
Aerial Ace

Team Reference Material
(unlimited version)

2.0



顾逸
陈彤
陈雪阳



目录

1 Graph Theory	1
1.1 2-SAT (ct)	1
1.2 割点与桥 (ct)	2
1.3 Steiner tree (lhy)	3
1.4 K 短路 (lhy)	3
1.5 最大团 (Nightfall)	7
1.6 极大团计数 (Nightfall)	9
1.7 三元环计数 (cxy)	9
1.8 二分图最大匹配 (lhy)	10
1.9 一般图最大匹配 (lhy)	11
1.10 KM 算法 (Nightfall)	12
1.11 最小树形图 (Nightfall)	13
1.12 支配树 (ct,Nightfall)	15
1.13 虚树 (ct)	17
1.14 点分治 (ct)	18
1.15 树上倍增 (ct)	19
1.16 树哈希 (lhy,Luna)	20
1.17 Link-Cut Tree (ct)	21
1.18 圆方树 (ct)	23
1.19 无向图最小割 (Nightfall)	24
1.20 网络流 (lhy,ct)	25
1.21 欧拉回路 (cxy)	27
1.22 图论知识 (gy,lhy)	28
2 Math	33
2.1 int64 相乘取模 (Durandal)	33
2.2 ex-Euclid (gy)	33
2.3 中国剩余定理 (Durandal)	34
2.4 线性同余不等式 (Durandal)	34
2.5 平方剩余 (Nightfall)	34
2.6 组合数 (Nightfall)	35
2.7 高斯消元 (ct)	36
2.8 Miller Rabin & Pollard Rho (gy)	36
2.9 $O(m^2 \log n)$ 线性递推 (lhy)	38
2.10 Berlekamp Massey (NightFall)	38
2.11 线性基 (ct)	40
2.12 多项式 (lhy,ct,gy)	40
2.13 筛 (ct,cxy,Nightfall)	44
2.14 BSGS (ct,Durandal)	53
2.15 直线下整点个数 (gy)	54
2.16 Pell equation (gy)	55
2.17 单纯形 (gy)	55
2.18 数学知识 (gy)	56
3 Geometry	61
3.1 点、直线、圆 (gy)	61
3.2 平面最近点对 (Grimoire)	65
3.3 凸包游戏 (Grimoire)	65
3.4 半平面交 (Grimoire)	67
3.5 点在多边形内 (Grimoire)	68
3.6 最小圆覆盖 (Grimoire)	68

3.7 最小球覆盖 (Grimoire)	69
3.8 圆并 (gy)	70
3.9 圆与多边形并 (Grimoire)	71
3.10 三角剖分 (Grimoire)	72
3.11 三维几何基础 (Grimoire)	74
3.12 三维凸包 (Grimoire)	75
3.13 三维绕轴旋转 (gy)	77
3.14 几何知识 (gy)	77
4 String	81
4.1 KMP (ct)	81
4.2 AC 自动机 (ct)	81
4.3 Lydon Word Decomposition (Nightfall)	82
4.4 后缀数组 (ct)	83
4.5 后缀自动机 (ct)	83
4.6 Manacher (ct)	84
4.7 回文树 (ct)	84
4.8 最小表示法 (ct)	85
4.9 字符串知识 (Nightfall)	86
5 Data Structure	87
5.1 莫队 (ct)	87
5.2 ST 表 (ct)	87
5.3 长链剖分 (ct)	88
5.4 DSU (ct)	89
5.5 带权并查集 (ct)	89
5.6 可并堆 (ct)	90
5.7 线段树 (ct)	91
5.8 Splay (ct)	95
5.9 Treap (ct)	98
5.10 可持久化平衡树 (ct)	100
5.11 二进制分组 (ct)	103
5.12 CDQ 分治 (ct)	104
5.13 挑战 (ct)	105
5.14 斜率优化 (ct)	105
5.15 树分块 (ct)	108
5.16 KD tree (lhy,cxy)	110
5.17 DLX (Nightfall)	114
5.18 数据结构知识 (gy)	115
6 Others	117
6.1 Config (gy)	117
6.2 模拟退火 (ct)	117
6.3 Simpson 积分 (gy)	118
6.4 Zeller Congruence (gy)	118
6.5 博弈论模型 (gy)	118
6.6 C++ Template (Durandal,gy,ct)	120
6.7 Python Template (gy)	121
6.8 Java Template (gy)	121
6.9 积分表 (integral-table.com)	124
6.10 环境测试 (gy)	126
A Additional Documents	127
A.1 超自然数和不平等博弈	127
A.2 生成函数	140
A.3 多项式操作	150
A.4 多项式操作	158

Graph Theory

1.1 2-SAT (ct)

```
1 struct Edge {
2     Edge *next;
3     int to;
4 } *last[maxn << 1], e[maxn << 2], *ecnt = e;
5 inline void link(int a, int b)
6 {
7     *++ecnt = (Edge) {last[a], b}; last[a] = ecnt;
8 }
9 int dfn[maxn], low[maxn], timer, st[maxn], top, id[maxn], colcnt, n;
10 bool fail, used[maxn];
11 void tarjan(int x, int fa)
12 {
13     dfn[x] = low[x] = ++timer; st[++top] = x;
14     for (R Edge *iter = last[x]; iter; iter = iter -> next)
15         if (iter -> to != fa)
16         {
17             if (!dfn[iter -> to])
18             {
19                 tarjan(iter -> to, x);
20                 cmin(low[x], low[iter -> to]);
21             }
22             else if (!id[iter -> to]) cmin(low[x], dfn[iter -> to]);
23         }
24     if (dfn[x] == low[x])
25     {
26         ++colcnt; bool flag = 1;
27         for ( ; )
28         {
29             int now = st[top--];
30             id[now] = colcnt;
31             if (now <= 2 * n)
32             {
33                 flag &= !used[id[now] <= n ? now + n : now - n];
34                 now <= n ? fail |= (id[now + n] == id[now]) : fail |= (id[now - n] == id[now]);
35             }
36             if (now == x) break;
37         }
38         used[colcnt] = flag;
39     }
40 }
41 int ans[maxn], tot;
42 int main()
43 {
44     /*
45      build your graph here.
46 */
```

```

47     for (R int i = 1; !fail && i <= n; ++i) if (!dfn[i]) tarjan(i, 0);
48     if (fail)
49     {
50         puts("Impossible");
51         return 0;
52     }
53     for (R int i = 1; i <= n; ++i) if (used[id[i]]) ans[++tot] = i;
54     printf("%d\n", tot);
55     std::sort(ans + 1, ans + tot + 1);
56     for (R int i = 1; i <= tot; ++i) printf("%d ", ans[i]);
57     return 0;
58 }
```

1.2 割点与桥 (ct)

割点

```

1 int dfn[maxn], low[maxn], timer, ans, num;
2 void tarjan(int x, int fa)
3 {
4     dfn[x] = low[x] = ++timer;
5     for (Edge *iter = last[x]; iter; iter = iter -> next)
6         if (iter -> to != fa)
7         {
8             if (!dfn[iter -> to])
9             {
10                 tarjan(iter -> to, x);
11                 cmin(low[x], low[iter -> to]);
12                 if (dfn[x] <= low[iter -> to])
13                 {
14                     cut[x] = 1;
15                     if (!fa && dfn[x] < low[iter -> to]) num = 233;
16                     else if (!fa) ++num;
17                 }
18             }
19             else cmin(low[x], dfn[iter -> to]);
20         }
21 }
22 int main()
23 {
24     for (int i = 1; i <= n; ++i)
25         if (!dfn[i])
26         {
27             num = 0;
28             tarjan(i, 0);
29             if (num == 1) cut[i] = 0;
30         }
31 }
```

桥

```

1 int dfn[maxn], low[maxn], timer;
2 void tarjan(int x, int fa)
3 {
4     dfn[x] = low[x] = ++timer;
5     for (R Edge *iter = last[x]; iter; iter = iter -> next)
6         if (iter -> to != fa)
7         {
```

```

8     if (!dfn[iter -> to])
9     {
10        dfs(iter -> to, x);
11        cmin(low[x], low[iter -> to]);
12        if (dfn[x] < low[iter -> to]) ans[x][iter -> to] = ans[iter -> to][x] = 1;
13    }
14    else cmin(low[x], dfn[iter -> to]);
15  }
16}

```

1.3 Steiner tree (lhy)

```

1 void Steiner_Tree()
2 {
3   memset(f, 0x3f, sizeof(f));
4   for(int i = 1; i <= n; i++)
5     f[0][i] = 0;
6   for(int i = 1; i <= p; i++)
7     f[1 << (i - 1)][idx[i]] = 0;
8   int S = 1 << p;
9   for(int s = 1; s < S; s++)
10  {
11    for(int i = 1; i <= n; i++)
12    {
13      for(int k = (s - 1) & s; k; k = (k - 1) & s)
14        f[s][i] = min(f[s][i], f[k][i] + f[s ^ k][i]);
15    }
16    SPFA(f[s]);
17  }
18  int ans = inf;
19  for(int i = 1; i <= n; i++)
20    ans = min(ans, f[S - 1][i]);
21}

```

1.4 K 短路 (lhy)

```

1 const int MAXNODE = MAXN + MAXM * 2;
2
3 bool used[MAXN];
4 int n, m, cnt, S, T, Kth, N, TT;
5 int rt[MAXN], seq[MAXN], adj[MAXN], from[MAXN], dep[MAXN];
6 LL dist[MAXN], w[MAXM], ans[MAXK];
7
8 struct GivenEdge{
9   int u, v, w;
10  GivenEdge() {};
11  GivenEdge(int _u, int _v, int _w) : u(_u), v(_v), w(_w){};
12}edge[MAXM];
13
14 struct Edge{
15   int v, nxt, w;
16   Edge() {};
17   Edge(int _v, int _nxt, int _w) : v(_v), nxt(_nxt), w(_w) {};
18}e[MAXM];
19
20 inline void addedge(int u, int v, int w)
21 {
22   e[++cnt] = Edge(v, adj[u], w); adj[u] = cnt;

```

```

23 }
24
25 void dij(int S)
26 {
27     for(int i = 1; i <= N; i++)
28     {
29         dist[i] = INF;
30         dep[i] = 0x3f3f3f3f;
31         used[i] = false;
32         from[i] = 0;
33     }
34     static priority_queue<pair<LL, int>, vector<pair<LL, int>>, greater<pair<LL, int>> hp;
35     while(!hp.empty()) hp.pop();
36     hp.push(make_pair(dist[S] = 0, S));
37     dep[S] = 1;
38     while(!hp.empty())
39     {
40         pair<LL, int> now = hp.top();
41         hp.pop();
42         int u = now.second;
43         if(used[u]) continue;
44         else used[u] = true;
45         for(int p = adj[u]; p; p = e[p].nxt)
46         {
47             int v = e[p].v;
48             if(dist[u] + e[p].w < dist[v])
49             {
50                 dist[v] = dist[u] + e[p].w;
51                 dep[v] = dep[u] + 1;
52                 from[v] = p;
53                 hp.push(make_pair(dist[v], v));
54             }
55         }
56     }
57     for(int i = 1; i <= m; i++) w[i] = 0;
58     for(int i = 1; i <= N; i++)
59     {
60         if(from[i]) w[from[i]] = -1;
61     }
62     for(int i = 1; i <= m; i++)
63     {
64         if(~w[i] && dist[edge[i].u] < INF && dist[edge[i].v] < INF)
65         {
66             w[i] = -dist[edge[i].u] + (dist[edge[i].v] + edge[i].w);
67         }
68         else
69         {
70             w[i] = -1;
71         }
72     }
73 }
74 inline bool cmp_dep(int p, int q)
75 {
76     return dep[p] < dep[q];
77 }
78 struct Heap{
79     LL key;
80     int id, lc, rc, dist;
81     Heap() {};
82     Heap(LL k, int i, int l, int r, int d) : key(k), id(i), lc(l), rc(r), dist(d) {};
83     inline void clear()

```

```

84     {
85         key = 0;
86         id = lc = rc = dist = 0;
87     }
88 }hp[MAXNODE];
89
90 inline int merge_simple(int u, int v)
91 {
92     if(!u)return v;
93     if(!v)return u;
94     if(hp[u].key > hp[v].key)
95     {
96         swap(u, v);
97     }
98     hp[u].rc = merge_simple(hp[u].rc, v);
99     if(hp[hp[u].lc].dist < hp[hp[u].rc].dist)
100    {
101        swap(hp[u].lc, hp[u].rc);
102    }
103    hp[u].dist = hp[hp[u].rc].dist + 1;
104    return u;
105}
106
107 inline int merge_full(int u, int v)
108 {
109     if(!u)return v;
110     if(!v)return u;
111     if(hp[u].key > hp[v].key)
112     {
113         swap(u, v);
114     }
115     int nownode = ++cnt;
116     hp[nownode] = hp[u];
117     hp[nownode].rc = merge_full(hp[nownode].rc, v);
118     if(hp[hp[nownode].lc].dist < hp[hp[nownode].rc].dist)
119     {
120         swap(hp[nownode].lc, hp[nownode].rc);
121     }
122     hp[nownode].dist = hp[hp[nownode].rc].dist + 1;
123     return nownode;
124 }
125
126 priority_queue<pair<LL, intint>, greater<pair<LL, int>> Q;
127
128 int main()
129 {
130     while(scanf("%d%d", &n, &m) != EOF)
131     {
132         scanf("%d%d%d%d", &S, &T, &Kth, &TT);
133         for(int i = 1; i <= m; i++)
134         {
135             int u, v, w;
136             scanf("%d%d%d", &u, &v, &w);
137             edge[i] = {u, v, w};
138         }
139         N = n;
140         memset(adj, 0, sizeof(*adj) * (N + 1));
141         cnt = 0;
142         for(int i = 1; i <= m; i++)
143             addedge(edge[i].v, edge[i].u, edge[i].w);
144         dij(T);

```

```

145     if(dist[S] > TT)
146     {
147         puts("Whitesnake!");
148         continue;
149     }
150     for(int i = 1; i <= N; i++)
151         seq[i] = i;
152     sort(seq + 1, seq + N + 1, cmp_dep);
153
154     cnt = 0;
155     memset(adj, 0, sizeof(*adj) * (N + 1));
156     memset(rt, 0, sizeof(*rt) * (N + 1));
157     for(int i = 1; i <= m; i++)
158         addedge(edge[i].u, edge[i].v, edge[i].w);
159     rt[T] = cnt = 0;
160     hp[0].dist = -1;
161     for(int i = 1; i <= N; i++)
162     {
163         int u = seq[i], v = edge[from[u]].v;
164         rt[u] = 0;
165         for(int p = adj[u]; p; p = e[p].nxt)
166         {
167             if(~w[p])
168             {
169                 hp[++cnt] = Heap(w[p], p, 0, 0, 0);
170                 rt[u] = merge_simple(rt[u], cnt);
171             }
172         }
173         if(i == 1)continue;
174         rt[u] = merge_full(rt[u], rt[v]);
175     }
176     while(!Q.empty())Q.pop();
177     Q.push(make_pair(dist[S], 0));
178     edge[0].v = S;
179     for(int kth = 1; kth <= Kth; kth++)
180     {
181         if(Q.empty())
182         {
183             ans[kth] = -1;
184             continue;
185         }
186         pair<LL, int> now = Q.top(); Q.pop();
187         ans[kth] = now.first;
188         int p = now.second;
189         if(hp[p].lc)
190         {
191             Q.push(make_pair(+hp[hp[p].lc].key + now.first - hp[p].key, hp[p].lc));
192         }
193         if(hp[p].rc)
194         {
195             Q.push(make_pair(+hp[hp[p].rc].key + now.first - hp[p].key, hp[p].rc));
196         }
197         if(rt[edge[hp[p].id].v])
198         {
199             Q.push(make_pair(hp[rt[edge[hp[p].id].v]].key + now.first, rt[edge[hp[p].id].v]));
200         }
201     }
202     if(ans[Kth] == -1 || ans[Kth] > TT)
203     {
204         puts("Whitesnake!");
205     }

```

```

206     else
207     {
208         puts("yareyaredawa");
209     }
210 }
211 }
```

1.5 最大团 (Nightfall)

时间复杂度建议 $n \leq 150$

```

1 typedef bool BB[N];
2
3 struct Maxclique {
4     const BB *e;
5     int pk, level;
6     const float Tlimit;
7
8     struct Vertex {
9         int i, d;
10
11     Vertex(int i) : i(i), d(0) {}
12 };
13
14 typedef vector <Vertex> Vertices;
15 Vertices V;
16 typedef vector<int> ColorClass;
17 ColorClass QMAX, Q;
18 vector <ColorClass> C;
19
20 static bool desc_degree(const Vertex &vi, const Vertex &vj) { return vi.d > vj.d; }
21
22 void init_colors(Vertices &v) {
23     const int max_degree = v[0].d;
24     for (int i = 0; i < (int) v.size(); i++)
25         v[i].d = min(i, max_degree) + 1;
26 }
27
28 void set_degrees(Vertices &v) {
29     for (int i = 0, j; i < (int) v.size(); i++)
30         for (v[i].d = j = 0; j < (int) v.size(); j++)
31             v[i].d += e[v[i].i][v[j].i];
32 }
33
34 struct StepCount {
35     int i1, i2;
36
37     StepCount() : i1(0), i2(0) {}
38 };
39
40 vector <StepCount> S;
41
42 bool cut1(const int pi, const ColorClass &A) {
43     for (int i = 0; i < (int) A.size(); i++)
44         if (e[pi][A[i]]) return true;
45     return false;
46 }
47
48 void cut2(const Vertices &A, Vertices &B) {
49     for (int i = 0; i < (int) A.size() - 1; i++)
```

```

50     if (e[A.back().i][A[i].i]) B.push_back(A[i].i);
51 }
52
53 void color_sort(Vertices &R) {
54     int j = 0, maxno = 1;
55     int min_k = max((int) QMAX.size() - (int) Q.size() + 1, 1);
56     C[1].clear(), C[2].clear();
57     for (int i = 0; i < (int) R.size(); i++) {
58         int pi = R[i].i, k = 1;
59         while (cut1(pi, C[k])) k++;
60         if (k > maxno) maxno = k, C[maxno + 1].clear();
61         C[k].push_back(pi);
62         if (k < min_k) R[j++].i = pi;
63     }
64     if (j > 0) R[j - 1].d = 0;
65     for (int k = min_k; k <= maxno; k++)
66         for (int i = 0; i < (int) C[k].size(); i++)
67             R[j].i = C[k][i], R[j++].d = k;
68 }
69
70 void expand_dyn(Vertices &R) {
71     S[level].i1 = S[level].i1 + S[level - 1].i1 - S[level].i2;
72     S[level].i2 = S[level - 1].i1;
73     while ((int) R.size()) {
74         if ((int) Q.size() + R.back().d > (int) QMAX.size()) {
75             Q.push_back(R.back().i);
76             Vertices Rp;
77             cut2(R, Rp);
78             if ((int) Rp.size()) {
79                 if ((float) S[level].i1 / ++pk < Tlimit) degree_sort(Rp);
80                 color_sort(Rp);
81                 S[level].i1++, level++;
82                 expand_dyn(Rp);
83                 level--;
84             } else if ((int) Q.size() > (int) QMAX.size()) QMAX = Q;
85             Q.pop_back();
86         } else return;
87         R.pop_back();
88     }
89 }
90
91 void mcqdyn(int *maxclique, int &sz) {
92     set_degrees(V);
93     sort(V.begin(), V.end(), desc_degree);
94     init_colors(V);
95     for (int i = 0; i < (int) V.size() + 1; i++) S[i].i1 = S[i].i2 = 0;
96     expand_dyn(V);
97     sz = (int) QMAX.size();
98     for (int i = 0; i < (int) QMAX.size(); i++) maxclique[i] = QMAX[i];
99 }
100
101 void degree_sort(Vertices &R) {
102     set_degrees(R);
103     sort(R.begin(), R.end(), desc_degree);
104 }
105
106 Maxclique(const BB *conn, const int sz, const float tt = .025)
107     : pk(0), level(1), Tlimit(tt) {
108     for (int i = 0; i < sz; i++) V.push_back(Vertex(i));
109     e = conn, C.resize(sz + 1), S.resize(sz + 1);
110 }
```

```

111 } ;
112
113 BB e[N];
114 int ans, sol[N];
115 for (...) e[x][y]=e[y][x]=true;
116 Maxclique mc(e, n);
117 mc.mcqdyn(sol, ans); // 全部 0 下标
118 for (int i = 0; i<ans; ++i) cout << sol[i] << endl;

```

1.6 极大团计数 (Nightfall)

0-based, 需删除自环

极大团计数, 最坏情况 $O(3^{n/3})$

```

1 ll ans;
2 ull E[64];
3 #define bit(i) (1ULL << (i))
4
5 void dfs(ull P, ull X, ull R) { // 不需要方案时可去掉 R 相关语句
6     if (!P && !X) {
7         ++ans;
8         sol.pb(R);
9         return;
10    }
11    ull Q = P & ~E[__builtin_ctzll(P | X)];
12    for (int i; i = __builtin_ctzll(Q), Q; Q &= ~bit(i)) {
13        dfs(P & E[i], X & E[i], R | bit(i));
14        P &= ~bit(i), X |= bit(i);
15    }
16 }
17
18 ans = 0;
19 dfs(n == 64 ? ~0ULL : bit(n) - 1, 0, 0);

```

1.7 三元环计数 (cxy)

```

1 #include <bits/stdc++.h>
2 using namespace std;
3 const int maxn = 1e5 + 10;
4
5 long long ternary_ring(int n, const vector<pair<int, int>>&E) {
6     static vector<int> G[maxn];
7     static int visby[maxn], deg[maxn];
8     for (int i = 0; i <= n; i++)
9         visby[i] = deg[i] = 0, G[i].clear();
10    for (auto e : E) deg[e.first]++;
11    for (auto e : E) deg[e.first] > deg[e.second] ?
12        G[e.first].push_back(e.second) : G[e.second].push_back(e.first);
13    long long ans = 0;
14    for (int u = 0; u <= n; u++) {
15        for (auto v : G[u]) visby[v] = u;
16        for (auto v : G[u]) for (auto w : G[v])
17            if (w[visby] == u) ans++; // (u, v, w) unordered
18    }
19    return ans;
20} // O(n + m \times \sqrt(m))
21
22 int main() {}

```

1.8 二分图最大匹配 (lhy)

左侧 n 个点, 右侧 m 个点, 1-based, 初始化将 $matx$ 和 $maty$ 置为 0

```

1 int BFS()
2 {
3     int flag = 0, h = 0, l = 0;
4     for(int i = 1; i <= k; i++)
5         dy[i] = 0;
6     for(int i = 1; i <= n; i++)
7     {
8         dx[i] = 0;
9         if(!matx[i])q[++l] = i;
10    }
11    while(h < l)
12    {
13        int x = q[++h];
14        for(int i = son[x]; i; i = edge[i].next)
15        {
16            int y = edge[i].y;
17            if(!dy[y])
18            {
19                dy[y] = dx[x] + 1;
20                if(!maty[y])flag = 1;
21                else
22                {
23                    dx[maty[y]] = dx[x] + 2;
24                    q[++l] = maty[y];
25                }
26            }
27        }
28    }
29    return flag;
30 }
31
32 int DFS(int x)
33 {
34     for(int i = son[x]; i; i = edge[i].next)
35     {
36         int y = edge[i].y;
37         if(dy[y] == dx[x] + 1)
38         {
39             dy[y] = 0;
40             if(!maty[y] || DFS(maty[y]))
41             {
42                 matx[x] = y, maty[y] = x;
43                 return 1;
44             }
45         }
46     }
47     return 0;
48 }
49
50 void Hopcroft()
51 {
52     for(int i = 1; i <= n; i++)
53         matx[i] = maty[i] = 0;
54     while(BFS())
55         for(int i = 1; i <= n; i++)
56             if(!matx[i])DFS(i);

```

57 }

1.9 一般图最大匹配 (lhy)

```

1 struct blossom{
2
3     struct Edge{
4         int x, y, next;
5     }edge[M];
6
7     int n, W, tot, h, l, son[N];
8     int mat[N], pre[N], tp[N], q[N], vis[N], F[N];
9
10    void Prepare(int n_)
11    {
12        n = n_;
13        W = tot = 0;
14        for(int i = 1; i <= n; i++)
15            son[i] = mat[i] = vis[i] = 0;
16    }
17
18    void add(int x, int y)
19    {
20        edge[++tot].x = x; edge[tot].y = y; edge[tot].next = son[x]; son[x] = tot;
21    }
22
23    int find(int x)
24    {
25        return F[x] ? F[x] = find(F[x]) : x;
26    }
27
28    int lca(int u, int v)
29    {
30        for(++W;; u = pre[mat[u]], swap(u, v))
31            if(vis[u = find(u)] == W) return u;
32            else vis[u] = u ? W : 0;
33    }
34
35    void aug(int u, int v)
36    {
37        for(int w; u; v = pre[u = w])
38            w = mat[v], mat[mat[u] = v] = u;
39    }
40
41    void blo(int u, int v, int f)
42    {
43        for(int w; find(u) ^ f; u = pre[v = w])
44            pre[u] = v, F[u] ? 0 : F[u] = f, F[w = mat[u]] ? 0 : F[w] = f, tp[w] ^ 1 ? 0 : tp[q[++l] =
45            ↵ w] = -1;
46    }
47
48    int bfs(int x)
49    {
50        for(int i = 1; i <= n; i++)
51            tp[i] = F[i] = 0;
52        h = l = 0;
53        q[++l] = x;
54        tp[x]--;
55        while(h < l)

```

```

55     {
56         x = q[++h];
57         for(int i = son[x]; i; i = edge[i].next)
58         {
59             int y = edge[i].y, Lca;
60             if(!tp[y])
61             {
62                 if(!mat[y])return aug(y, x), 1;
63                 pre[y] = x, ++tp[y], --tp[q[++l] = mat[y]];
64             }
65             else if(tp[y] ^ 1 && find(x) ^ find(y))
66                 blo(x, y, Lca = lca(x, y)), blo(y, x, Lca);
67         }
68     }
69     return 0;
70 }
71
72 int solve()
73 {
74     int ans = 0;
75     for(int i = 1; i <= n; i++)
76         if(!mat[i])ans += bfs(i);
77     return ans;
78 }
}G;

```

1.10 KM 算法 (Nightfall)

$O(n^3)$, 1-based, 最大权匹配

不存在的边权值开到 $-n \times (|MAXV|)$, ∞ 为 $3n \times (|MAXV|)$

匹配为 (lk_i, i)

```

1 long long KM(int n, long long w[N][N])
2 {
3     long long ans = 0;
4     int x, py, p;
5     long long d;
6     for(int i = 1; i <= n; i++)
7         lx[i] = ly[i] = 0, lk[i] = -1;
8     for(int i = 1; i <= n; i++)
9         for(int j = 1; j <= n; j++)
10            lx[i] = max(lx[i], w[i][j]);
11     for(int i = 1; i <= n; i++)
12     {
13         for(int j = 1; j <= n; j++)
14             slk[j] = inf, vy[j] = 0;
15         for(lk[py = 0] = i; lk[py]; py = p)
16         {
17             vy[py] = 1; d = inf; x = lk[py];
18             for(int y = 1; y <= n; y++)
19                 if(!vy[y])
20                 {
21                     if(lx[x] + ly[y] - w[x][y] < slk[y])
22                         slk[y] = lx[x] + ly[y] - w[x][y], pre[y] = py;
23                     if(slk[y] < d)d = slk[y], p = y;
24                 }
25             for(int y = 0; y <= n; y++)
26                 if(vy[y])lx[lk[y]] -= d, ly[y] += d;
27                 else slk[y] -= d;
28         }

```

```

29     for(; py; py = pre[py]) lk[py] = lk[pre[py]];
30 }
31 for(int i = 1; i <= n; i++)
32     ans += lx[i] + ly[i];
33 return ans;
34 }
```

1.11 最小树形图 (Nightfall)

```

1 using Val = long long;
2 #define nil mem
3 struct Node {
4     Node *l, *r;
5     int dist;
6     int x, y;
7     Val val, laz;
8 }
9     mem[M] = {{nil, nil, -1}};
10 int sz = 0;
11 #define NEW(arg...) (new(mem + ++sz)Node{nil,nil,0,arg})
12
13 void add(Node *x, Val o) { if (x != nil) { x->val += o, x->laz += o; } }
14
15 void down(Node *x) {
16     add(x->l, x->laz);
17     add(x->r, x->laz);
18     x->laz = 0;
19 }
20
21 Node *merge(Node *x, Node *y) {
22     if (x == nil) return y;
23     if (y == nil) return x;
24     if (y->val < x->val) swap(x, y); //smalltop heap
25     down(x);
26     x->r = merge(x->r, y);
27     if (x->l->dist < x->r->dist) swap(x->l, x->r);
28     x->dist = x->r->dist + 1;
29     return x;
30 }
31
32 Node *pop(Node *x) {
33     down(x);
34     return merge(x->l, x->r);
35 }
36
37 struct DSU {
38     int f[N];
39
40     void clear(int n) {
41         for (int i = 0; i <= n; ++i) f[i] = i;
42     }
43
44     int fd(int x) {
45         if (f[x] == x) return x;
46         return f[x] = fd(f[x]);
47     }
48
49     int &operator[](int x) { return f[fd(x)]; }
50};
```

```

51 DSU W, S;
52 Node *H[N], *pe[N];
53 vector<pair<int, int>> G[N];
54 int dist[N], pa[N];
55
56 // addedge(x, y, w) : NEW(x, y, w, 0)
57 Val chuliu(int s, int n) { // O(ElogE)
58     for (int i = 1; i <= n; ++i) G[i].clear();
59     Val re = 0;
60     W.clear(n);
61     S.clear(n);
62     int rid = 0;
63     fill(H, H + n + 1, (Node *) nil);
64     for (auto i = mem + 1; i <= mem + sz; ++i)
65         H[i->y] = merge(i, H[i->y]);
66     for (int i = 1; i <= n; ++i)
67         if (i != s)
68             for (;;) {
69                 auto in = H[S[i]];
70                 H[S[i]] = pop(H[S[i]]);
71                 if (in == nil) return INF; // no solution
72                 if (S[in->x] == S[i]) continue;
73                 re += in->val;
74                 pe[S[i]] = in;
75                 // if (in->x == s) true root = in->y
76                 add(H[S[i]], -in->val);
77                 if (W[in->x] != W[i]) {
78                     W[in->x] = W[i];
79                     break;
80                 }
81                 G[in->x].push_back({in->y, ++rid});
82                 for (int j = S[in->x]; j != S[i]; j = S[pe[j]->x]) {
83                     G[pe[j]->x].push_back({pe[j]->y, rid});
84                     H[j] = merge(H[S[i]], H[j]);
85                     S[i] = S[j];
86                 }
87             }
88             ++rid;
89         for (int i = 1; i <= n; ++i)
90             if (i != s && S[i] == i)
91                 G[pe[i]->x].push_back({pe[i]->y, rid});
92     return re;
93 }
94
95 void makeSol(int s, int n) {
96     fill(dist, dist + n + 1, n + 1);
97     pa[s] = 0;
98     for (multiset<pair<int, int>> h = {{0, s}}; !h.empty();) {
99         int x = h.begin()->second;
100        h.erase(h.begin());
101        dist[x] = 0;
102        for (auto i : G[x])
103            if (i.second < dist[i.first]) {
104                h.erase({dist[i.first], i.first});
105                h.insert({dist[i.first] = i.second, i.first});
106                pa[i.first] = x;
107            }
108    }
109 }
110

```

1.12 支配树 (ct,Nightfall)

DAG (ct)

```

1 struct Edge {
2     Edge *next;
3     int to;
4 } ;
5 Edge *last[maxn], e[maxm], *ecnt = e; // original graph
6 Edge *rlast[maxn], re[maxm], *recnt = re; // reversed-edge graph
7 Edge *tlast[maxn], te[maxn << 1], *tecnt = te; // dominate tree graph
8 int deg[maxn], q[maxn], fa[maxn][20], all_fa[maxn], fa_cnt, size[maxn], dep[maxn];
9 inline void link(int a, int b)
10 {
11     *++ecnt = (Edge) {last[a], b}; last[a] = ecnt; ++deg[b];
12 }
13 inline void link_rev(int a, int b)
14 {
15     *++recnt = (Edge) {rlast[a], b}; rlast[a] = recnt;
16 }
17 inline void link_tree(int a, int b)
18 {
19     *++tecnt = (Edge) {tlast[a], b}; tlast[a] = tecnt;
20 }
21 inline int getlca(int a, int b)
22 {
23     if (dep[a] < dep[b]) std::swap(a, b);
24     int temp = dep[a] - dep[b];
25     for (int i; temp; temp -= 1 << i)
26         a = fa[a][i = __builtin_ctz(temp)];
27     for (int i = 16; ~i; --i)
28         if (fa[a][i] != fa[b][i])
29             a = fa[a][i], b = fa[b][i];
30     if (a == b) return a;
31     return fa[a][0];
32 }
33 void dfs(int x)
34 {
35     size[x] = 1;
36     for (Edge *iter = tlast[x]; iter; iter = iter -> next)
37         dfs(iter -> to), size[x] += size[iter -> to];
38 }
39 int main()
40 {
41     q[1] = 0;
42     int head = 0, tail = 1;
43     while (head < tail)
44     {
45         int now = q[++head];
46         fa_cnt = 0;
47         for (Edge *iter = rlast[now]; iter; iter = iter -> next)
48             all_fa[+fa_cnt] = iter -> to;
49         for (; fa_cnt > 1; --fa_cnt)
50             all_fa[fa_cnt - 1] = getlca(all_fa[fa_cnt], all_fa[fa_cnt - 1]);
51         fa[now][0] = all_fa[fa_cnt];
52         dep[now] = dep[all_fa[fa_cnt]] + 1;
53         if (now) link_tree(fa[now][0], now);
54
55         for (int i = 1; i <= 16; ++i)
56             fa[now][i] = fa[fa[now][i - 1]][i - 1];

```

```

57     for (Edge *iter = last[now]; iter; iter = iter -> next)
58         if (--deg[iter -> to] == 0) q[++tail] = iter -> to;
59     }
60     dfs(0);
61     for (int i = 1; i <= n; ++i) printf("%d\n", size[i] - 1 );
62     return 0;
63 }
```

一般图 (Nightfall)

```

1 struct Dominator_Tree{
2     int n, s, cnt;
3     int dfn[N], id[N], pa[N], semi[N], idom[N], p[N], mn[N];
4     vector<int> e[N], dom[N], be[N];
5
6     void ins(int x, int y){e[x].push_back(y);}
7
8     void dfs(int x)
9     {
10         dfn[x] = ++cnt; id[cnt] = x;
11         for(auto i:e[x])
12         {
13             if(!dfn[i])dfs(i), pa[dfn[i]] = dfn[x];
14             be[dfn[i]].push_back(dfn[x]);
15         }
16     }
17
18     int get(int x)
19     {
20         if(p[x] != p[p[x]])
21         {
22             if(semi[mn[x]] > semi[get(p[x])])mn[x] = get(p[x]);
23             p[x] = p[p[x]];
24         }
25         return mn[x];
26     }
27
28     void LT()
29     {
30         for(int i = cnt; i > 1; i--)
31         {
32             for(auto j:be[i])semi[i] = min(semi[i], semi[get(j)]);
33             dom[semi[i]].push_back(i);
34             int x = p[i] = pa[i];
35             for(auto j:dom[x])
36                 idom[j] = (semi[get(j)] < x ? get(j) : x);
37             dom[x].clear();
38         }
39         for(int i = 2; i <= cnt; i++)
40         {
41             if(idom[i] != semi[i])idom[i] = idom[idom[i]];
42             dom[idom[i]].push_back(id[i]);
43         }
44     }
45     void build()
46     {
47         for(int i = 1; i <= n; i++)
48             dfn[i] = 0, dom[i].clear(), be[i].clear(), p[i] = mn[i] = semi[i] = i;
49         cnt = 0, dfs(s), LT();
50     }
}
```

51 };

1.13 虚树 (ct)

```

1 struct Edge {
2     Edge *next;
3     int to;
4 } *last[maxn], e[maxn << 1], *ecnt = e;
5 inline void link(int a, int b)
6 {
7     *++ecnt = (Edge) {last[a], b}; last[a] = ecnt;
8     *++ecnt = (Edge) {last[b], a}; last[b] = ecnt;
9 }
10 int a[maxn], n, dfn[maxn], pos[maxn], timer, inv[maxn], st[maxn];
11 int fa[maxn], size[maxn], dep[maxn], son[maxn], top[maxn];
12 bool vis[maxn];
13 void dfs1(int x); // 树剖
14 void dfs2(int x);
15 inline int getlca(int a, int b);
16 inline bool cmp(int a, int b)
17 {
18     return dfn[a] < dfn[b];
19 }
20 inline bool isson(int a, int b)
21 {
22     return dfn[a] <= dfn[b] && dfn[b] <= inv[a];
23 }
24 typedef long long ll;
25 bool imp[maxn];
26 struct sEdge {
27     sEdge *next;
28     int to, w;
29 } *slast[maxn], se[maxn << 1], *secnt = se;
30 inline void slink(int a, int b, int w)
31 {
32     *++secnt = (sEdge) {slast[a], b, w}; slast[a] = secnt;
33 }
34 int main()
35 {
36     scanf("%d", &n);
37     for (int i = 1; i < n; ++i)
38     {
39         int a, b; scanf(" %d%d", &a, &b);
40         link(a, b);
41     }
42     int m; scanf(" %d", &m);
43     dfs1(1); dfs2(1);
44     memset(size, 0, (n + 1) << 2);
45     for (; m; --m)
46     {
47         int top = 0; scanf(" %d", &k);
48         for (int i = 1; i <= k; ++i) scanf(" %d", &a[i]), vis[a[i]] = imp[a[i]] = 1;
49         std::sort(a + 1, a + k + 1, cmp);
50         int p = k;
51         for (int i = 1; i < k; ++i)
52         {
53             int lca = getlca(a[i], a[i + 1]);
54             if (!vis[lca]) vis[a[++p]] = lca = 1;
55         }
56     }
57 }
```

```

56     std::sort(a + 1, a + p + 1, cmp);
57     st[++top] = a[1];
58     for (int i = 2; i <= p; ++i)
59     {
60         while (!isson(st[top], a[i])) --top;
61         slink(st[top], a[i], dep[a[i]] - dep[st[top]]);
62         st[++top] = a[i];
63     }
64     /*
65      write your code here.
66     */
67     for (int i = 1; i <= p; ++i) vis[a[i]] = imp[a[i]] = 0, slast[a[i]] = 0;
68     secnt = se;
69 }
70 return 0;
71 }
```

1.14 点分治 (ct)

```

1 int root, son[maxn], size[maxn], sum;
2 bool vis[maxn];
3 void dfs_root(int x, int fa)
4 {
5     size[x] = 1; son[x] = 0;
6     for (Edge *iter = last[x]; iter; iter = iter -> next)
7     {
8         if (iter -> to == fa || vis[iter -> to]) continue;
9         dfs_root(iter -> to, x);
10        size[x] += size[iter -> to];
11        cmax(son[x], size[iter -> to]);
12    }
13    cmax(son[x], sum - size[x]);
14    if (!root || son[x] < son[root]) root = x;
15 }
16 void dfs_chain(int x, int fa)
17 {
18     /*
19      write your code here.
20     */
21     for (Edge *iter = last[x]; iter; iter = iter -> next)
22     {
23         if (vis[iter -> to] || iter -> to == fa) continue;
24         dfs_chain(iter -> to, x);
25     }
26 }
27 void calc(int x)
28 {
29     for (Edge *iter = last[x]; iter; iter = iter -> next)
30     {
31         if (vis[iter -> to]) continue;
32         dfs_chain(iter -> to, x);
33         /*
34          write your code here.
35         */
36     }
37 }
38 void work(int x)
39 {
40     vis[x] = 1;
```

```

41 calc(x);
42 for (Edge *iter = last[x]; iter; iter = iter -> next)
43 {
44     if (vis[iter -> to]) continue;
45     root = 0;
46     sum = size[iter -> to];
47     dfs_root(iter -> to, 0);
48     work(root);
49 }
50
51 int main()
52 {
53     root = 0; sum = n;
54     dfs_root(1, 0);
55     work(root);
56     return 0;
57 }
```

1.15 树上倍增 (ct)

```

1 int fa[maxn][17], mn[maxn][17], dep[maxn];
2 bool vis[maxn];
3 void dfs(int x)
4 {
5     vis[x] = 1;
6     for (int i = 1; i <= 16; ++i)
7     {
8         if (dep[x] < (1 << i)) break;
9         fa[x][i] = fa[fa[x][i - 1]][i - 1];
10        mn[x][i] = dmin(mn[x][i - 1], mn[fa[x][i - 1]][i - 1]);
11    }
12    for (Edge *iter = last[x]; iter; iter = iter -> next)
13    {
14        if (!vis[iter -> to])
15        {
16            fa[iter -> to][0] = x;
17            mn[iter -> to][0] = iter -> w;
18            dep[iter -> to] = dep[x] + 1;
19            dfs(iter -> to);
20        }
21    }
22    inline int getlca(int x, int y)
23    {
24        if (dep[x] < dep[y]) std::swap(x, y);
25        int t = dep[x] - dep[y];
26        for (int i = 0; i <= 16 && t; ++i)
27        {
28            if ((1 << i) & t)
29                x = fa[x][i], t ^= 1 << i;
30        }
31        for (int i = 16; i >= 0; --i)
32        {
33            if (fa[x][i] != fa[y][i])
34            {
35                x = fa[x][i];
36                y = fa[y][i];
37            }
38            if (x == y) return x;
39        }
40    }
41    inline int getans(int x, int f)
42    {
43        int ans = inf, t = dep[x] - dep[f];
44        for (int i = 0; i <= 16; ++i)
45        {
46            if ((1 << i) & t)
47                ans = min(ans, mn[x][i] + i);
48        }
49        return ans;
50    }
51 }
```

```

40     for (int i = 0; i <= 16 && t; ++i)
41         if (t & (1 << i))
42         {
43             cmin(ans, mn[x][i]);
44             x = fa[x][i];
45             t ^= 1 << i;
46         }
47     return ans;
48 }
```

1.16 树哈希 (lhy,Luna)

简易版 (lhy)

每个节点的哈希值为每个子树哈希值的三次方和 +1

高级版 (Luna)

A_i 表示以 i 为根的子树的哈希值

B_i 表示整棵树以 i 为根的哈希值

```

1 template <int MAXN = 100000, int MAXM = 200000, long long MOD = 10000000000000000003ll>
2 struct tree_hash {
3     static long long ra[MAXN];
4     tree_hash() {
5         std::mt19937_64 mt(time(0));
6         std::uniform_int_distribution<long long> uid(0, MOD - 1);
7         for (int i = 0; i < MAXN; ++i) ra[i] = uid(mt);
8     }
9
10    struct node {
11        std::vector<long long> s;
12        int d1, d2;
13        long long h1, h2;
14        node() { d1 = d2 = 0; }
15        void add(int d, long long v) {
16            s.push_back(v);
17            if (d > d1) d2 = d1, d1 = d; else if (d > d2) d2 = d;
18        }
19        long long hash() {
20            h1 = h2 = 1;
21            for (long long i : s) {
22                h1 = mul_mod(h1, ra[d1] + i, MOD);
23                h2 = mul_mod(h2, ra[d2] + i, MOD);
24            }
25            return h1;
26        }
27        std::pair<int, long long> del(int d, long long v) {
28            if (d == d1) return {d2 + 1, mul_mod(h2, inverse(ra[d2] + v, MOD), MOD)};
29            return {d1 + 1, mul_mod(h1, inverse(ra[d1] + v, MOD), MOD)};
30        }
31    };
32
33    std::pair<int, long long> u[MAXN];
34    node tree[MAXN];
35    long long A[MAXN], B[MAXN];
36    void dfs1(const edge_list <MAXN, MAXM> &e, int x, int p = -1) {
37        tree[x] = node();
38        for (int i = e.begin[x]; ~i; i = e.next[i]) {
39            int c = e.dest[i];

```

```

40         if (c != p) {
41             dfs1(e, c, x);
42             tree[x].add(tree[c].d1 + 1, tree[c].h1);
43         }
44     }
45     A[x] = tree[x].hash();
46 }
47 void dfs2(const edge_list <MAXN, MAXM> &e, int x, int p = -1) {
48     if (~p) tree[x].add(u[x].first, u[x].second);
49     B[x] = tree[x].hash();
50     for (int i = e.begin[x]; ~i; i = e.next[i]) {
51         int c = e.dest[i];
52         if (c != p) {
53             u[c] = tree[x].del(tree[c].d1 + 1, tree[c].h1);
54             dfs2(e, c, x);
55         }
56     }
57 }
58 void solve(const edge_list <MAXN, MAXM> &e, int root) {
59     dfs1(e, root);
60     dfs2(e, root);
61 }
62 };
63 template <int MAXN, int MAXM, long long MOD>
64 long long tree_hash<MAXN, MAXM, MOD>::ra[MAXN];
65

```

1.17 Link-Cut Tree (ct)

LCT 常见应用

- **动态维护边双**

可以通过 LCT 来解决一类动态边双连通分量问题。即静态的询问可以用边双连通分量来解决，而树有加边等操作的问题。

把一个边双连通分量缩到 LCT 的一个点中，然后在 LCT 上求出答案。缩点的方法为加边时判断两点的连通性，如果已经联通则把两点在目前 LCT 路径上的点都缩成一个点。

- **动态维护基环森林**

通过 LCT 可以动态维护基环森林，即每个点有且仅有一个出度的图。有修改操作，即改变某个点的出边。对于每颗基环森林记录一个点为根，并把环上额外的一条边单独记出，剩下的边用 LCT 维护。一般使用有向 LCT 维护。

修改时分以下几种情况讨论：

- 修改的点是根，如果改的父亲在同一个连通块中，直接改额外边，否则删去额外边，在 LCT 上加边。
- 修改的点不是根，那么把这个点和其父亲的联系切除。如果该点和根在一个环上，那么把多的那条边加到 LCT 上。最后如果改的那个父亲和修改的点在一个连通块中，记录额外边，否则 LCT 上加边。

- **子树询问**

通过记录轻边信息可以快速地维护出整颗 LCT 的一些值。如子树和，子树最大值等。在 Access 时要进行虚实边切换，这时减去实边的贡献，并加上新加虚边的贡献即可。有时需要套用数据结构，如 Set 来维护最值等问题。

模板：

- $x \rightarrow y$ 链 $+z$
- $x \rightarrow y$ 链变为 z
- 在以 x 为根的树对 y 子树的点权求和
- $x \rightarrow y$ 链取 max
- $x \rightarrow y$ 链求和
- 连接 x, y

- 断开 x, y

V 单点值, sz 平衡树的 size, mv 链上最大, S 链上和, sm 区间相同标记, lz 区间加标记, B 虚边之和, ST 子树信息和, SM 子树和链上信息和。更新时:

$$\begin{aligned} S[x] &= S[c[x][0]] + S[c[x][1]] + V[x] \\ ST[x] &= B[x] + ST[c[x][0]] + ST[c[x][1]] \\ SM[x] &= S[x] + ST[x] \end{aligned}$$

```

1 struct Node *null;
2 struct Node {
3     Node *ch[2], *fa, *pos;
4     int val, mn, l, len; bool rev;
5     // min_val in chain
6     inline bool type()
7     {
8         return fa -> ch[1] == this;
9     }
10    inline bool check()
11    {
12        return fa -> ch[type()] == this;
13    }
14    inline void pushup()
15    {
16        pos = this; mn = val;
17        ch[0] -> mn < mn ? mn = ch[0] -> mn, pos = ch[0] -> pos : 0;
18        ch[1] -> mn < mn ? mn = ch[1] -> mn, pos = ch[1] -> pos : 0;
19        len = ch[0] -> len + ch[1] -> len + l;
20    }
21    inline void pushdown()
22    {
23        if (rev)
24        {
25            ch[0] -> rev ^= 1;
26            ch[1] -> rev ^= 1;
27            std::swap(ch[0], ch[1]);
28            rev ^= 1;
29        }
30    }
31    inline void pushdownall()
32    {
33        if (check()) fa -> pushdownall();
34        pushdown();
35    }
36    inline void rotate()
37    {
38        bool d = type(); Node *f = fa, *gf = f -> fa;
39        (fa = gf, f -> check()) ? fa -> ch[f -> type()] = this : 0;
40        (f -> ch[d] = ch[!d]) != null ? ch[!d] -> fa = f : 0;
41        (ch[!d] = f) -> fa = this;
42        f -> pushup();
43    }
44    inline void splay(bool need = 1)
45    {
46        if (need) pushdownall();
47        for (; check(); rotate())
48            if (fa -> check())
49                (type() == fa -> type() ? fa : this) -> rotate();
50        pushup();
51    }
52    inline Node *access()
53    {
54        Node *i = this, *j = null;

```

```

55     for ( ; i != null; i = (j = i) -> fa)
56     {
57         i -> splay();
58         i -> ch[1] = j;
59         i -> pushup();
60     }
61     return j;
62 }
63 inline void make_root()
64 {
65     access();
66     splay();
67     rev ^= 1;
68 }
69 inline void link(Node *that)
70 {
71     make_root();
72     fa = that;
73     splay(0);
74 }
75 inline void cut(Node *that)
76 {
77     make_root();
78     that -> access();
79     that -> splay(0);
80     that -> ch[0] = fa = null;
81     that -> pushup();
82 }
83 } mem[maxn];
84 inline Node *query(Node *a, Node *b)
85 {
86     a -> make_root(); b -> access(); b -> splay(0);
87     return b -> pos;
88 }
89 inline int dist(Node *a, Node *b)
90 {
91     a -> make_root(); b -> access(); b -> splay(0);
92     return b -> len;
93 }

```

1.18 圆方树 (ct)

```

1 int dfn[maxn], low[maxn], timer, st[maxn], top, id[maxn], scc;
2 void dfs(int x)
3 {
4     dfn[x] = low[x] = ++timer; st[++top] = x;
5     for (Edge *iter = last[x]; iter; iter = iter -> next)
6         if (!dfn[iter -> to])
7         {
8             dfs(iter -> to);
9             cmin(low[x], low[iter -> to]);
10            if (dfn[x] == low[iter->to])
11            {
12                int now, elder = top, minn = c[x];
13                ++scc;
14                do
15                {
16                    now = st[top--];
17                    cmin(minn, c[now]);

```

```

18         }
19     while (iter -> to != now);
20     for (int i = top + 1; i <= elder; ++i)
21         add(scc, st[i], minn);
22     add(scc, x, minn);
23   }
24 } else if (!id[iter -> to]) cmin(low[x], dfn[iter -> to]);
25 }
26 }
```

1.19 无向图最小割 (Nightfall)

```

1 int d[N];
2 bool v[N], g[N];
3
4 int get(int &s, int &t) {
5     CL(d);
6     CL(v);
7     int i, j, k, an, mx;
8     for (i = 1; i <= n; i++) {
9         k = mx = -1;
10        for (j = 1; j <= n; j++) if (!g[j] && !v[j] && d[j] > mx) k = j, mx = d[j];
11        if (k == -1) return an;
12        s = t;
13        t = k;
14        an = mx;
15        v[k] = 1;
16        for (j = 1; j <= n; j++) if (!g[j] && !v[j]) d[j] += w[k][j];
17    }
18    return an;
19 }
20
21 int mincut(int n, int w[N][N]) {
22     //n 为点数, w[i][j] 为 i 到 j 的流量, 返回无向图所有点对最小割之和
23     int ans = 0, i, j, s, t, x, y, z;
24     for (i = 1; i <= n - 1; i++) {
25         ans = min(ans, get(s, t));
26         g[t] = 1;
27         if (!ans) break;
28         for (j = 1; j <= n; j++) if (!g[j]) w[s][j] = (w[j][s] += w[j][t]);
29     }
30     return ans;
31 }
32
33 // 无向图最小割树
34 void fz(int l, int r) {// 左闭右闭, 分治建图
35     if (l == r) return;
36     S = a[l];
37     T = a[r];
38     reset(); // 将所有边权复原
39     flow(S, T); // 做网络流
40     dfs(S); // 找割集, v[x]=1 属于 S 集, 否则属于 T 集
41     ADD(S, T, fl); // 在最小割树中建边
42     L = l, R = r;
43     for (i = l; i <= r; i++) if (v[a[i]]) q[L++] = a[i]; else q[R--] = a[i];
44     for (i = l; i <= r; i++) a[i] = q[i];
45     fz(l, L - 1);
46     fz(R + 1, r);
47 }
```

1.20 网络流 (lhy,ct)

Dinic (ct)

```

1 struct Edge {
2     Edge *next, *rev;
3     int to, cap;
4 } *last[maxn], *cur[maxn], e[maxn], *ecnt = e;
5 inline void link(R int a, R int b, R int w)
6 {
7     *++ecnt = (Edge) {last[a], ecnt + 1, b, w}; last[a] = ecnt;
8     *++ecnt = (Edge) {last[b], ecnt - 1, a, 0}; last[b] = ecnt;
9 }
10 int ans, s, t, q[maxn], dep[maxn];
11 inline bool bfs()
12 {
13     memset(dep, -1, (t + 1) << 2);
14     dep[q[1] = t] = 0; int head = 0, tail = 1;
15     while (head < tail)
16     {
17         int now = q[++head];
18         for (Edge *iter = last[now]; iter; iter = iter -> next)
19             if (dep[iter -> to] == -1 && iter -> rev -> cap)
20                 dep[q[++tail] = iter -> to] = dep[now] + 1;
21     }
22     return dep[s] != -1;
23 }
24 int dfs(int x, int f)
25 {
26     if (x == t) return f;
27     int used = 0;
28     for (Edge* &iter = cur[x]; iter; iter = iter -> next)
29         if (iter -> cap && dep[iter -> to] + 1 == dep[x])
30         {
31             int v = dfs(iter -> to, dmin(f - used, iter -> cap));
32             iter -> cap -= v;
33             iter -> rev -> cap += v;
34             used += v;
35             if (used == f) return f;
36         }
37     return used;
38 }
39 inline void dinic()
40 {
41     while (bfs())
42     {
43         memcpy(cur, last, sizeof cur);
44         ans += dfs(s, inf);
45     }
46 }
```

SAP (lhy)

```

1 void SAP(int n, int st, int ed)
2 {
3     for(int i = 1; i <= n; i++)
4         now[i] = son[i];
5     sumd[0] = n;
6     int flow = inf, x = st;
7     while(dis[st] < n)
```

```

8     {
9         back[x] = flow;
10        int flag = 0;
11        for(int i = now[x]; i != -1; i = edge[i].next)
12        {
13            int y = edge[i].y;
14            if(edge[i].f && dis[y] + 1 == dis[x])
15            {
16                flag = 1;
17                now[x] = i;
18                pre[y] = i;
19                flow = min(flow, edge[i].f);
20                x = y;
21                if(x == ed)
22                {
23                    ans += flow;
24                    while(x != st)
25                    {
26                        edge[pre[x]].f -= flow;
27                        edge[pre[x] ^ 1].f += flow;
28                        x = edge[pre[x]].x;
29                    }
30                    flow = inf;
31                }
32                break;
33            }
34        }
35        if(flag)continue;
36        int minn = n - 1, tmp;
37        for(int i = son[x]; i != -1; i = edge[i].next)
38        {
39            int y = edge[i].y;
40            if(edge[i].f && dis[y] < minn)
41            {
42                minn = dis[y];
43                tmp = i;
44            }
45        }
46        now[x] = tmp;
47        if(!(--sumd[dis[x]]))return;
48        sumd[dis[x] = minn + 1]++;
49        if(x != st)flow = back[x = edge[pre[x]].x];
50    }
51}

```

zkw 费用流 (lhy)

```

1 int aug(int no, int res)
2 {
3     if(no == ED) return mincost += 1ll * pil * res, res;
4     v[no] = 1;
5     int flow = 0;
6     for(int i = son[no]; i != -1; i = edge[i].next)
7     {
8         if(edge[i].f && !v[edge[i].y] && !edge[i].c)
9         {
10             int d = aug(edge[i].y, min(res, edge[i].f));
11             edge[i].f -= d, edge[i ^ 1].f += d, flow += d, res -= d;
12             if(!res) return flow;
13         }
14     }
15     return flow;

```

```

14 }
15
16 bool modlabel()
17 {
18     long long d = 0x3f3f3f3f3f3f3f3f3f3f3f3f3f11;
19     for(int i = 1; i <= cnt; i++)
20         if(v[i])
21         {
22             for(int j = son[i]; j != -1; j = edge[j].next)
23                 if(edge[j].f && !v[edge[j].y] && edge[j].c < d)d = edge[j].c;
24         }
25     if(d == 0x3f3f3f3f3f3f3f3f3f3f3f3f3f11) return 0;
26     for(int i = 1; i <= cnt; i++)
27         if(v[i])
28         {
29             for(int j = son[i]; j != -1; j = edge[j].next)
30                 edge[j].c -= d, edge[j ^ 1].c += d;
31         }
32     pil += d;
33     return 1;
34 }
35
36 void minimum_cost_flow_zkw()
37 {
38     pil = 0;
39     int nowans = 0;
40     nowf = 0;
41     do{
42         do{
43             for(int i = 1; i <= cnt; i++)
44                 v[i] = 0;
45             nowans = aug(ST, inf);
46             nowf += nowans;
47         }while(nowans);
48     }while(modlabel());
49 }
```

1.21 欧拉回路 (cxy)

有向图欧拉回路无向图欧拉回路

```

1 int To[maxm];
2 bool vis[maxm];
3 vector<int> G[maxn];
4 int deg[maxn];
5 int cur[maxn];
6 stack<int> stk;
7
8 void dfs(int u) {
9     for (cur[u]; cur[u] < G[u].size(); cur[u]++) {
10         int i = cur[u];
11         if (vis[abs(G[u][i])]) continue;
12         int v = To[abs(G[u][i])] ^ u;
13         vis[abs(G[u][i])] = true;
14         dfs(v);
15         stk.push(G[u][i]);
16     }
17 }
```

```

19 void solve1(int n, int m) {
20     int s = 1;
21     for (int i = 1; i <= m; i++) {
22         int u = read(), v = read();
23         To[i] = u ^ v; s = u;
24         G[u].pb(i); G[v].pb(-i);
25     }
26     for (int i = 1; i <= n; i++) if (G[i].size() & 1) { print("NO\n"); return; }
27     dfs(s);
28     if (stk.size() != m) { print("NO\n"); return; }
29     print("YES\n");
30     while (stk.size()) print(stk.top()), print(' '), stk.pop();
31 }
32
33
34 void solve2(int n, int m) {
35     int s = 1;
36     for (int i = 1; i <= m; i++) {
37         int u = read(), v = read();
38         To[i] = u ^ v; s = u;
39         G[u].pb(i); deg[v]++;
40     }
41     for (int i = 1; i <= n; i++) if (G[i].size() != deg[i]) { print("NO\n"); return; }
42     dfs(s);
43     if (stk.size() != m) { print("NO\n"); return; }
44     print("YES\n");
45     while (stk.size()) print(stk.top()), print(' '), stk.pop();
46 }

```

混合图欧拉回路

将无向边任意定向，随后对每个点计算权值 $\frac{1}{2}|\text{入度} - \text{出度}|$ ，如果 $\text{入度} < \text{出度}$ ，从 S 向该点连接一条流量为该点权值的边，如果 $\text{入度} > \text{出度}$ ，从该点向 T 连接一条流量为该点权值的，随后将所有无向边按照之前所定方向连接一条流量为 1 的边，随后跑最大流。如果两边的边都满流，则混合图欧拉回路有解，只需将中间满流的边反向，跑有向图欧拉回路即可

1.22 图论知识 (gy,lhy)

树链的交

记两条链分别为 (a_1, b_1) 和 (a_2, b_2) ，记 $\text{LCA}(a_1, b_1) = c_1, \text{LCA}(a_2, b_2) = c_2$ 。记 $\text{LCA}(a_1, a_2) = d_1, \text{LCA}(a_1, b_2) = d_2, \text{LCA}(b_1, a_2) = d_3, \text{LCA}(b_1, b_2) = d_4$ 。按深度排序后得 $\text{depth}(d_1) \leq \text{depth}(d_2) \leq \text{depth}(d_3) \leq \text{depth}(d_4), \text{depth}(c_1) \leq \text{depth}(c_2)$ 。两条链有交集当且仅当 $\text{depth}(c_1) \leq \text{depth}(d_1)$ 且 $\text{depth}(c_2) \leq \text{depth}(d_3)$ ，此时两条链得交为 (d_3, d_4)

Hall theorem

二分图 $G = (X, Y, E)$ 有完备匹配的充要条件是：对于 X 的任意一个子集 S 都满足 $|S| \leq |A(S)|$ ， $A(S)$ 是 Y 的子集，是 S 的邻集（与 S 有边的边集）。

Prufer 编码

树和其 prufer 编码一一对应，一颗 n 个点的树，其 prufer 编码长度为 $n-2$ ，且度数为 d_i 的点在 prufer 编码中出现 d_i-1 次。

由树得到序列：总共需要 $n-2$ 步，第 i 步在当前的树中寻找具有最小标号的叶子节点，将其与其相连的点的标号设为 Prüfer 序列的第 i 个元素 p_i ，并将此叶子节点从树中删除，直到最后得到一个长度为 $n-2$ 的 Prüfer 序列和一个只有两个节点的树。

由序列得到树：先将所有点的度赋初值为 1，然后加上它的编号在 Prüfer 序列中出现的次数，得到每个点的度；

执行 $n-2$ 步, 第 i 步选取具有最小标号的度为 1 的点 u 与 $v = p_i$ 相连, 得到树中的一条边, 并将 u 和 v 的度减 1。最后再把剩下的两个度为 1 的点连边, 加入到树中。

相关结论:

- n 个点完全图, 每个点度数依次为 d_1, d_2, \dots, d_n , 这样生成树的棵树为: $\frac{(n-2)!}{(d_1-1)!(d_2-1)!\dots(d_n-1)!}$
- 左边有 n_1 个点, 右边有 n_2 个点的完全二分图的生成树棵树为: $n_1^{n_2-1} + n_2^{n_1-1}$
- m 个连通块, 每个连通块有 c_i 个点, 把他们全部连通的生成树方案数: $(\sum c_i)^{m-2} \prod c_i$

差分约束

若要使得所有量两两的值最接近, 则将如果将源点到各点的距离初始化为 0。若要使得某一变量与其余变量的差最大, 则将源点到各点的距离初始化为 ∞ , 其中之一为 0。若求最小方案则跑最长路, 否则跑最短路。

弦图

弦图: 任意点数 ≥ 4 的环皆有弦的无向图

单纯点: 与其相邻的点的诱导子图为完全图的点

完美消除序列: 每次选择一个单纯点删去的序列

弦图必有完美消除序列

$O(m+n)$ 求弦图的完美消除序列: 每次选择未选择的标号最大的点, 并将与其相连的点标号 +1, 得到完美消除序列的反序

最大团数 = 最小染色数: 按完美消除序列从后往前贪心地染色

最小团覆盖 = 最大点独立集: 按完美消除序列从前往后贪心地选点加入点独立集

计数问题

- 有根树计数

$$a_1 = 1$$

$$a_{n+1} = \frac{\sum_{j=1}^n j \cdot a_j \cdot S_{n,j}}{n}$$

$$S_{n,j} = \sum_{i=1}^{n/j} a_{n+1-ij} = S_{n-j,j} + a_{n+1-j}$$

- 无根树计数

$$\begin{cases} a_n - \sum_{i=1}^{n/2} a_i a_{n-i} & n \text{ is odd} \\ a_n - \sum_{i=1}^{n/2} a_i a_{n-i} + \frac{1}{2} a_n (\frac{a_n}{2} + 1) & n \text{ is even} \end{cases}$$

- 生成树计数

Kirchhoff Matrix $T = \text{Deg} - A$, Deg 是度数对角阵, A 是邻接矩阵。无向图度数矩阵是每个点度数; 有向图度数矩阵是每个点入度。邻接矩阵 $A[u][v]$ 表示 $u \rightarrow v$ 边个数, 重边按照边数计算, 自环不计入度数。

无向图生成树计数: $c = |K \text{ 的任意 } 1 \text{ 个 } n-1 \text{ 阶主子式}|$

有向图外向树计数: $c = |\text{去掉根所在的那阶得到的主子式}|$

- Edmonds Matrix

Edmonds matrix A of a balanced ($|U| = |V|$) bipartite graph $G = (U, V, E)$:

$$A_{ij} = \begin{cases} x_{ij} & (u_i, v_j) \in E \\ 0 & (u_i, v_j) \notin E \end{cases}$$

where the x_{ij} are indeterminates.

G 有完备匹配当且仅当关于 x_{ij} 的多项式 $\det(A_{ij})$ 不恒为 0。

完备匹配的个数等于多项式中单项式的个数

- 偶数点完全图完备匹配计数
 $(n - 1)!!$
- 无根二叉树计数
 $(2n - 5)!!$
- 有根二叉树计数
 $(2n - 3)!!$

上下界网络流

$B(u, v)$ 表示边 (u, v) 流量的下界, $C(u, v)$ 表示边 (u, v) 流量的上界, 设 $F(u, v)$ 表示边 (u, v) 的实际流量
设 $G(u, v) = F(u, v) - B(u, v)$, 则 $0 \leq G(u, v) \leq C(u, v) - B(u, v)$

- 无源汇的上下界可行流

建立超级源点 S^* 和超级汇点 T^* , 对于原图每一条边 (u, v) 在新网络中连如下三条边: $S^* \rightarrow v$, 容量为 $B(u, v)$; $u \rightarrow T^*$, 容量为 $B(u, v)$; $u \rightarrow v$, 容量为 $C(u, v) - B(u, v)$ 。最后求新网络的最大流, 判断从超级源点 S^* 出发的边是否都满流即可, 边 (u, v) 的最终解中的实际流量为 $G(u, v) + B(u, v)$ 。

- 有源汇的上下界可行流

从汇点 T 到源点 S 连一条上界为 ∞ , 下界为 0 的边。按照无源汇的上下界可行流一样做即可, 流量即为 $T \rightarrow S$ 边上的流量。

- 有源汇的上下界最大流

– 在有源汇的上下界可行流中, 从汇点 T 到源点 S 的边改为连一条上界为 ∞ , 下界为 x 的边。 x 满足二分性质, 找到最大的 x 使得新网络存在有源汇的上下界可行流即为原图的最大流。
– 从汇点 T 到源点 S 连一条上界为 ∞ , 下界为 0 的边, 变成无源汇的网络。按照无源汇的上下界可行流的方法, 建立超级源点 S^* 与超级汇点 T^* , 求一遍 $S^* \rightarrow T^*$ 的最大流, 再将从汇点 T 到源点 S 的这条边拆掉, 求一次 $S \rightarrow T$ 的最大流即可。

- 有源汇的上下界最小流

– 在有源汇的上下界可行流中, 从汇点 T 到源点 S 的边改为连一条上界为 x , 下界为 0 的边。 x 满足二分性质, 找到最小的 x 使得新网络存在有源汇的上下界可行流即为原图的最大流。
– 按照无源汇的上下界可行流的方法, 建立超级源点 S^* 与超级汇点 T^* , 求一遍 $S^* \rightarrow T^*$ 的最大流, 但是注意不加上汇点 T 到源点 S 的这条边, 即不使之改为无源汇的网络去求解。求完后, 再加上那条汇点 T 到源点 S 的边, 上界为 ∞ 的边。因为这条边的下界为 0, 所以 S^*, T^* 无影响, 再求一次 $S^* \rightarrow T^*$ 的最大流。若超级源点 S^* 出发的边全部满流, 则 $T \rightarrow S$ 边上的流量即为原图的最小流, 否则无解。

- 上下界费用流

求无源汇上下界最小费用可行流或有源汇上下界最小费用最大可行流, 用相应构图方法, 给边加上费用即可。求有源汇上下界最小费用最小可行流, 先按相应构图方法建图, 求出一个保证必要边满流情况下的最小费用。如果费用全部非负, 那么此时的费用即为答案。如果费用有负数, 继续做从 S 到 T 的流量任意的最小费用流, 加上原来的费用就是答案。

费用流消负环

新建超级源 S^* 和超级汇 T^* , 对于所有流量非空的负权边 e , 先满流 ($ans += e.f * e.c$, $e.rev.f += e.f$, $e.f = 0$), 再连边 $S^* \rightarrow e.to$, $e.from \rightarrow T^*$, 流量均为 $e.f (> 0)$, 费用均为 0。再连边 $T \rightarrow S$, 流量为 ∞ , 费用为 0。跑一遍 $S^* \rightarrow T^*$ 的最小费用最大流, 将费用累加 ans , 拆掉 $T \rightarrow S$ 那条边 (此边的流量为残量网络中 $S \rightarrow T$ 的流量。此时负环已消, 再继续跑最小费用最大流。

二物流

水源 S_1 , 水汇 T_1 , 油源 S_2 , 油汇 T_2 , 每根管道流量共用, 使流量和最大。

建超级源 S_1^* , 超级汇 T_1^* , 连边 $S_1^* \rightarrow S_1$, $S_1^* \rightarrow S_2$, $T_1 \rightarrow T_1^*$, $T_2 \rightarrow T_1^*$, 设最大流为 x_1 。

建超级源 S_2^* , 超级汇 T_2^* , 连边 $S_2^* \rightarrow S_1$, $S_2^* \rightarrow T_2$, $T_1 \rightarrow T_2^*$, $S_2 \rightarrow T_2^*$, 设最大流为 x_2 。则最大流中水流 $\frac{x_1 + x_2}{2}$, 油流量 $\frac{x_1 - x_2}{2}$ 。

最大权闭合子图

给定一个带点权的有向图，求其最大权闭合子图。

从源点 S 向每一条正权点连一条容量为权值的边，每个负权点向汇点 T 连一条容量为权值绝对值的边，有向图原来的边容量为 ∞ 。求它的最小割，与源点 S 连通的点构成最大权闭合子图，权值为正权值和 - 最小割。

最大密度子图

给定一个无向图，求其一个子图，使得子图的边数 $|E|$ 和点数 $|V|$ 满足 $\frac{|E|}{|V|}$ 最大。

二分答案 k ，使得 $|E| - k|V| \geq 0$ 有解，将原图边和点都看作点，边 (u, v) 分别向 u 和 v 连边求最大权闭合子图。

Math

2.1 int64 相乘取模 (Durandal)

```

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

2.2 ex-Euclid (gy)

```

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

2.3 中国剩余定理 (Durandal)

中国剩余定理

若 m_1, m_2, \dots, m_n 两两互素，则方程组 $x \equiv a_i \pmod{m_i}$ 有解，且解可以由以下方法构造：

$$\begin{aligned} M &= \prod_{i=1}^n m_i \\ M_i &= \frac{M}{m_i} \\ t_i &= \frac{1}{M_i} \pmod{m_i} \\ ans &\equiv \sum_{i=1}^n a_i t_i M_i \pmod{M} \end{aligned}$$

ex-Euclid 解二元中国剩余定理

返回是否可行，余数和模数结果为 r_1, m_1

```

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

2.4 线性同余不等式 (Durandal)

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

```

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

2.5 平方剩余 (Nightfall)

$$x^2 \equiv a \pmod{p}, 0 \leq a < p$$

返回是否存在解

p 必须是质数，若是多个单次质数的乘积可以分别求解再用 CRT 合并
复杂度为 $O(\log n)$

```

1 bool solve(int n, int &x) {
2     if (n == 0) return x = 0, true;
3     if (MOD == 2) return x = 1, true;
4     if (power(n, MOD / 2, MOD) == MOD - 1) return false;
5     ll c = 1, d = 0, b = 1, a, w;
6     // finding a such that a^2 - n is not a square
7     do {
```

```

8     a = rand() % MOD;
9     w = (a * a - n + MOD) % MOD;
10    if (w == 0) return x = a, true;
11 } while (power(w, MOD / 2, MOD) != MOD - 1);
12 for (int times = (MOD + 1) / 2; times; times >= 1) {
13     if (times & 1) multiply(c, d, a, b, w);
14     multiply(a, b, a, b, w);
15 }
16 // x = (a + sqrt(w)) ^ ((p + 1) / 2)
17 return x = c, true;
18 }
```

2.6 组合数 (Nightfall)

```

1 int l, a[33], p[33], P[33];
2
3 U fac(int k, LL n) { // 求  $n! \bmod p_k^t$ , 返回值 U{ 不包含  $p_k$  的值,  $p_k$  出现的次数 }
4     if (!n) return U{1, 0};
5     LL x = n / p[k], y = n / P[k], ans = 1;
6     int i;
7     if (y) { // 求出循环节的答案
8         for (i = 2; i < P[k]; i++) if (i % p[k]) ans = ans * i % P[k];
9         ans = Pw(ans, y, P[k]);
10    }
11    for (i = y * P[k]; i <= n; i++) if (i % p[k]) ans = ans * i % M; // 求零散部分
12    U z = fac(k, x);
13    return U{ans * z.x % M, x + z.z};
14 }
15
16 LL get(int k, LL n, LL m) { // 求  $C(n, m) \bmod p_k^t$ 
17     U a = fac(k, n), b = fac(k, m), c = fac(k, n - m); // 分三部分求解
18     return Pw(p[k], a.z - b.z - c.z, P[k]) * a.x % P[k] * inv(b.x, P[k]) % P[k] * inv(c.x, P[k]) % P[k];
19 }
20
21 LL CRT() { // CRT 合并答案
22     LL d, w, y, x, ans = 0;
23     for (int i = 1; i <= l; i++)
24         w = M / P[i], exgcd(w, P[i], x, y),
25         ans = (ans + w * x % M * a[i]) % M;
26     return (ans + M) % M;
27 }
28
29 LL C(LL n, LL m) { // 求  $C(n, m)$ 
30     for (int i = 1; i <= l; i++)
31         a[i] = get(i, n, m);
32     return CRT();
33 }
34
35 LL exLucas(LL n, LL m, int M) {
36     int jj = M, i; // 求  $C(n, m) \bmod M$ , M=prod( $p_i^{k_i}$ ), 时间  $O(p_i^{k_i} \lg^{2n})$ 
37     for (i = 2; i * i <= jj; i++) if (jj % i == 0) for (p[++l] = i, P[l] = 1; jj % i == 0; P[l] *=
38         p[l]) jj /= i;
39     if (jj > 1) l++, p[l] = P[l] = jj;
40     return C(n, m);
}
```

2.7 高斯消元 (ct)

```

1 db a[maxn][maxn], x[maxn];
2 int main()
3 {
4     int rank = 0;
5
6     for (int i = 1, now = 1; i <= n && now <= m; ++now)
7     {
8         int tmp = i;
9         for (int j = i + 1; j <= n; ++j)
10            if (fabs(a[j][now]) > fabs(a[tmp][now])) tmp = j;
11         for (int k = now; k <= m; ++k)
12            std::swap(a[i][k], a[tmp][k]);
13         if (fabs(a[i][now]) < eps) continue;
14
15         for (int j = i + 1; j <= n; ++j)
16         {
17             db tmp = a[j][now] / a[i][now];
18             for (int k = now; k <= m; ++k)
19                 a[j][k] -= tmp * a[i][k];
20         }
21         ++i; ++rank;
22     }
23
24     if (rank == n)
25     {
26         x[n] = a[n][n + 1] / a[n][n];
27         for (int i = n - 1; i; --i)
28         {
29             for (int j = i + 1; j <= n; ++j)
30                 a[i][n + 1] -= x[j] * a[i][j];
31             x[i] = a[i][n + 1] / a[i][i];
32         }
33     }
34     else puts("Infinite Solution!");
35     return 0;
36 }
```

2.8 Miller Rabin & Pollard Rho (gy)

Test Set	First Wrong Answer
2, 3, 5, 7	(INT32_MAX)
2, 7, 61	4, 759, 123, 141
2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37	(INT64_MAX)

```

1 const int test_case_size = 12;
2 const int test_cases[test_case_size] = {2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37};
3
4 int64_t multiply_mod(int64_t x, int64_t y, int64_t p) {
5     int64_t t = (x * y - (int64_t)((long double)x / p * y + 1e-3) * p) % p;
6     return t < 0 ? t + p : t;
7 }
8
9 int64_t add_mod(int64_t x, int64_t y, int64_t p) {
10    return (0ull + x + y) % p;
11 }
12
13 int64_t power_mod(int64_t x, int64_t exp, int64_t p) {
```

```

14     int64_t ans = 1;
15     while (exp) {
16         if (exp & 1)
17             ans = multiply_mod(ans, x, p);
18         x = multiply_mod(x, x, p);
19         exp >= 1;
20     }
21     return ans;
22 }

23
24 bool miller_rabin_check(int64_t prime, int64_t base) {
25     int64_t number = prime - 1;
26     for (; ~number & 1; number >= 1)
27         continue;
28     int64_t result = power_mod(base, number, prime);
29     for (; number != prime - 1 && result != 1 && result != prime - 1; number <= 1)
30         result = multiply_mod(result, result, prime);
31     return result == prime - 1 || (number & 1) == 1;
32 }

33
34 bool miller_rabin(int64_t number) {
35     if (number < 2)
36         return false;
37     if (number < 4)
38         return true;
39     if (~number & 1)
40         return false;
41     for (int i = 0; i < test_case_size && test_cases[i] < number; i++)
42         if (!miller_rabin_check(number, test_cases[i]))
43             return false;
44     return true;
45 }

46
47 int64_t gcd(int64_t x, int64_t y) {
48     return y == 0 ? x : gcd(y, x % y);
49 }

50
51 int64_t pollard_rho_test(int64_t number, int64_t seed) {
52     int64_t x = rand() % (number - 1) + 1, y = x;
53     int head = 1, tail = 2;
54     while (true) {
55         x = multiply_mod(x, x, number);
56         x = add_mod(x, seed, number);
57         if (x == y)
58             return number;
59         int64_t answer = gcd(std::abs(x - y), number);
60         if (answer > 1 && answer < number)
61             return answer;
62         if (++head == tail) {
63             y = x;
64             tail <= 1;
65         }
66     }
67 }

68
69 void factorize(int64_t number, std::vector<int64_t> &divisor) {
70     if (number > 1) {
71         if (miller_rabin(number))
72             divisor.push_back(number);
73     } else {
74         int64_t factor = number;

```

```

75     while (factor >= number)
76         factor = pollard_rho_test(number, rand() % (number - 1) + 1);
77         factorize(number / factor, divisor);
78         factorize(factor, divisor);
79     }
80 }
81 }
```

2.9 $O(m^2 \log n)$ 线性递推 (lhy)

```

1 typedef vector<int> poly;
2 // {1, 3} {2, 1} an = 2an-1 + an-2, calc(3) = 7
3 struct LinearRec{
4     int n, LOG;
5     poly first, trans;
6     vector<poly> bin;
7     poly add(poly &a, poly &b)
8     {
9         poly res(n * 2 + 1, 0);
10        for(int i = 0; i <= n; i++)
11            for(int j = 0; j <= n; j++)
12                (res[i + j] += 1ll * a[i] * b[j] % mo) %= mo;
13        for(int i = 2 * n; i > n; i--)
14        {
15            for(int j = 0; j < n; j++)
16                (res[i - 1 - j] += 1ll * res[i] * trans[j] % mo) %= mo;
17            res[i] = 0;
18        }
19        res.erase(res.begin() + n + 1, res.end());
20        return res;
21    }
22    LinearRec(poly &first, poly &trans, int LOG): LOG(LOG), first(first), trans(trans)
23    {
24        n = first.size();
25        poly a(n + 1, 0);
26        a[1] = 1;
27        bin.push_back(a);
28        for(int i = 1; i < LOG; i++)
29            bin.push_back(add(bin[i - 1], bin[i - 1]));
30    }
31    int calc(long long k)
32    {
33        poly a(n + 1, 0);
34        a[0] = 1;
35        for(int i = 0; i < LOG; i++)
36            if((k >> i) & 1)a = add(a, bin[i]);
37        int ret = 0;
38        for(int i = 0; i < n; i++)
39            if((ret += 1ll * a[i + 1] * first[i] % mo) >= mo)ret -= mo;
40        return ret;
41    }
42};
```

2.10 Berlekamp Massey (NightFall)

```

1 // Complexity: O(n^2) Requirement: const MOD, inverse(int)
2 // Input: the first elements of the sequence
3 // Output: the recursive equation of the given sequence
```

```

4 // Example In: {1, 1, 2, 3}
5 // Example Out: {1, 1000000006, 1000000006} (MOD = 1e9+7)
6 struct Poly {
7     vector<int> a;
8     Poly() { a.clear(); }
9     Poly(vector<int> &a) : a(a) {}
10    int length() const { return a.size(); }
11    Poly move(int d) {
12        vector<int> na(d, 0);
13        na.insert(na.end(), a.begin(), a.end());
14        return Poly(na);
15    }
16    int calc(vector<int> &d, int pos) {
17        int ret = 0;
18        for (int i = 0; i < (int) a.size(); ++i) {
19            if ((ret += (LL) d[pos - i] * a[i] % MOD) >= MOD) {
20                ret -= MOD;
21            }
22        }
23        return ret;
24    }
25    Poly operator-(const Poly &b) {
26        vector<int> na(max(this->length(), b.length()));
27        for (int i = 0; i < (int) na.size(); ++i) {
28            int aa = i < this->length() ? this->a[i] : 0,
29                  bb = i < b.length() ? b.a[i] : 0;
30            na[i] = (aa + MOD - bb) % MOD;
31        }
32        return Poly(na);
33    }
34};
35
36 Poly operator*(const int &c, const Poly &p) {
37     vector<int> na(p.length());
38     for (int i = 0; i < (int) na.size(); ++i) {
39         na[i] = (LL) c * p.a[i] % MOD;
40     }
41     return na;
42 }
43 vector<int> solve(vector<int> a) {
44     int n = a.size();
45     Poly s, b;
46     s.a.push_back(1), b.a.push_back(1);
47     for (int i = 0, j = -1, ld = 1; i < n; ++i) {
48         int d = s.calc(a, i);
49         if (d) {
50             if ((s.length() - 1) * 2 <= i) {
51                 Poly ob = b;
52                 b = s;
53                 s = s - (LL) d * inverse(ld) % MOD * ob.move(i - j);
54                 j = i;
55                 ld = d;
56             } else {
57                 s = s - (LL) d * inverse(ld) % MOD * b.move(i - j);
58             }
59         }
60     }
61     //Caution: s.a might be shorter than expected
62     return s.a;
63 }
64 /* 求行列式 -> 求特征多项式 : det(A)=(-1)^nPA(0)

```

65 求矩阵或向量列最小多项式 : 随机投影成数列
 66 如果最小多项式里面有 x 的因子, 那么行列式为 0, 否则
 67 随机乘上对角阵 D, $\det(A) = \det(AD)/\det(D)* /$

2.11 线性基 (ct)

```

1 int main()
2 {
3     for (int i = 1; i <= n; ++i)
4     {
5         ull x = F();
6         cmax(m, 63 - __builtin_clzll(x));
7         for ( ; x; )
8         {
9             tmp = __builtin_ctzll(x);
10            if (!b[tmp])
11            {
12                b[tmp] = x;
13                break;
14            }
15            x ^= b[tmp];
16        }
17    }
18 }
```

2.12 多项式 (lhy,ct,gy)

FFT (ct)

```

1 typedef double db;
2 const db pi = acos(-1);
3
4 struct Complex {
5     db x, y;
6     inline Complex operator * (const Complex &that) const {return (Complex) {x * that.x - y * that.y, x
7         + that.y + y * that.x};}
8     //inline Complex operator + (const Complex &that) const {return (Complex) {x + that.x, y + that.y};}
9     inline Complex operator += (const Complex &that){x+=that.x;y+=that.y;};
10    inline Complex operator - (const Complex &that) const {return (Complex) {x - that.x, y - that.y};;}
11 } buf_a[maxn], buf_b[maxn], buf_c[maxn], w[maxn], c[maxn], a[maxn], b[maxn];
12
13 int n;
14 void bit_reverse(Complex *x, Complex *y)
15 {
16     for (int i = 0; i < n; ++i) y[i] = x[i];
17     Complex tmp;
18     for (int i = 0, j = 0; i < n; ++i)
19     {
20         (i>j)?tmp=y[i],y[i]=y[j],y[j]=tmp,0:1;
21         for (int l = n >> 1; (j ^= l) < l; l >>= 1);
22     }
23 }
24 void init()
25 {
26     int h=n>>1;
27     for (int i = 0; i < h; ++i) w[i+h] = (Complex) {cos(2 * pi * i / n), sin(2 * pi * i / n)};
28     for (int i = h; i-- ; )w[i]=w[i<<1];
29 }
```

```

29 void dft(Complex *a)
30 {
31     Complex tmp;
32     for(int p = 2, m = 1; m != n; p = (m = p) << 1)
33         for(int i = 0; i != n; i += p) for(int j = 0; j != m; ++j)
34         {
35             tmp = a[i + j + m] * w[j + m];
36             a[i + j + m] = a[i + j] - tmp;
37             a[i + j] += tmp;
38         }
39 }
40
41 int main()
42 {
43     fread(S, 1, 1 << 20, stdin);
44     int na = F(), nb = F(), x;
45     for (int i = 0; i <= na; ++i) a[i].x=F();
46     for (int i = 0; i <= nb; ++i) b[i].x=F();
47     for (n = 1; n < na + nb + 1; n <= 1) ;
48     bit_reverse(a, buf_a);
49     bit_reverse(b, buf_b);
50     init();
51     dft(buf_a);
52     dft(buf_b);
53     for (int i = 0; i < n; ++i) c[i] = buf_a[i] * buf_b[i];
54     std::reverse(c + 1, c + n);
55     bit_reverse(c, buf_c);
56     dft(buf_c);
57     for (int i = 0; i <= na + nb; ++i) printf("%d%c", int(buf_c[i].x / n + 0.5), " \n"[i==na+nb]);
58     return 0;
59 }
```

MTT (gy)

```

1 // long double seems unnecessary
2 using number = double;
3 const int EXP = 15, POW2 = (1 << EXP) - 1, N = 3e5 + 10;;
4 const int64_t MOD = 1e9 + 7;
5 const number pi = std::acos((number) -1);
6
7 int nn, rev[N];
8
9 struct complex {
10     number r, i;
11     complex() : r(0), i(0) {}
12     complex(number theta) : r(std::cos(theta)), i(std::sin(theta)) {}
13     complex(number r, number i) : r(r), i(i) {}
14     friend complex operator+(const complex &a, const complex &b) {
15         return complex(a.r + b.r, a.i + b.i);
16     }
17     friend complex operator-(const complex &a, const complex &b) {
18         return complex(a.r - b.r, a.i - b.i);
19     }
20     friend complex operator*(const complex &a, const complex &b) {
21         return complex(a.r * b.r - a.i * b.i, a.r * b.i + a.i * b.r);
22     }
23     complex operator~() const {
24         return complex(r, -i);
25     }
26 } w[N];
```

```

27
28 void prepare(int len) {
29     int x = 0;
30     for (nn = 1; nn < len; nn <= 1) x++;
31     for (int i = 1; i < nn; i++) rev[i] = (rev[i >> 1] >> 1) | ((i & 1) << (x - 1));
32     for (int i = 0; i < nn; i++) w[i] = complex(2 * pi * i / nn);
33 }
34
35 void fft(complex *a) {
36     for (int i = 0; i < nn; i++) if (i < rev[i]) std::swap(a[i], a[rev[i]]);
37     for (int i = 2, d = nn >> 1; i <= nn; i <= 1, d >= 1)
38         for (int j = 0; j < nn; j += i) {
39             complex *l = a + j, *r = a + j + (i >> 1), *p = w, tp;
40             for (int k = 0; k < (i >> 1); k++, l++, r++, p += d)
41                 tp = *r * *p, *r = *l - tp, *l = *l + tp;
42         }
43 }
44
45 // A & B must be in [0, MOD), ret will be in [0, MOD)
46 // A & B & ret being int32 doesn't matter
47 void mtt_main(int n, int m, int64_t *A, int64_t *B, int64_t *ret) {
48     prepare(n + m);
49     static complex a[N], b[N], Da[N], Db[N], Dc[N], Dd[N];
50     for (int i = 0; i < nn; i++) a[i] = complex(A[i] & POW2, A[i] >> EXP);
51     for (int i = 0; i < nn; i++) b[i] = complex(B[i] & POW2, B[i] >> EXP);
52     fft(a), fft(b);
53     for (int i = 0; i < nn; i++) {
54         int j = (nn - i) & (nn - 1);
55         static complex da, db, dc, dd;
56         da = (a[i] + ~a[j]) * complex(0.5, 0);
57         db = (a[i] - ~a[j]) * complex(0, -0.5);
58         dc = (b[i] + ~b[j]) * complex(0.5, 0);
59         dd = (b[i] - ~b[j]) * complex(0, -0.5);
60         Da[j] = da * dc, Db[j] = db * dd, Dc[j] = dc * dd, Dd[j] = db * dd;
61     }
62     for (int i = 0; i < nn; i++) a[i] = Da[i] + Db[i] * complex(0, 1);
63     for (int i = 0; i < nn; i++) b[i] = Dc[i] + Dd[i] * complex(0, 1);
64     fft(a), fft(b);
65     for (int i = 0; i < nn; i++) {
66         static int64_t da, db, dc, dd;
67         da = int64_t(a[i].r / nn + 0.5) % MOD;
68         db = int64_t(a[i].i / nn + 0.5) % MOD;
69         dc = int64_t(b[i].r / nn + 0.5) % MOD;
70         dd = int64_t(b[i].i / nn + 0.5) % MOD;
71         ret[i] = (da + ((db + dc) << EXP) + (dd << (EXP << 1))) % MOD;
72     }
73 }

```

NTT (gy)

Prime	G	2^k
167772161	3	33554432
469762049	3	67108864
998244353	3	8388608
1004535809	3	2097152

```

1 int nn, rev[N];
2 int64_t w[N], invn;
3
4 void prepare(int len) {

```

```

5   int x = 0;
6   for (nn = 1; nn < len; nn <= 1) x++;
7   for (int i = 1; i < nn; i++) rev[i] = (rev[i >> 1] >> 1) | ((i & 1) << (x - 1));
8   w[0] = 1, w[1] = pow(G, (MOD - 1) / nn);
9   for (int i = 2; i < nn; i++) w[i] = w[i - 1] * w[1] % MOD;
10  invn = pow(nn, MOD - 2);
11 }
12
13 void ntt(int64_t *a) {
14  for (int i = 0; i < nn; i++) if (i < rev[i]) std::swap(a[i], a[rev[i]]);
15  for (int i = 2, d = nn >> 1; i <= nn; i <= 1, d >= 1)
16   for (int j = 0; j < nn; j += i) {
17    int64_t *l = a + j, *r = a + j + (i >> 1), *p = w, tp;
18    for (int k = 0; k < (i >> 1); k++, l++, r++, p += d) {
19     tp = *r * *p % MOD;
20     *r = *l - tp < 0 ? *l - tp + MOD : *l - tp;
21     *l = *l + tp < MOD ? *l + tp : *l + tp - MOD;
22    }
23   }
24 }
25
26 void ntt_main(int n, int m, int64_t *A, int64_t *B, int64_t *ret) {
27  prepare(n + m);
28  static int64_t a[N], b[N];
29  for (int i = 0; i < nn; i++) a[i] = A[i];
30  for (int i = 0; i < nn; i++) b[i] = B[i];
31  ntt(a), ntt(b);
32  for (int i = 0; i < nn; i++) (a[i] *= b[i]) %= MOD;
33  std::reverse(a + 1, a + nn);
34  ntt(a);
35  for (int i = 0; i < nn; i++) ret[i] = a[i] * invn % MOD;
36 }

```

FWT (lhy)

```

1 void fwt(int n, int *x, bool inv = false)
2 {
3  for(int i = 0; i < n; i++)
4   for(int j = 0; j < (1 << n); j++)
5    if((j >> i) & 1)
6    {
7     int p = x[j ^ (1 << i)], q = x[j];
8     if(!inv)
9     {
10      //xor
11      x[j ^ (1 << i)] = p - q;
12      x[j] = p + q;
13      //or
14      x[j ^ (1 << i)] = p;
15      x[j] = p + q;
16      //and
17      x[j ^ (1 << i)] = p + q;
18      x[j] = q;
19    }
20    else
21    {
22      //xor
23      x[j ^ (1 << i)] = (p + q) >> 1;
24      x[j] = (q - p) >> 1;
25      //or

```

```

26         x[j ^ (1 << i)] = p;
27         x[j] = q - p;
28         //and
29         x[j ^ (1 << i)] = p - q;
30         x[j] = q;
31     }
32 }
33 }
34
35 void solve(int n, int *a, int *b, int *c)
36 {
37     fwt(n, a);
38     fwt(n, b);
39     for(int i = 0; i < (1 << n); i++)
40         c[i] = a[i] * b[i];
41     fwt(n, c, 1);
42 }
```

多项式操作 (gy)

- 求逆:

$$A(x)B(x) \equiv 1 \pmod{x^t} \rightarrow A(x)(2B(x) - A(x)B^2(x)) \equiv 1 \pmod{x^{2t}}$$

- 平方根:

$$A^2(x) \equiv B(x) \pmod{x^t} \rightarrow \left(\frac{B(x) + A^2(x)}{2A(x)}\right)^2 \equiv B(x) \pmod{x^{2t}}$$

- \ln (常数项为 1):

$$A(x) = \ln B(x) \rightarrow A'(x) = \frac{B'(x)}{B(x)}$$

- \exp (常数项为 0):

$$B(x) \equiv e^{A(x)} \pmod{x^t} \rightarrow B(x)(1 - \ln B(x) + A(x)) \equiv e^{A(x)} \pmod{x^{2t}}$$

```

1 void inv_main(int n, int64_t *a, int64_t *b) {
2     if (n == 1) {
3         b[0] = pow(a[0], MOD - 2);
4         return;
5     }
6     inv_main(n >> 1, a, b);
7     prepare(n << 1);
8     static int64_t tmp[N];
9     for (int i = 0; i < n; i++) tmp[i] = a[i];
10    for (int i = n; i < nn; i++) tmp[i] = 0;
11    ntt(tmp), ntt(b);
12    for (int i = 0; i < nn; i++) b[i] = (MOD + 2 - tmp[i] * b[i] % MOD) * b[i] % MOD;
13    std::reverse(b + 1, b + nn);
14    ntt(b);
15    for (int i = 0; i < n; i++) b[i] = b[i] * invn % MOD;
16    for (int i = n; i < nn; i++) b[i] = 0;
17 }
```

2.13 篩 (ct,cxy,Nightfall)

杜教筛 (ct)

Dirichlet 卷积: $(f * g)(n) = \sum_{d|n} f(d)g\left(\frac{n}{d}\right)$

对于积性函数 $f(n)$, 求其前缀和 $S(n) = \sum_{i=1}^n f(i)$

寻找一个恰当的积性函数 $g(n)$, 使得 $g(n)$ 和 $(f * g)(n)$ 的前缀和都容易计算

$$\text{则 } g(1)S(n) = \sum_{i=1}^n (f * g)(i) - \sum_{i=2}^n ng(i)S(\lfloor \frac{n}{i} \rfloor)$$

$\mu(n)$ 和 $\phi(n)$ 取 $g(n) = 1$

两种常见形式:

- $S(n) = \sum_{i=1}^n (f * g)(i)$ 且 $g(i)$ 为完全积性函数

$$S(n) = \sum_{i=1}^n ((f * 1) * g)(i) - \sum_{i=2}^n S(\lfloor \frac{n}{i} \rfloor)g(i)$$

- $S(n) = \sum_{i=1}^n (f * g)(i)$

$$S(n) = \sum_{i=1}^n g(i) \sum_{ij \leq n} (f * 1)(j) - \sum_{i=2}^n S(\lfloor \frac{n}{i} \rfloor)$$

```

1 int phi[maxn], pr[maxn / 10], prcnt;
2 ll sph[maxn];
3 bool vis[maxn];
4 const int moha = 3333331;
5 struct Hash {
6     Hash *next;
7     int ps; ll ans;
8 } *last1[moha], mem[moha], *tot = mem;
9 inline ll S1(int n)
10 {
11     if (n < maxn) return sph[n];
12     for (R Hash *iter = last1[n % moha]; iter; iter = iter -> next)
13         if (iter -> ps == n) return iter -> ans;
14
15     ll ret = 1ll * n * (n + 1ll) / 2;
16     for (ll i = 2, j; i <= n; i = j + 1)
17     {
18         j = n / (n / i);
19         ret -= S1(n / i) * (j - i + 1);
20     }
21     *++tot = (Hash) {last1[n % moha], n, ret}; last1[n % moha] = tot;
22     return ret;
23 }
24 int main()
25 {
26     int T; scanf("%d", &T);
27     phi[1] = sph[1] = 1;
28     for (int i = 2; i < maxn; ++i)
29     {
30         if (!vis[i]) pr[prcnt] = i, phi[i] = i - 1;
31         sph[i] = sph[i - 1] + phi[i];
32         for (int j = 1; j <= prcnt && 1ll * i * pr[j] < maxn; ++j)
33         {
34             vis[i * pr[j]] = 1;
35             if (i % pr[j])
36                 phi[i * pr[j]] = phi[i] * (pr[j] - 1);
37             else
38             {
39                 phi[i * pr[j]] = phi[i] * pr[j];
40                 break;
41             }
42         }
43     }
44     for (; T; --T)
45     {

```

```

46     int N; scanf("%d", &N);
47     printf("%lld\n", S1(N));
48 }
49 return 0;
50 }
```

Extended Eratosthenes Sieve (Nightfall)

一般积性函数的前缀和，要求： $f(p)$ 为多项式

```

1 struct poly {
2     LL a[2];
3
4     poly() {}
5
6     int size() const { return 2; }
7
8     poly(LL x, LL y) {
9         a[0] = x;
10        a[1] = y;
11    }
12 };
13
14 poly operator*(poly a, int p) {
15     return poly(a.a[0], a.a[1] * p);
16 }
17
18 poly operator-(const poly &a, const poly &b) {
19     return poly(a.a[0] - b.a[0], a.a[1] - b.a[1]);
20 }
21
22 poly sum_fp(LL l, LL r) { // f(p) = 1 + p
23     return poly(r - l + 1, (l + r) * (r - l + 1) / 2);
24 }
25
26 LL fpk(LL p, LL k) { // f(p^k) = sum{i in 0..k | p^i}
27     LL res = 0, q = 1;
28     for (int i = 0; i <= k; ++i) {
29         res += q;
30         q *= p;
31     }
32     return res;
33 }
34
35 LL Value(poly p) { return p.a[0] + p.a[1]; }
36
37 LL n;
38 int m;
39 vector<poly> A, B;
40 vector<int> P;
41
42 //need w = ⌈n/k⌉, about O(w^0.7)
43 LL calc(LL w, int id, LL f) {
44     LL T = w > m ? Value(B[n / w]) : Value(A[w]);
45     if (id) T -= Value(A[P[id - 1]]);
46     LL ret = T * f;
47     for (int i = id; i < P.size(); ++i) {
48         int p = P[i], e = 1;
49         LL q = (LL) p * p;
50         if (q > w) break;
51         ret += calc(w / p, i + 1, f * fpk(p, 1));
52     }
53 }
```

```

52     while (1) {
53         ++e;
54         LL f2 = f * fpk(p, e);
55         ret += f2;
56         LL qq = q * p;
57         if (qq <= w) {
58             ret += calc(w / q, i + 1, f2);
59             q = qq;
60         } else break;
61     }
62 }
63 return ret;
64 }

65 void prepare(LL N) { // about O(n^0.67)
66     n = N;
67     m = (int) sqrt(n + .5L);
68     A.resize(m + 1);
69     B.resize(m + 1);
70     P.clear();
71     vector<int> isp;
72     isp.resize(m + 1, 1);
73     for (int i = 1; i <= m; ++i) {
74         A[i] = sum_fp(2, i);
75         B[i] = sum_fp(2, n / i);
76     }
77     for (int p = 2; p <= m; ++p) {
78         if (isp[p]) P.push_back(p);
79         for (int j : P) {
80             if (j * p > m) break;
81             isp[j * p] = 0;
82             if (j % p == 0) break;
83         }
84         if (!isp[p]) continue;
85         poly d = A[p - 1];
86         LL p2 = (LL) p * p;
87         int to = (int) min(n / p2, (LL) m);
88         for (int i = 1; i <= m / p; ++i)
89             B[i] = B[i] - (B[i * p] - d) * p;
90         for (int i = m / p + 1; i <= to; ++i)
91             B[i] = B[i] - (A[n / p / i] - d) * p;
92         for (int i = m; i >= p2; --i)
93             A[i] = A[i] - (A[i / p] - d) * p;
94     }
95 }
96
97 main() : prepare(n);
98 LL ans = calc(n, 0, 1);
99
100

```

Min25 篩 (cxy)

Min25 篩

$O(\frac{n^{\frac{3}{4}}}{\log n})$ 求 $\sum_{i=1}^n f(i)$, 其中 $f(i)$ 是积性函数且 $f(p^e)$ 是关于 p 的低阶多项式

$$g(n, j) = \sum_{i=1}^n [i \text{ is a prime or } \min_p(i) > p_j] i^k$$

$$g(n, j) = g(n, j - 1) - p_j^k \left(g\left(\frac{n}{p_j}, j - 1\right) - g(p_{j-1}, j - 1) \right)$$

$g(n) = g(n, x)$, 其中 p_x 是最后一个 $\leq \sqrt{n}$ 的素数

$$sp_n = \sum_{i=1}^n p_i^k$$

$$S(n, x) = g(n) - sp_x + \sum_{p_k^e \leq n \wedge k > x} f(p_k^e) \left(S\left(\frac{n}{p_k^e}, k\right) + [e \neq 1] \right)$$

$$\sum_{i=1}^n = S(n, 0)$$

实现 $f(p^k) = p^k(p^k - 1)$

```

1  namespace min25 {
2      const int maxq = 2e5 + 10;
3      const int mod = 1e9 + 7, inv6 = 166666668, inv2 = 500000004, inv3 = 333333336;
4      int P[maxq], cnt;
5      bool isNotP[maxq];
6
7      inline ll f1(ll pk) { pk %= mod; return pk % mod; }
8      inline ll f2(ll pk) { pk %= mod; return pk * pk % mod; }
9      inline ll f(ll pk) { return (f2(pk) - f1(pk) + mod) % mod; }
10
11     inline ll sf1(ll pk) { pk %= mod; return (pk + 1) % mod * pk % mod * inv2 % mod; }
12     inline ll sf2(ll pk) { pk %= mod; return pk % mod * (pk + 1) % mod * (2 * pk + 1) % mod * inv6 %
13         ↪ mod; }
14     inline ll sf(ll pk) { pk %= mod; return (sf2(pk) - sf1(pk) + mod) % mod; }
15     int sfp1[maxq], sfp2[maxq];
16
17     void sieve(int n) {
18         for (int i = 2; i <= n; i++) {
19             if (!isNotP[i]) {
20                 P[++cnt] = i;
21                 sfp1[cnt] = (sfp1[cnt - 1] + f1(i)) % mod;
22                 sfp2[cnt] = (sfp2[cnt - 1] + f2(i)) % mod;
23             }
24             for (int j = 1; j <= cnt && i * P[j] <= n; j++) {
25                 isNotP[i * P[j]] = true;
26                 if (i % P[j] == 0) break;
27             }
28         }
29
30         ll w[maxq];
31         int id1[maxq], id2[maxq];
32         ll N;
33         int SQRT;
34         int id(ll x) {
35             return x <= SQRT ? id1[x] : id2[N / x];
36         }
37         int g1[maxq], g2[maxq];
38
39         int s(ll n, int x) {
40             if (P[x] >= n) return 0;
41             int k = id(n);
42             int ans = (0ll + g2[k] - g1[k] + sfp1[x] - sfp2[x]) % mod;
43             for (int k = x + 1; k <= cnt && 1ll * P[k] * P[k] <= n; k++) {
44                 ll Pe = P[k];
45                 for (int e = 1; Pe <= n; e++, Pe *= P[k])
46                     (ans += f(Pe) * (s(n / Pe, k) + (e != 1)) % mod) %= mod;
47             }
48         }
49     }
50 }
```

```

48     return ans < 0 ? ans + mod : ans;
49 }
50
51 int solve(ll n) {
52     N = n; SQRT = sqrt(n); sieve(SQRT);
53     int m = 0;
54
55     for (ll i = 1, j; i <= n; i = j + 1) {
56         j = n / (n / i); w[++m] = n / i;
57         g1[m] = (sf1(w[m]) - f1(1) + mod) % mod;
58         g2[m] = (sf2(w[m]) - f2(1) + mod) % mod;
59         if (w[m] <= SQRT) id1[w[m]] = m; else id2[n / w[m]] = m;
60     }
61
62     for (int j = 1; j <= cnt; j++) {
63         for (int i = 1; i <= m && 1ll * P[j] * P[j] <= w[i]; i++) {
64             int k = id(w[i] / P[j]);
65             (g1[i] -= f1(P[j]) * (g1[k] - sfp1[j - 1]) % mod) %= mod;
66             (g2[i] -= f2(P[j]) * (g2[k] - sfp2[j - 1]) % mod) %= mod;
67             g1[i] < 0 ? g1[i] += mod : 0; g2[i] < 0 ? g2[i] += mod : 0;
68         }
69     }
70     return (s(n, 0) + 1) % mod; // f(1) = 1
71 }
72 }
```

洲阁筛 (ct)

实现区间 $[a, b]$ 里有多少个数满足：含有至少一个 $> p$ 的素因数

```

1 typedef long long ll;
2 int pr[maxn / 10], prcnt, N, lim, fp[maxn];
3 bool vis[maxn];
4 const int moha = 333331;
5 struct Hash {
6     int next, ans; ll key;
7 } mem[moha * 21], *tot = mem;
8 int last[moha];
9 /*
10 inline int qpow(R int power)
11 {
12     R int ret = 1;
13     for (R int i = 1; i <= power; ++i) ret <= 1;
14     return ret;
15 }
16 */
17 int S(R int n, R int k)
18 {
19     if (!k) return 1;
20     if (n <= pr[k]) return n;
21     if (n < 1ll * pr[k] * pr[k])
22     {
23         R int nk = fp[(int) sqrt(n)];
24         return S(n, k - 1) + n / pr[k];
25     }
26     R int ret;
27     R ll key = 1ll * n * prcnt + k - 1; R int pos = key % moha;
28     for (R int iter = last[pos]; iter; iter = mem[iter].next) if (mem[iter].key == key) return
29         → mem[iter].ans;
30     ret = S(n, k - 1) + S(n / pr[k], k);
```

```

30     *++tot = (Hash) {last[pos], ret, key}; last[pos] = tot - mem;
31     return ret;
32 }
33 inline int calc(R int x)
34 {
35     N = x; lim = (int) sqrt(x);
36     memset(last, 0, sizeof last);
37     tot = mem;
38     return S(x, prcnt);
39 }
40 int main()
41 {
42 //    printf("%d\n", (sizeof (last) + sizeof (mem) + sizeof (pr)) >> 20);
43 R int p, a, b; scanf("%d%d%d", &p, &a, &b);
44 if (p == 1) {printf("%d\n", b - a + 1 - (a <= 1 && 1 <= b)); return 0;}
45 for (R int i = 2; i <= p; ++i)
46 {
47     if (!vis[i]) pr[++prcnt] = i;
48     fp[i] = fp[i - 1] + !vis[i];
49     for (R int j = 1; j <= prcnt && i * pr[j] <= p; ++j)
50     {
51         vis[i * pr[j]] = 1;
52         if (i % pr[j] == 0) break;
53     }
54 }
55 //    printf("%d\n", prcnt);
56 //    printf("%d\n", S(b, prcnt));
57 printf("%d\n", b - a + 1 - calc(b) + calc(a - 1));
58 //    printf("%d\n", tot - mem);
59 return 0;
60 }
```

实现 $\sum_{i=1}^n \sum_{j=1}^n \text{sgcd}(i, j)^k$, 其中 $\text{sgcd}(i, j) = \begin{cases} 0 & \text{gcd}(i, j) = 1 \\ \text{The second-largest common divisor of } i \text{ and } j & \text{otherwise} \end{cases}$

```

1 const int moha = 3102331;
2 int lim;
3 int pr[maxn / 10], prcnt, fp[maxn], K, pkk[maxn / 10], cnt2[110], N;
4 uint phi[maxn], ksum[maxn / 10], kpw[maxn], kpow[maxn], dif[110], y[110], inv[110];
5 bool vis[maxn];
6 inline uint qpow(R uint base, R uint power)
7 {
8     R uint ret = 1;
9     for (; power; power >= 1, base *= base)
10        power & 1 ? ret *= base : 0;
11    return ret;
12 }
13 uint phi1[maxn];
14 uint Phi(R int x)
15 {
16     if (x <= lim) return phi[x];
17     if (phi1[N / x]) return phi1[N / x];
18     R uint ret = 1ll * x * (x + 1) / 2;
19     for (R int i = 2, j; i <= x; i = j + 1)
20     {
21         j = x / (x / i);
22         ret -= (j - i + 1) * Phi(x / i);
23     }
24     return phi1[N / x] = ret;
25 }
```

```

26 std::vector<uint> p1[maxn / 10], p2[maxn / 10], f1[maxn / 10], f2[maxn / 10], s1[maxn / 10], s2[maxn /
27   ~ 10];
28 uint P(R int x, R int k)
29 {
30     if (!k || !x) return 0;
31     if (pr[k] * pr[k] > x) return P(x, fp[(int) sqrt(x)]);
32
33     if (x <= N / x)
34     {
35         if (p1[x].size() > k && p1[x][k]) return p1[x][k];
36         while (p1[x].size() <= k) p1[x].push_back(0);
37         uint ans = P(x, k - 1) + (x / pr[k]) - P(x / pr[k], k - 1) - k;
38         return p1[x][k] = ans;
39     }
40     else
41     {
42         if (p2[N / x].size() > k && p2[N / x][k]) return p2[N / x][k];
43         while (p2[N / x].size() <= k) p2[N / x].push_back(0);
44         uint ans = P(x, k - 1) + (x / pr[k]) - P(x / pr[k], k - 1) - k;
45         return p2[N / x][k] = ans;
46     }
47 inline uint getinv(R uint x)
48 {
49     return qpow(x, ~0u >> 1);
50 }
51 inline uint C(R int a, R int b)
52 {
53     R uint num = 0;
54     if (b > a) return 0;
55
56     R uint ret = 1;
57     for (R int i = 1; i <= b; ++i)
58     {
59         R int tmp = a - i + 1;
60         while (tmp % 2 == 0) ++num, tmp >>= 1;
61         ret *= tmp;
62     }
63 //    printf("C(%d, %d) = %u\n", a, b, ret * qpow(2, num) * inv[b]);
64     return ret * qpow(2, num - cnt2[b]) * inv[b];
65 }
66 std::map<int, int> Sky;
67 inline uint Sk(R int x)
68 {
69     uint ret = 0;
70     if (x <= lim) return kpw[x];
71     if (Sky[x]) return Sky[x];
72     for (R int i = 1; i <= K + 1; ++i) ret += dif[i] * C(x, i);
73     return Sky[x] = ret;
74 }
75 uint F(R int x, R int k)
76 {
77     if (!k || !x) return 0;
78     if (pr[k] * pr[k] > x) return F(x, fp[(int) sqrt(x)]);
79     if (x <= N / x)
80     {
81         if (f1[x].size() > k && f1[x][k]) return f1[x][k];
82         while (f1[x].size() <= k) f1[x].push_back(0);
83         uint ans = F(x, k - 1) + (Sk(x / pr[k]) - 1 - ksum[k - 1] - F(x / pr[k], k - 1)) * pkk[k];
84         return f1[x][k] = ans;
85     }

```

```

86     else
87     {
88         if (f2[N / x].size() > k && f2[N / x][k]) return f2[N / x][k];
89         while (f2[N / x].size() <= k) f2[N / x].push_back(0);
90         uint ans = F(x, k - 1) + (Sk(x / pr[k]) - 1 - ksum[k - 1] - F(x / pr[k], k - 1)) * pkk[k];
91         return f2[N / x][k] = ans;
92     }
93 }
94 uint S(R int x, R int k)
95 {
96     if (!k || !x) return 0;
97     if (pr[k] * pr[k] > x) return S(x, fp[(int) sqrt(x)]);
98
99     if (x <= N / x)
100    {
101        if (s1[x].size() > k && s1[x][k]) return s1[x][k];
102        while (s1[x].size() <= k) s1[x].push_back(0);
103        uint ans = S(x, k - 1) + Sk(x / pr[k]) - 1 - ksum[k - 1] - F(x / pr[k], k - 1);
104        return s1[x][k] = ans;
105    }
106    else
107    {
108        if (s2[N / x].size() > k && s2[N / x][k]) return s2[N / x][k];
109        while (s2[N / x].size() <= k) s2[N / x].push_back(0);
110        uint ans = S(x, k - 1) + Sk(x / pr[k]) - 1 - ksum[k - 1] - F(x / pr[k], k - 1);
111        return s2[N / x][k] = ans;
112    }
113 }
114 inline uint Sum(R int x)
115 {
116     uint pp = S(x, fp[(int) sqrt(x)]);
117     return x - 1 - P(x, fp[(int) sqrt(x)]) + S(x, fp[(int) sqrt(x)]);
118 }
119 int main()
120 {
121 //     printf("%d\n", (sizeof(phi) * 5 + sizeof(last1) * 4 + sizeof(mem)) >> 20);
122     scanf("%d%d", &N, &K);
123     lim = (int) pow(N, 2.0 / 3.0);
124 //     printf("%d\n", lim);
125     phi[1] = 1; kpw[1] = 1;
126     for (R int i = 2; i <= lim; ++i)
127     {
128         if (!vis[i]) pr[++prcnt] = i, phi[i] = i - 1, ksum[prcnt] = ksum[prcnt - 1] + (pkk[prcnt] =
129             → kpow[i] = qpow(i, K));
130         fp[i] = fp[i - 1] + !vis[i];
131         kpw[i] = kpw[i - 1] + kpow[i];
132         for (R int j = 1; j <= prcnt && lll * i * pr[j] <= lim; ++j)
133         {
134             vis[i * pr[j]] = 1;
135             kpow[i * pr[j]] = kpow[i] * kpow[pr[j]];
136             if (i % pr[j]) phi[i * pr[j]] = phi[i] * (pr[j] - 1);
137             else {phi[i * pr[j]] = phi[i] * pr[j]; break;}
138         }
139     }
140     for (R int i = 2; i <= lim; ++i) phi[i] += phi[i - 1];
141     R uint last = 0, ans = 0;
142     for (R int i = 1; i <= K + 1; ++i) y[i] = kpw[i];
143     uint ppw = 1; R int ccnt = 0;
144     for (R int i = 1; i <= K + 1; ++i)
145     {
146         R int tmp = i;

```

```

146     while (tmp % 2 == 0) tmp >>= 1, ++ccnt;
147     ppw *= getinv(tmp); inv[i] = ppw; cnt2[i] = ccnt;
148     // printf("inv[%d] = %u\n", i, inv[i]);
149 }
150 for (R int i = 1; i <= K + 1; ++i)
151 {
152     for (R int j = K + 1; j >= i; --j)
153         y[j] -= y[j - 1];
154     dif[i] = y[i];
155     // printf("%d\n", dif[i]);
156 }
157 // for (R int i = 1; i <= 5; ++i) printf("kpow[%d] = %u\n", i, Sk(i));
158 for (R int i = 2, j; i <= N; i = j + 1)
159 {
160     j = N / (N / i);
161     R uint Phi2 = 2 * Phi(N / i) - 1, dk = Sum(j);
162     ans += (dk - last) * Phi2;
163     last = dk;
164 }
165 printf("%u\n", ans);
166 return 0;
167 }
```

2.14 BSGS (ct,Durandal)

BSGS (ct)

p 是素数, 返回 $\min\{x \geq 0 \mid y^x \equiv z \pmod{p}\}$

```

1 const int mod = 19260817;
2 struct Hash
3 {
4     Hash *next;
5     int key, val;
6 } *last[mod], mem[100000], *tot = mem;
7 inline void insert(R int x, R int v)
8 {
9     *++tot = (Hash) {last[x % mod], x, v}; last[x % mod] = tot;
10 }
11 inline int query(R int x)
12 {
13     for (R Hash *iter = last[x % mod]; iter; iter = iter -> next)
14         if (iter -> key == x) return iter -> val;
15     return -1;
16 }
17 inline void del(R int x)
18 {
19     last[x % mod] = 0;
20 }
21 int main()
22 {
23     for (; T; --T)
24     {
25         R int y, z, p; scanf("%d%d%d", &y, &z, &p);
26         R int m = (int) sqrt(p * 1.0);
27         y %= p; z %= p;
28         if (!y && !z) {puts("0"); continue;}
29         if (!y) {puts("Orz, I cannot find x!"); continue;}
30         R int pw = 1;
31         for (R int i = 0; i < m; ++i, pw = 1ll * pw * y % p) insert(1ll * z * pw % p, i);
```

```

32     R int ans = -1;
33     for (R int i = 1, t, pw2 = pw; i <= p / m + 1; ++i, pw2 = 1ll * pw2 * pw % p)
34         if ((t = query(pw2)) != -1) {ans = i * m - t; break;}
35     if (ans == -1) puts("Orz, I cannot find x!");
36     else printf("%d\n", ans );
37     tot = mem; pw = 1;
38     for (R int i = 0; i < m; ++i, pw = 1ll * pw * y % p) del(1ll * z * pw % p);
39 }
40 return 0;
41 }
```

ex-BSGS (Durandal)

必须满足 $0 \leq a < p$, $0 \leq b < p$, 返回 $\min\{x \geq 0 \mid a^x \equiv b \pmod{p}\}$

```

1 int64_t ex_bsgs(int64_t a, int64_t b, int64_t p) {
2     if (b == 1)
3         return 0;
4     int64_t t, d = 1, k = 0;
5     while ((t = std::__gcd(a, p)) != 1) {
6         if (b % t) return -1;
7         k++, b /= t, p /= t, d = d * (a / t) % p;
8         if (b == d) return k;
9     }
10    map.clear();
11    int64_t m = std::ceil(std::sqrt((long double) p));
12    int64_t a_m = pow_mod(a, m, p);
13    int64_t mul = b;
14    for (int j = 1; j <= m; j++) {
15        (mul *= a) %= p;
16        map[mul] = j;
17    }
18    for (int i = 1; i <= m; i++) {
19        (d *= a_m) %= p;
20        if (map.count(d))
21            return i * m - map[d] + k;
22    }
23    return -1;
24 }
25
26 int main() {
27     int64_t a, b, p;
28     while (scanf("%lld%lld%lld", &a, &b, &p) != EOF)
29         printf("%lld\n", ex_bsgs(a, b, p));
30     return 0;
31 }
```

2.15 直线下整点个数 (gy)

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

```

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

```

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

2.16 Pell equation (gy)

$x^2 - ny^2 = 1$ 有解当且仅当 n 不为完全平方数

求其特解 (x_0, y_0)

其通解为 $(x_{k+1}, y_{k+1}) = (x_0x_k + ny_0y_k, x_0y_k + y_0x_k)$

```

1 std::pair<int64_t, int64_t> pell(int64_t n) {
2     static int64_t p[N], q[N], g[N], h[N], a[N];
3     p[1] = q[0] = h[1] = 1;
4     p[0] = q[1] = g[1] = 0;
5     a[2] = std::sqrt(n) + 1e-7L;
6     for (int i = 2; true; i++) {
7         g[i] = -g[i - 1] + a[i] * h[i - 1];
8         h[i] = (n - g[i] * g[i]) / h[i - 1];
9         a[i + 1] = (g[i] + a[2]) / h[i];
10        p[i] = a[i] * p[i - 1] + p[i - 2];
11        q[i] = a[i] * q[i - 1] + q[i - 2];
12        if (p[i] * p[i] - n * q[i] * q[i] == 1)
13            return std::make_pair(p[i], q[i]);
14    }
15 }
```

2.17 单纯形 (gy)

返回 $x_{m \times 1}$ 使得 $\max\{c_{1 \times m} \cdot x_{m \times 1} \mid x_{m \times 1} \geq 0_{m \times 1}, A_{n \times m} \cdot x_{m \times 1} \leq b_{n \times 1}\}$

```

1 std::vector<std::vector<double>> A;
2 std::vector<double> b, c;
3
4 const double eps = 1e-8;
5 std::vector<double> simplex() {
6     int n = int(A.size()), m = int(A[0].size()) + 1, r = n, s = m - 1;
7     std::vector<std::vector<double>> D(n + 2, std::vector<double>(m + 1));
8     std::vector<int> ix(n + m);
9     for (int i = 0; i < n + m; i++) ix[i] = i;
10    for (int i = 0; i < n; i++) {
11        for (int j = 0; j < m - 1; j++) D[i][j] = -A[i][j];
12        D[i][m - 1] = 1;
13        D[i][m] = b[i];
14        if (D[r][m] > D[i][m]) r = i;
15    }
16    for (int j = 0; j < m - 1; j++) D[n][j] = c[j];
17    D[n + 1][m - 1] = -1;
18    for (double d; true; ) {
19        if (r < n) {
20            std::swap(ix[s], ix[r + m]);
21            D[r][s] = 1. / D[r][s];
22            for (int j = 0; j <= m; j++) if (j != s) D[r][j] *= -D[r][s];
23            for (int i = 0; i <= n + 1; i++) if (i != r) {
24                for (int j = 0; j <= m; j++)
25                    if (j != s) D[i][j] += D[r][j] * D[i][s];
26                D[i][s] *= D[r][s];
27            }
28        }
29        r = s = -1;
```

```

30     for (int j = 0; j < m; j++) if (s < 0 || ix[s] > ix[j])
31         if (D[n + 1][j] > eps || D[n + 1][j] > -eps && D[n][j] > eps)
32             s = j;
33     if (s < 0) break;
34     for (int i = 0; i < n; i++) if (D[i][s] < -eps)
35         if (r < 0 || (d = D[r][m] / D[r][s] - D[i][m] / D[i][s]) < eps || d < eps && ix[r + m] >
36             ix[i + m])
37             r = i;
38     if (r < 0) assert(false);
39     if (D[n + 1][m] < -eps) assert(false);
40     std::vector<double> x(m - 1);
41     for (int i = m; i < n + m; i++)
42         if (ix[i] < m - 1)
43             x[ix[i]] = D[i - m][m];
44     return x;
45 }

```

2.18 数学知识 (gy)

扩展欧拉定理

$$a^c \equiv \begin{cases} a^c & c < \phi(m) \\ a^{c \bmod \phi(m)+\phi(m)} & c \geq \phi(m) \end{cases}$$

类欧几里得算法

$$f(a, b, c, n) = \sum_{i=0}^n \lfloor \frac{ai+b}{c} \rfloor, g(a, b, c, n) = \sum_{i=0}^n i \lfloor \frac{ai+b}{c} \rfloor, h(a, b, c, n) = \sum_{i=0}^n \lfloor \frac{ai+b}{c} \rfloor^2, m = \lfloor \frac{an+b}{c} \rfloor$$

$a \geq c$ or $b \geq c$:

$$f(a, b, c, n) = f(a \bmod c, b \bmod c, c, n) + \frac{1}{2}n(n+1)\lfloor \frac{a}{c} \rfloor + (n+1)\lfloor \frac{b}{c} \rfloor$$

$$g(a, b, c, n) = g(a \bmod c, b \bmod c, c, n) + \frac{1}{6}n(n+1)(2n+1)\lfloor \frac{a}{c} \rfloor + \frac{1}{2}n(n+1)\lfloor \frac{b}{c} \rfloor$$

$$h(a, b, c, n) = 2\lfloor \frac{b}{c} \rfloor f(a \bmod c, b \bmod c, c, n) + 2\lfloor \frac{a}{c} \rfloor g(a \bmod c, b \bmod c, c, n) + h(a \bmod c, b \bmod c, c, n) + \frac{1}{6}n(n+1)(2n+1)\lfloor \frac{a}{c} \rfloor^2 + (n+1)\lfloor \frac{b}{c} \rfloor^2 + n(n+1)\lfloor \frac{a}{c} \rfloor \lfloor \frac{b}{c} \rfloor$$

$a < c$ and $b < c$:

$$f(a, b, c, n) = nm - f(c, c-b-1, a, m-1)$$

$$g(a, b, c, n) = \frac{1}{2}(nm(n+1) - f(c, c-b-1, a, m-1) - h(c, c-b-1, a, m-1))$$

$$h(a, b, c, n) = nm(m+1) - f(a, b, c, n) - 2f(c, c-b-1, a, m-1) - 2g(c, c-b-1, a, m-1)$$

原根

当 $\gcd(a, m) = 1$ 时, 使 $a^x \equiv 1 \pmod{m}$ 成立的最小正整数 x 称为 a 对于模 m 的阶, 计为 $\text{ord}_m(a)$ 。

阶的性质: $a^n \equiv 1 \pmod{m}$ 的充要条件是 $\text{ord}_m(a) | n$, 可推出 $\text{ord}_m(a) | \psi(m)$ 。

当 $\text{ord}_m(g) = \psi(m)$ 时, 则称 g 是模 n 的一个原根, $g^0, g^1, \dots, g^{\psi(m)-1}$ 覆盖了 m 以内所有与 m 互素的数。

原根存在的充要条件: $m = 2, 4, p^k, 2p^k$, 其中 p 为奇素数, $k \in \mathbb{N}^*$

求和公式

$$\bullet \quad \sum_{k=1}^n (2k-1)^2 = \frac{1}{3}n(4n^2 - 1)$$

$$\bullet \quad \sum_{k=1}^n k^3 = \frac{1}{4}n^2(n+1)^2$$

- $\sum_{k=1}^n (2k-1)^3 = n^2(2n^2-1)$
- $\sum_{k=1}^n k^4 = \frac{1}{30}n(n+1)(2n+1)(3n^2+3n-1)$
- $\sum_{k=1}^n k^5 = \frac{1}{12}n^2(n+1)^2(2n^2+2n-1)$
- $\sum_{k=1}^n k(k+1) = \frac{1}{3}n(n+1)(n+2)$
- $\sum_{k=1}^n k(k+1)(k+2) = \frac{1}{4}n(n+1)(n+2)(n+3)$
- $\sum_{k=1}^n k(k+1)(k+2)(k+3) = \frac{1}{5}n(n+1)(n+2)(n+3)(n+4)$

错排公式

D_n 表示 n 个元素错位排列的方案数

$$D_1 = 0, D_2 = 1, D_n = (n-1)(D_{n-2} + D_{n-1}), n \geq 3$$

$$D_n = n! \cdot \left(1 - \frac{1}{1!} + \frac{1}{2!} - \cdots + (-1)^n \frac{1}{n!}\right)$$

Fibonacci sequence

$$F_0 = 0, F_1 = 1$$

$$F_n = F_{n-1} + F_{n-2}$$

$$F_{n+1} \cdot F_{n-1} - F_n^2 = (-1)^n$$

$$F_{-n} = (-1)^n F_n$$

$$F_{n+k} = F_k \cdot F_{n+1} + F_{k-1} \cdot F_n$$

$$\gcd(F_m, F_n) = F_{\gcd(m, n)}$$

$$F_m \mid F_n^2 \Leftrightarrow nF_n \mid m$$

$$F_n = \frac{\varphi^n - \Psi^n}{\sqrt{5}}, \varphi = \frac{1 + \sqrt{5}}{2}, \Psi = \frac{1 - \sqrt{5}}{2}$$

$$F_n = \lfloor \frac{\varphi^n}{\sqrt{5}} + \frac{1}{2} \rfloor, n \geq 0, n(F) = \lfloor \log_{\varphi}(F \cdot \sqrt{5} + \frac{1}{2}) \rfloor$$

Stirling number (1st kind)

用 $[n]_k$ 表示 Stirling number (1st kind), 为将 n 个元素分成 k 个环的方案数

$$[n+1]_k = n[n]_k + [n]_{k-1}, k > 0$$

$$[0]_0 = 1, [n]_0 = [0]_n = 0, n > 0$$

$[n]_k$ 为将 n 个元素分成 k 个环的方案数

$$[x]_{-n} = \sum_{k=0}^n \langle\langle n \rangle\rangle \binom{x+k}{2n}$$

$$x^n = \sum_{k=0}^n [n]_k (-1)^{n-k} x^k$$

$$x^{\bar{n}} = \sum_{k=0}^n [n]_k x^k$$

Stirling number (2nd kind)

用 $\{n\}_k$ 表示 Stirling number (2nd kind), 为将 n 个元素划分成 k 个非空集合的方案数

$$\{n+1\}_k = k\{n\}_k + \{n\}_{k-1}, k > 0, \{0\}_0 = 1, \{n\}_0 = \{0\}_n = 0, n > 0$$

$$\{n\}_k = \frac{1}{k!} \sum_{j=0}^k (-1)^{k-j} \binom{k}{j} j^n, \{x\}_{-n} = \sum_{k=0}^n \langle\langle n \rangle\rangle \binom{x+n-k-1}{2n}$$

Catalan number

c_n 表示长度为 $2n$ 的合法括号序的数量
 $c_1 = 1, c_{n+1} = \sum_{i=1}^n c_i \times c_{n+1-i}, c_n = \frac{\binom{2n}{n}}{n+1}$

Bell number

B_n 表示基数为 n 的集合的划分方案数

$$B_i = \begin{cases} 1 & i = 0 \\ \sum_{k=0}^{i-1} \binom{i-1}{k} B_k & i > 0 \end{cases}$$

$$B_n = \sum_{k=0}^n \left\{ \begin{matrix} n \\ k \end{matrix} \right\}$$

$$B_{p^m+n} \equiv mB_n + B_{n+1} \pmod{p}$$

五边形数定理

$p(n)$ 表示将 n 划分为若干个正整数之和的方案数

$$p(n) = \sum_{k \in \mathbb{N}^*} (-1)^{k-1} p\left(n - \frac{k(3k-1)}{2}\right)$$

Bernoulli number

$$\sum_{j=0}^m \binom{m+1}{j} B_j = 0, m > 0$$

$$B_i = \begin{cases} 1 & i = 0 \\ \sum_{j=0}^{i-1} \binom{i+1}{j} B_j & i > 0 \end{cases}$$

$$\sum_{k=1}^n k^m = \frac{1}{m+1} \sum_{k=0}^m \binom{m+1}{k} B_k n^{m+1-k}$$

$$\text{生成函数 } \sum_{i=0}^{\infty} B_i \frac{x^i}{i!} = \frac{x}{e^x - 1} = \frac{1}{\sum_{i=0}^{\infty} \frac{x^i}{(i+1)!}}$$

Stirling permutation

$1, 1, 2, 2, \dots, n, n$ 的排列中，对于每个 i ，都有两个 i 之间的数大于 i
 排列方案数为 $(2n-1)!!$

Eulerian number

$\langle \begin{smallmatrix} n \\ k \end{smallmatrix} \rangle$ 表示 1 到 n 的排列中，恰有 k 个数比前一个大的方案数

$$\langle \begin{smallmatrix} n \\ 0 \end{smallmatrix} \rangle = \langle \begin{smallmatrix} n \\ n-1 \end{smallmatrix} \rangle = 1, \langle \begin{smallmatrix} 0 \\ m \end{smallmatrix} \rangle = [m=0]$$

$$\langle \begin{smallmatrix} n \\ m \end{smallmatrix} \rangle = \langle \begin{smallmatrix} n \\ n-1-m \end{smallmatrix} \rangle$$

$$\langle \begin{smallmatrix} n \\ m \end{smallmatrix} \rangle = (m+1) \langle \begin{smallmatrix} n-1 \\ m \end{smallmatrix} \rangle + (n-m) \langle \begin{smallmatrix} n-1 \\ m-1 \end{smallmatrix} \rangle$$

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

Eulerian number (2nd kind)

$\langle\langle \begin{smallmatrix} n \\ k \end{smallmatrix} \rangle\rangle$ 表示 Stirling permutation 中，恰有 k 个数比前一个大的方案数

$$\langle\langle \begin{smallmatrix} n \\ m \end{smallmatrix} \rangle\rangle = (2n-m-1) \langle\langle \begin{smallmatrix} n-1 \\ m-1 \end{smallmatrix} \rangle\rangle + (m+1) \langle\langle \begin{smallmatrix} n-1 \\ m \end{smallmatrix} \rangle\rangle$$

$$\langle\langle \begin{smallmatrix} n \\ 0 \end{smallmatrix} \rangle\rangle = 1, \langle\langle \begin{smallmatrix} 0 \\ m \end{smallmatrix} \rangle\rangle = [m=0]$$

二项式反演

$$\begin{aligned} f(n) = \sum_{i=0}^n \binom{n}{i} g(i) &\Leftrightarrow g(n) = \sum_{i=0}^n (-1)^{n-i} \binom{n}{i} f(i) \\ f(n) = \sum_{i=0}^n (-1)^i \binom{n}{i} g(i) &\Leftrightarrow g(n) = \sum_{i=0}^n (-1)^i \binom{n}{i} f(i) \end{aligned}$$

Stirling 反演

$$f(n) = \sum_{i=0}^n \{ \binom{n}{i} \} g(i) \Leftrightarrow g(n) = \sum_{i=0}^n (-1)^{n-i} [\binom{n}{i}] f(i)$$

Möbius function

$$\mu(n) = \begin{cases} 1 & n \text{ square-free, even number of prime factors} \\ -1 & n \text{ square-free, odd number of prime factors} \\ 0 & n \text{ has a squared prime factor} \end{cases}$$

$$\sum_{d|n} \mu(d) = \begin{cases} 1 & n = 1 \\ 0 & n > 1 \end{cases}$$

$$f(n) = \sum_{d|n} g(d) \Leftrightarrow g(n) = \sum_{d|n} \mu(d) f\left(\frac{n}{d}\right)$$

$$f(n) = \sum_{n|d} g(d) \Leftrightarrow g(n) = \sum_{n|d} \mu\left(\frac{d}{n}\right) f(d)$$

Lagrange polynomial

给定次数为 n 的多项式函数 $L(x)$ 上的 $n+1$ 个点 $(x_0, y_0), (x_1, y_1), \dots, (x_n, y_n)$

$$\text{则 } L(x) = \sum_{j=0}^n y_j \prod_{0 \leq m \leq n, m \neq j} \frac{x - x_m}{x_j - x_m}$$

Burnside lemma

Let G be a finite group that acts on a set X . For each g in G let X^g denote the set of elements in X that are fixed by g (also said to be left invariant by g), i.e. $X^g = \{x \in X \mid g.x = x\}$. Burnside's lemma asserts the following formula for the number of orbits, denoted $|X/G|$:

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

Pólya theorem

设 \overline{G} 是 n 个对象的置换群, 用 m 种颜色对 n 个对象染色, 则不同染色方案为:

$$L = \frac{1}{|\overline{G}|} (m^{c(\overline{P_1})} + m^{c(\overline{P_2})} + \dots + m^{c(\overline{P_g})})$$

其中 $\overline{G} = \{\overline{P_1}, \overline{P_2}, \dots, \overline{P_g}\}$, $c(\overline{P_k})$ 为 $\overline{P_k}$ 的循环节数

Geometry

3.1 点、直线、圆 (gy)

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

```

47     }
48     friend point operator-(const point &a, const point &b) {
49         return point(a.x - b.x, a.y - b.y);
50     }
51     friend point operator*(const point &a, number b) {
52         return point(a.x * b, a.y * b);
53     }
54     friend point operator/(const point &a, number b) {
55         return point(a.x / b, a.y / b);
56     }
57     friend number dot(const point &a, const point &b) {
58         return a.x * b.x + a.y * b.y;
59     }
60     friend number det(const point &a, const point &b) {
61         return a.x * b.y - a.y * b.x;
62     }
63     friend bool operator==(const point &a, const point &b) {
64         return cmp(a.x, b.x) == 0 && cmp(a.y, b.y) == 0;
65     }
66 };
67
68 bool polar_cmp(const point &a, const point &b) {
69     // These two lines can be omitted if (0, 0) never appears
70     if (b == point()) return false;
71     if (a == point()) return true;
72     // In [0, 2*pi)
73     if (a.sgn() != b.sgn()) return a.sgn();
74     if (sgn(det(a, b)) != 0) return sgn(det(a, b)) > 0;
75     // a == b or add other condition
76     return false;
77 }
78
79 number dis2(const point &a, const point &b) {
80     return (a - b).len2();
81 }
82 number dis(const point &a, const point &b) {
83     return (a - b).len();
84 }
85
86 struct line {
87     point a, b;
88     line() {}
89     line(point _a, point _b) : a(_a), b(_b) {
90         /* for polar sort */ if (!(b - a).sgn()) std::swap(a, b);
91     }
92     point value() const {
93         return b - a;
94     }
95 };
96
97 bool polar_cmp(const line &a, const line &b) {
98     if (sgn(det(a.value(), b.value())) != 0)
99         return sgn(det(a.value(), b.value())) > 0;
100    if (sgn(det(a.value(), b.a - a.a)) != 0)
101        return sgn(det(a.value(), b.a - a.a)) > 0;
102    // a == b or add other condition
103    return false;
104 }
105
106 bool point_on_line(const point &p, const line &l) {
107     return sgn(det(p - l.a, p - l.b)) == 0;

```

```

108 }
109 // including endpoint
110 bool point_on_ray(const point &p, const line &l) {
111     return sgn(det(p - l.a, p - l.b)) == 0 &&
112         sgn(dot(p - l.a, l.b - l.a)) >= 0;
113 }
114 // including endpoints
115 bool point_on_seg(const point &p, const line &l) {
116     return sgn(det(p - l.a, p - l.b)) == 0 &&
117         sgn(dot(p - l.a, l.b - l.a)) >= 0 &&
118         sgn(dot(p - l.b, l.a - l.b)) >= 0;
119 }
120 bool seg_has_intersection(const line &a, const line &b) {
121     if (point_on_seg(a.a, b) || point_on_seg(a.b, b) ||
122         point_on_seg(b.a, a) || point_on_seg(b.b, a))
123         return /* including endpoints */ true;
124     return sgn(det(a.a - b.a, b.b - b.a)) * sgn(det(a.b - b.a, b.b - b.a)) < 0
125         && sgn(det(b.a - a.a, a.b - a.a)) * sgn(det(b.b - a.a, a.b - a.a)) < 0;
126 }
127 point intersect(const line &a, const line &b) {
128     number s1 = det(a.b - a.a, b.a - a.a);
129     number s2 = det(a.b - a.a, b.b - a.a);
130     return (b.a * s2 - b.b * s1) / (s2 - s1);
131 }
132 point projection(const point &p, const line &l) {
133     return l.a + (l.b - l.a) * dot(p - l.a, l.b - l.a) / (l.b - l.a).len2();
134 }
135 number dis(const point &p, const line &l) {
136     return std::abs(det(p - l.a, l.b - l.a)) / (l.b - l.a).len();
137 }
138 point symmetry_point(const point &a, const point &o) {
139     return o + o - a;
140 }
141 point reflection(const point &p, const line &l) {
142     return symmetry_point(p, projection(p, l));
143 }
144
145 struct circle {
146     point o;
147     number r;
148     circle() {}
149     circle(point o, number r) : o(o), r(r) {}
150
151     friend bool operator==(const circle &a, const circle &b) {
152         return a.o == b.o && cmp(a.r, b.r) == 0;
153     }
154 };
155
156 bool intersect(const line &l, const circle &a, point &p1, point &p2) {
157     number x = dot(l.a - a.o, l.b - l.a);
158     number y = (l.b - l.a).len2();
159     number d = x * x - y * ((l.a - a.o).len2() - a.r * a.r);
160     if (sgn(d) < 0) return false;
161     point p = l.a - (l.b - l.a) * (x / y), delta = (l.b - l.a) * (_sqrt(d) / y);
162     p1 = p + delta, p2 = p - delta;
163     return true;
164 }
165 bool intersect(const circle &a, const circle &b, point &p1, point &p2) {
166     if (a.o == b.o && cmp(a.r, b.r) == 0)
167         return /* value for coincident circles */ false;
168     number s1 = (b.o - a.o).len();

```

```

169     if (cmp(s1, a.r + b.r) > 0 || cmp(s1, std::abs(a.r - b.r)) < 0)
170         return false;
171     number s2 = (a.r * a.r - b.r * b.r) / s1;
172     number aa = (s1 + s2) / 2, bb = (s1 - s2) / 2;
173     point p = (b.o - a.o) * (aa / (aa + bb)) + a.o;
174     point delta = (b.o - a.o).unit().rotate90() * _sqrt(a.r * a.r - aa * aa);
175     p1 = p + delta, p2 = p - delta;
176     return true;
177 }
178 bool tangent(const point &p0, const circle &c, point &p1, point &p2) {
179     number x = (p0 - c.o).len2();
180     number d = x - c.r * c.r;
181     if (sgn(d) < 0) return false;
182     if (sgn(d) == 0)
183         return /* value for point_on_line */ false;
184     point p = (p0 - c.o) * (c.r * c.r / x);
185     point delta = ((p0 - c.o) * (-c.r * _sqrt(d) / x)).rotate90();
186     p1 = c.o + p + delta;
187     p2 = c.o + p - delta;
188     return true;
189 }
190 bool ex_tangent(const circle &a, const circle &b, line &l1, line &l2) {
191     if (cmp(std::abs(a.r - b.r), (b.o - a.o).len()) == 0) {
192         point p1, p2;
193         intersect(a, b, p1, p2);
194         l1 = l2 = line(p1, p1 + (a.o - p1).rotate90());
195         return true;
196     } else if (cmp(a.r, b.r) == 0) {
197         point dir = b.o - a.o;
198         dir = (dir * (a.r / dir.len())).rotate90();
199         l1 = line(a.o + dir, b.o + dir);
200         l2 = line(a.o - dir, b.o - dir);
201         return true;
202     } else {
203         point p = (b.o * a.r - a.o * b.r) / (a.r - b.r);
204         point p1, p2, q1, q2;
205         if (tangent(p, a, p1, p2) && tangent(p, b, q1, q2)) {
206             l1 = line(p1, q1);
207             l2 = line(p2, q2);
208             return true;
209         } else {
210             return false;
211         }
212     }
213 }
214 bool in_tangent(const circle &a, const circle &b, line &l1, line &l2) {
215     if (cmp(a.r + b.r, (b.o - a.o).len()) == 0) {
216         point p1, p2;
217         intersect(a, b, p1, p2);
218         l1 = l2 = line(p1, p1 + (a.o - p1).rotate90());
219         return true;
220     } else {
221         point p = (b.o * a.r + a.o * b.r) / (a.r + b.r);
222         point p1, p2, q1, q2;
223         if (tangent(p, a, p1, p2) && tangent(p, b, q1, q2)) {
224             l1 = line(p1, q1);
225             l2 = line(p2, q2);
226             return true;
227         } else {
228             return false;
229         }

```

```
230 }
231 }
```

3.2 平面最近点对 (Grimoire)

```
1 bool byY(P a,P b){return a.y<b.y;}
2 LL solve(P *p,int l,int r){
3     LL d=1LL<<62;
4     if(l==r)
5         return d;
6     if(l+1==r)
7         return dis2(p[l],p[r]);
8     int mid=(l+r)>>1;
9     d=min(solve(l,mid),d);
10    d=min(solve(mid+1,r),d);
11    vector<P>tmp;
12    for(int i=l;i<=r;i++)
13        if(sqrt(p[mid].x-p[i].x)<=d)
14            tmp.push_back(p[i]);
15    sort(tmp.begin(),tmp.end(),byY);
16    for(int i=0;i<tmp.size();i++)
17        for(int j=i+1;j<tmp.size()&&j-i<10;j++)
18            d=min(d,dis2(tmp[i],tmp[j]));
19    return d;
20 }
```

3.3 凸包游戏 (Grimoire)

给定凸包, $O(n \log n)$ 完成询问:

- 点在凸包内
- 凸包外的点到凸包的两个切点
- 向量关于凸包的切点
- 直线与凸包的交点

传入凸包要求 1 号点为 $\text{Pair}(x, y)$ 最小的

```
1 const int INF = 1000000000;
2 struct Convex
3 {
4     int n;
5     vector<Point> a, upper, lower;
6     Convex(vector<Point> _a) : a(_a) {
7         n = a.size();
8         int ptr = 0;
9         for(int i = 1; i < n; ++ i) if (a[ptr] < a[i]) ptr = i;
10        for(int i = 0; i <= ptr; ++ i) lower.push_back(a[i]);
11        for(int i = ptr; i < n; ++ i) upper.push_back(a[i]);
12        upper.push_back(a[0]);
13    }
14    int sign(long long x) { return x < 0 ? -1 : x > 0; }
15    pair<long long, int> get_tangent(vector<Point> &convex, Point vec) {
16        int l = 0, r = (int)convex.size() - 2;
17        for( ; l + 1 < r; ) {
18            int mid = (l + r) / 2;
19            if (sign((convex[mid + 1] - convex[mid]).det(vec)) > 0) r = mid;
```

```

20         else l = mid;
21     }
22     return max(make_pair(vec.det(convex[r]), r)
23             , make_pair(vec.det(convex[0]), 0));
24 }
25 void update_tangent(const Point &p, int id, int &i0, int &i1) {
26   if ((a[i0] - p).det(a[id] - p) > 0) i0 = id;
27   if ((a[i1] - p).det(a[id] - p) < 0) i1 = id;
28 }
29 void binary_search(int l, int r, Point p, int &i0, int &i1) {
30   if (l == r) return;
31   update_tangent(p, l % n, i0, i1);
32   int sl = sign((a[l % n] - p).det(a[(l + 1) % n] - p));
33   for( ; l + 1 < r; ) {
34     int mid = (l + r) / 2;
35     int smid = sign((a[mid % n] - p).det(a[(mid + 1) % n] - p));
36     if (smid == sl) l = mid;
37     else r = mid;
38   }
39   update_tangent(p, r % n, i0, i1);
40 }
41 int binary_search(Point u, Point v, int l, int r) {
42   int sl = sign((v - u).det(a[l % n] - u));
43   for( ; l + 1 < r; ) {
44     int mid = (l + r) / 2;
45     int smid = sign((v - u).det(a[mid % n] - u));
46     if (smid == sl) l = mid;
47     else r = mid;
48   }
49   return l % n;
50 }
51 // 判定点是否在凸包内，在边界返回 true
52 bool contain(Point p) {
53   if (p.x < lower[0].x || p.x > lower.back().x) return false;
54   int id = lower_bound(lower.begin(), lower.end()
55                       , Point(p.x, -INF)) - lower.begin();
56   if (lower[id].x == p.x) {
57     if (lower[id].y > p.y) return false;
58   } else if ((lower[id - 1] - p).det(lower[id] - p) < 0) return false;
59   id = lower_bound(upper.begin(), upper.end(), Point(p.x, INF)
60                   , greater<Point>()) - upper.begin();
61   if (upper[id].x == p.x) {
62     if (upper[id].y < p.y) return false;
63   } else if ((upper[id - 1] - p).det(upper[id] - p) < 0) return false;
64   return true;
65 }
66 // 求点 p 关于凸包的两个切点，如果在凸包外则有序返回编号
67 // 共线的多个切点返回任意一个，否则返回 false
68 bool get_tangent(Point p, int &i0, int &i1) {
69   if (contain(p)) return false;
70   i0 = i1 = 0;
71   int id = lower_bound(lower.begin(), lower.end(), p) - lower.begin();
72   binary_search(0, id, p, i0, i1);
73   binary_search(id, (int)lower.size(), p, i0, i1);
74   id = lower_bound(upper.begin(), upper.end(), p
75                   , greater<Point>()) - upper.begin();
76   binary_search((int)lower.size() - 1, (int)lower.size() - 1 + id, p, i0, i1);
77   binary_search((int)lower.size() - 1 + id
78               , (int)lower.size() - 1 + (int)upper.size(), p, i0, i1);
79   return true;
80 }

```

```

81 // 求凸包上和向量 vec 叉积最大的点, 返回编号, 共线的多个切点返回任意一个
82 int get_tangent(Point vec) {
83     pair<long long, int> ret = get_tangent(upper, vec);
84     ret.second = (ret.second + (int)lower.size() - 1) % n;
85     ret = max(ret, get_tangent(lower, vec));
86     return ret.second;
87 }
88 // 求凸包和直线 u,v 的交点, 如果无严格相交返回 false.
89 //如果有则是和 (i,next(i)) 的交点, 两个点无序, 交在点上不确定返回前后两条线段其中之一
90 bool get_intersection(Point u, Point v, int &i0, int &i1) {
91     int p0 = get_tangent(u - v), p1 = get_tangent(v - u);
92     if (sign((v - u).det(a[p0] - u)) * sign((v - u).det(a[p1] - u)) < 0) {
93         if (p0 > p1) swap(p0, p1);
94         i0 = binary_search(u, v, p0, p1);
95         i1 = binary_search(u, v, p1, p0 + n);
96         return true;
97     } else {
98         return false;
99     }
100 }
101 };

```

3.4 半平面交 (Grimoire)

如果需要考虑是否形成无限大区域的情况, 可添加四条新边作为边界, 并通过新边是否在半平面交上来判断原问题是否形成无限大区域

```

1 bool same_dir(const point &a, const point &b) {
2     return sgn(det(a.v(), b.v())) == 0 && sgn(dot(a.v(), b.v())) >= 0;
3 }
4
5 std::deque<line> half_plane_intersection(std::vector<line> &l) {
6     std::sort(l.begin(), l.end(), [] (const line &a, const line &b) {
7         if (same_dir(a, b))
8             return sgn(det(a.v(), b.a - a.a)) < 0;
9         else
10            return polar_cmp(a.v(), b.v());
11    });
12    std::deque<line> q;
13    for (int i = 0; i < int(l.size()); l++) {
14        if (i && same_dir(l[i - 1], l[i])) continue;
15        while (q.size() > 1 && check(q[q.size() - 2], q.back(), l[i]))
16            q.pop_back();
17        while (q.size() > 1 && check(q[1], q.front(), l[i]))
18            q.pop_front();
19        q.push_back(l[i]);
20    }
21    while (q.size() > 2 && !check(q[q.size() - 2], q.back(), q.front()))
22        q.pop_back();
23    while (q.size() > 2 && !check(q[1], q.front(), q.back()))
24        q.pop_front();
25    // if q.size() <= 2, no solution
26    if (q.size() == 2
27        && sgn(det(q.front().v(), q.back().v())) == 0
28        && sgn(dot(q.front().v(), q.back().v())) < 0
29        && sgn(det(q.front().v(), q.back().a - q.front().a)) < 0)
30        q.clear();
31    return q;
32 }

```

3.5 点在多边形内 (Grimoire)

```

1 bool inPoly(P p, vector<P>poly){
2     int cnt=0;
3     for(int i=0;i<poly.size();i++){
4         P a=poly[i],b=poly[(i+1)%poly.size()];
5         if(onSeg(p,L(a,b)))
6             return false;
7         int x=sgn(det(a,p,b));
8         int y=sgn(a.y-p.y);
9         int z=sgn(b.y-p.y);
10        cnt+=(x>0&&y<=0&&z>0);
11        cnt-=(x<0&&y<=0&&z>0);
12    }
13    return cnt;
14 }
```

3.6 最小圆覆盖 (Grimoire)

```

1 struct line{
2     point p,v;
3 };
4 point Rev(point v){return point(-v.y,v.x);}
5 point operator*(line A,line B){
6     point u=B.p-A.p;
7     double t=(B.v*u)/(B.v*A.v);
8     return A.p+A.v*t;
9 }
10 point get(point a,point b){
11     return (a+b)/2;
12 }
13 point get(point a,point b,point c){
14     if(a==b) return get(a,c);
15     if(a==c) return get(a,b);
16     if(b==c) return get(a,b);
17     line ABO=(line){(a+b)/2,Rev(a-b)};
18     line BCO=(line){(c+b)/2,Rev(b-c)};
19     return ABO*BCO;
20 }
21 int main(){
22     scanf("%d",&n);
23     for(int i=1;i<=n;i++)scanf("%lf%lf",&p[i].x,&p[i].y);
24     random_shuffle(p+1,p+1+n);
25     O=p[1];r=0;
26     for(int i=2;i<=n;i++){
27         if(dis(p[i],O)<r+1e-6)continue;
28         O=get(p[1],p[i]);r=dis(O,p[i]);
29         for(int j=1;j<i;j++){
30             if(dis(p[j],O)<r+1e-6)continue;
31             O=get(p[i],p[j]);r=dis(O,p[i]);
32             for(int k=1;k<j;k++){
33                 if(dis(p[k],O)<r+1e-6)continue;
34                 O=get(p[i],p[j],p[k]);r=dis(O,p[i]);
35             }
36         }
37     }printf("%.2lf %.2lf %.2lf\n",O.x,O.y,r);
38     return 0;
39 }
```

3.7 最小球覆盖 (Grimoire)

```

1 bool equal(const double & x, const double & y) {
2     return x + eps > y and y + eps > x;
3 }
4 double operator % (const Point & a, const Point & b) {
5     return a.x * b.x + a.y * b.y + a.z * b.z;
6 }
7 Point operator * (const Point & a, const Point & b) {
8     return Point(a.y * b.z - a.z * b.y, a.z * b.x - a.x * b.z, a.x * b.y - a.y * b.x);
9 }
10 struct Circle {
11     double r; Point o;
12 };
13 struct Plane {
14     Point nor;
15     double m;
16     Plane(const Point & nor, const Point & a) : nor(nor){
17         m = nor % a;
18     }
19 };
20 Point intersect(const Plane & a, const Plane & b, const Plane & c) {
21     Point c1(a.nor.x, b.nor.x, c.nor.x), c2(a.nor.y, b.nor.y, c.nor.y), c3(a.nor.z, b.nor.z, c.nor.z),
22     ↪ c4(a.m, b.m, c.m);
23     return 1 / ((c1 * c2) % c3) * Point((c4 * c2) % c3, (c1 * c4) % c3, (c1 * c2) % c4);
24 }
25 bool in(const Point & a, const Circle & b) {
26     return sign((a - b.o).len() - b.r) <= 0;
27 }
28 bool operator < (const Point & a, const Point & b) {
29     if(!equal(a.x, b.x)) {
30         return a.x < b.x;
31     }
32     if(!equal(a.y, b.y)) {
33         return a.y < b.y;
34     }
35     if(!equal(a.z, b.z)) {
36         return a.z < b.z;
37     }
38     return false;
39 }
40 bool operator == (const Point & a, const Point & b) {
41     return equal(a.x, b.x) and equal(a.y, b.y) and equal(a.z, b.z);
42 }
43 vector<Point> vec;
44 Circle calc() {
45     if(vec.empty()) {
46         return Circle(Point(0, 0, 0), 0);
47     } else if(1 == (int)vec.size()) {
48         return Circle(vec[0], 0);
49     } else if(2 == (int)vec.size()) {
50         return Circle(0.5 * (vec[0] + vec[1]), 0.5 * (vec[0] - vec[1]).len());
51     } else if(3 == (int)vec.size()) {
52         double r((vec[0] - vec[1]).len() * (vec[1] - vec[2]).len() * (vec[2] - vec[0]).len() / 2 /
53         ↪ fabs((vec[0] - vec[2]) * (vec[1] - vec[2]).len()));
54         return Circle(intersect(Plane(vec[1] - vec[0], 0.5 * (vec[1] + vec[0])),,
55             Plane(vec[2] - vec[1], 0.5 * (vec[2] + vec[1])),,
56             Plane((vec[1] - vec[0]) * (vec[2] - vec[0]), vec[0])), r);
57     } else {
58         Point o(intersect(Plane(vec[1] - vec[0], 0.5 * (vec[1] + vec[0])),,

```

```

57         Plane(vec[2] - vec[0], 0.5 * (vec[2] + vec[0])),  

58         Plane(vec[3] - vec[0], 0.5 * (vec[3] + vec[0]))));  

59     return Circle(o, (o - vec[0]).len());  

60 }  

61 }  

62 Circle miniBall(int n) {  

63     Circle res(calc());  

64     for(int i=0; i < n; i++) {  

65         if(!in(a[i], res)) {  

66             vec.push_back(a[i]);  

67             res = miniBall(i);  

68             vec.pop_back();  

69             if(i) {  

70                 Point tmp(a[i]);  

71                 memmove(a + 1, a, sizeof(Point) * i);  

72                 a[0] = tmp;  

73             }  

74         }  

75     }  

76     return res;  

77 }  

78 int main() {  

79     int n;  

80     sort(a, a + n);  

81     n = unique(a, a + n) - a;  

82     vec.clear();  

83     printf("%.10f\n", miniBall(n).r);  

84 }

```

3.8 圆并 (gy)

```

1 int n;
2 circle a[N];
3 number ans[N];
4 std::pair<number, int> op[N << 1];
5
6 number circle_integral(const circle &x, double s, double t) {
7     return x.r * (x.r * (t - s)
8                 - x.o.y * (std::cos(t) - std::cos(s))
9                 + x.o.x * (std::sin(t) - std::sin(s)));
10 }
11
12 void circle_union() {
13     // remove circle with r = 0
14     for (int i = 1; i <= n; i++) if (sgn(a[i].r) <= 0) std::swap(a[i], a[n--]);
15     // remove coincident circle
16     std::sort(a + 1, a + n + 1, [] (const circle &a, const circle &b) {
17         if (sgn(a.o.x) != sgn(b.o.x)) return a.o.x < b.o.x;
18         if (sgn(a.o.y) != sgn(b.o.y)) return a.o.y < b.o.y;
19         return cmp(a.r, b.r) < 0;
20     });
21     n = int(std::unique(a + 1, a + n + 1) - a - 1);
22     for (int i = 1; i <= n; i++) {
23         int cnt = 0, opcnt = 0;
24         for (int j = 1; j <= n; j++) if (i != j) {
25             point delta = a[j].o - a[i].o;
26             number d = delta.len();
27             if (cmp(d, a[i].r + a[j].r) >= 0) continue;
28             if (cmp(d + a[j].r, a[i].r) <= 0) continue;

```

```

29     if (cmp(d + a[i].r, a[j].r) <= 0) {
30         // if only k = 1 needed, just continue@i
31         cnt++;
32         continue;
33     }
34     number t0 = delta.angle();
35     number t1 = _acos((a[i].r * a[i].r + d * d - a[j].r * a[j].r) / (2 * a[i].r * d));
36     number l = t0 - t1, r = t0 + t1;
37     if (l < -pi) l += 2 * pi, cnt++;
38     if (r > pi) r -= 2 * pi, cnt++;
39     op[++opcnt] = {l, 1};
40     op[++opcnt] = {r, -1};
41 }
42 op[0] = {-pi, 0}, op[++opcnt] = {pi, 0};
43 std::sort(op + 1, op + opcnt);
44 for (int j = 1; j <= opcnt; cnt += op[j++].second)
45     ans[cnt + 1] += circle_integral(a[i], op[j - 1].first, op[j].first);
46 }
47 for (int i = 1; i <= n; i++) ans[i] /= 2;
48 }
```

3.9 圆与多边形并 (Grimoire)

```

1 double form(double x){
2     while(x>=2*pi)x-=2*pi;
3     while(x<0)x+=2*pi;
4     return x;
5 }
6 double calcCir(C cir){
7     vector<double>ang;
8     ang.push_back(0);
9     ang.push_back(pi);
10    double ans=0;
11    for(int i=1;i<=n;i++){
12        if(cir==c[i])continue;
13        P p1,p2;
14        if(intersect(cir,c[i],p1,p2)){
15            ang.push_back(form(cir.ang(p1)));
16            ang.push_back(form(cir.ang(p2)));
17        }
18    }
19
20    for(int i=1;i<=m;i++){
21        vector<P>tmp;
22        tmp=intersect(poly[i],cir);
23        for(int j=0;j<tmp.size();j++){
24            ang.push_back(form(cir.ang(tmp[j])));
25        }
26    }
27    sort(ang.begin(),ang.end());
28    for(int i=0;i<ang.size();i++){
29        double t1=ang[i],t2=(i+1==ang.size())?ang[0]+2*pi:ang[i+1];
30        P p=cir.at((t1+t2)/2);
31        int ok=1;
32        for(int j=1;j<=n;j++){
33            if(cir==c[j])continue;
34            if(inC(p,c[j],true)){
35                ok=0;
36                break;
```

```

37         }
38     }
39     for(int j=1;j<=m&&ok;j++){
40         if(inPoly(p,poly[j],true)){
41             ok=0;
42             break;
43         }
44     }
45     if(ok){
46         double r=cir.r,x0=cir.o.x,y0=cir.o.y;
47         ans+=(r*r*(t2-t1)+r*x0*(sin(t2)-sin(t1))-r*y0*(cos(t2)-cos(t1)))/2;
48     }
49 }
50 return ans;
51 }
52 P st;
53 bool bySt(P a,P b){
54     return dis(a,st)<dis(b,st);
55 }
56 double calcSeg(L l){
57     double ans=0;
58     vector<P>pt;
59     pt.push_back(l.a);
60     pt.push_back(l.b);
61     for(int i=1;i<=n;i++){
62         P p1,p2;
63         if(intersect(c[i],l,p1,p2)){
64             if(onSeg(p1,l))
65                 pt.push_back(p1);
66             if(onSeg(p2,l))
67                 pt.push_back(p2);
68         }
69     }
70     st=l.a;
71     sort(pt.begin(),pt.end(),bySt);
72     for(int i=0;i+1<pt.size();i++){
73         P p1=pt[i],p2=pt[i+1];
74         P p=(p1+p2)/2;
75         int ok=1;
76         for(int j=1;j<=n;j++){
77             if(sgn(dis(p,c[j].o),c[j].r)<0){
78                 ok=0;
79                 break;
80             }
81         }
82     }
83     if(ok){
84         double x1=p1.x,y1=p1.y,x2=p2.x,y2=p2.y;
85         double res=(x1*y2-x2*y1)/2;
86         ans+=res;
87     }
88 }
89 return ans;
90 }
```

3.10 三角剖分 (Grimoire)

Triangulation::find 返回包含某点的三角形
 Triangulation::add_point 将某点加入三角剖分

某个 *Triangle* 在三角剖分中当且仅当它的 *has_children* 为 0

如果要找到三角形 u 的邻域，则枚举它的所有 u.edge[i].tri，该条边的两个点为 u.p[(i + 1) % 3], u.p[(i + 2) % 3]

通过三角剖分构造 V 图：连接相邻三角形外接圆圆心

注意初始化内存池和 *Triangulation* :: *LOTS*

复杂度 $O(n \log n)$

```

1 const int N = 100000 + 5, MAX_TRIS = N * 6;
2 const double eps = 1e-6, PI = acos(-1.0);
3 struct P {
4     double x,y; P():x(0),y(0){}
5     P(double x, double y):x(x),y(y){}
6     bool operator ==(P const& that) const {return x==that.x&&y==that.y;}
7 };
8 inline double sqr(double x) { return x*x; }
9 double dist_sqr(P const& a, P const& b){return sqr(a.x-b.x)+sqr(a.y-b.y);}
10 bool in_circumcircle(P const& p1, P const& p2, P const& p3, P const& p4) { //p4 in C(p1,p2,p3)
11     double u11 = p1.x - p4.x, u21 = p2.x - p4.x, u31 = p3.x - p4.x;
12     double u12 = p1.y - p4.y, u22 = p2.y - p4.y, u32 = p3.y - p4.y;
13     double u13 = sqr(p1.x) - sqr(p4.x) + sqr(p1.y) - sqr(p4.y);
14     double u23 = sqr(p2.x) - sqr(p4.x) + sqr(p2.y) - sqr(p4.y);
15     double u33 = sqr(p3.x) - sqr(p4.x) + sqr(p3.y) - sqr(p4.y);
16     double det = -u13*u22*u31 + u12*u23*u31 + u13*u21*u32 - u11*u23*u32 - u12*u21*u33 + u11*u22*u33;
17     return det > eps;
18 }
19 double side(P const& a, P const& b, P const& p) { return (b.x-a.x)*(p.y-a.y) - (b.y-a.y)*(p.x-a.x); }
20 typedef int SideRef; struct Triangle; typedef Triangle* TriangleRef;
21 struct Edge {
22     TriangleRef tri; SideRef side; Edge() : tri(0), side(0) {}
23     Edge(TriangleRef tri, SideRef side) : tri(tri), side(side) {}
24 };
25 struct Triangle {
26     P p[3]; Edge edge[3]; TriangleRef children[3]; Triangle() {}
27     Triangle(P const& p0, P const& p1, P const& p2) {
28         p[0] = p0; p[1] = p1; p[2] = p2;
29         children[0] = children[1] = children[2] = 0;
30     }
31     bool has_children() const { return children[0] != 0; }
32     int num_children() const {
33         return children[0] == 0 ? 0
34             : children[1] == 0 ? 1
35             : children[2] == 0 ? 2 : 3;
36     }
37     bool contains(P const& q) const {
38         double a=side(p[0],p[1],q), b=side(p[1],p[2],q), c=side(p[2],p[0],q);
39         return a >= -eps && b >= -eps && c >= -eps;
40     }
41 } triangle_pool[MAX_TRIS], *tot_triangles;
42 void set_edge(Edge a, Edge b) {
43     if (a.tri) a.tri->edge[a.side] = b;
44     if (b.tri) b.tri->edge[b.side] = a;
45 }
46 class Triangulation {
47 public:
48     Triangulation() {
49         const double LOTS = 1e6; //初始为极大三角形
50         the_root = new(tot_triangles++) Triangle(P(-LOTS,-LOTS),P(+LOTS,-LOTS),P(0,+LOTS));
51     }
52     TriangleRef find(P p) const { return find(the_root,p); }
53     void add_point(P const& p) { add_point(find(the_root,p),p); }
54 }
```

```

55     TriangleRef the_root;
56     static TriangleRef find(TriangleRef root, P const& p) {
57         for( ; ; ) {
58             if (!root->has_children()) return root;
59             else for (int i = 0; i < 3 && root->children[i] ; ++i)
60                 if (root->children[i]->contains(p))
61                     {root = root->children[i]; break;}
62         }
63     }
64     void add_point(TriangleRef root, P const& p) {
65         TriangleRef tab,tbc,tca;
66         tab = new(tot_triangles++) Triangle(root->p[0], root->p[1], p);
67         tbc = new(tot_triangles++) Triangle(root->p[1], root->p[2], p);
68         tca = new(tot_triangles++) Triangle(root->p[2], root->p[0], p);
69         set_edge(Edge(tab,0), Edge(tbc,1)); set_edge(Edge(tbc,0), Edge(tca,1));
70         set_edge(Edge(tca,0), Edge(tab,1)); set_edge(Edge(tab,2), root->edge[2]);
71         set_edge(Edge(tbc,2), root->edge[0]); set_edge(Edge(tca,2), root->edge[1]);
72         root->children[0]=tab; root->children[1]=tbc; root->children[2]=tca;
73         flip(tab,2); flip(tbc,2); flip(tca,2);
74     }
75     void flip(TriangleRef tri, SideRef pi) {
76         TriangleRef trj = tri->edge[pi].tri; int pj = tri->edge[pi].side;
77         if(!trj || !in_circumcircle(tri->p[0],tri->p[1],tri->p[2],trj->p[pj])) return;
78         TriangleRef trk = new(tot_triangles++) Triangle(tri->p[(pi+1)%3], trj->p[pj], tri->p[pi]);
79         TriangleRef trl = new(tot_triangles++) Triangle(trj->p[(pj+1)%3], tri->p[pi], trj->p[pj]);
80         set_edge(Edge(trk,0), Edge(trl,0));
81         set_edge(Edge(trk,1), tri->edge[(pi+2)%3]); set_edge(Edge(trk,2), trj->edge[(pj+1)%3]);
82         set_edge(Edge(trl,1), trj->edge[(pj+2)%3]); set_edge(Edge(trl,2), tri->edge[(pi+1)%3]);
83         tri->children[0]=trk; tri->children[1]=trl; tri->children[2]=0;
84         trj->children[0]=trk; trj->children[1]=trl; trj->children[2]=0;
85         flip(trk,1); flip(trk,2); flip(trl,1); flip(trl,2);
86     }
87 };
88 int n; P ps[N];
89 void build(){
90     tot_triangles = triange_pool; cin >> n;
91     for(int i = 0; i < n; ++ i) scanf("%lf%lf",&ps[i].x,&ps[i].y);
92     random_shuffle(ps, ps + n); Triangulation tri;
93     for(int i = 0; i < n; ++ i) tri.add_point(ps[i]);
94 }
```

3.11 三维几何基础 (Grimoire)

```

1 struct P {
2     double x, y, z;
3     P(){}
4     P(double _x,double _y,double _z):x(_x),y(_y),z(_z){}
5     double len2(){
6         return (x*x+y*y+z*z);
7     }
8     double len(){
9         return sqrt(x*x+y*y+z*z);
10    }
11 };
12 bool operator==(P a,P b){
13     return sgn(a.x-b.x)==0 && sgn(a.y-b.y)==0 && sgn(a.z-b.z)==0 ;
14 }
15 bool operator<(P a,P b){
16     return sgn(a.x-b.x) ? a.x<b.x :(sgn(a.y-b.y)?a.y<b.y :a.z<b.z);

```

```

17 }
18 P operator+(P a,P b){
19     return P(a.x+b.x,a.y+b.y,a.z+b.z);
20 }
21 P operator-(P a,P b){
22     return P(a.x-b.x,a.y-b.y,a.z-b.z);
23 }
24 P operator*(P a,double b){
25     return P(a.x*b,a.y*b,a.z*b);
26 }
27 P operator/(P a,double b){
28     return P(a.x/b,a.y/b,a.z/b);
29 }
30 P operator*(const P &a, const P &b) {
31     return P(a.y * b.z - a.z * b.y, a.z * b.x - a.x * b.z, a.x * b.y - a.y * b.x);
32 }
33 double operator^(const P &a, const P &b) {
34     return a.x*b.x+a.y*b.y+a.z*b.z;
35 }
36
37 double dis(P a,P b){return (b-a).len();}
38 double dis2(P a,P b){return (b-a).len2();}
39
40 // 3D line intersect
41 P intersect(const P &a0, const P &b0, const P &a1, const P &b1) {
42     double t = ((a0.x - a1.x) * (a1.y - b1.y) - (a0.y - a1.y) * (a1.x - b1.x)) / ((a0.x - b0.x) * (a1.y
43     ↵ - b1.y) - (a0.y - b0.y) * (a1.x - b1.x));
44     return a0 + (b0 - a0) * t;
45 }
46 // area-line intersect
47 P intersect(const P &a, const P &b, const P &c, const P &l0, const P &l1) {
48
49     P p = (b-a)*(c-a); // 平面法向量
50     double t = (p^(a-l0)) / (p^(l1-l0));
51     return l0 + (l1 - l0) * t;
52 }

```

3.12 三维凸包 (Grimoire)

```

1 int mark[1005][1005],n, cnt;;
2 double mix(const P &a, const P &b, const P &c) {
3     return a^(b*c);
4 }
5 double area(int a, int b, int c) {
6     return ((info[b] - info[a])*(info[c] - info[a])).len();
7 }
8 double volume(int a, int b, int c, int d) {
9     return mix(info[b] - info[a], info[c] - info[a], info[d] - info[a]);
10 }
11 struct Face {
12     int a, b, c; Face() {}
13     Face(int a, int b, int c): a(a), b(b), c(c) {}
14     int &operator [](int k) {
15         if (k == 0) return a; if (k == 1) return b; return c;
16     }
17 };
18 vector<Face> face;
19 inline void insert(int a, int b, int c) {
20     face.push_back(Face(a, b, c));

```

```

21 }
22 void add(int v) {
23     vector<Face> tmp; int a, b, c; cnt++;
24     for (int i = 0; i < SIZE(face); i++) {
25         a = face[i][0]; b = face[i][1]; c = face[i][2];
26         if (sgn(volume(v, a, b, c)) < 0)
27             mark[a][b] = mark[b][a] = mark[b][c] = mark[c][b] = mark[c][a] = mark[a][c] = cnt;
28         else tmp.push_back(face[i]);
29     } face = tmp;
30     for (int i = 0; i < SIZE(tmp); i++) {
31         a = face[i][0]; b = face[i][1]; c = face[i][2];
32         if (mark[a][b] == cnt) insert(b, a, v);
33         if (mark[b][c] == cnt) insert(c, b, v);
34         if (mark[c][a] == cnt) insert(a, c, v);
35     }
36 }
37 int Find() {
38     for (int i = 2; i < n; i++) {
39         P ndir = (info[0] - info[i])*(info[1] - info[i]);
40         if (ndir == P()) continue; swap(info[i], info[2]);
41         for (int j = i + 1; j < n; j++) if (sgn(volume(0, 1, 2, j)) != 0) {
42             swap(info[j], info[3]); insert(0, 1, 2); insert(0, 2, 1); return 1;
43         }
44     }
45     return 0;
46 }
47 //find the weight center
48 double calcDist(const P &p, int a, int b, int c) {
49     return fabs(mix(info[a] - p, info[b] - p, info[c] - p) / area(a, b, c));
50 }
51 //compute the minimal distance of center of any faces
52 P findCenter() { //compute center of mass
53     double totalWeight = 0;
54     P center(.0, .0, .0);
55     P first = info[face[0][0]];
56     for (int i = 0; i < SIZE(face); ++i) {
57         P p = (info[face[i][0]]+info[face[i][1]]+info[face[i][2]]+first)*.25;
58         double weight = mix(info[face[i][0]] - first, info[face[i][1]] - first, info[face[i][2]] -
59             first);
60         totalWeight += weight; center = center + p * weight;
61     }
62     center = center / totalWeight;
63     return center;
64 }
65 double minDis(P p) {
66     double res = 1e100; //compute distance
67     for (int i = 0; i < SIZE(face); ++i)
68         res = min(res, calcDist(p, face[i][0], face[i][1], face[i][2]));
69     return res;
70 }
71 void findConvex(P *info, int n) {
72     sort(info, info + n); n = unique(info, info + n) - info;
73     face.clear(); random_shuffle(info, info + n);
74     if (!Find()) return abort();
75     memset(mark, 0, sizeof(mark)); cnt = 0;
76     for (int i = 3; i < n; i++) add(i);
77 }

```

3.13 三维绕轴旋转 (gy)

右手大拇指指向 $axis$ 方向, 四指弯曲方向旋转 w 弧度

```

1 P rotate(const P& s, const P& axis, double w) {
2     double x = axis.x, y = axis.y, z = axis.z;
3     double s1 = x * x + y * y + z * z, ss1 = msqrt(s1),
4         cosw = cos(w), sinw = sin(w);
5     double a[4][4];
6     memset(a, 0, sizeof a);
7     a[3][3] = 1;
8     a[0][0] = ((y * y + z * z) * cosw + x * x) / s1;
9     a[0][1] = x * y * (1 - cosw) / s1 + z * sinw / ss1;
10    a[0][2] = x * z * (1 - cosw) / s1 - y * sinw / ss1;
11    a[1][0] = x * y * (1 - cosw) / s1 - z * sinw / ss1;
12    a[1][1] = ((x * x + z * z) * cosw + y * y) / s1;
13    a[1][2] = y * z * (1 - cosw) / s1 + x * sinw / ss1;
14    a[2][0] = x * z * (1 - cosw) / s1 + y * sinw / ss1;
15    a[2][1] = y * z * (1 - cosw) / s1 - x * sinw / ss1;
16    a[2][2] = ((x * x + y * y) * cos(w) + z * z) / s1;
17    double ans[4] = {0, 0, 0, 0}, c[4] = {s.x, s.y, s.z, 1};
18    for (int i = 0; i < 4; ++ i)
19        for (int j = 0; j < 4; ++ j)
20            ans[i] += a[j][i] * c[j];
21    return P(ans[0], ans[1], ans[2]);
22 }
```

3.14 几何知识 (gy)

Pick theorem

顶点为整点的简单多边形, 其面积 A , 内部格点数 i , 边上格点数 b 满足:

$$A = i + \frac{b}{2} - 1$$

欧拉示性数

- 三维凸包的顶点个数 V , 边数 E , 面数 F 满足:

$$V - E + F = 2$$

- 平面图的顶点个数 V , 边数 E , 平面被划分的区域数 F , 组成图形的连通部分的数目 C 满足:

$$V - E + F = C + 1$$

闵可夫斯基和

A, B 的闵可夫斯基和定义为 $C = \{a + b \mid a \in A, b \in B\}$

闵可夫斯基和的边是由两凸包构成的, 把两凸包的边极角排序后顺次连接起来就是闵可夫斯基和。需要注意的是可能会有三点共线的情况, 因此需要重新求一次凸包

最小乘积生成树

找到 $\sum_i x_i \cdot \sum_i y_i$ 最小的的生成树

首先找出两个在凸包上的点 $A(\min_x, y), B(x, \min_y)$, 在直线 AB 下方找一个在凸包上且 $x \cdot y$ 最小的点。可以每次找距离直线 AB 最远的点, 记每条边的 $y_i = y_i \cdot (x_B, x_A)$, $x_i = x_i \cdot (y_A, y_B)$, 以 $x_i + y_i$ 为关键字排序做 Kruskal。递归处理直到叉积 ≥ 0

几何公式

• 三角形

$$\text{半周长 } p = \frac{a+b+c}{2}$$

$$\text{面积 } S = \frac{1}{2}aH_a = \frac{1}{2}ab \cdot \sin C = \sqrt{p(p-a)(p-b)(p-c)} = pr = \frac{abc}{4R}$$

$$\text{中线长 } M_a = \frac{1}{2}\sqrt{2(b^2 + c^2) - a^2} = \frac{1}{2}\sqrt{b^2 + c^2 + 2bc \cdot \cos A}$$

$$\text{角平分线长 } T_a = \frac{\sqrt{bc((b+c)^2 - a^2)}}{b+c} = \frac{2bc}{b+c} \cos \frac{A}{2}$$

$$\text{高 } H_a = b \sin C = \sqrt{b^2 - (\frac{a^2 + b^2 - c^2}{2a})^2}$$

$$\text{内切圆半径 } r = \frac{S}{p} = 4R \sin \frac{A}{2} \sin \frac{B}{2} \sin \frac{C}{2} = \sqrt{\frac{(p-a)(p-b)(p-c)}{p}} = p \tan \frac{A}{2} \tan \frac{B}{2} \tan \frac{C}{2}$$

$$\text{外接圆半径 } R = \frac{abc}{4S} = \frac{a}{2 \sin A}$$

$$\text{旁切圆半径 } r_A = \frac{2S}{-a+b+c}$$

$$\text{重心 } (\frac{x_1 + x_2 + x_3}{3}, \frac{y_1 + y_2 + y_3}{3})$$

$$\text{外心 } (\left| \begin{array}{ccc} x_1^2 + y_1^2 & y_1 & 1 \\ x_2^2 + y_2^2 & y_2 & 1 \\ x_3^2 + y_3^2 & y_3 & 1 \end{array} \right|, \left| \begin{array}{ccc} x_1 & x_1^2 + y_1^2 & 1 \\ x_2 & x_2^2 + y_2^2 & 1 \\ x_3 & x_3^2 + y_3^2 & 1 \end{array} \right|)$$

$$\text{内心 } (\frac{ax_1 + bx_2 + cx_3}{a+b+c}, \frac{ay_1 + by_2 + cy_3}{a+b+c})$$

$$\text{垂心 } (\left| \begin{array}{ccc} x_1 & y_1 & 1 \\ x_2 & y_2 & 1 \\ x_3 & y_3 & 1 \end{array} \right|, \left| \begin{array}{ccc} x_1 & y_1 & 1 \\ x_2 & y_2 & 1 \\ x_3 & y_3 & 1 \end{array} \right|)$$

$$\text{旁心 } (\frac{-ax_1 + bx_2 + cx_3}{-a+b+c}, \frac{-ay_1 + by_2 + cy_3}{-a+b+c})$$

$$\text{Trilinear coordinates: } \frac{ax}{ax+by+cz}A + \frac{by}{ax+by+cz}B + \frac{cz}{ax+by+cz}C$$

x, y, z 分别代表点 P 到边的距离

$$\text{Fermat point: } x : y : z = \csc(A + \frac{\pi}{3}) : \csc(B + \frac{\pi}{3}) : \csc(C + \frac{\pi}{3})$$

• 圆

$$\text{弧长 } l = rA$$

$$\text{弦长 } a = 2\sqrt{2hr - h^2} = 2r \cdot \sin \frac{A}{2}$$

$$\text{弓形高 } h = r - \sqrt{r^2 - \frac{a^2}{4}} = r(1 - \cos \frac{A}{2})$$

$$\text{扇形面积 } S_1 = \frac{1}{2}lr = \frac{1}{2}Ar^2$$

$$\text{弓形面积 } S_2 = \frac{1}{2}r^2(A - \sin A)$$

• Circles of Apollonius

已知三个两两相切的圆，半径为 r_1, r_2, r_3

$$\text{与它们外切的圆半径为 } \left| \frac{r_1 r_2 r_3}{r_1 r_2 + r_2 r_3 + r_3 r_1 - 2\sqrt{r_1 r_2 r_3(r_1 + r_2 + r_3)}} \right|$$

$$\text{与它们内切的圆半径为 } \left| \frac{r_1 r_2 r_3}{r_1 r_2 + r_2 r_3 + r_3 r_1 + 2\sqrt{r_1 r_2 r_3(r_1 + r_2 + r_3)}} \right|$$

• 棱台

$$\text{体积 } V = \frac{1}{3}h(A_1 + A_2 + \sqrt{A_1 A_2})$$

$$\text{正棱台侧面积 } S = \frac{1}{2}(p_1 + p_2)l, l \text{ 为侧高}$$

• 球

$$\text{体积 } V = \frac{4}{3}\pi r^3$$

$$\text{表面积 } S = 4\pi r^2$$

• 球台

$$\text{侧面积 } S = 2\pi rh$$

$$\text{体积 } V = \frac{1}{6}\pi h(3(r_1^2 + r_2^2) + h_h)$$

• 球扇形

$$\text{球面面积 } S = 2\pi rh$$

$$\text{体积 } V = \frac{2}{3}\pi r^2 h = \frac{2}{3}\pi r^3 h(1 - \cos \varphi)$$

• 球面三角形

考虑单位球上的球面三角形, a, b, c 表示三边长 (弧所对球心角), A, B, C 表示三角大小 (切线夹角)

$$\text{余弦定理 } \cos a = \cos b \cdot \cos c + \sin a \cdot \sin b \cdot \cos A$$

$$\text{正弦定理 } \frac{\sin A}{\sin a} = \frac{\sin B}{\sin b} = \frac{\sin C}{\sin c}$$

$$\text{球面面积 } S = A + B + C - \pi$$

• 四面体

$$\text{体积 } V = \frac{1}{6} |\overrightarrow{AB} \cdot (\overrightarrow{AC} \times \overrightarrow{AD})|$$

String

4.1 KMP (ct)

KMP

```

1 int main()
2 {
3     for (int i = 2, j = 0; i <= n; ++i)
4     {
5         for ( ; j && s[j + 1] != s[i]; j = fail[j]);
6         s[i] == s[j + 1] ? ++j : 0;
7         fail[i] = j;
8     }
9     return 0;
10 }
```

exKMP

$$a_i = \text{lcp}(s + i, t), b_i = \text{lcp}(t + i, t)$$

```

1 void exkmp(char *s, int *a, int n) {
2     a[0] = n;
3     int p = 0, r = 0;
4     for (int i = 1; i < n; ++i) {
5         a[i] = (r > i) ? std::min(r - i, a[i - p]) : 0;
6         while (i + a[i] < n && s[i + a[i]] == s[a[i]]) ++a[i];
7         if (r < i + a[i]) r = i + a[i], p = i;
8     }
9 }
10 void mat(char *s, char *t, int *a, int *b, int n, int m) {
11     exkmp(t, b, m);
12     int p = 0, r = 0;
13     for (int i = 0; i < n; ++i) {
14         a[i] = (r > i) ? std::min(r - i, b[i - p]) : 0;
15         while (i + a[i] < n && a[i] < m && s[i + a[i]] == t[a[i]]) ++a[i];
16         if (r < i + a[i]) r = i + a[i], p = i;
17     }
18 }
```

4.2 AC 自动机 (ct)

```

1 struct Trie {
2     Trie *next[26], *fail;
3     int end;
4 } mem[maxn * maxl], *tot = mem, *q[maxn * maxl];
5 char s[maxl];
6 inline void insert(int v)
```

```

7  {
8      Trie *now = mem; int n = strlen(s + 1);
9      for (int i = 1; i <= n; ++i)
10     {
11         int v = s[i] - 'a';
12         if (!now -> next[v])
13         {
14             now -> next[v] = ++tot;
15             for (int i = 0; i < 26; ++i) tot -> next[i] = 0;
16             tot -> fail = 0;
17             tot -> end = 0;
18         }
19         now = now -> next[v];
20     }
21     now -> end |= v;
22 }
23 inline void build()
24 {
25     int head = 0, tail = 0;
26     for (int i = 0; i < 26; ++i)
27         if (mem -> next[i]) (q[++tail] = mem -> next[i]) -> fail = mem;
28         else mem -> next[i] = mem;
29     while (head < tail)
30     {
31         Trie *now = q[++head];
32         now -> end |= now -> fail -> end;
33         for (int i = 0; i < 26; ++i)
34             if (!now -> next[i])
35                 now -> next[i] = now -> fail -> next[i];
36             else
37                 (q[++tail] = now -> next[i]) -> fail = now -> fail -> next[i];
38     }
39 }

```

4.3 Lydon Word Decomposition (Nightfall)

满足 s 的最小后缀等于 s 本身的串称为 Lyndon 串.

等价于: s 是它自己的所有循环移位中唯一最小的一个.

任意字符串 s 可以分解为 $s = s_1 s_2 \dots s_k$, 其中 s_i 是 Lyndon 串, $s_i \geq s_{i+1}$. 且这种分解方法是唯一的。

```

1 void mnsuf(char *s, int *mn, int n) { // 每个前缀的最小后缀
2     for (int i = 0; i < n;) {
3         int j = i, k = i + 1;
4         mn[i] = i;
5         for (; k < n && s[j] <= s[k]; ++k)
6             if (s[j] == s[k]) mn[k] = mn[j] + k - j, ++j;
7             else mn[k] = j = i;
8         for (; i <= j; i += k - j) {}
9     }
10 } // lyn+=s[i..i+k-j-1]
11 void mxsuf(char *s, int *mx, int n) { // 每个前缀的最大后缀
12     fill(mx, mx + n, -1);
13     for (int i = 0; i < n;) {
14         int j = i, k = i + 1;
15         if (mx[i] == -1) mx[i] = i;
16         for (; k < n && s[j] >= s[k]; ++k) {
17             j = s[j] == s[k] ? j + 1 : i;
18             if (mx[k] == -1) mx[k] = i;
19         }
20         for (; i <= j; i += k - j) {}

```

```

21 }
22 }
```

4.4 后缀数组 (ct)

```

1 char s[maxn];
2 int sa[maxn], rank[maxn], wa[maxn], wb[maxn], cnt[maxn], height[maxn];
3 inline void build(int n, int m)
4 {
5     int *x = wa, *y = wb, *t;
6     for (int i = 1; i <= n; ++i) cnt[x[i]] = s[i] - 'a' + 1]++;
7     for (int i = 1; i <= m; ++i) cnt[i] += cnt[i - 1];
8     for (int i = n; i; --i) sa[cnt[x[i]]--] = i;
9
10    for (int j = 1; j < n || (j == 1 && m < n); j <= 1, t = x, x = y, y = t)
11    {
12        memset(cnt + 1, 0, m << 2);
13        int p = 0;
14        for (int i = n - j + 1; i <= n; ++i) y[++p] = i;
15        for (int i = 1; i <= n; ++i)
16        {
17            ++cnt[x[i]];
18            sa[i] > j ? y[++p] = sa[i] - j : 0;
19        }
20        for (int i = 1; i <= m; ++i) cnt[i] += cnt[i - 1];
21        for (int i = n; i; --i) sa[cnt[x[y[i]]]] = y[i];
22        m = 0;
23        for (int i = 1; i <= n; ++i)
24            y[sa[i]] = (i == 1 || x[sa[i]] != x[sa[i - 1]] || x[sa[i - 1] + j] != x[sa[i] + j]) ? ++m :
25            m;
26    }
27    for (int i = 1; i <= n; ++i) rank[sa[i]] = i;
28    for (int i = 1, j, k = 0; i <= n; height[rank[i++]] = k)
29        for (k ? --k : 0, j = sa[rank[i] - 1]; s[i + k] == s[j + k]; ++k);
```

4.5 后缀自动机 (ct)

```

1 struct SAM {
2     SAM *next[26], *fa;
3     int val;
4 } mem[maxn], *last = mem, *tot = mem;
5 void extend(int c)
6 {
7     R SAM *p = last, *np;
8     last = np = ++tot; np -> val = p -> val + 1;
9     for (; p && !p -> next[c]; p = p -> fa) p -> next[c] = np;
10    if (!p) np -> fa = rt[id];
11    else
12    {
13        SAM *q = p -> next[c];
14        if (q -> val == p -> val + 1) np -> fa = q;
15        else
16        {
17            SAM *nq = ++tot;
18            memcpy(nq -> next, q -> next, sizeof nq -> next);
19            nq -> val = p -> val + 1;
20            nq -> fa = q -> fa;
```

```

21         q -> fa = np -> fa = nq;
22         for ( ; p && p -> next[c] == q; p = p -> fa) p -> next[c] = nq;
23     }
24 }

```

4.6 Manacher (ct)

```

1 char str[maxn];
2 int p1[maxn], p2[maxn], n;
3 void manacher1()
4 {
5     int mx = 0, id;
6     for(int i = 1; i <= n; ++i)
7     {
8         if (mx >= i) p1[i] = dmin(mx - i, p1[(id << 1) - i]);
9         else p1[i] = 1;
10        for ( ; str[i + p1[i]] == str[i - p1[i]]; ++p1[i]) ;
11        if (p1[i] + i - 1 > mx) id = i, mx = p1[i] + i - 1;
12    }
13 }
14 void manacher2()
15 {
16     int mx = 0, id;
17     for(int i = 1; i <= n; i++)
18     {
19         if (mx >= i) p2[i] = dmin(mx - i, p2[(id << 1) - i]) ;
20         else p2[i] = 0;
21         for ( ; str[i + p2[i] + 1] == str[i - p2[i]]; ++p2[i]);
22         if (p2[i] + i > mx) id = i, mx = p2[i] + i;
23     }
24 }
25 int main()
26 {
27     scanf("%"s, str + 1);
28     n = strlen(str + 1);
29     str[0] = '#';
30     str[n + 1] = '$';
31     manacher1();
32     manacher2();
33     return 0;
34 }

```

4.7 回文树 (ct)

```

1 char str[maxn];
2 int next[maxn][26], fail[maxn], len[maxn], cnt[maxn], last, tot, n;
3 inline int new_node(int l)
4 {
5     len[++tot] = l;
6     return tot;
7 }
8 inline void init()
9 {
10    tot = -1;
11    new_node(0);
12    new_node(-1);
13    str[0] = -1;

```

```

14     fail[0] = 1;
15 }
16 inline int get_fail(int x)
17 {
18     while (str[n - len[x] - 1] != str[n]) x = fail[x];
19     return x;
20 }
21 inline void extend(int c)
22 {
23     ++n;
24     int cur = get_fail(last);
25     if (!next[cur][c])
26     {
27         int now = new_node(len[cur] + 2);
28         fail[now] = next[get_fail(fail[cur])][c];
29         next[cur][c] = now;
30     }
31     last = next[cur][c];
32     ++cnt[last];
33 }
34 long long ans;
35 inline void count()
36 {
37     for (int i = tot; i; --i)
38     {
39         cnt[fail[i]] += cnt[i];
40         cmax(ans, 1ll * len[i] * cnt[i]);
41     }
42 }
43 int main()
44 {
45     scanf("%s", str + 1);
46     init();
47     for (int i = 1; str[i]; ++i)
48         extend(str[i] - 'a');
49     count();
50     printf("%lld\n", ans );
51     return 0;
52 }
```

4.8 最小表示法 (ct)

```

1 int main()
2 {
3     int i = 0, j = 1, k = 0;
4     while (i < n && j < n && k < n)
5     {
6         int tmp = a[(i + k) % n] - a[(j + k) % n];
7         if (!tmp) k++;
8         else
9         {
10             if (tmp > 0) i += k + 1;
11             else j += k + 1;
12             if (i == j) ++j;
13             k = 0;
14         }
15     }
16     j = dmin(i, j);
17     for (int i = j; i < n; ++i) printf("%d ", a[i]);
```

```

18     for (int i = 0; i < j - 1; ++i) printf("%d ", a[i]);
19     if (j > 0) printf("%d\n", a[j - 1]);
20     return 0;
21 }
```

4.9 字符串知识 (Nightfall)

双回文串

如果 $s = x_1x_2 = y_1y_2 = z_1z_2$, $|x_1| < |y_1| < |z_1|$, x_2, y_1, y_2, z_1 是回文串, 则 x_1 和 z_2 也是回文串。

Border 的结构

字符串 s 的所有不小于 $|s|/2$ 的 border 长度构成一个等差数列。

字符串 s 的所有 border 按长度排序后可分成 $O(\log |s|)$ 段, 每段是一个等差数列。

回文串的回文后缀同时也是它的 border。

子串最小后缀

设 $s[p..n]$ 是 $s[i..n], (l \leq i \leq r)$ 中最小者, 则 $\text{minsuf}(l, r)$ 等于 $s[p..r]$ 的最短非空 border。 $\text{minsuf}(l, r) = \min\{s[p..r], \text{minsuf}(r - 2^k + 1, r)\}, (2^k < r - l + 1 \leq 2^{k+1})$ 。

子串最大后缀

从左往右, 用 set 维护后缀的字典序递减的单调队列, 并在对应时刻添加“小于事件”点以便以后修改队列; 查询直接在 set 里 lower_bound

Data Structure

5.1 莫队 (ct)

```

1 int size;
2 struct Query {
3     int l, r, id;
4     inline bool operator < (const Query &that) const {return l / size != that.l / size ? l < that.l :
5         ((l / size) & 1 ? r < that.r : r > that.r);}
6 } q[maxn];
7 int main()
8 {
9     size = (int) sqrt(n * 1.0);
10    std::sort(q + 1, q + m + 1);
11    int l = 1, r = 0;
12    for (int i = 1; i <= m; ++i)
13    {
14        for (; r < q[i].r; ) add(++r);
15        for (; r > q[i].r; ) del(r--);
16        for (; l < q[i].l; ) del(l++);
17        for (; l > q[i].l; ) add(--l);
18        /*
19         * write your code here.
20         */
21    }
22    return 0;
}

```

5.2 ST 表 (ct)

```

1 int a[maxn], f[20][maxn], n;
2 int Log[maxn];
3
4 void build()
5 {
6     for (int i = 1; i <= n; ++i) f[0][i] = a[i];
7
8     int lim = Log[n];
9     for (int j = 1; j <= lim; ++j)
10    {
11        int *fj = f[j], *fj1 = f[j - 1];
12        for (int i = 1; i <= n - (1 << j) + 1; ++i)
13            fj[i] = dmax(fj1[i], fj1[i + (1 << (j - 1))]);
14    }
15 }
16 int Query(int l, int r)
17 {
18     int k = Log[r - l + 1];
19     return dmax(f[k][l], f[k][r - (1 << k) + 1]);
}

```

```

20 }
21 int main()
22 {
23     scanf("%d", &n);
24     Log[0] = -1;
25     for (int i = 1; i <= n; ++i)
26     {
27         scanf("%d", &a[i]);
28         Log[i] = Log[i >> 1] + 1;
29     }
30     build();
31     int q;
32     scanf("%d", &q);
33     for (; q; --q)
34     {
35         int l, r; scanf("%d%d", &l, &r);
36         printf("%d\n", Query(l, r));
37     }
38 }
```

5.3 长链剖分 (ct)

```

1 void dfs(int x, int fa) {
2     for (int i = 1; i <= 20; ++i) {
3         if ((1 << i) > dep[x]) break;
4         Fa[x][i] = Fa[Fa[x][i - 1]][i - 1];
5     }
6     for (int to : e[x])
7         if (to != fa) {
8             Fa[to][0] = x;
9             dep[to] = dep[x] + 1;
10            dfs(to, x);
11            if (depmax[to] > depmax[son[x]]) son[x] = to;
12        }
13     depmax[x] = depmax[son[x]] + 1;
14 }
15
16 std::vector<int> v[maxn];
17
18 void dfs2(int x, int fa) {
19     dfn[x] = ++timer;
20     pos[timer] = x;
21     top[x] = son[fa] == x ? top[fa] : x;
22     if (top[x] == x) {
23         int now = fa;
24         v[x].push_back(x);
25         for (int i = 1; now && i <= depmax[x] + 1; ++i) {
26             v[x].push_back(now);
27             now = Fa[now][0];
28         }
29     }
30     if (son[x]) dfs2(son[x], x);
31     for (int to : e[x])
32         if (to != fa && to != son[x])
33             dfs2(to, x);
34 }
35
36 int jump(int x, int k) {
37     if (!k) return x;
```

```

38     int l = Log[k];
39     x = Fa[x][l];
40     k -= 1 << l;
41     if (k) {
42         if (dep[x] - dep[top[x]] >= k)x = pos[dfn[x] - k];
43         else k -= dep[x] - dep[top[x]];
44         x = v[top[x]][k];
45     }
46     return x;
47 }
48
49 int main() {
50     Log[0] = -1;
51     for (int i = 1; i <= n; ++i)Log[i] = Log[i >> 1] + 1;
52     dep[1] = 1;
53     dfs(1, 0);
54     dfs2(1, 0);
55 }
```

5.4 DSU (ct)

```

1 void gs(int x, int f = 0) {
2     sz[x] = 1;
3     for (int to:e[x]) {
4         if (to == f)continue;
5         gs(to, x);
6         sz[x] += sz[to];
7         if (sz[to] > sz[son[x]])son[x] = to;
8     }
9 }
10
11 void edt(int x, int f, int v) {
12     cc[col[x]] += v;
13     for (int to:e[x])
14         if (to != f && to != skip) edt(to, x, v);
15 }
16
17 void dfs(int x, int f = 0, bool kep = 0) {
18     for (int to:e[x])
19         if (to != f && to != son[x]) dfs(to, x);
20     if (son[x]) dfs(son[x], x, 1), skip = son[x];
21     edt(x, f, 1);
22     anss[x] = cc[ks[x]];
23     skip = 0;
24     if (!kep) edt(x, f, -1);
25 }
```

5.5 带权并查集 (ct)

```

1 struct edge
2 {
3     int a, b, w;
4     inline bool operator < (const edge &that) const {return w > that.w;}
5 } e[maxm];
6 int fa[maxn], f1[maxn], f2[maxn], f1cnt, f2cnt, val[maxn], size[maxn];
7 int main()
8 {
9     int n, m; scanf("%d%d", &n, &m);
```

```

10   for (int i = 1; i <= m; ++i)
11     scanf("%d%d%d", &e[i].a, &e[i].b, &e[i].w);
12   for (int i = 1; i <= n; ++i) size[i] = 1;
13   std::sort(e + 1, e + m + 1);
14   for (int i = 1; i <= m; ++i)
15   {
16     int x = e[i].a, y = e[i].b;
17     for ( ; fa[x]; x = fa[x]) ;
18     for ( ; fa[y]; y = fa[y]) ;
19     if (x != y)
20     {
21       if (size[x] < size[y]) std::swap(x, y);
22       size[x] += size[y];
23       val[y] = e[i].w;
24       fa[y] = x;
25     }
26   }
27
28   int q; scanf("%d", &q);
29   for (; q; --q)
30   {
31     int a, b; scanf("%d%d", &a, &b); f1cnt = f2cnt = 0;
32     for (; fa[a]; a = fa[a]) f1[++f1cnt] = a;
33     for (; fa[b]; b = fa[b]) f2[++f2cnt] = b;
34     if (a != b) {puts("-1"); continue;}
35     while (f1cnt && f2cnt && f1[f1cnt] == f2[f2cnt]) --f1cnt, --f2cnt;
36     int ret = 0xffffffff;
37     for (; f1cnt; --f1cnt) cmin(ret, val[f1[f1cnt]]);
38     for (; f2cnt; --f2cnt) cmin(ret, val[f2[f2cnt]]);
39     printf("%d\n", ret);
40   }
41   return 0;
42 }

```

5.6 可并堆 (ct)

```

1 struct Node {
2   Node *ch[2];
3   ll val; int size;
4   inline void update()
5   {
6     size = ch[0] -> size + ch[1] -> size + 1;
7   }
8 } mem[maxn], *rt[maxn];
9 Node *merge(Node *a, Node *b)
10 {
11   if (a == mem) return b;
12   if (b == mem) return a;
13   if (a -> val < b -> val) std::swap(a, b);
14   // a -> pushdown();
15   std::swap(a -> ch[0], a -> ch[1]);
16   a -> ch[1] = merge(a -> ch[1], b);
17   a -> update();
18   return a;
19 }

```

5.7 线段树 (ct)

zkw 线段树

0-based

```

1 inline void build()
2 {
3     for (int i = M - 1; i; --i) tr[i] = dmax(tr[i << 1], tr[i << 1 | 1]);
4 }
5 inline void Change(int x, int v)
6 {
7     x += M; tr[x] = v; x >= 1;
8     for (; x; x >= 1) tr[x] = dmax(tr[x << 1], tr[x << 1 | 1]);
9 }
10 inline int Query(int s, int t)
11 {
12     int ret = -0x7fffffff;
13     for (s = s + M - 1, t = t + M + 1; s ^ t ^ 1; s >= 1, t >= 1)
14     {
15         if (~s & 1) cmax(ret, tr[s ^ 1]);
16         if (t & 1) cmax(ret, tr[t ^ 1]);
17     }
18     return ret;
19 }
20 int main()
21 {
22     int n; scanf("%"d, &n);
23     for (M = 1; M < n; M <= 1) ;
24     for (int i = 0; i < n; ++i)
25         scanf("%"d, &tr[i + M]);
26     for (int i = n; i < M; ++i) tr[i + M] = -0x7fffffff;
27     build();
28     int q; scanf("%"d, &q);
29     for (; q; --q)
30     {
31         int l, r; scanf("%"d%"d", &l, &r); --l, --r;
32         printf("%"d\n", Query(l, r));
33     }
34     return 0;
35 }
```

李超线段树

```

1 int size[maxn], dep[maxn], son[maxn], fa[maxn], top[maxn], dfn[maxn], pos[maxn], timer, rig[maxn];
2 ll dis[maxn];
3 bool vis[maxn];
4 // 树链剖分 begin
5 void dfs1(int x);
6 void dfs2(int x){cmax(rig[top[x]], dfn[x]);}
7 inline int getlca(int a, int b);
8 // 树链剖分 end
9 struct Seg {
10     Seg *ls, *rs;
11     ll min, k, b, vl, vr;
12     // min 表示区间最小值
13     // k 表示区间内 直线标记的斜率
14     // b 表示区间内 直线标记的截距
15     // vl, vr 表示区间内 x 的最小值和最大值
16     inline void update()

```

```

17     {
18         min = dmin(ls -> min, rs -> min);
19         k > 0 ? cmin(min, k * vl + b) : cmin(min, k * vr + b);
20     }
21 } ssegg[maxn << 2], *scnt = ssegg, *rt[maxn];
22 void build(int l, int r)
23 {
24     R Seg *o = scnt; o -> k = 0; o -> b = inf;
25     o -> vl = dis[pos[l]]; o -> vr = dis[pos[r]]; o -> min = inf;
26     if (l == r) return ;
27     int mid = l + r >> 1;
28     o -> ls = ++scnt; build(l, mid);
29     o -> rs = ++scnt; build(mid + 1, r);
30     o -> update();
31 }
32 int ql, qr, qk;
33 ll qb;
34 void modify(R Seg *o, int l, int r, int k, ll b)
35 {
36     int mid = l + r >> 1;
37
38     if (ql <= l && r <= qr)
39     {
40         if (l == r)
41         {
42             cmin(o -> min, k * o -> vl + b);
43             return ;
44         }
45         ll
46         val = o -> vl * k + b,
47         var = o -> vr * k + b,
48         vbl = o -> vl * o -> k + o -> b,
49         vbr = o -> vr * o -> k + o -> b;
50         if (val <= vbl && var <= vbr)
51         {
52             o -> k = k; o -> b = b;
53             o -> update();
54             return ;
55         }
56         if (val >= vbl && var >= vbr) return ;
57         ll dam = dis[pos[mid]], vam = dam * k + b, vbm = dam * o -> k + o -> b;
58         if (val >= vbl && vam <= vbm)
59         {
60             modify(o -> ls, l, mid, o -> k, o -> b);
61             o -> k = k; o -> b = b;
62         }
63         else if (val <= vbl && vam >= vbm)
64             modify(o -> ls, l, mid, k, b);
65         else
66         {
67             if (vam <= vbm && var >= vbr)
68             {
69                 modify(o -> rs, mid + 1, r, o -> k, o -> b);
70                 o -> k = k; o -> b = b;
71             }
72             else
73                 modify(o -> rs, mid + 1, r, k, b);
74         }
75         o -> update();
76     }
77 }

```

```

78     if (ql <= mid) modify(o -> ls, l, mid, k, b);
79     if (mid < qr) modify(o -> rs, mid + 1, r, k, b);
80     o -> update();
81 }
82 ll query(R Seg *o, int l, int r)
83 {
84     if (ql <= l && r <= qr) return o -> min;
85     int mid = l + r >> 1; ll ret = inf, tmp;
86     cmin(ret, dis[pos[dmax(ql, l)]] * o -> k + o -> b);
87     cmin(ret, dis[pos[dmin(qr, r)]] * o -> k + o -> b);
88     if (ql <= mid) tmp = query(o -> ls, l, mid), cmin(ret, tmp);
89     if (mid < qr) tmp = query(o -> rs, mid + 1, r), cmin(ret, tmp);
90     return ret;
91 }
92 inline void tr_modify(int x, int f)
93 {
94     while (top[x] != top[f])
95     {
96         ql = dfn[top[x]]; qr = dfn[x];
97         modify(rt[top[x]], ql, rig[top[x]], qk, qb);
98         x = fa[top[x]];
99     }
100    ql = dfn[f]; qr = dfn[x];
101    modify(rt[top[x]], dfn[top[x]], rig[top[x]], qk, qb);
102 }
103 inline ll tr_query(int s, int t)
104 {
105     ll ret = inf, tmp;
106     while (top[s] != top[t])
107     {
108         if (dep[top[s]] < dep[top[t]])
109         {
110             ql = dfn[top[t]]; qr = dfn[t];
111             tmp = query(rt[top[t]], ql, rig[top[t]]));
112             cmin(ret, tmp);
113             t = fa[top[t]];
114         }
115         else
116         {
117             ql = dfn[top[s]]; qr = dfn[s];
118             tmp = query(rt[top[s]], ql, rig[top[s]]));
119             cmin(ret, tmp);
120             s = fa[top[s]];
121         }
122     }
123     ql = dfn[s]; qr = dfn[t]; ql > qr ? std::swap(ql, qr), 1 : 0;
124     tmp = query(rt[top[s]], dfn[top[s]], rig[top[s]]));
125     cmin(ret, tmp);
126     return ret;
127 }
128 int main()
129 {
130     int n, m; scanf("%d%d", &n, &m);
131     for (int i = 1; i < n; ++i)
132     {
133         int a, b, w; scanf("%d%d%d", &a, &b, &w); link(a, b, w);
134     }
135     dfs1(1); dfs2(1);
136     for (int i = 1; i <= n; ++i)
137         if (top[i] == i)
138         {

```

```

139     rt[i] = ++scnt;
140     build(dfn[i], rig[i]);
141 }
142 for (; m; --m)
143 {
144     int opt, s, t, lca; scanf("%d%d%d", &opt, &s, &t);
145     lca = getlca(s, t);
146     if (opt == 1)
147     {
148         int a; ll b; scanf("%d%lld", &a, &b);
149         lca = getlca(s, t);
150         qk = -a; qb = a * dis[s] + b;
151         tr_modify(s, lca);
152         qk = a; qb = a * dis[s] - dis[lca] * 2 * a + b;
153         tr_modify(t, lca);
154     }
155     else
156     {
157         printf("%lld\n", tr_query(s, t));
158     }
159 }
160 return 0;
161 }
```

吉利线段树

吉利线段树能解决一类区间与某个数取最大或最小，区间求和的问题。以区间取最小值为例，在线段树的每一个节点额外维护区间中的最大值 ma ，严格次大值 se 以及最大值个树 t 。现在假设我们要让区间 $[L, R]$ 对 x 取最小值，先在线段树中定位若干个节点，对于每个节点分三种情况讨论：

- 当 $ma \leq x$ 时，显然这一次修改不会对这个节点产生影响，直接推出。
- 当 $se < x < ma$ 时，显然这一次修改只会影响到所有最大值，所以把 num 加上 $t \times (x - ma)$ ，把 ma 更新为 x ，打上标记推出。
- 当 $x \leq se$ 时，无法直接更新这一个节点的信息，对当前节点的左儿子和右儿子递归处理。

单次操作的均摊复杂度为 $O(\log^2 n)$

线段树维护折线

对于线段树每个结点维护两个值： ans 和 max ， ans 表示只考虑这个区间的可视区间的答案， max 表示这个区间的最大值。那么问题的关键就在于如何合并两个区间，显然左区间的答案肯定可以作为总区间的答案，那么接下来就是看右区间有多少个在新加入左区间的约束后是可行的。考虑如果右区间最大值都小于等于左区间最大值那么右区间就没有贡献了，相当于是被整个挡住了。

如果大于最大值，就再考虑右区间的两个子区间：左子区间、右子区间，加入左子区间的最大值小于等于左区间最大值，那么就递归处理右子区间；否则就递归处理左子区间，然后加上右子区间原本的答案。考虑这样做的必然性：因为加入左区间最高的比左子区间最高的矮，那么相当于是左区间对于右子区间没有约束，都是左子区间产生的约束。但是右子区间的答案要用右区间答案 – 左子区间答案，不能直接调用右子区间本身答案，因为其本身答案没有考虑左子区间的约束。

线段树维护矩形面积并

线段树上维护两个值： $Cover$ 和 Len

$Cover$ 意为这个区间被覆盖了多少次

Len 意为区间被覆盖的总长度

Maintain 的时候，如果 $Cover > 0$ ， Len 直接为区间长

否则从左右子树递推 Len

修改的时候直接改 $Cover$ 就好

5.8 Splay (ct)

Splay(指针版)

```

1 struct Node *null;
2 struct Node {
3     Node *ch[2], *fa;
4     int val; bool rev;
5     inline bool type()
6     {
7         return fa -> ch[1] == this;
8     }
9     inline void pushup()
10    {
11    }
12    inline void pushdown()
13    {
14        if (rev)
15        {
16            ch[0] -> rev ^= 1;
17            ch[1] -> rev ^= 1;
18            std::swap(ch[0], ch[1]);
19            rev ^= 1;
20        }
21    }
22    inline void rotate()
23    {
24        bool d = type(); Node *f = fa, *gf = f -> fa;
25        (fa = gf, f -> fa != null) ? fa -> ch[f -> type()] = this : 0;
26        (f -> ch[d] = ch[!d]) != null ? ch[!d] -> fa = f : 0;
27        (ch[!d] = f) -> fa = this;
28        f -> pushup();
29    }
30    inline void splay()
31    {
32        for (; fa != null; rotate())
33            if (fa -> fa != null)
34                (type() == fa -> type() ? fa : this) -> rotate();
35        pushup();
36    }
37 } mem[maxn];

```

维修序列

```

1 int fa[maxn], ch[maxn][2], a[maxn], size[maxn], cnt;
2 int sum[maxn], lmx[maxn], rmx[maxn], mx[maxn], v[maxn], id[maxn], root;
3 bool rev[maxn], tag[maxn];
4 inline void update(int x)
5 {
6     int ls = ch[x][0], rs = ch[x][1];
7     size[x] = size[ls] + size[rs] + 1;
8     sum[x] = sum[ls] + sum[rs] + v[x];
9     mx[x] = gmax(mx[ls], mx[rs]);
10    cmax(mx[x], lmx[rs] + rmx[ls] + v[x]);
11    lmx[x] = gmax(lmx[ls], sum[ls] + v[x] + lmx[rs]);
12    rmx[x] = gmax(rmx[rs], sum[rs] + v[x] + rmx[ls]);
13 }
14 inline void pushdown(int x)
15 {

```

```

16     int ls = ch[x][0], rs = ch[x][1];
17     if (tag[x])
18     {
19         rev[x] = tag[x] = 0;
20         if (ls) tag[ls] = 1, v[ls] = v[x], sum[ls] = size[ls] * v[x];
21         if (rs) tag[rs] = 1, v[rs] = v[x], sum[rs] = size[rs] * v[x];
22         if (v[x]>= 0)
23         {
24             if (ls) lmx[ls] = rmx[ls] = mx[ls] = sum[ls];
25             if (rs) lmx[rs] = rmx[rs] = mx[rs] = sum[rs];
26         }
27     else
28     {
29         if (ls) lmx[ls] = rmx[ls] = 0, mx[ls] = v[x];
30         if (rs) lmx[rs] = rmx[rs] = 0, mx[rs] = v[x];
31     }
32 }
33 if (rev[x])
34 {
35     rev[x] ^= 1; rev[ls] ^= 1; rev[rs] ^= 1;
36     swap(lmx[ls], rmx[ls]);swap(lmx[rs], rmx[rs]);
37     swap(ch[ls][0], ch[ls][1]); swap(ch[rs][0], ch[rs][1]);
38 }
39 }
40 inline void rotate(int x)
{
41     int f = fa[x], gf = fa[f], d = ch[f][1] == x;
42     if (f == root) root = x;
43     (ch[f][d] = ch[x][d ^ 1]) > 0 ? fa[ch[f][d]] = f : 0;
44     (fa[x] = gf) > 0 ? ch[gf][ch[gf][1] == f] = x : 0;
45     fa[ch[x][d ^ 1] = f] = x;
46     update(f);
47 }
48 inline void splay(int x, int rt)
{
49     while (fa[x] != rt)
50     {
51         int f = fa[x], gf = fa[f];
52         if (gf != rt) rotate((ch[gf][1] == f) ^ (ch[f][1] == x) ? x : f);
53         rotate(x);
54     }
55     update(x);
56 }
57 void build(int l, int r, int rt)
{
58     if (l > r) return ;
59     int mid = l + r >> 1, now = id[mid], last = id[rt];
60     if (l == r)
61     {
62         sum[now] = a[l];
63         size[now] = 1;
64         tag[now] = rev[now] = 0;
65         if (a[l] >= 0) lmx[now] = rmx[now] = mx[now] = a[l];
66         else lmx[now] = rmx[now] = 0, mx[now] = a[l];
67     }
68     else
69     {
70         build(l, mid - 1, mid);
71         build(mid + 1, r, mid);
72     }
73     v[now] = a[mid];
74 }
```

```

77     fa[now] = last;
78     update(now);
79     ch[last][mid >= rt] = now;
80 }
81 int find(int x, int rank)
82 {
83     if (tag[x] || rev[x]) pushdown(x);
84     int ls = ch[x][0], rs = ch[x][1], lsize = size[ls];
85     if (lsize + 1 == rank) return x;
86     if (lsize >= rank)
87         return find(ls, rank);
88     else
89         return find(rs, rank - lsize - 1);
90 }
91 inline int prepare(int l, int tot)
92 {
93     int x = find(root, l - 1), y = find(root, l + tot);
94     splay(x, 0);
95     splay(y, x);
96     return ch[y][0];
97 }
98 std::queue<int> q;
99 inline void Insert(int left, int tot)
100 {
101     for (int i = 1; i <= tot; ++i) a[i] = FastIn();
102     for (int i = 1; i <= tot; ++i)
103         if (!q.empty()) id[i] = q.front(), q.pop();
104         else id[i] = ++cnt;
105     build(1, tot, 0);
106     int z = id[1 + tot] >> 1;
107     int x = find(root, left), y = find(root, left + 1);
108     splay(x, 0);
109     splay(y, x);
110     fa[z] = y;
111     ch[y][0] = z;
112     update(y);
113     update(x);
114 }
115 void rec(int x)
116 {
117     if (!x) return ;
118     int ls = ch[x][0], rs = ch[x][1];
119     rec(ls); rec(rs); q.push(x);
120     fa[x] = ch[x][0] = ch[x][1] = 0;
121     tag[x] = rev[x] = 0;
122 }
123 inline void Delete(int l, int tot)
124 {
125     int x = prepare(l, tot), f = fa[x];
126     rec(x); ch[f][0] = 0;
127     update(f); update(fa[f]);
128 }
129 inline void Makesame(int l, int tot, int val)
130 {
131     int x = prepare(l, tot), y = fa[x];
132     v[x] = val; tag[x] = 1; sum[x] = size[x] * val;
133     if (val >= 0) lmx[x] = rmx[x] = mx[x] = sum[x];
134     else lmx[x] = rmx[x] = 0, mx[x] = val;
135     update(y); update(fa[y]);
136 }
137 inline void Reverse(int l, int tot)

```

```

138 {
139     int x = prepare(l, tot), y = fa[x];
140     if (!tag[x])
141     {
142         rev[x] ^= 1;
143         swap(ch[x][0], ch[x][1]);
144         swap(lmx[x], rmx[x]);
145         update(y); update(fa[y]);
146     }
147 }
148 inline void Query(int l, int tot)
149 {
150     int x = prepare(l, tot);
151     printf("%d\n", sum[x] );
152 }
153 #define inf ((1 << 30))
154 int main()
155 {
156     int n = FastIn(), m = FastIn(), l, tot, val;
157     char op, op2;
158     mx[0] = a[1] = a[n + 2] = -inf;
159     for (int i = 2; i <= n + 1; i++)
160     {
161         a[i] = FastIn();
162     }
163     for (int i = 1; i <= n + 2; ++i) id[i] = i;
164     n += 2; cnt = n; root = (n + 1) >> 1;
165     build(1, n, 0);
166     for (int i = 1; i <= m; i++)
167     {
168         op = getc();
169         while (op < 'A' || op > 'Z') op = getc();
170         getc(); op2 = getc(); getc(); getc(); getc();
171         if (op == 'M' && op2 == 'X')
172         {
173             printf("%d\n", mx[root] );
174         }
175         else
176         {
177             l = FastIn() + 1; tot = FastIn();
178             if (op == 'I') Insert(l, tot);
179             if (op == 'D') Delete(l, tot);
180             if (op == 'M') val = FastIn(), Makesame(l, tot, val);
181             if (op == 'R')
182                 Reverse(l, tot);
183             if (op == 'G')
184                 Query(l, tot);
185         }
186     }
187     return 0;
188 }

```

5.9 Treap (ct)

```

1 struct Treap {
2     Treap *ls, *rs;
3     int size;
4     bool rev;
5     inline void update()

```

```

6     {
7         size = ls -> size + rs -> size + 1;
8     }
9     inline void set_rev()
10    {
11        rev ^= 1;
12        std::swap(ls, rs);
13    }
14    inline void pushdown()
15    {
16        if (rev)
17        {
18            ls -> set_rev();
19            rs -> set_rev();
20            rev = 0;
21        }
22    }
23 } mem[maxn], *root, *null = mem;
24 struct Pair {
25     Treap *fir, *sec;
26 };
27 Treap *build(R int l, R int r)
28 {
29     if (l > r) return null;
30     R int mid = l + r >> 1;
31     R Treap *now = mem + mid;
32     now -> rev = 0;
33     now -> ls = build(l, mid - 1);
34     now -> rs = build(mid + 1, r);
35     now -> update();
36
37     return now;
38 }
39 inline Treap *Find_kth(R Treap *now, R int k)
40 {
41     if (!k) return mem;
42     if (now -> ls -> size >= k) return Find_kth(now -> ls, k);
43     else if (now -> ls -> size + 1 == k) return now;
44     else return Find_kth(now -> rs, k - now -> ls -> size - 1);
45 }
46 Treap *merge(R Treap *a, R Treap *b)
47 {
48     if (a == null) return b;
49     if (b == null) return a;
50     if (rand() % (a -> size + b -> size) < a -> size)
51     {
52         a -> pushdown();
53         a -> rs = merge(a -> rs, b);
54         a -> update();
55         return a;
56     }
57     else
58     {
59         b -> pushdown();
60         b -> ls = merge(a, b -> ls);
61         b -> update();
62         return b;
63     }
64 }
65 Pair split(R Treap *now, R int k)
66 {

```

```

67     if (now == null) return (Pair) {null, null};
68     R Pair t = (Pair) {null, null};
69     now -> pushdown();
70     if (k <= now -> ls -> size)
71     {
72         t = split(now -> ls, k);
73         now -> ls = t.sec;
74         now -> update();
75         t.sec = now;
76     }
77     else
78     {
79         t = split(now -> rs, k - now -> ls -> size - 1);
80         now -> rs = t.fir;
81         now -> update();
82         t.fir = now;
83     }
84     return t;
85 }
86 inline void set_rev(int l, int r)
87 {
88     R Pair x = split(root, l - 1);
89     R Pair y = split(x.sec, r - l + 1);
90     y.fir -> set_rev();
91     root = merge(x.fir, merge(y.fir, y.sec));
92 }
```

5.10 可持久化平衡树 (ct)

```

1 char str[maxn];
2 struct Treap
3 {
4     Treap *ls, *rs;
5     char data; int size;
6     inline void update()
7     {
8         size = ls -> size + rs -> size + 1;
9     }
10 } *root[maxn], mem[maxcnt], *tot = mem, *last = mem, *null = mem;
11 inline Treap* new_node(char ch)
12 {
13     *++tot = (Treap) {null, null, ch, 1};
14     return tot;
15 }
16 struct Pair
17 {
18     Treap *fir, *sec;
19 };
20 inline Treap* copy(Treap* x)
21 {
22     if (x == null) return null;
23     if (x > last) return x;
24     *++tot = *x;
25     return tot;
26 }
27 Pair Split(Treap* x, int k)
28 {
29     if (x == null) return (Pair) {null, null};
30     Pair y;
```

```
31     Treap *nw = copy(x);
32     if (nw -> ls -> size >= k)
33     {
34         y = Split(nw -> ls, k);
35         nw -> ls = y.sec;
36         nw -> update();
37         y.sec = nw;
38     }
39     else
40     {
41         y = Split(nw -> rs, k - nw -> ls -> size - 1);
42         nw -> rs = y.fir;
43         nw -> update();
44         y.fir = nw;
45     }
46     return y;
47 }
48 Treap *Merge(Treap *a, Treap *b)
49 {
50     if (a == null) return b;
51     if (b == null) return a;
52     Treap *nw;
53     if (rand() % (a -> size + b -> size) < a -> size)
54     {
55         nw = copy(a);
56         nw -> rs = Merge(nw -> rs, b);
57     }
58     else
59     {
60         nw = copy(b);
61         nw -> ls = Merge(a, nw -> ls);
62     }
63     nw -> update();
64     return nw;
65 }
66 Treap *Build(int l, int r)
67 {
68     if (l > r) return null;
69     int mid = l + r >> 1;
70     Treap *nw = new_node(str[mid]);
71     nw -> ls = Build(l, mid - 1);
72     nw -> rs = Build(mid + 1, r);
73     nw -> update();
74     return nw;
75 }
76 int now;
77 inline void Insert(int k, char ch)
78 {
79     Pair x = Split(root[now], k);
80     Treap *nw = new_node(ch);
81     root[++now] = Merge(Merge(x.fir, nw), x.sec);
82 }
83 inline void Del(int l, int r)
84 {
85     Pair x = Split(root[now], l - 1);
86     Pair y = Split(x.sec, r - l + 1);
87     root[++now] = Merge(x.fir, y.sec);
88 }
89 inline void Copy(int l, int r, int ll)
90 {
91     Pair x = Split(root[now], l - 1);
```

```

92     Pair y = Split(x.sec, r - l + 1);
93     Pair z = Split(root[now], ll);
94     Treap *ans = y.fir;
95     root[++now] = Merge(Merge(z.fir, ans), z.sec);
96 }
97 void Print(Treap *x, int l, int r)
98 {
99     if (!x) return ;
100    if (l > r) return;
101    int mid = x -> ls -> size + 1;
102    if (r < mid)
103    {
104        Print(x -> ls, l, r);
105        return ;
106    }
107    if (l > mid)
108    {
109        Print(x -> rs, l - mid, r - mid);
110        return ;
111    }
112    Print(x -> ls, l, mid - 1);
113    printf("%c", x -> data );
114    Print(x -> rs, 1, r - mid);
115 }
116 void Printtree(Treap *x)
117 {
118     if (!x) return;
119     Printtree(x -> ls);
120     printf("%c", x -> data );
121     Printtree(x -> rs);
122 }
123 int main()
124 {
125     srand(time(0) + clock());
126     null -> ls = null -> rs = null; null -> size = 0; null -> data = 0;
127     int n = F();
128     gets(str + 1);
129     int len = strlen(str + 1);
130     root[0] = Build(1, len);
131     while (1)
132     {
133         last = tot;
134         char opt = getc();
135         while (opt < 'A' || opt > 'Z')
136         {
137             if (opt == EOF) return 0;
138             opt = getc();
139         }
140         if (opt == 'I')
141         {
142             int x = F();
143             char ch = getc();
144             Insert(x, ch);
145         }
146         else if (opt == 'D')
147         {
148             int l = F(), r = F();
149             Del(l, r);
150         }
151         else if (opt == 'C')
152         {

```

```

153     int x = F(), y = F(), z = F();
154     Copy(x, y, z);
155 }
156 else if (opt == 'P')
157 {
158     int x = F(), y = F(), z = F();
159     Print(root[now - x], y, z);
160     puts("");
161 }
162 }
163 return 0;
164 }
```

5.11 二进制分组 (ct)

用线段树维护时间的操作序列，每次操作一个一个往线段树里面插，等到一个线段被插满的时候用归并来维护区间的信息。查询的时候如果一个线段没有被插满就递归下去。定位到一个区间的时候在区间里面归并出来的信息二分。

```

1 int x[maxn], tnum;
2 struct Seg {
3     int l, r, a, b;
4 } p[maxn * 200];
5 int lef[maxm << 2], rig[maxm << 2], pcnt, ta, tb, ql, qr, n, m, k, ans;
6 void update(int o, int l, int r)
7 {
8     lef[o] = pcnt + 1;
9     for (int i = lef[o << 1], j = lef[o << 1 | 1], head = 1; i <= rig[o << 1] || j <= rig[o << 1 | 1];
10    )
11        if (p[i].r <= p[j].r)
12        {
13            p[++pcnt] = (Seg) {head, p[i].r, 1ll * p[i].a * p[j].a % m, (1ll * p[j].a * p[i].b +
14                           → p[j].b) % m};
15            head = p[i].r + 1;
16            p[i].r == p[j].r ? ++j : 0; ++i;
17        }
18        else
19        {
20            p[++pcnt] = (Seg) {head, p[j].r, 1ll * p[i].a * p[j].a % m, (1ll * p[j].a * p[i].b +
21                           → p[j].b) % m};
22            head = p[j].r + 1; ++j;
23        }
24    rig[o] = pcnt;
25 }
26 int find(int o, int t, int &s)
27 {
28     int l = lef[o], r = rig[o];
29     while (l < r)
30     {
31         int mid = l + r >> 1;
32         if (t <= p[mid].r) r = mid;
33         else l = mid + 1;
34     }
35     // printf("%d %d t %d s %d %d %d\n", p[l].l, p[l].r, t, s, p[l].a, p[l].b);
36     s = (1ll * s * p[l].a + p[l].b) % m;
37 }
38 void modify(int o, int l, int r, int t)
39 {
40     if (l == r)
41     {
```

```

39     lef[o] = pcnt + 1;
40     ql > 1 ? p[++pcnt] = (Seg) {1, ql - 1, 1, 0}, 1: 0;
41     p[++pcnt] = (Seg) {ql, qr, ta, tb};
42     qr < n ? p[++pcnt] = (Seg) {qr + 1, n, 1, 0}, 1: 0;
43     rig[o] = pcnt;
44     return ;
45 }
46 int mid = l + r >> 1;
47 if (t <= mid) modify(o << 1, l, mid, t);
48 else modify(o << 1 | 1, mid + 1, r, t);
49
50 if (t == r) update(o, l, r);
51 }
52 void query(int o, int l, int r)
{
53     if (ql <= l && r <= qr)
54     {
55         find(o, k, ans);
56         return ;
57     }
58     int mid = l + r >> 1;
59     if (ql <= mid) query(o << 1, l, mid);
60     if (mid < qr) query(o << 1 | 1, mid + 1, r);
61 }
62
63 int main()
{
64     int type; scanf("%d%d%d", &type, &n, &m);
65     for (int i = 1; i <= n; ++i) scanf("%d", &x[i]);
66     int Q; scanf("%d", &Q);
67     for (int QQ = 1; QQ <= Q; ++QQ)
68     {
69         int opt, l, r; scanf("%d%d%d", &opt, &l, &r);
70         type & 1 ? l ^= ans, r ^= ans : 0;
71         if (opt == 1)
72         {
73             scanf("%d%d", &ta, &tb); ++tnum; ql = l; qr = r;
74             modify(1, 1, Q, tnum);
75         }
76         else
77         {
78             scanf("%d", &k); type & 1 ? k ^= ans : 0; ql = l; qr = r;
79             ans = x[k];
80             query(1, 1, Q);
81             printf("%d\n", ans);
82         }
83     }
84     return 0;
85 }
86 }
```

5.12 CDQ 分治 (ct)

```

1 struct event
2 {
3     int x, y, id, opt, ans;
4 } t[maxn], q[maxn];
5 void cdq(int left, int right)
6 {
7     if (left == right) return ;
8     int mid = left + right >> 1;
```

```

9 cdq(left, mid);
10 cdq(mid + 1, right);
11 //分成若干个子问题
12 ++now;
13 for (int i = left, j = mid + 1; j <= right; ++j)
14 {
15     for (; i <= mid && q[i].x <= q[j].x; ++i)
16         if (!q[i].opt)
17             add(q[i].y, q[i].ans);
18     //考虑前面的修改操作对后面的询问的影响
19     if (q[j].opt)
20         q[j].ans += query(q[j].y);
21 }
22 int i, j, k = 0;
23 //以下相当于归并排序
24 for (i = left, j = mid + 1; i <= mid && j <= right; )
25 {
26     if (q[i].x <= q[j].x)
27         t[k++] = q[i++];
28     else
29         t[k++] = q[j++];
30 }
31 for (; i <= mid; )
32     t[k++] = q[i++];
33 for (; j <= right; )
34     t[k++] = q[j++];
35 for (int i = 0; i < k; ++i)
36     q[left + i] = t[i];
37 }

```

5.13 挑战 (ct)

```

1 ef unsigned int uint;
2 typedef unsigned long long ull;

```

5.14 斜率优化 (ct)

对于斜截式 $y = kx + b$, 如果把 k_i 看成斜率, 那 dp 时需要最小化截距, 把斜截式转化为 $b_i = -k_i x_j + y_j$, 就可以把可以转移到这个状态的点看作是二维平面上的点 $(-x_j, y_j)$, 问题转化为了在平面上找一个点使得斜率为 k_i 的直线的截距最小。这样的点一定在凸包上, 这样的点在凸包上和前一个点的斜率 $\leq k_i$, 和后面一个点的斜率 $\geq k_i$ 。这样就可以在凸包上二分来加速转移。当点的横坐标 x_i 和斜率 k_i 都是单调的, 还可以用单调队列来维护凸包。

单调队列

```

1 int a[maxn], n, l;
2 ll sum[maxn], f[maxn];
3 inline ll sqr(ll x) {return x * x;}
4 #define y(_i) (f[_i] + sqr(sum[_i] + l))
5 #define x(_i) (2 * sum[_i])
6 inline double slope(int i, int j)
7 {
8     return (y(i) - y(j)) / (1.0 * (x(i) - x(j)));
9 }
10 int q[maxn];
11 int main()
12 {

```

```

13     n = F(), l = F() + 1;
14     for (int i = 1; i <= n; ++i) a[i] = F(), sum[i] = sum[i - 1] + a[i];
15     for (int i = 1; i <= n; ++i) sum[i] += i;
16     f[0] = 0;
17     /*
18      memset(f, 63, sizeof (f));
19      for (int i = 1; i <= n; ++i)
20      {
21          int pos;
22          for (int j = 0; j < i; ++j)
23          {
24              long long tmp = f[j] + sqrt(sum[i] - sum[j] - l);
25              f[i] > tmp ? f[i] = tmp, pos = j : 0;
26          }
27      }
28 */
29     int h = 1, t = 1;
30     q[h] = 0;
31     for (int i = 1; i <= n; ++i)
32     {
33         while (h < t && slope(q[h], q[h + 1]) <= sum[i]) ++h;
34         f[i] = f[q[h]] + sqrt(sum[i] - sum[q[h]] - l);
35         while (h < t && slope(q[t - 1], i) < slope(q[t - 1], q[t])) --t;
36         q[++t] = i;
37     }
38     printf("%lld\n", f[n] );
39     return 0;
40 }
```

线段树

```

1 // NOI 2014 购票
2 int dep[maxn], fa[maxn], son[maxn], dfn[maxn], timer, pos[maxn], size[maxn], n, top[maxn];
3 ll d[maxn], p[maxn], q[maxn], l[maxn], f[maxn];
4 int stcnt;
5 void dfs1(int x);
6 void dfs2(int x);
7 #define P pair<ll, ll>
8 #define mfp make_pair
9 #define x first
10 #define y second
11 #define inf ~ULL >> 2
12 inline double slope(const P &a, const P &b)
13 {
14     return (b.y - a.y) / (double) (b.x - a.x);
15 }
16 struct Seg
17 {
18     vector<P> v;
19     inline void add(const P &that)
20     {
21         int top = v.size();
22         P *v = this -> v.data() - 1;
23         while (top > 1 && slope(v[top - 1], v[top]) > slope(v[top], that)) --top;
24         this -> v.erase(this -> v.begin() + top, this -> v.end());
25         this -> v.push_back(that);
26     }
27     inline ll query(ll k)
28     {
29         if (v.empty()) return inf;
```

```

30     int l = 0, r = v.size() - 1;
31     while (l < r)
32     {
33         int mid = l + r >> 1;
34         if (slope(v[mid], v[mid + 1]) > k) r = mid;
35         else l = mid + 1;
36     }
37     cmin(l, v.size() - 1);
38     return v[l].y - v[l].x * k;
39 }
40 } tr[1 << 19];
41 void Change(int o, int l, int r, int x, P val)
42 {
43     tr[o].add(val);
44     if (l == r) return;
45     int mid = l + r >> 1;
46     if (x <= mid) Change(o << 1, l, mid, x, val);
47     else Change(o << 1 | 1, mid + 1, r, x, val);
48 }
49 int ql, qr, now, tmp;
50 ll len;
51 inline ll Query(int o, int l, int r)
52 {
53     if (ql <= l && r <= qr && d[tmp] - d[pos[r]] > len) return inf;
54     if (ql <= l && r <= qr && d[tmp] - d[pos[l]] <= len)
55         return tr[o].query(p[now]);
56     ll ret = inf, temp;
57     int mid = l + r >> 1;
58     if (ql <= mid) temp = Query(o << 1, l, mid), cmin(ret, temp);
59     if (mid < qr) temp = Query(o << 1 | 1, mid + 1, r), cmin(ret, temp);
60     return ret;
61 }
62 inline ll calc()
63 {
64     ll ret = inf;
65     ll lx = l[now];
66     tmp = now;
67     while (lx >= 0 && tmp)
68     {
69         len = lx;
70         ql = dfn[top[tmp]];
71         qr = dfn[tmp];
72         ll g = Query(1, 1, n);
73         cmin(ret, g);
74         lx -= d[tmp] - d[fa[top[tmp]]];
75         tmp = fa[top[tmp]];
76     }
77     return ret;
78 }
79 int main()
80 {
81     n = F(); int t = F();
82     for (int i = 2; i <= n; ++i)
83     {
84         fa[i] = F(); ll dis = F(); p[i] = F(), q[i] = F(), l[i] = F();
85         link(fa[i], i); d[i] = d[fa[i]] + dis;
86     }
87     dfs1(1);
88     dfs2(1);
89     Change(1, 1, n, 1, mfp(0, 0));
90     for (now = 2; now <= n; ++now)

```

```

91     {
92         f[now] = calc() + q[now] + d[now] * p[now];
93         Change(1, 1, n, dfn[now], mfp(d[now], f[now]));
94         printf("%lld\n", f[now] );
95     }
96     return 0;
97 }
```

5.15 树分块 (ct)

树分块套分块：给定一棵有点权的树，每次询问链上不同点权个数

```

1 int col[maxn], hash[maxn], hcnt, n, m;
2 int near[maxn];
3 bool vis[maxn];
4 int mark[maxn], mcnt, tcnt[maxn], tans;
5 int pre[256][maxn];
6 struct Block {
7     int cnt[256];
8 } mem[maxn], *tot = mem;
9 inline Block *nw(Block *last, int v)
10 {
11     Block *ret = ++tot;
12     memcpy(ret -> cnt, last -> cnt, sizeof (ret -> cnt));
13     ++ret -> cnt[v & 255];
14     return ret;
15 }
16 struct Arr {
17     Block *b[256];
18     inline int v(int c) {return b[c >> 8] -> cnt[c & 255];}
19 } c[maxn];
20 inline Arr cp(Arr last, int v)
21 {
22     Arr ret;
23     memcpy(ret.b, last.b, sizeof (ret.b));
24     ret.b[v >> 8] = nw(last.b[v >> 8], v);
25     return ret;
26 }
27 void bfs()
28 {
29     int head = 0, tail = 1; q[1] = 1;
30     while (head < tail)
31     {
32         int now = q[++head]; size[now] = 1; vis[now] = 1; dep[now] = dep[fa[now]] + 1;
33         for (Edge *iter = last[now]; iter; iter = iter -> next)
34             if (!vis[iter -> to])
35                 fa[q[++tail] = iter -> to] = now;
36     }
37     for (int i = n; i; --i)
38     {
39         int now = q[i];
40         size[fa[now]] += size[now];
41         size[son[fa[now]]] < size[now] ? son[fa[now]] = now : 0;
42     }
43     for (int i = 0; i < 256; ++i) c[0].b[i] = mem;
44     for (int i = 1; i <= n; ++i)
45     {
46         int now = q[i];
47         c[now] = cp(c[fa[now]], col[now]);
48         top[now] = son[fa[now]] == now ? top[fa[now]] : now;

```

```

49     }
50 }
51 inline int getlca(int a, int b) ;
52 void dfs_init(int x)
53 {
54     vis[x] = 1; ++tcnt[col[x]] == 1 ? ++tans : 0;
55     pre[mcnt][x] = tans;
56     for (Edge *iter = last[x]; iter; iter = iter -> next)
57         if (!vis[iter -> to]) dfs_init(iter -> to);
58     --tcnt[col[x]] == 0 ? --tans : 0;
59 }
60 int jp[maxn];
61 int main()
62 {
63     scanf("d%d", &n, &m);
64     for (int i = 1; i <= n; ++i) scanf("d", &col[i]), hash[+hcnt] = col[i];
65     std::sort(hash + 1, hash + hcnt + 1);
66     hcnt = std::unique(hash + 1, hash + hcnt + 1) - hash - 1;
67     for (int i = 1; i <= n; ++i) col[i] = std::lower_bound(hash + 1, hash + hcnt + 1, col[i]) - hash;
68     for (int i = 1; i < n; ++i)
69     {
70         int a, b; scanf("d%d", &a, &b); link(a, b);
71     }
72     bfs();
73     int D = sqrt(n);
74     for (int i = 1; i <= n; ++i)
75         if (dep[i] % D == 0 && size[i] >= D)
76         {
77             memset(vis, 0, n + 1);
78             mark[i] = ++mcnt;
79             dfs_init(i);
80         }
81     for (int i = 1; i <= n; ++i) near[q[i]] = mark[q[i]] ? q[i] : near[fa[q[i]]];
82     int ans = 0;
83     memset(vis, 0, n + 1);
84     for (; m; --m)
85     {
86         int x, y; scanf("d%d", &x, &y);
87         x ^= ans; ans = 0;
88         int lca = getlca(x, y);
89         if (dep[near[x]] < dep[lca]) std::swap(x, y);
90         if (dep[near[x]] >= dep[lca])
91         {
92             Arr *_a = c + near[x];
93             Arr *_b = c + y;
94             Arr *_c = c + lca;
95             Arr *_d = c + fa[lca];
96             for (; !mark[x]; x = fa[x])
97                 if (_a -> v(col[x]) + _b -> v(col[x]) == _c -> v(col[x]) + _d -> v(col[x]) &&
98                     !vis[col[x]])
99                     vis[jp[++ans] = col[x]] = 1;
100                for (int i = 1; i <= ans; ++i) vis[jp[i]] = 0;
101                ans += pre[mark[near[x]]][y];
102            }
103            else
104            {
105                for (; x != lca; x = fa[x]) !vis[col[x]] ? vis[jp[++ans] = col[x]] = 1 : 0;
106                for (; y != lca; y = fa[y]) !vis[col[y]] ? vis[jp[++ans] = col[y]] = 1 : 0;
107                !vis[col[lca]] ? vis[jp[++ans] = col[lca]] = 1 : 0;
108                for (int i = 1; i <= ans; ++i) vis[jp[i]] = 0;
109            }
}

```

```

109         printf("%d\n", ans);
110     }
111     return 0;
112 }
```

5.16 KD tree (lhy,cxy)

KD tree (lhy)

```

1 inline int cmp(const lhy &a,const lhy &b)
2 {
3     return a.d[D]<b.d[D];
4 }
5
6 inline void updata(int x)
7 {
8     if(p[x].l)
9     {
10         for(int i=0;i<2;i++)
11             p[x].min[i]=min(p[x].min[i],p[p[x].l].min[i]),
12             p[x].max[i]=max(p[x].max[i],p[p[x].l].max[i]);
13     }
14     if(p[x].r)
15     {
16         for(int i=0;i<2;i++)
17             p[x].min[i]=min(p[x].min[i],p[p[x].r].min[i]),
18             p[x].max[i]=max(p[x].max[i],p[p[x].r].max[i]);
19     }
20 }
21
22 int build(int l,int r,int d)
23 {
24     D=d;
25     int mid=(l+r)>>1;
26     nth_element(p+l,p+mid,p+r+1,cmp);
27     for(int i=0;i<2;i++)
28         p[mid].max[i]=p[mid].min[i]=p[mid].d[i];
29     if(l<mid)p[mid].l=build(l,mid-1,d^1);
30     if(mid<r)p[mid].r=build(mid+1,r,d^1);
31     updata(mid);
32     return mid;
33 }
34
35 void insert(int now,int D)
36 {
37     if(p[now].d[D]>=p[n].d[D])
38     {
39         if(p[now].l)insert(p[now].l,D^1);
40         else p[now].l=n;
41         updata(now);
42     }
43     else
44     {
45         if(p[now].r)insert(p[now].r,D^1);
46         else p[now].r=n;
47         updata(now);
48     }
49 }
```

```

51 int dist(lhy &P,int X,int Y)
52 {
53     int nowans=0;
54     if(X>=P.max[0])nowans+=X-P.max[0];
55     if(X<=P.min[0])nowans+=P.min[0]-X;
56     if(Y>=P.max[1])nowans+=Y-P.max[1];
57     if(Y<=P.min[1])nowans+=P.min[1]-Y;
58     return nowans;
59 }
60
61 void ask1(int now)
62 {
63     int pl,pr;
64     ans=min(ans,abs(x-p[now].d[0])+abs(y-p[now].d[1]));
65     if(p[now].l)pl=dist(p[p[now].l],x,y);
66     else pl=0x3f3f3f3f;
67     if(p[now].r)pr=dist(p[p[now].r],x,y);
68     else pr=0x3f3f3f3f;
69     if(pl<pr)
70     {
71         if(pl<ans)ask(p[now].l);
72         if(pr<ans)ask(p[now].r);
73     }
74     else
75     {
76         if(pr<ans)ask(p[now].r);
77         if(pl<ans)ask(p[now].l);
78     }
79 }
80
81 void ask2(int now)
82 {
83     if(x1<=p[now].min[0]&&x2>=p[now].max[0]&&y1<=p[now].min[1]&&y2>=p[now].max[1])
84     {
85         ans+=p[now].sum;
86         return;
87     }
88     if(x1>p[now].max[0]||x2<p[now].min[0]||y1>p[now].max[1]||y2<p[now].min[1])return;
89     if(x1<=p[now].d[0]&&x2>=p[now].d[0]&&y1<=p[now].d[1]&&y2>=p[now].d[1])ans+=p[now].val;
90     if(p[now].l)ask(p[now].l);
91     if(p[now].r)ask(p[now].r);
92 }

```

KD tree (cxy)

```

1 namespace KD_Tree {
2     const long long inf = (long long)1e18;
3     const double Alpha = 0.8;
4     struct Point {
5         int x, y;
6         Point(int x = 0, int y = 0) : x(x), y(y) {}
7     };
8     long long disof(const Point &a, const Point &b) {
9         long long dx = a.x - b.x, dy = a.y - b.y;
10        return dx * dx + dy * dy;
11    }
12    struct Node {
13        Node *son[2];
14        int d;
15        int id, size;

```

```

16     Point pos;
17 #define L first
18 #define R second
19     pair<int, int> segx, segy;
20
21     friend int get_size(Node *a) { return a == NULL ? 0 : a->size; }
22
23     bool is_balanced() {
24         return get_size.son[0]) < size * Alpha
25             && get_size.son[1]) < size * Alpha;
26     }
27
28     void maintain() {
29         size = 1;
30         segx = make_pair(pos.x, pos.x);
31         segy = make_pair(pos.y, pos.y);
32         for (int i = 0; i < 2; i++) if (son[i] != NULL) {
33             chkmin(segx.L, son[i]->segx.L),
34             chkmin(segy.L, son[i]->segy.L),
35             chkmax(segx.R, son[i]->segx.R),
36             chkmax(segy.R, son[i]->segy.R),
37             size += son[i]->size;
38         }
39     }
40     void init(Point p, int i) {
41         son[0] = son[1] = NULL;
42         d = rand() % 2;
43         pos = p; id = i;
44         maintain();
45     }
46
47     long long get_dis(Point pos) {
48         long long ans = inf;
49         if (segx.L <= pos.x && pos.x <= segx.R
50             && segy.L <= pos.y && pos.y <= segy.R) return 0;
51         if (segx.L <= pos.x && pos.x <= segx.R)
52             chkmin(ans, 1ll * abs(pos.y - segy.L) * abs(pos.y - segy.L)),
53             chkmin(ans, 1ll * abs(pos.y - segy.R) * abs(pos.y - segy.R));
54         if (segy.L <= pos.y && pos.y <= segy.R)
55             chkmin(ans, 1ll * abs(pos.x - segx.L) * abs(pos.x - segx.L)),
56             chkmin(ans, 1ll * abs(pos.x - segx.R) * abs(pos.x - segx.R));
57         long long dx = min(abs(pos.x - segx.L), abs(pos.x - segx.R));
58         long long dy = min(abs(pos.y - segy.L), abs(pos.y - segy.R));
59         ans = min(ans, dx * dx + dy * dy);
60         return ans;
61     }
62     #undef L
63     #undef R
64 } mem[maxn], *tot = mem, *root, *tmp[maxn];
65
66     bool cmpx(Node *a, Node *b) { return a->pos.x < b->pos.x; }
67     bool cmpy(Node *a, Node *b) { return a->pos.y < b->pos.y; }
68     bool (*cmp[2])(Node *, Node *) = {cmpx, cmpy};
69
70     Node *build(int l, int r) {
71         if (l > r) return NULL;
72         int mid = l + r >> 1, d = rand() % 2;
73         nth_element(tmp + l, tmp + mid, tmp + r + 1, cmp[d]);
74         tmp[mid]->son[0] = build(l, mid - 1);
75         tmp[mid]->son[1] = build(mid + 1, r);
76         tmp[mid]->maintain();

```

```

77     tmp[mid]->d = d;
78     return tmp[mid];
79 }
80
81 int top;
82 void travel(Node *x) {
83     if (x == NULL) return;
84     tmp[++top] = x;
85     travel(x->son[0]);
86     travel(x->son[1]);
87 }
88 void rebuild(Node **x) {
89     top = 0, travel(*x);
90     *x = build(1, top);
91 }
92
93 Node **insert(Node * &x, Node *y) {
94     if (x == NULL) {
95         x = y; return NULL;
96     } else {
97         Node **node = insert(x->son[cmp[x->d](x, y)], y);
98         x->maintain();
99         if (!x->is_balanced()) node = &x;
100        return node;
101    }
102 }
103
104 void insert(Point pos, int id) {
105     Node *cur = tot++;
106     cur->init(pos, id);
107     Node **node = insert(root, cur);
108     if (node != NULL) rebuild(node);
109 }
110
111 pair<long long, int> ask_nearest(Node *x, Point pos, pair<long long, int> ans) {
112     if (!x) return ans;
113     chkmn(ans, make_pair(disof(x->pos, pos), x->id));
114     if (x->son[0] && x->son[1]) {
115         long long dis[2] = {x->son[0]->get_dis(pos), x->son[1]->get_dis(pos)};
116         int id[2] = {0, 1};
117         if (dis[id[0]] > dis[id[1]]) swap(id[0], id[1]);
118         for (int i = 0; i < 2; i++)
119             if (dis[id[i]] <= ans.first)
120                 chkmn(ans, ask_nearest(x->son[id[i]], pos, ans));
121     } else {
122         if (x->son[0]) chkmn(ans, ask_id(x->son[0], pos, ans));
123         if (x->son[1]) chkmn(ans, ask_id(x->son[1], pos, ans));
124     }
125     return ans;
126 }
127
128 int ask(Point pos) {
129     return ask_nearest(root, pos, make_pair(inf, 0)).second;
130 }
131 }
```

5.17 DLX (Nightfall)

```

1 struct node{
2     node *left,*right,*up,*down,*col; int row,cnt;
3 }*head,*col[MAXC],Node[MAXNODE],*ans[MAXNODE];
4 int totNode, ansNode;
5 void insert(const std::vector<int> &V,int rounum){
6     std::vector<node*> N;
7     for(int i=0;i<V.size();++i){
8         node* now=Node+(totNode++); now->row=rounum;
9         now->col=now->up=col[V[i]], now->down=col[V[i]]->down;
10        now->up->down=now, now->down->up=now;
11        now->col->cnt++; N.push_back(now); }
12     for(int i=0;i<V.size();++i)
13         N[i]->right=N[(i+1)%V.size()], N[i]->left=N[(i-1+V.size())%V.size()];
14 }
15 void Remove(node *x){
16     x->left->right=x->right, x->right->left=x->left;
17     for(node *i=x->down;i!=x;i=i->down)
18         for(node *j=i->right;j!=i;j=j->right)
19             j->up->down=j->down, j->down->up=j->up, --(j->col->cnt);
20 }
21 void Resume(node *x){
22     for(node *i=x->up;i!=x;i=i->up)
23         for(node *j=i->left;j!=i;j=j->left)
24             j->up->down=j->down->up=j, ++(j->col->cnt);
25     x->left->right=x, x->right->left=x;
26 }
27 bool search(int tot){
28     if(head->right==head) return ansNode = tot, true;
29     node *choose=NULL;
30     for(node *i=head->right;i!=head;i=i->right){
31         if(choose==NULL||choose->cnt>i->cnt) choose=i;
32         if(choose->cnt<2) break; }
33     Remove(choose);
34     for(node *i=choose->down;i!=choose;i=i->down){
35         for(node *j=i->right;j!=i;j=j->right) Remove(j->col);
36         ans[tot]=i;
37         if(search(tot+1)) return true;
38         ans[tot]=NULL;
39         for(node *j=i->left;j!=i;j=j->left) Resume(j->col); }
40     Resume(choose); return false;
41 }
42 void prepare(int totC){
43     head=Node+totC;
44     for(int i=0;i<totC;++i) col[i]=Node+i;
45     totNode=totC+1; ansNode = 0;
46     for(int i=0;i<=totC;++i){
47         (Node+i)->right=Node+(i+1)%(totC+1);
48         (Node+i)->left=Node+(i+totC)%(totC+1);
49         (Node+i)->up=(Node+i)->down=Node+i;
50         (Node+i)->cnt=0; }
51 }
52 prepare(C); for (i (rows)) insert({col_id}, C); search(0);

```

5.18 数据结构知识 (gy)

Young Tableau

如果 $a_{i,j}$ 没有元素，则 $a_{i+1,j}$ 没有元素，否则要么 $a_{i+1,j}$ 没有元素，要么 $a_{i+1,j} > a_{i,j}$ ，对 $a_{i,j+1}$ 有同样要求，则称该矩阵为杨氏矩阵。

记 $hook_{i,j}$ 表示该元素上面和右边的元素个数之和。

对于 1 到 n 组成的杨氏矩阵，固定形状的杨氏矩阵个数为 $cnt = \frac{n!}{\prod_{ij} (hook_{i,j} + 1)}$ 。

对于 1 到 n 组成的杨氏矩阵，所有形状的杨氏矩阵个数满足： $F(1) = 1$, $F(2) = 2$, $F(n) = F(n-1) + (n-1)F(n-2)$, ($n > 2$)。

Others

6.1 Config (gy)

bash

```
1 export CPPFLAGS='-std=c++11 -Wall -Wextra -Wconversion -fsanitize=undefined -fsanitize=address'
```

vim

```
1 se et ts=4 sw=4 sts=4 nu sc sm lbr is hls mouse=a cin fdm=syntax mp=make\ %<
2 sy on
3 ino <tab> <c-n>
4 ino <s-tab> <tab>
5 au bufwinenter * winc L
6
7 nm <f6> ggVG"+y
8
9 nm <F7> :w<cr>:make<cr>
10 nm <F8> :!.%<<cr>
11 nm <F9> :!.%< < in<cr>
12 nm <S-F9> :!(time ./%< < in &> out) &>> out<cr>:sp out<cr>
13
14 sy keyword Type point line circle info data
```

6.2 模拟退火 (ct)

```
1 db ans_x, fans;
2 inline double rand01() {return rand() / 2147483647.0;}
3 inline double randp() {return (rand() & 1 ? 1 : -1) * rand01();}
4 inline double f(double x)
5 {
6     /*
7      * write your function here.
8     */
9     if (maxx < fans) {fans = maxx; ans_x = x;}
10    return maxx;
11 }
12 int main()
13 {
14     srand(time(NULL) + clock());
15     db x = 0, fnow = f(x);
16     fans = 1e30;
17     for (db T = 1e4; T > 1e-4; T *= 0.997)
18     {
19         db nx = x + randp() * T, fnext = f(nx);
20         db delta = fnext - fnow;
```

```

21     if (delta < 1e-9 || exp(-delta / T) > rand01())
22     {
23         x = nx;
24         fnow = fnext;
25     }
26     return 0;
27 }
```

6.3 Simpson 积分 (gy)

```

1 number f(number x) {
2     return /* Take circle area as example */ std::sqrt(1 - x * x) * 2;
3 }
4 number simpson(number a, number b) {
5     number c = (a + b) / 2;
6     return (f(a) + f(b) + 4 * f(c)) * (b - a) / 6;
7 }
8 number integral(number a, number b, number eps) {
9     number c = (a + b) / 2;
10    number mid = simpson(a, b), l = simpson(a, c), r = simpson(c, b);
11    if (std::abs(l + r - mid) <= 15 * eps)
12        return l + r + (l + r - mid) / 15;
13    else
14        return integral(a, c, eps / 2) + integral(c, b, eps / 2);
15 }
```

6.4 Zeller Congruence (gy)

0 → Sunday, …, 6 → Saturday

```

1 int day_in_week(int year, int month, int day) {
2     if (month == 1 || month == 2)
3         month += 12, year--;
4     int c = year / 100, y = year % 100, m = month, d = day;
5     int ret = (y + y / 4 + c / 4 + 5 * c + 13 * (m + 1) / 5 + d + 6) % 7;
6     return ret >= 0 ? ret : ret + 7;
7 }
```

6.5 博弈论模型 (gy)

Nim

Name	Pick Limit	$SG(n)$
Nim	$[1, a_i]$	n
Nim (powers)	$\{k^m m \geq 0\}$	$\begin{cases} 2 & n \bmod (k+1) = k \\ n \bmod (k+1) \bmod 2 & \text{otherwise} \end{cases}$
Nim (no greater than half)	$[1, \lfloor \frac{a_i}{2} \rfloor]$	$\begin{cases} \frac{n}{2} & 2 \mid n \\ SG(\frac{n-1}{2}) & 2 \nmid n \end{cases}$
Nim (always greater than half)	$[\lceil \frac{a_i}{2} \rceil, a_i]$	$\begin{cases} 0 & n = 0 \\ \lfloor \log_2 n \rfloor + 1 & n > 0 \end{cases}$

Nim (proper divisor)	$\{x x < n \wedge x \mid a_i\}$	$\begin{cases} 0 & n = 0 \\ \max_x 2^x \mid n & \text{otherwise} \end{cases}$
Nim (divisor)	$\{x x \mid a_i\}$	$\begin{cases} 0 & n = 0 \\ 1 + \max_x 2^x \mid n & \text{otherwise} \end{cases}$
Nim (fixed)	a finite set S $S \cup \{a_i\}$	periodic SG_1 $SG_1(n) + 1$
Moore's Nim anti Moore's Nim	$[1, a_i]$ from $[1, k]$ piles	win - $\wedge \text{cnt}_{a_i}(k\text{-th digit of } (a_i)_2 \text{ is } 1) = 0$ The same, opposite if $\wedge a_i = 1$
Staircase Nim	Move from a_i to $a_i - 1$	win - $\bigoplus SG(a_{2i+1}) > 0$
Lasker's Nim	$[1, a_i]$ or split a_i	$\begin{cases} n & n \bmod 4 = 1, 2 \\ n + 1 & n \bmod 4 = 3 \\ n - 1 & n \bmod 4 = 0 \end{cases}$

Kayles

一行 n 个石子，每次可以取走一个或相邻两个还未被取走的， SG 函数从 $n = 72$ 开始以 12 为循环节

Dawson's chess

一行 n 个石子，每次可以取走相邻三个未被取走的石子中中间的那个， SG 函数从 $n = 52$ 开始以 34 为循环节

Ferguson game

m 和 n 两堆石子，每次可以取光一堆石子并将另一堆分成非空的两堆
 $\text{win} - 2 \mid m \vee 2 \mid n$

Mock Turtles

一行 n 枚硬币，每次可以翻 1, 2, 3 个硬币，但最右的那个必须正面向上
 $SG(n) = 2n + [2 \mid \text{popcount}(n)]$

Ruler

一行 n 枚硬币，每次可以翻连续的任意数量的硬币，但最右的那个必须正面向上
 $SG(n) = \text{lowbit}(n)$

Wythoff's game

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

先手胜当且仅当石子数满足：

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

先手胜对应的石子数构成两个序列：

$$a_n = \lfloor n \times \phi \rfloor, b_n = \lfloor n \times \phi^2 \rfloor$$

Fibonacci nim

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

先手胜当且仅当石子数为斐波那契数

anti-SG

决策集合为空的游戏者胜

先手胜当且仅当满足以下任一条件

- 所有单一游戏的 SG 值都 < 2 且游戏的 SG 值为 0
- 至少有一个单一游戏的 SG 值 ≥ 2 且游戏的 SG 值不为 0

6.6 C++ Template (Durandal,gy,ct)

运算符优先级 (gy)

Precedence	Associativity	Operator
1	L	::
2	L	Suffix++ Suffix-- Functional cast Function call Subscript[] Member access . ->
3	R	Prefix++ Prefix-- Unary+ Unary- ! ~ (type) *a &a sizeof co_await new new[] delete delete[]
4	L	Pointer-to-member .* ->*
5	L	* / %
6	L	Binary+ Binary-
7	L	<< >>
8	L	<=> (C++20)
9	L	< <= > =
10	L	== !=
11	L	Bitwise&
12	L	Bitwise^
13	L	Bitwise
14	L	Logical&&
15	L	Logical
16	R	Ternary conditional ?: throw co_yield = += -= *= /= %= <=>= &= ^= =
17	L	,

STL 释放内存 (Durandal)

```

1 template <typename T>
2 __inline void clear(T &container) {
3     container.clear();

```

```

4     T(container).swap(container);
5 }
```

开栈 (Durandal)

```

1 register char *_sp __asm__("rsp");
2 int main() {
3     const int size = 400 << 20; // 400 MB
4     static char *sys, *mine(new char[size] + size - 4096);
5     sys = _sp; _sp = mine;
6     _main(); // main method
7     _sp = sys;
8     return 0;
9 }
```

O3 (gy)

```

1 #pragma GCC optimize(3)
2 __attribute__((optimize("-O3"))) int main() { return 0; }
```

读入优化 (ct)

```

1 char S[1 << 20], *T = S;
2 inline int F()
3 {
4     char ch; int cnt = 0;
5     while (ch = *T++, ch < '0' || ch > '9') ;
6     cnt = ch - '0';
7     while (ch = *T++, ch >= '0' && ch <= '9') cnt = cnt * 10 + ch - '0';
8     return cnt;
9 }
10 fread(S, 1, 1 << 20, stdin);
```

6.7 Python Template (gy)

文件操作

```

1 input_file = open("hello.in", "r")
2 output_file = open("hello.out", "w")
3 [a, b] = map(int, input_file.readline().split(" "))
4 output_file.write("{}\n".format(a + b))
5 input_file.close()
6 output_file.close()
```

6.8 Java Template (gy)

读入优化

```

1 import java.io.*;
2 import java.util.*;
3
4 public class InputOptimize {
5     private static BufferedReader reader;
6     private static StringTokenizer tokenizer;
7     private static String next() {
```

```

8   try {
9     while (tokenizer == null || !tokenizer.hasMoreTokens())
10       tokenizer = new StringTokenizer(reader.readLine());
11   } catch (IOException e) {
12     // do nothing
13   }
14   return tokenizer.nextToken();
15 }
16 private static int nextInt() {
17   return Integer.parseInt(next());
18 }
19 public static void main(String[] args) {
20   reader = new BufferedReader(new InputStreamReader(System.in));
21 }
22 }
```

BigInteger BigInteger

```

1 import java.math.*;
2
3 public class BigTemplate {
4   // BigInteger & BigDecimal
5   private static void bigDecimal() {
6     BigDecimal a = BigDecimal.valueOf(1.0);
7     BigDecimal b = a.setScale(50, RoundingMode.HALF_EVEN);
8     BigDecimal c = b.abs();
9     // if scale omitted, b.scale is used
10    BigDecimal d = c.divide(b, 50, RoundingMode.HALF_EVEN);
11    // since Java 9
12    BigDecimal e = d.sqrt(new MathContext(50, RoundingMode.HALF_EVEN));
13    BigDecimal x = new BigDecimal(BigInteger.ZERO);
14    BigInteger y = BigDecimal.ZERO.toBigInteger(); // RoundingMode.DOWN
15    y = BigDecimal.ZERO.setScale(0, RoundingMode.HALF_EVEN).unscaledValue();
16  }
17
18  // sqrt for Java 8
19  // can solve scale=100 for 10000 times in about 1 second
20  private static BigDecimal sqrt(BigDecimal a, int scale) {
21    if (a.compareTo(BigDecimal.ZERO) < 0)
22      return BigDecimal.ZERO.setScale(scale, RoundingMode.HALF_EVEN);
23    int length = a.precision() - a.scale();
24    BigDecimal ret = new BigDecimal(BigInteger.ONE, -length / 2);
25    for (int i = 1; i <= Integer.highestOneBit(scale) + 10; i++)
26      ret = ret.add(a.divide(ret, scale, RoundingMode.HALF_EVEN)).divide(BigDecimal.valueOf(2),
27        scale, RoundingMode.HALF_EVEN);
28    return ret;
29  }
30
31  // can solve a=2^10000 for 100000 times in about 1 second
32  private static BigInteger sqrt(BigInteger a) {
33    int length = a.bitLength() - 1;
34    BigInteger l = BigInteger.ZERO.setBit(length / 2), r = BigInteger.ZERO.setBit(length / 2);
35    while (!l.equals(r)) {
36      BigInteger m = l.add(r).shiftRight(1);
37      if (m.multiply(m).compareTo(a) < 0)
38        l = m.add(BigInteger.ONE);
39      else
40        r = m;
41    }
42    return l;
43 }
```

```

42 }
43
44 private static class BigFraction {
45     private BigInteger a, b;
46
47     BigFraction(BigInteger a, BigInteger b) {
48         BigInteger gcd = a.gcd(b);
49         this.a = a.divide(gcd);
50         this.b = b.divide(gcd);
51     }
52
53     BigFraction add(BigFraction o) {
54         BigInteger gcd = b.gcd(o.b);
55         BigInteger tempProduct = b.divide(gcd).multiply(o.b.divide(gcd));
56         BigInteger ansA = a.multiply(o.b.divide(gcd)).add(o.a.multiply(b.divide(gcd)));
57         BigInteger gcd2 = ansA.gcd(gcd);
58         ansA = ansA.divide(gcd2);
59         gcd2 = gcd.divide(gcd2);
60         return new BigFraction(ansA, gcd2.multiply(tempProduct));
61     }
62
63     BigFraction subtract(BigFraction o) {
64         BigInteger gcd = b.gcd(o.b);
65         BigInteger tempProduct = b.divide(gcd).multiply(o.b.divide(gcd));
66         BigInteger ansA = a.multiply(o.b.divide(gcd)).subtract(o.a.multiply(b.divide(gcd)));
67         BigInteger gcd2 = ansA.gcd(gcd);
68         ansA = ansA.divide(gcd2);
69         gcd = gcd.divide(gcd2);
70         return new BigFraction(ansA, gcd2.multiply(tempProduct));
71     }
72
73     BigFraction multiply(BigFraction o) {
74         BigInteger gcd1 = a.gcd(o.b);
75         BigInteger gcd2 = b.gcd(o.a);
76         return new BigFraction(a.divide(gcd1).multiply(o.a.divide(gcd2)),
77             ↵ b.divide(gcd2).multiply(o.b.divide(gcd1)));
78     }
79
80     @Override
81     public String toString() {
82         return a + "/" + b;
83     }
84 }
```

STL

```

1 import java.util.*;
2
3 public class STLTemplate {
4     private static void stl() {
5         List<Integer> list = new ArrayList<>();
6         List[] lists = new List[100]; lists[0] = new ArrayList<Integer>();
7         list.remove(list.get(1)); list.remove(list.size() - 1); list.clear();
8         Queue<Integer> queue = new LinkedList<>();
9         Queue<Integer> priorityQueue = new PriorityQueue<>();
10        queue.peek(); queue.poll();
11        Deque<Integer> deque = new ArrayDeque<>();
12        deque.peekFirst(); deque.peekLast(); deque.pollFirst();
13        TreeSet<Integer> set = new TreeSet<>();
```

```

14     TreeSet<Integer> anotherSet = new TreeSet<>(Comparator.reverseOrder());
15     set.ceiling(1); set.floor(1); set.lower(1); set.higher(1); set.contains(1);
16     HashSet<Integer> hashSet = new HashSet<>();
17     HashMap<String, Integer> map = new HashMap<>();
18     TreeMap<String, Integer> treeMap = new TreeMap<>();
19     map.put("", 1); map.get("");
20     map.forEach((string, integer) -> System.out.println(string + integer));
21     Arrays.sort(new int[10]);
22     Arrays.sort(new Integer[10], (a, b) -> b.compareTo(a));
23     Arrays.sort(new Integer[10], Comparator.comparingInt((a) -> (int) a).reversed());
24     long a = 1_000_000_000_000_000L;
25     int b = Integer.MAX_VALUE;
26     int c = 'a';
27 }
28 }
```

6.9 积分表 (integral-table.com)

$$\begin{aligned}
\int x^n dx &= \frac{1}{n+1} x^{n+1}, \quad n \neq -1 \\
\int \frac{1}{x} dx &= \ln|x| \\
\int u dv &= uv - \int v du \\
\int \frac{1}{ax+b} dx &= \frac{1}{a} \ln|ax+b| \\
\int \frac{1}{(x+a)^2} dx &= -\frac{1}{x+a} \\
\int (x+a)^n dx &= \frac{(x+a)^{n+1}}{n+1}, \quad n \neq -1 \\
\int x(x+a)^n dx &= \frac{(x+a)^{n+1}((n+1)x-a)}{(n+1)(n+2)} \\
\int \frac{1}{1+x^2} dx &= \tan^{-1} x \\
\int \frac{1}{a^2+x^2} dx &= \frac{1}{a} \tan^{-1} \frac{x}{a} \\
\int \frac{x}{a^2+x^2} dx &= \frac{1}{2} \ln|a^2+x^2| \\
\int \frac{x^2}{a^2+x^2} dx &= x - a \tan^{-1} \frac{x}{a} \\
\int \frac{x^3}{a^2+x^2} dx &= \frac{1}{2} x^2 - \frac{1}{2} a^2 \ln|a^2+x^2| \\
\int \frac{1}{ax^2+bx+c} dx &= \frac{2}{\sqrt{4ac-b^2}} \tan^{-1} \frac{2ax+b}{\sqrt{4ac-b^2}} \\
\int \frac{1}{(x+a)(x+b)} dx &= \frac{1}{b-a} \ln \frac{a+x}{b+x}, \quad a \neq b \\
\int \frac{x}{(x+a)^2} dx &= \frac{a}{a+x} + \ln|x+a| \\
\int \frac{x}{ax^2+bx+c} dx &= \frac{1}{2a} \ln|ax^2+bx+c| - \frac{b}{a\sqrt{4ac-b^2}} \tan^{-1} \frac{2ax+b}{\sqrt{4ac-b^2}} \\
\int \sqrt{x-a} dx &= \frac{2}{3}(x-a)^{3/2} \\
\int \frac{1}{\sqrt{x \pm a}} dx &= 2\sqrt{x \pm a} \\
\int \frac{1}{\sqrt{a-x}} dx &= -2\sqrt{a-x} \\
\int x\sqrt{x-a} dx &= \begin{cases} \frac{2a}{3}(x-a)^{3/2} + \frac{2}{5}(x-a)^{5/2}, & \text{or} \\ \frac{2}{5}x(x-a)^{3/2} - \frac{4}{15}(x-a)^{5/2}, & \text{or} \\ \frac{2}{15}(2a+3x)(x-a)^{3/2} \end{cases} \\
\int \frac{x^2}{\sqrt{x^2 \pm a^2}} dx &= \frac{1}{2}x\sqrt{x^2 \pm a^2} \mp \frac{1}{2}a^2 \ln|x + \sqrt{x^2 \pm a^2}| \\
\int \sqrt{ax+b} dx &= \left(\frac{2b}{3a} + \frac{2x}{3} \right) \sqrt{ax+b} \\
\int (ax+b)^{3/2} dx &= \frac{2}{5a}(ax+b)^{5/2} \\
\int \frac{x}{\sqrt{x \pm a}} dx &= \frac{2}{3}(x \mp 2a)\sqrt{x \pm a} \\
\int \sqrt{\frac{x}{a-x}} dx &= -\sqrt{x(a-x)} - a \tan^{-1} \frac{\sqrt{x(a-x)}}{x-a} \\
\int \sqrt{\frac{x}{a+x}} dx &= \sqrt{x(a+x)} - a \ln(\sqrt{x} + \sqrt{x+a}) \\
\int x\sqrt{ax+b} dx &= \frac{2}{15a^2}(-2b^2 + abx + 3a^2x^2)\sqrt{ax+b}
\end{aligned}$$

$$\begin{aligned}
& \int \sqrt{ax^2 + bx + c} \, dx = \\
& \frac{b+2ax}{4a} \sqrt{ax^2 + bx + c} + \frac{4ac - b^2}{8a^{3/2}} \ln \left| 2ax + b + 2\sqrt{a(ax^2 + bx + c)} \right| \\
& \int x \sqrt{ax^2 + bx + c} \, dx = \\
& \frac{1}{48a^{5/2}} \left(2\sqrt{a} \sqrt{ax^2 + bx + c} (-3b^2 + 2abx + 8a(c + ax^2)) \right. \\
& \quad \left. + 3(b^3 - 4abc) \ln \left| b + 2ax + 2\sqrt{a} \sqrt{ax^2 + bx + c} \right| \right) \\
& \int \frac{1}{\sqrt{ax^2 + bx + c}} \, dx = \frac{1}{\sqrt{a}} \ln \left| 2ax + b + 2\sqrt{a(ax^2 + bx + c)} \right| \\
& \int \frac{x}{\sqrt{ax^2 + bx + c}} \, dx = \\
& \frac{1}{a} \sqrt{ax^2 + bx + c} - \frac{b}{2a^{3/2}} \ln \left| 2ax + b + 2\sqrt{a(ax^2 + bx + c)} \right| \\
& \int \frac{dx}{(a^2 + x^2)^{3/2}} = \frac{x}{a^2 \sqrt{a^2 + x^2}} \\
& \int \sin ax \, dx = -\frac{1}{a} \cos ax \\
& \int \sin^2 ax \, dx = \frac{x}{2} - \frac{\sin 2ax}{4a} \\
& \int \sin^3 ax \, dx = -\frac{3 \cos ax}{4a} + \frac{\cos 3ax}{12a} \\
& \int \cos ax \, dx = \frac{1}{a} \sin ax \\
& \int \cos^2 ax \, dx = \frac{x}{2} + \frac{\sin 2ax}{4a} \\
& \int \cos^3 ax \, dx = \frac{3 \sin ax}{4a} + \frac{\sin 3ax}{12a} \\
& \int \cos x \sin x \, dx = \frac{1}{2} \sin^2 x + c_1 = -\frac{1}{2} \cos^2 x + c_2 = -\frac{1}{4} \cos 2x + c_3 \\
& \int \cos ax \sin bx \, dx = \frac{\cos[(a-b)x]}{2(a-b)} - \frac{\cos[(a+b)x]}{2(a+b)}, a \neq b \\
& \int \sin^2 ax \cos bx \, dx = -\frac{\sin[(2a-b)x]}{4(2a-b)} + \frac{\sin bx}{2b} - \frac{\sin[(2a+b)x]}{4(2a+b)} \\
& \int \sin^2 x \cos x \, dx = \frac{1}{3} \sin^3 x \\
& \int \cos^2 ax \sin bx \, dx = \frac{\cos[(2a-b)x]}{4(2a-b)} - \frac{\cos bx}{2b} - \frac{\cos[(2a+b)x]}{4(2a+b)} \\
& \int \cos^2 ax \sin ax \, dx = -\frac{1}{3a} \cos^3 ax \\
& \int \sin^2 ax \cos^2 bx \, dx = \\
& \frac{x}{4} - \frac{\sin 2ax}{8a} - \frac{\sin[2(a-b)x]}{16(a-b)} + \frac{\sin 2bx}{8b} - \frac{\sin[2(a+b)x]}{16(a+b)}
\end{aligned}$$

$$\begin{aligned}
& \int \sin^2 ax \cos^2 ax \, dx = \frac{x}{8} - \frac{\sin 4ax}{32a} \\
& \int \tan ax \, dx = -\frac{1}{a} \ln |\cos ax| \\
& \int \tan^2 ax \, dx = -x + \frac{1}{a} \tan ax \\
& \int \tan^3 ax \, dx = \frac{1}{a} \ln |\cos ax| + \frac{1}{2a} \sec^2 ax \\
& \int \sec x \, dx = \ln |\sec x + \tan x| = 2 \tanh^{-1} \left(\tan \frac{x}{2} \right) \\
& \int \sec^2 ax \, dx = \frac{1}{a} \tan ax \\
& \int \sec^3 x \, dx = \frac{1}{2} \sec x \tan x + \frac{1}{2} \ln |\sec x + \tan x| \\
& \int \sec x \tan x \, dx = \sec x \\
& \int \sec^2 x \tan x \, dx = \frac{1}{2} \sec^2 x \\
& \int \sec^n x \tan x \, dx = \frac{1}{n} \sec^n x, n \neq 0 \\
& \int \csc x \, dx = \ln \left| \tan \frac{x}{2} \right| = \ln |\csc x - \cot x| + C \\
& \int \csc^2 ax \, dx = -\frac{1}{a} \cot ax \\
& \int \csc^3 x \, dx = -\frac{1}{2} \cot x \csc x + \frac{1}{2} \ln |\csc x - \cot x| \\
& \int \csc^n x \cot x \, dx = -\frac{1}{n} \csc^n x, n \neq 0 \\
& \int \sec x \csc x \, dx = \ln |\tan x| \\
& \int x \cos x \, dx = \cos x + x \sin x \\
& \int x \cos ax \, dx = \frac{1}{a^2} \cos ax + \frac{x}{a} \sin ax \\
& \int x^2 \cos x \, dx = 2x \cos x + (x^2 - 2) \sin x \\
& \int x^2 \cos ax \, dx = \frac{2x \cos ax}{a^2} + \frac{a^2 x^2 - 2}{a^3} \sin ax \\
& \int x \sin x \, dx = -x \cos x + \sin x \\
& \int x \sin ax \, dx = -\frac{x \cos ax}{a} + \frac{\sin ax}{a^2} \\
& \int x^2 \sin x \, dx = (2 - x^2) \cos x + 2x \sin x \\
& \int x^2 \sin ax \, dx = \frac{2 - a^2 x^2}{a^3} \cos ax + \frac{2x \sin ax}{a^2} \\
& \int x \cos^2 x \, dx = \frac{x^2}{4} + \frac{1}{8} \cos 2x + \frac{1}{4} x \sin 2x \\
& \int x \sin^2 x \, dx = \frac{x^2}{4} - \frac{1}{8} \cos 2x - \frac{1}{4} x \sin 2x
\end{aligned}$$

6.10 环境测试 (gy)

- C++ 版本
- O2
- pb.ds
- 栈空间 and 开栈
- assert
- long double and %Lf
- __int128 and __float128
- map 速度
- unordered_map 速度
- ntt 速度
- fft 速度
- Java 版本
- assert
- 栈空间
- Python 支持 (python/python3/pypy/pypy3)
- vim 复制
- 返回结果速度
- CE 罚时
- AC 后提交罚时
- 内存限制 and MLE
- PE
- MLE and TLE
- OLE
- 代码长度限制

Additional Documents

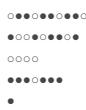
A.1 超自然数和不平等博弈



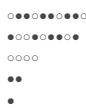
This page is (maybe un-) intentionally (almost) blank.

surreal number应对不平等博弈**实数、超实数和博弈游戏：数学的结构之美****(一) 一个博弈游戏**

让我们来玩一个游戏。下面有五行石子，白色的石子都是我的，黑色的石子都是你的。我们轮流拿走一个自己的石子，并且规定如果一个石子被拿走了，它后面的所有石子都要被扔掉。谁先没有拿的了，谁就输了。



比如说，如果你先走的话，你可以把第四行的第三个石子拿走，按规定第四行将会只剩下前面两个石子：



现在轮到我走了。我可以拿走第二行倒数第二个石子，于是整个棋局变成了这样：



现在，假如说你拿走了第二行中的第一个石子（于是第二行就没了），那么我就赢定了。我可以拿走第一行中的第一个石子，从而让整个棋局只剩下后面三行：



这三行中有四个白石子，有三个黑石子，并且每一行都是同种石子。于是整个局面完全变成了一个拼石子个数的游戏，我只需要一个一个地拿走白色石子，你必然将会率先无路可走。受此启发，我们自然地想到一种刻画棋局的方式：把每个白色石子记作 $+1$ ，把每个黑色石子记作 -1 。于是 $\circ\circ\circ\circ + \bullet\bullet + \bullet = 4 - 2 - 1 = 1$ ，结果是一个正数，这就表明该局面下我将必胜，即使此时轮到我先走。

你会发现上面的说法很有道理。毕竟白色的石子越多，对我越有利，给我带来的效用为正；而黑色的石子会减小我获胜的希望，当然应该给它赋上一个负的值。四白三黑算出来的结果为正1，直观意义就是我能以一步的优势获胜。如果棋局是这样：



那么 $\circ\circ + \bullet\bullet\bullet = 2 - 4 = -2$ ，是一个负数，这就意味着不管谁先走，你都能必胜，因为你能比我多走两步。我们不妨把一个棋局对应的数叫做棋局的“特征值”。特征值为正，就表明不管谁先走我都能必胜；特征值为负，就表明不管谁先走你都能必胜；而特征值的绝对值，则直观地量化了胜负的悬殊。现在，考虑下面这个棋局：



那么 $\circ\circ + \bullet\bullet = 2 - 2 = 0$ ，这表明此时的情形介于“我必胜”和“你必胜”之间。事实上也是这样——如果我先走你后走，你就赢定了；如果你先走我后走，我就赢定了。这是因为，这个棋局的特征值为0，双方能够走的步数相同，当然谁后走谁就赢定了。

真正有趣的事情出现了。考虑下面的棋局：

◦•

它的特征值应该是多少呢？容易看出，它的特征值是一个正数，因为不管谁先走，显然我都能赢。同时，它的特征值也应该比 1 小，因为只有一个◦要比◦•赢得更爽一些。事实上，单看◦•黑方确实无论如何都会输，但在某些场合下，◦后面的这个•能让黑方喘上一口气。比如下面这个棋局：

◦•

•

如果我先走，显然你必胜无疑。如果你先走呢？先走的人获胜的难度更大，你要好好想想策略。你可以拿走那个单独成行的•，但当我拿走◦之后，你就得眼睁睁地看着自己剩下的那个•被一并收走。此时，你或许会意识到，你本来还能多走一步的，可惜这一步被浪费掉了。因此，你更好的做法就是，一上来先拿走◦后面的•。运用这种策略，显然你也会必胜无疑。可见，不管是先走还是后走，整个棋局对于你来说都是必胜的，从而有 $◦• + • = ◦ - 1 < 0$ ，这再次说明了◦是一个小于 1 的数。那么，◦是否等于 $1/2$ 呢？答案是肯定的，考虑下面这个棋局：

◦•

◦•

•

如果我先走，本质上不同的走法只有一种，并且局面将会立即变成刚才那种你必胜的情形，因而你将必胜。如果是先走呢？拿走那个单独成行的•会让你提前锁定胜局，更好的选择则是像刚才一样，先拿走某个◦后面的•，为自己多赢得一步。现在，石子只剩下了◦•、◦、•这么三行。那么，我应该拿走哪一个◦呢？拿走那个单独成行的◦会让局面再次变回刚才那种你必胜的情形，更好的选择则是拿走后面有•的◦，这样我便能让你损失一步。掌握了这个技巧后，我就能做到必胜了。综上所述，整个棋局是一个谁后走谁就赢定了的局面。于是 $◦• + ◦• + • = 0$ ，也就是 $◦• + ◦ - 1 = 0$ ，可以解得◦等于 $1/2$ 。也就是说，在◦中，我将以半步的优势获胜。对应地，◦就等于 $-1/2$ ，此时你将以半步的优势获胜。

注意，目前并没有任何理论告诉我们，这么加减是合理的。不过我们却发现，这么加减出来的结果真的是对的。比如说， $◦• + ◦• + ◦ - • = 1/2 + 1/2 + 1/2 - 1 = 1/2$ ，而棋局

◦•

◦•

•

•

对我来说真的就是必胜的局面！

这背后一定有一个更深层次的原因：棋局之间的加减和数与数之间的加减一定存在着某些共通的地方。也就是说，为了解释“为什么棋局的加法和数的加法如此之像”，我们需要证明，两者具有完全相同的代数结构。我们需要建立一个从棋局到实数的映射法则，然后说明全体棋局（或者全体棋局的一部分）与全体实数（或者全体实数的一部分）是同构的。

正如 Poincaré 所说，诗歌的艺术在于给相同的东西取不同的名字，数学的艺术在于给不同的东西取相同的名字。两个看似毫不相关的东西竟然是同构的，这在数学中最令人激动的事情之一。

(二) 熟悉而又陌生的算术

为了说明棋局就是算术，我们首先要定义，什么是算术。我们需要站在系统之外，对算术的本质做一个描述。算术，就是对一些数做加减乘除的运算。这似乎对我们完全没有帮助——什么是数，什么是加减乘除？其实，数，就是一些具有大小关系的元素，这些元素可以按照它们的大小关系串成一根链条。这意味着，首先，任意两个数都是可以比较大小的，并且对于两个不同的数 x 和 y 来说， x 要么大于 y ，要么小于 y 。然后，大小关系必须具有传递性，如果 x 小于 y ， y 又小于 z ，那么 x 一定小于 z 。形式化地说，我们要求元素之间存在某种暂且记作 \leq 的关系，使得它们满足：

1. 完全性：对于任意的 x 和 y ， $x \leq y$ 和 $y \leq x$ 当中至少有一个成立。
2. 反对称性：对于任意的 x 和 y ，如果 $x \leq y$ 和 $y \leq x$ 同时成立，那么 x 和 y 一定是同一个元素。（今后我们用符号 $x = y$ 表示 x 和 y 是同一个元素，注意这个新符号和 \leq 的区别：前者是一个真实的符号，它用来表达“是同一个元素”这个概念；后者则是我们假想的一种抽象符号，它不见得有什么具体的意义。）
3. 传递性：对于任意的 x 、 y 和 z ，如果 $x \leq y$ 和 $y \leq z$ 同时成立，那么一定有 $x \leq z$ 。

这样的话，所有的元素都被 \leq 穿成了一条绳子。我们就说，这些元素构成了一个“全序关系”。

我们还需要元素之间的“加法”和“乘法”满足一系列的性质：

4. 对加法封闭：对于任意的 x 和 y ， $x + y$ 也仍然是某一个元素
5. 对乘法封闭：对于任意的 x 和 y ， $x \cdot y$ 也仍然是某一个元素
6. 加法交换律：对于任意的 x 和 y ， $x + y = y + x$
7. 乘法交换律：对于任意的 x 和 y ， $x \cdot y = y \cdot x$
8. 加法结合律：对于任意的 x 、 y 和 z ， $(x + y) + z = x + (y + z)$
9. 乘法结合律：对于任意的 x 、 y 和 z ， $(x \cdot y) \cdot z = x \cdot (y \cdot z)$
10. 乘法对加法的分配律：对于任意的 x 、 y 和 z ， $x \cdot (y + z) = x \cdot y + x \cdot z$
11. 存在加法单位：存在某一个特殊的元素，通常记作 0 ，使得对于任意的 x ，都有 $x + 0 = x$

12. 存在乘法单位：存在某一个特殊的元素，通常记作 1，使得对于任意的 x ，都有 $x \cdot 1 = x$
13. 存在加法逆元：对于任意的 x ，总能找到某一个特殊的元素，通常记作 $-x$ ，使得 $x + (-x) = 0$
14. 存在乘法逆元：对于任意不为 0 的 x ，总能找到某一个特殊的元素，通常记作 x^{-1} ，使得 $x \cdot x^{-1} = 1$
15. 对于任意的 x 、 y 和 z ，如果 $x \leq y$ ，那么一定有 $x + z \leq y + z$
16. 对于任意的 x 、 y ，如果 $0 \leq x$ 和 $0 \leq y$ 同时成立，那么一定有 $0 \leq x \cdot y$

有几点需要注意。我们虽然只说了加法单位满足 $x + 0$ 等于 x ，但其实由于加法交换律， $0 + x$ 也等于 x 。乘法单位也是如此。我们说“存在加法单位”，而没说“存在唯一的加法单位”，这是因为加法单位最多只能有一个，如果存在，必定唯一。假如 a 和 b 是两个加法单位的话，你会发现因为 a 是加法单位，所以 $a + b$ 等于 b ，又因为 b 是加法单位，所以 $a + b$ 等于 a ，因而 $a = b$ 。我们说“存在加法逆元”，而没说“存在唯一的加法逆元”，也是因为同一个元素的加法逆元最多只能有一个，如果存在，必定唯一。假设 a 有两个加法逆元 b 和 c ，那么就有 $b = b + 0 = b + (a + c) = (b + a) + c = 0 + c = c$ ，因此 b 其实就是 c 。乘法单位和乘法逆元也是如此。

这基本上刻画出了作为一个算术系统所有需要满足的性质。第 4 条和第 5 条告诉了我们，加法和乘法并没有那么玄妙，它们不过是从元素对到元素的映射。第 6 条到第 10 条列举了加法和乘法的基本性质。第 11 条和第 12 条定义了 0 和 1 这两个特殊的数。第 13 条中出现了“加法逆元”一词，不过“相反数”一词或许会更亲切一些。第 14 条中出现了“乘法逆元”一词，不过“倒数”一词或许也会更亲切一些。第 13 条和第 14 条实际上定义了减法和除法。减去 x ，就相当于加上 $-x$ ；除以 x ，就相当于乘以 x^{-1} 。这两条性质保证了，我们可以自由地做减法，我们可以自由地做除法，得到的数也仍然是一个合法的元素，不会出现不够减、不允许减、不够除、不允许除的情况（除以 0 除外）。因为，每一个元素都有相反数，每一个非 0 元素都有倒数。

等等，这些性质就能刻画算术系统了？似乎漏掉了不少其他关键的性质吧。比如，0 乘以任何数都等于 0，这条性质哪儿去了？其实，剩下的没有写出的性质，包括 $(-1) \cdot x = -x$ 、 $0 \leq 1$ 、 $0 \leq x \cdot x$ 等等，都是已有性质的推论。我们可以从已有性质出发，完全不关心它们的实际意义，抽象地证明出算术系统应该具有的其他性质。例如，为了证明 $0 \cdot x = 0$ ，只需要注意到：

$$0 \cdot x = (0 + 0) \cdot x = 0 \cdot x + 0 \cdot x$$

两边同时加上 $0 \cdot x$ 的加法逆元，等式仍然成立（这是因为，如果 $a = b$ ，那么加法运算将会把 $a + c$ 和 $b + c$ 映射到同一个数）：

$$\begin{aligned} 0 \cdot x + (-0 \cdot x) &= (0 \cdot x + 0 \cdot x) + (-0 \cdot x) \\ 0 &= 0 \cdot x + (0 \cdot x + (-0 \cdot x)) \\ 0 &= 0 \cdot x + 0 \\ 0 &= 0 \cdot x \end{aligned}$$

同时满足第 4 条到第 14 条的话，这些元素就构成了一个“域”。如果还满足第 15 条和第 16 条（当然还有第 1 条到第 3 条），这就是一个“有序域”。最小的有序域是有理数域。如果把上面所说的元素想成是全体有理数，把小于等于、加法和乘法解读为我们平常所说的小于等于、加法和乘法，那么整个系统满足上面所有 16 条性质。实数域则是一个更大的有序域，而有理数域只是它的其中一个子有序域。当然，还有一些有序域，它里面的元素根本不是常规意义上的数，它们之间的大小关系、加法乘法也根本不是常规意义上的大小关系和加法乘法。不过，人们已经证明了，任何一个有序域里一定都包含有一个子有序域，它和有理数域同构——它里面的一切都和有理数一样，只是各个元素的名字不一样罢了。也就是说，在任何一个有序域中，我们都可以从加法单位和乘法单位出发，把全部的或者部分的、本来有可能并不是数的元素看成是一个个的数，得到一个与有理数等价的系统：加法单位就是 0，乘法单位就是 1，等于 $1 + 1$ 的那个元素就是 2，等于 $2 + 1$ 的那个元素就是 3，2 的加法逆元就是 -2 ，2 的乘法逆元就是 $1/2$ ，等等，而且这些元素确实是不同的元素，它们之间的小于等于、加法乘法的运算结果也确实符合有理数之间的运算结果。正因为这样，我们才说，这 16 个有序域条件比较完整地描述了算术系统的要素，足以刻画算术系统了。

这里，我们突然有了一个非常激动人心的问题：一个有序域有没有可能比实数域更大（实数域只和它的某个子有序域同构）？有理数域已经封闭了，但加进去一系列新的元素之后，我们有可能得到一个封闭的、更大的实数域；那么，能否在实数域里面也加进去一些新的元素，让实数域进一步扩张成某个更大的有序域？不不，答案没有“复数域”那么简单——复数域不是有序域，因为它不是有序的。也就是说，我们无法为所有复数安排 \leq 关系，使得它能满足有序域的全部条件。在这个比实数域更大的有序域中，我们不但能继续加减乘除，还能继续做大小比较，每一个新的数和每一个原有的实数之间都有一个确定的大小关系。直观上，这似乎是不大可能的，因为实数已经把那根“线”填满了。

(三) 外星人的数学

对算术结构的形式化，让我想起了一些更折磨人的哲学问题：我们的数学体系为什么是这样的？一个文明的数学体系一定是这个样子的吗？比方说，如果人类没有视觉这种感官，我们还会创立几何这门学科吗？同时，人类是否会因为缺乏某种独特的感官，而没能建立起一个更加神奇的数学分支？

同样地，我们总会认为，自然数是最自然的概念，是每一种文明都不可避免会遇到的概念。是这样吗？会不会有某一种外星文明，他们也有一套自己的算术系统，虽然也有加减，虽然也有乘除，虽然跟我们正在使用的算术系统是同构的，但其直观意义和表示方法都跟我们完全不同。或许，在我们这里很难表达的数，在他们那里看来几乎是平凡的；或许，在我们这里很直观很自然的数，在他们的体系里却需要异常复杂的记号来表示。

作为一个科幻迷，我甚至马上想到了，这应该是一个绝佳的科幻素材：数学家发现了地外生命的某种全新的数学系统，并且一步一步尝试去解读它。为此，数学家们必须抛弃任何已有的假设，忘掉什么是自然数，什么是零，什么是加减，什么是乘除，从一片空白开始，尝试理解外星算术系统，逐渐搭建起外星数学的大厦。《你一生的故事》的数学版，多么激动人心的一件事情啊！

这样的小说已经有了。你猜谁写的？Donald Knuth！第一次听说这个时，第一反应除了膜拜还是膜拜：Knuth 简直是太牛了，他竟然写过一本小说！这本 100 多页的小说叫做 Surreal Numbers（中译《研究之美》，电子工业出版社 2012 年 1 月出版，顺便吐槽中译名），中文意思大致就是“超现实的数”。我打算直接把它叫做“超实数”——正如“实数”就是“现实的数”一样，“超实数”也就是“超现实的数”了（其实，不得不说的是，“超实数”这个译名有一个很大的弊端：它可能会与另一个叫做 hyperreal numbers 的概念发生混淆）。小说的副标题则是 How Two Ex-Students Turned on to Pure Mathematics and Found Total Happiness。小说讲述了 Alice 和 Bill，一对年轻男女，逃到一个漂亮的孤岛上“寻找自我”。他们在岛上发现了一块古老的石板，石板上用希伯来语写着一些数学公理。然后，我们将跟随 Alice 和 Bill 的思路，逐渐探索这些公理的意义，期间会走不少的弯路，历经一些失败的尝试，到最后终于成功恢复出了一个完整的算术系统。根据 Wikipedia 的描述，一个全新的数学概念竟然首先出自于一本小说，这是极罕见的例子。

在下面的讨论中，时刻记住：我们在学习外星数学。我们不能有任何先入为主的概念。请忘记“数”的概念，忘记“小于等于”的概念。下面所说的“数”是某种全新的物体，它可能与我们熟知的数有某种对应的关系，也可能与我们熟知的数没有任何关系。“小于等于”也是一种全新的概念，这四个字整个儿是一个新词，它不是由“小于”和“等于”合成的。它的意思有可能和我们的“小于等于”相吻合，也有可能和我们的“小于等于”不吻合，甚至有可能根本就没有什么合理的“意思”。“大于等于”只是一个与“小于等于”对称的说法。如果 x 小于等于 y ，我们就说 y 大于等于 x 。偶尔我们会用 \leq 代替“小于等于”，用 \geq 代替“大于等于”，不过也请假装这些符号对你来说是完全陌生的。

石板上的公理只有两个。

1. 每个数都是用两个（由已有的数构成的）集合表示的，其中左边集合里不存在这样的数，它大于等于右边集合里的某个数。假如 a 、 b 、 c 、 d 、 e 都是已经构造出来的数，并且从 a 、 b 里任取一个数，从 c 、 d 、 e 里任取一个数，前者都不会大于等于后者，那么把 a 、 b 放入左边的集合，把 c 、 d 、 e 放入右边的集合，我们就能得到一个新的数，比如 f 。我们把它记作 $f = \{a, b | c, d, e\}$ 。
2. 一个数小于等于另一个数，当且仅当前者的左集合里不存在大于等于后者的数，并且后者的右集合里不存在小于等于前者的数。比如，如果 $x = \{a, b | c\}$ ， $y = \{d | e\}$ ，那么 x 小于等于 y ，当且仅当 a 、 b 都不大于等于 y ，并且 e 不小于等于 x 。

可以看出，所有的数，包括它们之间的 \leq 、 \geq 关系，都是递归地表达出来的。不过，我们每次都需要用已有的数来定义新的数，那我们从什么地方开始呢？答案是，我们从空集开始。左集合和右集合都为空，记作 $\{\}$ ，这是我们能够得到的第一个数。再次提醒，这个数或许能被等价地理解为我们世界中的某个数，或许不能。我们暂时不做这些假设。我们把这个数记作 α 。也就是说， $\alpha = \{\}$ 。容易看出， $\alpha \leq \alpha$ ，因为 α 的左集合里确实没有哪个数大于等于 α ，并且 α 的右集合里确实没有哪个数小于等于 α （事实上 α 的左集合里和右集合里根本就没有数）。既然 $\alpha \leq \alpha$ ，反过来，也可以说成是 $\alpha \geq \alpha$ 了。在接下来的过程中时刻记住， α 既小于等于自己，又大于等于自己。

接下来，我们可以构造两个新的数 $\{\alpha\}$ 和 $\{\alpha\}$ ，不妨把它们俩分别叫做 β 和 γ 。注意， $\{\alpha\}$ 是不合法的，它违反了公理 1：左集合里的数不能大于等于右集合里的数。可以验证， $\beta \leq \alpha$ ，而 $\alpha \leq \gamma$ 。另外， β 也是小于等于 γ 的，不过请注意，这并不是根据“小于等于”的传递性推出的，而是根据“小于等于”的定义直接验证的。我们还不知道“小于等于”满足传递性。事实上，我们还不知道“小于等于”有什么直观意义。

x 小于等于 y 当且仅当 x 的左集合里不存在大于等于 y 的数，并且 y 的右集合里不存在小于等于 x 的数
 $\beta = \{\alpha\}$, $\alpha = \{\} \Rightarrow \beta$ 小于等于 α ，或者说 α 大于等于 β
 $\alpha = \{\}$, $\gamma = \{\alpha\} \Rightarrow \alpha$ 小于等于 γ ，或者说 γ 大于等于 α
 $\beta = \{\alpha\}$, $\gamma = \{\alpha\} \Rightarrow \beta$ 小于等于 γ ，或者说 γ 大于等于 β

接下来，我们完全不理会 \leq 背后有何直观意义，而是纯粹利用刚才的两个公理来证明，刚才看到的传递现象其实并不是巧合。换句话说，在这个神秘的算术系统中，如果 $x \leq y$ ，并且 $y \leq z$ ，那么 $x \leq z$ 。证明的基本方法是数学归纳法。

我们对三元组 (x, y, z) 施加归纳。假设传递性对于更早产生的三元组都是成立的。根据定义，如果 $x \leq y$ ，那么对于 x 的左集合中的任意一个元素 x^L 来说， $y \leq x^L$ 都不成立。这立即表明， $z \leq x^L$ 是绝对不可能发生的，因为我们已经有了 $y \leq z$ ，如果还有 $z \leq x^L$ 的话，根据归纳假设，就会有 $y \leq x^L$ ，矛盾。于是我们知道， x 的左集合里不存在大于等于 z 的元素。用类似的方法可以推出，不可能出现 $z^R \leq x$ 的情况，其中 z^R 是 z 的右集合中的任意一个元素。理由和刚才相仿：如果 $x \leq y$ 和 $z^R \leq x$ 同时成立，由归纳假设就会得到 $z^R \leq y$ ，这与 $y \leq z$ 相矛盾。于是我们知道， z 的右集合里不可能出现小于等于 x 的数。综合上述两条便有 $x \leq z$ 。

由于超实数所具有的递归本质，在研究超实数的性质时，最基本的方法就是数学归纳法。为了证明某个结论对任意一个数 x 都成立，我们通常会假设这个结论对所有可能的 x^L 和 x^R 都已经成立了；类似地，为了证明某个结论对任意一个二元组 (x, y) 都成立，我们通常会假设这个结论对所有可能的 (x^L, y) 、 (x^R, y) 、 (x, y^L) 、 (x, y^R) 都已经成立了。让我们来做几个练习吧。首先，让我们试着用数学归纳法来证明，对于任意一个数 x 都有：

1. $x \leq x^L$ ，其中 x^L 是 x 的左集合中的任意一个元素
2. $x \geq x^R$ ，其中 x^R 是 x 的右集合中的任意一个元素
3. $x \leq x$

证明过程并不复杂。假设这几条性质对于更早产生的数都成立。为了证明对于任意 x^L 都有 $x \leq x^L$ ，我们只需要说明，在 x 的左集合中存在一个大于等于这个 x^L 的数即可。这样的数是显然存在的，因为由归纳假设， $x^L \leq x^L$ 或者说 $x^L \geq x^L$ 始终成立。证明对于任意 x^R 都有 $x \geq x^R$ 的方法也是类似的。好了，既然 x^L 都不大于等于 x ，并且 x^R 都不小于等于 x ，就说明 $x \leq x$ 了。我们不妨把这三个结论分别叫做补充性质 1、补充性质 2 和补充性质 3。我们今后会反复用到这三个补充性质。

有人或许会说： $x \leq x^L$ ，岂不意味着 $x \geq x^L$ 吗？为什么要说得那么麻烦呢？别急，别急，我们目前还不知道 $x \leq y$ 能否推出 $x \geq y$ 呢。不过，不管怎么说， $x \geq x^L$ 还真是成立的。我们可以证明这一点，基本思路仍然是数学归纳法。为了证明对于任意 x^L 都有 $x \geq x^L$ ，我们只需要说明，这个 x^L 的左集合里不存在大于等于 x 的数，并且 x 的右集合里不存在小于等于这个 x^L 的数。根据公理 1 的规定，后者显然是成立的。前者的正确性则不太容易看出来：根据归纳假设，这个 x^L 大于等于它的左集合里的所有元素，如果这个 x^L 的左集合里存在大于等于 x 的数，由传递性便可推出 $x^L \geq x$ ，与刚才的补充性质 1 矛盾。类似地，补充性质 2 也有一个推论，即对于任意 x^R 都有 $x \leq x^R$ 。证明方法几乎完全相同： $x \leq x^R$ 的意思就是 x 的左集合里不存在大于等于 x^R 的数，并且 x^R 的右集合里不存在小于等于 x 的数，两者可以分别由公理 1 和归纳假设推出。

补充性质 1 和补充性质 2 拥有这样的推论，这使我们开始猜测：会不会在任何情况下，由 $x \leq y$ 都能推出 $x \geq y$ 呢？的确是这样，而且证明过程很简单。 $x \leq y$ 意味着某个 x^L 大于等于 y ，或者某个 x^R 小于等于 y 。如果有某个 x^L 大于等于 y ，但刚才我们证明了 x 大于等于一切 x^L ，两者结合便有了 $x \geq y$ ；如果有某个 x^R 小于等于 y ，但刚才我们证明了 y 小于等于一切 x^R ，两者结合也能推出 $x \geq y$ 。所以，不管是哪种情况， $x \geq y$ 都是成立的。

或许有人会无比兴奋地说：这个结论太有用了，之前很多别扭的说法这下都能大大简化了。例如，公理 1 原本说的是，一个数是合法的，当且仅当它的左集合里的每一个数都不大于等于右集合里的每一个数。现在，我们是不是就能把它重新叙述为，一个数是合法的，当且仅当它的左集合里的每一个数都小于等于右集合里的每一个数？且慢！我们目前只知道 $x \leq y$ 可以推出 $x \geq y$ ，但并不知道 $x \geq y$ 是否反过来推出 $x \leq y$ ，因而也就不知道 $x \leq y$ 和 $x \geq y$ 是否是完全等价的说法。对已有的数进行试验，你会觉得 $x \geq y$ 似乎真的能够推出 $x \leq y$ 。之前我们验证过 $\beta \leq \alpha$ 、 $\alpha \leq \gamma$ 和 $\beta \leq \gamma$ ，你会心安地发现，它们反过来真的都不成立。

x 小于等于 y 当且仅当 x 的左集合里不存在大于等于 y 的数，并且 y 的右集合里不存在小于等于 x 的数
 $\alpha = \{\}$, $\beta = \{\alpha\} \Rightarrow \alpha$ 小于等于 β ，或者说 β 不大于等于 α
 $\gamma = \{\alpha\}$, $\alpha = \{\} \Rightarrow \gamma$ 小于等于 α ，或者说 α 不大于等于 γ
 $\beta = \{\alpha\}$, $\gamma = \{\alpha\} \Rightarrow \beta$ 小于等于 γ ，或者说 γ 不大于等于 β

但是，如果再多生成一些数，问题就出来了。考虑这个数 $\{\beta \mid y\}$ ，由于 β 确实不大于等于 y ，因此这个数是合法的。把这个数叫做 δ 。现在，考虑 δ 和 a 的大小关系：

x 小于等于 y 当且仅当 x 的左集合里不存在大于等于 y 的数，并且 y 的右集合里不存在小于等于 x 的数
 $a = \{\}\right\}$, $\delta = \{\beta \mid y\} \Rightarrow a < \delta$, 或者说 $\delta > a$
 $\delta = \{\beta \mid y\}$, $a = \{\}\right\} \Rightarrow \delta < a$, 或者说 $a > \delta$

结果， $a \leq \delta$ 和 $a \geq \delta$ 同时成立。这说明，虽然 $x \leq y$ 可以推出 $x \geq y$ ，但是 $x \geq y$ 不能推出 $x \leq y$ ，两者并不是等价的。换言之， $x \leq y$ 和 $y \leq x$ 不可能都不成立，但却有可能同时成立。

现在，让我们来回顾一下“小于等于”这个关系所满足的性质。它满足传递性，即 $x \leq y$ 和 $y \leq z$ 可以推出 $x \leq z$ 。它满足完全性，即 $x \leq y$ 和 $y \leq x$ 至少有一个成立。但是，它不满足反对称性。反对称性的意思是，如果 $x \leq y$ 并且 $y \leq x$ ，那么 x 和 y 必然是同一个元素。然而，我们刚才已经看到了， $a = \{\}\right\}$ 和 $\delta = \{\beta \mid y\}$ 互相之间都具有 \leq 关系，但它们并不是同一个元素。我们刚才反复强调，这里的“小于等于”是一个全新的概念，不要把它想象成我们熟知的那个小于等于；现在看来，这是一个无比正确的决定——这里的“小于等于”就是不能看成是我们熟知的那个小于等于。这里的“小于等于”不满足反对称性，但我们熟知的那个小于等于却得满足这一点。

先别管它。我们继续。在 Knuth 的小说中，Alice 和 Bill 又发现了一块新的石板，石板上定义了一种叫做“加法”的二元运算，用符号 $+$ 表示。也请暂时不要和大家熟知的加法联系在一起。定义 $x + y$ 的结果是一个新的数：它的左集合是由 x 的左集合的所有元素分别加 y ，以及 x 分别加上 y 的左集合的所有元素构成的；它的右集合是由 x 的右集合的所有元素分别加 y ，以及 x 分别加上 y 的右集合的所有元素构成的。如果把 x 的左集合和右集合分别记作 $L(x)$ 和 $R(x)$ ，把 y 的左集合和右集合分别记作 $L(y)$ 和 $R(y)$ ，那么 $x + y$ 的左集合 L 和右集合 R 分别是

$$\begin{aligned} L &= \{u + y \mid u \in L(x)\} \cup \{x + v \mid v \in L(y)\} \\ R &= \{u + y \mid u \in R(x)\} \cup \{x + v \mid v \in R(y)\} \end{aligned}$$

例如，若 $x = \{a, b \mid c\}$, $y = \{d, e, f \mid g, h\}$ ，那么 $x + y$ 就是 $\{a + y, b + y, x + d, x + e, x + f \mid c + y, x + g, x + h\}$ 。显然，加法也是递归地定义出来的。在研究加法所具备的性质之前，我们必须要先做一件事情：证明如此加出来的数一定是满足公理 1 的，换句话说加出来的一定都是合法的数。这个证明是很有必要的，它直接关系到加法运算的封闭性。然而，这件事情却很容易被人忽略。Knuth 的书中有一段情节我非常喜欢，讲的就是 Alice 和 Bill 突然意识到自己忘了做这件事情。在第 12 章《灾难》的开头，Alice 说：“一小时前我醒来，发现我们昨天的证明有一个很大很大的漏洞，就是我们忘了证明 $x + y$ 是一个数了。”听了这话之后，Bill 说：“开玩笑吧，这是两个数之和，当然是一个数咯！等等，我想想……哦，我们还要检查它是否满足公理 1。”接下来的 11 页则都是 Alice 和 Bill 对这个问题的探究，可见证明这件事还真不容易。这里，我们略去证明。

这么定义加法，效果如何呢？还是用我们刚才的那些数来做试验： $a = \{\}\right\}$ 、 $\beta = \{a\}$ 、 $\gamma = \{a\}$ 。容易看出： $a + a = a$ ， $a + \beta = \{a + a\} = \{a\} = \beta$ ， $a + y = \{a + a\} = \{a\} = y$ 。利用数学归纳法不难证明，事实上， a 加上任意一个数 x 都仍然等于 x 。如果 $x = \{a, b \mid c\}$ ，那么 $a + x$ 就等于 $\{a + a, a + b \mid a + c\}$ ，根据归纳假设，这就是 $\{a, b \mid c\}$ 。根据同样的道理，任意一个数 x 加上 a 也都仍然等于 x 。如果 $x = \{a, b \mid c\}$ ，那么 $x + a$ 就等于 $\{a + a, b + a \mid c + a\}$ ，根据归纳假设，这就是 $\{a, b \mid c\}$ 。也就是说，假如我们这里所说的“加法”真的符合我们现实意义中的加法的话， $a = \{\}\right\}$ 其实就应该是个 0！

在我们给出 $a + x = x$ 始终成立的结论后，立即又说 $x + a = x$ 也是始终成立的，这并不是废话，因为我们并不知道这里的加法是否满足交换律。那么，这里的加法是否满足交换律呢？让我们试着比较一下 $\beta + y$ 和 $y + \beta$ 的结果，来做一番探究吧。由于任何数加上 a 都不变，并且 a 加上任何数也都不变，因而：

$$\begin{aligned} \beta + y &= \{|\alpha\} + \{a\} = \{\beta + \alpha \mid \alpha + y\} = \{\beta \mid y\} = \delta \\ y + \beta &= \{a\} + \{|\alpha\} = \{a + \beta \mid y + \alpha\} = \{\beta \mid y\} = \delta \end{aligned}$$

这样看来，这里的加法似乎是满足交换律的。

再次利用数学归纳法，我们可以很快得出，这里的加法就是满足交换律的。若 $x = \{a, b \mid c\}$, $y = \{d, e, f \mid g, h\}$ ，那么 $x + y$ 就是 $\{a + y, b + y, x + d, x + e, x + f \mid c + x, x + g, x + h\}$ ，而 $y + x$ 就是 $\{d + x, e + x, f + x, y + a, y + b \mid g + x, h + x, y + c\}$ ，根据归纳假设，两个数内部的组成元素是完全一致的。

同样是利用数学归纳法，我们还可以得出，这里的加法也是满足结合律的。 $(x + y) + z$ 的左集合，归根结底是由以下三类数组成： $(x^L + y) + z$ 、 $(x + y^L) + z$ 、 $(x + y) + z^L$ 。而 $x + (y + z)$ 的左集合，则是由以下三类数组成： $x^L + (y + z)$ 、 $x + (y^L + z)$ 、 $x + (y + z^L)$ 。根据归纳假设，你会发现， $(x + y) + z$ 的左集合和 $x + (y + z)$ 的左集合本质上是一样的。类似地， $(x + y) + z$ 的右集合和 $x + (y + z)$ 的右集合本质上也是一样的，因而 $(x + y) + z = x + (y + z)$ 始终成立。

这里的加法还满足有序域条件列表中的第 15 条，即 $x \leq y$ 可推出 $x + z \leq y + z$ 。事实上，我们会证明一个更强的结论： $x \leq y$ 当且仅当 $x + z \leq y + z$ ，两者互相之间都是可推导的。事实上，我们会证明一组比这还要强的结论：

1. 若 $x \leq y$ ，且 $z \leq w$ ，则 $x + z \leq y + w$
2. 若 $x + z \leq y + w$ ，但是 $z \geq w$ ，则 $x \leq y$

就像之前证明那三个补充性质一样，我们会把这两个结论捆绑到一块儿，共同施加数学归纳法。首先，我们来证明结论 1，即已知 $x \leq y$ 和 $z \leq w$ 可推出 $x + z \leq y + w$ 。根据加法的定义， $x + z$ 的左集合由所有可能的 $x^L + z$ 和 $x + z^L$ 组成， $y + w$ 的右集合由所有可能的 $y^R + w$ 和 $y + w^R$ 组成。别忘了，一个数小于等于另一个数，当且仅当前者的左集合里不存在大于等于后者的数，并且后者的右集合里不存在小于等于前者的数。因此，如果 $x + z \leq y + w$ 不成立，这就意味着 $x^L + z \geq y + w$ 、 $x + z^L \geq y + w$ 、 $y^R + w \leq x + z$ 、 $y + w^R \leq x + z$ 这四种情况当中至少会有一种出现。结果你会发现，这四个式子中的任何一个式子都会引出矛盾。例如，在第一个式子当中， $x^L + z \geq y + w$ ，但我们有 $z \leq w$ ，根据结论 2 的归纳假设便有 $x^L \geq y + w$ 。但是我们还有 $y \geq x$ ，由传递性便得到 $x^L \geq x$ ，这与之前的补充性质 1 矛盾。类似地，从后面三个式子出发，可以分别得出 $z^L \geq w$ 、 $y^R \leq y$ 和 $w^R \leq w$ ，这都与之补充性质相矛盾。

接下来，让我们看看结论 2，即已知 $x + z \leq y + w$ 和 $z \geq w$ 可推出 $x \leq y$ 。如果 $x \leq y$ 不成立，这就意味着 $x^L \geq y$ 和 $y^R \leq x$ 这两种情况当中至少会有一种出现。结果你会发现，这两个式子中的任何一个式子都会引出矛盾。例如，在第一个式子当中， $x^L \geq y$ ，但我们有 $z \geq w$ ，根据结论 1 的归纳假设便有 $x^L + z \geq y + w$ ，这与已知的 $x + z \leq y + w$ 矛盾。类似地，从第二个式子出发，可以得出 $y^R \leq x + z$ ，这也与已知的 $x + z \leq y + w$ 矛盾。

让我们来回顾一下，这里定义的加法和我们熟知的那个加法究竟有多少差距。这里定义的加法满足加法交换律、加法结合律。这里定义的加法还满足 $x \leq y \Rightarrow x + z \leq y + z$ 。这里定义的加法甚至存在加法单位。我们刚刚引入加法时就发现了， $a = \{\}\right\}$ 就是加法单位，任何数加上它的结果都不变。不过，我们目前还不知道的是，

是否每个元素都存在加法逆元？换句话说，是否对于每一个数，我们都能找出一个相应的数，使得两者相加等于那个加法单位？结果你会发现，这是根本不可能的，除非两个加数都是 a ，否则根据加法的定义，两数之和的左右集合不可能都是空的，自然也就不可能等于加法单位 a 了。

此时，又该轮到谜一般的石板登场了。Alice 和 Bill 发现，石板上还写有 x 的“逆元”的定义。递归地定义 x 的“逆元”为，将 x 的左右集合颠倒，并把集合里的每个数都变为“逆元”所得的新数。如果把 x 的左右集合分别记作 $L(x)$ 和 $R(x)$ ，把 x 的“逆元”记作 $-x$ ，那么 $-x$ 的左右集合就应该是：

$$\begin{aligned} L &= \{-u \mid u \in R(x)\} \\ R &= \{-u \mid u \in L(x)\} \end{aligned}$$

因此，如果 $x = \{a, b \mid c\}$ ，那么 $-x$ 就等于 $\{-c \mid -a, -b\}$ 。可以证明，一个合法的数的“逆元”一定是满足公理 1 的，换句话说，一个合法的数的“逆元”一定也是合法的数。这里略去证明。容易看出， a 的“逆元”就是它本身。 $\{|a|\}$ 的“逆元”为 $\{|-a|\}$ ，也就是 $\{|a|\}$ ； $\{|a|\}$ 的“逆元”为 $\{|-a|\}$ ，也就是 $\{|a|\}$ 。这表明，我们刚才的 β 和 γ 互为“逆元”。这里，我们给所有的“逆元”一词都加上了引号，因为它并不是真正的逆元，不满足加法逆元的要求。如果这里的“逆元”是真正的逆元，那么 $\beta + \gamma$ 应该等于 a ，但实际情况却并不是这样。翻翻前文你会发现，之前我们算过 $\beta + \gamma$ ，它等于 $\{\beta \mid \gamma\}$ ，也就是后来被我们记作 δ 的数。此时，一件非常有趣的事情出现了： $\beta + \gamma$ 的结果虽然不等于 a ，但却和 a 之间有一个极其特别的关系——它既小于等于 a ，又大于等于 a 。

事实上，我们可以证明，任何一个数加上它的“逆元”后，都既小于等于 a ，又大于等于 a 。我们先来证明这个结论的前半部分，即 $x + (-x)$ 一定是小于等于 a 的。再次回想一下，一个数小于等于另一个数，当且仅当前者的左集合里不存在大于等于后者的数，并且后者的右集合里不存在小于等于前者的数。反证，如果 $x + (-x)$ 不小于等于 a ，这就意味着， $x + (-x)$ 的左集合里有大于等于 a 的元素，或者 a 的右集合里有小于等于 $x + (-x)$ 的元素。由于 a 的右集合是空的，因此上述两种可能性中，后者显然是不可能的。我们只需要考虑前者即可。根据加法的定义， $x + (-x)$ 的左集合是由所有可能的 $x^L + (-x)^L$ 和 $x + (-x)^L$ 构成，但很容易看出，它们也都不可能大于等于 a 。以 $x = \{a, b \mid c\}$ 为例，根据定义有 $x = \{-c \mid -a, -b\}$ 。把所有可能的 $x^L + (-x)^L$ 和 $x + (-x)^L$ 都列出来，就是 $a + (-x) \mid b + (-x) \mid x + (-c)$ ，而它们显然都不可能大于等于 a ，因为它们的右集合里都存在小于等于 a 的数—— $a + (-a) \mid b + (-b) \mid x + (-c)$ 的右集合里有一个 $c + (-c)$ ，根据归纳假设，它们都小于等于 a 。这样，我们就证明了 $x + (-x)$ 必然小于等于 a 。同样的方法可以证明 $x + (-x)$ 一定也是大于等于 a 的：否则， $x + (-x)$ 的右集合里将会有存在小于等于 a 的元素，即所有可能的 $x^R + (-x) \mid x + (-x)^R$ 当中存在小于等于 a 的数，但这是不可能的，因为这些数的左集合里总会形如 $a + (-a) \mid b + (-b) \mid c + (-c)$ 的数，根据归纳假设，它们都是大于等于 a 的。

这个算术世界中的数没有真正的加法逆元，只有与之相加之后既小于等于加法单位，又大于等于加法单位的数；同时，回想我们之前曾经说过的，正是这种一个数既能小于等于另一个数，又能大于等于另一个数的特性，才使得这里的“小于等于”与我们现实生活中的“小于等于”不符。此时，一个神奇的想法突然冒了出来：如果我们把任意两个互相之间都存在 \leq 关系的数视为同一个元素，上面两个问题不就一并解决了吗？

更具体地说，如果 a 和 b 同时满足 $a \leq b$ 和 $b \leq a$ ，我们就说 a 和 b 是同一类数。补充性质 3 告诉我们对于任意一个数 x 都有 $x \leq x$ ，因而一个数一定和自己同类；由于 \leq 满足传递性，因而如果 a 和 b 同类， b 和 c 同类，则 a 和 c 也一定同类。于是，这个算术世界里的所有数就被划分成了一个一个不交叉的类。站在类上来看，刚才的一切都和我们现实生活中的算术系统相符了，仅有的两点区别也被消除了：对于任何一个类 A ，都存在另一个类 $-A$ ，使得两者之和等于加法单位类；任何两个不同的类 A 和 B 不但满足 $A \leq B$ 和 $B \geq A$ 至少有一个成立，而且还满足，它们不可能同时成立（否则它们就不是不同的类了）。既然这样，我们何不把每一个类当作一个元素，把之前那些运算全都搬到类上，从而得到一个更合我们意的算术系统呢？

呃……我们可以这么做吗？且慢，这里面有个问题：究竟如何把之前那些运算全都搬到类上？这个“搬”字用得太随意了，我们要严谨地刻画一下。当然，我们可以规定，为了得出 A 和 B 两个类的运算结果，只需要从 A 中选一个 a ，从 B 中选一个 b ，看看 a 和 b 的运算结果在哪个类里。但是，你怎么知道，不同的选法一定对应相同的结果呢？

好在，利用小于等于的传递性以及 $x \leq y \Leftrightarrow x + z \leq y + z$ 的性质，我们可以证明，这种事情是不会发生的。大家可以试着证明下面这些事实：如果 a 和 b 是同类的数，那么 $-a$ 和 $-b$ 也是同类的数， $a + (-a) \mid b + (-b) \mid a + (-b) \mid (-a) + b$ 也都是和 a 同类的数；如果 a 和 b 是同类的数，并且 c 和 d 也是同类的数，那么 $a \leq c$ 就一定意味着 $b \leq d$ ， $a + c$ 和 $b + d$ 也一定是同类的数。事实上，在任何一个由 $\leq, \geq, +, -$ 组成的表达式中，把某些数替换为与之同类的数，表达式都仍然是正确的。也就是说， $\leq, \geq, +, -$ 都可以应用在类的级别上，都可以变成类与类之间的运算，不会出现有矛盾、有冲突的情况。这个以类为元素的结构完全符合有序域的定义中所有涉及到小于等于和加法运算的条件，包括之前不成立的小于等于的反对称性和加法逆元的存在性！

为了完成这个有序域的构造，我们只需要定义一个满足有序域条件的乘法就行了。这件事是可以办到的。规定 x 和 y 的乘积 $x \cdot y$ 的左集合 L 和右集合 R 分别是：

$$\begin{aligned} L &= \{u \cdot y + x \cdot v - u \cdot v \mid u \in L(x), v \in L(y)\} \cup \{u \cdot y + x \cdot v - u \cdot v \mid u \in R(x), v \in R(y)\} \\ R &= \{u \cdot y + x \cdot v - u \cdot v \mid u \in L(x), v \in R(y)\} \cup \{u \cdot y + x \cdot v - u \cdot v \mid u \in R(x), v \in L(y)\} \end{aligned}$$

这里，“ $-u \cdot v$ ”是一种简写记号，它表示的是“加上 $u \cdot v$ 的加法逆元”。如果 $x = \{a, b \mid c\}$ ， $y = \{d, e, f \mid g, h\}$ ，那么 $x \cdot y$ 的左集合就是下表中所有红色区域里的数，右集合就是所有黄色区域里的数。可以证明，这种乘法的定义满足交换律、结合律以及对加法的分配律，这里我们略去证明。

	d	e	f	g	h
a	$a \cdot y + x \cdot d - a \cdot d$	$a \cdot y + x \cdot e - a \cdot e$	$a \cdot y + x \cdot f - a \cdot f$	$a \cdot y + x \cdot g - a \cdot g$	$a \cdot y + x \cdot h - a \cdot h$
b	$b \cdot y + x \cdot d - b \cdot d$	$b \cdot y + x \cdot e - b \cdot e$	$b \cdot y + x \cdot f - b \cdot f$	$b \cdot y + x \cdot g - b \cdot g$	$b \cdot y + x \cdot h - b \cdot h$
c	$c \cdot y + x \cdot d - c \cdot d$	$c \cdot y + x \cdot e - c \cdot e$	$c \cdot y + x \cdot f - c \cdot f$	$c \cdot y + x \cdot g - c \cdot g$	$c \cdot y + x \cdot h - c \cdot h$

容易看出， a 乘以任意数都还是 a 。再来试试 β 乘以 γ ，它应该是 $\{|a\rangle \cdot |a\rangle\} = \{|a\rangle \cdot |y + \beta \cdot a - a \cdot a\rangle\} = \{|a + a - a\rangle\} = \{|a\rangle\} = \beta$ 。事实上，用数学归纳法很容易证明，用任何一个数 x 来乘以 y ，结果都仍然是 x ：让 $x = \{a, b \mid c\}$ 去乘以 $y = \{|a\rangle\}$ ，将会得到 $\{a \cdot y, b \cdot y \mid c \cdot y\}$ ，根据归纳假设它就是 $\{a, b \mid c\}$ 。于是， y 就成为了乘法单位。最后，我们还可以证明，对于任意的 x 、 y 都有，若 $a \leq x$ 和 $a \leq y$ ，则 $a \leq x \cdot y$ 。“乘法逆元” x^{-1} 的公式则是：

$$\begin{aligned} L &= \{0\} \cup \{(1 + (u - x) \cdot v) / u \mid u \in R(x), v \in L(x^{-1})\} \cup \{(1 + (u - x) \cdot v) / u \mid u \in L(x), v \in R(x^{-1})\} \\ R &= \{(1 + (u - x) \cdot v) / u \mid u \in L(x), v \in L(x^{-1})\} \cup \{(1 + (u - x) \cdot v) / u \mid u \in R(x), v \in R(x^{-1})\} \end{aligned}$$

其中“ $/ u$ ”也是一种简写记号，它表示“乘以 u 的乘法逆元”。可以证明，每个数的“乘法逆元”都是一个合法的数，两者相乘都既小于等于 y ，又大于等于 y 。类似地，我们可以把它们全部搬到类上，可以证明，这么做不会产生冲突。这些证明冗长而乏味，我们都略去了。这样一来，外星数学的数类，以及它们之间的小于等于、加法运算、乘法运算，就完全符合有序域的 16 个要求了。外星算术系统就是一个有序域！根据之前提到的结论，我们世界里的有理数运算可以被放进这个外星系统当中， a 所在的类就是那个 0， y 所在的类就是那个 1。

这种“把类当作元素”的算术结构建立方法并不怪异。其实，我们世界中的有理数域也是用这种方法建立起来的。有理数可以看作是一个一个的整数对。不妨用 a 、 b 、 c 、 d 来表示整数，用 0 、 $+$ 、 $-$ 、 \cdot 、 \leq 来表示整数世界里的零、加法、减法、乘法、小于等于，那么规定 (a, b) 加上 (c, d) 的结果为 $(a \cdot d + b \cdot c, b \cdot d)$ ，规定 (a, b) 乘以 (c, d) 的结果为 $(a \cdot c, b \cdot d)$ ，再规定 (a, b) 小于等于 (c, d) 当且仅当 $(0 \leq b \cdot d \wedge a \cdot d \leq b \cdot c) \vee (b \cdot d \leq 0 \wedge b \cdot c \leq a \cdot d)$ ，其中 \wedge 和 \vee 分别表示“并且”和“或者”。这种定义会带来很多问题，比如两个不同的元素可能互相之间都存在小于等于的关系，比如不是所有元素都有加法逆元，不是所有元素都有乘法逆元。然而，如果我们定义 (a, b) 和 (c, d) 同类当且仅当 $a \cdot d - b \cdot c = 0$ （这里的 \cdot 、 $-$ 、 $=$ 、 0 都取它们平常的意义），那么所有元素就会被划分成一个一个的类，这些类以及它们之间的运算就满足有序域的条件了。

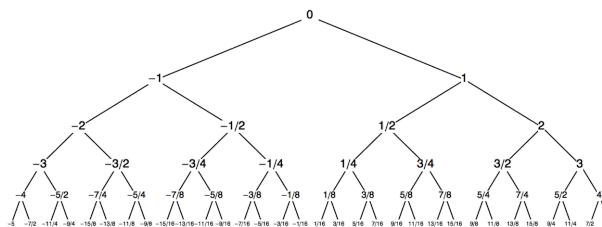
由于这个外星有序域是建立在数类之上的，因此严格地讲，我们不能说 a 相当于我们世界中的 0，而只能说 a 所在的类相当于我们世界中的 0。为了叙述简便，我们今后就用上横线来表示“所在的类”， x 意即 x 所在的类。有了这个记号，我们就可以这么说了： a 对应于我们世界中的 0， y 对应于我们世界中的 1。之前已经说过， β 和 γ 互为加法逆元，看来 β 就应该对应于我们世界中的 -1 了吧。因此， $\beta \cdot \beta$ 与 y 就应该是同一个数类吗？简单验证一下，嘿，还真是！事实上，不但 $\beta \cdot \beta$ 与 y 属于同一个数类，而且它们是同一个数： $\beta \cdot \beta = \{ | a \} \cdot \{ | a \} = \{ a \cdot \beta + \beta \cdot a - a \cdot a \} = \{ a + a - a \} = \{ a \} = y$ 。其实，我们完全不必担心算着算着会出现不一致的现象，因为我们已经知道了，外星系统中的加减乘除与我们的数域是相吻合的（至少在有理数的范围内是相吻合的）。因此，接下来，我们就直接使用 0、-1、1 来代表 a 、 β 、 y 了。

$\{1\}$ 相当于我们的什么数呢（注意，现在的以及接下来将会出现的所有构造，都是符合公理 1 的）？它相当于我们的 2，因为 $\{1\} = 1 + 1$ 。 $\{0, 1\}$ 相当于我们的什么数呢？它也相当于我们的 2，因为可以验证， $\{0, 1\}$ 既小于等于 $\{1\}$ ，又大于等于 $\{1\}$ ，两者所在的类是同一个数类。接下来，我们会直接使用 2 来代表 $\{1\}$ 。类似地， $\{2\} = 2 + 1$ ，因而它所在的类相当于我们的 3；可以验证， $\{1, 2\}$ 和 $\{0, 1, 2\}$ 所在的类也是这个类，它们也都相当于我们的 3。 $\{-1\}$ 则相当于我们的 -2，你有很多种不同的方法看出这一点来： $\{| -1\} = (-1) + (-1)$ ，同时也确实是 2 的相反数。 $\{0, 1\}$ 呢？它相当于我们的 $1/2$ ，可以验证，拿 $\{0, 1\}$ 和它自身相加，或者拿 $\{0, 1\}$ 乘以 2，都将会得到一个既小于等于 1 又大于等于 1 的数。很容易想到， $\{-1, 0\}$ 当然也就相当于我们的 $-1/2$ 了。

进一步了解外星有序域之后，为了让接下来的叙述更加简洁，我们将彻底省略“所在的类”的提法，并放宽各种运算符号的用法。其实，在我们的世界里，我们也是这么做的。我们不会说 $1/2$ 所在的类和 $2/4$ 所在的类是同一个数类，而是说 $1/2$ 和 $2/4$ 是同一个数的两种不同的表示方法，并以 $1/2 = 2/4$ 记之。类似地，我们今后也认为， $\{1\}$ 和 $\{0, 1\}$ 是同一个数的两种不同的表示方法，并以 $\{1\} = \{0, 1\}$ 记之。另外，我们世界里的符号和外星世界里的符号之间的界限也将越来越模糊。数字 2 既可以代表我们世界里的数，也可以代表外星世界里对应的数，还可以代表这个数的任何一种特定的表示方法。这样，我们便能方便地给出更多的例子，来揭示两个世界之间的联系：

$$\begin{aligned}\{0, 1/2\} &= \{0, 1/2, 1\} = \{0, 1/2, 1, 2\} = \{-1, 0, 1/2\} = 1/4 \\ \{0, 1/4\} &= \{0, 1/4, 1/2\} = \{0, 1/4, 1\} = \{-1, 0, 1/4\} = 1/8 \\ \{1/2, 1\} &= \{0, 1/2, 1\} = \{1/2, 1, 2\} = \{0, 1/2, 1, 2\} = 3/4 \\ \{1, 2\} &= \{0, 1, 2\} = \{3/4, 1, 2, 3\} = \{0, 1/2, 1/2, 3\} = 3/2 \\ \{1, 3/2\} &= \{0, 1, 3/2, 2\} = \{1/4, 1/2, 1, 3/2, 2, 3\} = 5/4 \\ \{0, 2\} &= \{1/2, 2\} = \{3/4, 2\} = \{0, 1/2, 5/4, 3/2, 2\} = 1\end{aligned}$$

由之前证明过的补充性质 1 和补充性质 2 可知，用左右集合表达出来的数，其数值一定大于左集合里的最大数，小于右集合里的最小数。事实上，可以证明，这个数值一定就是下图所示的树里所有大小夹在左右集合之间的数中最上层的那一个。图中所示的树只画了前面几层，实际上这棵树会无穷伸展下去。



大家或许会敏锐地发现，这里面漏掉了好多数。比如说， $1/3$ 到哪里去了？既然外星系统是一个有序域，它里面包含有理数域，那应该有 $1/3$ 呀？为了得出 $1/3$ 的表达式，我们不妨把 $3 = \{2\}$ 代入到之前给出的 x^1 的公式里去。由于 $\{2\}$ 的左集合只有一个 2，右集合是空的，因此代入后的式子并不复杂：

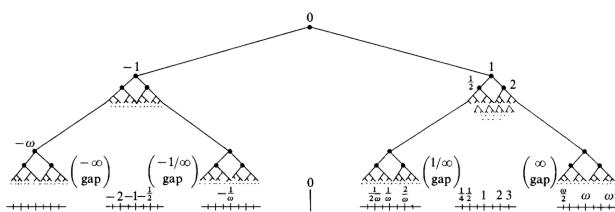
$$\begin{aligned}L &= \{0\} \cup \{(1 + (u - x) \cdot v) / u \mid u \in R(x), v \in L(x^{-1})\} \cup \{(1 + (u - x) \cdot v) / u \mid u \in L(x), v \in R(x^{-1})\} = \{0\} \cup \{(1 + (2 - 3) \cdot v) / 2 \mid v \in R(1/3)\} \\ R &= \{(1 + (u - x) \cdot v) / u \mid u \in L(x), v \in L(x^{-1})\} \cup \{(1 + (u - x) \cdot v) / u \mid u \in R(x), v \in R(x^{-1})\} = \{(1 + (2 - 3) \cdot v) / 2 \mid v \in L(1/3)\}\end{aligned}$$

等等，这不对呀？怎么 $1/3$ 的表达式里面还有 $1/3$ 呢？其实，第一次介绍 x^1 的公式时，我们有意没提这件事，也不知道大家是否已经注意到了。实际上，这个公式的含义是需要另作说明的。它并不是什么循环定义，而是让我们从 $\{0\}$ 开始，根据规则不断地迭代下去。刚开始 L 里只有 0，而 R 里什么都没有。把 L 中的这个 0 代入 R 中的 $(1 + (2 - 3) \cdot v) / 2$ ，于是得到 $1/2$ ，它便成了 R 里的成员；再把 R 中的 $1/2$ 代入 L 中的 $(1 + (2 - 3) \cdot v) / 2$ ，于是得到了 $1/4$ ，它便成了 L 里的新成员；再把 L 中的 $1/4$ 代入 R 中的 $(1 + (2 - 3) \cdot v) / 2$ ，于是得到了 $3/8$ ，它便成了 R 里的新成员……你会很快意识到，这样迭代下去是没个完的。因而，最后得到的结果当中，左右集合里都有无穷多个项。如果非得把它写出来，那大致就是 $\{0, 1/4, 5/16, 21/64, \dots | 1/2, 3/8, 11/32, 43/128, \dots\}$ 。当然，这只是 $1/3$ 的其中一种表示方法罢了，不过大家可以立即看到其原理，本质上就是把已有的小于 $1/3$ 但越来越接近 $1/3$ 的数放在左边，把已有的大于 $1/3$ 但越来越接近 $1/3$ 的数放在右边，就像 Dedekind 分割那样。根据刚才提到的结论，如此得来的数将夹在左集合的最大数和右集合的最小数之间，它就只能是 $1/3$ 了。注意，石板上的公理并没有告诉我们，左集合和右集合内是否允许含有无穷多个元素。但是，为了得到一个有序域，为了写出一个 x^1 的公式，左集合和右集合内必须允许含有无穷多个元素。我们也就这么规定了。

允许集合里有无穷个数，这不得了。我们可以用同样的方法产生包括无理数在内的所有实数。可以证明这些实数的行为与我们通常所说的实数也是一致的。因此，实数域也是外星系统的一个子有序域。同时，我们还能构造出一系列更匪夷所思的数。比如， $\{0, 1, 2, 3, \dots | \omega\}$ 等于多少？根据公理 1，这是一个合法的数。根据刚才的结论，它的数值大于左集合里的所有数。因此， $\{0, 1, 2, 3, \dots | \omega\}$ 将会是实实在在等于无穷大的一个数！不妨把它记作 ω ，故事远没到此结束。 ω 只是第一个比所有数都更大的数。你可以利用 ω 构造出 $\{\omega | \}$ 。它等于多少呢？你会发现，它等于 $\omega + 1$ 。你可以根据加法的定义写出 $\omega + 1$ 的结果，并检验它和 $\{\omega | \}$ 互相之间的 \leq 关系，从而证实 $\{\omega | \}$ 和 $\omega + 1$ 的确是同一个数。类似地， $\{\omega, \omega + 1 | \}$ 将会是 $\omega + 2$ ， $\{\omega, \omega + 1, \omega + 2 | \}$ 将会是 $\omega + 3$ 。等到 $\omega + 4$ 、 $\omega + 5$ 也都有了之后，接下来或许该考虑的就是 $\{\omega, \omega + 1, \omega + 2, \omega + 3, \dots | \}$ 了，它应该等于 $\omega + \omega$ ，也就是 ω^2 ！类似地， $\{\omega \cdot 2 | \}$ 就是 $\omega \cdot 2 + 1$ ， $\{\omega \cdot 2, \omega \cdot 2 + 1 | \}$ 就是 $\omega \cdot 2 + 2$ ，而 $\{\omega \cdot 2, \omega \cdot 2 + 1, \omega \cdot 2 + 2, \dots | \}$ 就是 $\omega \cdot 3$ 了。等到 $\omega \cdot 4$ 、 $\omega \cdot 5$ 也都有了之后，接下来或许该考虑的就是 $\{\omega, \omega \cdot 2, \omega \cdot 3, \dots | \}$ 了，它应该等于 $\omega \cdot \omega$ ，也就是 ω^{ω} ！你不妨想象一下 ω^3 、 ω^4 的构造过程，等你想好了之后，我们便有了 $\{\omega, \omega^2, \omega^3, \dots | \}$ ，它就是 ω^{ω} ！

熟悉序数理论的朋友可能知道，这里的 ω 借用自第一个极限序数符号。但是，和序数里的 ω 不一样，这里的 ω 是一个实实在在的数。在序数理论中， $1 + \omega$ 和 $\omega + 1$ 是不同的， $\omega \cdot 2$ 和 $2 \cdot \omega$ 也是不同的；但在这里，我们有 $1 + \omega = \omega + 1$ ，并且 $\omega \cdot 2 = 2 \cdot \omega$ 。因为，这里的 ω 是一个实实在在的数，它仍然满足加法交换律、乘法交换律等一切有序域应该具有的性质。

更奇怪的则是 $\{0, 1, 2, 3, \dots | \omega\}$ ，它将等于 $\omega - 1$ ；而 $\{0, 1, 2, 3, \dots | \omega, \omega - 1\}$ 则等于 $\omega - 2$ 。那么， $\{0, 1, 2, 3, \dots | \omega, \omega - 1, \omega - 2, \dots\}$ 呢？它将等于 $\omega/2$ 。而 $\{0, 1, 2, 3, \dots | \omega, \omega/2, \omega/4, \omega/8, \dots\}$ 将会带来 $\sqrt{\omega}$ 。当然，从另一个方向上看，我们还有 $\{0, -1, -2, \dots\}$ ，它显然应该等于 $-\omega$ 。别忘了，作为一个有序域，每个数不但都有自己的相反数，还有自己的倒数。 $\{0 | 1, 1/2, 1/4, 1/8, \dots\}$ 将等于 $1/\omega$ ，直观地说也就是无穷小量。 $\{0 | 1/\omega\}$ 则等于 $1/(1/\omega)$ ，而 $\{1/\omega | 1/2, 1/4, 1/8, \dots\}$ 则等于 $2/\omega$ 。我们刚才创造的所有数都是合法的，它们确实都是一个个的数，它们依旧能比较大小，依旧能参与运算。不过，它们明显超出了实数的范围。看来，一个有序域完全有可能比实数域更大。



所以说，Knuth 才把这个外星系统叫做“超实数”。可以证明，由超实数构成的有序域是最大的有序域，其他所有可能的有序域，都只是超实数域里的一个子有序域。

我们把石板上描述的数学对象解读为我们熟知的数字（以及我们不熟知的数字），因为两者具有相同的代数结构。但实际上，别忘了，这些数学对象其实可以被我们解读成任何东西，可能是对我们有用的，可能是对我们没用的。

(四) 并行的棋局

事实上，这个全新的算术系统是 Knuth 从另一位数学大神 John Conway 那儿听来的。Knuth 对此非常感兴趣，他把 TAOCP 的写作计划搁置了一周，完成了这部 100 多页的小说，并在 1974 年出版。“超实数”这个词是 Knuth 自己杜撰的。Conway 非常喜欢这个词，并把它用到了他在 1976 年出版的专著 On Numbers and Games 中。这是一本非常有特色，非常 geek 的书。书的内容分成了第零部分 On Numbers 和第一部分 and Games，全书的第一句是，This book is in two = {zero, one | } parts。在这本书中，Conway 给出了超实数背后的直观意义：博弈游戏。

	□	□	日	田
图 1	图 2	图 3	图 4	图 5

让我们来考虑这样一个游戏，游戏的名字叫做 Domineering。两个玩家轮流在一个形状不规则的棋盘上放置多米诺骨牌。不妨把这两个玩家分别记作“左玩家”和“右玩家”。左玩家只允许在棋盘上放置竖直的多米诺骨牌，右玩家只允许在棋盘上放置水平的多米诺骨牌。谁先不能放了，谁就输了。显然，有的棋局可能会对左玩家更有利，有的棋局可能会对右玩家更有利。让我们来分析几种局面吧。图 1 显示的是一个“空棋盘”。显然，这是一个“中立棋局”，先走的人直接就输了，后走的人直接就赢了。图 2 所示的棋局虽然与空棋盘不同，但本质上一样——这仍然是一个谁后走谁赢的公平局面。图 3 所示的棋局则明显对右玩家更有利一些，事实上不管谁先走，右玩家一定都会获胜。图 4 所示的棋局则偏向于左玩家，不管谁先走，左玩家都会获胜。

Conway 给出了两个棋局相加的含义。定义两个棋局的和为同时包含这两个棋局的新棋局。如果说有两个棋局，棋局 A 和棋局 B，那么 $A + B$ 的意思就是两个玩家同时在棋局 A 和棋局 B 上游戏。轮到某个玩家行动时，这个玩家可以选择在 A 上走一步，也可以选择在 B 上走一步，只要 A 和 B 当中至少一个还能走，他就不算输。例如，我们可以让两个玩家同时在图 3 和图 4 上游戏，这个游戏就记作



不难看出，它的效果等价于一个空的棋局——谁后走谁赢。

图 5 所示的 L 形棋盘就更有意思了。如果左玩家先行，他有两种走法。其中一种走法将刚好给右玩家留下一步，把自己送上了绝路；更聪明的走法则是把棋盘弄断，从而形成 $\square + \square$ 的局面，让右玩家直接死掉。如果是右玩家先行，他只有一步可走，但走完之后就再也无计可施，而左玩家还能多走一步。因此，不管谁先走，这个 L 形棋盘都是左玩家必胜的。

一个有趣的问题：图 4 的 I 形棋盘和图 5 的 L 形棋盘都是左玩家必胜的局面，都是对左玩家有利的局面，那么哪个局面会让左玩家赢得更爽一些？大家或许会说，当然是 I 形棋盘更爽。凭什么这么说？有什么理由吗？注意这里我们实际上在做的事情：我们希望能找出一种（对于左玩家来说）棋局优劣的判断标准。其中一种绝妙的想法是，让 I 和 L 两个棋局同时进行，不过左玩家要在 L 中扮演右玩家的角色。换句话说，考虑这样的复合棋盘：

目 录

两个玩家仍然轮流放置多米诺骨牌，不过规则稍有修改：在 I 形棋盘上，左玩家仍然竖直地放牌，右玩家仍然水平地放牌；但在 L 形棋盘上，两个玩家的身份颠倒了：左玩家只能放水平的牌，右玩家则改放竖直的牌。此时，左玩家在 I 形棋盘上有优势，右玩家在 L 形棋盘上有优势。最终整个棋局更偏向于谁？这显然是一场赤裸裸的“优势绝对值”之战。不难验证，事实上整个棋局仍然是左玩家必胜，不管谁先走谁后走。这说明，左玩家在 I 形棋盘中具有更大的优势，因而 I 形棋盘比 L 形棋盘更好。

对于一个棋局 A，如果我们完全交换左玩家和右玩家的地位，得到的新棋局就叫做棋局 A 的“反棋局”，记作 $-A$ 。我们规定，对于棋局 A 和棋局 B 来说，如果棋局 A 和棋局 B 的反棋局同时进行（即游戏在 $A + (-B)$ 上进行），左玩家必胜（不管谁先走），那么对于左玩家来说，棋局 A 比棋局 B 更好，或者说棋局 A 优于棋局 B；反之，如果右玩家必胜（不管谁先走），那么对于左玩家来说，棋局 A 比棋局 B 更差，或者说棋局 A 劣于棋局 B。如果 $A + (-B)$ 是一个谁后走谁必胜的中立局面，我们就认为，棋局 A 和棋局 B 对于左玩家来说优劣相同，或者说棋局 A 平于棋局 B。注意，今后不做特殊说明时，棋盘的优劣都是针对左玩家来说的。

棋局 A 劣于棋局 B 的另一个等价的定义就是，如果棋局 B 优于棋局 A，我们就说棋局 A 劣于棋局 B。如果棋局 B 优于棋局 A，说明 $B + (-A)$ 是左玩家必胜的，因而整个复合棋局的反棋局将会是右玩家必胜的。整个复合棋局的反棋局是什么？就是 B 的反棋局，加上 A 的反棋局的反棋局，也就是 $A + (-B)$ 了。而 $A + (-B)$ 对于右玩家必胜，这正好符合之前给出的 A 劣于 B 的定义。

不过目前，没有任何东西可以向我们保证，这些定义是合理的。按照这个优劣定义，没准会出现 A 优于 B 、 B 优于 C 、 C 优于 A 的荒唐一幕，或者棋局 A 优于棋局 B 但 $A + C$ 劣于 $B + C$ 的悖理情形。不过，或许我们能证明，这些情况都不会出现。至少我们可以证明一些最基本的事实，比如根据上面的定义，一个棋局一定和它自身的优劣程度相同。为此，我们只需要说明棋局 $A + (-A)$ 是一个中立局面即可。其实， $A + (-A)$ 显然是谁后走谁必胜的，因为后走的人有一个绝妙的无敌必胜招数：它只需要完全模仿对方的行为即可。先走的人每走一步，他都能在另一个棋盘上照着走一步，从而永远不可能先无路可走。

左玩家必胜（不管谁先走）的棋局 A 也一定优于任意一个中立棋局。我们也可以证明这一点，只需要说明棋局 A 加上中立棋局的反棋局后，左玩家能必胜即可。他显然是能获胜的，他可以在 A 上该怎么走就怎么走，除非回应右玩家在中立棋局上的行动。反过来，右玩家必胜（不管谁先走）的棋局也就一定劣于中立棋局。

现在开始，就是真正神奇的地方了。我们把左玩家走一步之后可能形成的所有局面构成的集合叫做左集合，把右玩家走一步之后可能形成的所有局面构成的集合叫做右集合。看看图 4 的 I 形棋盘：左玩家无论走哪一步，剩余的棋盘都是图 2 的样子；同时，右玩家没有可以走的空间，右集合是一个空集。一个棋局的左集合和右集合完整地描述了再走一步之后这个棋局所有可能的结果，于是我们可以用这两个集合来刻画该棋局。我们就说：

目 录

图 1 和图 2 的棋局实质上是相同的：左右集合都是空集。它们都应该表示成 $\{\}$ 。图 3 所示的棋盘需要引起额外的注意：“不能走”和“可以走到空棋盘状态”是两个完全不同的概念。因此，图 3 的棋盘应该被表示为

$$\square = \{\text{空棋盘}\}$$

类似地，

$$\blacksquare = \{\text{空棋盘}\}$$

图 5 的 L 形棋盘则可以表示为

$$\begin{array}{cccc} \blacksquare & \square & \square & \square \\ =\{ & , & + & | \} \end{array}$$

下面我们来看看，在这种棋局表示法下，如何形式化地定义棋局的加法、棋局的反棋局以及棋局的优劣。

如果棋局 A 和棋局 B 同时进行，那么左玩家走一步后可以得到的局面，有可能是在棋局 A 当中走一步（同时棋局 B 保持不动）所得，也可能是在棋局 B 当中走一步（同时棋局 A 保持不动）所得；类似的，右玩家能走出的局面，则是由棋局 A 的所有右集合元素都加上棋局 B，以及棋局 B 的所有右集合元素都加上棋局 A 构成的。如果把棋局 A 的左右集合分别记作 $L(A)$ 和 $R(A)$ ，把棋局 B 的左右集合分别记作 $L(B)$ 和 $R(B)$ ，那么 $A + B$ 的左右集合就应该分别是：

$$\begin{aligned} L &= \{u + B \mid u \in L(A)\} \cup \{A + v \mid v \in L(B)\} \\ R &= \{u + B \mid u \in R(A)\} \cup \{A + v \mid v \in R(B)\} \end{aligned}$$

左玩家在 $-A$ 中的合法移动，也就是 A 的右集合中的元素；右玩家在 $-A$ 中的合法移动，也就是 A 的左集合中的元素。别忘了，移动完之后，双方的决策继续保持反着的状态。如果棋局 A 的左右集合分别记作 $L(A)$ 和 $R(A)$ ，那么 A 的反棋局的左右集合就应该是：

$$\begin{aligned} L &= \{ -u \mid u \in R(A) \} \\ R &= \{ -u \mid u \in L(A) \} \end{aligned}$$

最后，我们给出“劣于或平于”的形式化描述。如果棋局 A 劣于或者平于棋局 B ，这意味着 $A + (-B)$ 有两种可能：右玩家必胜，或者谁后走谁必胜。总之，左玩家先走是绝不可能有必胜策略的。反过来，如果左玩家先走真的没有必胜策略（从而右玩家后走将必胜），这正好对应着那两种情况：如果此时再已知右玩家先走也没有必胜策略，那就是谁后走谁必胜；如果此时右玩家先走将必胜，那就属于右玩家必胜的情况。因此， $A + (-B)$ 是右玩家必胜或者谁后走谁必胜的，当且仅当左玩家先走没有必胜策略。这等于是说，左玩家不会在 $A + (-B)$ 上走出一种谁后走谁必胜或者左玩家必胜的局面。如果左玩家在 A 上面走一步，结果一定不会优于或平于 B ；同时， A 也一定不会优于或平于左玩家在 B 上任意走一步（相当于右玩家在 B 上走一步）的结果。换句话说 A 的左集合中的元素不会优于或平于 B ，同时 B 的右集合中的元素不会劣于或者平于 A 。至此为止，大家已经发现，棋局的加法运算，棋局的反棋局运算，以及棋局之间的“劣于或平于”关系，完全等同于超实数的公理和定义！为了保证所有的棋局都是超实数，我们强行规定，所有棋局也都必须满足超实数本身的构造限制，即左集合中没有元素优于或平于右集合中的某个元素。换句话说，对于左玩家来说，左玩家走了一步后所得的棋局，局势一定比右玩家走了一步后所得的棋局更险恶。更直观地说，我们这里考虑的都是先走比后走更不利、每多走一步就更接近死亡的游戏。

为了让棋局完全符合超实数，安全地继承所有超实数中的结论，我们可以生硬地、脱离实际意义地定义棋局的乘法：

$$\begin{aligned} L &= \{ u \cdot y + x \cdot v - u \cdot v \mid u \in L(A), v \in L(B) \} \cup \{ u \cdot y + x \cdot v - u \cdot v \mid u \in R(A), v \in R(B) \} \\ R &= \{ u \cdot y + x \cdot v - u \cdot v \mid u \in L(A), v \in R(B) \} \cup \{ u \cdot y + x \cdot v - u \cdot v \mid u \in R(A), v \in L(B) \} \end{aligned}$$

这样便能把谁是 1 确定下来。现在，超实数的所有公理和定义，都能用来描述棋局的世界了。于是，超实数的一切定理一切性质，包括小于等于的传递性，加法乘法的交换率，乃至整个系统是一个有序域，也都将适用于棋局。事实上，棋局和超实数将会变得同构，它们之间存在最为深入的对应关系。每一个棋局都将对应一个数，棋局之间的加法就是数与数之间的加法，我们完全不必担心棋局的运算结果不符合数的运算结果。和超实数一样，更准确地说，应该是每一个棋局类都将对应一个数，棋局类之间的加法就是数与数之间的加法。不过，和之前讲超实数时一样，为了简便起见，接下来我们都省去“类”字。

空棋盘 = { | }，也就是 0，我们的中立棋局。根据我们之前的结论，左玩家必胜的棋局都将是优于中立棋局，从而对应一个大于 0 的数；右玩家必胜的棋局都将是劣于中立棋局，从而对应一个小于 0 的数。因此，看看棋局所对的数是正是负，就能判断棋局究竟对谁有利。举例来说：

$$\begin{aligned} \square \\ = \{ \mid \text{空棋盘} \} = \{ \mid 0 \} = -1 \end{aligned}$$

这个值比 0 小，说明该棋局中右玩家必胜。而

$$\begin{aligned} \square \quad \square \\ = \{ \mid \} = \{ 0 \mid \} = 1 \end{aligned}$$

这个值比 0 大，说明该棋局中左玩家必胜。 L 形棋盘

$$\begin{aligned} \square \quad \square \quad \square \quad \square \quad \square \\ = \{ \quad , \quad + \quad | \quad \} = \{ -1, 0 + 0 \mid 1 \} = \{ -1, 0 \mid 1 \} = 1/2 \end{aligned}$$

说明它也是对于左玩家来说必胜的。注意到 I 形棋盘的值大于 L 形棋盘，这说明 I 形棋盘优于 L 形棋盘，是一个左玩家能赢得更痛快的棋局。一个直观意义就是，在 I 形棋盘中，左玩家能以一步领先的优势获胜，这个步数优势比 L 形棋盘更大。在 L 形棋盘上，左玩家只能以半步的优势险胜。注意 $1/2 + 1/2 + (-1) = 0$ ，而

$$\begin{aligned} \square \quad \square \quad \square \\ \end{aligned}$$

真的就是一个中立棋局！

在我们给出棋局的形式化描述时，我们并没有关心实际的游戏规则。这是一个非常一般的棋局理论，它还可以用到很多其他的游戏中。回到本文最开头最开头的地方。我们当时留下了一个问题。 $\circ\bullet$ 真的等于 $1/2$ 吗？我们现在可以很轻易地回答了。如果把白方想成是左玩家，那么 $\circ\bullet = \{\text{空棋盘} \mid \circ\} = \{0 \mid 1\} = 1/2$ 。我们之前曾经用 $\circ\bullet + \circ\bullet + \bullet = 0$ 的办法推出了 $\circ\bullet = 1/2$ ，这也是完全有道理的。

有了这套理论，我们便能用纯代数的方法对新的棋局进行分析。考虑下面这个问题：对于左玩家来说，棋局 $\circ\bullet$ 和 $\circ\bullet\bullet$ 相比，哪个的优势更大？为了解决这个问题，我们只需要计算一下 $\circ\bullet\bullet$ 的值：

$$\circ\bullet\bullet = \{\text{空棋盘} \mid \circ, \circ\bullet\} = \{0 \mid 1, 1/2\} = 1/4$$

而 $\circ\bullet = 1/2$ ，它的优势更大一些。由此可以进一步推出，棋局

○●●
●○

的值为 $1/4 + (-1/2) = -1/4$ ，即右玩家必胜。而棋局

○●●
○●●
●○

的值则为 $1/4 + 1/4 + (-1/2) = 0$ ，即这是一个中立棋局，谁后走谁就能必胜。事实上也确实是这样，你可以自己分析一下。

更厉害的是，由于超实数的左右集合都允许有无穷多个元素，因此这里所说的棋局也允许是无限的。例如，我们可以算出 $\circ\bullet\circ\bullet\circ\bullet\circ\bullet\dots = 2/3$ ，而棋局

○●○○○○○○●○…
○●○○○○○○●○…
○●○○○○○○●○…
●
●

真的就是一个中立棋局！

这还没结束。由于超实数的值有可能是无穷大量，因而棋局的值也有可能是无穷大量。 $\circ\circ\circ\dots$ 显然等于 ω ，而 $\circ\circ\circ\dots\dots$ 则等于 $\omega + 1$ 。把这样的行加入任何一个原本有限的棋局里，都会使得左玩家必胜。由于超实数的值有可能是无穷小量，因而棋局的值也有可能是无穷小量。 $\circ\bullet\bullet\bullet\dots$ 等于 $1/\omega$ ，而 $\circ\bullet\bullet\bullet\dots\bullet$ 则等于 $1/(\omega \cdot 2)$ 。把这样的行加入任何一个原本有限的并且右玩家必胜的棋局里，都不可能使局面发生逆转；但把这样的行加入任何一个中立棋局里，左玩家就会因为获得一个无穷小的步数优势而必胜。

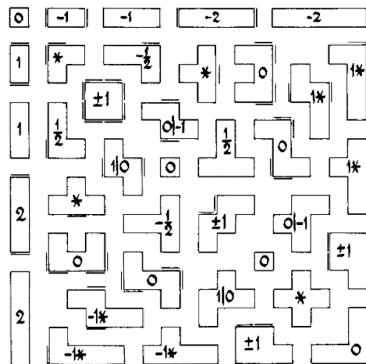
需要补充的是，并不是所有的棋局都能用数来衡量。让我们回到 Domineering 游戏，看一看下图所示的棋盘：

田

它既不属于左玩家必胜的情况，又不属于右玩家必胜的情况，当然也不是中立棋局。这是一种全新的情况——谁先走谁就赢。那这个棋盘的值是多少呢？答案是，这个棋盘不对应任何数值，因为它被排除在了我们的考虑范围之外。为了让棋局都是超实数，我们规定了棋局左集合的元素不能优于或平于右集合的元素，这将会漏掉很多棋局。事实上，

田 □ □
 $= \{ \quad | \quad \} = \{0 | 0\}$

大家可以立即看出，它并不是一个合法的数。在 On Numbers and Games 中，Conway 建议用符号 * 来表示这种特殊的棋局 $\{0 | 0\}$ 。棋局 * 扮演着很重要的角色，例如不难验证等式 $* + * = 0$ 成立。Conway 用符号 ↑ 表示棋局 $\{0 | *$ ，容易看出这是一个左玩家必胜的棋局。可以证明，存在等式 $\{0 | \uparrow\} = \uparrow + \uparrow + *$ 。可见，这个里面的水非常地深，继续挖掘下去还大有文章可作。在书中，Conway 还分析了很多其他的游戏以及其他棋局运算，感兴趣的朋友不妨深入阅读下去。下面这张漂亮的图片来自 Winning Ways for Your Mathematical Plays 一书，直观地展示了各种各样的棋盘及其对应的数。



(五) 后记

经常听人问，什么是数学之美？最终发现，还是数学大神 Paul Erdős 的回答犀利：“问什么是数学之美，就像问什么是贝多芬第九交响曲之美一样。”没人能给你讲明白，除非你自己去经历它。

几年前，我就对超实数这一话题特别感兴趣，在看到一个个奇迹般的同构时，感觉全身热血沸腾，沉浸在数学的结构之美当中。最近终于下定决心，写完了整个超实数理论的介绍。这可能是我写过的单篇文章中耗时最长的了，期间我曾经纠结了很多细节问题，一次又一次地被搞晕，一次又一次地陷入哲学思考。我希望能够通过这篇长文，把我这段时间学到的和想到的分享给更多的人，让更多的人理解到数学的乐趣。开篇的博弈游戏出自 [Hackenstrings, and the 0.999... ?= 1 FAQ](#)（链接已死，但可在 [archive.org](#) 上查看），对超实数的描述参考了 Gretchen Grimm 的 [An Introduction to Surreal Numbers](#) 和 Claus Tøndering 的 [Surreal Numbers – An Introduction](#)，多米诺游戏的例子出自 [On Numbers and Games](#)，但对棋局的形式化整理来自于自己的思考，这里面可能存在大量不严谨的地方，希望网友们提出。

A.2 生成函数



This page is (maybe un-) intentionally (almost) blank.

生成函数

By Yuekai Jia

【定义】

对于一个无穷项的序列 $\{g_0, g_1, g_2, \dots\}$ ，用如下方法把它们和一个无穷级数联系起来：

$$G(x) = g_0 + g_1x + g_2x^2 + \dots = \sum_{i=0}^{\infty} g_i x^i$$

则称函数 $G(x)$ 为序列 $\{g_i\}$ 的生成函数(也叫母函数)。

这里的 x 无实际意义，可以根据不同的情况取任意值。当级数收敛时，可得到生成函数的闭形式。例如，序列 $\{1,1,1,1,\dots\}$ 的生成函数为

$$G(x) = 1 + x + x^2 + x^3 + \dots = \sum_{i=0}^{\infty} x^i$$

当取 $0 < x < 1$ 时，闭形式为 $\frac{1}{1-x}$ 。

举一个简单的例子：在4个不同的数中选出 n 个数的不同方案数有多少？显然答案为 $\binom{4}{n}$ 。用一个序列表示 n 从0开始取的答案，即 $\{1,4,6,4,1,0,0,\dots\}$ ，这个序列的生成函数就是 $1 + 4x + 6x^2 + 4x^3 + x^4$ ，即 $(1+x)^4$ 。

【基本操作】

- 放缩：

$$cG(x) = \sum_{i=0}^{\infty} cg_i x^i \Leftrightarrow \{cg_0, cg_1, cg_2, \dots\}$$

$$G(cx) = \sum_{i=0}^{\infty} c^i g_i x^i \Leftrightarrow \{g_0, cg_1, c^2 g_2, \dots\}$$

- 加减法:

$$F(x) \pm G(x) = \sum_{i=0}^{\infty} (f_i + g_i)x^i \Leftrightarrow \{f_0 + g_0, f_1 + g_1, f_2 + g_2, \dots\}$$

- 右移:

将序列向右平移 k 位，并在前 k 位补0:

$$x^k G(x) = \sum_{i=0}^{\infty} g_i x^{i+k} \Leftrightarrow \underbrace{\{0, 0, \dots, 0\}}_{k \uparrow 0}, g_0, g_1, g_2, \dots$$

- 求导:

将序列第 i 项乘 i ，并左移一位:

$$G'(x) = \sum_{i=1}^{\infty} i g_i x^{i-1} \Leftrightarrow \{g_1, 2g_2, 3g_3, 4g_4, \dots\}$$

- 积分:

将序列第 i 项除以 $i+1$ ，并右移一位:

$$\int G(x) dx = \sum_{i=0}^{\infty} \frac{1}{i+1} g_i x^{i+1} \Leftrightarrow \{0, g_0, \frac{1}{2}g_1, \frac{1}{3}g_2, \frac{1}{4}g_3, \dots\}$$

- 乘法:

$$\begin{aligned} F(x)G(x) &= (f_0 + f_1 x + f_2 x^2 + \dots)(g_0 + g_1 x + g_2 x^2 + \dots) \\ &= \sum_{i=0}^{\infty} \left(\sum_{k=0}^i f_k g_{i-k} \right) x^i \\ &\Leftrightarrow \{h_i\} \end{aligned}$$

其中序列 $\{h_i\}$ 为序列 $\{f_i\}$ 和 $\{g_i\}$ 的卷积。

【常见序列的生成函数】

- 序列{1,1,1,1, ... }:

$$G(x) = 1 + x + x^2 + x^3 + \dots = \frac{1}{1-x}$$

- 序列{1,2,3,4, ... }:

$$\begin{aligned} G(x) &= 1 + 2x + 3x^2 + 4x^3 + \dots \\ &= (1 + x + x^2 + x^3)' \\ &= \left(\frac{1}{1-x}\right)' \\ &= \frac{1}{(1-x)^2} \end{aligned}$$

- 组合数序列{1, $\binom{n}{1}$, $\binom{n}{2}$, $\binom{n}{3}$, ... }:

$$\begin{aligned} G(x) &= 1 + \binom{n}{1}x + \binom{n}{2}x^2 + \binom{n}{3}x^3 + \dots \\ &= \sum_{i=0}^{\infty} \binom{n}{i}x^i \\ &= (1+x)^n \end{aligned}$$

- Fibonacci 数列{0,1,1,2,3,5,8, ... }:

$$G(x) = x + x^2 + 2x^3 + 3x^4 + 5x^5 + 8x^6 + \dots$$

右移一位, 得

$$xG(x) = x^2 + x^3 + 2x^4 + 3x^5 + 5x^6 + 8x^7 + \dots$$

两式相加, 得

$$\begin{aligned} G(x) + xG(x) &= x + 2x^2 + 3x^3 + 5x^4 + 8x^5 + 13x^6 + \dots \\ &= \frac{G(x)}{x} - 1 \end{aligned}$$

所以有

$$(1 - x - x^2)G(x) = x$$

$$G(x) = \frac{x}{1 - x - x^2}$$

- 常见生成函数对应的序列:

考虑这样一个生成函数:

$$\begin{aligned} G(x) &= (1 + x + x^2 + x^3 + \dots)^n \\ &= \frac{1}{(1-x)^n} \end{aligned}$$

$$= (1-x)^{-n}$$

它所对应的序列是什么，即它的展开式中每一项的系数。

来看一个定理：

■ **扩展二项式定理：**

对于任意实数 c ，对任意满足 $0 \leq |x| < |y|$ 的 x, y ，有：

$$(x+y)^c = \sum_{i=0}^{\infty} \binom{c}{i} x^i y^{c-i}$$

其中

$$\binom{c}{i} = \frac{c(c-1)(c-2)\cdots(c-i+1)}{i!}$$

当 c 为正整数 n ，对于 $i > n$ ， $\binom{n}{i} = 0$ ，上式变为

$$(x+y)^n = \sum_{i=0}^n \binom{n}{i} x^i y^{n-i}$$

即普通的二项式定理。

当 c 为负整数 $-n$ ，有

$$\begin{aligned} \binom{-n}{i} &= \frac{-n(-n-1)(-n-2)\cdots(-n-i+1)}{i!} \\ &= (-1)^i \frac{n(n+1)(n+2)\cdots(n+i-1)}{i!} \\ &= (-1)^i \binom{n+i-1}{i} \end{aligned}$$

于是就有

$$\begin{aligned} G(x) &= \frac{1}{(1-x)^n} = (1-x)^{-n} = \sum_{i=0}^{\infty} \binom{n+i-1}{i} x^i \\ &\Leftrightarrow \{1, \binom{n}{1}, \binom{n+1}{2}, \binom{n+2}{3}, \dots\} \end{aligned}$$

由此得到一个重要的公式：

$$\begin{aligned} \frac{1}{(1-cx)^n} &= 1 + c \binom{n}{1} x + c^2 \binom{n+1}{2} x^2 + c^3 \binom{n+2}{3} x^3 + \dots \\ &= \sum_{i=0}^{\infty} c^i \binom{n+i-1}{i} x^i \end{aligned}$$

根据这个生成函数可以轻松解决一个经典问题：求不定方程 $x_1 + x_2 + \dots + x_n = m$ 的非负整数解的个数。答案等于它的生成函数 $G(x) = (1 + x + x^2 + x^3 + \dots)^n$ 中 x^m 项的系数，即 $\binom{n+m-1}{m}$ 。

【指类型生成函数】

在有些排列组合问题中，不仅要考虑选出了哪些元素，还要考虑选出的元素是怎样排列的。上述一般的生成函数适用于组合问题，而下面的指类型生成函数适用于排列问题。

● 定义：

对于一个无穷项的序列 $\{g_0, g_1, g_2, \dots\}$ ，它的指类型生成函数为：

$$G_e(x) = g_0 + g_1 \frac{x}{1!} + g_2 \frac{x^2}{2!} + g_3 \frac{x^3}{3!} + \dots = \sum_{i=0}^{\infty} g_i \frac{x^i}{i!}$$

例如，序列{1,1,1,1, ... }的指类型生成函数为

$$G_e(x) = 1 + \frac{x}{1!} + \frac{x^2}{2!} + \frac{x^3}{3!} + \dots$$

根据 Taylor 展开式， $G_e(x) = e^x$ 。

● 基本操作：

类似地，指类型生成函数的左移(右移)可以通过求导(积分)来做，这里不作展开，只讨论两个指类型生成函数的乘积：

$$\begin{aligned} F_e(x)G_e(x) &= \left(f_0 + f_1 \frac{x}{1!} + f_2 \frac{x^2}{2!} + \dots \right) \left(g_0 + g_1 \frac{x}{1!} + g_2 \frac{x^2}{2!} + \dots \right) \\ &= \sum_{i=0}^{\infty} \left(\sum_{k=0}^i \frac{f_k}{k!} \frac{g_{i-k}}{(i-k)!} i! \right) \frac{x^i}{i!} \\ &= \sum_{i=0}^{\infty} \left(\sum_{k=0}^i \binom{i}{k} f_k g_{i-k} \right) \frac{x^i}{i!} \end{aligned}$$

相当于在做卷积时，乘上了一个系数 $\binom{i}{k}$ ，所以指类型生成函数适用于排列问题。

【应用】

- 求递推数列的通项公式:

- 例 1:

求递推数列 $\{f_n\}$ 的通项公式, 其中 $f_0 = 1, f_1 = -2, f_n = 5f_{n-1} - 6f_{n-2}$ 。

设 $F(x)$ 为数列 $\{f_n\}$ 的生成函数, 有:

$$\begin{aligned} F(x) &= f_0 + f_1x + f_2x^2 + f_3x^3 + \dots \\ -5xF(x) &= -5f_0x - 5f_1x^2 - 5f_2x^3 - 5f_3x^4 - \dots \\ 6x^2F(x) &= 6f_0x^2 + 6f_1x^3 + 6f_2x^4 + 6f_3x^5 + \dots \end{aligned}$$

相加得

$$\begin{aligned} (1 - 5x + 6x^2)F(x) &= \\ f_0 + (f_1 - 5f_0)x + (f_2 - 5f_1 + 6f_0)x^2 + (f_3 - 5f_2 + 6f_1)x^3 + \dots + \\ (f_n - 5f_{n-1} + 6f_{n-2})x^n + \dots \end{aligned}$$

由已知条件得

$$\begin{aligned} (1 - 5x + 6x^2)F(x) &= f_0 + (f_1 - 5f_0)x = 1 - 7x \\ F(x) &= \frac{1 - 7x}{1 - 5x + 6x^2} \end{aligned}$$

由于 $1 - 5x + 6x^2 = (1 - 2x)(1 - 3x)$, 所以一定有

$$\frac{1 - 7x}{1 - 5x + 6x^2} = \frac{c_1}{1 - 2x} + \frac{c_2}{1 - 3x}$$

解得 $c_1 = 5, c_2 = -4$,

$$F(x) = \frac{1 - 7x}{1 - 5x + 6x^2} = \frac{5}{1 - 2x} - \frac{4}{1 - 3x}$$

而

$$\frac{5}{1 - 2x} = 5(1 + 2x + 2^2x^2 + \dots + 2^n x^n + \dots)$$

$$\frac{4}{1 - 3x} = 4(1 + 3x + 3^2x^2 + \dots + 3^n x^n + \dots)$$

$$F(x) = 5(1 + 2x + 2^2x^2 + \dots) - 4(1 + 3x + 3^2x^2 + \dots)$$

$$= 1 + (-2)x + (-16)x^2 + \dots + (5 \times 2^n - 4 \times 3^n)x^n + \dots$$

所以数列 $\{f_n\}$ 的通项公式为 $f_n = 5 \times 2^n - 4 \times 3^n$ 。

- 生成函数形式的 Pólya 定理:

Pólya 计数法是解决在置换群作用下本质不同的染色方案的好方法。但是它只能求出总方案, 无法针对某一种特殊的染色状态求方案。用生成函数形式的 Pólya 定理可以就能很好地解决。

■ 例 2:

有一个长度为4的环形序列，在每个位置涂上黑色或白色，对于某一种黑白状态(如 3 黑 1 白等)，求本质不同的方案数。

由于只考虑黑色和白色的个数，所以，例如方案 wbbw 就可以用两个变量的单项式 b^2w^2 表示。

如果不考虑循环同构，则所有方案可以表示成 $(b+w)^4 = b^4 + 4b^3w + 6b^2w^2 + 4bw^3 + w^4$ ，即全黑、全白有 1 种，3 白 1 黑、3 黑 1 白有 4 种，2 黑 2 白有 6 种。

如果只求本质不同的方案数，则只要用 Pólya 定理

$$\frac{1}{|G|} \sum_{f \in G} m^{c(f)}$$

即可。

现在把两者结合，即把 Pólya 定理中的 $m^{c(f)}$ 替换成

$$(b_1 + b_2 + \dots + b_m)^{c_1(f)}(b_1^2 + b_2^2 + \dots + b_m^2)^{c_2(f)} \dots (b_1^n + b_2^n + \dots + b_m^n)^{c_n(f)}$$

其中 $c_i(f)$ 表示置换 f 的长度为 i 的循环节个数， b_i 为 m 个颜色变量。于是生成函数形式的 Pólya 定理就可以表示成

$$P(b_1, b_2, \dots, b_m) = \frac{1}{|G|} \sum_{f \in G} \prod_{i=0}^n \left(\sum_{j=0}^m b_j^i \right)^{c_i(f)}$$

对于本题，共有 4 个置换，分别为 (1)(2)(3)(4), (2 3 4 1), (1 3)(2 4), (4 1 2 3)，两种颜色变量为 b, w ，用生成函数形式的 Pólya 定理，得

$$\begin{aligned} P(b, w) &= \frac{1}{4} ((b+w)^4 + (b^4 + w^4) + (b^2 + w^2)^2 + (b^4 + w^4)) \\ &= b^4 + b^3w + 2b^2w^2 + bw^3 + w^4 \end{aligned}$$

即全黑、全白有 1 种，3 白 1 黑、3 黑 1 白有 1 种，2 黑 2 白有 2 种。

● 其他例题：

■ 例 3:

有 4 种 BG，颜色分别为红、黄、蓝、绿，每种 BG 个数无限。现要求选出 n 只 BG，要求红色 BG 只能有偶数只，黄色 BG 个数要为 5 的倍数，蓝色 BG 最多选出 4 只，绿色 BG 要么不选，要么只选 1 只。求方案数。

$$n \leq 10^{1000000}$$

常规算法似乎很难处理，考虑用生成函数来做。对于每种颜色的 BG，可选个数的集合分别为 $\{0, 2, 4, 6, \dots\}$, $\{0, 5, 10, 15, \dots\}$, $\{0, 1, 2, 3, 4\}$, $\{0, 1\}$ 写成生成函数的形式，即

$$1 + x^2 + x^4 + x^6 + \dots = \frac{1}{1 - x^2}$$

$$1 + x^5 + x^{10} + x^{15} + \dots = \frac{1}{1 - x^5}$$

$$1 + x + x^2 + x^3 + x^4 = \frac{1 - x^5}{1 - x}$$

$$1 + x$$

相乘后得到原问题的生成函数

$$G(x) = \frac{1}{1 - x^2} \frac{1}{1 - x^5} \frac{1 - x^5}{1 - x} (1 + x)$$

$$= \frac{1}{(1 - x)^2}$$

$$= 1 + 2x + 3x^2 + 4x^3 + 5x^4 + \dots$$

所以原问题的答案即为 $n + 1$ 。

■ 例 4:

求满足下列条件的十进制 n 位数的个数：

1. 每位数字都是奇数；
2. 其中数字 1 和数字 3 只出现偶数次。

答案模 1 000 000 007。

$$n \leq 10^{1000000}$$

常规算法似乎很难处理，考虑用生成函数来做。由于是一个十进制数，所以要考虑选出的数之间的排列情况，要用指类型生成函数来做。可用数字只有 1, 3, 5, 7, 9 这 5 种，每种的个数集合也已知了，原问题的指类型生成函数为

$$G_e(x) = \left(1 + \frac{x^2}{2!} + \frac{x^4}{4!} + \dots\right)^2 \left(1 + \frac{x}{1!} + \frac{x^2}{2!} + \dots\right)^3$$

$$= \left(\frac{e^x + e^{-x}}{2}\right)^2 e^{3x}$$

$$= \left(\frac{e^{2x} + 1}{2}\right)^2 e^x$$

$$= \frac{1}{4}(e^{5x} + 2e^{3x} + e^x)$$

$$= \frac{1}{4} \left(\sum_{i=0}^{\infty} 5^i \frac{x^i}{i!} + 2 \sum_{i=0}^{\infty} 3^i \frac{x^i}{i!} + \sum_{i=0}^{\infty} \frac{x^i}{i!} \right)$$

$$= \sum_{i=0}^{\infty} \frac{1}{4} (5^i + 2 \times 3^i + 1) \frac{x^i}{i!}$$

所以原问题的答案即为 $\frac{1}{4}(5^n + 2 \times 3^n + 1)$ 。

【作业】

1. 认真理解上述性质和例题;
2. 用生成函数推导 Fibonacci 数列的通项公式;
3. 求如下递推数列 $\{f_n\}$ 的通项公式:

$$f_n = \begin{cases} 1 & n = 0 \\ \sum_{i=0}^{n-1} f_i f_{n-i-1} & n > 0 \end{cases}$$

4. <http://www.lydsy.com/JudgeOnline/problem.php?id=3142>
5. <http://www.spoj.com/problems/TSUM/>
6. 参考资料 1 第 11 页 Chocolate;
7. 参考资料 1 第 17 页 Polygon;

如果想得到更多关于生成函数的知识, 请阅读参考资料中的某些部分。

【参考资料】

1. 国家集训队 2009 论文《母函数的性质及应用》
2. 冬令营 2008 周源讲稿
3. 《组合数学》
4. 《具体数学》
5. 《算法艺术与信息学竞赛》

A.3 多项式操作



This page is (maybe un-) intentionally (almost) blank.

【多项式】多项式逆元/开方/取模/多点求值/插值/牛顿迭代/对数/exp/幂

多项式求逆元

已知多项式 $F(x)$, 求 $F(x)$ 在保留前 n 项 (当然 n 要是 2 的次幂) 的情况下的逆元 $G(x)$, 也就是:

$$F(x)G(x) \equiv 1 \pmod{x^n}$$

首先, 如果 $n = 1$, 那么直接就是常数项的逆元,

如果 $n > 1$, 那么怎么办?

设: $G'(x)$ 使得 $F(x)G'(x) \equiv 1 \pmod{x^{n/2}}$, 且我们已知 $G'(x)$
第一个式子可以变成:

$$F(x)G(x) \equiv 1 \pmod{x^{n/2}}$$

(把模数开了个方, 依旧成立)

把两个式子相减:

$$F(x)(G(x) - G'(x)) \equiv 0 \pmod{x^{n/2}}$$

$$G(x) - G'(x) \equiv 0 \pmod{x^{n/2}}$$

同时平方: (当然模数也平方依旧成立)

$$(G(x) - G'(x))^2 \equiv 0 \pmod{x^n}$$

$$G^2(x) + G'^2(x) - 2G(x)G'(x) \equiv 0 \pmod{x^n}$$

两边同时乘上 $F(x)$, 消掉部分 $G(x)$:

$$G(x) + G'^2(x)F(x) - 2G'(x)F(x) \equiv 0 \pmod{x^n}$$

$$G(x) \equiv 2G'(x) - G'^2(x)F(x) \pmod{x^n}$$

那么, $G(x)$ 就可以快速求出了,

(同时发现, 只要常数项有逆元, 这个多项式就有逆元)

复杂度: $T(n) = T(n/2) + O(n \log(n)) = O(n \log(n))$

多项式开方

多项式的开方同样可以以这种方法做出来，

已知多项式 $F(x)$ ，求 $F(x)$ 在保留前 n 项（当然 n 要是2的次幂）的情况下平方根 $G(x)$ ，也就是：

$$G^2(x) \equiv F(x) \pmod{x^n}$$

首先，如果 $n = 1$ ，那么直接就是常数项的开方，可以暴力枚举，也可以用二次项剩余（CZY的二次剩余 Cipolla算法学习小记），

对于 $n > 1$ 的情况，

设： $G'(x)$ 使得 $G'(x)^2 \equiv F(x) \pmod{x^{n/2}}$ ，且我们已知 $G'(x)$ ，
(把平方写在后面好看QuQ)

第一个式子可以变成：

$$G^2(x) \equiv F(x) \pmod{x^{n/2}}$$

(把模数开了个方，依旧成立)

把两个式子相减：

$$G^2(x) - G'(x)^2 \equiv 0 \pmod{x^{n/2}}$$

因式分解：

$$(G(x) + G'(x))(G(x) - G'(x)) \equiv 0 \pmod{x^{n/2}}$$

可得 $G(x)$ 有两个解（平方嘛），讨论 $G(x) - G'(x) \equiv 0 \pmod{x^{n/2}}$ 的情况，

$$G(x) - G'(x) \equiv 0 \pmod{x^{n/2}}$$

(历史总是惊人的相识)

同时平方：(当然模数也平方依旧成立)

$$(G(x) - G'(x))^2 \equiv 0 \pmod{x^n}$$

$$G^2(x) + G'(x)^2 - 2G(x)G'(x) \equiv 0 \pmod{x^n}$$

因为： $G^2(x) \equiv F(x) \pmod{x^n}$

$$F(x) + G'(x)^2 - 2G(x)G'(x) \equiv 0 \pmod{x^n}$$

$$G(x) \equiv \frac{F(x) + G'(x)^2}{2G'(x)} \pmod{x^n}$$

那么, $G(x)$ 就可以快速求出了,

(同时发现, 只要常数项是二次项剩余且有逆元, 这个多项式就可以开方)

复杂度: $T(n) = T(n/2) + 2 * O(n \log(n)) = O(n \log(n))$

多项式取模

已知 $A(x), B(x)$, 求 $D(x) = A(x) \pmod{B(x)}$,

令 $A(x) = B(x)C(x) + D(x)$

设 $n = A(x)$ 的次数, $m = B(x)$ 的次数, 显然有 $m \leq n$,

上面的等式两边同时乘上 x^n 得:

$$x^n A\left(\frac{1}{x}\right) = x^m B\left(\frac{1}{x}\right) x^{n-m} C\left(\frac{1}{x}\right) + x^n D\left(\frac{1}{x}\right)$$

($\frac{1}{x}$ 的作用相当于是把多项式头尾翻转一下)

设 $A'(x) = x^n A\left(\frac{1}{x}\right)$, $B'(x) = x^m B\left(\frac{1}{x}\right)$, $C'(x) = x^{n-m} C\left(\frac{1}{x}\right)$, $D'(x) = x^n D\left(\frac{1}{x}\right)$

可以发现, 经过翻转后,

$D'(x)$ 中只有次数 $\in [n - m + 1, n]$ 的项是有效的, 其他项均为 0,

$A'(x)$ 中次数 $\in [0, n]$ 的项是有效的,

$B'(x)$ 中次数 $\in [0, m]$ 的项是有效的,

$C'(x)$ 中次数 $\in [0, n - m]$ 的项是有效的,

现在再对原来的等式 $\pmod{x^{n-m+1}}$, 也就是

$$A'(x) = B'(x)C'(x) + D'(x) \pmod{x^{n-m+1}}$$

这样 $D'(x)$ 这一项就能模掉了, 也就是:

$$A'(x) = B'(x)C'(x) \pmod{x^{n-m+1}}$$

$$\frac{A'(x)}{B'(x)} = C'(x) \pmod{x^{n-m+1}}$$

因为 $C'(x)$ 的次数范围刚好在模以内, 所以就可以直接求出 $C'(x)$, 变换得 $C(x)$, 求出了 $C(x)$, 剩下直接减就好了

$$D(x) = A(x) - B(x)C(x)$$

复杂度: $O(n \log(n))$

多项式多点求值

已知多项式 $F(x)$, 给出 a_1, a_2, \dots, a_n , 要求 $F(a_1), F(a_2), \dots, F(a_n)$

先抛出一个显然的结论: $F(a_1) = F(x) \bmod (x + a_1)$,

(意思是: 多项式 $F(x)$ 模多项式 $(x + a_1)$ 的余数就是当 $x = a_1$ 时 $F(x)$ 的值)

有这个结论就好办了,

设多项式 $C_{l,r}(x) = \prod_{i=l}^r (x + a_i)$, $G_{l,r}(x) = F(x) \bmod C_{l,r}(x)$,

考虑分治求解,

显然的: $G_{l,mid}(x) = G_{l,r}(x) \bmod C_{l,r}(x)$, 右边同理,

这样当 $l=r$ 时 $G_{l,l}(x)$ 就是 $F(a_l)$ 的值了,

做两遍分治FFT,

复杂度: $O(n \log^2(n))$

多项式插值

给出 $a_1, b_1, a_2, b_2, \dots, a_k, b_k$, 要求多项式 $F(x)$ 满足 $F(a_i) = b_i$,

考虑使用拉格朗日插值法,

$$F(x) = \sum_{i=1}^k b_i \left(\prod_{1 \leq j \leq k, j \neq i} \frac{x - a_j}{a_i - a_j} \right)$$

将式子拆成两部分:

$$F(x) = \sum_{i=1}^k b_i \left(\prod_{1 \leq j \leq k, j \neq i} \frac{1}{a_i - a_j} \right) \left(\prod_{1 \leq j \leq k, j \neq i} (x - a_j) \right)$$

先做前面那一部分: (先取个倒数方便书写)

设 $G_i = \prod_{1 \leq j \leq k, j \neq i} (a_i - a_j)$, $M(x) = \prod_{j=1}^k (a_i - a_j)$

显然有 $M(a_i) = 0$,

有:

$$G_i = \lim_{x \rightarrow a_i} \frac{M(x) - M(a_i)}{x - a_i}$$

我们发现这个东西相当于 $M(x)$ 在 $x = a_i$ 处的导数, 于是有:

$$G_i = \lim_{x \rightarrow a_i} \frac{M(x) - M(a_i)}{x - a_i} = M'(a_i)$$

所以对 $M'(x)$ 做一次多点求值即可,

这个东西还可以用洛必达法则证明,

设 $f(x) = x - a_i$

已知有 $\lim_{x \rightarrow a_i} M(x) = 0, \lim_{x \rightarrow a_i} f(x) = 0$

所以:

$$\lim_{x \rightarrow a_i} \frac{M(x)}{f(x)} = \lim_{x \rightarrow a_i} \frac{M'(x)}{f'(x)} = M'(a_i)$$

现在的原始变成了:

$$F(x) = \sum_{i=1}^k b_i G_i \left(\prod_{1 \leq j \leq k, j \neq i} (x - a_j) \right)$$

这个东西可以直接使用分治FFT实现, 分治的时候记录两个多项式分别表示是否已经空缺了一位,

复杂度: $O(n \log^2(n))$

多点求值+几遍分治FFT常数爆炸

多项式牛顿迭代

已知多项式 $G(x)$, 要求多项式 F 使得 $G(F) = 0 \pmod{x^n}$

前置技能:

泰勒展开

对于多项式 $f(x)$ 它在 x_0 处的泰勒展开为:

$$f(x) = f(x_0) + \frac{f'(x_0)}{1!}(x - x_0) + \frac{f''(x_0)}{2!}(x - x_0)^2 + \dots$$

考虑倍增求 F

现在要求的 F 是 $\pmod{x^{2n}}$ 的, 假设我们已经求出了 F_0 表示 $\pmod{x^n}$ 时的答案,

把 $G(F)$ 在 F_0 处展开:

$$G(F) = G(F_0) + G'(F_0)(F - F_0) + \frac{1}{2} G''(F_0)(F - F_0)^2 \pmod{x^{2n}}$$

我们注意到是在 $\pmod{x^{2n}}$ 意义下的, 而从第3项开始最低次项的指数均大于 $2n$, 所以可以直接省去, 于

是：

$$G(F) = G(F_0) + G'(F_0)(F - F_0) \mod x^{2n}$$

又因为 $G(F) = 0 \mod x^{2n}$,

$$\begin{aligned} 0 &= G(F_0) + G'(F_0)F - G'(F_0)F_0 \mod x^{2n} \\ F &= F_0 - \frac{G(F_0)}{G'(F_0)} \end{aligned}$$

多项式求对数

给出多项式 $G(x)$, 要求 $F(x)$ 使得 $F(x) = \ln(G(x)) \mod x^n$

对两边同时求导, 最后再积分回来,

有:

$$(\ln(G(x)))' = \frac{G'(x)}{G(x)}$$

所以

$$F(x) = \int \frac{G'(x)}{G(x)} dx$$

复杂度: $O(n \log(n))$

多项式求EXP

给出多项式 $G(x)$, 求 $F(x)$ 满足 $F(x) = e^{G(x)}$,

考虑使用牛顿迭代,

设多项式 $g(x) = \ln(x) - G(x)$, 即 $g(F) = 0$

$$F = F_0 - \frac{g(F_0)}{g'(F_0)}$$

又因为

$$g'(F_0) = (\ln(F_0))' - (A)' = \frac{1}{F_0}$$

所以:

$$F = F_0 - F_0(\ln(F_0) - A)$$

复杂度: $O(n \log(n))$

多项式求幂

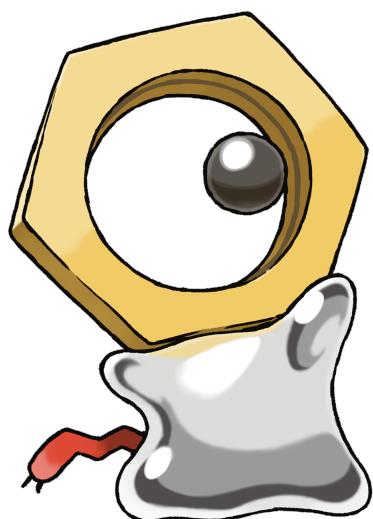
给出多项式 $F(x)$, 求 $F(x)^k$,

$$F(x)^k = e^{k \ln(F(x))}$$

这样如果 k 不为整数也能求了

复杂度: $O(n \log(n))$

A.4 多项式操作



This page is (maybe un-) intentionally (almost) blank.

【解题报告】网络流24题

1. 飞行员配对方案问题

链接: [「网络流 24 题」搭配飞行员](#)

题意: 每架飞机需要两个驾驶员, 一个正驾驶员和一个副驾驶员。由于种种原因, 有些驾驶员不能在同一架飞机上飞行, 问如何搭配驾驶员才能使出航的飞机最多。两个正驾驶员或两个副驾驶员都不能同机飞行。

分析: 二分图最大匹配课题。

View Code

2. 太空飞行计划问题

链接: [「网络流 24 题」太空飞行计划](#)

题意: 现已确定了一个可供选择的实验集合 $E = \{E_1, E_2, \dots, E_m\}$, 和进行这些实验需要使用的全部仪器的集合 $I = \{I_1, I_2, \dots, I_n\}$ 。实验 E_j 需要用到的仪器是 I 的子集 $R_j \subseteq I$ 。配置仪器 I_k 的费用为 c_k 美元。实验 E_j 的赞助商已同意为该实验结果支付 p_j 美元。对于给定的实验和仪器配置情况, 找出净收益最大的试验计划。

分析: 最大权闭合子图。从 S 向每个实验连一条容量为 p_i 的边, 每个实验向所需要的仪器连一条容量为 \inf 的边, 每个仪器向 T 连一条容量为 c_i 的边。答案为 $\sum p_i - \text{maxflow}$ 。

View Code

3. 最小路径覆盖问题

链接: [「网络流 24 题」最小路径覆盖](#)

题意: 给定有向图 $G = (V, E)$ 。设 P 是 G 的一个简单路 (顶点不相交) 的集合。如果 V 中每个顶点恰好在 P 的一条路上, 则称 P 是 G 的一个路径覆盖。 P 中路径可以从 V 的任何一个顶点开始, 长度也是任意的, 特别地, 可以为 0。 G 的最小路径覆盖是 G 的所含路径条数最少的路径覆盖。求一个有向无环图 G 的最小路径覆盖。

分析: 二分图最大匹配。将每个点拆为两个点, 在新图中对应连边。二分图的每一个合法匹配都可以视为一种路径覆盖的方式, 路径条数为总点数-匹配数。最小不相交路径覆盖即为总点数-二分图最大匹配。(建图方式仅限DAG)

View Code

4. 魔术球问题

链接: [「网络流 24 题」魔术球](#)

题意: 假设有 n 根柱子, 按下述规则在这 n 根柱子中依次放入编号为 $1, 2, 3, 4, \dots$ 的球: 每次只能在某根柱子的最上面放球; 在同一根柱子中, 任何 2 个相邻球的编号之和为完全平方数。试计算出在 n 根柱子上最多能放多少个球。

分析: 二分图最大匹配。题目限制了在放置好每个 x 之前, 需先放置好 $1, 2, 3, \dots, x-1$ 的球。考虑枚举答案 x , 建立关于 $1, 2, 3, \dots, x$ 的图: 若 $i < j$ 且 $i + j$ 为完全平方数, 则连接边 (i, j) 。原图是一个DAG, 问题转化为求DAG的最小不相交路径覆盖。按顺序枚举 x , 当最小路径覆盖数大于 n 时结束。

View Code

5. 圆桌问题

链接: [「网络流 24 题」圆桌聚餐](#)

题意: 假设有来自 n 个不同单位的代表参加一次国际会议。每个单位的代表数分别为 r_i 。会议餐厅共有 m 张餐桌, 每张餐桌可容纳 c_i 个代表就餐。为了使代表们充分交流, 希望从同一个单位来的代表不在同一个餐桌就餐。试给出满足要求的代表就餐方案。

分析: 最大流。从 S 向每个单位连一条容量为人数的边, 从餐桌向 T 连接一条容量为餐桌容量的边, 从单位向每个餐桌连一条容量为 1 的边。直接跑最大流求解。(或者是贪心+线段树, 餐桌和人数都从大到小排序, 每次安排时按餐桌剩余容量从大到小安排。为

维护单调性，对于最后一段相等的区间，需要减线段树上靠后的部分。)

View Code

6.最长递增子序列问题

链接: [「网络流 24 题」最长递增子序列](#)

题意: 给定正整数序列 $x_1 \sim x_n$ ，以下递增均为非严格递增。计算其最长递增子序列的长度 s 。计算从给定的序列中最多可取出多少个长度为 s 的递增子序列。如果允许在取出的序列中多次使用 x_1 和 x_n ，则从给定序列中最多可取出多少个长度为 s 的递增子序列。

分析: 令 f_i 表示以第 i 位开头的最长递增子序列长度，可用dp求解。若 $f_i = s$ ，则从 S 向 i 连一条容量为 1 的边；若 $f_i = 1$ ，则从 i 向 T 连一条容量为 1 的边；限定每个点只选一次，拆点，连一条容量为 1 的边；若 $i < j$ 且 $x_i \leq x_j$ 且 $f_i = f_j + 1$ ，则从 i 向 j 连一条容量为 1 的边。直接跑最大流求解即可。回答第三问时，把对应边的限制改成 \inf ，再跑一次最大流。

View Code

7.试题库问题

链接: [「网络流 24 题」试题库](#)

题意: 假设一个试题库中有 n 道试题。每道试题都标明了所属类别。同一道题可能有多个类别属性。现要从题库中抽取 m 道题组成试卷。并要求试卷包含指定类型的试题。试设计一个满足要求的组卷算法。

分析: 最大流。从 S 向每种类别连一条容量为需求的边，从题目向 T 连接一条容量为 1 的边，从每个类别向题目连一条容量为 1 的边。直接跑最大流求解即可。

View Code

8.机器人路径规划问题

链接: [PowerOJ1743: 机器人路径规划问题](#)

题意: 给定树状路径上的起点 s 和终点 t ，机器人要从 s 运动到 t 。树状路径上有若干可移动的障碍物。由于路径狭窄，任何时刻在路径的任何位置不能同时容纳 2 个物体。每一步可以将障碍物或机器人移到相邻的空顶点上。设计一个有效算法用最少移动次数使机器人从 s 运动到 t 。编程任务：对于给定的树，以及障碍物在树中的分布情况，计算机器人的最少移动次数。

分析: 网络流的复杂度是 $O(n^9)$ 的，ImmortalCO给出了一种 $O(n^6)$ 的动态规划解法。参考[《机器人路径规划问题\(TMP1R\)题解》](#)。

9.方格取数问题

链接: [「网络流 24 题」方格取数](#)

题意: 在一个有 $m \times n$ 个方格的棋盘中，每个方格中有一个正整数。现要从方格中取数，使任意 2 个数所在方格没有公共边，且取出的数的总和最大。试设计一个满足要求的取数算法。

分析: 最大点权独立集。先将棋盘黑白染色，从 S 向每个黑点连一条容量为黑点数值的边，从白点向 T 连接一条容量为白点数值的边，从每个黑点向白点连一条容量为 \inf 的边。答案为总价值-最小割。

10.餐巾计划问题

链接: [「网络流 24 题」餐巾计划](#)

题意: 一个餐厅在相继的 n 天里，每天需用的餐巾数不尽相同。假设第 i 天需要 r_i 块餐巾。餐厅可以购买新的餐巾，每块餐巾的费用为 P 分；或者把旧餐巾送到快洗部，洗一块需 M 天，其费用为 F 分；或者送到慢洗部，洗一块需 N 天，其费用为 S 分 ($S < F$)。每天结束时，餐厅必须决定将多少块脏的餐巾送到快洗部，多少块餐巾送到慢洗部，以及多少块保存起来延期送洗。但是每天洗好的餐巾和购买的新餐巾数之和，要满足当天的需求量。试设计一个算法为餐厅合理地安排好 n 天中餐巾使用计划，使总的花费最小。

分析：把每一天都拆成一对点 x_i 和 y_i ， x_i 表示脏的餐巾， y_i 表示干净的餐巾。从 S 向 y_i 连一条容量为 \inf 费用为 P 的边，代表购买决策；从 y_i 向 T 连一条容量为 r_i 费用为 0 的边，代表每天需求；从 S 向 x_i 连一条容量为 r_i 费用为 0 的边，代表每天剩余的未洗餐巾；从 x_i 向 x_{i+1} 连一条容量为 \inf 费用为 0 的边，代表将脏餐巾屯到下一天；从 x_i 向 y_{i+m} 连一条容量为 \inf 费用为 F 的边，代表快洗决策；从 x_i 向 y_i+n 连一条容量为 \inf 费用为 S 的边，代表慢洗决策。直接跑最小费用最大流即可。

11.航空路线问题

链接：[「网络流 24 题」航空路线问题](#)

题意：给定一张航空图，图中顶点代表城市，边代表两个城市间的直通航线。现要求找出一条满足下述限制条件的且途经城市最多的旅行路线：从最西端城市出发，单向从西向东途经若干城市到达最东端城市，然后再单向从东向西飞回起点（可途经若干城市）；除起点城市外，任何城市只能访问一次。对于给定的航空图，试设计一个算法找出一条满足要求的最佳航空旅行路线。

分析：问题转化为求两条不相交的路径且路径之和最长。考虑拆点，连一条容量为 1 费用为 -1 的边（1 和 n 的容量为 2）。跑最小费用最大流求解，若最大流不等于 2 则无解。

12.软件补丁问题

链接：[「网络流 24 题」软件补丁](#)

题意：某公司发现其研制的一个软件中有 n 个错误，随即为该软件发放了一批共 m 个补丁程序。对于每一个补丁 i ，都有 2 个与之相应的错误集合 $B_1(i)$ 和 $B_2(i)$ ，使得仅当软件包含 $B_1(i)$ 中的所有错误，而不包含 $B_2(i)$ 中的任何错误时，才可以使用补丁 i 。补丁 i 将修复软件中的某些错误 $F_1(i)$ ，而同时加入另一些错误 $F_2(i)$ 。另外，每个补丁都耗费一定的时间。试设计一个算法，利用公司提供的 m 个补丁程序将原软件修复成一个没有错误的软件，并使修复后的软件耗时最少。

分析：将bug状态压成二进制之后跑最短路。

13.星际转移问题

链接：[「网络流 24 题」星际转移](#)

题意：现有 n 个太空站位于地球与月球之间，且有 m 艘公共交通太空船在其间来回穿梭。每个太空站可容纳无限多的人，而每艘太空船 i 只可容纳 H_i 个人。每艘太空船将周期性地停靠一系列的太空站，例如：1, 3, 4 表示该太空船将周期性地停靠太空站 134134134…… 每一艘太空船从一个太空站驶往任一太空站耗时均为 1。人们只能在太空船停靠太空站（或月球、地球）时上、下船。初始时所有人在地球上，太空船全在初始站。试设计一个算法，找出让所有人尽快地全部转移到月球上的运输方案。

分析：建立分层图。从 S 向每一天的地球连一条容量为 \inf 的边；从每一天的月球向 T 连一条容量为 \inf 的边；从每一天的节点向第二天的对应节点连一条容量为 \inf 的边；对于每一艘飞船，从每一天的位置向第二天的位置连一条容量为 H_i 的边。枚举天数建图，跑最大流直到不小于总人数即可。

14.孤岛营救问题

链接：[「网络流 24 题」孤岛营救问题](#)

题意：迷宫的外形是一个长方形，其南北方向被划分为 n 行，东西方向被划分为 m 列，于是整个迷宫被划分为 $n \times m$ 个单元。南北或东西方向相邻的 2 个单元之间可能互通，也可能有一扇锁着的门，或者是一堵不可逾越的墙。迷宫中有一些单元存放着钥匙，并且所有的门被分成 p 类，打开同一类的门的钥匙相同，不同类门的钥匙不同。大兵瑞恩被关押在 (n, m) 单元里，并已经昏迷。迷宫只有一个入口，在 $(1, 1)$ 单元。麦克从一个单元移动到另一个相邻单元的时间为 1，拿取所在单元的钥匙的时间以及用钥匙开门的时间可忽略不计。试设计一个算法，帮助麦克以最快的方式到达瑞恩所在单元，营救大兵瑞恩。

分析：将获得钥匙的状态压成二进制，对每一个状态建一层图，跑分层图最短路。

15.汽车加油驾驶问题

链接：[「网络流 24 题」汽车加油行驶问题](#)

题意：给定一个 $N \times N$ 的方形网格，设其左上角为起点，坐标为 $(1, 1)$ ， X 轴向右为正， Y 轴向下为正，每个方格边长为 1。一辆汽车从起点出发驶向右下角终点 (N, N) 。在若干个网格交叉点处，设置了油库。汽车在行驶过程中应遵守如下规则：汽车只能沿网格边行驶，装满油后能行驶 K 条网格边；出发时汽车已装满油，在起点与终点处不设油库；汽车经过一条网格边时，若其 X 坐标

或 Y 坐标减小，则应付费用 B ，否则免付费用；汽车在行驶过程中遇油库则应加满油并付加油费用 A ；在需要时可在网格点处增设油库，并付增设油库费用 C （不含加油费用 A ）。求出汽车从起点出发到达终点的一条所付费用最少的行驶路线。

分析：令 $f(i, j, k)$ 为点 (i, j) 剩余油量为 k 的最小费用，跑最短路。

16. 数字梯形问题

链接：[「网络流 24 题」数字梯形](#)

题意：给定一个由 n 行数字组成的数字梯形。梯形的第一行有 m 个数字。从梯形的顶部的 m 个数字开始，在每个数字处可以沿左下或右下方向移动，形成一条从梯形的顶至底的路径。分别遵守以下规则：从梯形的顶至底的 m 条路径互不相交；从梯形的顶至底的 m 条路径仅在数字结点处相交；从梯形的顶至底的 m 条路径允许在数字结点相交或边相交。将按照三个规则计算出的最大数字总和输出。

分析：从 S 向顶部的每一个点连一条容量为 1 费用为 0 的边。规则一：拆点之后跑费用流；规则二：不拆点，跑费用流；规则三：把边的容量改为 \inf 。

17. 运输问题

链接：[「网络流 24 题」运输问题](#)

题意：公司有 m 个仓库和 n 个零售商店。第 i 个仓库有 a_i 个单位的货物；第 j 个零售商店需要 b_j 个单位的货物。货物供需平衡，即 $\sum_{i=1}^m a_i = \sum_{j=1}^n b_j$ 。从第 i 个仓库运送每单位货物到第 j 个零售商店的费用为 c_{ij} 。试设计一个将仓库中所有货物运送到零售商店的运输方案，使总运输费用最少。

分析：从 S 向仓库连一条容量为 a_i 费用为 0 的边；从商店向 T 连一条容量为 b_j 费用为 0 的边；从仓库向商店连一条容量为 \inf 费用为 c_{ij} 的边，跑费用流即可。

18. 分配工作问题

链接：[「网络流 24 题」分配问题](#)

题意：有 n 件工作要分配给 n 个人做。第 i 个人做第 j 件工作产生的效益为 c_{ij} 。试设计一个将 n 件工作分配给 n 个人做的分配方案，使产生的总效益最大。

分析：二分图最大权匹配，常规建图跑费用流即可。

19. 负载平衡问题

链接：[「网络流 24 题」负载平衡](#)

题意：公司有 n 个沿铁路运输线环形排列的仓库，每个仓库存储的货物数量不等。如何用最少搬运量可以使 n 个仓库的库存数量相同。搬运货物时，只能在相邻的仓库之间搬运。

分析：由于总量是固定的，我们可以得出每个仓库的最终储量，令 a_i 表示原有储量-最终储量。对于每一个仓库，拆成两个点 x_i 和 y_i ，一个代表供应，一个代表需求。若 $a_i > 0$ ，从 S 向 x_i 连一条容量为 a_i 费用为 0 的边；若 $a_i < 0$ ，从 y_i 向 T 连一条容量为 $-a_i$ 费用为 0 的边；对于两个相邻的顶点 j ，从 x_i 分别向 x_j 和 y_j 连一条容量为 \inf 费用为 1 的边。跑最小费用最大流即可。

20. 深海机器人问题

链接：[「网络流 24 题」深海机器人问题](#)

题意：潜艇内有多个深海机器人。潜艇到达深海海底后，深海机器人将离开潜艇向预定目标移动。深海机器人在移动中还必须沿途采集海底生物标本，沿途生物标本由最先遇到它的深海机器人完成采集。每条预定路径上的生物标本的价值是已知的，而且生物标本只能被采集一次。本题限定深海机器人只能从其出发位置沿着向北或向东的方向移动，而且多个深海机器人可以在同一时间占据同一位置。用一个 $P \times Q$ 网格表示深海机器人的可移动位置。西南角的坐标为 $(0, 0)$ ，东北角的坐标为 (Q, P) 。给定每个深海机器人的出发位置和目标位置，以及每条网格边上生物标本的价值。计算深海机器人的最优移动方案，使深海机器人到达目的地后，采集到的生物标本的总价值最高。

分析：每条边的贡献只能计算一次，考虑拆边：一条边容量为 1 费用为价值，一条边容量为 inf 费用为 0。跑最小费用最大流即可。

21.最长 k 可重区间集问题

链接：[「网络流 24 题」最长 k 可重区间集](#)

题意：给定实直线 L 上 n 个开区间组成的集合 I ，和一个正整数 k ，试设计一个算法，从开区间集合 I 中选取出开区间集合 $S \subseteq I$ ，使得在实直线 L 的任何一点 x ， S 中包含点 x 的开区间个数不超过 k 。且 $\sum_{z \in S} |z|$ 达到最大。这样的集合 S 称为开区间集合 I 的最长 k 可重区间集。 $\sum_{z \in S} |z|$ 称为最长 k 可重区间集的长度。对于给定的开区间集合 I 和正整数 k ，计算开区间集合 I 的最长 k 可重区间集的长度。

分析：离散化区间的所有端点，从小到大为 x_1, x_2, \dots, x_m 。从 S 向 1 连一条容量为 k 费用为 0 的边；从 m 向 T 连一条容量为 k 费用为 0 的边；从 i 向 $i+1$ 连一条容量为 k 费用为 0 的边；对于每一个区间，从其左端点的对应顶点向右端点的对应顶点连一条容量为 1 费用为区间长度的边。跑最小费用最大流即可。

22.最大 k 可重线段集问题

链接：[「网络流 24 题」最长 k 可重线段集问题](#)

题意：给定平面 xoy 上 n 个开线段组成的集合 I ，和一个正整数 k 。从开线段集合 I 中选取出开线段集合 $S \in I$ ，使得在 x 轴上的任何一点 p ， S 中与直线 $x = p$ 相交的开线段个数不超过 k ，且 $\sum_{z \in S} |z|$ 达到最大。这样的集合 S 称为开线段集合 I 的最长 k 可重线段集的长度。对于任何开线段 z ，设其端点坐标为 (x_0, y_0) 和 (x_1, y_1) ，则开线段 z 的长度 $|z|$ 定义为： $|z| = \lfloor \sqrt{(x_1 - x_0)^2 + (y_1 - y_0)^2} \rfloor$ 。对于给定的开线段集合 I 和正整数 k ，计算开线段集合 I 的最长 k 可重线段集的长度。

分析：和上一道题基本一致，为了避免线段与 x 轴垂直的情况，需要将所有的端点坐标 $\times 2$ ，然后左端点 -1 。

23.火星探险问题

链接：[「网络流 24 题」火星探险问题](#)

题意：登陆舱着陆后，探测车将离开登陆舱向先期到达的传送器方向移动。探测车在移动中还必须采集岩石标本，每一块岩石标本由最先遇到它的探测车完成采集。每块岩石标本只能被采集一次。岩石标本被采集后，其他探测车可以从原来岩石标本所在处通过。探测车不能通过有障碍的地面。本题限定探测车只能从登陆处沿着向南或向东的方向朝传送器移动，而且多个探测车可以在同一时间占据同一位置。如果某个探测车在到达传送器以前不能继续前进，则该车所采集的岩石标本将全部损失。用一个 $P \times Q$ 网格表示登陆舱与传送器之间的位置。登陆舱的位置在 (X_1, Y_1) 处，传送器的位置在 (X_P, Y_Q) 处。给定每个位置的状态，计算探测车的最优移动方案，使到达传送器的探测车的数量最多，而且探测车采集到的岩石标本的数量最多。

分析：将原图中的每一个顶点拆成两个点：若其为岩石顶点，从 i 向 i' 连一条容量为 1 费用为 1 的边；若其为非障碍点，从 i 向 i' 连一条容量为 inf 费用为 0 的边；对于相邻的点 j ，从 i' 向 j 连一条容量为 inf 费用为 0 的边。从 S 向 (X_1, Y_1) 连一条容量为探测车数量费用为 0 的边；从 (X_P, Y_Q) 向 T 连一条容量为探测车数量费用为 0 的边。跑最小费用最大流即可。

24.骑士共存问题

链接：[「网络流 24 题」骑士共存问题](#)

题意：在一个 $n \times n$ 个方格的国际象棋棋盘上，马（骑士）可以攻击的棋盘方格为“日”字。棋盘上某些方格设置了障碍，骑士不得进入。对于给定的 $n \times n$ 个方格的国际象棋棋盘和障碍标志，计算棋盘上最多可以放置多少个骑士，使得它们彼此互不攻击。

分析：二分图最大独立集。将棋盘黑白染色，从 S 向每一个可用的黑格连一条容量为 1 的边，从每一个可用的白格向 T 连一条容量为 1 的边，每一个黑格向可以攻击的白格连一条容量为 inf 的边。答案为总的可用格数-最大流。



—— “还有谁不会飞？” ——