len = inv(len);
30            for (int i = 0; i < len; ++i)
31                F[i] = F[i] * inv_len % MD;
32        }

```

```

33 }
34 using NTT_ns::NTT;
35 Poly operator * (Poly F, Poly G) {
36     int siz = Size(F) + Size(G) - 1;
37     int len = 1 << (lg(siz - 1) + 1);
38     if (siz <= 300) {
39         Poly H(siz);
40         for (int i = Size(F) - 1; ~i; --i)
41             for (int j = Size(G) - 1; ~j; --j)
42                 H[i + j] = (H[i + j] + F[i] * G[j]) % MD;
43         return H;
44     }
45     F.resize(len), G.resize(len);
46     NTT(F.data(), len, 1), NTT(G.data(), len, 1);
47     for (int i = 0; i < len; ++i)
48         F[i] = F[i] * G[i] % MD;
49     NTT(F.data(), len, 0), F.resize(siz);
50     return F;
51 }
52 Poly operator + (Poly F, Poly G) {
53     int siz = max(Size(F), Size(G));
54     F.resize(siz), G.resize(siz);
55     for (int i = 0; i < siz; ++i)
56         F[i] = (F[i] + G[i]) % MD;
57     return F;
58 }
59 Poly operator - (Poly F, Poly G) {
60     int siz = max(Size(F), Size(G));
61     F.resize(siz), G.resize(siz);
62     for (int i = 0; i < siz; ++i)
63         F[i] = (F[i] - G[i] + MD) % MD;
64     return F;
65 }
66 Poly lsh(Poly F, int k) {
67     F.resize(Size(F) + k);
68     for (int i = Size(F) - 1; i >= k; --i)
69         F[i] = F[i - k];
70     for (int i = 0; i < k; ++i) F[i] = 0;
71     return F;
72 }
73 Poly rsh(Poly F, int k) {
74     int siz = Size(F) - k;
75     for (int i = 0; i < siz; ++i) F[i] = F[i + k];
76     return F.resize(siz), F;
77 }
78 Poly cut(Poly F, int len) {
79     return F.resize(len), F;
80 }
81 Poly der(Poly F) {
82     int siz = Size(F) - 1;
83     for (int i = 0; i < siz; ++i)

```

```

84 F[i] = F[i + 1] * (i + 1) % MD;
85 return F.pop_back(), F;
86 }
87 Poly inte(Poly F) {
88     F.emplace_back(0);
89     for (int i = Size(F) - 1; ~i; --i)
90         F[i] = F[i - 1] * inv(i) % MD;
91     return F[0] = 0, F;
92 }
93 Poly inv(Poly F) {
94     int siz = Size(F); Poly G{inv(F[0])};
95     for (int i = 2; (i >> 1) < siz; i <= 1) {
96         G = G + G - G * G * cut(F, i), G.resize(i);
97     }
98     return G.resize(siz), G;
99 }
100 Poly ln(Poly F) {
101     return cut(inte(cut(der(F) * inv(F), Size(F))
102         )), Size(F));
103 }
104 Poly exp(Poly F) {
105     int siz = Size(F); Poly G{1};
106     for (int i = 2; (i >> 1) < siz; i <= 1) {
107         G = G * (Poly{1} - ln(cut(G, i)) + cut(F, i)), G.resize(i);
108     }
109     return G.resize(siz), G;
110 }

```

7.3 定系数短多项式卷积

有 n 个多项式 $F_i = \sum_{j=1}^k w_j x^{a_{i,j}}$, (w_i) 序列固定, 并且

k 很小, $a_i \in [0, 2^m)$, 现在要把他们位运算卷积起来, 求最终的序列。

异或版本, 复杂度 $O(2^k((n+m)k + 2^m(m + \log V)))$, 按通常大小关系可以认为是 $O(nk2^k + m2^{m+k})$, 最后那个 $\log V$ 是快速幂, 可以 $O(n2^k)$ 预处理去掉:

```

1 #include "poly-fwt.cpp"
2 #define vv vector
3 int main() {
4     ios::sync_with_stdio(false);
5     cin.tie(nullptr);
6     ll n, k, m;
7     cin >> n >> m, k = 3;
8     vv<ll> w(k);
9     for(auto &i : w) cin >> i;
10    vv<vv<ll>> a(n, vv<ll>(k));

```

```

11 for(auto &i : a) for(auto &j : i) cin >> j;
12 ll uk = 1 << k, V = 1 << m;
13 vv<vv<ll>> c(V, vv<ll>(uk));
14 vv<ll> val(uk);
15 for (ll i = 0; i < uk; ++i) {
16     for (ll p = 0; p < k; ++p)
17         if ((i >> p) & 1)
18             val[i] = (val[i] - w[p] + MD) % MD;
19         else val[i] = (val[i] + w[p]) % MD;
20     vv<ll> f(V);
21     for (ll j = 0; j < n; ++j) {
22         ll z = 0;
23         for (ll p = 0; p < k; ++p)
24             if ((i >> p) & 1) z ^= a[j][p];
25         ++f[z];
26     }
27     FWT(f.data(), V, 1, 1, 1, MD - 1);
28     for (ll j = 0; j < V; ++j) c[j][i] = f[j];
29 }
30 for (ll i = 0; i < V; ++i) FWT(c[i].data(),
31     uk, inv2, inv2, inv2, MD - inv2);
32 vv<ll> Ans(V, 1);
33 for (ll i = 0; i < V; ++i)
34     for (ll j = 0; j < uk; ++j)
35         Ans[i] = Ans[i] * qpow(val[j], c[i][j])
36             % MD;
37 FWT(Ans.data(), V, inv2, inv2, inv2, MD - inv2);
38 for (ll i = 0; i < V; ++i)
39     cout << Ans[i] << "\n"[i == V - 1];
40 return 0;

```

7.4 FWT 全家桶

$$(FA)_i = \sum_j \prod_{k=n}^0 c(B(i, k), B(j, k)) A_j, c_{*, i} c_{*, j} = c_{*, i \oplus j}$$

```

1 #include "../header.cpp"
2 void FWT(ll *F, ll len, ll c00, ll c01, ll c10, ll c11) {
3     for (ll t = 1; t < len; t <= 1) {
4         for (ll i = 0; i < len; i += t * 2) {
5             for (ll j = 0; j < t; ++j) {
6                 ll x = F[i + j];
7                 ll y = F[i + j + t];
8                 F[i + j] = (x * c00 + y * c01) % MD;
9                 F[i + j + t] = (x * c10 + y * c11) % MD;
10            }

```

```

11 }
12 }
13 }

```

7.5 任意模数 NTT

```

1 #include "../header.cpp"
2 using cp = complex<ld>;
3 vector<int> rev;
4 vector<cp> om; // exp(sgn * pi * k / len)
5 void FFT(cp* F, int len, int sgn) {
6     rev.resize(len);
7     for (int i = 1; i < len; ++i) {
8         rev[i] = (rev[i >> 1] >> 1)
9             | ((i & 1) * (len >> 1));
10    if (i < rev[i]) swap(F[i], F[rev[i]]);
11 }
12 const ld pi = std::acos(ld(-1));
13 om.resize(len);
14 for (int i = 0; i < len; ++i) {
15     om[i] = polar(ld(1), sgn * i * pi / len);
16 }
17 for (int tmp = 1; tmp < len; tmp <= 1) {
18     for (int i = 0; i < len; i += tmp < 1) {
19         int K = len / tmp, pos = 0;
20         for (int j = 0; j < tmp; ++j, pos += K) {
21             cp x = F[i + j];
22             cp y = F[i + j + tmp] * om[pos];
23             F[i + j] = x + y;
24             F[i + j + tmp] = x - y;
25         }
26     }
27 }
28 if (sgn == -1) {
29     cp inv_len(ld(1) / len);
30     for (int i = 0; i < len; ++i)
31         F[i] = F[i] * inv_len;
32 }
33 }
34 ll MD, M; // 输入 MD 后, 需要设置 M 为 sqrt(MD)
35 using Poly = vector<ll>;
36 Poly polyMul(Poly F, Poly G, int tmp = 0) { // tmp 用于循环卷积技巧, 卡常
37     for (auto &k : F) k %= MD;
38     for (auto &k : G) k %= MD;
39     int n = (int)F.size() - 1, m = (int)G.size() - 1;
40     if (tmp == 0) tmp = n + m + 1;
41     int len = 1;
42     while (len < tmp) len <= 1;

```

```

43 vector<cp> P(len), tP(len), Q(len);
44 for (int i = 0; i ≤ n; ++i)
45     P[i] = cp(F[i] / M, F[i] % M),
46     tP[i] = cp(F[i] / M, -(F[i] % M));
47 for (int i = 0; i ≤ m; ++i)
48     Q[i] = cp(G[i] / M, G[i] % M);
49 for(auto &X: {&P, &Q, &tP})
50     FFT(X → data(), len, 1);
51 for (int i = 0; i < len; ++i)
52     P[i] *= Q[i], tP[i] *= Q[i];
53 FFT(P.data(), len, -1);
54 FFT(tP.data(), len, -1);
55 vector<ll> H(n + m + 1);
56 for (int i = 0; i < tmp; ++i) {
57     H[i] = ll((P[i].real() + tP[i].real()) / 2
58         + 0.5) % MD * M % MD * M % MD
59         + ll(P[i].imag() + 0.5) % MD * M % MD
60         + ll((tP[i].real() - P[i].real()) / 2 +
61             0.5) % MD;
62     H[i] = (H[i] + MD) % MD;
63 }

```

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 p = F[u], v = C[u][i];
19                if(v == 0) continue;
20                while(!C[p][i] && p ≠ 0) p = F[p];
21                if(C[p][i] && C[p][i] ≠ v)
22                    F[v] = C[p][i];
23                Q.push(v);

```

```

24     }
25 }
26 }
27 }

```

8.2 扩展 KMP

8.2.1 定义

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

```

1 #include "../header.cpp"
2 int Z[MAXN];
3 void exkmp(char A[]){
4     int l = 0, r = 0; Z[1] = 0;
5     for(int i = 2; A[i]; ++ i){
6         Z[i] = i ≤ r ? min(r - i + 1, Z[i - l +
7             1]) : 0;
8         while(A[Z[i] + 1] == A[i + Z[i]]) ++ Z[i];
9         if(i + Z[i] - 1 > r)
10            r = i + Z[i] - 1, l = i;
11    }

```

8.3 Manacher

```

1 #include "../header.cpp"
2 char S[MAXN], T[MAXN]; int n, R[MAXN];
3 int main(){
4     scanf("%s", S + 1);
5     n = strlen(S + 1);
6     for(int i = 1; i ≤ n; ++ i){
7         T[2 * i - 1] = S[i], T[2 * i] = '#';
8     }
9     T[0] = '#', n = 2 * n;
10    int p = 0, x = 0, ans = 0;
11    for(int i = 1; i ≤ n; ++ i){
12        if(i ≤ p) R[i] = min(R[2 * x - i], p - i);
13        while(i - R[i] - 1 ≥ 0
14            && T[i + R[i] + 1] == T[i - R[i] - 1])
15            ++ R[i];
16        if(i + R[i] > p){
17            p = i + R[i];
18            x = i;
19        }
20        ans = max(ans, R[i]);
21    }
22    printf("%d\n", ans);
23    return 0;
24 }

```

8.4 最小表示法

```

1 #include "../header.cpp"
2 int min_pos(const vector<int> &a) {
3     int n = a.size(), i = 0, j = 1, k = 0;
4     while (i < n && j < n && k < n) {
5         int u = a[(i + k) % n], v = a[(j + k) % n];
6         int t = u > v ? 1 : (u < v ? -1 : 0);
7         if (t == 0) k++; else {
8             if (t > 0) i += k + 1; else j += k + 1;
9             if (i == j) j++;
10            k = 0;
11        }
12    }
13    return min(i, j);
14 }

```

8.5 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, last = cur;
30        }
31    }
32 }

```

8.6 后缀平衡树

8.6.1 本代码尚未完成

8.7 后缀数组 (倍增)

```

1 #include "../header.cpp"
2 int n, m, A[MAXN], B[MAXN], C[MAXN], R[MAXN],
  P[MAXN], Q[MAXN];
3 char S[MAXN];
4 int main(){
5     scanf("%s", S), n = strlen(S), m = 256;
6     for(int i = 0; i < n; ++i) R[i] = S[i];
7     for (int k = 1; k ≤ n; k <= 1){
8         for(int i = 0; i < n; ++i){
9             Q[i] = ((i + k > n - 1) ? 0 : R[i + k]);
10            P[i] = R[i];
11            m = max(m, R[i]);
12        }
13 #define fun(a, b, c) \
14     memset(C, 0, sizeof(int) * (m + 1)); \
15     for(int i = 0; i < n; ++i) C[a] += 1; \
16     for(int i = 1; i ≤ m; ++i) C[i] += C[i - 1]; \
17     for(int i = n - 1; i ≥ 0; --i) c[--C[a]] = b; \
18     fun(Q[i], i, B)
19     fun(P[B[i]], B[i], A)
20 #undef fun
21     int p = 1; R[A[0]] = 1;
22     for(int i = 1; i ≤ n - 1; ++i){
23         bool f1 = P[A[i]] = P[A[i - 1]];
24         bool f2 = Q[A[i]] = Q[A[i - 1]];
25         R[A[i]] = f1 && f2 ? R[A[i - 1]] : ++p;
26     }
27     if (m == n) break;
28 }
29 for(int i = 0; i < n; ++i)
30     printf("%u", A[i] + 1);
31 return 0;
32 }
```

8.8 后缀数组 (SAIS)

```

1 #include "../header.cpp"
2 #define LTYPE 0
3 #define STYPE 1
4 void induce_sort(int n, int S[], int T[], int
  m, int LM[], int SA[], int C[]){
5     vector<int> BL(n), BS(n), BM(n);
6     fill(SA, SA + n, -1);
7     for(int i = 0; i < n; ++i){ // 预处理
8         BM[i] = BS[i] = C[i] - 1;
```

```

9         BL[i] = i == 0 ? 0 : C[i - 1];
10    }
11    for(int i = m - 1; i ≥ 0; --i) // 放置
12        LMS 后缀
13        SA[BM[S[LM[i]]] --] = LM[i];
14    for(int i = 0, p; i < n; ++i) // 计算 L
15        类型后缀的位置
16        if(SA[i] > 0 && T[p = SA[i] - 1] == LTYPE)
17            SA[BL[S[p]] ++] = p;
18    for(int i = n - 1, p; i ≥ 0; --i) // 计算 S
19        类型后缀的位置
20        if(SA[i] > 0 && T[p = SA[i] - 1] == STYPE)
21            SA[BS[S[p]] --] = p;
22 }
23 // 长度 n, 字符集 [0, n), 要求最后一个元素为 0
24 // 例如输入 ababa 传入 n = 6, S = [1 2 1 2 1
25 // 0]
26 void sais(int n, int S[], int SA[]){
27     vector<int> T(n), C(n), I(n, -1);
28     T[n - 1] = STYPE;
29     for(int i = n - 2; i ≥ 0; --i){ // 递推类
30         型
31         T[i] = S[i] == S[i + 1] ? T[i + 1] : (S[i]
32             < S[i + 1] ? STYPE : LTYPE);
33     }
34     for(int i = 0; i < n; ++i) // 统计个数
35         C[S[i]] ++;
36     for(int i = 1; i < n; ++i) // 前缀累加
37         C[i] += C[i - 1];
38     vector<int> P;
39     for(int i = 0; i < n; ++i){ // 统计 LMS 后
40         缀
41         if(T[i] == STYPE && (i == 0 || T[i - 1] ==
42             LTYPE)){
43             I[i] = P.size(), P.push_back(i);
44         }
45     }
46     int m = P.size(), tot = 0, cnt = 0;
47     induce_sort(n, S, T.data(), m, P.data(), SA,
48         C.data());
49     vector<int> S0(m), SA0(m);
50     for(int i = 0, x, y = -1; i < n; ++i){
51         if((x = I[SA[i]]) ≠ -1){
52             if(tot == 0 || P[x + 1] - P[x] ≠ P[y +
53                 1] - P[y])
54                 tot ++;
55             else for(int p1 = P[x], p2 = P[y]; p2 ≤
56                 P[y + 1]; ++ p1, ++ p2){
57                 if((S[p1] << 1 | T[p1]) ≠ (S[p2] << 1
58                     | T[p2])){
59                     tot ++; break;
60                 }
61             }
62         }
63     }
64     S0[y = x] = tot - 1;
65 }
```

```

48    }
49    }
50    S0[y = x] = tot - 1;
51    }
52 }
53 if(tot == m){
54     for(int i = 0; i < m; ++i)
55         SA0[S0[i]] = i;
56 } else {
57     sais(m, S0.data(), SA0.data());
58 }
59 for(int i = 0; i < m; ++i)
60     S0[i] = P[SA0[i]];
61 induce_sort(n, S, T.data(), m, S0.data(), SA,
62     C.data());
63 int S[MAXN], SA[MAXN], H[MAXM], G[MAXM];
64 int main(){
65     int n = 0, t = 0, m = 256;
66     for(char c = cin.get(); isgraph(c); c = cin.
67         get()){
68         S[n++] = c;
69         H[c] ++;
70     }
71     for(int i = 0; i < m; ++i){
72         t += !!H[i], G[i] = t;
73     }
74     for(int i = 0; i < n; ++i){
75         S[i] = G[S[i]];
76     }
77     sais(n + 1, S, SA);
78     for(int i = 1; i ≤ n; ++i){
79         cout << SA[i] + 1 << " ";
80     }
81     return 0;
82 }
```

8.9 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], L[SIZ], F[SIZ], S[SIZ], s
5     = 0, h = 25;
6     void init(){ F[0] = -1, s = 0; }
7     void extend(int &last, char c){
8         int e = c - 'a';
9         int cur = ++s;
10        L[cur] = L[last] + 1;
11        int p = last;
12        while(p ≠ -1 && !M[p][e])
```

```

12 M[p][e] = cur, p = F[p];
13 if(p == -1){
14     F[cur] = 0;
15 } else {
16     int q = M[p][e];
17     if(L[p] + 1 == L[q]){
18         F[cur] = q;
19     } else {
20         int clone = ++s;
21         L[clone] = L[p] + 1, F[clone] = F[q];
22         for(int i = 0; i ≤ h; ++i)
23             M[clone][i] = M[q][i];
24         while(p ≠ -1 && M[p][e] == q)
25             M[p][e] = clone, p = F[p];
26         F[cur] = F[q] = clone;
27     }
28 }
29 last = cur;
30 }
31 }
32 namespace Trie{
33     int M[MAXN][MAXM], O[MAXN], s, h = 25;
34     void insert(char *S);
35     void build_sam(){
36         queue<int> Q; Q.push(0);
37         while(!Q.empty()){
38             int u = Q.front(); Q.pop();
39             for(int i = 0; i ≤ h; ++i){
40                 char c = i + 'a';
41                 if(M[u][i]){
42                     int v = M[u][i];
43                     O[v] = O[u];
44                     SAM :: extend(O[v], c);
45                     Q.push(v);
46                 }
47             }
48         }
49     }
50 }

```

8.10 广义后缀自动机（在线）

```

1 #include "../header.cpp"
2 namespace SAM{
3     int M[MAXN][MAXM], L[MAXN], F[MAXN], S[MAXN]
4     ], s = 0, h = 25;
5     void init(){
6         F[0] = -1, s = 0;
7     }
8     // 每次插入新字符串前将 last 清零
9     void extend(int &last, char c){

```

```

9     int e = c - 'a';
10    if(M[last][e]){
11        int p = last;
12        int q = M[p][e];
13        if(L[q] == L[p] + 1){
14            last = q;
15        } else {
16            int clone = ++s;
17            L[clone] = L[p] + 1, F[clone] = F[q];
18            for(int i = 0; i ≤ h; ++i)
19                M[clone][i] = M[q][i];
20            while(p ≠ -1 && M[p][e] == q)
21                M[p][e] = clone, p = F[p];
22            F[q] = clone;
23            last = clone;
24        }
25    } else {
26        int cur = ++s;
27        L[cur] = L[last] + 1;
28        int p = last;
29        while(p ≠ -1 && !M[p][e])
30            M[p][e] = cur, p = F[p];
31        if(p == -1){
32            F[cur] = 0;
33        } else {
34            int q = M[p][e];
35            if(L[p] + 1 == L[q]){
36                F[cur] = q;
37            } else {
38                int clone = ++s;
39                L[clone] = L[p] + 1, F[clone] = F[q];
40                for(int i = 0; i ≤ h; ++i)
41                    M[clone][i] = M[q][i];
42                while(p ≠ -1 && M[p][e] == q)
43                    M[p][e] = clone, p = F[p];
44                F[q] = F[clone] = clone;
45            }
46        }
47        last = cur;
48    }
49 }
50 }

```

8.11 后缀自动机

```

1 #include "../header.cpp"
2 namespace SAM{
3     int M[MAXN][MAXM], L[MAXN], F[MAXN], S[MAXN]
4     ], last = 0, s = 0, h = 25;
5     void init(){
6         F[0] = -1, last = s = 0;

```

```

6 }
7 void extend(char c){
8     int cur = ++s, e = c - 'a';
9     L[cur] = L[last] + 1, S[cur] = 1;
10    int p = last;
11    while(p ≠ -1 && !M[p][e])
12        M[p][e] = cur, p = F[p];
13    if(p == -1){
14        F[cur] = 0;
15    } else {
16        int q = M[p][e];
17        if(L[p] + 1 == L[q]){
18            F[cur] = q;
19        } else {
20            int clone = ++s;
21            L[clone] = L[p] + 1, F[clone] = F[q];
22            S[clone] = 0;
23            for(int i = 0; i ≤ h; ++i)
24                M[clone][i] = M[q][i];
25            while(p ≠ -1 && M[p][e] == q)
26                M[p][e] = clone, p = F[p];
27            F[q] = F[clone] = clone;
28        }
29    }
30    last = cur;
31 }
32 }

```

9 计算几何

9.1 二维圆形

```

1 #include "2d.cpp"
2 struct circ: pp { db r; }; // 圆形
3 circ circ_i(pp a, pp b, pp c) {
4     db A = dis(a,b), B = dis(a,c), C = dis(a,b);
5     return {
6         (a * A + b * B + c * C) / (A + B + C),
7         abs((b - a) * (c - a)) / (A + B + C)
8     };
9 } // 三点确定内心
10 circ circ_2pp(pp a, pp b){
11     return {(a + b) / 2, dis(a, b) / 2};
12 } // 两点确定直径
13 circ circ_3pp(pp a, pp b, pp c) {
14     pp bc = c - b, ca = a - c, ab = b - a;
15     pp o = (b + c - r90(bc) * (ca % ab) / (ca *
16         ab)) / 2;
17     return {o, (a - o).abs()};
18 } // 三点确定外心
19 circ minimal(vector<pp> V){ // 最小圆覆盖

```

```

19 shuffle(V.begin(), V.end(), MT);
20 circ C(V[0], 0);
21 for(int i = 0; i < V.size(); ++ i) {
22     if (dis((pp)C, V[i]) < C.r) continue;
23     C = circ_2pp(V[i], V[0]);
24     for(int j = 0; j < i; ++ j) {
25         if (dis((pp)C, V[j]) < C.r) continue;
26         C = circ_2pp(V[i], V[j]);
27         for(int k = 0; k < j; ++ k) {
28             if (dis((pp)C, V[k]) < C.r) continue;
29             C = circ_3pp(V[i], V[j], V[k]);
30         }
31     }
32 }
33 return C;
34 }

```

9.2 二维凸包

9.2.1 例题

给定 n 个点，保证每三点不共线。要求找到一个简单多边形满足它不是凸包，使得该多边形面积最大。

```

1 #include <bits/stdc++.h>
2 using namespace std;
3 using i64 = long long;
4 const int MAXN = 2e5 + 3;
5 int X[MAXN], Y[MAXN];
6 struct Frac {
7     int a, b;
8     Frac(int _a, int _b){
9         if(_b < 0){
10             a = -_a, b = -_b;
11         } else {
12             a = _a, b = _b;
13         }
14     }
15 };
16 struct Node {
17     int x, y;
18 } P[MAXN];
19 bool operator < (const Frac A, const Frac B){
20     return 1ll * A.a * B.b - 1ll * A.b * B.a < 0;
21 }
22 bool operator < (const Node A, const Node B){
23     return A.x == B.x ? A.y > B.y : A.x < B.x;
24 }
25 const Frac intersect(Node A, Node B){
26     int a = B.y - A.y;
27     int b = A.x - B.x;

```

```

28     assert(b != 0);
29     if(b < 0){
30         a = -a, b = -b;
31     }
32     return Frac(a, b);
33 }
34 bool F[MAXN];
35 int main(){
36     int TT;
37     cin >> TT;
38     while(TT -- ){
39         int n;
40         cin >> n;
41         int maxx = -1e9, minx = 1e9;
42         for(int i = 1; i ≤ n; ++ i){
43             auto &[x, y] = P[i];
44             cin >> x >> y;
45             F[i] = false;
46         }
47         sort(P + 1, P + 1 + n);
48         vector <int> Q1, Q2, Q;
49         // Q1 计算上凸壳, Q2 计算下凸壳
50         for(int i = 1; i ≤ n; ++ i){
51             auto &[x, y] = P[i];
52             if(Q1.size() ≤ 1){
53                 Q1.push_back(i);
54             } else {
55                 while(Q1.size() ≥ 2){
56                     auto &[x1, y1] = P[Q1[Q1.size() - 1]];
57                     auto &[x2, y2] = P[Q1[Q1.size() - 2]];
58                     long long cmp = 1ll * (y - y1) * (x1 - x2) - 1ll * (x - x1) * (y1 - y2);
59                     if(cmp > 0){
60                         Q1.pop_back();
61                     } else break;
62                 }
63                 Q1.push_back(i);
64             }
65             if(Q2.size() ≤ 1){
66                 Q2.push_back(i);
67             } else {
68                 while(Q2.size() ≥ 2){
69                     auto &[x1, y1] = P[Q2[Q2.size() - 1]];
70                     auto &[x2, y2] = P[Q2[Q2.size() - 2]];
71                     long long cmp = 1ll * (y - y1) * (x1 - x2) - 1ll *

```

```

72             (x - x1) * (y1 - y2);
73             if(cmp < 0){
74                 Q2.pop_back();
75             } else break;
76             Q2.push_back(i);
77         }
78     }
79     Q = Q1;
80     for(int i = Q2.size(); i != 0; i --){
81         if(i != Q2.size())
82             Q.push_back(Q2[i - 1]);
83     }
84     long long area = 0;
85     int x0 = P[Q[0]].x;
86     int y0 = P[Q[0]].y;
87     for(int i = 1; i + 1 < Q.size(); ++ i){
88         auto &[x1, y1] = P[Q[i]];
89         auto &[x2, y2] = P[Q[i + 1]];
90         area += 1ll * (x1 - x0) * (y2 - y0) - 1ll * (x2 - x0) * (y1 - y0);
91     }
92     area = -area;
93     for(auto &i: Q1) F[i] = true;
94     for(auto &i: Q2) F[i] = true;
95     bool ok = false;
96     for(int i = 1; i ≤ n; ++ i) if(!F[i]){
97         ok = true;
98         maxx = max(maxx, P[i].x);
99         minx = min(minx, P[i].x);
100     }
101     if(!ok){
102         cout << -1 << "\n";
103         continue;
104     }
105     vector <int> L1;
106     vector <int> L2;
107     // L1 插入 kx + b 维护下凸壳
108     for(int i = 1; i ≤ n; ++ i) if(!F[i]){
109         auto &[k, b] = P[i];
110         if(!L1.empty() && k == P[L1.back()].x)
111             continue;
112         while(L1.size() ≥ 2){
113             auto &P1 = P[L1[L1.size() - 1]];
114             auto &P2 = P[L1[L1.size() - 2]];
115             Frac i1 = intersect(P1, P[i]);
116             Frac i2 = intersect(P2, P[i]);
117             if(i1 < i2){

```

```

118         L1.pop_back();
119     } else break;
120 }
121 L1.push_back(i);
122 }
123 // L2 插入 kx + b 维护上凸壳
124 for(int i = n; i ≥ 1; -- i) if(!F[i]){
125     auto &k, b = P[i];
126     if(!L2.empty() && k == P[L2.back()
127         ].x)
128         continue;
129     while(L2.size() ≥ 2){
130         auto &P1 = P[L2[L2.size() -
131             1]];
132         auto &P2 = P[L2[L2.size() -
133             2]];
134         Frac i1 = intersect(P1, P[i]);
135         Frac i2 = intersect(P2, P[i]);
136         if(i1 < i2){
137             L2.pop_back();
138         } else break;
139     }
140     L2.push_back(i);
141 }
142 vector <Frac> E1;
143 E1.push_back(Frac( -2e9, 1 ));
144 for(int i = 0; i + 1 < L1.size(); ++ i){
145     auto &P1 = P[L1[i]];
146     auto &P2 = P[L1[i + 1]];
147     E1.push_back(intersect(P1, P2));
148 }
149 vector <Frac> E2;
150 E2.push_back(Frac( -2e9, 1 ));
151 for(int i = 0; i + 1 < L2.size(); ++ i){
152     auto &P1 = P[L2[i]];
153     auto &P2 = P[L2[i + 1]];
154     E2.push_back(intersect(P1, P2));
155 }
156 long long ans = 0;
157 for(int i = 0; i + 1 < Q.size(); ++ i){
158     auto &x1, y1 = P[Q[i]];
159     auto &x2, y2 = P[Q[i + 1]];
160     long long w = 1ll * x2 * y1 - 1ll
161         * x1 * y2;
162     int A = y2 - y1;
163     int B = x1 - x2;
164     int x = 0, y = 0;
165     if(B == 0){
166         if(A > 0){
167             x = minx, y = 0;
168         } else {

```

```

165         x = maxx, y = 0;
166     }
167 } else
168 if(B < 0){
169     Frac K = Frac(-A, -B);
170     int p = 0;
171     for(int k = 20; k ≥ 0; -- k){
172         int pp = p | 1 << k;
173         if(pp < E1.size() && E1[pp
174             ] < K){
175             p = pp;
176         }
177     }
178     x = P[L1[p]].x;
179     y = P[L1[p]].y;
180 } else {
181     Frac K = Frac( A, B);
182     int p = 0;
183     for(int k = 20; k ≥ 0; -- k){
184         int pp = p | 1 << k;
185         if(pp < E2.size() && E2[pp
186             ] < K){
187             p = pp;
188         }
189     }
190     x = P[L2[p]].x;
191     y = P[L2[p]].y;
192     ans = max(ans, area - (w + 1ll * A
193         * x + 1ll * B * y));
194 }
195 // cerr << "ans = " << ans << endl;
196 cout << ans << "\n";
197 }
198 return 0;

```

9.3 Delaunay 三角剖分

```

1 // From Let it Rot. 在整数坐标同样可用
2 #include "2d.cpp"
3 using Q = struct Quad *;
4 const pp arb(LLONG_MAX, LLONG_MAX);
5 struct Quad {
6     Q rot, o; pp p = arb; bool mark;
7     pp& F() { return r() → p; }
8     Q& r() { return rot→rot; }
9     Q prev() { return rot→o→rot; }
10    Q next() { return r()→prev(); }
11 } *H;
12 ll cross(pp a, pp b, pp c) {

```

```

13     return (b - a) * (c - a);
14 }
15 // p 是否在 a, b, c 外接圆中
16 bool incirc(pp p, pp a, pp b, pp c) {
17     i80 p2 = p.norm(), A = a.norm() - p2, B = b.
18         norm() - p2, C = c.norm() - p2;
19     a = a - p, b = b - p, c = c - p;
20     return (a * b) * C + (b * c) * A + (c * a) *
21         B > 0;
22 }
23 Q link(pp orig, pp dest) {
24     Q r = H ? H : new Quad{new Quad{new Quad{new
25         Quad{0}}}};
26     H = r → o; r → r() → r() = r;
27     for(int i = 0; i < 4; ++ i)
28         r = r → rot, r → p = arb,
29         r → o = i & 1 ? r : r → r();
30     return r;
31 }
32 void splice(Q a, Q b) {
33     swap(a → o → rot → o, b → o → rot → o);
34     swap(a → o, b → o);
35 }
36 Q conn(Q a, Q b) {
37     Q q = link(a → F(), b → p);
38     splice(q, a → next());
39     splice(q → r(), b);
40     return q;
41 }
42 pair<Q, Q> rec(const vector<pp> &s){
43     int N = size(s);
44     if(N ≤ 3) {
45         Q a = link(s[0], s[1]), b = link(s[1], s.
46             back());
47         if(N == 2) return {a, a → r()};
48         splice(a → r(), b);
49         ll side = cross(s[0], s[1], s[2]);
50         Q c = side ? conn(b, a) : 0;
51         return { side < 0 ? c → r() : a, side < 0
52             ? c : b → r() };
53     }
54     #define H(e) e → F(), e → p
55     #define valid(e) (cross(e→F(), H(base)) > 0)
56     int half = N / 2;
57     auto [ra, A]=rec({s.begin(), s.end()-half});
58     auto [B, rb]=rec({s.end() - half, s.end()});
59     while((cross(B → p, H(A)) < 0 && (A = A →
60         next())) || (cross(A → p, H(B)) > 0 && (B
61             = B → r() → o)));
62     Q base = conn(B → r(), A);

```

```

57 if(A → p = ra → p) ra = base → r();
58 if(B → p = rb → p) rb = base;
59 #define DEL(e, init, dir) \
60 Q e = init → dir; \
61 if(valid(e)) \
62 for(;incirc(e → dir → F(), H(base), e →
63 F());) { \
64 Q t = e → dir; \
65 splice(e, e → prev()); \
66 splice(e → r(), e → r() → prev()); \
67 e → o = H, H = e, e = t; \
68 }
69 for(;;) {
70 DEL(LC, base → r(), o);
71 DEL(RC, base, prev());
72 if(!valid(LC) && !valid(RC)) break;
73 if(!valid(LC) || (valid(RC) && incirc(H(RC)
74 ), H(LC)))
75 base = conn(RC, base → r());
76 else
77 base = conn(base → r(), LC → r());
78 }
79 // 返回若干逆时针三角形
80 vector<pp> deluanay(vector<pp> a){
81 if((int)size(a) < 2) return {};
82 sort(a.begin(), a.end()); // unique
83 Q e = rec(a).first;
84 vector<Q> q = {e};
85 while(cross(e → o → F(), e → F(), e → p)
86 < 0) e = e → o;
87 #define ADD { Q c = e; do { c → mark = 1; a
88 .push_back(c → p); q.push_back(c → r()),
89 c = c → next(); } while(c ≠ e); }
90 ADD; a.clear();
91 for(int qi = 0; qi < size(q);)
92 if(!(e = q[qi++] → mark) ADD;
93 return a;
94 }

```

9.4 动态凸包（不支持删除）

```

1 #include "2d.cpp"
2 struct Hull {
3 set<pp> S;
4 using Iter = set<pp>::iterator;
5 long long ans = 0;
6 long long area(pp a, pp b, pp c) {
7 return llabs((b - a) * (c - a));
8 }

```

```

9 Iter remove_dir(Iter it, bool dir, pp x){
10 Iter nxt;
11 while (dir ? (it ≠ S.begin() && (nxt =
12 prev(it), 1)) : ((nxt = next(it)) ≠ S.
13 end())) {
14 if (((*it - *nxt) * (x - *it) < 0) =
15 dir) break;
16 ans += area(*nxt, *it, x), S.erase(it),
17 it = nxt;
18 }
19 return it;
20 }
21 void insert(pp x){
22 if (S.empty()){ S.insert(x); return; }
23 auto r = S.lower_bound(x);
24 if (r = S.end()) {
25 remove_dir(prev(r), 1, x);
26 S.insert(x);
27 } else {
28 auto l = prev(r);
29 if (((*r - *l) * (x - *l)) < 0)
30 return; // 在凸包外侧
31 ans += area(*l, *r, x);
32 l = remove_dir(l, 1, x);
33 r = remove_dir(r, 0, x);
34 S.insert(x);
35 }
36 }

```

9.5 半平面交

```

1 // From Let it Rot
2 #include "2d-lines.cpp"
3 vector<pp> HPI(vector<line> vs) {
4 auto cmp = [](line a, line b) {
5 return paraS(a, b) ? dis(a) < dis(b)
6 : ::cmp(pp(a), pp(b));
7 };
8 sort(vs.begin(), vs.end(), cmp);
9 int ah = 0, at = 0, n = size(vs);
10 vector<line> deq(n + 1);
11 vector<pp> ans(n);
12 deq[0] = vs[0];
13 for(int i = 1; i ≤ n; ++i) {
14 line o = i < n ? vs[i] : deq[ah];
15 if(paraS(vs[i - 1], o)) continue;
16 // maybe ≤

```

```

17 while(ah < at && check(deq[at - 1], deq[at
18 ], o) < 0)
19 -- at;
20 if(i ≠ n)
21 while(ah < at && check(deq[ah], deq[ah +
22 1], o) < 0)
23 ++ ah;
24 if(!is_parallel(o, deq[at])) {
25 ans[at] = o & deq[at], deq[++at] = o;
26 }
27 if(at - ah ≤ 2) return {};
28 return {ans.begin() + ah, ans.begin() + at};
29 }

```

9.6 二维线段

```

1 // From Let it Rot
2 #include "2d.cpp"
3 struct line : pp {
4 db z; // a * x + b * y + c (= or >) 0
5 line() = default;
6 line(db a, db b, db c): pp{a, b}, z(c){}
7 // 有向平面 a → b 左侧区域
8 line(pp a, pp b): pp(r90(b - a)), z(a * b){}
9 db operator()(pp a) const { // ax + by + c
10 return a % pp(*this) + z;
11 }
12 line vertical() const {
13 return {y, -x, 0};
14 } // 过 0 的垂直线
15 line parallel(pp o) {
16 return {x, y, z - this → operator()(o)};
17 } // 过 0 的平行线
18 };
19 pp operator & (line x, line y) { // 求交
20 return pp{
21 pp{x.z, x.y} * pp{y.z, y.y},
22 pp{x.x, x.z} * pp{y.x, y.z}
23 } / -(pp(x) * pp(y));
24 } // 注意此处精度误差较大, res.y 需要较高精度
25 pp project(pp x, line l) { // 投影
26 return x - pp(l) * (l(x) / l.norm());
27 }
28 pp reflect(pp x, line l) { // 对称
29 return x - pp(l) * (l(x) / l.norm()) * 2;
30 }
31 db dis(line l, pp x = {0, 0}) { // 有向点距离
32 return l(x) / l.abs();
33 }
34 bool is_parallel(line x, line y) { // 判断平行

```

```

35 return equal(pp(x) * pp(y), 0);
36 }
37 bool is_vertical(line x, line y){ // 判断垂直
38 return equal(pp(x) % pp(y), 0);
39 }
40 bool online(pp x, line l) { // 判断点在线
41 return equal(l(x), 0);
42 }
43 int ccw(pp a, pp b, pp c) {
44 int s = sign((b - a) * (c - a));
45 if(s == 0) {
46 if(sign((b - a) % (c - a)) == -1)
47 return 2;
48 if((c - a).norm() > (b - a).norm() + EPS)
49 return -2;
50 }
51 return s;
52 }
53 db det(line a, line b, line c) {
54 pp A = a, B = b, C = c;
55 return c.z * (A * B) + a.z * (B * C) + b.z *
    (C * A);
56 }
57 db check(line a, line b, line c) { // sgn same
    as c(a & b), 0 if error
58 return sign(det(a, b, c)) * sign(pp(a) * pp(
    b));
59 }
60 bool paraS(line a, line b) { // 射线同向
61 return is_parallel(a, b) && pp(a)%pp(b) > 0;
62 }

```

9.7 最左转线

```

1 #include "2d.cpp"
2 namespace DSU{
3     const int MAXN = 1e5 + 3;
4     int F[MAXN];
5     int getfa(int u){
6         return u = F[u] ? u : F[u] = getfa(F[
            u]);
7     }
8 }
9 namespace Dual{
10     const int MAXN = 1e5 + 3;
11     const int MAXM = 1e5 + 3;
12     int A[MAXM], B[MAXM], W[MAXM], I[MAXM], n,
        m;
13     int outer;
14     bool cmp(int a, int b){
15         return W[a] < W[b];

```

```

16 }
17 vector <pair<int, int> > E[MAXN];
18 const int MAXT = 20 + 3;
19 int F[MAXN][MAXT], G[MAXN][MAXT], D[MAXN],
    h = 20;
20 void dfs(int u, int f){
21     D[u] = D[f] + 1;
22     for(int i = 1; i ≤ h; ++ i)
23         F[u][i] = F[F[u][i - 1]][i - 1],
24         G[u][i] = max(G[u][i - 1], G[F[u][
            i - 1]][i - 1]);
25     for(auto &[v, w] : E[u]) if(v ≠ f){
26         G[v][0] = w;
27         F[v][0] = u;
28         dfs(v, u);
29     }
30 }
31 void build(){
32     for(int i = 1; i ≤ n; ++ i)
33         DSU :: F[i] = i;
34     for(int i = 1; i ≤ m; ++ i)
35         I[i] = i;
36     sort(I + 1, I + 1 + m, cmp);
37     for(int i = 1; i ≤ m; ++ i){
38         int a = A[I[i]];
39         int b = B[I[i]];
40         int w = W[I[i]];
41         int fa = DSU :: getfa(a);
42         int fb = DSU :: getfa(b);
43         if(fa ≠ fb){
44             DSU :: F[fa] = fb;
45             E[a].push_back({b, w});
46             E[b].push_back({a, w});
47         }
48     }
49     dfs(1, 0);
50 }
51 int solve(int u, int v){
52     if(u == outer || v == outer)
53         return -1;
54     int ans = 0;
55     if(D[u] < D[v]) swap(u, v);
56     for(int i = h; i ≥ 0; -- i)
57         if(D[F[u][i]] ≥ D[v]){
58             ans = max(ans, G[u][i]);
59             u = F[u][i];
60         }
61     if(u == v) return ans;
62     for(int i = h; i ≥ 0; -- i)
63         if(F[u][i] ≠ F[v][i]){
64             ans = max(ans, G[u][i]);

```

```

65         ans = max(ans, G[v][i]);
66         u = F[u][i];
67         v = F[v][i];
68     }
69     ans = max(ans, G[u][0]);
70     ans = max(ans, G[v][0]);
71     return ans;
72 }
73 }
74 namespace Planer{
75     const int MAXN = 1e5 + 3 + 3;
76     const int MAXE = 2e5 + 3;
77     const int MAXG = 1e5 + 3;
78     const int MAXQ = 2e5 + 3;
79     point P[MAXN];
80     using edge = tuple<int, int>;
81     double gety(int a, int b, double x){
82         return P[a].y + (x - P[a].x) / (P[b].x
            - P[a].x) * (P[b].y - P[a].y);
83     }
84     double scanx;
85     struct Cmp1{
86         bool operator()(const pair<edge, int>
            l1, const pair<edge, int> l2) const
            {
87             const edge &e1 = l1.first;
88             const edge &e2 = l2.first;
89             double h1 = gety(get<0>(e1), get
                <1>(e1), scanx);
90             double h2 = gety(get<0>(e2), get
                <1>(e2), scanx);
91             return h1 < h2;
92         };
93     };
94     struct Cmp2{
95         bool operator()(const pair<edge, int>
            l1, const pair<edge, int> l2) const
            {
96             if(l1.second == l2.second)
97                 return false;
98             const edge &e1 = l1.first;
99             const edge &e2 = l2.first;
100             vec v1 = P[get<1>(e1)] - P[get<0>(
                e1)];
101             vec v2 = P[get<1>(e2)] - P[get<0>(
                e2)];
102             if(sign(v1.y) ≠ sign(v2.y)){
103                 return v1.y > 0;
104             } else {
105                 return sign(mulx(v1, v2)) ==
                    1;

```

```

106     };
107 };
108 };
109 vector <pair<edge, int> > E[MAXN];
110 vector <int> G[MAXG];
111 int L[MAXE], R[MAXE], W[MAXE], n, m, q, o;
112 double theta;
113 int outer;
114 void rotate(){
115     srand(time(0));
116     theta = PI * rand() / RAND_MAX;
117 }
118 int add(double x, double y){
119     srand(time(0));
120     P[++ n] = rotate(vec(x, y), theta);
121     return n;
122 }
123 int link(int u, int v, int w){
124     ++ m;
125     E[u].push_back({{u, v}, ++ o});
126     L[o] = u, R[o] = v, W[o] = w;
127     E[v].push_back({{v, u}, ++ o});
128     L[o] = v, R[o] = u, W[o] = w;
129     return m;
130 }
131 int I[MAXE];
132 int polys;
133 pair<edge, int> findleft(int l, int r){
134     auto it = lower_bound(E[r].begin(), E[
135         r].end(), make_pair(edge(r, l), 0),
136         Cmp2());
137     if(it == E[r].begin())
138         return E[r].back();
139     else
140         return *(it - 1);
141 }
142 void leftmost(){
143     for(int i = 1; i ≤ n; ++ i){
144         sort(E[i].begin(), E[i].end(),
145             Cmp2());
146     }
147     for(int p = 1; p ≤ n; ++ p){
148         for(auto &[e1, id1] : E[p]){
149             auto &[x, y] = e1;
150             if(!I[id1]){
151                 int l = x;
152                 int r = y;
153                 I[id1] = ++ polys;
154                 G[polys].push_back(id1);
155                 while(r ≠ p){
156                     auto [e2, id2] =
157                         findleft(l, r);
158                     auto [a, b] = e2;
159                     I[id2] = polys;
160                     G[polys].push_back(id2
161                         );
162                     l = r;
163                     r = b;
164                 }
165             }
166         }
167     }
168     for(int i = 1; i ≤ polys; ++ i){
169         double area = 0;
170         for(int j = 0; j < G[i].size(); ++ j
171             ){
172             area += mulx(P[L[G[i][j]]], P[
173                 R[G[i][j]]]);
174         }
175         if(area < 0)
176             outer = i;
177     }
178 }
179 void dual(){
180     Dual :: n = polys;
181     Dual :: m = 0;
182     for(int i = 1; i ≤ m; ++ i){
183         int u = I[2 * i - 1], v = I[2 * i
184             ], w = W[2 * i];
185         if(u == outer || v == outer)
186             w = 1e9L + 1;
187         ++ Dual :: m;
188         Dual :: A[Dual :: m] = u;
189         Dual :: B[Dual :: m] = v;
190         Dual :: W[Dual :: m] = w;
191     }
192     Dual :: build();
193     Dual :: outer = outer;
194 }
195 set <pair<edge, int>, Cmp1> S;
196 vector <pair<double, int> > T;
197 vector <pair<double, int> > Q;
198 double X[MAXQ], Y[MAXQ];
199 int Z[MAXQ];
200 int ask(double x, double y){
201     ++ q;
202     point p = rotate(vec(x, y), theta);
203     X[q] = p.x;
204     Y[q] = p.y;
205     return q;
206 }
207 void locate(){
208     T.clear(), Q.clear(), S.clear();
209     for(int i = 1; i ≤ q; ++ i){
210         Q.push_back(make_pair(X[i], i));
211     }
212     for(int i = 1; i ≤ polys; ++ i){
213         for(auto &e : G[i]){
214             int u = L[e];
215             int v = R[e];
216             if(P[u].x > P[v].x){
217                 T.push_back(make_pair(P[v
218                     ].x + 1e-5, e));
219                 T.push_back(make_pair(P[u
220                     ].x - 1e-5, -e));
221             }
222         }
223     }
224     sort(T.begin(), T.end());
225     sort(Q.begin(), Q.end());
226     int p1 = 0, p2 = 0;
227     scanx = -1e9;
228     Cmp1 CMP;
229     while(p1 < Q.size() || p2 < T.size()){
230         // for(auto it1 = S.begin(), it2 =
231             next(S.begin()); it2 ≠ S.end()
232             ; ++ it1, ++ it2)
233             // assert(CMP(*it1, *it2));
234         double x1 = p1 < Q.size() ? Q[p1].
235             first : 1e9;
236         double x2 = p2 < T.size() ? T[p2].
237             first : 1e9;
238         scanx = min(x1, x2);
239         if(equal(scanx, x1)){
240             auto &x = X[Q[p1].second];
241             auto &y = Y[Q[p1].second];
242             auto &z = Z[Q[p1].second];
243             P[n + 1] = point(-1e9, y);
244             P[n + 2] = point(1e9, y);
245             auto it = S.lower_bound({{n +
246                 1, n + 2}, 0});
247             if(it == S.end())
248                 z = outer;
249             else
250                 z = it → second;
251             ++ p1;
252         }
253         if(equal(scanx, x2)){
254             int g = T[p2].second;
255             if(g > 0){
256                 assert(!S.count({{L[g], R[
257                     g]}, I[g]}));
258                 S.insert({{L[g], R[g]}, I[

```

```

    g}});
243     } else {
244         g = -g;
245         assert( S.count({{L[g], R[
246             g}}, I[g]{}));
247         S.erase({{L[g], R[g]}, I[
248             g]});
249     }
250 }
251 }
252 }
253 const int MAXN = 1e5 + 3;
254 int A[MAXN], B[MAXN];
255 int main(){
256 #ifndef ONLINE_JUDGE
257     freopen("test.in", "r", stdin);
258     freopen("test.out", "w", stdout);
259 #endif
260 int n, m, q;
261 Planer :: rotate();
262 cin >> n >> m;
263 for(int i = 1; i ≤ n; ++ i){
264     double x, y;
265     cin >> x >> y;
266     Planer :: add(x, y);
267 }
268 for(int i = 1; i ≤ m; ++ i){
269     int u, v, w;
270     cin >> u >> v >> w;
271     Planer :: link(u, v, w);
272 }
273 Planer :: leftmost();
274 Planer :: dual();
275 cin >> q;
276 for(int i = 1; i ≤ q; ++ i){
277     double a1, b1, a2, b2;
278     cin >> a1 >> b1;
279     A[i] = Planer :: ask(a1, b1);
280     cin >> a2 >> b2;
281     B[i] = Planer :: ask(a2, b2);
282 }
283 Planer :: locate();
284 for(int i = 1; i ≤ q; ++ i)
285     A[i] = Planer :: Z[A[i]],
286     B[i] = Planer :: Z[B[i]];
287 for(int i = 1; i ≤ q; ++ i){
288     int ans = Dual :: solve(A[i], B[i]);
289     cout << ans << endl;
290 }

```

```

291     return 0;
292 }

```

9.8 Voronoi 图

```

1 // From Let it Rot
2 #include "2d-lines.cpp"
3 vector<line> cut(const vector<line> & o, line
4     l) {
5     vector<line> res;
6     int n = size(o);
7     for(int i = 0; i < n; ++ i) {
8         line a = o[i], b = o[(i + 1) % n], c = o[(
9             i + 2) % n];
10        int va = check(a, b, l), vb = check(b, c,
11            l);
12        if(va > 0 || vb > 0 || (va == 0 && vb ==
13            0))
14            res.push_back(b);
15        if(va ≥ 0 && vb < 0)
16            res.push_back(l);
17    }
18    return res.size() ≤ 2 ? vector<line>{}:res;
19 } // 切凸包
20 line bisector(pp a, pp b) {return line(a.x - b
21     .x, a.y - b.y, (b.norm() - a.norm()) / 2); }
22 vector<vector<line>> voronoi(vector<pp> p) {
23     int n = p.size(); auto b = p;
24     shuffle(b.begin(), b.end(), MT);
25     const db V = 1e5; // 边框大小, 重要
26     vector<vector<line>> a(n, {
27         { V, 0, V * V}, {0, V, V * V},
28         {-V, 0, V * V}, {0, -V, V * V},
29     });
30     for(int i = 0; i < n; ++ i) {
31         for(pp x : b) if((x - p[i]).abs() > EPS){
32             a[i] = cut(a[i], bisector(p[i], x));
33         }
34     }
35     return a;
36 }

```

9.9 二维基础

```

1 #include "../header.cpp"
2 const db EPS = 1e-9, PI = acos(-1);
3 bool equal(db a, db b) { return fabs(a - b) <
4     EPS; }
5 int sign(db a) { if(equal(a, 0)) return 0;
6     return a > 0 ? 1 : -1; }
7 db sqr(db x) { return x * x; }

```

```

6 struct v2{ // 二维向量
7     db x, y;
8     v2(db _x = 0, db _y = 0) : x(_x), y(_y){}
9     db norm() const {return (x * x + y * y);}
10    db abs() const {return sqrt(x * x + y * y);}
11    db arg() const {return atan2(y, x);}
12 }
13 bool operator ==(v2 a, v2 b){
14     return a.x == b.x && a.y == b.y;
15 }
16 bool operator < (v2 a, v2 b){
17     return a.x == b.x ? a.y < b.y : a.x < b.x;
18 }
19 v2 r90(v2 x) { return {-x.y, x.x}; }
20 v2 operator +(v2 a, v2 b){
21     return {a.x + b.x, a.y + b.y}; }
22 v2 operator -(v2 a, v2 b){
23     return {a.x - b.x, a.y - b.y}; }
24 v2 operator *(v2 a, db w){
25     return {a.x * w, a.y * w}; }
26 v2 operator /(v2 a, db w){
27     return {a.x / w, a.y / w}; }
28 db operator *(v2 a, v2 b) { // 叉乘, b > a 为
29     负
30     return a.x * b.y - a.y * b.x; }
31 db operator %(v2 a, v2 b) { // 点乘
32     return a.x * b.x + a.y * b.y; }
33 bool equal(v2 a, v2 b){
34     return equal(a.x, b.x) && equal(a.y, b.y);
35 }
36 using pp = v2;
37 pp rotate(pp a, db t){
38     db c = cos(t), s = sin(t);
39     return pp(a.x * c - a.y * s, a.y * c + a.x *
40         s);
41 }
42 db dis(pp a, pp b){
43     return (a - b).abs();
44 }
45 int half(pp x) { // 为 1 则 arg ≥ \pi
46     return x.y < 0 || (x.y == 0 && x.x ≤ 0);
47 }
48 // int half(pp x){return x.y < -EPS || (std::
49     fabs(x.y) < EPS && x.x < EPS);}
50 bool cmp(pp a, pp b) { // arg a < arg b
51     return half(a) == half(b) ? a * b > 0 : half
52         (b);
53 }

```

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; A[n + 1] = -1e9;
6     for(int i = 1; i ≤ n + 1; ++ i){
7         int v = 0;
8         while(!S.empty() && A[S.top()] > A[i]){
9             auto u = S.top();
10            R[u] = v, v = u, S.pop();
11        }
12        L[i] = v, S.push(i);
13    }
14 }

```

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{int id, a, b, c;}A[MAXN], B[MAXN];
3 bool cmp(Node a, Node b){
4     return a.a == b.a ? (a.b == b.b ? (a.c == b.c ? a.c < b.c : a.id < b.id) : a.b < b.b) : a.a < b.a;
5 }
6 int K[MAXN], H[MAXN], n, m, D[MAXM];
7 namespace BIT{
8     void modify(int x, int w); // 加到 m
9     void query(int x, int &r);
10 }
11 using BIT :: modify, BIT :: query;
12 void cdq(int l, int r){
13     if(l ≠ r){
14         int t = l + r >> 1;
15         cdq(l, t), cdq(t + 1, r);
16         int p = l, q = t + 1, u = l;
17         while(p ≤ t && q ≤ r){
18             if(A[p].b ≤ A[q].b)
19                 modify(A[p].c, 1), B[u++] = A[p++];
20             else
21                 query(A[q].c, K[A[q].id]), B[u++] = A[q++];
22         }
23         while(p ≤ t) modify(A[p].c, 1), B[u++] = A[p++];

```

```

24     while(q ≤ r) query(A[q].c, K[A[q].id]), B
        [u++] = A[q++];
25     up(l, t, i) modify(A[i].c, -1);
26     up(l, r, i) A[i] = B[i];
27 }
28 }
29 int main(){
30     n = qread(), m = qread();
31     up(1, n, i) A[i].id = i, A[i].a = qread(), A
        [i].b = qread(), A[i].c = qread();
32     sort(A + 1, A + 1 + n, cmp), cdq(1, n);
33     sort(A + 1, A + 1 + n, cmp);
34     dn(n, 1, i){
35         if(A[i].a == A[i + 1].a && A[i].b == A[i +
            1].b && A[i].c == A[i + 1].c)
36             K[A[i].id] = K[A[i + 1].id];
37             H[K[A[i].id]] ++;
38         }
39     up(0, n - 1, i) printf("%d\n", H[i]);
40     return 0;
41 }

```

10.3 日期公式

```

1 // 从公元 1 年 1 月 1 日经过了多少天
2 int getday(int y, int m, int d) {
3     if(m < 3) -- y, m += 12;
4     return (365 * y + y / 4 - y / 100 + y / 400
        + (153 * (m - 3) + 2) / 5 + d - 307);
5 }
6 // 公元 1 年 1 月 1 日往后数 n 天是哪一天
7 void date(int n, int &y, int &m, int &d) {
8     n += 429 + ((4 * n + 1227) / 146097 + 1) * 3
        / 4;
9     y = (4 * n - 489) / 1461, n -= y * 1461 / 4;
10    m = (5 * n - 1) / 153, d = n - m * 153 / 5;
11    if (--m > 12) m -= 12, ++y;
12 }

```

10.4 misc

```

1 #pragma GCC optimize("Ofast", "unroll-loops")
2 #pragma GCC target("sse,sse2,sse3,ssse3,sse4,
    popcnt,abm,mmx,avx,avx2")
3 #pragma pack(1) // 卡空间, 默认 = 8

```

10.5 pbds 库

```

1 #include "../header.cpp"
2 #include <ext/pb_ds/assoc_container.hpp>
3 #include <ext/pb_ds/tree_policy.hpp> // 树

```

```

4 #include <ext/pb_ds/priority_queue.hpp> // 堆
5 using namespace __gnu_pbds;
6 // insert, erase, order_of_key, find_by_order,
7 // [lower|upper]_bound, join, split, size
8 __gnu_pbds::tree<int, null_type, less<int>,
9     rb_tree_tag,
10     tree_order_statistics_node_update> T;
11 // push, pop, top, size, empty, modify, erase,
12 // join 其中 modify 修改寄存器, join 合并堆
13 // 还可以用 rc_binomial_heap_tag
14 __gnu_pbds::priority_queue<int, less<int>,
15     pairing_heap_tag> Q1, Q2;

```

10.6 Python 技巧

```

1 import random
2 random.normalvariate(0.5, 0.1) # 正态分布
3 l = [str(i) for i in range(9)] # 列表
4 sorted(l), min(l), max(l), len(l)
5 random.shuffle(l)
6 l.sort(key=lambda x:x ^ 1,reverse=True)
7 from functools import cmp_to_key
8 l.sort(key=cmp_to_key(lambda x,y:(y^1)-(x^1)))
9 from itertools import *
10 for i in product('ABCD', repeat=2):
11     pass # AA AB AC AD BA BB BC BD CA CB CC CD
        DA DB DC DD
12 for i in permutations('ABCD', repeat=2):
13     pass # AB AC AD BA BC BD CA CB CD DA DB DC
14 for i in combinations('ABCD', repeat=2):
15     pass # AB AC AD BC BD CD
16 for i in combinations_with_replacement('ABCD',
    repeat=2):
17     pass # AA AB AC AD BB BC BD CC CD DD
18 from fractions import Fraction
19 a.numerator, a.denominator, str(a)
20 a = Fraction(0.233).limit_denominator(1000)
21 from decimal import Decimal, getcontext,
    FloatOperation
22 getcontext().prec = 100
23 getcontext().rounding = getattr(decimal, '
    ROUND_HALF_EVEN')
24 # default; other: FLOOR, CEILING, DOWN, ...
25 getcontext().traps[FloatOperation] = True
26 Decimal((0, (1, 4, 1, 4), -3)) # 1.414
27 a = Decimal(1<<31) / Decimal(100000)
28 print(f"{a:.9f}") # 21474.83648
29 print(a.sqrt(), a.ln(), a.log10(), a.exp(), a
    ** 2)
30 a = 1-2j
31 print(a.real, a.imag, a.conjugate())
32 import sys, atexit, io

```

```

33 _INPUT_LINES = sys.stdin.read().splitlines()
34 input = iter(_INPUT_LINES).__next__
35 _OUTPUT_BUFFER = io.StringIO()
36 sys.stdout = _OUTPUT_BUFFER
37 @atexit.register
38 def write():
39     sys.__stdout__.write(_OUTPUT_BUFFER.getvalue())

```

10.7 快速测试

```

1 export CXXFLAGS='-g-Wall-fsanitize=address,
   undefined-Dzwb-std=gnu++20'
2 mk() { g++ -O2 -Dzwb -std=gnu++20 1.cpp -o1; }
3 ulimit -s 1048576
4 ulimit -v 1048576

```

10.8 自适应辛普森

```

1 #include "../header.cpp"
2 db simpson(db (*f)(db), db l, db r){
3     return (r - l) / 6 *
4         (f(l) + 4 * f((l + r) / 2) + f(r));
5 }
6 db adapt_simpson(db (*f)(db), db l, db r, db
   EPS, int step){
7     db mid = (l + r) / 2;
8     db w0 = simpson(f, l, r), w1 = simpson(f, l,
   mid), w2 = simpson(f, mid, r);
9     return fabs(w0 - w1 - w2) < EPS && step < 0?
10     w1 + w2 :
11     adapt_simpson(f, l, mid, EPS, step - 1) +
12     adapt_simpson(f, mid, r, EPS, step - 1);
13 }

```

10.9 模拟退火

10.9.1 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 db T0 = 2e3, Tk = 1e-14, delta = 0.993,
   R = 1e-3;
3 mt19937 MT(114514);
4 db distance(db x, db y, db a, db b){
5     return sqrt(pow(a - x, 2) + pow(b - y, 2));
6 }
7 const int MAXN = 1e3 + 3;
8 db X[MAXN], Y[MAXN], W[MAXN]; int n;

```

```

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

```

10.10 对拍脚本

```

1 while true; do
2     ./gen > 1.in; ./naive < 1.in > std.out; ./a
   < 1.in > 1.out
3     if diff 1.out std.out; then echo ac; else
   echo wa; break fi
4 done

```

10.11 伪随机生成

```

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

```

11 公共头文件

```

1 #include <bits/stdc++.h>
2 using namespace std;
3 #define up(l, r, i) for(int i = l, END##i = r;
   i ≤ END##i; ++ i)
4 #define dn(r, l, i) for(int i = r, END##i = l;
   i ≥ END##i; -- i)
5 using i64 = long long;
6 using i80 = __int128;
7 using f80 = long double;
8 using u32 = unsigned;
9 using u64 = unsigned long long;
10 const int INF = 1e9;
11 const i64 INFL = 1e18;
12 const int MAXN = 10 + 3, MAXM = 10 + 3;
13 const int MOD = 998244353, MD = MOD;
14 const int inv2 = (MOD + 1) / 2;
15 using ll = i64;
16 using db = double;
17 using ld = long double;
18 int qread();
19 int power(int a, int b);
20 int power(int a, int b, int p);
21 ll qpow(ll a, ll b);
22 ll inv(ll x);
23 mt19937 MT;

```

$n \leq$	10	100	10^3	10^4	10^5	10^6	10^7	10^8	10^9	10^{10}	10^{11}	10^{12}	10^{13}	10^{14}	10^{15}	10^{16}	10^{17}	10^{18}
$\max \omega(n)$	2	3	4	5	6	7	8	8	9	10	10	11	12	12	13	13	14	15
$\max d(n)$	4	12	32	64	128	240	448	768	1344	2304	4032	6720	10752	17280	26880	41472	64512	103680
$\pi(n)$	4	25	168	1229	9592	78498	664579	5761455	5.08e7	4.55e8	4.12e9	3.7e10	$n/\ln(n)$					
$n =$	2	3	4	5	6	7	8	9	10	11	12	15	20	25	30	40	50	114
$\log_{10} n$	0.301	0.477	0.602	0.698	0.778	0.845	0.903	0.954	1	1.041	1.079	1.176	1.301	1.398	1.477			
$C(n,n/2)$	2	3	6	10	20	35	70	126	252	462	924	6435	184756	5200300	155117520			
$\text{LCM}(1 \dots n)$	2	6	12	60	60	420	840	2520	2520	27720	27720	360360	232792560	26771144400	1.444e14			
P_n	2	3	5	7	11	15	22	30	42	56	77	176	627	1958	5604	37338	204226	10^9

- 质数：918733, 940649, 923641, 992402371, 903936311, 977499077, 908370375016605079, 973150773996892879, 990191901472122599, $10^{18} + 3$;
- NTT 模数：167772161, 1004535809, 2924438830427668481 = $174310137655 \times 2^{24} + 1$.

二项式系数

$$\sum_{k=0}^n \binom{k}{m} = \binom{n+1}{m+1}, \sum_{k=0}^r \binom{r-k}{m} \binom{s+k}{n} = \binom{r+s+1}{m+n+1}$$

斐波那契数

$$\sum_{k=1}^n f_k^2 = f_n f_{n+1}, \sum_{k=1}^n k f_k = n f_{n+2} - f_{n+3} + 2, \sum_{k=0}^n f_k f_{n-k} = \frac{1}{5}(n-1)f_n + \frac{2}{5}n f_{n-1}$$
$$f_{n+k} = f_n f_{k+1} + f_{n-1} f_k, (-1)^k f_{n-k} = f_n f_{k-1} - f_{n-1} f_k$$

```
1 def fib(n): # F(n), F(n + 1)
2     if not n: return (0, 1)
3     a, b = fib(n >> 1); c = a * (2 * b - a); d = a * a + b * b
4     return (d, c + d) if n & 1 else (c, d)
```

斯特林数

第一类 n 个元素集合分作 k 个非空轮换方案数。

$$\begin{bmatrix} n+1 \\ k \end{bmatrix} = n \begin{bmatrix} n \\ k \end{bmatrix} + \begin{bmatrix} n \\ k-1 \end{bmatrix}, x^{\underline{n}} = \sum_k \begin{bmatrix} n \\ k \end{bmatrix} (-1)^{n-k} x^k, x^{\overline{n}} = \sum_k \begin{bmatrix} n \\ k \end{bmatrix} x^k$$

第二类 把 n 个元素集合分作 k 个非空子集方案数。

$$\left\{ \begin{matrix} n+1 \\ k \end{matrix} \right\} = k \left\{ \begin{matrix} n \\ k \end{matrix} \right\} + \left\{ \begin{matrix} n \\ k-1 \end{matrix} \right\}, x^n = \sum_k \left\{ \begin{matrix} n \\ k \end{matrix} \right\} x^{\underline{k}} = \sum_k \left\{ \begin{matrix} n \\ k \end{matrix} \right\} (-1)^{n-k} x^{\overline{k}}$$
$$m! \left\{ \begin{matrix} n \\ m \end{matrix} \right\} = \sum_k \binom{m}{k} k^n (-1)^{m-k}$$

求法 对于列，使用下述 EGF；对于行，第一类使用 $x^{\overline{n}}$ ，第二类使用二项式反演。

$$\sum_{n=0}^{\infty} \begin{bmatrix} n \\ k \end{bmatrix} \frac{x^n}{n!} = \frac{x^k}{k!} \left(\frac{\ln(1-x)}{x} \right)^k, \sum_{n=0}^{\infty} \left\{ \begin{matrix} n \\ k \end{matrix} \right\} \frac{x^n}{n!} = \frac{x^k}{k!} \left(\frac{e^x - 1}{x} \right)^k$$

欧拉数

恰好有 m 次上升 ($p_i > p_{i-1}$) 的长为 n 的排列个数记为 $\langle \begin{smallmatrix} n \\ m-1 \end{smallmatrix} \rangle$ 。

$$\left\langle \begin{matrix} n \\ k \end{matrix} \right\rangle = (k+1) \left\langle \begin{matrix} n-1 \\ k \end{matrix} \right\rangle + (n-k) \left\langle \begin{matrix} n-1 \\ k-1 \end{matrix} \right\rangle, \left\langle \begin{matrix} n \\ m \end{matrix} \right\rangle = \sum_{k=0}^m \binom{n+1}{k} (m+1-k)^n (-1)^k$$

贝尔数 (1, 1, 2, 5, 15, 52, 203, 877, 4140)

n 个元素集合划分的方案数。

$$B_n = \sum_{k=1}^n \left\{ \begin{matrix} n \\ k \end{matrix} \right\}, B_{n+1} = \sum_{k=0}^n \binom{n}{k} B_k, B_{p^m+n} \equiv m B_n + B_{n+1} \pmod{p}, B(x) = e^{e^x-1}$$

拆分数

将整数 n 拆分成若干个正整数之和的方案数为 $p(n)$ 。

```
1 def penta(k):
2     return k*(3*k-1)//2
3 def compute_partition(goal):
4     p = [1]
5     for n in range(1,goal+1):
6         p.append(0)
7         for k in range(1,n+1):
8             c = (-1)**(k+1)
9             for t in [penta(k), penta(-k)]:
10                 if (n-t) >= 0: p[n] = p[n] + c*p[n-t]
11     return p
```