

1 Data_Structure

1.1 hull_dynamic

```

1 const ll is_query = -(1LL<<62);
2 struct Line {
3     ll m, b;
4     mutable function<const Line*> succ;
5     bool operator<(const Line& rhs) const {
6         if (rhs.b != is_query) return m < rhs.m;
7         const Line* s = succ();
8         if (!s) return 0;
9         ll x = rhs.m;
10        return b - s->b < (s->m - m) * x;
11    }
12 };
13 // Upper envelope, erase cannot be done.
14 // Even if you do erase, the popped lines
15 // are gone, it won't be a correct hull.
16 struct HullDynamic : public multiset<Line> {
17     bool bad(iterator y) {
18         auto z = next(y);
19         if (y == begin()) {
20             if (z == end()) return 0;
21             return y->m == z->m && y->b <= z->b;
22         }
23         auto x = prev(y);
24         if (z == end()) return y->m == x->m && y
25             ->b <= x->b;
26         return 1.0 * (x->b - y->b) * (z->m - y->m)
27             >= 1.0 * (y->b - z->b) * (y->m - x->m)
28             );
29     }
30     void insert_line(ll m, ll b) {
31         auto y = insert({ m, b });
32         y->succ = [=] { return next(y) == end()
33             ? 0 : &*next(y); };
34         if (bad(y)) { erase(y); return; }
35         while (next(y) != end() && bad(next(y)))
36             erase(next(y));
37         while (y != begin() && bad(prev(y)))
38             erase(prev(y));
39     }
40     ll eval(ll x) {
41         auto l = *lower_bound((Line) { x,
42             is_query });
43         return l.m * x + l.b;
44     }
45 };

```

1.2 Treap

```

1 #include <iostream>
2 #include <algorithm>
3 #include <cstdlib>
4 using namespace std;
5 const int INF = 9e9;
6 struct Node{
7     int val, pri, size, mi, tag;
8     bool rev;

```

```

9     Node *l, *r;
10    Node(){
11        Node(int v):val(v),pri(rand()),size(1),
12            rev(0),mi(v),tag(0){ l = r = NULL; }
13    void down();
14    void up();
15 }*root;
16 int Size(Node *o){ return o? o->size:0; }
17 int Min(Node *o){ return o? o->mi:INF; }
18 int Val(Node *o){ return o? o->val:-1; }
19 void Node::down(){
20     if(tag){
21         val += tag;
22         mi += tag;
23         if(l) l->tag += tag;
24         if(r) r->tag += tag;
25         tag = 0;
26     }
27     if(rev){
28         swap(l,r);
29         if(l) l->rev ^= 1;
30         if(r) r->rev ^= 1;
31         rev = 0;
32     }
33 }
34 void Node::up(){
35     if(l) l->down();
36     if(r) r->down();
37     size = 1 + Size(l) + Size(r);
38     mi = std::min( min(Min(l), Min(r)),val )
39         ;
40 }
41 void print(Node *o){
42     if(o){
43         print(o->l);
44         printf("%d ", o->val);
45         print(o->r);
46     }
47 }
48 Node* merge(Node* a, Node* b){
49     if(!a || !b) return a? a:b;
50     if(a->pri < b->pri){
51         a->down();
52         a->r = merge(a->r, b);
53         a->up();
54         return a;
55     }else{
56         b->down();
57         b->l = merge(a, b->l);
58         b->up();
59         return b;
60     }
61 }
62 void spilt(Node *o, Node *a, Node *b, int
63     k){
64     if(!o) a = b = NULL;
65     else{
66         o->down();
67         if(Size(o->l)>=k){
68             b = o;

```

```

69             spilt(o->l, a, b->l, k);
70         }
71         else{
72             a = o;
73             spilt(o->r, a->r, b, k-Size(o->l
74                 )-1);
75         }
76         o->up();
77     }
78 }
79 void Insert(Node *o, int k, int v){
80     if(!o) o = new Node(v);
81     else{
82         Node* tmp = new Node(v);
83         Node *a, *b;
84         spilt(o, a, b, k);
85         o = merge(merge(a,tmp), b);
86     }
87 }
88 void Del(Node *o, int k){
89     if(!o) return;
90     else{
91         Node *a, *b, *c;
92         spilt(o, a, b, k);
93         spilt(a, a, c, k-1);
94         o = merge(a, b);
95     }
96 }
97 int Min(Node *o, int x, int y){
98     if(!o) return 0;
99     else{
100        Node *a, *b, *c;
101        spilt(o, a, b, y);
102        spilt(a, a, c, x-1);
103        if(c->mi < o->mi)
104            return c->mi;
105        else
106            return o->mi;
107    }
108 }
109 void Add(Node *o, int x, int y, int v){
110     if(!o) return;
111     Node *a, *b, *c;
112     spilt(o, a, b, y);
113     spilt(a, a, c, x-1);
114     if(c->tag < v)
115         o = merge(merge(a,c), b);
116 }
117 void Reverse(Node *o, int x, int y){
118     if(!o) return;
119     Node *a, *b, *c;
120     spilt(o, a, b, y); // a b c
121     spilt(a, a, c, x-1);
122     if(c->rev ^= 1)
123         o = merge(merge(a,c), b);
124 }
125 void Rotate(Node *o, int x, int y, int t){
126     if(!o) return;
127     Node *a, *b, *c;
128     spilt(o, a, b, y);
129     spilt(a, a, c, x-1);
130     if(c->rev ^= 1)
131         o = merge(merge(a,c), b);
132 }
133 void Rotate(Node *o, int x, int y, int t){
134     if(!o) return;
135     Node *a, *b, *c;

```

```

136     spilt(o, a, b, y);
137     spilt(a, a, c, x-1);
138 }
139 Node *d, *e;
140 t %= (y-x+1);
141 if(t<0) t = y-x+1+t;
142 spilt(c,d,e, Size(c)-t);
143 c = merge(e, d);
144 o = merge(merge(a,c),b);
145 }
146 /*
147 ADD x y D: Add D to each number in sub-
148 sequence {Ax ... Ay}. For example,
149 performing "ADD 2 4 1" on {1, 2, 3, 4,
150 5} results in {1, 3, 4, 5, 5}
151 REVERSE x y: reverse the sub-sequence {Ax
152 ... Ay}. For example, performing "
153 REVERSE 2 4" on {1, 2, 3, 4, 5} results
154 in {1, 4, 3, 2, 5}
155 REVOLVE x y T: rotate sub-sequence {Ax ...
156 Ay} T times. For example, performing "
157 REVOLVE 2 4 2" on {1, 2, 3, 4, 5}
158 results in {1, 3, 4, 2, 5}
159 INSERT x P: insert P after Ax. For example,
160 performing "INSERT 2 4" on {1, 2, 3, 4,
161 5} results in {1, 2, 4, 3, 4, 5}
162 DELETE x: delete Ax. For example, performing
163 "DELETE 2" on {1, 2, 3, 4, 5} results
164 in {1, 3, 4, 5}
165 MIN x y: query the participant what is the
166 minimum number in sub-sequence {Ax ...
167 Ay}. For example, the correct answer to
168 "MIN 2 4" on {1, 2, 3, 4, 5} is 2
169 */
170 int main()
171 {
172     int n;
173     while(scanf("%d",&n)==1){
174         root = NULL;
175         for(int i=0; i<n; i++){
176             scanf("%d",&a);
177             root = merge(root, new Node(a));
178         }
179         int m, x, y, c;
180         char s[20];
181         scanf("%d", &m);
182         for(int i=0; i<m; i++){
183             scanf("%s", s);
184             if(s[0]=='A'){
185                 scanf("%d%d",&x,&y,&c);
186                 Add(root, x, y, c);
187             }
188             else if(s[0]=='R' && s[3]=='E'){
189                 scanf("%d%d",&x,&y);
190                 Reverse(root, x, y);
191             }
192             else if(s[0]=='I'){
193                 scanf("%d%d",&x,&y);
194                 Insert(root, x, y, c);
195             }
196             else if(s[0]=='D'){
197                 scanf("%d%d",&x,&y);
198                 Delete(root, x, y);
199             }
200             else if(s[0]=='M'){
201                 scanf("%d%d",&x,&y);
202                 printf("%d\n", Min(root, x, y));
203             }
204         }
205     }
206 }

```

```

187     else if(s[0]=='D'){
188         scanf("%d",&x);
189         Del(root, x);
190         n--;
191     }
192     else{
193         scanf("%d%d",&x,&y);
194         printf("%d\n", Min(root, x,
195             y));
196     }
197 }
198 return 0;
199 }

```

1.3 undo_disjoint_set

```

1 struct DisjointSet {
2     // save() is like recursive
3     // undo() is like return
4     int n, fa[MXN], sz[MXN];
5     vector<pair<int*,int>> h;
6     vector<int> sp;
7     void init(int tn) {
8         n=tn;
9         for (int i=0; i<n; i++) sz[fa[i]=i]=1;
10        sp.clear(); h.clear();
11    }
12    void assign(int *k, int v) {
13        h.pb({k, *k});
14        *k=v;
15    }
16    void save() { sp.pb(SZ(h)); }
17    void undo() {
18        assert(!sp.empty());
19        int last=sp.back(); sp.pop_back();
20        while (SZ(h)!=last) {
21            auto x=h.back(); h.pop_back();
22            *x.f=x.S;
23        }
24    }
25    int f(int x) {
26        while (fa[x]!=x) x=fa[x];
27        return x;
28    }
29    void uni(int x, int y) {
30        x=f(x); y=f(y);
31        if (x==y) return;
32        if (sz[x]<sz[y]) swap(x, y);
33        assign(&sz[x], sz[x]+sz[y]);
34        assign(&fa[y], x);
35    }
36 } djs;

```

1.4 整體二分

```

1 void totBS(int L, int R, vector<Item> M){
2     if(Q.empty()) return; //維護全域B陣列
3     if(L==R) 整個M的答案=r, return;

```

```

4     int mid = (L+R)/2;
5     vector<Item> mL, mR;
6     do_modify_B_with_divide(mid,M);
7     //讓B陣列在遞迴的時候只會保留[L~mid]的資訊
8     undo_modify_B(mid,M);
9     totBS(L,mid,mL);
10    totBS(mid+1,R,mR);
11 }

```

2 Flow

2.1 DFSflow

```

1 struct Edge{
2     int to, cap, rev;
3     Edge(int a, int b, int c) {
4         to = a; cap = b; rev = c;
5     }
6 };
7 // IMPOTANT, MAXV != MAXN
8 vector<Edge> G[MAXV];
9 int V, flow[MAXV];
10 void init(int _V){
11     V = _V;
12     for(int i=0; i<=V; i++) G[i].clear();
13 }
14 void add_edge(int f, int t, int c, bool
15     directed){
16     int s1 = G[f].size(), s2 = G[t].size();
17     G[f].push_back(Edge(t, c, s2));
18     G[t].push_back(Edge(f, c*!directed, s1));
19 }
20 int dfs(int v, int t) {
21     if(v == t) return flow[t];
22     for(Edge &e : G[v]){
23         if(e.cap==0 || flow[e.to]==-1)
24             continue;
25         flow[e.to] = min(flow[v], e.cap);
26         int f = dfs(e.to, t);
27         if (f!=0) {
28             e.cap -= f;
29             G[e.to][e.rev].cap += f;
30             return f;
31         }
32     }
33     return 0;
34 }
35 int max_flow(int s, int t){
36     int ans = 0, add = 0;
37     do {
38         fill(flow, flow+V+1, -1);
39         flow[s] = INF;
40         add = dfs(s, t);
41         ans += add;
42     } while (add != 0);
43     return ans;
44 }

```

2.2 Dinic

```

1 struct Edge{
2     int f, to, rev;
3     T c;
4     Edge(int _to, int _r, T _c):to(_to), rev(_r
5         ), c(_c){}
6 };
7 // IMPOTANT
8 // maxn is the number of vertices in the
9 // graph
10 // Not the N in the problem statement!!
11 vector<Edge> G[maxn];
12 int level[maxn], st, end, n;
13 int cur[maxn];
14 void init(int _n){
15     n = _n;
16     for(int i=0; i<=n; i++) G[i].clear();
17 }
18 void addEdge(int f, int t, T c, bool directed)
19 {
20     int r1 = G[f].size(), r2 = G[t].size();
21     G[f].push_back(Edge(t, r2, c));
22     G[t].push_back(Edge(f, r1, directed?0:c));
23 }
24 bool BFS(int s, int t){
25     queue<int> Q;
26     for(int i=0; i<=n; i++) level[i] = 0;
27     level[s] = 1;
28     Q.push(s);
29     while(!Q.empty()){
30         int x = Q.front(); Q.pop();
31         for(int i=0; i<G[x].size(); i++){
32             Edge e = G[x][i];
33             if(e.c==0 || level[e.to])
34                 continue;
35             level[e.to] = level[x] + 1;
36             Q.push(e.to);
37         }
38     }
39     return level[t]!=0;
40 }
41 T DFS(int s, T cur_flow){ // can't exceed c
42     if(s==end) return cur_flow;
43     T ans = 0, temp, total = 0;
44     for(int& i=cur[s]; i<G[s].size(); i++){
45         Edge &e = G[s][i];
46         if(e.c==0 || level[e.to]!=level[s
47             ]+1) continue;
48         temp = DFS(e.to, min(e.c, cur_flow))
49             ;
50         if(temp!=0){
51             e.c -= temp;
52             G[e.to][e.rev].c += temp;
53             cur_flow -= temp;
54             total += temp;
55             if(cur_flow==0) break;
56         }
57     }
58     return total;
59 }

```

```

58 }
59 T maxFlow(int s, int t){
60     /* If you want to incrementally doing
61        maxFlow,
62        you need to add the result manually.
63        This function returns difference in
64        that case. */
65     T ans = 0;
66     st = s, end = t;
67     while(BFS(s, t)){
68         while(true) {
69             memset(cur, 0, sizeof(cur));
70             T temp = DFS(s, INF);
71             if(temp==0) break;
72             ans += temp;
73         }
74     }
75     return ans;
76 }

```

2.3 min_cost_flow

```

1 // 0-based
2 #define fi first
3 #define se second
4 struct Edge {
5     int to, cap;
6     int cost, rev;
7 };
8 static const int MAXV = 605;
9 int V, E;
10 vector<Edge> G[MAXV];
11 void init(int _V) {
12     V=_V;
13     for (int i=0; i<=V; i++) G[i].clear();
14 }
15 void add_edge(int fr, int to, int cap, int
16     cost) {
17     int a = G[fr].size(), b = G[to].size();
18     G[fr].push_back({to, cap, cost, b});
19     G[to].push_back({fr, 0, -cost, a});
20 }
21 bool SPFA(int s, int t, int &ans_flow, int &
22     ans_cost) {
23     queue<int> que;
24     PII pre[MAXV];
25     int flow[MAXV], dist[MAXV];
26     bool inque[MAXV];
27     for (int i=0; i<=V; i++) {
28         dist[i]=INF;
29         inque[i]=false;
30     }
31     dist[s]=0;
32     flow[s]=INF;
33     inque[s]=true;
34     que.push(s);
35     while (!que.empty()) {
36         int v=que.front(); que.pop();
37         inque[v]=false;
38         for (int i=0; i<G[v].size(); i++) {

```

```

39     const Edge &e = G[v][i];
40     if (e.cap>0 && dist[v]+e.cost<
41         dist[e.to]) {
42         flow[e.to]=min(flow[v],e.cap
43             );
44         dist[e.to]=dist[v]+e.cost;
45         pre[e.to]={v,i};
46         if (!inque[e.to]) que.push(e
47             .to),inque[e.to]=true;
48     }
49 }
50 if (dist[t]==INF) return false;
51 //if (dist[t]>=0) return false;
52 // Add above line -> min cost > max flow
53 (priority)
54 // Without -> max flow > min cost
55
56 int v=t,f=flow[t];
57 ans_flow+=flow[t];
58 ans_cost+=(dist[t]*flow[t]);
59 while (v!=s) {
60     Edge &e = G[pre[v].fi][pre[v].se];
61     e.cap-=f;
62     G[v][e.rev].cap+=f;
63     v=pre[v].fi;
64 }
65 return true;
66
67 pair<int,int> min_cost_flow(int s, int t) {
68     int ans_flow=0, ans_cost=0;
69     while (SPFA(s,t,ans_flow,ans_cost));
70     return make_pair(ans_flow,ans_cost);
71 }

```

3 Geometry

3.1 circle

```

1  /* Common tangent, circle is a point c and
2     radius r */
3  void get_tangent(Point c, double r1, double
4     r2, vector<Line> &ans) {
5     double r = r2 - r1;
6     double z = c.x*c.x + c.y*c.y;
7     double d = z - r*r;
8     if (d < -EPS) return;
9     d = sqrt(abs(d));
10     Line l;
11     l.a = (c.x * r + c.y * d) / z;
12     l.b = (c.y * r - c.x * d) / z;
13     l.c = r1;
14     ans.push_back(l);
15 }
16 vector<Line> tangents(Circle a, Circle b) {
17     // Tangent line of two circles, may have
18     // 0, 1, 2, 3, 4, inf solutions
19     // In case 0 or inf (a = b), no line will
20     // be reported. Otherwise,
21     // this program always find 4 lines, even
22     // if some of them are the same.

```

```

18     vector<Line> ans;
19     for (int i=-1; i<=1; i+=2)
20         for (int j=-1; j<=1; j+=2)
21             get_tangent(b.c-a.c, a.r*i, b.r*
22                 j, ans);
23     for (size_t i=0; i<ans.size(); ++i)
24         ans[i].c -= ans[i].a * a.c.x + ans[i
25             ].b * a.c.y;
26     return ans;
27 }
28 // Circle-line intersection
29 double r, a, b, c; // input, circle:((0,0),r
30 ), line:ax+by+c=0
31 double x0 = -a*c/(a*a+b*b), y0 = -b*c/(a*a+b
32     *b);
33 if (c*c > r*r*(a*a+b*b)+EPS)
34     puts ("no points");
35 else if (abs (c*c - r*r*(a*a+b*b)) < EPS) {
36     puts ("1 point");
37     cout << x0 << " " << y0 << "\n";
38 }
39 else {
40     double d = r*r - c*c/(a*a+b*b);
41     double mult = sqrt (d / (a*a+b*b));
42     double ax, ay, bx, by;
43     ax = x0 + b * mult;
44     bx = x0 - b * mult;
45     ay = y0 - a * mult;
46     by = y0 + a * mult;
47     puts ("2 points");
48     cout << ax << " " << ay << "\n" << bx <<
49         " " << by << "\n";
50 }
51 // Circle-circle intersection
52 // Circle ((0,0),r1) and ((x2,y2),r2)
53 // Then reduce to circle 1 intersect with
54 // line Ax+By+C
55 // A=-2*x_2, B=-2*y_2, C=(x_2)^2+(y_2)^2+(
56 // r_1)^2-(r_2)^2
57 // Special case: two circle are the same =>
58 // inf points

```

3.2 convex_hull

```

1  void convex_hull(vector<Point> &ps, vector<
2     Point> &hull) {
3     // Find convex hull of ps, store in hull
4     vector<Point> &stk=hull;
5     stk.resize(ps.size()+1);
6     sort(ps.begin(),ps.end()); // Using x to
7     cmp, y secondary.
8     int t=-1; // top
9     for (int i=0;i<ps.size();i++) {
10         // cross<-EPS -> count collinear, cross<
11         EPS -> not
12         while (t>1&&(stk[t]-stk[t-1]).cross(ps[
13             i]-stk[t])<EPS) t--;
14         stk[++t]=ps[i];
15     }
16     int low=t;
17     for (int i=ps.size()-2;i>=0;i--) {
18         // cross<-EPS -> count collinear, cross<
19         EPS -> not

```

3.3 geometry

```

1  #include <bits/stdc++.h>
2  using namespace std;
3  const double PI=acos(-1);
4
5  struct Point {
6     double x,y;
7     double cross(const Point &v) const {
8         return x*v.y-y*v.x;
9     }
10     double dot(const Point &v) const {
11         return x*v.x+y*v.y;
12     }
13     Point normal() { // Normal vector to the
14         left
15         return {-y,x};
16     }
17     double angle(const Point &v) const {
18         // Angle from *this to v in [-pi,pi].
19         double ang = atan2(cross(v),dot(v));
20         return ang < 0 ? ang + PI * 2 : ang;
21     }
22     double getA()const{//angle to x-axis
23         T A=atan2(y,x);//<0 when exceed PI
24         if (A<=-PI/2)A+=PI*2;
25         return A;
26     }
27     Point rotate_about(double theta, const
28         Point &p) const {
29         // Rotate this point counterclockwise by
30         theta about p
31         double nx=x-p.x,ny=y-p.y;
32         return {nx*cos(theta)-ny*sin(theta)+p.x,
33             nx*sin(theta)+ny*cos(theta)+p.y};
34     }
35 }
36
37 struct Line {
38     // IMPORTANT, remember to transform
39     // between two-point form
40     // and normal form by yourself, some
41     // methods may need them.
42     Point p1,p2;
43     double a,b,c; // ax+by+c=0
44     Line(){}
45     void pton() {
46         a=p1.y-p2.y;
47         b=p2.x-p1.x;
48         c=-a*p1.x-b*p1.y;
49     }
50     int relation(const Point &p) {
51         // For line, 0 if point on line
52         // -1 if left, 1 if right
53         Point dir=p2-p1;

```

```

48     double crs=dir.cross(p-p1);
49     return crs==0?0:crs>0?-1:1;
50 }
51 Point normal() { // normal vector to the
52     left.
53     Point dir=p2-p1;
54     return {-dir.y,dir.x};
55 }
56 bool on_segment(const Point &p) {
57     // Point on segment
58     return relation(p)==0&&(p2-p).dot(p1-p)
59         <=0;
60 }
61 bool parallel(const Line &l) {
62     // Two line parallel
63     return (p2-p1).cross(l.p2-l.p1)==0;
64 }
65 bool equal(const Line &l) {
66     // Two line equal
67     return relation(l.p1)==0&&relation(l.p2)
68         ==0;
69 }
70 bool cross_seg(const Line &seg) {
71     // Line intersect segment
72     Point dir=p2-p1;
73     return dir.cross(seg.p1-p1)*dir.cross(
74         seg.p2-p1)<=0;
75 }
76 int seg_intersect(const Line &s) const{
77     // Two segment intersect
78     // 0 -> no, 1 -> one point, -1 ->
79     // infinity
80     Point dir=p2-p1, dir2=s.p2-s.p1;
81     double c1=dir.cross(s.p2-p1);
82     double c2=dir.cross(s.p1-p1);
83     double c3=dir2.cross(p2-s.p1);
84     double c4=dir2.cross(p1-s.p1);
85     if (c1==0&&c2==0) {
86         if ((s.p2-p1).dot(s.p1-p1)>0&&(s.p2-p2)
87             .dot(s.p1-p2)>0&&
88             (p1-s.p1).dot(p2-s.p1)>0&&(p1-s.p2)
89             .dot(p2-s.p2)>0)return 0;
90         if (p1==s.p1&&(p2-p1).dot(s.p2-p1)<=0)
91             return 1;
92         if (p1==s.p2&&(p2-p1).dot(s.p1-p1)<=0)
93             return 1;
94         if (p2==s.p1&&(p1-p2).dot(s.p2-p2)<=0)
95             return 1;
96         if (p2==s.p2&&(p1-p2).dot(s.p1-p2)<=0)
97             return 1;
98         return -1;
99     }else if (c1*c2<=0&&c3*c4<=0)return 1;
100     return 0;
101 }
102 Point intersection(Line l) {
103     // RE if d1.cross(d2) == 0 (parallel /
104     // coincide)
105     Point d1 = p2 - p1, d2 = l.p2 - l.p1;
106     double p1 + d1 * ((l.p1 - p1).cross(d2)
107         / d1.cross(d2));
108 }
109 Point seg_intersection(Line &s) const {
110     Point dir=p2-p1, dir2=s.p2-s.p1;
111     // pton(); L.pton();
112     double c1=dir.cross(s.p2-p1);
113     double c2=dir.cross(s.p1-p1);

```

```

101 double c3=dir2.cross(p2-s.p1);
102 double c4=dir2.cross(p1-s.p1);
103 if (c1==0&&c2==0) {
104     if(p1==s.p1&&(p2-p1).dot(s.p2-p1)<=0)
105         return p1;
106     if(p1==s.p2&&(p2-p1).dot(s.p1-p1)<=0)
107         return p2;
108     if(p2==s.p1&&(p1-p2).dot(s.p2-p2)<=0)
109         return p1;
110     if(p2==s.p2&&(p1-p2).dot(s.p1-p2)<=0)
111         return p2;
112 }else if(c1*c2<=0&&c3*c4<=0)return
113     line_intersection(s);
114 // Reaches here means either INF or NOT
115 // ANY
116 // Use seg_intersect to check OuO
117 return {1234,4321};
118 }
119 double dist(const Point &p, bool
120 is_segment) const {
121 // Point to Line/segment
122 Point dir=p2-p1,v=p-p1;
123 if (is_segment) {
124     if (dir.dot(v)<0) return v.len();
125     if ((p1-p2).dot(p-p2)<0) return (p-p2)
126         .len();
127 }
128 double d=abs(dir.cross(v))/dir.len();
129 return d;
130 }
131 };
132 template<typename T>
133 struct polygon{
134     polygon(){}
135     vector<point<T> > p; //counterclockwise
136     T area()const{
137         T ans=0;
138         for(int i=p.size()-1,j=0;j<(int)p.size()
139             ;i=j++){
140             ans+=p[i].cross(p[j]);
141         }
142         return ans/2;
143     }
144     point<T> center_of_mass()const{
145         T cx=0,cy=0,w=0;
146         for(int i=p.size()-1,j=0;j<(int)p.size()
147             ;i=j++){
148             T a=p[i].cross(p[j]);
149             cx+=(p[i].x+p[j].x)*a;
150             cy+=(p[i].y+p[j].y)*a;
151             w+=a;
152         }
153         return point<T>(cx/3/w,cy/3/w);
154     }
155     char ahas(const point<T>& t)const{//return
156         1 if in simple polygon, -1 if on, 0
157         if no.
158         bool c=0;
159         for(int i=0,j=p.size()-1;i<p.size();j=i
160             ++){
161             if(line<T>(p[i],p[j]).point_on_segment
162                 (t))return -1;
163             else if((p[i].y>t.y)!=p[j].y>t.y)&&
164                 t.x<p[j].x-p[i].x)*(t.y-p[i].y)/(p[j]
165                     .y-p[i].y)+p[i].x)
166                 c=!c;
167         }
168         return c;
169     }
170     char point_in_convex(const point<T>&x)
171     const{
172         int l=1,r=(int)p.size()-2;
173         while(l<r){//return 1 if in convex
174             polygon, -1 if on, 0 if no.
175             int mid=(l+r)/2;
176             T a1=(p[mid]-p[0]).cross(x-p[0]);
177             T a2=(p[mid+1]-p[0]).cross(x-p[0]);
178             if(a1>=0&&a2<=0){
179                 T res=(p[mid+1]-p[mid]).cross(x-p[
180                     mid]);
181                 return res>0?(res>=0?-1:0);
182             }else if(a1<0)r=mid-1;
183             else l=mid+1;
184         }
185         return 0;
186     }
187     vector<T> getA()const{//angle of each edge
188         to x-axis
189         vector<T>res; //must be increasing
190         for(size_t i=0;i<p.size();i++){
191             res.push_back((p[(i+1)%p.size()]-p[i])
192                 .getA());
193         }
194         return res;
195     }
196     bool line_intersect(const vector<T>&A,
197         const line<T> &l)const{//O(LogN)
198         int f1=upper_bound(A.begin(),A.end(),(l.
199             p1-l.p2).getA())-A.begin();
200         int f2=upper_bound(A.begin(),A.end(),(l.
201             p2-l.p1).getA())-A.begin();
202         return l.cross_seg(line<T>(p[f1],p[f2]))
203             ;
204     }
205     polygon cut(const line<T> &l)const{
206         polygon ans; //convex polygon cut by a
207         line, left side of the line is
208         remained.
209         for(int n=p.size(),i=n-1,j=0;j<n;i=j++){
210             if(l.ori(p[i])>=0){
211                 ans.p.push_back(p[i]);
212                 if(l.ori(p[j])<0)
213                     ans.p.push_back(l.
214                         line_intersection(line<T>(p[i]
215                             ,p[j])));
216             }else if(l.ori(p[j])>0)
217                 ans.p.push_back(l.line_intersection(
218                     line<T>(p[i],p[j])));
219             }
220             return ans;
221         }
222         static bool graham_cmp(const point<T>& a,
223             const point<T>& b){ //CMP for finding
224             hull
225             return (a.x<b.x)|| (a.x==b.x&&a.y<b.y);
226         }
227         void graham(vector<point<T> > &s){ //convex
228             hull
229             sort(s.begin(),s.end(),graham_cmp);
230             p.resize(s.size()+1);
231             int m=0;
232             for(size_t i=0;i<s.size();i++){
233                 while(m>=2&&(p[m]-p[m-2]).cross(s[i]
234                     )-p[m-2])<=0)--m;
235                 p[m++]=s[i];
236             }
237             for(int i=s.size()-2,t=m+1;i>0;--i){
238                 while(m>=2&&(p[m-1]-p[m-2]).cross(s[i]
239                     )-p[m-2])<=0)--m;
240                 p[m++]=s[i];
241             }
242             if(s.size()>1)--m;
243             p.resize(m);
244         }
245         T diameter(){
246             int n=p.size(),t=1;
247             T ans=0;p.push_back(p[0]);
248             for(int i=0;i<n;i++){
249                 point<T> now=p[i+1]-p[i];
250                 while(now.cross(p[t+1]-p[i])>now.cross
251                     (p[t]-p[i]))t=(t+1)%n;
252                 ans=max(ans,(p[i]-p[t]).abs2());
253             }
254             return p.pop_back(),ans;
255         }
256         T min_cover_rectangle(){ // find convex
257             hull before call this
258             int n=p.size(),t=1,r=1,l=1;
259             if(n<3)return 0;
260             T ans=1e99;p.push_back(p[0]);
261             for(int i=0;i<n;i++){
262                 point<T> now=p[i+1]-p[i];
263                 while(now.cross(p[t+1]-p[i])>now.cross
264                     (p[t]-p[i]))t=(t+1)%n;
265                 while(now.dot(p[l+1]-p[i])<=now.dot(p[
266                     l]-p[i]))l=(l+1)%n;
267                 T d=now.abs2();
268                 T tmp=now.cross(p[t]-p[i])*(now.dot(p[
269                     r]-p[i])-now.dot(p[l]-p[i]))/d;
270                 ans=min(ans,tmp);
271             }
272             return p.pop_back(),ans;
273         }
274         T dis2(polygon &p1){ //square of distance
275             of two convex polygon
276             vector<point<T> > &P=p,&Q=p1.p;
277             int n=P.size(),m=Q.size(),l=0,r=0;
278             for(int i=0;i<n;i++){
279                 if(P[i].y<P[l].y)l=i;
280                 for(int i=0;i<m;i++){
281                     if(Q[i].y<Q[r].y)r=i;
282                     P.push_back(P[0]),Q.push_back(Q[0]);
283                     T ans=1e99;
284                     for(int i=0;i<n;i++){
285                         while((P[l]-P[l+1]).cross(Q[r+1]-Q[r])
286                             <0)r=(r+1)%m;
287                         ans=min(ans,line<T>(P[l],P[l+1]).
288                             seg_dis2(line<T>(Q[r],Q[r+1])));
289                     }
290                     l=(l+1)%n;
291                 }
292             }
293             return P.pop_back(),Q.pop_back(),ans;
294         }
295         static char sign(const point<T>&t){
296             return (t.y==0?t.x:t.y)<0;
297         }
298         static bool angle_cmp(const line<T>& A,
299             const line<T>& B){
300             point<T> a=A.p2-A.p1,b=B.p2-B.p1;
301             return sign(a)<sign(b)|| (sign(a)==sign(b)
302                 )&&a.cross(b)>0;
303         }
304         int halfplane_intersection(vector<line<T>
305             > &s){
306             sort(s.begin(),s.end(),angle_cmp); //half
307             plane is left side of the line
308             int L,R,n=s.size();
309             vector<point<T> > px(n);
310             vector<line<T> > q(n);
311             q[L=R=0]=s[0];
312             for(int i=1;i<n;i++){
313                 while(L<R&&s[i].ori(px[R-1])<=0)--R;
314                 while(L<R&&s[i].ori(px[L])<=0)++L;
315                 q[++R]=s[i];
316                 if(q[R].parallel(q[R-1])){
317                     --R;
318                     if(q[R].ori(s[i].p1)>0)q[R]=s[i];
319                 }
320                 if(L<R)px[R-1]=q[R-1].
321                     line_intersection(q[R]);
322             }
323             while(L<R&&q[L].ori(px[R-1])<=0)--R;
324             p.clear();
325             if(R-L<=1)return 0;
326             px[R]=q[R].line_intersection(q[L]);
327             for(int i=L;i<R;i++)p.push_back(px[i]);
328             return R-L+1;
329         }
330     };
331     template<typename T>
332     struct triangle{
333         point<T> a,b,c;
334         triangle(){
335             triangle(const point<T> &a,const point<T>
336                 &b,const point<T> &c):a(a),b(b),c(c){}
337         }
338         T area()const{
339             T t=(b-a).cross(c-a)/2;
340             return t>0?t:-t;
341         }
342         point<T> barycenter()const{//center of
343             mass
344             return (a+b+c)/3;
345         }
346         point<T> circumcenter()const{//outer
347             center
348             static line<T> u,v;
349             u.p1=(a+b)/2;
350             u.p2=point<T>(u.p1.x-a.y+b.y,u.p1.y+a.x-
351                 b.x);
352             v.p1=(a+c)/2;
353             v.p2=point<T>(v.p1.x-a.y+c.y,v.p1.y+a.x-
354                 c.x);
355             return u.line_intersection(v);
356         }
357         point<T> incenter()const{//inner center
358             T A=sqrt((b-c).abs2()),B=sqrt((a-c).abs2
359                 ()),C=sqrt((a-b).abs2());
360             return point<T>(A*a.x+B*b.x+C*c.x,A*a.y+
361                 B*b.y+C*c.y)/(A+B+C);
362         }
363         point<T> perpencenter()const{//
364             perpendicular(?) center
365             return barycenter()*3-circumcenter()*2;
366         }
367     };

```


3.4 smallest_circle

```

1 using PT=point<T>; using CPT=const PT;
2 PT circumcenter(CPT &a,CPT &b,CPT &c){
3     PT u=b-a, v=c-a;
4     T c1=u.abs2()/2, c2=v.abs2()/2;
5     T d=u.cross(v);
6     return PT(a.x+(v.y*c1-u.y*c2)/d,a.y+(u.x*
7         c2-v.x*c1)/d);
8 }
9 void solve(PT p[],int n,PT &c,T &r2){
10     random_shuffle(p,p+n);
11     c=p[0]; r2=0; // c,r2 = ??,???
12     for(int i=1;i<n;i++){if((p[i]-c).abs2()>r2){
13         c=p[i]; r2=0;
14     }
15     for(int j=0;j<i;j++){if((p[j]-c).abs2()>r2){
16         c.x=(p[i].x+p[j].x)/2;
17         c.y=(p[i].y+p[j].y)/2;
18         r2=(p[j]-c).abs2();
19     }
20     for(int k=0;k<j;k++){if((p[k]-c).abs2()>r2){
21         c=circumcenter(p[i],p[j],p[k]);
22         r2=(p[i]-c).abs2();
23     }
24 }
25 }

```

3.5 最近點對

```

1 template<typename _IT=point<T>*>
2 T cloest_pair(_IT L, _IT R){
3     if(R-L <= 1) return INF;
4     _IT mid = L+(R-L)/2;
5     T x = mid->x;
6     T d = min(cloest_pair(L,mid),cloest_pair(
7         mid,R));
8     inplace_merge(L, mid, R, ycmp);
9     static vector<point> b; b.clear();
10    for(auto u=L;u<R;++u){
11        if((u->x-x)*(u->x-x)>=d) continue;
12        for(auto v=b.rbegin();v!=b.rend();++v){
13            T dx=u->x-v->x, dy=u->y-v->y;
14            if(dy*dy>=d) break;
15            d=min(d,dx*dx+dy*dy);
16        }
17        b.push_back(*u);
18    }
19    return d;
20 }
21 T closest_pair(vector<point<T>> &v){
22     sort(v.begin(),v.end(),xcmp);
23     return closest_pair(v.begin(),v.end());
24 }

```

4 Graph

4.1 3989_ 穩定婚姻

```

1 #include <bits/stdc++.h>
2
3 using namespace std;
4
5 const int maxn = 1100;
6
7 int manWant[maxn][maxn], nextW[maxn];
8 int women[maxn][maxn], order[maxn][maxn];
9 int wife[maxn], husband[maxn];
10 queue<int> singleDog;
11
12 void engage(int m, int w){
13     if(husband[w]!=0){
14         wife[ husband[w] ] = 0;
15         singleDog.push( husband[w] );
16         husband[w] = 0;
17     }
18     husband[w] = m;
19     wife[m] = w;
20     // cout << m << " --> " << w << endl;
21 }
22 int main()
23 {
24     int Time, n, cas = 0;
25     scanf("%d",&Time);
26
27     while(Time-- && scanf("%d",&n)==1){
28         for(int i=1; i<=n; i++){
29             for(int j=1; j<=n; j++) scanf("%d",
30                 &manWant[i][j]);
31             nextW[i] = 1;
32             wife[i] = 0;
33             singleDog.push(i);
34         }
35         for(int i=1; i<=n; i++){
36             for(int j=1; j<=n; j++){
37                 scanf("%d",&women[i][j]);
38                 order[i][ women[i][j] ] = j;
39             }
40             husband[i] = 0;
41         }
42         while(!singleDog.empty()){
43             int x = singleDog.front();
44             singleDog.pop();
45             // cout << x << endl;
46             int to = manWant[x][nextW[x]++];
47             if(husband[to]==0) engage(x, to);
48             ;
49             else if(order[to][husband[to]] >
50                 order[to][x]) engage(x, to);
51             ;
52             else singleDog.push(x);
53         }
54         if(cas++) printf("\n");
55         for(int i=1; i<=n; i++) printf("%d\n",
56             wife[i]);
57     }
58 }

```

4.2 blossom

```

1 struct Blossom {
2     #define MAXN 505 // Max solvable problem,
3     // DON'T CHANGE
4     // 1-based, IMPORTANT
5     vector<int> g[MAXN];
6     int parent[MAXN], match[MAXN], belong[MAXN],
7     state[MAXN];
8     int n;
9     int lca(int u, int v) {
10         static int cases = 0, used[MAXN] = {};
11         for (++cases; ; swap(u, v)) {
12             if (u == 0) continue;
13             if (used[u] == cases) return u;
14             used[u] = cases;
15             u = belong[parent[match[u]]];
16         }
17     }
18     void flower(int u, int v, int l, queue<int>
19         &q) {
20         while (belong[u] != l) {
21             parent[u] = v, v = match[u];
22             if (state[v] == 1)
23                 q.push(v), state[v] = 0;
24             belong[u] = belong[v] = l, u = parent[
25                 v];
26         }
27     }
28     bool bfs(int u) {
29         for (int i = 0; i <= n; i++)
30             belong[i] = i;
31         memset(state, -1, sizeof(state[0])*(n+1));
32         queue<int> q;
33         q.push(u), state[u] = 0;
34         while (!q.empty()) {
35             u = q.front(), q.pop();
36             for (int i = 0; i < g[u].size(); i++) {
37                 int v = g[u][i];
38                 if (state[v] == -1) {
39                     parent[v] = u, state[v] = 1;
40                     if (match[v] == 0) {
41                         for (int prev; u; v = prev, u =
42                             parent[v]) {
43                             prev = match[u];
44                             match[u] = v;
45                             match[v] = u;
46                         }
47                         return 1;
48                     }
49                     q.push(match[v]), state[match[v]]
50                         = 0;
51                 }
52                 else if (state[v] == 0 && belong[v]
53                     != belong[u]) {
54                     int l = lca(u, v);
55                     flower(v, u, l, q);
56                     flower(u, v, l, q);
57                 }
58             }
59         }
60         return 0;
61     }
62 }

```

```

55 }
56 int blossom() {
57     memset(parent, 0, sizeof(parent[0])*(n
58         +1));
59     memset(match, 0, sizeof(match[0])*(n+1));
60     ;
61     int ret = 0;
62     for (int i = 1; i <= n; i++) {
63         if (match[i] == 0 && bfs(i))
64             ret++;
65     }
66     return ret;
67 }
68 void addEdge(int x, int y) {
69     g[x].push_back(y), g[y].push_back(x);
70 }
71 void init(int _n) {
72     n = _n;
73     for (int i = 0; i <= n; i++)
74         g[i].clear();
75 }
76 }
77 }
78 }
79 }
80 }
81 }
82 }
83 }
84 }
85 }
86 }
87 }
88 }
89 }
90 }
91 }
92 }
93 }
94 }
95 }
96 }
97 }
98 }
99 }
100 }

```

4.3 KM

```

1 // Maximum Bipartite Weighted Matching (
2 // Perfect Match)
3 static const int MXN = 650;
4 static const int INF = 2147483647; // LL
5 int n, match[MXN], vx[MXN], vy[MXN];
6 int edge[MXN][MXN], lx[MXN], ly[MXN], slack[MXN];
7 // ^^^^ LL
8 void init(int _n){
9     n = _n;
10    for(int i=0; i<n; i++) for(int j=0; j<n; j
11        ++){
12        edge[i][j] = 0;
13    }
14    void addEdge(int x, int y, int w) // LL
15    { edge[x][y] = w; }
16    bool DFS(int x){
17        vx[x] = 1;
18        for (int y=0; y<n; y++){
19            if (vy[y]) continue;
20            if (lx[x]+ly[y] > edge[x][y]){
21                slack[y]=min(slack[y], lx[x]+ly[y]-
22                    edge[x][y]);
23            }
24            else {
25                vy[y] = 1;
26                if (match[y] == -1 || DFS(match[y]))
27                    { match[y] = x; return true; }
28            }
29        }
30        return false;
31    }
32    int solve(){
33        fill(match,match+n,-1);
34        fill(lx,lx+n,-INF); fill(ly,ly+n,0);
35        for (int i=0; i<n; i++)
36            for (int j=0; j<n; j++)
37                lx[i] = max(lx[i], edge[i][j]);
38        for (int i=0; i<n; i++){
39            if (DFS(i))
40                continue;
41            for (int j=0; j<n; j++)
42                if (slack[j] > 0)
43                    ly[j] += slack[j];
44            for (int j=0; j<n; j++)
45                if (slack[j] > 0)
46                    lx[i] -= slack[j];
47            if (DFS(i))
48                continue;
49            for (int j=0; j<n; j++)
50                if (slack[j] > 0)
51                    ly[j] += slack[j];
52            for (int j=0; j<n; j++)
53                if (slack[j] > 0)
54                    lx[i] -= slack[j];
55            if (DFS(i))
56                continue;
57            for (int j=0; j<n; j++)
58                if (slack[j] > 0)
59                    ly[j] += slack[j];
60            for (int j=0; j<n; j++)
61                if (slack[j] > 0)
62                    lx[i] -= slack[j];
63            if (DFS(i))
64                continue;
65            for (int j=0; j<n; j++)
66                if (slack[j] > 0)
67                    ly[j] += slack[j];
68            for (int j=0; j<n; j++)
69                if (slack[j] > 0)
70                    lx[i] -= slack[j];
71            if (DFS(i))
72                continue;
73            for (int j=0; j<n; j++)
74                if (slack[j] > 0)
75                    ly[j] += slack[j];
76            for (int j=0; j<n; j++)
77                if (slack[j] > 0)
78                    lx[i] -= slack[j];
79            if (DFS(i))
80                continue;
81            for (int j=0; j<n; j++)
82                if (slack[j] > 0)
83                    ly[j] += slack[j];
84            for (int j=0; j<n; j++)
85                if (slack[j] > 0)
86                    lx[i] -= slack[j];
87            if (DFS(i))
88                continue;
89            for (int j=0; j<n; j++)
90                if (slack[j] > 0)
91                    ly[j] += slack[j];
92            for (int j=0; j<n; j++)
93                if (slack[j] > 0)
94                    lx[i] -= slack[j];
95            if (DFS(i))
96                continue;
97            for (int j=0; j<n; j++)
98                if (slack[j] > 0)
99                    ly[j] += slack[j];
100            for (int j=0; j<n; j++)
101                if (slack[j] > 0)
102                    lx[i] -= slack[j];
103            if (DFS(i))
104                continue;
105            for (int j=0; j<n; j++)
106                if (slack[j] > 0)
107                    ly[j] += slack[j];
108            for (int j=0; j<n; j++)
109                if (slack[j] > 0)
110                    lx[i] -= slack[j];
111            if (DFS(i))
112                continue;
113            for (int j=0; j<n; j++)
114                if (slack[j] > 0)
115                    ly[j] += slack[j];
116            for (int j=0; j<n; j++)
117                if (slack[j] > 0)
118                    lx[i] -= slack[j];
119            if (DFS(i))
120                continue;
121            for (int j=0; j<n; j++)
122                if (slack[j] > 0)
123                    ly[j] += slack[j];
124            for (int j=0; j<n; j++)
125                if (slack[j] > 0)
126                    lx[i] -= slack[j];
127            if (DFS(i))
128                continue;
129            for (int j=0; j<n; j++)
130                if (slack[j] > 0)
131                    ly[j] += slack[j];
132            for (int j=0; j<n; j++)
133                if (slack[j] > 0)
134                    lx[i] -= slack[j];
135            if (DFS(i))
136                continue;
137            for (int j=0; j<n; j++)
138                if (slack[j] > 0)
139                    ly[j] += slack[j];
140            for (int j=0; j<n; j++)
141                if (slack[j] > 0)
142                    lx[i] -= slack[j];
143            if (DFS(i))
144                continue;
145            for (int j=0; j<n; j++)
146                if (slack[j] > 0)
147                    ly[j] += slack[j];
148            for (int j=0; j<n; j++)
149                if (slack[j] > 0)
150                    lx[i] -= slack[j];
151            if (DFS(i))
152                continue;
153            for (int j=0; j<n; j++)
154                if (slack[j] > 0)
155                    ly[j] += slack[j];
156            for (int j=0; j<n; j++)
157                if (slack[j] > 0)
158                    lx[i] -= slack[j];
159            if (DFS(i))
160                continue;
161            for (int j=0; j<n; j++)
162                if (slack[j] > 0)
163                    ly[j] += slack[j];
164            for (int j=0; j<n; j++)
165                if (slack[j] > 0)
166                    lx[i] -= slack[j];
167            if (DFS(i))
168                continue;
169            for (int j=0; j<n; j++)
170                if (slack[j] > 0)
171                    ly[j] += slack[j];
172            for (int j=0; j<n; j++)
173                if (slack[j] > 0)
174                    lx[i] -= slack[j];
175            if (DFS(i))
176                continue;
177            for (int j=0; j<n; j++)
178                if (slack[j] > 0)
179                    ly[j] += slack[j];
180            for (int j=0; j<n; j++)
181                if (slack[j] > 0)
182                    lx[i] -= slack[j];
183            if (DFS(i))
184                continue;
185            for (int j=0; j<n; j++)
186                if (slack[j] > 0)
187                    ly[j] += slack[j];
188            for (int j=0; j<n; j++)
189                if (slack[j] > 0)
190                    lx[i] -= slack[j];
191            if (DFS(i))
192                continue;
193            for (int j=0; j<n; j++)
194                if (slack[j] > 0)
195                    ly[j] += slack[j];
196            for (int j=0; j<n; j++)
197                if (slack[j] > 0)
198                    lx[i] -= slack[j];
199            if (DFS(i))
200                continue;
201            for (int j=0; j<n; j++)
202                if (slack[j] > 0)
203                    ly[j] += slack[j];
204            for (int j=0; j<n; j++)
205                if (slack[j] > 0)
206                    lx[i] -= slack[j];
207            if (DFS(i))
208                continue;
209            for (int j=0; j<n; j++)
210                if (slack[j] > 0)
211                    ly[j] += slack[j];
212            for (int j=0; j<n; j++)
213                if (slack[j] > 0)
214                    lx[i] -= slack[j];
215            if (DFS(i))
216                continue;
217            for (int j=0; j<n; j++)
218                if (slack[j] > 0)
219                    ly[j] += slack[j];
220            for (int j=0; j<n; j++)
221                if (slack[j] > 0)
222                    lx[i] -= slack[j];
223            if (DFS(i))
224                continue;
225            for (int j=0; j<n; j++)
226                if (slack[j] > 0)
227                    ly[j] += slack[j];
228            for (int j=0; j<n; j++)
229                if (slack[j] > 0)
230                    lx[i] -= slack[j];
231            if (DFS(i))
232                continue;
233            for (int j=0; j<n; j++)
234                if (slack[j] > 0)
235                    ly[j] += slack[j];
236            for (int j=0; j<n; j++)
237                if (slack[j] > 0)
238                    lx[i] -= slack[j];
239            if (DFS(i))
240                continue;
241            for (int j=0; j<n; j++)
242                if (slack[j] > 0)
243                    ly[j] += slack[j];
244            for (int j=0; j<n; j++)
245                if (slack[j] > 0)
246                    lx[i] -= slack[j];
247            if (DFS(i))
248                continue;
249            for (int j=0; j<n; j++)
250                if (slack[j] > 0)
251                    ly[j] += slack[j];
252            for (int j=0; j<n; j++)
253                if (slack[j] > 0)
254                    lx[i] -= slack[j];
255            if (DFS(i))
256                continue;
257            for (int j=0; j<n; j++)
258                if (slack[j] > 0)
259                    ly[j] += slack[j];
260            for (int j=0; j<n; j++)
261                if (slack[j] > 0)
262                    lx[i] -= slack[j];
263            if (DFS(i))
264                continue;
265            for (int j=0; j<n; j++)
266                if (slack[j] > 0)
267                    ly[j] += slack[j];
268            for (int j=0; j<n; j++)
269                if (slack[j] > 0)
270                    lx[i] -= slack[j];
271            if (DFS(i))
272                continue;
273            for (int j=0; j<n; j++)
274                if (slack[j] > 0)
275                    ly[j] += slack[j];
276            for (int j=0; j<n; j++)
277                if (slack[j] > 0)
278                    lx[i] -= slack[j];
279            if (DFS(i))
280                continue;
281            for (int j=0; j<n; j++)
282                if (slack[j] > 0)
283                    ly[j] += slack[j];
284            for (int j=0; j<n; j++)
285                if (slack[j] > 0)
286                    lx[i] -= slack[j];
287            if (DFS(i))
288                continue;
289            for (int j=0; j<n; j++)
290                if (slack[j] > 0)
291                    ly[j] += slack[j];
292            for (int j=0; j<n; j++)
293                if (slack[j] > 0)
294                    lx[i] -= slack[j];
295            if (DFS(i))
296                continue;
297            for (int j=0; j<n; j++)
298                if (slack[j] > 0)
299                    ly[j] += slack[j];
300            for (int j=0; j<n; j++)
301                if (slack[j] > 0)
302                    lx[i] -= slack[j];
303            if (DFS(i))
304                continue;
305            for (int j=0; j<n; j++)
306                if (slack[j] > 0)
307                    ly[j] += slack[j];
308            for (int j=0; j<n; j++)
309                if (slack[j] > 0)
310                    lx[i] -= slack[j];
311            if (DFS(i))
312                continue;
313            for (int j=0; j<n; j++)
314                if (slack[j] > 0)
315                    ly[j] += slack[j];
316            for (int j=0; j<n; j++)
317                if (slack[j] > 0)
318                    lx[i] -= slack[j];
319            if (DFS(i))
320                continue;
321            for (int j=0; j<n; j++)
322                if (slack[j] > 0)
323                    ly[j] += slack[j];
324            for (int j=0; j<n; j++)
325                if (slack[j] > 0)
326                    lx[i] -= slack[j];
327            if (DFS(i))
328                continue;
329            for (int j=0; j<n; j++)
330                if (slack[j] > 0)
331                    ly[j] += slack[j];
332            for (int j=0; j<n; j++)
333                if (slack[j] > 0)
334                    lx[i] -= slack[j];
335            if (DFS(i))
336                continue;
337            for (int j=0; j<n; j++)
338                if (slack[j] > 0)
339                    ly[j] += slack[j];
340            for (int j=0; j<n; j++)
341                if (slack[j] > 0)
342                    lx[i] -= slack[j];
343            if (DFS(i))
344                continue;
345            for (int j=0; j<n; j++)
346                if (slack[j] > 0)
347                    ly[j] += slack[j];
348            for (int j=0; j<n; j++)
349                if (slack[j] > 0)
350                    lx[i] -= slack[j];
351            if (DFS(i))
352                continue;
353            for (int j=0; j<n; j++)
354                if (slack[j] > 0)
355                    ly[j] += slack[j];
356            for (int j=0; j<n; j++)
357                if (slack[j] > 0)
358                    lx[i] -= slack[j];
359            if (DFS(i))
360                continue;
361            for (int j=0; j<n; j++)
362                if (slack[j] > 0)
363                    ly[j] += slack[j];
364            for (int j=0; j<n; j++)
365                if (slack[j] > 0)
366                    lx[i] -= slack[j];
367            if (DFS(i))
368                continue;
369            for (int j=0; j<n; j++)
370                if (slack[j] > 0)
371                    ly[j] += slack[j];
372            for (int j=0; j<n; j++)
373                if (slack[j] > 0)
374                    lx[i] -= slack[j];
375            if (DFS(i))
376                continue;
377            for (int j=0; j<n; j++)
378                if (slack[j] > 0)
379                    ly[j] += slack[j];
380            for (int j=0; j<n; j++)
381                if (slack[j] > 0)
382                    lx[i] -= slack[j];
383            if (DFS(i))
384                continue;
385            for (int j=0; j<n; j++)
386                if (slack[j] > 0)
387                    ly[j] += slack[j];
388            for (int j=0; j<n; j++)
389                if (slack[j] > 0)
390                    lx[i] -= slack[j];
391            if (DFS(i))
392                continue;
393            for (int j=0; j<n; j++)
394                if (slack[j] > 0)
395                    ly[j] += slack[j];
396            for (int j=0; j<n; j++)
397                if (slack[j] > 0)
398                    lx[i] -= slack[j];
399            if (DFS(i))
400                continue;
401            for (int j=0; j<n; j++)
402                if (slack[j] > 0)
403                    ly[j] += slack[j];
404            for (int j=0; j<n; j++)
405                if (slack[j] > 0)
406                    lx[i] -= slack[j];
407            if (DFS(i))
408                continue;
409            for (int j=0; j<n; j++)
410                if (slack[j] > 0)
411                    ly[j] += slack[j];
412            for (int j=0; j<n; j++)
413                if (slack[j] > 0)
414                    lx[i] -= slack[j];
415            if (DFS(i))
416                continue;
417            for (int j=0; j<n; j++)
418                if (slack[j] > 0)
419                    ly[j] += slack[j];
420            for (int j=0; j<n; j++)
421                if (slack[j] > 0)
422                    lx[i] -= slack[j];
423            if (DFS(i))
424                continue;
425            for (int j=0; j<n; j++)
426                if (slack[j] > 0)
427                    ly[j] += slack[j];
428            for (int j=0; j<n; j++)
429                if (slack[j] > 0)
430                    lx[i] -= slack[j];
431            if (DFS(i))
432                continue;
433            for (int j=0; j<n; j++)
434                if (slack[j] > 0)
435                    ly[j] += slack[j];
436            for (int j=0; j<n; j++)
437                if (slack[j] > 0)
438                    lx[i] -= slack[j];
439            if (DFS(i))
440                continue;
441            for (int j=0; j<n; j++)
442                if (slack[j] > 0)
443                    ly[j] += slack[j];
444            for (int j=0; j<n; j++)
445                if (slack[j] > 0)
446                    lx[i] -= slack[j];
447            if (DFS(i))
448                continue;
449            for (int j=0; j<n; j++)
450                if (slack[j] > 0)
451                    ly[j] += slack[j];
452            for (int j=0; j<n; j++)
453                if (slack[j] > 0)
454                    lx[i] -= slack[j];
455            if (DFS(i))
456                continue;
457            for (int j=0; j<n; j++)
458                if (slack[j] > 0)
459                    ly[j] += slack[j];
460            for (int j=0; j<n; j++)
461                if (slack[j] > 0)
462                    lx[i] -= slack[j];
463            if (DFS(i))
464                continue;
465            for (int j=0; j<n; j++)
466                if (slack[j] > 0)
467                    ly[j] += slack[j];
468            for (int j=0; j<n; j++)
469                if (slack[j] > 0)
470                    lx[i] -= slack[j];
471            if (DFS(i))
472                continue;
473            for (int j=0; j<n; j++)
474                if (slack[j] > 0)
475                    ly[j] += slack[j];
476            for (int j=0; j<n; j++)
477                if (slack[j] > 0)
478                    lx[i] -= slack[j];
479            if (DFS(i))
480                continue;
481            for (int j=0; j<n; j++)
482                if (slack[j] > 0)
483                    ly[j] += slack[j];
484            for (int j=0; j<n; j++)
485                if (slack[j] > 0)
486                    lx[i] -= slack[j];
487            if (DFS(i))
488                continue;
489            for (int j=0; j<n; j++)
490                if (slack[j] > 0)
491                    ly[j] += slack[j];
492            for (int j=0; j<n; j++)
493                if (slack[j] > 0)
494                    lx[i] -= slack[j];
495            if (DFS(i))
496                continue;
497            for (int j=0; j<n; j++)
498                if (slack[j] > 0)
499                    ly[j] += slack[j];
500            for (int j=0; j<n; j++)
501                if (slack[j] > 0)
502                    lx[i] -= slack[j];
503            if (DFS(i))
504                continue;
505            for (int j=0; j<n; j++)
506                if (slack[j] > 0)
507                    ly[j] += slack[j];
508            for (int j=0; j<n; j++)
509                if (slack[j] > 0)
510                    lx[i] -= slack[j];
511            if (DFS(i))
512                continue;
513            for (int j=0; j<n; j++)
514                if (slack[j] > 0)
515                    ly[j] += slack[j];
516            for (int j=0; j<n; j++)
517                if (slack[j] > 0)
518                    lx[i] -= slack[j];
519            if (DFS(i))
520                continue;
521            for (int j=0; j<n; j++)
522                if (slack[j] > 0)
523                    ly[j] += slack[j];
524            for (int j=0; j<n; j++)
525                if (slack[j] > 0)
526                    lx[i] -= slack[j];
527            if (DFS(i))
528                continue;
529            for (int j=0; j<n; j++)
530                if (slack[j] > 0)
531                    ly[j] += slack[j];
532            for (int j=0; j<n; j++)
533                if (slack[j] > 0)
534                    lx[i] -= slack[j];
535            if (DFS(i))
536                continue;
537            for (int j=0; j<n; j++)
538                if (slack[j] > 0)
539                    ly[j] += slack[j];
540            for (int j=0; j<n; j++)
541                if (slack[j] > 0)
542                    lx[i] -= slack[j];
543            if (DFS(i))
544                continue;
545            for (int j=0; j<n; j++)
546                if (slack[j] > 0)
547                    ly[j] += slack[j];
548            for (int j=0; j<n; j++)
549                if (slack[j] > 0)
550                    lx[i] -= slack[j];
551            if (DFS(i))
552                continue;
553            for (int j=0; j<n; j++)
554                if (slack[j] > 0)
555                    ly[j] += slack[j];
556            for (int j=0; j<n; j++)
557                if (slack[j] > 0)
558                    lx[i] -= slack[j];
559            if (DFS(i))
560                continue;
561            for (int j=0; j<n; j++)
562                if (slack[j] > 0)
563                    ly[j] += slack[j];
564            for (int j=0; j<n; j++)
565                if (slack[j] > 0)
566                    lx[i] -= slack[j];
567            if (DFS(i))
568                continue;
569            for (int j=0; j<n; j++)
570                if (slack[j] > 0)
571                    ly[j] += slack[j];
572            for (int j=0; j<n; j++)
573                if (slack[j] > 0)
574                    lx[i] -= slack[j];
575            if (DFS(i))
576                continue;
577            for (int j=0; j<n; j++)
578                if (slack[j] > 0)
579                    ly[j] += slack[j];
580            for (int j=0; j<n; j++)
581                if (slack[j] > 0)
582                    lx[i] -= slack[j];
583            if (DFS(i))
584                continue;
585            for (int j=0; j<n; j++)
586                if (slack[j] > 0)
587                    ly[j] += slack[j];
588            for (int j=0; j<n; j++)
589                if (slack[j] > 0)
590                    lx[i] -= slack[j];
591            if (DFS(i))
592                continue;
593            for (int j=0; j<n; j++)
594                if (slack[j] > 0)
595                    ly[j] += slack[j];
596            for (int j=0; j<n; j++)
597                if (slack[j] > 0)
598                    lx[i] -= slack[j];
599            if (DFS(i))
600                continue;
601            for (int j=0; j<n; j++)
602                if (slack[j] > 0)
603                    ly[j] += slack[j];
604            for (int j=0; j<n; j++)
605                if (slack[j] > 0)
606                    lx[i] -= slack[j];
607            if (DFS(i))
608                continue;
609            for (int j=0; j<n; j++)
610                if (slack[j] > 0)
611                    ly[j] += slack[j];
612            for (int j=0; j<n; j++)
613                if (slack[j] > 0)
614                    lx[i] -= slack[j];
615            if (DFS(i))
616                continue;
617            for (int j=0; j<n; j++)
618                if (slack[j] > 0)
619                    ly[j] += slack[j];
620            for (int j=0; j<n; j++)
621                if (slack[j] > 0)
622                    lx[i] -= slack[j];
623            if (DFS(i))
624                continue;
625            for (int j=0; j<n; j++)
626                if (slack[j] > 0)
627                    ly[j] += slack[j];
628            for (int j=0; j<n; j++)
629                if (slack[j] > 0)
630                    lx[i] -= slack[j];
631            if (DFS(i))
632                continue;
633            for (int j=0; j<n; j++)
634                if (slack[j] > 0)
635                    ly[j] += slack[j];
636            for (int j=0; j<n; j++)
637                if (slack[j] > 0)
638                    lx[i] -= slack[j];
639            if (DFS(i))
640                continue;
641            for (int j=0; j<n; j++)
642                if (slack[j] > 0)
643                    ly[j] += slack[j];
644            for (int j=0; j<n; j++)
645                if (slack[j] > 0)
646                    lx[i] -= slack[j];
647            if (DFS(i))
648                continue;
649            for (int j=0; j<n; j++)
650                if (slack[j] > 0)
651                    ly[j] += slack[j];
652            for (int j=0; j<n; j++)
653                if (slack[j] > 0)
654                    lx[i] -= slack[j];
655            if (DFS(i))
656                continue;
657            for (int j=0; j<n; j++)
658                if (slack[j] > 0)
659                    ly[j] += slack[j];
660            for (int j=0; j<n; j++)
661                if (slack[j] > 0)
662                    lx[i] -= slack[j];
663            if (DFS(i))
664                continue;
665            for (int j=0; j<n; j++)
666                if (slack[j] > 0)
667                    ly[j] += slack[j];
668            for (int j=0; j<n; j++)
669                if (slack[j] > 0)
670                    lx[i] -= slack[j];
671            if (DFS(i))
672                continue;
673            for (int j=0; j<n; j++)
674                if (slack[j] > 0)
675                    ly[j] += slack[j];
676            for (int j=0; j<n; j++)
677                if (slack[j] > 0)
678                    lx[i] -= slack[j];
679            if (DFS(i))
680                continue;
681            for (int j=0; j<n; j++)
682                if (slack[j] > 0)
683                    ly[j] += slack[j];
684            for (int j=0; j<n; j++)
685                if (slack[j] > 0)
686                    lx[i] -= slack[j];
687            if (DFS(i))
688                continue;
689            for (int j=0; j<n; j++)
690                if (slack[j] > 0)
691                    ly[j] += slack[j];
692            for (int j=0; j<n; j++)
693                if (slack[j] > 0)
694                    lx[i] -= slack[j];
695            if (DFS(i))
696                continue;
697            for (int j=0; j<n; j++)
698                if (slack[j] > 0)
699                    ly[j] += slack[j];
700            for (int j=0; j<n; j++)
701                if (slack[j] > 0)
702                    lx[i]
```

4.4 MaximumClique

```

1 struct MaxClique{
2     static const int MAXN=105;
3     int N,ans;
4     int g[MAXN][MAXN],dp[MAXN],stk[MAXN][MAXN
5         ];
6     int sol[MAXN],tmp[MAXN];//sol[0~ans-1] 為答案
7
8     void init(int n){
9         N=n;//0-base
10        memset(g,0,sizeof(g));
11    }
12    void add_edge(int u,int v){
13        g[u][v]=g[v][u]=1;
14    }
15    int dfs(int ns,int dep){
16        if(!ns){
17            if(dep>ans){
18                ans=dep;
19                memcpy(sol,tmp,sizeof tmp);
20                return 1;
21            }else return 0;
22        }
23        for(int i=0;i<ns;++i){
24            if(dep+ns-i<=ans) return 0;
25            int u=stk[dep][i],cnt=0;
26            if(dep+dp[u]<=ans) return 0;
27            for(int j=i+1;j<ns;++j){
28                int v=stk[dep][j];
29                if(g[u][v]) stk[dep+1][cnt++]=v;
30            }
31            tmp[dep]=u;
32            if(dfs(cnt,dep+1)) return 1;
33        }
34        return 0;
35    }
36    int clique(){
37        int u,v,ns;
38        for(ans=0,u=N-1;u>=0;--u){
39            for(ns=0,tmp[0]=u,v=u+1;v<N;++v)
40                if(g[u][v]) stk[1][ns++]=v;
41        }
42    }
43 }

```

```

39     dfs(ns, 1), dp[u]=ans;
40 }
41     return ans;
42 }
43 };

```

4.5 Minimum Mean Cycle

```

1 #include<cstdio> //for DBL_MAX
2 int dp[MAXN][MAXN]; // 1-base, 0(NM)
3 vector<tuple<int,int,int>> edge;
4 double mmc(int n){ //allow negative weight
5     const int INF=0x3f3f3f3f;
6     for(int t=0;t<n;++t){
7         memset(dp[t+1], 0x3f, sizeof(dp[t+1]));
8         for(const auto &e: edge){
9             int u,v,w;
10            tie(u,v,w) = e;
11            dp[t+1][v]=min(dp[t+1][v], dp[t][u]+w);
12        }
13    }
14    double res = DBL_MAX;
15    for(int u=1;u<=n;++u){
16        if(dp[n][u]==INF) continue;
17        double val = -DBL_MAX;
18        for(int t=0;t<n;++t)
19            val=max(val, (dp[n][u]-dp[t][u])*1.0/(n-
20                t));
21        res=min(res, val);
22    }
23    return res;
24 }

```

4.6 一般圖最小權完美匹配

```

1 struct Graph {
2     // Minimum General Weighted Matching (
3         Perfect Match) 0-base
4     static const int MXN = 105;
5     int n, edge[MXN][MXN];
6     match[MXN], dis[MXN], onstk[MXN];
7     vector<int> stk;
8     void init(int _n) {
9         n = _n;
10        for (int i=0; i<n; i++)
11            for (int j=0; j<n; j++)
12                edge[i][j] = 0;
13    }
14    void add_edge(int u, int v, int w) {
15        edge[u][v] = edge[v][u] = w;
16    }
17    bool SPFA(int u){
18        if (onstk[u]) return true;
19        stk.push_back(u);
20        onstk[u] = 1;
21        for (int v=0; v<n; v++){
22            if (u != v && match[u] != v && !onstk[
23                v]){
24                int m = match[v];

```

4.5 MinimumMeanCycle

```

23         if (dis[m] > dis[u] - edge[v][m] +
24             edge[u][v]){
25             dis[m] = dis[u] - edge[v][m] +
26                 edge[u][v];
27             onstk[v] = 1;
28             stk.push_back(v);
29             if (SPFA(m)) return true;
30             stk.pop_back();
31             onstk[v] = 0;
32         }
33     }
34     onstk[u] = 0;
35     stk.pop_back();
36     return false;
37 }
38
39 int solve() {
40     // find a match
41     for (int i=0; i<n; i+=2){
42         match[i] = i+1, match[i+1] = i;
43     }
44     for(;;){
45         int found = 0;
46         for (int i=0; i<n; i++) dis[i] = onstk[i] = 0;
47         for (int i=0; i<n; i++){
48             stk.clear();
49             if (!onstk[i] && SPFA(i)){
50                 found = 1;
51                 while (stk.size()>=2){
52                     int u = stk.back(); stk.pop_back();
53                     int v = stk.back(); stk.pop_back();
54                     match[u] = v;
55                     match[v] = u;
56                 }
57             }
58         }
59         if (!found) break;
60     }
61     int ret = 0;
62     for (int i=0; i<n; i++)
63         ret += edge[i][match[i]];
64     ret /= 2;
65     return ret;
66 }
67 }graph;

```

4.7 全局最小割

```

1  const int INF=0x3f3f3f3f;
2  template<typename T>
3  struct stoer_wagner{// 0-base
4      static const int MAXN=150;
5      T g[MAXN][MAXN],dis[MAXN];
6      int nd[MAXN],n,s,t;
7      void init(int _n){
8          n=_n;
9          for(int i=0;i<n;++i)
10             for(int j=0;j<n;++j)g[i][j]=0;
11     }
12     void add_edge(int u,int v,T w){

```

```

13         g[u][v]=g[v][u]+=w;
14     }
15     T min_cut(){
16         T ans=INF;
17         for(int i=0;i<n;++i)nd[i]=i;
18         for(int ind,tn=n;tn>1;--tn){
19             for(int i=1;i<tn;++i)dis[nd[i]]=0;
20             for(int i=1;i<tn;++i){
21                 ind=i;
22                 for(int j=i;j<tn;++j){
23                     dis[nd[j]]+=g[nd[i-1]][nd[j]];
24                     if(dis[nd[ind]]<dis[nd[j]])ans=j;
25                 }
26                 swap(nd[ind],nd[i]);
27             }
28             if(ans>dis[nd[ind]])ans=dis[t=nd[ind
29                 ]],s=nd[ind-1];
30             for(int i=0;i<tn;++i)
31                 g[nd[ind-1]][nd[i]]=g[nd[i]][nd[ind
32                     -1]]+=g[nd[i]][nd[ind]];
33         }
34         return ans;
35     }
36 };

```

4.8 平面圖判定

```

1 static const int MAXN = 20;
2 struct Edge{
3     int u, v;
4     Edge(int s, int d) : u(s), v(d) {}
5 };
6 bool isK33(int n, int degree[]){
7     int t = 0, z = 0;
8     for(int i=0;i<n;++i){
9         if(degree[i] == 3)++t;
10        else if(degree[i] == 0)++z;
11        else return false;
12    }
13    return t == 6 && t + z == n;
14 }
15 bool isK5(int n, int degree[]){
16     int f = 0, z = 0;
17     for(int i=0;i<n;++i){
18         if(degree[i] == 4)++f;
19         else if(degree[i] == 0)++z;
20         else return false;
21     }
22     return f == 5 && f + z == n;
23 }
24 // it judge a given graph is Homeomorphic
    with K33 or K5
25 bool isHomeomorphic(bool G[MAXN][MAXN],
    const int n){
26     for(;;){
27         int cnt = 0;
28         for(int i=0;i<n;++i){
29             vector<Edge> E;
30             for(int j=0;j<n&&E.size()<3;++j)
31                 if(G[i][j] && i != j)
32                     E.push_back(Edge(i, j));
33             if(E.size() == 1){
34                 G[i][E[0].v] = G[E[0].v][i] = false;

```

```

35 }else if(E.size() == 2){
36     G[i][E[0].v] = G[E[0].v][i] = false;
37     G[i][E[1].v] = G[E[1].v][i] = false;
38     G[E[0].v][E[1].v] = G[E[1].v][E[0].v]
39         = true;
40     ++cnt;
41 }
42 if(cnt == 0)break;
43 }
44 static int degree[MAXN];
45 fill(degree, degree + n, 0);
46 for(int i=0;i<n;++i){
47     for(int j=i+1; j<n; ++j){
48         if(!G[i][j])continue;
49         ++degree[i];
50         ++degree[j];
51     }
52 }
53 return !(isK33(n, degree) || isK5(n,
54     degree));

```

4.9 最小斯坦納樹 DP

```

1 //n個點，其中r個要構成斯坦納樹
2 //答案在max(dp[(1<<r)-1][k]) k=0~n-1
3 //p表示要構成斯坦納樹的點集
4 //O( n^3 + n*3^n + n^2*2^n )
5 #define REP(i,n) for(int i=0;i<(int)n;++i)
6 const int MAXN=30,MAXM=8; // 0-base
7 const int INF=0x3f3f3f3f;
8 int dp[1<<MAXM][MAXN];
9 int g[MAXN][MAXN]; //圖
10 void init(){memset(g,0x3f,sizeof(g));}
11 void add_edge(int u,int v,int w){
12     g[u][v]=g[v][u]=min(g[v][u],w);
13 }
14 void steiner(int n,int r,int *p){
15     REP(k,n)REP(i,n)REP(j,n)
16         g[i][j]=min(g[i][j],g[i][k]+g[k][j]);
17     REP(i,n)g[i][i]=0;
18     REP(i,r)REP(j,n)dp[1<<i][j]=g[p[i]][j];
19     for(int i=1;i<(1<<r);++i){
20         if(!(i&(i-1)))continue;
21         REP(j,n)dp[i][j]=INF;
22         REP(j,n){
23             int tmp=INF;
24             for(int s=i&(i-1);s;s=i&(s-1))
25                 tmp=min(tmp,dp[s][j]+dp[i^s][j]);
26             REP(k,n)dp[i][k]=min(dp[i][k],g[j][k]+
27                 tmp);
28         }
29     }

```

4.10 最小樹形圖 朱劉

```

1 template<typename T>

```

```

2 struct zhu_liu{
3     static const int MAXN=110,MAXM=10005;
4     struct node{
5         int u,v;
6         T w,tag;
7         node *l,*r;
8         node(int u=0,int v=0,T w=0):u(u),v(v),w(
9             w),tag(0),l(0),r(0){}
10        void down(){
11            w+=tag;
12            if(l)l->tag+=tag;
13            if(r)r->tag+=tag;
14            tag=0;
15        }
16        mem[MAXN]; //靜態記憶體
17        node *pq[MAXN*2],*E[MAXN*2];
18        int st[MAXN*2],id[MAXN*2],m;
19        void init(int n){
20            for(int i=1;i<n;++i){
21                pq[i]=E[i]=0, st[i]=id[i]=i;
22                m=0;
23            }
24            node *merge(node *a,node *b){ //skew heap
25                if(!a||!b)return a?a:b;
26                a->down(),b->down();
27                if(b->w<a->w)return merge(b,a);
28                swap(a->l,a->r);
29                a->l=merge(b,a->l);
30                return a;
31            }
32            void add_edge(int u,int v,T w){
33                if(u!=v)pq[v]=merge(pq[v],&mem[m++] =
34                    node(u,v,w));
35            }
36            int find(int x,int *st){
37                return st[x]=x?x:st[x]=find(st[x],st);
38            }
39            T build(int root,int n){
40                T ans=0;int N=n,all=n;
41                for(int i=1;i<N;++i){
42                    if(i==root||!pq[i])continue;
43                    while(pq[i]){
44                        pq[i]->down(),E[i]=pq[i];
45                        pq[i]=merge(pq[i]->l,pq[i]->r);
46                        if(find(E[i]->u,id)!=find(i,id))
47                            break;
48                    }
49                    if(find(E[i]->u,id)==find(i,id))
50                        continue;
51                    ans+=E[i]->w;
52                    if(find(E[i]->u,st)==find(i,st)){
53                        if(pq[i])pq[i]->tag-=E[i]->w;
54                        pq[++N]=pq[i];id[N]=N;
55                        for(int u=find(E[i]->u,id);u!=i;u=
56                            find(E[u]->u,id)){
57                            if(pq[u])pq[u]->tag-=E[u]->w;
58                            id[find(u,id)]=N;
59                            pq[N]=merge(pq[N],pq[u]);
60                        }
61                        st[N]=find(i,st);
62                        id[find(i,id)]=N;
63                    }else st[find(i,st)]=find(E[i]->u,st),
64                        --all;
65                }
66            }

```

```

60 return all==1?ans:-INT_MAX; //圖不連通就
61 無解
62 }
63 }
64 }

```

4.11 穩定婚姻模板

```

1 queue<int> Q;
2 for ( i : 所有考生 ) {
3     設定在第0志願;
4     Q.push(考生i);
5 }
6 while(Q.size()){
7     當前考生=Q.front();Q.pop();
8     while ( 此考生未分發 ) {
9         指標移到下一志願;
10        if ( 已經沒有志願 or 超出志願總數 )
11            break;
12        計算該考生在該科系加權後的總分;
13        if ( 不符合科系需求 ) continue;
14        if ( 目前科系有餘額 ) {
15            依加權後分數高低順序將考生id加入科系
16            錄取名單中;
17            break;
18        }
19        if ( 目前科系已額滿 ) {
20            if ( 此考生成績比最低分數還高 ) {
21                依加權後分數高低順序將考生id加入科系
22                錄取名單;
23                Q.push(被踢出的考生);
24            }
25        }
26    }
27 }

```

```

17 a[x][j] /= k;
18 if(a[x][j] != 0) pos.push_back(j);
19 }
20 for(int i = 0; i <= m; ++i){
21     if(a[i][y]==0 || i == x) continue;
22     k = a[i][y], a[i][y] = 0;
23     for(int j : pos) a[i][j] -= k*a[x][j];
24 }
25 };
26 for(int x,y;){
27     for(int i=x=1; i <= m; ++i)
28         if(a[i][0]<a[x][0]) x = i;
29     if(a[x][0]>=0) break;
30     for(int j=y=1; j <= n; ++j)
31         if(a[x][j]<a[x][y]) y = j;
32     if(a[x][y]>=0) return VDB(); //infeasible
33     pivot(x, y);
34 }
35 for(int x,y;){
36     for(int j=y=1; j <= n; ++j)
37         if(a[0][j] > a[0][y]) y = j;
38     if(a[0][y]<=0) break;
39     x = -1;
40     for(int i=1; i<=m; ++i) if(a[i][y] > 0)
41         if(x == -1 || a[i][0]/a[i][y]
42             < a[x][0]/a[x][y]) x = i;
43     if(x == -1) return VDB(); //unbounded
44     pivot(x, y);
45 }
46 VDB ans(n + 1);
47 for(int i = 1; i <= m; ++i)
48     if(left[i] <= n) ans[left[i]] = a[i][0];
49 ans[0] = -a[0][0];
50 return ans;
51 }

```

6 Number_Theory

6.1 basic

```

1 template<typename T>
2 void gcd(const T &a,const T &b,T &d,T &x,T &
3     y){
4     if(!b) d=a,x=1,y=0;
5     else gcd(b,a%b,d,y,x), y=-x*(a/b);
6 }
7 long long int phi[N+1];
8 void phiTable(){
9     for(int i=1;i<=N;i++)phi[i]=i;
10    for(int i=1;i<=N;i++)for(x=i*2;x<=N;x+=i)
11        phi[x]-=phi[i];
12 }
13 void all_divdown(const LL &n) { // all n/x
14     for(LL a=1;a<=n;a=n/(n/(a+1))) {
15         // dosomething;
16     }
17 }
18 const int MAXPRIME = 1000000;
19 int iscom[MAXPRIME], prime[MAXPRIME],
20     primecnt;
21 int phi[MAXPRIME], mu[MAXPRIME];

```

```

19 void sieve(void){
20     memset(iscom,0,sizeof(iscom));
21     primecnt = 0;
22     phi[1] = mu[1] = 1;
23     for(int i=2;i<MAXPRIME;++i) {
24         if(!iscom[i]) {
25             prime[primecnt++] = i;
26             mu[i] = -1;
27             phi[i] = i-1;
28         }
29         for(int j=0;j<primecnt;++j) {
30             int k = i * prime[j];
31             if(k>=MAXPRIME) break;
32             iscom[k] = prime[j];
33             if(i%prime[j]==0) {
34                 mu[k] = 0;
35                 phi[k] = phi[i] * prime[j];
36                 break;
37             } else {
38                 mu[k] = -mu[i];
39                 phi[k] = phi[i] * (prime[j]-1);
40             }
41         }
42     }
43 }
44
45 bool g_test(const LL &g, const LL &p, const
vector<LL> &v) {
46     for(int i=0;i<v.size();++i)
47         if(modexp(g,(p-1)/v[i],p)==1)
48             return false;
49     return true;
50 }
51
52 LL primitive_root(const LL &p) {
53     if(p==2) return 1;
54     vector<LL> v;
55     Factor(p-1,v);
56     v.erase(unique(v.begin(), v.end()), v.end
());
57     for(LL g=2;g<p;++g)
58         if(g_test(g,p,v))
59             return g;
60     puts("primitive_root NOT FOUND");
61     return -1;
62 }
63
64 int Legendre(const LL &a, const LL &p) {
65     return modexp(a%p,(p-1)/2,p); }
66
67 LL inv(const LL &a, const LL &n) {
68     LL d,x,y;
69     gcd(a,n,d,x,y);
70     return d==1 ? (x+n)%n : -1;
71 }
72
73 int inv[maxN];
74 LL invtable(int n,LL P){
75     inv[1]=1;
76     for(int i=2;i<n;++i)
77         inv[i]=(-P/i)*inv[P%i]%P;
78 }
79
80 LL log_mod(const LL &a, const LL &b, const
LL &p) {
81     // a ^ x = b ( mod p )
82     int m=sqrt(p+.5), e=1;
83     LL v=inv(modexp(a,m,p), p);

```

```

81     map<LL,int> x;
82     x[1]=0;
83     for(int i=1;i<m;++i) {
84         e = LLmul(e,a,p);
85         if(!x.count(e)) x[e] = i;
86     }
87     for(int i=0;i<m;++i) {
88         if(x.count(b)) return i*m + x[b];
89         b = LLmul(b,v,p);
90     }
91     return -1;
92 }
93
94 LL Tonelli_Shanks(const LL &n, const LL &p)
{
95     // x^2 = n ( mod p )
96     if(n==0) return 0;
97     if(Legendre(n,p)!=1) while(1) { puts("SQRT
ROOT does not exist"); }
98     int S = 0;
99     LL Q = p-1;
100     while( !(Q&1) ) { Q>>=1; ++S; }
101     if(S==1) return modexp(n%p,(p+1)/4,p);
102     LL z = 2;
103     for(; Legendre(z,p)!=-1;++z)
104         LL c = modexp(z,Q,p);
105     LL R = modexp(n%p,(Q+1)/2,p), t = modexp(n
%p,Q,p);
106     int M = S;
107     while(1) {
108         if(t==1) return R;
109         LL b = modexp(c,1L<<(M-i-1),p);
110         R = LLmul(R,b,p);
111         t = LLmul(LLmul(b,b,p), t, p);
112         c = LLmul(b,b,p);
113         M = i;
114     }
115     return -1;
116 }
117
118 template<typename T>
119 T Euler(T n){
120     T ans=n;
121     for(T i=2;i*i<=n;++i){
122         if(n%i==0){
123             ans=ans/i*(i-1);
124             while(n%i==0)n/=i;
125         }
126     }
127     if(n>1)ans=ans/n*(n-1);
128     return ans;
129 }
130
131 //Chinese_remainder_theorem
132 template<typename T>
133 T pow_mod(T n,T k,T m){
134     T ans=1;
135     for(n=(n>m?n%m:n);k;k>>=1){
136         if(k&1)ans=ans*n%m;
137         n=n*n%m;
138     }
139     return ans;
140 }
141
142 template<typename T>
143 T crt(vector<T> &m,vector<T> &a){
144     T M=1,tM,ans=0;

```

```

144     for(int i=0;i<(int)m.size();++i)M*=m[i];
145     for(int i=0;i<(int)a.size();++i){
146         tM=M/m[i];
147         ans=(ans+(a[i]*tM%M)*pow_mod(tM,Euler(m
[i])-1,m[i])%M)%M;
148         /*如果m[i]是質數 · Euler(m[i])-1=m[i]-2 ·
就不要再算Euler了*/
149     }
150     return ans;
151 }
152
153 //java code
154 //求sqrt(N)的連分數
155 public static void Pell(int n){
156     BigInteger N,p1,p2,q1,q2,a0,a1,a2,g1,g2,h1
,h2,p,q;
157     g1=q2=p1=BigInteger.ZERO;
158     h1=q1=p2=BigInteger.ONE;
159     a0=a1=BigInteger.valueOf((int)Math.sqrt
(1.0*n));
160     BigInteger ans=a0.multiply(a0);
161     if(ans.equals(BigInteger.valueOf(n))){
162         System.out.println("No solution!");
163         return ;
164     }
165     while(true){
166         g2=a1.multiply(h1).subtract(g1);
167         h2=N.subtract(g2.pow(2)).divide(h1);
168         a2=g2.add(a0).divide(h2);
169         p=a1.multiply(p2).add(p1);
170         q=a1.multiply(q2).add(q1);
171         if(p.pow(2).subtract(N.multiply(q.pow
(2))).compareTo(BigInteger.ONE)==0)
172             break;
173         g1=g2;h1=h2;a1=a2;
174         p1=p2;p2=p;
175         q1=q2;q2=q;
176     }
177     System.out.println(p+" "+q);

```

6.2 bit_set

```

1 void sub_set(int S){
2     int sub=S;
3     do{
4         //對某集合的子集合的處理
5         sub=(sub-1)&S;
6     }while(sub!=S);
7 }
8
9 void k_sub_set(int k,int n){
10     int comb=(1<<k)-1,S=1<<n;
11     while(comb<S){
12         //對大小為k的子集合的處理
13         int x=comb&~comb,y=comb+x;
14         comb=((comb&~y)/x>>1)|y;
15     }

```

6.3 EXT_GCD

```

1 #include <bits/stdc++.h>
2 using namespace std;
3 typedef long long LL;
4 typedef pair < LL, LL> ii;
5
6 ii exd_gcd( LL a, LL b) {
7     if (a % b == 0) return ii(0, 1);
8     ii T = exd_gcd(b, a % b);
9     return ii( T.second, T.first - a / b * T
.second);
10 }
11
12 LL mod_inv(LL x) { // P is mod number, gcd(x
,P) must be 1
13     return (exd_gcd(x,P).first%P+P)%P;

```

6.4 FFT

```

1 const double PI = acos(-1);
2 using cd = complex<double>;
3 // Do FFT. invert=true to do iFFT.
4 // n MUST be power of 2.
5 void fft(cd a[], int n, bool invert) {
6     for (int i = 1, j = 0; i < n; i++) {
7         int bit = n >> 1;
8         for (; j & bit; bit >>= 1)
9             j ^= bit;
10         j ^= bit;
11
12         if (i < j)
13             swap(a[i], a[j]);
14     }
15
16     for (int len = 2; len <= n; len <= 1) {
17         double ang = 2 * PI / len * (invert
? -1 : 1);
18         cd wlen(cos(ang), sin(ang));
19         for (int i = 0; i < n; i += len) {
20             cd w(1);
21             for (int j = 0; j < len / 2; j
++) {
22                 cd u = a[i+j], v = a[i+j+len
/2] * w;
23                 a[i+j] = u + v;
24                 a[i+j+len/2] = u - v;
25                 w *= wlen;
26             }
27         }
28     }
29
30     if (invert) {
31         for (int i = 0; i < n; i++)
32             a[i] /= n;
33     }
34 }

```

6.5 find_real_root


```

1 // an*x^n + ... + a1x + a0 = 0;
2 int sign(double x){
3     return x < -eps ? -1 : x > eps;
4 }
5
6 double get(const vector<double>&coef, double
7     x){
8     double e = 1, s = 0;
9     for(auto i : coef) s += i*e, e *= x;
10    return s;
11 }
12 double find(const vector<double>&coef, int n
13     , double lo, double hi){
14     double sign_lo, sign_hi;
15     if( !(sign_lo = sign(get(coef,lo))) )
16         return lo;
17     if( !(sign_hi = sign(get(coef,hi))) )
18         return hi;
19     if(sign_lo * sign_hi > 0) return INF;
20     for(int stp = 0; stp < 100 && hi - lo >
21         eps; ++stp){
22         double m = (lo+hi)/2.0;
23         int sign_mid = sign(get(coef,m));
24         if(!sign_mid) return m;
25         if(sign_lo*sign_mid < 0) hi = m;
26         else lo = m;
27     }
28     return (lo+hi)/2.0;
29 }
30
31 vector<double> cal(vector<double>coef, int n
32     ){
33     vector<double>res;
34     if(n == 1){
35         if(sign(coef[1])) res.pb(-coef[0]/coef
36             [1]);
37         return res;
38     }
39     vector<double>dcoef(n);
40     for(int i = 0; i < n; ++i) dcoef[i] = coef
41         [i+1]*(i+1);
42     vector<double>droot = cal(dcoef, n-1);
43     droot.insert(droot.begin(), -INF);
44     droot.pb(INF);
45     for(int i = 0; i+1 < droot.size(); ++i){
46         double tmp = find(coef, n, droot[i],
47             droot[i+1]);
48         if(tmp < INF) res.pb(tmp);
49     }
50     return res;
51 }
52
53 int main () {
54     vector<double>ve;
55     vector<double>ans = cal(ve, n);
56     // 視情況把答案 +eps · 避免 -0
57 }

```

6.6 FWT

```

1 vector<int> F_OR_T(vector<int> f, bool
2     inverse){

```

```

3     for(int i=0; (2<<i)<=f.size(); ++i)
4         for(int j=0; j<f.size(); j+=2<<i)
5             for(int k=0; k<(1<<i); ++k)
6                 f[j+k+(1<<i)] += f[j+k]*(inverse
7                     ?-1:1);
8     return f;
9 }
10 vector<int> rev(vector<int> A) {
11     for(int i=0; i<A.size(); i+=2)
12         swap(A[i],A[i^(A.size()-1)]);
13     return A;
14 }
15 vector<int> F_AND_T(vector<int> f, bool
16     inverse){
17     return rev(F_OR_T(rev(f), inverse));
18 }
19 vector<int> F_XOR_T(vector<int> f, bool
20     inverse){
21     for(int i=0; (2<<i)<=f.size(); ++i)
22         for(int j=0; j<f.size(); j+=2<<i)
23             for(int k=0; k<(1<<i); ++k){
24                 int u=f[j+k], v=f[j+k+(1<<i)];
25                 f[j+k+(1<<i)] = u-v, f[j+k] = u+v;
26             }
27     if(inverse) for(auto &a:f) a/=f.size();
28     return f;
29 }

```

6.7 gauss_elimination

```

1 typedef double Matrix[maxn][maxn];
2 void gausss_elimination(Matrix A, int n){
3     int r;
4     for(int i=0; i<n; i++){
5         r = i;
6         for(int j=i+1; j<n; j++)
7             if(fabs(A[j][i]) > fabs(A[r][i]))
8                 r = j;
9         if(r!=i) for(int j=0; j<n; j++)
10             swap(A[r][j], A[i][j]);
11         for(int k=i+1; k<n; k++){
12             double f = A[k][i]/A[i][i];
13             for(int j=i; j<n; j++) A[k][j]
14                 -= f*A[i][j];
15         }
16     }
17     for(int i=n-1; i>=0; i--){
18         for(int j=i+1; j<n; j++)
19             A[i][n] -= A[j][n] * A[i][j];
20         A[i][n] /= A[i][i];
21     }

```

6.8 LL_mul

```

1 long long mul(long long a, long long b) {
2     long long ans = 0, step = a % MOD;
3     while (b) {

```

```

4         if (b & 1L) ans += step;
5         if (ans >= MOD) ans %= MOD;
6         step <<= 1L;
7         if (step >= MOD) step %= MOD;
8         b >>= 1L;
9     }
10    return ans % MOD;
11 }

```

6.9 Lucas

```

1 int mod_fact(int n,int &e){
2     e=0;
3     if(n==0)return 1;
4     int res=mod_fact(n/P,e);
5     e += n/P;
6     if((n/P)%2==0)return res*fact[n%P]%P;
7     return res*(P-fact[n%P])%P;
8 }
9 int Cmod(int n,int m){
10    int a1,a2,a3,e1,e2,e3;
11    a1=mod_fact(n,e1);
12    a2=mod_fact(m,e2);
13    a3=mod_fact(n-m,e3);
14    if(e1>e2+e3)return 0;
15    return a1*inv(a2*a3%P)%P;
16 }

```

6.10 Matrix

```

1 template<typename T>
2 struct Matrix{
3     using rt = std::vector<T>;
4     using mt = std::vector<rt>;
5     using matrix = Matrix<T>;
6     int r,c;
7     mt m;
8     Matrix(int r,int c):r(r),c(c),m(r,rt(c)){
9         rt& operator[](int i){return m[i];}
10    matrix operator+(const matrix &a){
11        matrix rev(r,c);
12        for(int i=0;i<r;++i)
13            for(int j=0;j<c;++j)
14                rev[i][j]=m[i][j]+a.m[i][j];
15        return rev;
16    }
17    matrix operator-(const matrix &a){
18        matrix rev(r,c);
19        for(int i=0;i<r;++i)
20            for(int j=0;j<c;++j)
21                rev[i][j]=m[i][j]-a.m[i][j];
22        return rev;
23    }
24    matrix operator*(const matrix &a){
25        matrix rev(r,a.c);
26        matrix tmp(a.c,a.r);
27        for(int i=0;i<a.r;++i)
28            for(int j=0;j<a.c;++j)
29                tmp[j][i]=a.m[i][j];
30        for(int i=0;i<r;++i)

```

```

31        for(int j=0;j<a.c;++j)
32            for(int k=0;k<c;++k)
33                rev.m[i][j]+=m[i][k]*tmp[j][k];
34        return rev;
35    }
36    bool inverse(){
37        Matrix t(r,r+c);
38        for(int y=0;y<r;y++){
39            t.m[y][c+y] = 1;
40            for(int x=0;x<c;++x)
41                t.m[y][x]=m[y][x];
42        }
43        if( !t.gas() )
44            return false;
45        for(int y=0;y<r;y++){
46            for(int x=0;x<c;++x)
47                m[y][x]=t.m[y][c+x]/t.m[y][y];
48        }
49        return true;
50    }
51    T gas(){
52        vector<T> lazy(r,1);
53        bool sign=false;
54        for(int i=0;i<r;++i){
55            if( m[i][i]==0 ){
56                int j=i+1;
57                while(j<r&&!m[j][i])j++;
58                if(j==r)continue;
59                if(j==r)continue;
60                m[i].swap(m[j]);
61                sign=!sign;
62            }
63            for(int j=0;j<r;++j){
64                if(i==j)continue;
65                lazy[j]=lazy[j]*m[i][i];
66                T mx=m[j][i];
67                for(int k=0;k<c;++k)
68                    m[j][k]=m[j][k]*m[i][i]-m[i][k]*mx;
69            }
70        }
71        det=sign?-1:1;
72        for(int i=0;i<r;++i){
73            det = det*m[i][i];
74            for(auto &j:m[i])j/=lazy[i];
75        }
76        return det;
77    };

```

6.11 Miller_Rabin

```

1 LL mod_mul(LL a, LL b, LL mod) {
2     // return (__int128)a*b%mod;
3     /* In case __int128 doesn't work(32* multi
4         to avoid ovf) */
5     LL x=0,y=a%mod;
6     while(b > 0){
7         if (b&1) x = (x+y)%mod;
8         y = (y*2)%mod;
9         b >>= 1;
10    }
11    return x%mod;

```

```

12 LL qpow(LL a, LL p, LL mod) {
13     if (p<=0) return 1;
14     LL temp = qpow(a,p/2,mod);
15     temp = mod_mul(temp,temp,mod);
16     if (p&1) return mod_mul(temp,a,mod);
17     return temp;
18 }
19 bool MRtest(LL a, LL d, LL n) {
20     LL x = qpow(a,d,n);
21     if (x==1 || x==n-1) return true;
22     while (d != n-1) {
23         x = mod_mul(x,x,n);
24         d *= 2;
25         if (x==n-1) return true;
26         if (x==1) return false;
27     }
28     return false;
29 }
30 bool is_prime(LL n) {
31     if (n==2) return true;
32     if (n<2 || n%2==0) return false;
33     LL table[7] = {2, 325, 9375, 28178,
34         450775, 9780504, 1795265022}, d=n-1;
35     while (d%2 != 0) d>>=1; // n-1 = d * 2^r,
36         d is odd.
37     for (int i=0; i<7; i++) {
38         LL a = table[i] % n;
39         if (a==0 || a==1 || a==n-1) continue;
40         if (!MRtest(a,d,n)) {
41             return false;
42         }
43     }
44     return true;
45 }

```

6.12 NTT

```

1 const LL mod = 998244353;
2 const LL p_root = 3;
3 const LL root_pw = 1LL << 23;
4
5 // Do NTT under mod. invert=true to do iNTT.
6 // mod MUST be a prime, if mod=c*2^k+1, then
7 // p_root is any primitive root of mod
8 // root_pw=2^k, and n(size) MUST <= 2^k
9 // n MUST be power of 2.
10 // mod=2013265921, root_pw=1LL<<27, p_root
    =31
11
12 void ntt(LL a[], int n, bool invert) {
13     LL root = qpow(p_root, (mod-1)/root_pw,
14         mod);
15     LL root_1 = mod_inv(root, mod);
16
17     for (int i = 1, j = 0; i < n; i++) {
18         LL bit = n >> 1;
19         for (; j & bit; bit >>= 1)
20             j ^= bit;
21
22         if (i < j)
23             swap(a[i], a[j]);
24     }

```

```

25     for (int len = 2; len <= n; len <= 1) {
26         LL wlen = invert ? root_1 : root;
27         for (int i = len; i < root_pw; i <= 1)
28             wlen = wlen * wlen % mod;
29
30         for (int i = 0; i < n; i += len) {
31             LL w = 1;
32             for (int j = 0; j < len / 2; j++) {
33                 LL u = a[i+j], v = a[i+j+len/2] * w
34                     % mod;
35                 a[i+j] = u + v < mod ? u + v : u + v
36                     - mod;
37                 a[i+j+len/2] = u - v >= 0 ? u - v :
38                     u - v + mod;
39                 w = w * wlen % mod;
40             }
41         }
42     }
43     if (invert) {
44         LL n_1 = mod_inv(n, mod);
45         for (int i = 0; i < n; i++) {
46             a[i] = a[i] * n_1 % mod;
47         }
48     }

```

7 String

7.1 ACA

```

1 static const int MAXL=200005,SIGMA=26; //
    MAXL: sum of length in dictionary
2 // link: suffix link, next: DFA link, n: #
    of nodes, tag: ID of str ends here
3 // next and link always exist, others exist
    iff values != -1.
4 // nocc: next occurrence, first node with
    tag != -1 along suffix link
5 int n, dep[MAXL], link[MAXL], next[MAXL][
    SIGMA];
6 int trie[MAXL][SIGMA], tag[MAXL], nocc[MAXL
    ];
7
8 int new_node(int p) {
9     // Add you init if recording more values.
10    dep[n] = n == 0 ? 0 : dep[p] + 1;
11    link[n] = tag[n] = nocc[n] = -1;
12    for (int i = 0; i < SIGMA; i++) {
13        next[n][i] = 0;
14        trie[n][i] = -1;
15    }
16    return n++;
17 }
18 void build(vector<string> &dict) {
19     // Some init should be written in new_node
20     .
21     n = 0;
22     new_node(0);
23     for (int i = 0; i < dict.size(); i++) {

```

```

23     int v = 0;
24     for (char ch : dict[i]) {
25         int to = ch - 'a'; // CHANGE THIS !!
26         if (trie[v][to] == -1) {
27             trie[v][to] = next[v][to] = new_node
                (v);
28         }
29         v = trie[v][to];
30     }
31     tag[v] = i;
32 }
33
34 queue<int> Q;
35 link[0] = 0;
36 Q.push(0);
37 while (!Q.empty()) {
38     int v = Q.front(); Q.pop();
39     for (int to = 0; to < SIGMA; to++) {
40         if (trie[v][to] != -1) {
41             int u = trie[v][to];
42             link[u] = v == 0 ? 0 : next[link[v]
                ][to];
43             nocc[u] = tag[link[u]] != -1 ? link[
                u] : nocc[link[u]];
44             for (int j = 0; j < SIGMA; j++) {
45                 if (trie[u][j] == -1) {
46                     next[u][j] = next[link[u]]
                        [j];
47                 }
48             }
49             Q.push(u);
50         }
51     }
52 }
53 }

```

7.2 hash

```

1 #define MAXN 1000000
2 #define mod 1073676287
3 /*mod 必須要是質數*/
4 typedef long long T;
5 char s[MAXN+5];
6 T h[MAXN+5]; /*hash陣列*/
7 T h_base[MAXN+5]; /*h_base[n]=(prime^n)%mod*/
8 void hash_init(int len,T prime){
9     h_base[0]=1;
10    for(int i=1;i<=len;++i){
11        h[i]=(h[i-1]*prime+s[i-1])%mod;
12        h_base[i]=(h_base[i-1]*prime)%mod;
13    }
14 }
15 T get_hash(int l,int r){/*閉區間寫法・設編號
    為0 ~ len-1*/
16     return (h[r+1]-(h[l]*h_base[r-l+1])%mod+
17         mod)%mod;

```

7.3 KMP

```

1 vector<int> lps; // Longest prefix suffix,
    0-based
2 int match(const string &text, const string &
    pat) {
3     /* Init is included */
4     lps.resize(pat.size());
5     /* DP */
6     lps[0]=0;
7     for (int i=1; i<pat.size(); i++) {
8         int len=lps[i - 1];
9         while(len>0 && pat[len]!=pat[i]) len=lps
                [len - 1];
10        lps[i] = pat[len]==pat[i] ? len+1 : 0;
11    }
12    /* Match */
13    int i = 0, j = 0;
14    while (i < text.size() && j < pat.size())
15        {
16            if (text[i] == pat[j]) i++, j++;
17            else if (j == 0) i++;
18            else j = lps[j - 1];
19        }
20    if (j >= pat.size()) return i - j;
21    return -1;

```

7.4 manacher

```

1 vector<int> d1(n); // Max Len of palindrome
    centered at s[i]
2 for (int i = 0, l = 0, r = -1; i < n; i++) {
3     int k = (i > r) ? 1 : min(d1[l + r - i],
4         r - i + 1);
5     while (0 <= i - k && i + k < n && s[i -
6         k] == s[i + k]) {
7         k++;
8     }
9     d1[i] = k--;
10    if (i + k > r) {
11        l = i - k;
12        r = i + k;
13    }
14 }
15 vector<int> d2(n); // Max Len of centered
    at "gap" before s[i]
16 for (int i = 0, l = 0, r = -1; i < n; i++) {
17     int k = (i > r) ? 0 : min(d2[l + r - i +
18         1], r - i + 1);
19     while (0 <= i - k - 1 && i + k < n && s[
20         i - k - 1] == s[i + k]) {
21         k++;
22     }
23     d2[i] = k--;
24     if (i + k > r) {
25         l = i - k - 1;
26         r = i + k;
27     }
28 }

```

7.5 minimal_string_rotation

```
1 int min_string_rotation(const string &s){
2     int n=s.size(),i=0,j=1,k=0;
3     while(i<n&&j<n&&k<n){
4         int t=s[(i+k)%n]-s[(j+k)%n];
5         ++k;
6         if(t){
7             if(t>0)i+=k;
8             else j+=k;
9             if(i==j)++j;
10            k=0;
11        }
12    }
13    return min(i,j); //最小循環表示法起始位置
14 }
```

7.6 reverseBWT

```
1 const int MAXN = 305, MAXC = 'Z';
2 int ranks[MAXN], tots[MAXC], first[MAXC];
3 void rankBWT(const string &bw){
4     memset(ranks,0,sizeof(int)*bw.size());
5     memset(tots,0,sizeof(tots));
6     for(size_t i=0;i<bw.size();++i)
7         ranks[i] = tots[int(bw[i])]+1;
8 }
9 void firstCol(){
10    memset(first,0,sizeof(first));
11    int totc = 0;
12    for(int c='A';c<='Z';++c){
13        if(!tots[c]) continue;
14        first[c] = totc;
15        totc += tots[c];
16    }
17 }
18 string reverseBwt(string bw,int begin){
19     rankBWT(bw), firstCol();
20     int i = begin; //原字串最後一個元素的位置
21     string res;
22     do{
23         char c = bw[i];
24         res = c + res;
25         i = first[int(c)] + ranks[i];
26     }while( i != begin );
27     return res;
28 }
```

7.7 SA

```
1 /* rank: inverse sa */
2 /* MAXL: Maximum length of string, lcp[i]:
3    LCP(sa[i], sa[i-1]) */
4 string text;
5 int sa[MAXL], isa[MAXL], lcp[MAXL], cnt[MAXL]
6     +ALPHA];
7 void build(const vector<int> &_text) {
8     text = _text + '\0'; // Must add this,
9     must >= 0
```

```
7     int sz = text.size(), lim = ALPHA; //
8     Takes ALPHA time, note when #TC is
9     large
10    for (int i = 0; i < lim; i++) cnt[i] = 0;
11    for (int i = 0; i < sz; i++) cnt[ isa[i] =
12        text[i] ]++;
13    for (int i = 1; i < lim; i++) cnt[i] +=
14        cnt[i - 1];
15    for (int i = sz - 1; i >= 0; i--) sa[ --
16        cnt[text[i]] ] = i;
17
18    lim = max(sz, ALPHA);
19    int *rk = isa, *nsa = lcp, *nrk = lcp;
20    for (int len = 1; len < sz; len <= 1) {
21        int num = 0;
22        for (int i = sz - len; i < sz; i++) nsa[
23            num++] = i;
24        for (int i = 0; i < sz; i++) if (sa[i]
25            >= len) nsa[num++] = sa[i] - len;
26
27        for (int i = 0; i < lim; i++) cnt[i] =
28            0;
29        for (int i = 0; i < sz; i++) cnt[ rk[i]
30            ]++;
31        for (int i = 1; i < lim; i++) cnt[i] +=
32            cnt[i - 1];
33        for (int i = sz-1; i >= 0; i--) sa[ --
34            cnt[rk[nsa[i]]] ] = nsa[i];
35
36        num = 0;
37        nrk[sa[0]] = num++;
38        for (int i = 1; i < sz; i++) {
39            bool cond = rk[sa[i]] == rk[sa[i-1]]
40                && sa[i] + len < sz;
41            cond = cond && sa[i-1] + len < sz &&
42                rk[sa[i]+len] == rk[sa[i-1]+len];
43            if (cond) nrk[sa[i]] = num - 1;
44            else nrk[sa[i]] = num++;
45        }
46        if (num >= sz) break;
47        lim = num;
48        swap(rk, nrk);
49        nsa = nrk;
50    }
51    /* LCP */
52    int len = 0;
53    lcp[0] = 0; // Undefined
54    for (int i=0; i<sz; i++) {
55        if (isa[i] == 0) continue;
56        len = max(0, len-1);
57        int j = sa[isa[i]-1];
58        while (text[i+len] == text[j+len]) len
59            ++;
60        lcp[isa[i]] = len;
61    }
```

7.8 Z

```
1 void z_alg(char *s,int len,int *z){
```

```
2     int l=0,r=0;
3     z[0]=len;
4     for(int i=1;i<len;++i){
5         z[i]=i>r?0:(i-l+z[i-l]<z[l]?z[i-l]:r-i
6             +1);
7         while(i+z[i]<len&&s[i+z[i]]==s[z[i]]++)z
8             [i];
9         if(i+z[i]-1>r)r=i+z[i]-1,l=i;
10    }
```

8 Tarjan

8.1 dominator_tree

```
1 struct dominator_tree{
2     static const int MAXN=5005;
3     int n; // 1-base
4     vector<int> suc[MAXN],pre[MAXN];
5     int fa[MAXN],dfn[MAXN],id[MAXN],Time;
6     int semi[MAXN],idom[MAXN];
7     int anc[MAXN],best[MAXN]; //disjoint set
8     vector<int> dom[MAXN]; //dominator_tree
9     void init(int _n){
10        n=_n;
11        for(int i=1;i<=n;++i)suc[i].clear(),pre[
12            i].clear();
13    }
14    void add_edge(int u,int v){
15        suc[u].push_back(v);
16        pre[v].push_back(u);
17    }
18    void dfs(int u){
19        dfn[u]=++Time,id[Time]=u;
20        for(auto v:suc[u]){
21            if(dfn[v])continue;
22            dfs(v),fa[dfn[v]]=dfn[u];
23        }
24    }
25    int find(int x){
26        if(x==anc[x])return x;
27        int y=find(anc[x]);
28        if(semi[best[x]]>semi[best[anc[x]]])best
29            [x]=best[anc[x]];
30        return anc[x]=y;
31    }
32    void tarjan(int r){
33        Time=0;
34        for(int t=1;t<=n;++t){
35            dfn[t]=idom[t]=0; //u=r或是u無法到達r時
36            idom[id[u]]=0
37            dom[t].clear();
38            anc[t]=best[t]=semi[t]=t;
39        }
40        dfs(r);
41        for(int y=Time;y>2;--y){
42            int x=fa[y],idy=id[y];
43            for(auto z:pre[idy]){
44                if(!z=dfn[z])continue;
45                find(z);
46                semi[y]=min(semi[y],semi[best[z]]);
47            }
```

```
44        }
45        dom[semi[y]].push_back(y);
46        anc[y]=x;
47        for(auto z:dom[x]){
48            find(z);
49            idom[z]=semi[best[z]]<x?best[z]:x;
50        }
51        dom[x].clear();
52    }
53    for(int u=2;u<=Time;++u){
54        if(idom[u]!=semi[u])idom[u]=idom[idom[
55            u]];
56        dom[id[idom[u]]].push_back(id[u]);
57    }
58 } dom;
```

8.2 橋連通分量

```
1 #define N 1005
2 struct edge{
3     int u,v;
4     bool is_bridge;
5     edge(int u=0,int v=0):u(u),v(v),is_bridge
6         (0){}
7 };
8 vector<edge> E;
9 vector<int> G[N]; // 1-base
10 int low[N],vis[N],Time;
11 int bcc_id[N],bridge_cnt,bcc_cnt; // 1-base
12 int st[N],top; //BCC用
13 void add_edge(int u,int v){
14     G[u].push_back(E.size());
15     E.emplace_back(u,v);
16     G[v].push_back(E.size());
17     E.emplace_back(v,u);
18 }
19 void dfs(int u,int re=-1){ //u當前點，re為u連
20     接前一個點的邊
21     int v;
22     low[u]=vis[u]=++Time;
23     st[top++]=u;
24     for(int e:G[u]){
25         v=E[e].v;
26         if(!vis[v]){
27             dfs(v,E[e]); //e^1反向邊
28             low[u]=min(low[u],low[v]);
29             if(vis[u]<low[v]){
30                 E[e].is_bridge=E[e^1].is_bridge=1;
31                 ++bridge_cnt;
32             }
33         }else if(vis[v]<vis[u]&&e!=re){
34             low[u]=min(low[u],vis[v]);
35         }
36     }
37     if(vis[u]==low[u]){ //處理BCC
38         ++bcc_cnt; // 1-base
39         do bcc_id[v=st[--top]]=bcc_cnt; //每個點
40             所在的BCC
41         while(v!=u);
42     }
43 }
```

```

41 Time=bcc_cnt=bridge_cnt=top=0;
42 E.clear();
43 for(int i=1;i<n;++i){
44     G[i].clear();
45     vis[i]=bcc_id[i]=0;
46 }
47 }

```

8.3 雙連通分量 & 割點

```

1 #define N 1005
2 vector<int> G[N]; // 1-base
3 vector<int> bcc[N]; // 存每塊雙連通分量的點
4 int low[N], vis[N], Time;
5 int bcc_id[N], bcc_cnt; // 1-base
6 bool is_cut[N]; // 是否為割點
7 int st[N], top;
8 void dfs(int u, int pa=-1) { // u當前點, pa父親
9     int t, child=0;
10    low[u]=vis[u]=++Time;
11    st[top++]=u;
12    for(int v:G[u]){
13        if(!vis[v]){
14            dfs(v,u), ++child;
15            low[u]=min(low[u], low[v]);
16            if(vis[u]<=low[v]){
17                is_cut[u]=1;
18                bcc[++bcc_cnt].clear();
19                do{
20                    bcc_id[t=st[--top]]=bcc_cnt;
21                    bcc[bcc_cnt].push_back(t);
22                }while(t==v);
23                bcc_id[u]=bcc_cnt;
24                bcc[bcc_cnt].push_back(u);
25            }
26        } else if(vis[v]<vis[u]&&v!=pa) // 反向邊
27            low[u] = min(low[u], vis[v]);
28    } // u是dfs樹的根要特判
29    if(pa==-1&&child<2) is_cut[u]=0;
30 }
31 void bcc_init(int n){
32     Time=bcc_cnt=top=0;
33     for(int i=1;i<n;++i){
34         G[i].clear();
35         is_cut[i]=vis[i]=bcc_id[i]=0;
36     }
37 }

```

9 Tree

9.1 HLD

```

1 // In this template value is on the edge,
   // everything is 1-based
2 int N;
3 vector<Edge> G[MAXN+5];
4

```

```

5 // Preprocess info, setup in dfs1
6 int heavy[MAXN+5], pa_w[MAXN+5], sz[MAXN+5];
7 int pa[MAXN+5], dep[MAXN+5], recorder[MAXN+5]; // Which node record edge i.
8
9 // HLD info, setup in build, 1-based
10 // pos: position of node i in seg tree.
11 // head: For NODE i, points to head of the chain.
12 int chain_no, border, pos[MAXN+5], head[MAXN+5];
13
14 void dfs1(int v, int p) {
15     pa[v] = p;
16     sz[v] = 1;
17     dep[v] = dep[p] + 1;
18     heavy[v] = -1;
19
20     for (const Edge &e : G[v]) {
21         if (e.to == p) continue;
22         dfs1(e.to, v);
23         pa_w[e.to] = e.w;
24         recorder[e.id] = e.to;
25         sz[v] += sz[e.to];
26         if (heavy[v] == -1 || sz[e.to] > sz[heavy[v]]) {
27             heavy[v] = e.to;
28         }
29     }
30 }
31
32 void build(int v, int chain_head) {
33     pos[v] = ++border;
34     head[v] = chain_head;
35     tree.update(pos[v], pa_w[v], 1, N, 1);
36
37     if (heavy[v] != -1) build(heavy[v], chain_head);
38
39     for (const Edge &e : G[v]) {
40         if (e.to == pa[v] || e.to == heavy[v]) continue;
41         build(e.to, e.to);
42     }
43 }
44
45 void init_HLD() {
46     /* Only init used data, be careful. */
47     /* Does not init G!!!! */
48     border = dep[1] = pa_w[1] = 0;
49     dfs1(1, 1);
50     build(1, 1);
51 }
52
53 int query_up(int a, int b) {
54     int ans = 0;
55     while (head[a] != head[b]) {
56         if (dep[head[a]] < dep[head[b]]) swap(a, b);
57         ans = max(ans, tree.query(pos[head[a]], pos[a], 1, N, 1));
58         a = pa[head[a]];
59     }
60
61     if (a == b) return ans;
62     if (dep[a] < dep[b]) swap(a, b);
63

```

9.2 POJ_tree

```

1 #include<bits/stdc++.h>
2 using namespace std;
3 #define MAXN 10005
4 int n,k;
5 vector<pair<int,int>> g[MAXN];
6 int size[MAXN];
7 bool vis[MAXN];
8 inline void init(){
9     for(int i=0;i<n;++i){
10         g[i].clear();
11         vis[i]=0;
12     }
13 }
14 void get_dis(vector<int> &dis, int u, int pa, int d){
15     dis.push_back(d);
16     for(size_t i=0;i<g[u].size();++i){
17         int v=g[u][i].first, w=g[u][i].second;
18         if(v!=pa&&!vis[v]) get_dis(dis, v, u, d+w);
19     }
20 }
21 vector<int> dis; // 這東西如果放在函數裡會TLE
22 int cal(int u, int d){
23     dis.clear();
24     get_dis(dis, u, -1, d);
25     sort(dis.begin(), dis.end());
26     int l=0, r=dis.size()-1, res=0;
27     while(l<r){
28         while(l<r&&dis[l]+dis[r]>k)--r;
29         res+=r-(l++);
30     }
31     return res;
32 }
33 pair<int,int> tree_centroid(int u, int pa, const int sz){
34     size[u]=1; // 找樹重心, second是重心
35     pair<int,int> res(INT_MAX, -1);
36     int ma=0;
37     for(size_t i=0;i<g[u].size();++i){
38         int v=g[u][i].first;
39         if(v==pa||vis[v]) continue;
40         res=min(res, tree_centroid(v, u, sz));
41         size[u]+=size[v];
42         ma=max(ma, size[v]);
43     }
44     ma=max(ma, sz-size[u]);
45     return min(res, make_pair(ma, u));
46 }
47 int tree_DC(int u, int sz){
48     int center=tree_centroid(u, -1, sz).second;
49     int ans=cal(center, 0);
50     vis[center]=1;
51     for(size_t i=0;i<g[center].size();++i){
52

```

```

53     int v=g[center][i].first, w=g[center][i].second;
54     if(vis[v]) continue;
55     ans-=cal(v, w);
56     ans+=tree_DC(v, size[v]);
57 }
58 return ans;
59 }
60 int main(){
61     while(scanf("%d%d", &n, &k), n||k){
62         init();
63         for(int i=1;i<n;++i){
64             int u, v, w;
65             scanf("%d%d%d", &u, &v, &w);
66             g[u].push_back(make_pair(v, w));
67             g[v].push_back(make_pair(u, w));
68         }
69         printf("%d\n", tree_DC(1, n));
70     }
71     return 0;
72 }

```

10 others

10.1 vimrc

```

1 se ai nu ru cul mouse=a
2 se cin et ts=2 sw=2 sts=2
3 colo desert
4 se gfn=Monospace\ 14

```

11 zformula

11.1 formula

11.1.1 formula.txt

- 若多項式 $f(x)$ 有有理根 P/Q (P, Q 互質), 則 P 必為常數項 a_0 之因數, Q 必為領導係數 a_n 之因數
- 滿足 $\text{ceil}(n/i)=k$ 之最大 i :
 - INF, if $k=1$
 - $n/(k-1)-1$, else if $k-1$ 整除 n
 - $n/(k-1)$, else
- 滿足 $\text{floor}(n/i)=k$ 之最大 i : $\text{floor}(n/k)$
- 尤拉函數: $\phi(n)=n$ 乘上所有 $(1-1/p)$, 對 n 之所有質因數 p
- 費馬小定理: $a \cdot a^{p-2} = 1 \pmod{p}$, a, p 互質
- 次方同餘定理: $a^k \pmod{p} = (a \pmod{p})^{k \pmod{p-1}}$, p 是質數
- Modulo inverse: $\text{inv}[i] = -\text{floor}(p/i) * \text{inv}[p/i]$
- 中國剩餘定理: $x = A_i \pmod{m_i}$, m_i 互質, $M_i =$ 所有 m 的乘積 / m_i , $T_i = M_i^{-1} \pmod{m_i}$, 則 $x = \text{sigma}(M_i * T_i * A_i) \pmod{M}$
- 枚舉擴展歐幾里得之解: 若 x_0, y_0 為 $a*x + b*y = k$ 之一組解, 則 $x = x_0 + t*b/\text{gcd}(a, b)$, $y = y_0 + t*a/\text{gcd}(a, b)$ 亦為解, t 為整數

10. $\text{Sigma}\{i : \gcd(i,n) = 1 \text{ and } i \text{ in } [1, n]\} = n * \phi(n)/2$ for $n > 1$
11. $\text{Sigma}\{i * r^i : i \text{ in } [1, n]\} = (n * r^{(n+1)} - r * (r^n - 1)) / ((r - 1) * (r - 1))$
12. 最大獨立集: 點的集合, 其內點不相鄰
13. 最小點覆蓋: 點的集合, 所有邊都被覆蓋
14. 最大匹配: 邊的集合, 其內邊不共用點
15. 最小邊覆蓋: 邊的集合, 所有點都被覆蓋
16. 最大獨立集 + 最小點覆蓋 = V(數值)
17. 最大匹配 + 最小邊覆蓋 = V(數值)
18. 最大匹配 = 最大流 (二分圖)
19. 最大匹配 = 最小點覆蓋 (二分圖)
20. 最小點覆蓋 + 最小邊覆蓋 = V(數值, 二分圖)
21. 一矩陣 A 所有 eigen value 之合 = 對角線之合
22. 一矩陣 A 所有 eigen value 之積 = det(A)
23. 三角形 ABC, 對邊長 abc:
24. $\text{area} = \sqrt{s(s-a)(s-b)(s-c)}$, $s = \text{周長}/2$
25. $a/\sin A = b/\sin B = c/\sin C = 2R$, R 為外接圓半徑
26. 內接圓半徑 = $2 * \text{area} / (a+b+c)$
27. 外接圓半徑 = $abc / 4 * \text{area}$
28. 枚舉 submask: for (int s=m; s; s=(s-1)&m) // Take care of ZERO by yourself
29. 某些質數: 54018521, 370248451, 6643838879, 119218851371, 5600748293801 39916801, 479001599, 87178291199, 8589935681, 433494437, 2971215073

11.1.2 Pick 公式

給定頂點坐標均是整點的簡單多邊形, 面積 = 內部格點數 + 邊上格點數/2-1

11.1.3 圖論

1. 對於平面圖, $F = E - V + C + 1$, C 是連通分量數
2. 對於平面圖, $E \leq 3V - 6$
3. 對於連通圖 G, 最大獨立點集的大小設為 I(G), 最大匹配大小設為 M(G), 最小點覆蓋設為 Cv(G), 最小邊覆蓋設為 Ce(G). 對於任意連通圖:
- (a) $I(G) + Cv(G) = |V|$
- (b) $M(G) + Ce(G) = |V|$
4. 對於連通二分圖:
- (a) $I(G) = Cv(G)$
- (b) $M(G) = Ce(G)$
5. 最大權閉合圖:
- (a) $C(u, v) = \infty, (u, v) \in E$
- (b) $C(S, v) = W_v, W_v > 0$
- (c) $C(v, T) = -W_v, W_v < 0$
- (d) $\text{ans} = \sum_{W_v > 0} W_v - \text{flow}(S, T)$
6. 最大密度子圖:
- (a) 求 $\max \left(\frac{W_e + W_v}{|V'|} \right), e \in E', v \in V'$
- (b) $U = \sum_{v \in V} 2W_v + \sum_{e \in E} W_e$
- (c) $C(u, v) = W_{(u,v)}, (u, v) \in E$, 雙向邊
- (d) $C(S, v) = U, v \in V$
- (e) $D_u = \sum_{(u,v) \in E} W_{(u,v)}$
- (f) $C(v, T) = U + 2g - D_v - 2W_v, v \in V$

- (g) 二分搜 g :
- $l = 0, r = U, \text{eps} = 1/n^2$
- if $((U \times |V| - \text{flow}(S, T)) / 2 > 0)$ $l = \text{mid}$
- else $r = \text{mid}$
- (h) $\text{ans} = \min_cut(S, T)$
- (i) $|E| = 0$ 要特殊判斷
7. 弦圖:
- (a) 點數大於 3 的環都要有一條弦
- (b) 完美消除序列從後往前依次給每個點染色, 給每個點染上可以染的最小顏色
- (c) 最大團大小 = 色數
- (d) 最大獨立集: 完美消除序列從前往後能選就選
- (e) 最小團覆蓋: 最大獨立集的點和他延伸的邊構成
- (f) 區間圖是弦圖
- (g) 區間圖的完美消除序列: 將區間按造又端點由小到大数据序
- (h) 區間圖染色: 用線段樹做

11.1.4 dinic 特殊圖複雜度

1. 單位流: $O \left(\min \left(V^{3/2}, E^{1/2} \right) E \right)$
2. 二分圖: $O \left(V^{1/2} E \right)$

11.1.5 0-1 分數規劃

$x_i \in \{0, 1\} \cdot x_i$ 可能會有其他限制, 求 $\max \left(\frac{\sum B_i x_i}{\sum C_i x_i} \right)$

1. $D(i, g) = B_i - g \times C_i$
2. $f(g) = \sum D(i, g) x_i$
3. $f(g) = 0$ 時 g 為最佳解, $f(g) < 0$ 沒有意義
4. 因為 $f(g)$ 單調可以二分搜 g
5. 或用 Dinkelbach 通常比較快
- ```
1 binary_search(){
2 while(r-l>eps){
3 g=(l+r)/2;
4 for(i: 所有元素)D[i]=B[i]-g*C[i]; //D(i,g)
5 找出一組合法x[i]使f(g)最大;
6 if(f(g)>0) l=g;
7 else r=g;
8 }
9 Ans = r;
10 }
11 Dinkelbach(){
12 g=任意狀態 (通常設為0);
13 do{
14 Ans=g;
15 for(i: 所有元素)D[i]=B[i]-g*C[i]; //D(i,g)
16 找出一組合法x[i]使f(g)最大;
17 p=0,q=0;
18 for(i: 所有元素)
19 if(x[i])p+=B[i],q+=C[i];
20 g=p/q; //更新解, 注意g=0的情況
21 }while(abs(Ans-g)>EPS);
22 return Ans;
23 }
```

### 11.1.6 學長公式

1.  $\sum_{d|n} \phi(n) = n$
2.  $g(n) = \sum_{d|n} f(d) \Rightarrow f(n) = \sum_{d|n} \mu(d) \times g(n/d)$
3. Harmonic series  $H_n = \ln(n) + \gamma + 1/(2n) - 1/(12n^2) + 1/(120n^4)$
4.  $\gamma = 0.57721566490153286060651209008240243104215$
5. 格雷碼 =  $n \oplus (n >> 1)$
6.  $SG(A + B) = SG(A) \oplus SG(B)$
7. 選轉矩陣  $M(\theta) = \begin{pmatrix} \cos \theta & -\sin \theta \\ \sin \theta & \cos \theta \end{pmatrix}$

### 11.1.7 基本數論

1.  $\sum_{d|n} \mu(n) = [n == 1]$
2.  $g(m) = \sum_{d|m} f(d) \Leftrightarrow f(m) = \sum_{d|m} \mu(d) \times g(m/d)$
3.  $\sum_{i=1}^m \sum_{j=1}^m \text{互質數量} = \sum \mu(d) \lfloor \frac{n}{d} \rfloor \lfloor \frac{m}{d} \rfloor$
4.  $\sum_{i=1}^n \sum_{j=1}^m \text{lcm}(i, j) = n \sum_{d|n} d \times \phi(d)$

### 11.1.8 排組公式

1. k 卡特蘭  $\frac{C_n^n}{n(k-1)+1} \cdot C_m^n = \frac{n!}{m!(n-m)!}$
2.  $H(n, m) \cong x_1 + x_2 + \dots + x_n = k, num = C_{n+k-1}^k$
3. Stirling number of  $2^{nd}$ , n 人分 k 組方法數目
- (a)  $S(0, 0) = S(n, n) = 1$
- (b)  $S(n, 0) = 0$
- (c)  $S(n, k) = kS(n-1, k) + S(n-1, k-1)$
4. Bell number, n 人分任意多組方法數目
- (a)  $B_0 = 1$
- (b)  $B_n = \sum_{i=0}^n S(n, i)$
- (c)  $B_{n+1} = \sum_{k=0}^n C_k^n B_k$
- (d)  $B_{p+n} \equiv B_n + B_{n+1} \text{ mod } p$ , p is prime
- (e)  $B_p^{m+n} \equiv mB_n + B_{n+1} \text{ mod } p$ , p is prime
- (f) From  $B_0 : 1, 1, 2, 5, 15, 52, 203, 877, 4140, 21147, 115975$
5. Derangement, 錯排, 沒有人在自己位置上
- (a)  $D_n = n! \left( 1 - \frac{1}{1!} + \frac{1}{2!} - \frac{1}{3!} \dots + (-1)^n \frac{1}{n!} \right)$
- (b)  $D_n = (n-1)(D_{n-1} + D_{n-2}), D_0 = 1, D_1 = 0$
- (c) From  $D_0 : 1, 0, 1, 2, 9, 44, 265, 1854, 14833, 133496$
6. Binomial Equality

- (a)  $\sum_k \binom{r}{m+k} \binom{s}{n-k} = \binom{r+s}{m+n}$
- (b)  $\sum_k \binom{l}{m+k} \binom{s}{n+k} = \binom{l+s}{l-m+n}$
- (c)  $\sum_k \binom{m}{m+k} \binom{s+k}{n} (-1)^k = (-1)^{l+m} \binom{s-m}{n-l}$
- (d)  $\sum_{k \leq l} \binom{l-k}{m} \binom{s}{k-n} (-1)^k = (-1)^{l+m} \binom{s-m-1}{l-n-m}$
- (e)  $\sum_{0 \leq k \leq l} \binom{l-k}{m} \binom{q+k}{n} = \binom{l+q+1}{m+n+1}$
- (f)  $\binom{r}{k} = (-1)^k \binom{k-r-1}{k}$

- (g)  $\binom{r}{m} \binom{m}{k} = \binom{r}{k} \binom{r-k}{m-k}$
- (h)  $\sum_{k \leq n} \binom{r+k}{k} = \binom{r+n+1}{n}$
- (i)  $\sum_{0 \leq k \leq n} \binom{k}{m} = \binom{n+1}{m+1}$
- (j)  $\sum_{k \leq m} \binom{m+r}{m+k} x^k y^k = \sum_{k \leq m} \binom{r}{k} (-x)^k (x+y)^{m-k}$

### 11.1.9 幕次, 幕次和

1.  $a^b \% P = a^{b \% \varphi(P) + \varphi(P)}, b \geq \varphi(P)$
2.  $1^3 + 2^3 + 3^3 + \dots + n^3 = \frac{n^4}{4} + \frac{n^3}{2} + \frac{n^2}{4}$
3.  $1^4 + 2^4 + 3^4 + \dots + n^4 = \frac{n^5}{5} + \frac{n^4}{2} + \frac{n^3}{3} - \frac{n}{30}$
4.  $1^5 + 2^5 + 3^5 + \dots + n^5 = \frac{n^6}{6} + \frac{n^5}{2} + \frac{5n^4}{12} - \frac{n^2}{12}$
5.  $0^k + 1^k + 2^k + \dots + n^k = P(k), P(k) = \frac{(n+1)^{k+1} - \sum_{i=0}^{k-1} C_i^{k+1} P(i)}{k+1}, P(0) = n+1$
6.  $\sum_{k=0}^{m-1} k^n = \frac{1}{n+1} \sum_{k=0}^n C_n^{k+1} B_k m^{n+1-k}$
7.  $\sum_{j=0}^m C_j^{m+1} B_j = 0, B_0 = 1$
8. 除了  $B_1 = -1/2$ , 剩下的奇數項都是 0
9.  $B_2 = 1/6, B_4 = -1/30, B_6 = 1/42, B_8 = -1/30, B_{10} = 5/66, B_{12} = -691/2730, B_{14} = 7/6, B_{16} = -3617/510, B_{18} = 43867/798, B_{20} = -174611/330,$

### 11.1.10 Burnside's lemma

1.  $|X/G| = \frac{1}{|G|} \sum_{g \in G} |X^g|$
2.  $X^g = t^{c(g)}$
3. G 表示有幾種轉法,  $X^g$  表示在那種轉法下, 有幾種是會保持對稱的, t 是顏色數, c(g) 是循環節不動的面數。
4. 正立方體塗三顏色, 轉 0 有  $3^6$  個元素不變, 轉 90 有 6 種, 每種有  $3^3$  不變, 180 有  $3 \times 3^4$ , 120(角) 有  $8 \times 3^2$ , 180(邊) 有  $6 \times 3^3$ , 全部  $\frac{1}{24} (3^6 + 6 \times 3^3 + 3 \times 3^4 + 8 \times 3^2 + 6 \times 3^3) = \frac{24}{57}$

### 11.1.11 Count on a tree

1. Rooted tree:  $s_{n+1} = \frac{1}{n} \sum_{i=1}^n (i \times a_i \times \sum_{j=1}^{\lfloor n/i \rfloor} a_{n+1-i \times j})$
2. Unrooted tree:
- (a) Odd:  $a_n - \sum_{i=1}^{n/2} a_i a_{n-i}$
- (b) Even:  $Odd + \frac{1}{2} a_{n/2} (a_{n/2} + 1)$
3. Spanning Tree
- (a) 完全圖  $n^n - 2$
- (b) 一般圖 (Kirchhoff's theorem)  $M[i][i] = \text{degree}(V_i), M[i][j] = -1, \text{if have } E(i, j), 0 \text{ if no edge. delete any one row and col in } A, \text{ans} = \det(A)$

## 11.2 java

### 11.2.1 文件操作

```

1 import java.io.*;
2 import java.util.*;
3 import java.math.*;
4 import java.text.*;
5
6 public class Main{
7
8 public static void main(String args[]){
9 throws FileNotFoundException,
10 IOException
11 Scanner sc = new Scanner(new FileReader(
12 "a.in"));
13 PrintWriter pw = new PrintWriter(new
14 FileWriter("a.out"));
15 int n,m;
16 n=sc.nextInt();//读入下一个INT
17 m=sc.nextInt();
18
19 for(ci=1; ci<=c; ++ci){
20 pw.println("Case #"+ci+": easy for
21 output");
22 }
23
24 pw.close();//关闭流并释放 · 这个很重要 ·
25 否则是没有输出的
26 sc.close();//关闭流并释放
27 }
28 }

```

### 11.2.4 sort

```

1 static class cmp implements Comparator{
2 public int compare(Object o1,Object o2){
3 BigInteger b1=(BigInteger)o1;
4 BigInteger b2=(BigInteger)o2;
5 return b1.compareTo(b2);
6 }
7 }
8 public static void main(String[] args)
9 throws IOException{
10 Scanner cin = new Scanner(System.in);
11 int n;
12 n=cin.nextInt();
13 BigInteger[] seg = new BigInteger[n];
14 for (int i=0;i<n;i++)
15 seg[i]=cin.nextBigInteger();
16 Arrays.sort(seg,new cmp());
17 }

```

### 11.2.2 优先队列

```

1 PriorityQueue queue = new PriorityQueue(1,
2 new Comparator(){
3 public int compare(Point a, Point b){
4 if(a.x < b.x || a.x == b.x && a.y < b.y)
5 return -1;
6 else if(a.x == b.x && a.y == b.y)
7 return 0;
8 else return 1;
9 }
10 });

```

### 11.2.3 Map

```

1 Map map = new HashMap();
2 map.put("sa","dd");
3 String str = map.get("sa").toString;
4
5 for(Object obj : map.keySet()){
6 Object value = map.get(obj);
7 }

```

# ACM ICPC TEAM REFERENCE - POLARSHEEP

## Contents

|                                 |          |                                |          |                                       |           |                                    |           |
|---------------------------------|----------|--------------------------------|----------|---------------------------------------|-----------|------------------------------------|-----------|
| <b>1 Data_Structure</b>         | <b>1</b> | <b>3 Geometry</b>              | <b>3</b> | 6.3 EXT_GCD . . . . .                 | 8         | <b>9 Tree</b>                      | <b>12</b> |
| 1.1 hull_dynamic . . . . .      | 1        | 3.1 circle . . . . .           | 3        | 6.4 FFT . . . . .                     | 8         | 9.1 HLD . . . . .                  | 12        |
| 1.2 Treap . . . . .             | 1        | 3.2 convex_hull . . . . .      | 3        | 6.5 find_real_root . . . . .          | 8         | 9.2 POJ_tree . . . . .             | 12        |
| 1.3 undo_disjoint_set . . . . . | 2        | 3.3 geometry . . . . .         | 3        | 6.6 FWT . . . . .                     | 9         | <b>10 others</b>                   | <b>12</b> |
| 1.4 整體二分 . . . . .              | 2        | 3.4 smallest_circle . . . . .  | 5        | 6.7 gauss_elimination . . . . .       | 9         | 10.1 vimrc . . . . .               | 12        |
| <b>2 Flow</b>                   | <b>2</b> | 3.5 最近點對 . . . . .             | 5        | 6.8 LL_mul . . . . .                  | 9         | <b>11 zformula</b>                 | <b>12</b> |
| 2.1 DFSflow . . . . .           | 2        | <b>4 Graph</b>                 | <b>5</b> | 6.9 Lucas . . . . .                   | 9         | 11.1 formula . . . . .             | 12        |
| 2.2 Dinic . . . . .             | 2        | 4.1 3989_ 穩定婚姻 . . . . .       | 5        | 6.10 Matrix . . . . .                 | 9         | 11.1.1 formula.txt . . . . .       | 12        |
| 2.3 min_cost_flow . . . . .     | 2        | 4.2 blossom . . . . .          | 5        | 6.11 Miller_Rabin . . . . .           | 9         | 11.1.2 Pick 公式 . . . . .           | 13        |
|                                 |          | 4.3 KM . . . . .               | 5        | 6.12 NTT . . . . .                    | 10        | 11.1.3 圖論 . . . . .                | 13        |
|                                 |          | 4.4 MaximumClique . . . . .    | 6        | <b>7 String</b>                       | <b>10</b> | 11.1.4 dinic 特殊圖複雜度 . . . . .      | 13        |
|                                 |          | 4.5 MinimumMeanCycle . . . . . | 6        | 7.1 ACA . . . . .                     | 10        | 11.1.5 0-1 分數規劃 . . . . .          | 13        |
|                                 |          | 4.6 一般圖最小權完美匹配 . . . . .       | 6        | 7.2 hash . . . . .                    | 10        | 11.1.6 學長公式 . . . . .              | 13        |
|                                 |          | 4.7 全局最小割 . . . . .            | 6        | 7.3 KMP . . . . .                     | 10        | 11.1.7 基本數論 . . . . .              | 13        |
|                                 |          | 4.8 平面圖判定 . . . . .            | 6        | 7.4 manacher . . . . .                | 10        | 11.1.8 排組公式 . . . . .              | 13        |
|                                 |          | 4.9 最小斯坦納樹 DP . . . . .        | 7        | 7.5 minimal_string_rotation . . . . . | 11        | 11.1.9 冪次, 冪次和 . . . . .           | 13        |
|                                 |          | 4.10 最小樹形圖_朱劉 . . . . .        | 7        | 7.6 reverseBWT . . . . .              | 11        | 11.1.10 Burnside's lemma . . . . . | 13        |
|                                 |          | 4.11 穩定婚姻模板 . . . . .          | 7        | 7.7 SA . . . . .                      | 11        | 11.1.11 Count on a tree . . . . .  | 13        |
|                                 |          | <b>5 Linear_Programming</b>    | <b>7</b> | 7.8 Z . . . . .                       | 11        | 11.2 java . . . . .                | 14        |
|                                 |          | 5.1 simplex . . . . .          | 7        | <b>8 Tarjan</b>                       | <b>11</b> | 11.2.1 文件操作 . . . . .              | 14        |
|                                 |          | <b>6 Number_Theory</b>         | <b>7</b> | 8.1 dominator_tree . . . . .          | 11        | 11.2.2 优先队列 . . . . .              | 14        |
|                                 |          | 6.1 basic . . . . .            | 7        | 8.2 橋連通分量 . . . . .                   | 11        | 11.2.3 Map . . . . .               | 14        |
|                                 |          | 6.2 bit_set . . . . .          | 8        | 8.3 雙連通分量 & 割點 . . . . .              | 12        | 11.2.4 sort . . . . .              | 14        |