1 Data_Structure

1.1 hull_dynamic

```cpp
const ll is_query = -(1LL<<62);
struct Line {
ll m, b;
mutable function<const Line*()> succ;
bool operator<(const Line& rhs) const {
    if (rhs.b != is_query) return m < rhs.m;
    const Line* s = succ();
    if (!s) return 0;
    ll x = rhs.m;
    return b - s->b < (s->m - m) * x;
}
};
// Upper envelope, erase cannot be done.
// Even if you do erase, the popped lines are gone, it won't be a correct hull.
struct HullDynamic : public multiset<Line> {
bool bad(iterator y) {
    auto z = next(y);
    if (y == begin()) {
        if (z == end()) return 0;
        return y->m == z->m && y->b <= z->b;
    }
    auto x = prev(y);
    if (z == end()) return y->m == x->m && y->b <= x->b;
    return 1.0 * (x->b - y->b)*(z->m - y->m) >= 1.0 * (y->b - z->b)*(y->m - x->m);
}
void insert_line(ll m, ll b) {
    auto y = insert({ m, b });
    y->succ = [=] { return next(y) == end() ? 0 : &*next(y); };
    if (bad(y)) { erase(y); return; }
    while (next(y) != end() && bad(next(y))) erase(next(y));
    while (y != begin() && bad(prev(y))) erase(prev(y));
}
ll eval(ll x) {
    auto l = *lower_bound((Line) { x, is_query });
    return l.m * x + l.b;
}
};
```

1.2 persistent_treap

```cpp
// Once persistent, every op must not change tree struct.
// Also don't free them if any version may be referenced later.
Treap* merge(Treap *a, Treap *b) { // When new, also copy priority
  if (!a || !b) return a ? (new Treap(a)) : (new Treap(b));
  Treap *t;
  if (a->pri > b->pri) {
    t = new Treap(a);
    t->r = merge(t->r, b);
    pull(t);
    return t;
  }
  else {
    t = new Treap(b);
    t->l = merge(a, t->l);
    pull(t);
    return t;
  }
}
void split(Treap *t, int k, Treap *&a, Treap *&b ) {
  // First k numbers <-> in *a
  if (!t) { a = b = NULL; return; }
  t = new Treap(t);
  if (Size(t->l) < k) {
    a = t;
    split(t->r, k - Size(t->l) - 1, a->r, b);
    pull(a);
  }
  else {
    b = t;
    split(t->l, k, a, b->l);
    pull(b);
  }
}
```

1.3 Treap

```cpp
struct Treap{
  int sz , val , pri , tag;
  Treap *l , *r;
  Treap( int _val ){
    val = _val; sz = 1;
    pri = rand(); l = r = NULL; tag = 0;
  }
};
void push( Treap * a ){
  if( a->tag ){
    Treap *swp = a->l; a->l = a->r; a->r = swp;
    int swp2;
    if( a->l ) a->l->tag ^= 1;
    if( a->r ) a->r->tag ^= 1;
    a->tag = 0;
  }
}
int Size( Treap * a ){ return a ? a->sz : 0; }
void pull( Treap * a ){
  a->sz = Size( a->l ) + Size( a->r ) + 1;
}
Treap* merge( Treap *a , Treap *b ){
  if( !a || !b ) return a ? a : b;
  if( a->pri > b->pri ){
    push( a );
    a->r = merge( a->r , b );
    pull( a );
    return a;
  }else{
    push( b );
    b->l = merge( a , b->l );
    pull( b );
    return b;
  }
}
void split( Treap *t , int k , Treap*&a , Treap*&b ){
  // First k elements <-> in *a
  if( !t ){ a = b = NULL; return; }
  push( t );
  if( Size( t->l ) + 1 <= k ){
    a = t;
    split( t->r , k - Size( t->l ) - 1 , a->r , b );
    pull( a );
  }else{
    b = t;
    split( t->l , k , a , b->l );
    pull( b );
  }
}
void split2(Treap *t, int k, Treap *&a, Treap *&b ) {
  // key<k <-> in *a, when used as a BST
  if (!t) { a = b = NULL; return; }
  push(t);
  if (Key(t) < k) {
    a = t;
    split2(t->r, k, a->r, b);
    pull(a);
  }
  else {
    b = t;
    split2(t->l, k, a, b->l);
    pull(b);
  }
}
```

1.4 undo_disjoint_set

```cpp
struct DisjointSet {
  // save() is like recursive
  // undo() is like return
  int n, fa[MXN], sz[MXN];
  vector<pair<int*,int>> h;
  vector<int> sp;
  void init(int tn) {
    n=tn;
    for (int i=0; i<n; i++) sz[fa[i]=i]=1;
    sp.clear(); h.clear();
  }
  void assign(int *k, int v) {
    h.PB({k, *k});
    *k=v;
  }
  void save() { sp.PB(SZ(h)); }
  void undo() {
    assert(!sp.empty());
    int last=sp.back(); sp.pop_back();
    while (SZ(h)!=last) {
      auto x=h.back(); h.pop_back();
      *x.F=x.S;
    }
  }
  int f(int x) {
    while (fa[x]!=x) x=fa[x];
    return x;
  }
  void uni(int x, int y) {
    x=f(x); y=f(y);
    if (x==y) return ;
    if (sz[x]<sz[y]) swap(x, y);
    assign(&sz[x], sz[x]+sz[y]);
    assign(&fa[y], x);
  }
}djs;
```

1.5 整體二分

```cpp
void totBS(int L, int R, vector<Item> M){
  if(Q.empty()) return; //維護全域B陣列
  if(L==R) 整個M的答案=r, return;
  int mid = (L+R)/2;
  vector<Item> mL, mR;
  do_modify_B_with_divide(mid,M);
  //讓B陣列在遞迴的時候只會保留[L~mid]的資訊
  undo_modify_B(mid,M);
  totBS(L,mid,mL);
  totBS(mid+1,R,mR);
}
```

2 Flow

2.1 DFSflow

```cpp
struct Edge{
    int to,cap,rev;
    Edge(int a,int b,int c) {
      to = a; cap = b; rev = c;
    }
};
// IMPORETANT, MAXV != MAXN
vector<Edge> G[MAXV];
int V, flow[MAXV];
void init(int _V){
    V = _V;
    for(int i=0; i<=V; i++) G[i].clear();
}
void add_edge(int f,int t,int c, bool directed){
  int s1 = G[f].size(), s2 = G[t].size();
    G[f].push_back(Edge(t,c,s2));
  G[t].push_back(Edge(f,c*!directed,s1));
}
int dfs(int v, int t) {
    if(v == t) return flow[t];
    for(Edge &e : G[v]){
```

```
        if(e.cap==0||flow[e.to]!=-1) continue;
        flow[e.to] = min(flow[v], e.cap);
        int f = dfs(e.to, t);
        if (f!=0) {
            e.cap -= f;
            G[e.to][e.rev].cap += f;
            return f;
        }
    }
    return 0;
}
int max_flow(int s,int t){
    int ans = 0, add = 0;
    do {
        fill(flow,flow+V+1,-1);
        flow[s] = INF;
        add = dfs(s, t);
        ans += add;
    } while (add != 0);
    return ans;
}
```

2.2 Dinic

```
struct Edge{
    int f,to,rev;
    T c;
    Edge(int _to,int _r,T _c):to(_to),rev(_r),c(_c){}
};

// IMPORETANT
// maxn is the number of vertices in the graph
// Not the N in the problem statement!!
vector<Edge> G[maxn];
int level[maxn],st, end, n;
int cur[maxn];

void init(int _n){
    n = _n;
    for(int i=0; i<=n; i++) G[i].clear();
}

void add_edge(int f,int t,T c, bool directed){
    int r1 = G[f].size(), r2 = G[t].size();
    G[f].push_back(Edge(t,r2,c));
    G[t].push_back(Edge(f,r1,directed?0:c));
}

bool BFS(int s,int t){
    queue<int> Q;
    for(int i=0; i<=n; i++) level[i] = 0;
    level[s] = 1;
    Q.push(s);
    while(!Q.empty()){
        int x = Q.front(); Q.pop();
        for(int i=0; i<G[x].size(); i++){
            Edge e = G[x][i];
            if(e.c==0 || level[e.to]) continue;
            level[e.to] = level[x] + 1;
            Q.push(e.to);
        }
    }
    return level[t]!=0;
}

T DFS(int s,T cur_flow){ // can't exceed c
    if(s==end) return cur_flow;
    T ans = 0, temp, total = 0;
    for(int& i=cur[s]; i<G[s].size(); i++){
        Edge &e = G[s][i];
        if(e.c==0 || level[e.to]!=level[s]+1) continue;
        temp = DFS(e.to, min(e.c, cur_flow));
        if(temp!=0){
            e.c -= temp;
            G[e.to][e.rev].c += temp;
            cur_flow -= temp;
            total += temp;
            if(cur_flow==0) break;
        }
    }
    return total;
}

T max_flow(int s,int t){
  /* If you want to incrementally doing maxFlow,
   you need to add the result manually.
   This function returns difference in that case. */
    T ans = 0;
    st = s, end = t;
    while(BFS(s,t)){
        while(true) {
            memset(cur, 0, sizeof(cur));
            T temp = DFS(s,INF);
            if(temp==0) break;
            ans += temp;
        }
    }
    return ans;
}
```

2.3 min_cost_flow

```
// 0-based
#define fi first
#define se second
struct Edge {
    int to,cap;
  int cost,rev;
};
static const int MAXV = 605;

int V,E;
vector<Edge> G[MAXV];

void init(int _V) {
    V=_V;
    for (int i=0;i<=V;i++) G[i].clear();
}
void add_edge(int fr, int to, int cap, int cost) {
  int a = G[fr].size(), b = G[to].size();
  G[fr].push_back({to,cap,cost,b});
  G[to].push_back({fr,0,-cost,a});
}
bool SPFA(int s, int t, int &ans_flow, int &ans_cost) {
  queue<int> que;
    PII pre[MAXV];
  int flow[MAXV], dist[MAXV];
  bool inque[MAXV];
    for (int i=0;i<=V;i++) {
        dist[i]=INF;
        inque[i]=false;
    }
    dist[s]=0;
    flow[s]=INF;
    inque[s]=true;
    que.push(s);
    while (!que.empty()) {
        int v=que.front(); que.pop();
        inque[v]=false;
        for (int i=0;i<G[v].size();i++) {
          const Edge &e = G[v][i];
            if (e.cap>0 && dist[v]+e.cost<dist[e.to]) {
                flow[e.to]=min(flow[v],e.cap);
                dist[e.to]=dist[v]+e.cost;
                pre[e.to]={v,i};
                if (!inque[e.to]) que.push(e.to),inque[e.to]=true;
            }
        }
    }
    if (dist[t]==INF) return false;
    //if (dist[t]>=0) return false;
    // Add above line -> min cost > max flow (priority)
    // Without        -> max flow > min cost

    int v=t,f=flow[t];
    ans_flow+=flow[t];
    ans_cost+=(dist[t]*flow[t]);
    while (v!=s) {
      Edge &e = G[pre[v].fi][pre[v].se];
        e.cap-=f;
        G[v][e.rev].cap+=f;
        v=pre[v].fi;
    }
    return true;
}
pair<int,int> min_cost_flow(int s, int t) {
    int ans_flow=0, ans_cost=0;
    while (SPFA(s,t,ans_flow,ans_cost));
    return make_pair(ans_flow,ans_cost);
}
```

3 Geometry

3.1 circle

```
/* Common tangent, circle is a point c and radius r */
void get_tangent(Point c, double r1, double r2, vector<Line> &ans) {
    double r = r2 - r1;
    double z = c.x*c.x + c.y*c.y;
    double d = z - r*r;
    if (d < -EPS) return;
    d = sqrt(abs(d));
    Line l;
    l.a = (c.x * r + c.y * d) / z;
    l.b = (c.y * r - c.x * d) / z;
    l.c = r1;
    ans.push_back(l);
}
vector<Line> tangents(Circle a, Circle b) {
  // Tangent line of two circles, may have 0, 1, 2, 3, 4, inf solutions
  // In case 0 or inf (a = b), no line will be reported. Otherwise,
  // this program always find 4 lines, even if some of them are the same.
    vector<Line> ans;
    for (int i=-1; i<=1; i+=2)
        for (int j=-1; j<=1; j+=2)
            get_tangent(b.c-a.c, a.r*i, b.r*j, ans);
    for (size_t i=0; i<ans.size(); ++i)
        ans[i].c -= ans[i].a * a.c.x + ans[i].b * a.c.y;
    return ans;
}
// Circle-Line intersection, line:ax+by+c=0
vector<Point> CL_intersection(Circle cir, Line li) {
  // li.pton(); // To Ax+By+C=0
  Point o = cir.c;
  li.c += li.a*o.x + li.b*o.y; // Shift w.r.t. cir.c

  vector<Point> res;
  double r = cir.r, a = li.a, b = li.b, c = li.c;
  double x0 = -a*c/(a*a+b*b), y0 = -b*c/(a*a+b*b);
  if (c*c > r*r*(a*a+b*b)+EPS) {
    return res; // No point
  }
  else if (abs(c*c - r*r*(a*a+b*b)) < EPS) {
    res.push_back({x0 + o.x, y0 + o.y}); // 1 point
  }
  else {
      double d = r*r - c*c/(a*a+b*b);
      double mult = sqrt (d / (a*a+b*b));
      double ax, ay, bx, by;
      ax = x0 + b * mult;
      bx = x0 - b * mult;
      ay = y0 - a * mult;
```

```
    by = y0 + a * mult;
    res.push_back({ax + o.x, ay + o.y}); // 2 points
    res.push_back({bx + o.x, by + o.y});
  }
  return res;
}

// Circle-circle intersection
vector<Point> CC_intersection(Circle a, Circle b) {
  if (a.c.x == b.c.x && a.c.y == b.c.y && a.r == b.r) {
    return vector<Point>(); // coincide, inf points
  }
  Point o = a.c;
  b.c = b.c - o; // Shift
  a.c = {0.0, 0.0};

  double x2 = b.c.x, y2 = b.c.y, r1 = a.r, r2 = b.r;
  Line li = {-2*x2, -2*y2, x2*x2 + y2*y2 + r1*r1 - r2*r2}; // Ax+By+C = 0
  vector<Point> res = CL_intersection(a, li);

  for (Point &p : res) {
    p.x += o.x;
    p.y += o.y;
  }

  return res;
}
```

3.2 convex_hull

```
void convex_hull(vector<Point> &ps, vector<Point> &hull) {
  // Find convex hull of ps, store in hull
  vector<Point> &stk=hull;
  stk.resize(ps.size()+1);
  sort(ps.begin(),ps.end()); // Using x to cmp, y secondary.
  int t=-1; // top
  for (int i=0;i<ps.size();i++) {
    // cross<-EPS -> count collinear, cross<EPS -> not
    while (t>=1&&(stk[t]-stk[t-1]).cross(ps[i]-stk[t])<EPS) t--;
    stk[++t]=ps[i];
  }
  int low=t;
  for (int i=ps.size()-2;i>=0;i--) {
    // cross<-EPS -> count collinear, cross<EPS -> not
    while (t>low&&(stk[t]-stk[t-1]).cross(ps[i]-stk[t])<EPS) t--;
    stk[++t]=ps[i];
  }
  stk.resize(t); // pop_back contain in this instruction
}
```

3.3 geometry

```
#include <bits/stdc++.h>
using namespace std;
const double PI=acos(-1);

struct Point {
  double x,y;
  double cross(const Point &v) const {
    return x*v.y-y*v.x;
  }
  double dot(const Point &v) const {
    return x*v.x+y*v.y;
  }
  Point normal() { // Normal vector to the left
    return {-y,x};
  }
  double angle(const Point &v) const {
    // Angle from *this to v in [-pi,pi].
    double ang = atan2(cross(v),dot(v));
    return ang < 0 ? ang + PI * 2 : ang;
  }
  double getA()const{//angle to x-axis
    T A=atan2(y,x);//<0 when exceed PI
    if(A<=-PI/2)A+=PI*2;
    return A;
  }
  Point rotate_about(double theta, const Point &p) const {
    // Rotate this point conterclockwise by theta about p
    double nx=x-p.x,ny=y-p.y;
    return {nx*cos(theta)-ny*sin(theta)+p.x, nx*sin(theta)+ny*cos(theta)+p.y};
  }
};

struct Line {
  // IMPORTANT, remember to transform between two-point form
  // and normal form by yourself, some methods may need them.
  Point p1,p2;
  double a,b,c; // ax+by+c=0
  Line(){}
  void pton() {
    a=p1.y-p2.y;
    b=p2.x-p1.x;
    c=-a*p1.x-b*p1.y;
  }
  double ori(const Point &p) {
    // For directed line, 0 if point on line
    // >0 if left, <0 if right
    return (p2-p1).cross(p-p1);
  }
  Point normal() { // normal vector to the left.
    Point dir=p2-p1;
    return {-dir.y,dir.x};
  }
  bool on_segment(const Point &p) {
    // Point on segment
    return relation(p)==0&&(p2-p).dot(p1-p)<=0;
  }
  bool parallel(const Line &l) {
    // Two line parallel
    return (p2-p1).cross(l.p2-l.p1)==0;
  }
  bool equal(const Line &l) {
    // Two line equal
    return relation(l.p1)==0&&relation(l.p2)==0;
  }
  bool cross_seg(const Line &seg) {
    // Line intersect segment
    Point dir=p2-p1;
    return dir.cross(seg.p1-p1)*dir.cross(seg.p2-p1)<=0;
  }
  int seg_intersect(const Line &s) const{
    // Two segment intersect
    // 0 -> no, 1 -> one point, -1 -> infinity
    Point dir=p2-p1, dir2=s.p2-s.p1;
    double c1=dir.cross(s.p2-p1);
    double c2=dir.cross(s.p1-p1);
    double c3=dir2.cross(p2-s.p1);
    double c4=dir2.cross(p1-s.p1);
    if (c1==0&&c2==0) {
      if((s.p2-p1).dot(s.p1-p1)>0&&(s.p2-p2).dot(s.p1-p2)>0&&
        (p1-s.p1).dot(p2-s.p1)>0&&(p1-s.p2).dot(p2-s.p2)>0)return 0;
      if(p1==s.p1&&(p2-p1).dot(s.p2-p1)<=0) return 1;
      if(p1==s.p2&&(p2-p1).dot(s.p1-p1)<=0) return 1;
      if(p2==s.p1&&(p1-p2).dot(s.p2-p2)<=0) return 1;
      if(p2==s.p2&&(p1-p2).dot(s.p1-p2)<=0) return 1;
      return -1;
    }else if(c1*c2<=0&&c3*c4<=0)return 1; // Be aware overflow
    return 0;
  }
  Point intersection(Line l) {
    // RE if d1.cross(d2) == 0 (parallel / coincide)
    Point d1 = p2 - p1, d2 = l.p2 - l.p1;
    return p1 + d1 * ((l.p1 - p1).cross(d2) / d1.cross(d2));
  }
  Point seg_intersection(Line &s) const {
    Point dir=p2-p1, dir2=s.p2-s.p1;
    // pton(); l.pton();
    double c1=dir.cross(s.p2-p1);
    double c2=dir.cross(s.p1-p1);
    double c3=dir2.cross(p2-s.p1);
    double c4=dir2.cross(p1-s.p1);
    if (c1==0&&c2==0) {
      if(p1==s.p1&&(p2-p1).dot(s.p2-p1)<=0) return p1;
      if(p1==s.p2&&(p2-p1).dot(s.p1-p1)<=0) return p1;
      if(p2==s.p1&&(p1-p2).dot(s.p2-p2)<=0) return p2;
      if(p2==s.p2&&(p1-p2).dot(s.p1-p2)<=0) return p2;
    }else if(c1*c2<=0&&c3*c4<=0)return line_intersection(s);
    // Reaches here means either INF or NOT ANY
    // Use seg_intersect to check OuO
    return {1234,4321};
  }
  double dist(const Point &p, bool is_segment) const {
    // Point to Line/segment
    Point dir=p2-p1,v=p-p1;
    if (is_segment) {
      if (dir.dot(v)<0) return v.len();
      if ((p1-p2).dot(p-p2)<0) return (p-p2).len();
    }
    double d=abs(dir.cross(v))/dir.len();
    return d;
  }
};

template<typename T>
struct polygon{
  vector<point<T> > p;//counterclockwise
  T area()const{
    T ans=0;
    for(int i=p.size()-1,j=0;j<(int)p.size();i=j++)
      ans+=p[i].cross(p[j]);
    return ans/2;
  }
  point<T> center_of_mass()const{
    T cx=0,cy=0,w=0;
    for(int i=p.size()-1,j=0;j<(int)p.size();i=j++){
      T a=p[i].cross(p[j]);
      cx+=(p[i].x+p[j].x)*a;
      cy+=(p[i].y+p[j].y)*a;
      w+=a;
    }
    return point<T>(cx/3/w,cy/3/w);
  }
  char ahas(const point<T>& t)const{//return 1 if in simple polygon, -1 if on, 0 if no.
    bool c=0;
    for(int i=0,j=p.size()-1;i<p.size();j=i++)
      if(line<T>(p[i],p[j]).point_on_segment(t))return -1;
      else if((p[i].y>t.y)!=(p[j].y>t.y)&&
      t.x<(p[j].x-p[i].x)*(t.y-p[i].y)/(p[j].y-p[i].y)+p[i].x)
        c=!c;
    return c;
  }
  char point_in_convex(const point<T>&x)const{
    int l=1,r=(int)p.size()-2;
    while(l<=r){//return 1 if in convex polygon, -1 if on, 0 if no.
      int mid=(l+r)/2;
      T a1=(p[mid]-p[0]).cross(x-p[0]);
      T a2=(p[mid+1]-p[0]).cross(x-p[0]);
      if(a1>=0&&a2<=0){
```

```cpp
      T res=(p[mid+1]-p[mid]).cross(x-p[mid]);
      return res>0?1:(res>=0?-1:0);
    }else if(a1<0)r=mid-1;
    else l=mid+1;
  }
  return 0;
}
vector<T> getA()const{//angle of each edge to x-axis
  vector<T>res;//must be increasing
  for(size_t i=0;i<p.size();++i)
    res.push_back((p[(i+1)%p.size()]-p[i]).getA());
  return res;
}
bool line_intersect(const vector<T>&A,const line<T> &l)const{//O(logN)
  int f1=upper_bound(A.begin(),A.end(),(l.p1-l.p2).getA())-A.begin();
  int f2=upper_bound(A.begin(),A.end(),(l.p2-l.p1).getA())-A.begin();
  return l.cross_seg(line<T>(p[f1],p[f2]));
}
polygon cut(const line<T> &l)const{
  polygon ans;//convex polygon cut by a line, left side of the line is remained.
  for(int n=p.size(),i=n-1,j=0;j<n;i=j++){
    if(l.ori(p[i])>=0){
      ans.p.push_back(p[i]);
      if(l.ori(p[j])<0)
        ans.p.push_back(l.line_intersection(line<T>(p[i],p[j])));
    }else if(l.ori(p[j])>0)
      ans.p.push_back(l.line_intersection(line<T>(p[i],p[j])));
  }
  return ans;
}
static bool graham_cmp(const point<T>& a,const point<T>& b){//CMP for finding hull
  return (a.x<b.x)||(a.x==b.x&&a.y<b.y);
}
void graham(vector<point<T> > &s){//convex hull
  sort(s.begin(),s.end(),graham_cmp);
  p.resize(s.size()+1);
  int m=0;
  for(size_t i=0;i<s.size();++i){
    while(m>=2&&(p[m-1]-p[m-2]).cross(s[i]-p[m-2])<=0)--m;
    p[m++]=s[i];
  }
  for(int i=s.size()-2,t=m+1;i>=0;--i){
    while(m>=t&&(p[m-1]-p[m-2]).cross(s[i]-p[m-2])<=0)--m;
    p[m++]=s[i];
  }
  if(s.size()>1)--m;
  p.resize(m);
}
T diameter(){
  int n=p.size(),t=1;
  T ans=0;p.push_back(p[0]);
  for(int i=0;i<n;i++){
    point<T> now=p[i+1]-p[i];
    while(now.cross(p[t+1]-p[i])>now.cross(p[t]-p[i]))t=(t+1)%n;
    ans=max(ans,(p[i]-p[t]).abs2());
  }
  return p.pop_back(),ans;
}
T min_cover_rectangle(){// find convex hull before call this
  int n=p.size(),t=1,r=1,l;
  if(n<3)return 0;
  T ans=1e99;p.push_back(p[0]);
  for(int i=0;i<n;i++){
    point<T> now=p[i+1]-p[i];
    while(now.cross(p[t+1]-p[i])>now.cross(p[t]-p[i]))t=(t+1)%n;
    while(now.dot(p[r+1]-p[i])>now.dot(p[r]-p[i]))r=(r+1)%n;
    if(!i)l=r;
    while(now.dot(p[l+1]-p[i])<=now.dot(p[l]-p[i]))l=(l+1)%n;
    T d=now.abs2();
    T tmp=now.cross(p[t]-p[i])*(now.dot(p[r]-p[i])-now.dot(p[l]-p[i]))/d;
    ans=min(ans,tmp);
  }
  return p.pop_back(),ans;
}
T dis2(polygon &pl){//square of distance of two convex polygon
  vector<point<T> > &P=p,&Q=pl.p;
  int n=P.size(),m=Q.size(),l=0,r=0;
for(int i=0;i<n;++i)if(P[i].y<P[l].y)l=i;
for(int i=0;i<m;++i)if(Q[i].y<Q[r].y)r=i;
  P.push_back(P[0]),Q.push_back(Q[0]);
  T ans=1e99;
  for(int i=0;i<n;++i){
    while((P[l]-P[l+1]).cross(Q[r+1]-Q[r])<0)r=(r+1)%m;
    ans=min(ans,line<T>(P[l],P[l+1]).seg_dis2(line<T>(Q[r],Q[r+1])));
    l=(l+1)%n;
  }
  return P.pop_back(),Q.pop_back(),ans;
}
static char sign(const point<T>&t){
  return (t.y==0?t.x:t.y)<0;
}
static bool angle_cmp(const line<T>& A,const line<T>& B){
  point<T> a=A.p2-A.p1,b=B.p2-B.p1;
  return sign(a)<sign(b)||(sign(a)==sign(b)&&a.cross(b)>0);
}
int halfplane_intersection(vector<line<T> > &s){
  sort(s.begin(),s.end(),angle_cmp);//half plane is left side of the line
  int L,R,n=s.size();
  vector<point<T> > px(n);
  vector<line<T> > q(n);
  q[L=R=0]=s[0];
  for(int i=1;i<n;++i){
    while(L<R&&s[i].ori(px[R-1])<=0)--R;
    while(L<R&&s[i].ori(px[L])<=0)++L;
    q[++R]=s[i];
    if(q[R].parallel(q[R-1])){
      --R;
      if(q[R].ori(s[i].p1)>0)q[R]=s[i];
    }
    if(L<R)px[R-1]=q[R-1].line_intersection(q[R]);
  }
  while(L<R&&q[L].ori(px[R-1])<=0)--R;
  p.clear();
  if(R-L<=1)return 0;
  px[R]=q[R].line_intersection(q[L]);
  for(int i=L;i<=R;++i)p.push_back(px[i]);
  return R-L+1;
}
};
template<typename T>
struct triangle{
  point<T> a,b,c;
  triangle(){}
  triangle(const point<T> &a,const point<T> &b,const point<T> &c):a(a),b(b),c(c){}
  T area()const{
    T t=(b-a).cross(c-a)/2;
    return t>0?t:-t;
  }
  point<T> barycenter()const{//center of mass
    return (a+b+c)/3;
  }
  point<T> circumcenter()const{//outer center
    static line<T> u,v;
    u.p1=(a+b)/2;
    u.p2=point<T>(u.p1.x-a.y+b.y,u.p1.y+a.x-b.x);
    v.p1=(a+c)/2;
    v.p2=point<T>(v.p1.x-a.y+c.y,v.p1.y+a.x-c.x);
    return u.line_intersection(v);
  }
  point<T> incenter()const{//inner center
    T A=sqrt((b-c).abs2()),B=sqrt((a-c).abs2()),C=sqrt((a-b).abs2());
    return point<T>(A*a.x+B*b.x+C*c.x,A*a.y+B*b.y+C*c.y)/(A+B+C);
  }
  point<T> perpencenter()const{//perpendicular(?) center
    return barycenter()*3-circumcenter()*2;
  }
};
```

3.4 KD_TREE

```cpp
const int MXN = 100005;
struct KDTree {
  struct Node {
    int x,y,x1,y1,x2,y2;
    int id,f;
    Node *L, *R;
  }tree[MXN];
  int n;
  Node *root;
  LL dis2(int x1, int y1, int x2, int y2) {
    LL dx = x1-x2;
    LL dy = y1-y2;
    return dx*dx+dy*dy;
  }
  static bool cmpx(Node& a, Node& b){ return a.x<b.x; }
  static bool cmpy(Node& a, Node& b){ return a.y<b.y; }
  void init(vector<pair<int,int>> ip) {
    n = ip.size();
    for (int i=0; i<n; i++) {
      tree[i].id = i;
      tree[i].x = ip[i].first;
      tree[i].y = ip[i].second;
    }
    root = build_tree(0, n-1, 0);
  }
  Node* build_tree(int L, int R, int dep) {
    if (L>R) return nullptr;
    int M = (L+R)/2;
    tree[M].f = dep%2;
    nth_element(tree+L, tree+M, tree+R+1, tree[M].f ? cmpy : cmpx);
    tree[M].x1 = tree[M].x2 = tree[M].x;
    tree[M].y1 = tree[M].y2 = tree[M].y;

    tree[M].L = build_tree(L, M-1, dep+1);
    if (tree[M].L) {
      tree[M].x1 = min(tree[M].x1, tree[M].L->x1);
      tree[M].x2 = max(tree[M].x2, tree[M].L->x2);
      tree[M].y1 = min(tree[M].y1, tree[M].L->y1);
      tree[M].y2 = max(tree[M].y2, tree[M].L->y2);
    }
    tree[M].R = build_tree(M+1, R, dep+1);
    if (tree[M].R) {
      tree[M].x1 = min(tree[M].x1, tree[M].R->x1);
      tree[M].x2 = max(tree[M].x2, tree[M].R->x2);
      tree[M].y1 = min(tree[M].y1, tree[M].R->y1);
      tree[M].y2 = max(tree[M].y2, tree[M].R->y2);
    }
    return tree+M;
  }
  int touch(Node* r, int x, int y, LL d2){
    LL dis = sqrt(d2)+1;
    if (x<r->x1-dis || x>r->x2+dis ||
        y<r->y1-dis || y>r->y2+dis)
      return 0;
    return 1;
  }
  void nearest(Node* r, int x, int y,
               int &mID, LL &md2){
    if (!r || !touch(r, x, y, md2)) return;
    LL d2 = dis2(r->x, r->y, x, y);
```

```
    if (d2 < md2 || (d2 == md2 && mID < r->id)) {
      mID = r->id;
      md2 = d2;
    }
    // search order depends on split dim
    if ((r->f == 0 && x < r->x) ||
        (r->f == 1 && y < r->y)) {
      nearest(r->L, x, y, mID, md2);
      nearest(r->R, x, y, mID, md2);
    } else {
      nearest(r->R, x, y, mID, md2);
      nearest(r->L, x, y, mID, md2);
    }
  }
  int query(int x, int y) {
    int id = 1029384756;
    LL d2 = 102938475612345678LL;
    nearest(root, x, y, id, d2);
    return id;
  }
}tree;
```

3.5 smallest_circle

```
using PT=point<T>; using CPT=const PT;
PT circumcenter(CPT &a,CPT &b,CPT &c){
  PT u=b-a, v=c-a;
  T c1=u.abs2()/2,c2=v.abs2()/2;
  T d=u.cross(v);
  return PT(a.x+(v.y*c1-u.y*c2)/d,a.y+(u.x*c2-v.x*c1)/d);
}
void solve(PT p[],int n,PT &c,T &r2){
  random_shuffle(p,p+n);
  c=p[0]; r2=0; // c,r2 = center,radius square
for(int i=1;i<n;i++)if((p[i]-c).abs2()>r2){
    c=p[i]; r2=0;
for(int j=0;j<i;j++)if((p[j]-c).abs2()>r2){
      c.x=(p[i].x+p[j].x)/2;
      c.y=(p[i].y+p[j].y)/2;
      r2=(p[j]-c).abs2();
for(int k=0;k<j;k++)if((p[k]-c).abs2()>r2){
        c=circumcenter(p[i],p[j],p[k]);
        r2=(p[i]-c).abs2();
      }
    }
  }
}
```

3.6 最近點對

```
template<typename _IT=point<T>* >
T cloest_pair(_IT L, _IT R){
  if(R-L <= 1) return INF;
  _IT mid = L+(R-L)/2;
  T x = mid->x;
  T d = min(cloest_pair(L,mid),cloest_pair(mid,R));
  inplace_merge(L, mid, R, ycmp);
  static vector<point> b; b.clear();
  for(auto u=L;u<R;++u){
    if((u->x-x)*(u->x-x)>=d) continue;
    for(auto v=b.rbegin();v!=b.rend();++v){
      T dx=u->x-v->x, dy=u->y-v->y;
      if(dy*dy>=d) break;
      d=min(d,dx*dx+dy*dy);
    }
    b.push_back(*u);
  }
  return d;
}
T closest_pair(vector<point<T>> &v){
  sort(v.begin(),v.end(),xcmp);
  return closest_pair(v.begin(),v.end());
}
```

4 Graph

4.1 3989_ 穩定婚姻

```
#include <bits/stdc++.h>

using namespace std;

const int maxn = 1100;

int manWant[maxn][maxn], nextW[maxn];
int women[maxn][maxn], order[maxn][maxn];
int wife[maxn], husband[maxn];
queue<int> singleDog;

void engage(int m, int w){
    if(husband[w]!=0){
        wife[ husband[w] ] = 0;
        singleDog.push( husband[w] );
        husband[w] = 0;
    }
    husband[w] = m;
    wife[m] = w;
    // cout << m << " --> " << w << endl;
}
int main()
{
    int Time, n, cas = 0;
    scanf("%d",&Time);

    while(Time-- && scanf("%d",&n)==1){
        for(int i=1; i<=n; i++){
            for(int j=1; j<=n; j++) scanf("%d",&manWant[i][j]);
            nextW[i] = 1;
            wife[i] = 0;
            singleDog.push(i);
        }

        for(int i=1; i<=n; i++){
            for(int j=1; j<=n; j++){
                scanf("%d",&women[i][j]);
                order[i][ women[i][j] ] = j;
            }
            husband[i] = 0;
        }

        while(!singleDog.empty()){
            int x = singleDog.front(); singleDog.pop();
            // cout << x << endl;
            int to = manWant[x][nextW[x]++];

            if(husband[to]==0) engage(x, to);
            else if(order[to][husband[to]] > order[to][x]) engage(x, to);
            else singleDog.push(x);
        }
        if(cas++) printf("\n");
        for(int i=1; i<=n; i++) printf("%d\n", wife[i]);
    }
    return 0;
}
```

4.2 blossom

```
struct Blossom {
  #define MAXN 505 // Max solvable problem, DON'T CHANGE
  // 1-based, IMPORTANT
  vector<int> g[MAXN];
  int parent[MAXN], match[MAXN], belong[MAXN], state[MAXN];
  int n;
  int lca(int u, int v) {
    static int cases = 0, used[MAXN] = {};
    for (++cases; ; swap(u, v)) {
      if (u == 0)
        continue;
      if (used[u] == cases)
        return u;
      used[u] = cases;
      u = belong[parent[match[u]]];
    }
  }
  void flower(int u, int v, int l, queue<int> &q) {
    while (belong[u] != l) {
      parent[u] = v, v = match[u];
      if (state[v] == 1)
        q.push(v), state[v] = 0;
      belong[u] = belong[v] = l, u = parent[v];
    }
  }
  bool bfs(int u) {
    for (int i = 0; i <= n; i++)
      belong[i] = i;
    memset(state, -1, sizeof(state[0])*(n+1));
    queue<int> q;
    q.push(u), state[u] = 0;
    while (!q.empty()) {
      u = q.front(), q.pop();
      for (int i = 0; i < g[u].size(); i++) {
        int v = g[u][i];
        if (state[v] == -1) {
          parent[v] = u, state[v] = 1;
          if (match[v] == 0) {
            for (int prev; u; v = prev, u = parent[v]) {
              prev = match[u];
              match[u] = v;
              match[v] = u;
            }
            return 1;
          }
          q.push(match[v]), state[match[v]] = 0;
        } else if (state[v] == 0 && belong[v] != belong[u]) {
          int l = lca(u, v);
          flower(v, u, l, q);
          flower(u, v, l, q);
        }
      }
    }
    return 0;
  }
  int blossom() {
    memset(parent, 0, sizeof(parent[0])*(n+1));
    memset(match, 0, sizeof(match[0])*(n+1));
    int ret = 0;
    for (int i = 1; i <= n; i++) {
      if (match[i] == 0 && bfs(i))
        ret++;
    }
    return ret;
  }
  void addEdge(int x, int y) {
    g[x].push_back(y), g[y].push_back(x);
  }
  void init(int _n) {
    n = _n;
    for (int i = 0; i <= n; i++)
      g[i].clear();
  }
} algo;
```

4.3 Eulerian_cycle

```
// The cycle will be output in reverse order
// if you want eulerian "path",
// Add one edge, find cycle, transform to path
void dfs(int v) {
  while(!g[v].empty()) {
    int u = g[v].back();
    g[v].pop_back();
    dfs(u);
    output(Edge(v, u)); // v to u
  }
}
```

4.4 KM

```cpp
// Maximum Bipartite Weighted Matching (Perfect Match)
static const int MXN = 650;
static const int INF = 2147483647; // LL
int n,match[MXN],vx[MXN],vy[MXN];
int edge[MXN][MXN],lx[MXN],ly[MXN],slack[MXN];
// ^^^^ LL
void init(int _n){
  n = _n;
  for(int i=0; i<n; i++) for(int j=0; j<n; j++)
    edge[i][j] = 0;
}
void addEdge(int x, int y, int w) // LL
{ edge[x][y] = w; }
bool DFS(int x){
  vx[x] = 1;
  for (int y=0; y<n; y++){
    if (vy[y]) continue;
    if (lx[x]+ly[y] > edge[x][y]){
      slack[y]=min(slack[y], lx[x]+ly[y]-edge[x][y]);
    } else {
      vy[y] = 1;
      if (match[y] == -1 || DFS(match[y]))
      { match[y] = x; return true; }
    }
  }
  return false;
}
int solve(){
  fill(match,match+n,-1);
  fill(lx,lx+n,-INF); fill(ly,ly+n,0);
  for (int i=0; i<n; i++)
    for (int j=0; j<n; j++)
      lx[i] = max(lx[i], edge[i][j]);
  for (int i=0; i<n; i++){
    fill(slack,slack+n,INF);
    while (true){
      fill(vx,vx+n,0); fill(vy,vy+n,0);
      if ( DFS(i) ) break;
      int d = INF; // long long
      for (int j=0; j<n; j++)
        if (!vy[j]) d = min(d, slack[j]);
      for (int j=0; j<n; j++){
        if (vx[j]) lx[j] -= d;
        if (vy[j]) ly[j] += d;
        else slack[j] -= d;
      }
    }
  }
  int res=0;
  for (int i=0; i<n; i++)
    res += edge[match[i]][i];
  return res;
}
```

4.5 MaximumClique

```cpp
struct MaxClique{
  static const int MAXN=105;
  int N,ans;
  int g[MAXN][MAXN],dp[MAXN],stk[MAXN][MAXN];
  int sol[MAXN],tmp[MAXN];//sol[0~ans-1]為答案
  void init(int n){
    N=n;//0-base
    memset(g,0,sizeof(g));
  }
  void add_edge(int u,int v){
    g[u][v]=g[v][u]=1;
  }
  int dfs(int ns,int dep){
    if(!ns){
      if(dep>ans){
        ans=dep;
        memcpy(sol,tmp,sizeof tmp);
        return 1;
      }else return 0;
    }
    for(int i=0;i<ns;++i){
      if(dep+ns-i<=ans)return 0;
      int u=stk[dep][i],cnt=0;
      if(dep+dp[u]<=ans)return 0;
      for(int j=i+1;j<ns;++j){
        int v=stk[dep][j];
        if(g[u][v])stk[dep+1][cnt++]=v;
      }
      tmp[dep]=u;
      if(dfs(cnt,dep+1))return 1;
    }
    return 0;
  }
  int clique(){
    int u,v,ns;
    for(ans=0,u=N-1;u>=0;--u){
      for(ns=0,tmp[0]=u,v=u+1;v<N;++v)
        if(g[u][v])stk[1][ns++]=v;
      dfs(ns,1),dp[u]=ans;
    }
    return ans;
  }
};
```

4.6 MinimumMeanCycle

```cpp
#include<cfloat> //for DBL_MAX
int dp[MAXN][MAXN]; // 1-base,O(NM)
vector<tuple<int,int,int>> edge;
double mmc(int n){//allow negative weight
  const int INF=0x3f3f3f3f;
  for(int t=0;t<n;++t){
    memset(dp[t+1],0x3f,sizeof(dp[t+1]));
    for(const auto &e:edge){
      int u,v,w;
      tie(u,v,w) = e;
      dp[t+1][v]=min(dp[t+1][v],dp[t][u]+w);
    }
  }
  double res = DBL_MAX;
  for(int u=1;u<=n;++u){
    if(dp[n][u]==INF) continue;
    double val = -DBL_MAX;
    for(int t=0;t<n;++t)
      val=max(val,(dp[n][u]-dp[t][u])*1.0/(n-t));
    res=min(res,val);
  }
  return res;
}
```

4.7 SAT2

```cpp
int N, sid[MAXV*2]; // all 1-based
bool vis[MAXV*2], sol[MAXV]; // 1 if i is true
vector<int> stk, G[MAXV*2], Gr[MAXV*2];
void init(int _N) {
  N = _N; // number of variable
  for (int i = 0; i <= 2 * N; i++) {
    G[i].clear();
    Gr[i].clear();
  }
}
int get_not(int x) {
  return x <= N ? x + N : x - N;
}
void add_edge(int x, int y) {
  G[x].push_back(y);
  Gr[y].push_back(x);
}
void add_or(int x, int y) {
  add_edge(get_not(x), y);
  add_edge(get_not(y), x);
}
void dfs(int v) {
  vis[v] = 1;
  for (int to : G[v]) {
    if (!vis[to]) {
      dfs(to);
    }
  }
  stk.push_back(v);
}
void rdfs(int v, int root) {
  sid[v] = root;
  for (int to : Gr[v]) {
    if (sid[to] == 0) {
      rdfs(to, root);
    }
  }
}
bool solve() {
  int V = 2 * N;
  stk.clear();
  fill(vis, vis + V + 1, 0);
  fill(sid, sid + V + 1, 0);
  for (int i = 1; i <= V; i++) {
    if (!vis[i]) {
      dfs(i);
    }
  }
  int cnt = 0;
  for (int i = (int) stk.size() - 1; i >= 0; i--) {
    if (sid[stk[i]] == 0) {
      rdfs(stk[i], ++cnt);
    }
  }

  for (int i = 1; i <= N; i++) {
    if (sid[i] == sid[i + N]) return false;
    sol[i] = (sid[i + N] < sid[i]);
  }
  return true;
}
```

4.8 tree_isomorphism

```cpp
// Hash the parenthesis tuple given by AHU algorithm. O(nlgn)
// If you want exact, discretize the sorted euler tour layer by layer.
// The input should be a rooted tree, for unrooted, find centroid or center then do something.
#define ULL unsigned long long
static const ULL BASE = 7;
int sz[MAXN];
ULL dfs(int v, int p, vector<int> adj[]) {
  ULL res = 1;
  vector<pair<ULL, int>> h;
  sz[v] = 1;
  for (int to : adj[v]) {
    if (to == p) continue;
    h.push_back({dfs(to, v, adj), sz[to]});
    sz[v] += sz[to];
  }

  sort(h.begin(), h.end());
  for (auto it : h) {
    res *= qpow(BASE, it.second);
    res += it.first;
  }
  return res;
}
ULL get_hash(int root, vector<int> adj[]) {
  return dfs(root, root, adj);
}
```

4.9 一般圖最小權完美匹配

```cpp
struct Graph {
  // Minimum General Weighted Matching (Perfect Match) 0-base
  static const int MXN = 105;
  int n, edge[MXN][MXN];
  int match[MXN],dis[MXN],onstk[MXN];
  vector<int> stk;
  void init(int _n) {
    n = _n;
    for (int i=0; i<n; i++)
```

```cpp
    for (int j=0; j<n; j++)
      edge[i][j] = 0;
  }
  void add_edge(int u, int v, int w) {
    edge[u][v] = edge[v][u] = w;
  }
  bool SPFA(int u){
    if (onstk[u]) return true;
    stk.push_back(u);
    onstk[u] = 1;
    for (int v=0; v<n; v++){
      if (u != v && match[u] != v && !onstk[v]){
        int m = match[v];
        if (dis[m] > dis[u] - edge[v][m] + edge[u][v]){
          dis[m] = dis[u] - edge[v][m] + edge[u][v];
          onstk[v] = 1;
          stk.push_back(v);
          if (SPFA(m)) return true;
          stk.pop_back();
          onstk[v] = 0;
        }
      }
    }
    onstk[u] = 0;
    stk.pop_back();
    return false;
  }
  int solve() {
    // find a match
    for (int i=0; i<n; i+=2){
      match[i] = i+1, match[i+1] = i;
    }
    for(;;){
      int found = 0;
      for (int i=0; i<n; i++) dis[i] = onstk[i] = 0;
      for (int i=0; i<n; i++){
        stk.clear();
        if (!onstk[i] && SPFA(i)){
          found = 1;
          while (stk.size()>=2){
            int u = stk.back(); stk.pop_back();
            int v = stk.back(); stk.pop_back();
            match[u] = v;
            match[v] = u;
          }
        }
      }
      if (!found) break;
    }
    int ret = 0;
    for (int i=0; i<n; i++)
      ret += edge[i][match[i]];
    ret /= 2;
    return ret;
  }
}graph;
```

4.10 全局最小割

```cpp
const int INF=0x3f3f3f3f;
template<typename T>
struct stoer_wagner{// 0-base
  static const int MAXN=150;
  T g[MAXN][MAXN],dis[MAXN];
  int nd[MAXN],n,s,t;
  void init(int _n){
    n=_n;
    for(int i=0;i<n;++i)
      for(int j=0;j<n;++j)g[i][j]=0;
  }
  void add_edge(int u,int v,T w){
    g[u][v]=g[v][u]+=w;
  }
  T min_cut(){
    T ans=INF;
    for(int i=0;i<n;++i)nd[i]=i;
    for(int ind,tn=n;tn>1;--tn){
      for(int i=1;i<tn;++i)dis[nd[i]]=0;
      for(int i=1;i<tn;++i){
        ind=i;
        for(int j=i;j<tn;++j){
          dis[nd[j]]+=g[nd[i-1]][nd[j]];
          if(dis[nd[ind]]<dis[nd[j]])ind=j;
        }
        swap(nd[ind],nd[i]);
      }
      if(ans>dis[nd[ind]])ans=dis[t=nd[ind]],s=nd[ind-1];
      for(int i=0;i<tn;++i)
        g[nd[ind-1]][nd[i]]=g[nd[i]][nd[ind-1]]+=g[nd[i]][nd[ind]];
    }
    return ans;
  }
};
```

4.11 平面圖判定

```cpp
static const int MAXN = 20;
struct Edge{
  int u, v;
  Edge(int s, int d) : u(s), v(d) {}
};
bool isK33(int n, int degree[]){
  int t = 0, z = 0;
  for(int i=0;i<n;++i){
    if(degree[i] == 3)++t;
    else if(degree[i] == 0)++z;
    else return false;
  }
  return t == 6 && t + z == n;
}
bool isK5(int n, int degree[]){
  int f = 0, z = 0;
  for(int i=0;i<n;++i){
    if(degree[i] == 4)++f;
    else if(degree[i] == 0)++z;
    else return false;
  }
  return f == 5 && f + z == n;
}
// it judge a given graph is Homeomorphic with K33 or K5
bool isHomeomorphic(bool G[MAXN][MAXN], const int n){
  for(;;){
    int cnt = 0;
    for(int i=0;i<n;++i){
      vector<Edge> E;
      for(int j=0;j<n&&E.size()<3;++j)
        if(G[i][j] && i != j)
          E.push_back(Edge(i, j));
      if(E.size() == 1){
        G[i][E[0].v] = G[E[0].v][i] = false;
      }else if(E.size() == 2){
        G[i][E[0].v] = G[E[0].v][i] = false;
        G[i][E[1].v] = G[E[1].v][i] = false;
        G[E[0].v][E[1].v] = G[E[1].v][E[0].v] = true;
        ++cnt;
      }
    }
    if(cnt == 0)break;
  }
  static int degree[MAXN];
  fill(degree, degree + n, 0);
  for(int i=0;i<n;++i){
    for(int j=i+1; j<n; ++j){
      if(!G[i][j])continue;
      ++degree[i];
      ++degree[j];
    }
  }
  return  !(isK33(n, degree) || isK5(n, degree));
}
```

4.12 最小斯坦納樹 DP

```cpp
//n個點，其中r個要構成斯坦納樹
//答案在max(dp[(1<<r)-1][k]) k=0~n-1
//p表示要構成斯坦納樹的點集
//O( n^3 + n*3^r + n^2*2^r )
#define REP(i,n) for(int i=0;i<(int)n;++i)
const int MAXN=30,MAXM=8;// 0-base
const int INF=0x3f3f3f3f;
int dp[1<<MAXM][MAXN];
int g[MAXN][MAXN];//圖
void init(){memset(g,0x3f,sizeof(g));}
void add_edge(int u,int v,int w){
  g[u][v]=g[v][u]=min(g[v][u],w);
}
void steiner(int n,int r,int *p){
  REP(k,n)REP(i,n)REP(j,n)
    g[i][j]=min(g[i][j],g[i][k]+g[k][j]);
  REP(i,n)g[i][i]=0;
  REP(i,r)REP(j,n)dp[1<<i][j]=g[p[i]][j];
  for(int i=1;i<(1<<r);++i){
    if(!(i&(i-1)))continue;
    REP(j,n)dp[i][j]=INF;
    REP(j,n){
      int tmp=INF;
      for(int s=i&(i-1);s;s=i&(s-1))
        tmp=min(tmp,dp[s][j]+dp[i^s][j]);
      REP(k,n)dp[i][k]=min(dp[i][k],g[j][k]+tmp);
    }
  }
}
```

4.13 最小樹形圖 _ 朱劉

```cpp
template<typename T>
struct zhu_liu{
  static const int MAXN=110,MAXM=10005;
  struct node{
    int u,v;
    T w,tag;
    node *l,*r;
    node(int u=0,int v=0,T w=0):u(u),v(v),w(w),tag(0),l(0),r(0){}
    void down(){
      w+=tag;
      if(l)l->tag+=tag;
      if(r)r->tag+=tag;
      tag=0;
    }
  }mem[MAXM];//靜態記憶體
  node *pq[MAXN*2],*E[MAXN*2];
  int st[MAXN*2],id[MAXN*2],m;
  void init(int n){
    for(int i=1;i<=n;++i){
      pq[i]=E[i]=0, st[i]=id[i]=i;
    }m=0;
  }
  node *merge(node *a,node *b){//skew heap
    if(!a||!b)return a?a:b;
    a->down(),b->down();
    if(b->w<a->w)return merge(b,a);
    swap(a->l,a->r);
    a->l=merge(b,a->l);
    return a;
  }
  void add_edge(int u,int v,T w){
    if(u!=v)pq[v]=merge(pq[v],&(mem[m++]=node(u,v,w)));
  }
  int find(int x,int *st){
    return st[x]==x?x:st[x]=find(st[x],st);
  }
  T build(int root,int n){
    T ans=0;int N=n,all=n;
    for(int i=1;i<=N;++i){
      if(i==root||!pq[i])continue;
      while(pq[i]){
        pq[i]->down(),E[i]=pq[i];
        pq[i]=merge(pq[i]->l,pq[i]->r);
        if(find(E[i]->u,id)!=find(i,id))
          break;
      }
      if(find(E[i]->u,id)==find(i,id))
        continue;
      ans+=E[i]->w;
      if(find(E[i]->u,st)==find(i,st)){
```

```
        if(pq[i])pq[i]->tag-=E[i]->w;
        pq[++N]=pq[i];id[N]=N;
        for(int u=find(E[i]->u,id);u!=i;u=find(E[u]->u,id)){
          if(pq[u])pq[u]->tag-=E[u]->w;
          id[find(u,id)]=N;
          pq[N]=merge(pq[N],pq[u]);
        }
        st[N]=find(i,st);
        id[find(i,id)]=N;
      }else st[find(i,st)]=find(E[i]->u,st),--all;
    }
    return all==1?ans:-INT_MAX;//圖不連通就無解
  }
};
```

4.14 穩定婚姻模板

```
queue<int> Q;
for ( i : 所有考生 ) {
  設定在第0志願;
  Q.push(考生i);
}
while(Q.size()){
  當前考生=Q.front();Q.pop();
  while ( 此考生未分發 ) {
    指標移到下一志願;
    if ( 已經沒有志願 or 超出志願總數 ) break;
    計算該考生在該科系加權後的總分;
    if ( 不符合科系需求 ) continue;
    if ( 目前科系有餘額 ) {
      依加權後分數高低順序將考生id加入科系錄取名單中;
      break;
    }
    if ( 目前科系已額滿 ) {
      if ( 此考生成績比最低分數還高 ) {
        依加權後分數高低順序將考生id加入科系錄取名單;
        Q.push(被踢出的考生);
      }
    }
  }
}
```

5 Linear_Programming

5.1 simplex

```
/*target:
  max \sum_{j=1}^n A_{0,j}*x_j
condition:
  \sum_{j=1}^n A_{i,j}*x_j <= A_{i,0} |i=1~m
  x_j >= 0 |j=1~n
VDB = vector<double>*/
template<class VDB>
VDB simplex(int m,int n,vector<VDB> a){
  vector<int> left(m+1), up(n+1);
  iota(left.begin(), left.end(), n);
  iota(up.begin(), up.end(), 0);
  auto pivot = [&](int x, int y){
    swap(left[x], up[y]);
    auto k = a[x][y]; a[x][y] = 1;
    vector<int> pos;
    for(int j = 0; j <= n; ++j){
      a[x][j] /= k;
      if(a[x][j] != 0) pos.push_back(j);
    }
    for(int i = 0; i <= m; ++i){
      if(a[i][y]==0 || i == x) continue;
      k = a[i][y], a[i][y] = 0;
      for(int j : pos) a[i][j] -= k*a[x][j];
    }
  };
  for(int x,y;;){
    for(int i=x=1; i <= m; ++i)
      if(a[i][0]<a[x][0]) x = i;
    if(a[x][0]>=0) break;
    for(int j=y=1; j <= n; ++j)
      if(a[x][j]<a[x][y]) y = j;
    if(a[x][y]>=0) return VDB();//infeasible
    pivot(x, y);
  }
  for(int x,y;;){
    for(int j=y=1; j <= n; ++j)
      if(a[0][j] > a[0][y]) y = j;
    if(a[0][y]<=0) break;
    x = -1;
    for(int i=1; i<=m; ++i) if(a[i][y] > 0)
      if(x == -1 || a[i][0]/a[i][y]
        < a[x][0]/a[x][y]) x = i;
    if(x == -1) return VDB();//unbounded
    pivot(x, y);
  }
  VDB ans(n + 1);
  for(int i = 1; i <= m; ++i)
    if(left[i] <= n) ans[left[i]] = a[i][0];
  ans[0] = -a[0][0];
  return ans;
}
```

6 Number_Theory

6.1 basic

```
template<typename T>
void gcd(const T &a,const T &b,T &d,T &x,T &y){
  if(!b) d=a,x=1,y=0;
  else gcd(b,a%b,d,y,x), y-=x*(a/b);
}
long long int phi[N+1];
void phiTable(){
  for(int i=1;i<=N;i++)phi[i]=i;
  for(int i=1;i<=N;i++)for(x=i*2;x<=N;x+=i)phi[x]-=phi[i];
}
void all_divdown(const LL &n) {// all n/x
  for(LL a=1;a<=n;a=n/(n/(a+1))){
    // dosomething;
  }
}
const int MAXPRIME = 1000000;
int iscom[MAXPRIME], prime[MAXPRIME], primecnt;
int phi[MAXPRIME], mu[MAXPRIME];
void sieve(void){
  memset(iscom,0,sizeof(iscom));
  primecnt = 0;
  phi[1] = mu[1] = 1;
  for(int i=2;i<MAXPRIME;++i) {
    if(!iscom[i]) {
      prime[primecnt++] = i;
      mu[i] = -1;
      phi[i] = i-1;
    }
    for(int j=0;j<primecnt;++j) {
      int k = i * prime[j];
      if(k>=MAXPRIME) break;
      iscom[k] = prime[j];
      if(i%prime[j]==0) {
        mu[k] = 0;
        phi[k] = phi[i] * prime[j];
        break;
      } else {
        mu[k] = -mu[i];
        phi[k] = phi[i] * (prime[j]-1);
      }
    }
  }
}

bool g_test(const LL &g, const LL &p, const vector<LL> &v) {
  for(int i=0;i<v.size();++i)
    if(modexp(g,(p-1)/v[i],p)==1)
      return false;
  return true;
}
LL primitive_root(const LL &p) {
  if(p==2) return 1;
  vector<LL> v;
  Factor(p-1,v);
  v.erase(unique(v.begin(), v.end()), v.end());
  for(LL g=2;g<p;++g)
    if(g_test(g,p,v))
      return g;
  puts("primitive_root NOT FOUND");
  return -1;
}
int Legendre(const LL &a, const LL &p) { return modexp(a%p,(p-1)/2,p); }

LL inv(const LL &a, const LL &n) {
  LL d,x,y;
  gcd(a,n,d,x,y);
  return d==1 ? (x+n)%n : -1;
}

int inv[maxN];
LL invtable(int n,LL P){
  inv[1]=1;
  for(int i=2;i<n;++i)
    inv[i]=(P-(P/i))*inv[P%i]%P;
}

LL Tonelli_Shanks(const LL &n, const LL &p) {
  // x^2 = n ( mod p )
  if(n==0) return 0;
  if(Legendre(n,p)!=1) while(1) { puts("SQRT ROOT does not exist"); }
  int S = 0;
  LL Q = p-1;
  while( !(Q&1) ) { Q>>=1; ++S; }
  if(S==1) return modexp(n%p,(p+1)/4,p);
  LL z = 2;
  for(;Legendre(z,p)!=-1;++z)
  LL c = modexp(z,Q,p);
  LL R = modexp(n%p,(Q+1)/2,p), t = modexp(n%p,Q,p);
  int M = S;
  while(1) {
    if(t==1) return R;
    LL b = modexp(c,1L<<(M-i-1),p);
    R = LLmul(R,b,p);
    t = LLmul( LLmul(b,b,p), t, p);
    c = LLmul(b,b,p);
    M = i;
  }
  return -1;
}

template<typename T>
T Euler(T n){
  T ans=n;
  for(T i=2;i*i<=n;++i){
    if(n%i==0){
      ans=ans/i*(i-1);
      while(n%i==0)n/=i;
    }
  }
  if(n>1)ans=ans/n*(n-1);
  return ans;
}

//Chinese_remainder_theorem
template<typename T>
T pow_mod(T n,T k,T m){
  T ans=1;
  for(n=(n>=m?n%m:n);k;k>>=1){
    if(k&1)ans=ans*n%m;
    n=n*n%m;
  }
  return ans;
}
template<typename T>
T crt(vector<T> &m,vector<T> &a){
  T M=1,tM,ans=0;
  for(int i=0;i<(int)m.size();++i)M*=m[i];
  for(int i=0;i<(int)a.size();++i){
    tM=M/m[i];
    ans=(ans+(a[i]*tM%M)*pow_mod(tM,Euler(m[i])-1,m[i]))%M;
```

```
    /* If m is prime, Euler(m[i])-1=m[i]-2, or use extgcd? */
  }
  return ans;
}

//java code
//continued fraction of sqrt(n)
public static void Pell(int n){
  BigInteger N,p1,p2,q1,q2,a0,a1,a2,g1,g2,h1,h2,p,q;
  g1=q2=p1=BigInteger.ZERO;
  h1=q1=p2=BigInteger.ONE;
  a0=a1=BigInteger.valueOf((int)Math.sqrt(1.0*n));
  BigInteger ans=a0.multiply(a0);
  if(ans.equals(BigInteger.valueOf(n))){
    System.out.println("No solution!");
    return ;
  }
  while(true){
    g2=a1.multiply(h1).substract(g1);
    h2=N.substract(g2.pow(2)).divide(h1);
    a2=g2.add(a0).divide(h2);
    p=a1.multiply(p2).add(p1);
    q=a1.multiply(q2).add(q1);
    if(p.pow(2).substract(N.multiply(q.pow(2))).compareTo(BigInteger.ONE)==0) break;
    g1=g2;h1=h2;a1=a2;
    p1=p2;p2=p;
    q1=q2;q2=q;
  }
  System.out.println(p+" "+q);
}
```

6.2 bit_set

```
void sub_set(int S){
  int sub=S;
  do{
    //對某集合的子集合的處理
    sub=(sub-1)&S;
  }while(sub!=S);
}
void k_sub_set(int k,int n){
  int comb=(1<<k)-1,S=1<<n;
  while(comb<S){
    //對大小為k的子集合的處理
    int x=comb&-comb,y=comb+x;
    comb=((comb&~y)/x>>1)|y;
  }
}
```

6.3 EXT_GCD

```
#include <bits/stdc++.h>
using namespace std;
typedef long long LL;
typedef pair < LL, LL> ii;

ii exd_gcd( LL a, LL b) {
    if (a % b == 0) return ii(0, 1);
    ii T = exd_gcd(b, a % b);
    return ii( T.second, T.first - a / b * T.second);
}
LL mod_inv(LL x) { // P is mod number, gcd(x,P) must be 1
    return (exd_gcd(x,P).first%P+P)%P;
}
```

6.4 FFT

```
const double PI = acos(-1);
using cd = complex<double>;
// Do FFT. invert=true to do iFFT.
// n MUST be power of 2.
void fft(cd a[], int n, bool invert) {
    for (int i = 1, j = 0; i < n; i++) {
        int bit = n >> 1;
        for (; j & bit; bit >>= 1)
            j ^= bit;
        j ^= bit;

        if (i < j)
            swap(a[i], a[j]);
    }

    for (int len = 2; len <= n; len <<= 1) {
        double ang = 2 * PI / len * (invert ? -1 : 1);
        cd wlen(cos(ang), sin(ang));
        for (int i = 0; i < n; i += len) {
            cd w(1);
            for (int j = 0; j < len / 2; j++) {
                cd u = a[i+j], v = a[i+j+len/2] * w;
                a[i+j] = u + v;
                a[i+j+len/2] = u - v;
                w *= wlen;
            }
        }
    }

    if (invert) {
        for (int i = 0; i < n; i++)
            a[i] /= n;
    }
}
```

6.5 find_real_root

```
// an*x^n + ... + a1x + a0 = 0;
int sign(double x){
  return x < -eps ? -1 : x > eps;
}

double get(const vector<double>&coef, double x){
  double e = 1, s = 0;
  for(auto i : coef) s += i*e, e *= x;
  return s;
}

double find(const vector<double>&coef, int n, double lo, double hi){
  double sign_lo, sign_hi;
  if( !(sign_lo = sign(get(coef,lo))) )
    return lo;
  if( !(sign_hi = sign(get(coef,hi))) )
    return hi;
  if(sign_lo * sign_hi > 0) return INF;
  for(int stp = 0; stp < 100 && hi - lo > eps; ++stp){
    double m = (lo+hi)/2.0;
    int sign_mid = sign(get(coef,m));
    if(!sign_mid) return m;
    if(sign_lo*sign_mid < 0) hi = m;
    else lo = m;
  }
  return (lo+hi)/2.0;
}

vector<double> cal(vector<double>coef, int n){
  vector<double>res;
  if(n == 1){
    if(sign(coef[1])) res.pb(-coef[0]/coef[1]);
    return res;
  }
  vector<double>dcoef(n);
  for(int i = 0; i < n; ++i) dcoef[i] = coef[i+1]*(i+1);
  vector<double>droot = cal(dcoef, n-1);
  droot.insert(droot.begin(), -INF);
  droot.pb(INF);
  for(int i = 0; i+1 < droot.size(); ++i){
    double tmp = find(coef, n, droot[i], droot[i+1]);
    if(tmp < INF) res.pb(tmp);
  }
  return res;
}

int main () {
  vector<double>ve;
  vector<double>ans = cal(ve, n);
  // Add EPS to answers when needed, to avoid -0.
}
```

6.6 FWT

```
vector<int> F_OR_T(vector<int> f, bool inverse){
  for(int i=0; (2<<i)<=f.size(); ++i)
    for(int j=0; j<f.size(); j+=2<<i)
      for(int k=0; k<(1<<i); ++k)
        f[j+k+(1<<i)] += f[j+k]*(inverse?-1:1);
  return f;
}
vector<int> rev(vector<int> A) {
  for(int i=0; i<A.size(); i+=2)
    swap(A[i],A[i^(A.size()-1)]);
  return A;
}
vector<int> F_AND_T(vector<int> f, bool inverse){
  return rev(F_OR_T(rev(f), inverse));
}
vector<int> F_XOR_T(vector<int> f, bool inverse){
  for(int i=0; (2<<i)<=f.size(); ++i)
    for(int j=0; j<f.size(); j+=2<<i)
      for(int k=0; k<(1<<i); ++k){
        int u=f[j+k], v=f[j+k+(1<<i)];
        f[j+k+(1<<i)] = u-v, f[j+k] = u+v;
      }
  if(inverse) for(auto &a:f) a/=f.size();
  return f;
}
```

6.7 gauss_elimination

```
typedef double Matrix[maxn][maxn];
void guauss_elimination(Matrix A, int n){
    int r;
    for(int i=0; i<n; i++){
        r = i;
        for(int j=i+1; j<n; j++)
            if(fabs(A[j][i]) > fabs(A[r][i])) r = j;
        if(r!=i) for(int j=0; j<=n; j++)
            swap(A[r][i], A[i][j]);

        for(int k=i+1; k<n; k++){
            double f = A[k][i]/A[i][i];
            for(int j=i; j<=n; j++) A[k][j] -= f*A[i][j];
        }
    }

    for(int i=n-1; i>=0; i--){
        for(int j=i+1; j<n; j++)
            A[i][n] -= A[j][n] * A[i][j];
        A[i][n] /= A[i][i];
    }
}
```

6.8 LL_mul

```
long long  mul(long long a, long long b) {
    long long ans = 0, step = a % MOD;
    while (b) {
        if (b & 1L) ans += step;
        if (ans >= MOD) ans %= MOD;
        step <<= 1L;
```

```
        if (step >= MOD) step %= MOD;
        b >>= 1L;
    }
    return ans % MOD;
}
```

6.9 Lucas

```
int mod_fact(int n,int &e){
  e=0;
  if(n==0)return 1;
  int res=mod_fact(n/P,e);
  e += n/P;
  if((n/P)%2==0)return res*fact[n%P]%P;
  return res*(P-fact[n%P])%P;
}
int Cmod(int n,int m){
  int a1,a2,a3,e1,e2,e3;
  a1=mod_fact(n,e1);
  a2=mod_fact(m,e2);
  a3=mod_fact(n-m,e3);
  if(e1>e2+e3)return 0;
  return a1*inv(a2*a3%P,P)%P;
}
```

6.10 Matrix

```
template<typename T>
struct Matrix{
  using rt = std::vector<T>;
  using mt = std::vector<rt>;
  using matrix = Matrix<T>;
  int r,c;
  mt m;
  Matrix(int r,int c):r(r),c(c),m(r,rt(c)){}
  rt& operator[](int i){return m[i];}
  matrix operator+(const matrix &a){
    matrix rev(r,c);
    for(int i=0;i<r;++i)
      for(int j=0;j<c;++j)
        rev[i][j]=m[i][j]+a.m[i][j];
    return rev;
  }
  matrix operator-(const matrix &a){
    matrix rev(r,c);
    for(int i=0;i<r;++i)
      for(int j=0;j<c;++j)
        rev[i][j]=m[i][j]-a.m[i][j];
    return rev;
  }
  matrix operator*(const matrix &a){
    matrix rev(r,a.c);
    matrix tmp(a.c,a.r);
    for(int i=0;i<a.r;++i)
      for(int j=0;j<a.c;++j)
        tmp[j][i]=a.m[i][j];
    for(int i=0;i<r;++i)
      for(int j=0;j<a.c;++j)
        for(int k=0;k<c;++k)
          rev.m[i][j]+=m[i][k]*tmp[j][k];
    return rev;
  }
  bool inverse(){
    Matrix t(r,r+c);
    for(int y=0;y<r;y++){
      t.m[y][c+y] = 1;
      for(int x=0;x<c;++x)
        t.m[y][x]=m[y][x];
    }
    if( !t.gas() )
      return false;
    for(int y=0;y<r;y++)
      for(int x=0;x<c;++x)
        m[y][x]=t.m[y][c+x]/t.m[y][y];
    return true;
  }
  T gas(){
    vector<T> lazy(r,1);
    bool sign=false;
    for(int i=0;i<r;++i){
      if( m[i][i]==0 ){
        int j=i+1;
        while(j<r&&!m[j][i])j++;
        if(j==r)continue;
        m[i].swap(m[j]);
        sign=!sign;
      }
      for(int j=0;j<r;++j){
        if(i==j)continue;
        lazy[j]=lazy[j]*m[i][i];
        T mx=m[j][i];
        for(int k=0;k<c;++k)
          m[j][k]=m[j][k]*m[i][i]-m[i][k]*mx;
      }
    }
    T det=sign?-1:1;
    for(int i=0;i<r;++i){
      det = det*m[i][i];
      det = det/lazy[i];
      for(auto &j:m[i])j/=lazy[i];
    }
    return det;
  }
};
```

6.11 Miller_Rabin

```
LL mod_mul(LL a, LL b, LL mod) {
//  return (__int128)a*b%mod;
  /* In case __int128 doesn't work(32* multi to avoid ovf) */
  LL x=0,y=a%mod;
    while(b > 0){
      if (b&1) x = (x+y)%mod;
      y = (y*2)%mod;
      b >>= 1;
    }
    return x%mod;
}
LL qpow(LL a, LL p, LL mod) {
  if (p<=0) return 1;
  LL temp = qpow(a,p/2,mod);
  temp = mod_mul(temp,temp,mod);
  if (p&1) return mod_mul(temp,a,mod);
  return temp;
}
bool MRtest(LL a, LL d, LL n) {
  LL x = qpow(a,d,n);
  if (x==1 || x==n-1) return true;
  while (d != n-1) {
    x = mod_mul(x,x,n);
    d *= 2;
    if (x==n-1) return true;
    if (x==1) return false;
  }
  return false;
}
bool is_prime(LL n) {
  if (n==2) return true;
  if (n<2 || n%2==0) return false;
  LL table[7] = {2, 325, 9375, 28178,
      450775, 9780504, 1795265022}, d=n-1;
  while (d%2 != 0) d>>=1; // n-1 = d * 2^r, d is odd.
  for (int i=0; i<7; i++) {
    LL a = table[i] % n;
    if (a==0 || a==1 || a==n-1) continue;
    if (!MRtest(a,d,n)) {
      return false;
    }
  }
  return true;
}
```

6.12 mod_log

```
const LL SQRT = 10005;
pair<LL, LL> bs[SQRT];
// O(sqrt(n)log(n))
LL baby_giant(LL a, LL b, LL m) {
  // Solve a^x = b (mod m) for x, gcd(a, m) = 1
  bs[0] = {1, 0};
  for (int i = 1; i < SQRT; i++) {
    bs[i] = {bs[i - 1].first * a % m, i};
  }

  LL cur = b, inv = mod_inv(bs[SQRT - 1].first * a % m, m); // inv of G.S.
  sort(bs, bs + SQRT);
  for (int i = 0; i < m; i += SQRT) {
    auto it = upper_bound(bs, bs + SQRT,
        make_pair(cur, (LL)-1));
    if (it != bs + SQRT && it->first == cur) {
      return i + it->second;
    }
    cur = cur * inv % m;
  }
  return -1; // no solution
}
```

6.13 NTT

```
const LL mod = 998244353;
const LL p_root = 3;
const LL root_pw = 1LL << 23;

// Do NTT under mod. invert=true to do iNTT.
// mod MUST be a prime, if mod=c*2^k+1, then
// p_root is any primitive root of mod
// root_pw=2^k, and n(size) MUST <= 2^k
// n MUST be power of 2.
// mod=2013265921, root_pw=1LL<<27, p_root=31

void ntt(LL a[], int n, bool invert) {
    LL root = qpow(p_root, (mod-1)/root_pw, mod);
  LL root_1 = mod_inv(root, mod);

  for (int i = 1, j = 0; i < n; i++) {
    LL bit = n >> 1;
    for (; j & bit; bit >>= 1)
      j ^= bit;
    j ^= bit;

    if (i < j)
      swap(a[i], a[j]);
  }

  for (int len = 2; len <= n; len <<= 1) {
    LL wlen = invert ? root_1 : root;
    for (int i = len; i < root_pw; i <<= 1)
      wlen = wlen * wlen % mod;

    for (int i = 0; i < n; i += len) {
      LL w = 1;
      for (int j = 0; j < len / 2; j++) {
        LL u = a[i+j], v = a[i+j+len/2] * w % mod;
        a[i+j] = u + v < mod ? u + v : u + v - mod;
        a[i+j+len/2] = u - v >= 0 ? u - v : u - v + mod;
        w = w * wlen % mod;
      }
    }
  }

  if (invert) {
    LL n_1 = mod_inv(n, mod);
    for (int i = 0; i < n; i++)
      a[i] = a[i] * n_1 % mod;
  }
}
```

6.14 pollard

```
LL pollard_rho(LL n, int c = 1) {
  // c is seed, rand can be replaced by 2, much faster
  LL x = rand() % n, y = x, d = 1;
```

```cpp
  while (d == 1) {
    x = mod_mul(x, x, n) + c;
    y = mod_mul(y, y, n) + c;
    y = mod_mul(y, y, n) + c;
    d = gcd(x - y >= 0 ? x - y : y - x, n);
  }
  if (d == n) return pollard_rho(n, c + 1);
  return d;
}

void factorize(LL n, vector<LL> &pf) {
  // N^(1/3) + LogN*(N^(1/4))
  // For all primes <= N^(1/3)
  for (LL p = 2; p <= (LL)1e6+5; p++) {
    while (n % p == 0) {
      pf.push_back(p);
      n /= p;
    }
  }
  // Use Miller-Rabin pls
  if (n == 1) return;
  else if (is_prime(n)) pf.push_back(n);
  else {
    LL d = pollard_rho(n);
    pf.push_back(d);
    pf.push_back(n / d);
  }
}
```

6.15 Simpson

```cpp
double simpson(double a,double b){
  double c=a+(b-a)/2;
  return (F(a)+4*F(c)+F(b))*(b-a)/6;
}
double asr(double a,double b,double eps,double A){
  double c=a+(b-a)/2;
  double L=simpson(a,c),R=simpson(c,b);
  if( abs(L+R-A)<15*eps )
    return L+R+(L+R-A)/15.0;
  return asr(a,c,eps/2,L)+asr(c,b,eps/2,R);
}
double asr(double a,double b,double eps){
  return asr(a,b,eps,simpson(a,b));
}
```

7 String

7.1 ACA

```cpp
static const int MAXL=200005,SIGMA=26; // MAXL: sum of length in dictionary
// link: suffix link, next: DFA link, n: # of nodes, tag: ID of str ends here
// next and link always exist, others exist iff values != -1.
// nocc: next occurrence, first node with tag != -1 along suffix link
int n, dep[MAXL], link[MAXL], next[MAXL][SIGMA];
int trie[MAXL][SIGMA], tag[MAXL], nocc[MAXL];

int new_node(int p) {
  // Add you init if recording more values.
  dep[n] = n == 0 ? 0 : dep[p] + 1;
  link[n] = tag[n] = nocc[n] = -1;
  for (int i = 0; i < SIGMA; i++) {
    next[n][i] = 0;
    trie[n][i] = -1;
  }
  return n++;
}
void build(vector<string> &dict) {
  // Some init should be written in new_node, O(N*SIGMA).
  n = 0;
  new_node(0);
  for (int i = 0; i < dict.size(); i++) {
    int v = 0;
    for (char ch : dict[i]) {
      int to = ch - 'a'; // CHANGE THIS !!
      if (trie[v][to] == -1) {
        trie[v][to] = next[v][to] = new_node(v);
      }
      v = trie[v][to];
    }
    tag[v] = i;
  }

  queue<int> Q;
  link[0] = 0;
  Q.push(0);
  while (!Q.empty()) {
    int v = Q.front(); Q.pop();
    for (int to = 0; to < SIGMA; to++) {
      if (trie[v][to] != -1) {
        int u = trie[v][to];
        link[u] = v == 0 ? 0 : next[link[v]][to];
        nocc[u] = tag[link[u]] != -1 ? link[u] : nocc[link[u]];
        for (int j = 0; j < SIGMA; j++) {
          if (trie[u][j] == -1) {
            next[u][j] = next[link[u]][j];
          }
        }
        Q.push(u);
      }
    }
  }
}
```

7.2 hash

```cpp
#define MAXN 1000000
#define mod 1073676287
/*mod 必須要是質數*/
typedef long long T;
char s[MAXN+5];
T h[MAXN+5];/*hash陣列*/
T h_base[MAXN+5];/*h_base[n]=(prime^n)%mod*/
void hash_init(int len,T prime){
  h_base[0]=1;
  for(int i=1;i<=len;++i){
    h[i]=(h[i-1]*prime+s[i-1])%mod;
    h_base[i]=(h_base[i-1]*prime)%mod;
  }
}
T get_hash(int l,int r){/*閉區間寫法，設編號為0 ~ len-1*/
  return (h[r+1]-(h[l]*h_base[r-l+1])%mod+mod)%mod;
}
```

7.3 KMP

```cpp
vector<int> lps; // longest prefix suffix, 0-based
int match(const string &text, const string &pat) {
  /* Init is included */
  lps.resize(pat.size());
  /* DP */
  lps[0]=0;
  for (int i=1; i<pat.size(); i++) {
    int len=lps[i - 1];
    while(len>0 && pat[len]!=pat[i]) len=lps[len - 1];
    lps[i] = pat[len]==pat[i] ? len+1 : 0;
  }
  /* Match */
  int i = 0, j = 0;
  while (i < text.size() && j < pat.size()) {
    if (text[i] == pat[j]) i++, j++;
    else if (j == 0) i++;
    else j = lps[j - 1];
  }
  if (j >= pat.size()) return i - j;
  return -1;
}
```

7.4 manacher

```cpp
vector<int> d1(n); // Max len of palindrome centerred at s[i]
for (int i = 0, l = 0, r = -1; i < n; i++) {
    int k = (i > r) ? 1 : min(d1[l + r - i], r - i + 1);
    while (0 <= i - k && i + k < n && s[i - k] == s[i + k]) {
      k++;
    }
    d1[i] = k--;
    if (i + k > r) {
      l = i - k;
      r = i + k;
    }
}
vector<int> d2(n); // Max len of centerred at "gap" before s[i]
for (int i = 0, l = 0, r = -1; i < n; i++) {
    int k = (i > r) ? 0 : min(d2[l + r - i + 1], r - i + 1);
    while (0 <= i - k - 1 && i + k < n && s[i - k - 1] == s[i + k]) {
      k++;
    }
    d2[i] = k--;
    if (i + k > r) {
      l = i - k - 1;
      r = i + k ;
    }
}
```

7.5 minimal_string_rotation

```cpp
int min_string_rotation(const string &s){
  int n=s.size(),i=0,j=1,k=0;
  while(i<n&&j<n&&k<n){
    int t=s[(i+k)%n]-s[(j+k)%n];
    ++k;
    if(t){
      if(t>0)i+=k;
      else j+=k;
      if(i==j)++j;
      k=0;
    }
  }
  return min(i,j);//最小循環表示法起始位置
}
```

7.6 reverseBWT

```cpp
const int MAXN = 305, MAXC = 'Z';
int ranks[MAXN], tots[MAXC], first[MAXC];
void rankBWT(const string &bw){
  memset(ranks,0,sizeof(int)*bw.size());
  memset(tots,0,sizeof(tots));
  for(size_t i=0;i<bw.size();++i)
    ranks[i] = tots[int(bw[i])]++;
}
void firstCol(){
  memset(first,0,sizeof(first));
  int totc = 0;
  for(int c='A';c<='Z';++c){
    if(!tots[c]) continue;
    first[c] = totc;
    totc += tots[c];
  }
}
string reverseBwt(string bw,int begin){
  rankBWT(bw), firstCol();
  int i = begin; //原字串最後一個元素的位置
  string res;
```

```cpp
  do{
    char c = bw[i];
    res = c + res;
    i = first[int(c)] + ranks[i];
  }while( i != begin );
  return res;
}
```

7.7 SA

```cpp
/* rank: inverse sa */
/* MAXL: Maximum length of string, lcp[i]: LCP(sa[i], sa[i-1]) */
string text;
int sa[MAXL], isa[MAXL], lcp[MAXL], cnt[MAXL+ALPHA];
void build(const vector<int> &_text) {
  text = _text + '\0'; // Must add this, must >= 0
  int sz = text.size(), lim = ALPHA; // Takes ALPHA time, note when #TC is large
  for (int i = 0; i < lim; i++) cnt[i] = 0;
  for (int i = 0; i < sz; i++) cnt[ isa[i] = text[i] ]++;
  for (int i = 1; i < lim; i++) cnt[i] += cnt[i - 1];
  for (int i = sz - 1; i >= 0; i--) sa[ --cnt[text[i]] ] = i;

  lim = max(sz, ALPHA);
  int *rk = isa, *nsa = lcp, *nrk = lcp;
  for (int len = 1; len < sz; len <<= 1) {
    int num = 0;
    for (int i = sz - len; i < sz; i++) nsa[num++] = i;
    for (int i = 0; i < sz; i++) if (sa[i] >= len) nsa[num++] = sa[i] - len;

    for (int i = 0; i < lim; i++) cnt[i] = 0;
    for (int i = 0; i < sz; i++) cnt[ rk[i] ]++;
    for (int i = 1; i < lim; i++) cnt[i] += cnt[i - 1];
    for (int i = sz-1; i >= 0; i--) sa[ --cnt[rk[nsa[i]]] ] = nsa[i];

    num = 0;
    nrk[sa[0]] = num++;
    for (int i = 1; i < sz; i++) {
      bool cond = rk[sa[i]] == rk[sa[i-1]] && sa[i] + len < sz;
      cond = cond && sa[i-1] + len < sz && rk[sa[i]+len] == rk[sa[i-1]+len];
      if (cond) nrk[sa[i]] = num - 1;
      else nrk[sa[i]] = num++;
    }

    if (num >= sz) break;
    lim = num;
    swap(rk, nrk);
    nsa = nrk;
  }
  for (int i=0; i<sz; i++) isa[sa[i]] = i;

  /* LCP */
  int len = 0;
  lcp[0] = 0; // Undefined
  for (int i=0; i<sz; i++) {
    if (isa[i] == 0) continue;
    len = max(0, len-1);
    int j = sa[isa[i]-1];
    while (text[i+len] == text[j+len]) len++;
    lcp[isa[i]] = len;
  }
}
```

7.8 Z

```cpp
void z_alg(char *s,int len,int *z){
  int l=0,r=0;
  z[0]=len;
  for(int i=1;i<len;++i){
    z[i]=i>r?0:(i-l+z[i-l]<z[l]?z[i-l]:r-i+1);
    while(i+z[i]<len&&s[i+z[i]]==s[z[i]])++z[i];
    if(i+z[i]-1>r)r=i+z[i]-1,l=i;
  }
}
```

8 Tarjan

8.1 dominator_tree

```cpp
struct dominator_tree{
  static const int MAXN=5005;
  int n;// 1-base
  vector<int> suc[MAXN],pre[MAXN];
  int fa[MAXN],dfn[MAXN],id[MAXN],Time;
  int semi[MAXN],idom[MAXN];
  int anc[MAXN],best[MAXN];//disjoint set
  vector<int> dom[MAXN];//dominator_tree
  void init(int _n){
    n=_n;
    for(int i=1;i<=n;++i)suc[i].clear(),pre[i].clear();
  }
  void add_edge(int u,int v){
    suc[u].push_back(v);
    pre[v].push_back(u);
  }
  void dfs(int u){
    dfn[u]=++Time,id[Time]=u;
    for(auto v:suc[u]){
      if(dfn[v])continue;
      dfs(v),fa[dfn[v]]=dfn[u];
    }
  }
  int find(int x){
    if(x==anc[x])return x;
    int y=find(anc[x]);
    if(semi[best[x]]>semi[best[anc[x]]])best[x]=best[anc[x]];
    return anc[x]=y;
  }
  void tarjan(int r){
    Time=0;
    for(int t=1;t<=n;++t){
      dfn[t]=idom[t]=0;//u=r或是u無法到達r時idom[id[u]]=0
      dom[t].clear();
      anc[t]=best[t]=semi[t]=t;
    }
    dfs(r);
    for(int y=Time;y>=2;--y){
      int x=fa[y],idy=id[y];
      for(auto z:pre[idy]){
        if(!(z=dfn[z]))continue;
        find(z);
        semi[y]=min(semi[y],semi[best[z]]);
      }
      dom[semi[y]].push_back(y);
      anc[y]=x;
      for(auto z:dom[x]){
        find(z);
        idom[z]=semi[best[z]]<x?best[z]:x;
      }
      dom[x].clear();
    }
    for(int u=2;u<=Time;++u){
      if(idom[u]!=semi[u])idom[u]=idom[idom[u]];
      dom[id[idom[u]]].push_back(id[u]);
    }
  }
}dom;
```

8.2 橋連通分量

```cpp
#define N 1005
struct edge{
  int u,v;
  bool is_bridge;
  edge(int u=0,int v=0):u(u),v(v),is_bridge(0){}
};
vector<edge> E;
vector<int> G[N];// 1-base
int low[N],vis[N],Time;
int bcc_id[N],bridge_cnt,bcc_cnt;// 1-base
int st[N],top;//BCC用
void add_edge(int u,int v){
  G[u].push_back(E.size());
  E.emplace_back(u,v);
  G[v].push_back(E.size());
  E.emplace_back(v,u);
}
void dfs(int u,int re=-1){//u當前點，re為u連接前一個點的邊
  int v;
  low[u]=vis[u]=++Time;
  st[top++]=u;
  for(int e:G[u]){
    v=E[e].v;
    if(!vis[v]){
      dfs(v,e^1);//e^1反向邊
      low[u]=min(low[u],low[v]);
      if(vis[u]<low[v]){
        E[e].is_bridge=E[e^1].is_bridge=1;
        ++bridge_cnt;
      }
    }else if(vis[v]<vis[u]&&e!=re)
      low[u]=min(low[u],vis[v]);
  }
  if(vis[u]==low[u]){//處理BCC
    ++bcc_cnt;// 1-base
    do bcc_id[v=st[--top]]=bcc_cnt;//每個點所在的BCC
    while(v!=u);
  }
}
void bcc_init(int n){
  Time=bcc_cnt=bridge_cnt=top=0;
  E.clear();
  for(int i=1;i<=n;++i){
    G[i].clear();
    vis[i]=bcc_id[i]=0;
  }
}
```

8.3 雙連通分量 & 割點

```cpp
#define N 1005
vector<int> G[N];// 1-base
vector<int> bcc[N];//存每塊雙連通分量的點
int low[N],vis[N],Time;
int bcc_id[N],bcc_cnt;// 1-base
bool is_cut[N];//是否為割點
int st[N],top;
void dfs(int u,int pa=-1){//u當前點，pa父親
  int t, child=0;
  low[u]=vis[u]=++Time;
  st[top++]=u;
  for(int v:G[u]){
    if(!vis[v]){
      dfs(v,u),++child;
      low[u]=min(low[u],low[v]);
      if(vis[u]<=low[v]){
        is_cut[u]=1;
        bcc[++bcc_cnt].clear();
        do{
          bcc_id[t=st[--top]]=bcc_cnt;
          bcc[bcc_cnt].push_back(t);
        }while(t!=v);
        bcc_id[u]=bcc_cnt;
        bcc[bcc_cnt].push_back(u);
      }
    }else if(vis[v]<vis[u]&&v!=pa)//反向邊
      low[u] = min(low[u],vis[v]);
  }//u是dfs樹的根要特判
  if(pa==-1&&child<2)is_cut[u]=0;
```

```cpp
}
void bcc_init(int n){
  Time=bcc_cnt=top=0;
  for(int i=1;i<=n;++i){
    G[i].clear();
    is_cut[i]=vis[i]=bcc_id[i]=0;
  }
}
```

9 Tree

9.1 HLD

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

// Preprocess info, setup in dfs1
int heavy[MAXN+5], pa_w[MAXN+5], sz[MAXN+5];
int pa[MAXN+5], dep[MAXN+5], recorder[MAXN+5]; // Which node record edge i.

// HLD info, setup in build, 1-based
// pos: position of node i in seg tree.
// head: For NODE i, points to head of the chain.
int chain_no, border, pos[MAXN+5], head[MAXN+5];

void dfs1(int v, int p) {
    pa[v] = p;
    sz[v] = 1;
    dep[v] = dep[p] + 1;
    heavy[v] = -1;

    for (const Edge &e : G[v]) {
        if (e.to == p) continue;
        dfs1(e.to, v);
        pa_w[e.to] = e.w;
        recorder[e.id] = e.to;
        sz[v] += sz[e.to];
        if (heavy[v] == -1 || sz[e.to] > sz[heavy[v]]) {
            heavy[v] = e.to;
        }
    }
}

void build(int v, int chain_head) {
    pos[v] = ++border;
    head[v] = chain_head;
    tree.update(pos[v], pa_w[v], 1, N, 1);

    if (heavy[v] != -1) build(heavy[v], chain_head);
    for (const Edge &e : G[v]) {
        if (e.to == pa[v] || e.to == heavy[v]) continue;
        build(e.to, e.to);
    }
}

void init_HLD() {
    /* Only init used data, be careful. */
    /* Does not init G!!!!! */
    border = dep[1] = pa_w[1] = 0;
    dfs1(1, 1);
    build(1, 1);
}

int query_up(int a, int b) {
    int ans = 0;
    while (head[a] != head[b]) {
      if (dep[head[a]] < dep[head[b]]) swap(a, b);
      ans = max(ans, tree.query(pos[head[a]], pos[a], 1, N, 1));
      a = pa[head[a]];
    }

    if (a == b) return ans;
    if (dep[a] < dep[b]) swap(a, b);
    // Query range is pos[b] if value is on node.
    ans = max(ans, tree.query(pos[b] + 1, pos[a], 1, N, 1));
    return ans;
}
```

9.2 treeDC

```cpp
int get_size(int v, int p) {
  sz[v] = 1;
  for (int to : G[v]) {
    if (to != p && !vis[to]) {
      get_size(to, v);
      sz[v] += sz[to];
    }
  }
  return sz[v];
}

void find_cent(int v, int p, int &cent, int S) {
  int big = S - sz[v];
  for (int to : G[v]) {
    if (!vis[to] && to != p) {
      big = max(big, sz[to]);
      find_cent(to, v, cent, S);
    }
  }
  maxs[v] = big;
  if (cent == -1 || big < maxs[cent]) {
    cent = v;
  }
}

void dfs(int v, int p, int d, vector<int> &sub) {
  dep[v] = d;
  sub.push_back(v);
  for (int to : G[v]) {
    if (!vis[to] && to != p) {
      dfs(to, v, d + 1, sub);
    }
  }
}

LL solve(int v, int l, int r) {
  // # unordered (x, y), l <= dist(x, y) <= r, in tree of v.
  int S = get_size(v, v), root = -1;
  find_cent(v, v, root, S);
  vis[root] = 1;

  LL res = 0;
  tree.add(0, 1); // ***** tree MUST be 0-based RSQ
  vector<int> all;
  for (int to : G[root]) {
    if (!vis[to]) {
      vector<int> sub;
      dfs(to, root, 1, sub);
      for (int u : sub) {
        all.push_back(u);
        if (r - dep[u] >= 0) {
          res += tree.get(r - dep[u]);
        }
        if (l - 1 - dep[u] >= 0) {
          res -= tree.get(l - 1 - dep[u]);
        }
      }
      for (int u : sub) {
        tree.add(dep[u], 1);
      }
    }
  }

  tree.add(0, -1);
  for (int u : all) {
    tree.add(dep[u], -1);
  }
  all.clear();
  all.shrink_to_fit();

  for (int to : G[root]) {
    if (!vis[to]) {
      res += solve(to, l, r);
    }
  }

  return res;
}
```

10 others

10.1 pbds

```cpp
#include <bits/stdc++.h>
#include <ext/pb_ds/assoc_container.hpp>
#include <ext/pb_ds/tree_policy.hpp>
using namespace std;
namespace __gnu_pbds{
typedef tree<
int,
null_type,
less<int>,
rb_tree_tag,
tree_order_statistics_node_update>
ordered_set;
}

int main() {
  __gnu_pbds::ordered_set S;
  S.insert(5);
  S.insert(7);
  S.insert(10);
  cout << S.order_of_key(4) << '\n'; // How many smaller
  cout << S.order_of_key(5) << '\n';
  cout << S.order_of_key(6) << '\n';
  cout << *S.find_by_order(0) << '\n';
  cout << *S.find_by_order(2) << '\n';
  return 0;
}
```

10.2 vimrc

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

11 zformula

11.1 formula

11.1.1 formula.txt

1. 若多項式 f(x) 有有理根 P/Q(PQ 互質), 則 P 必為常數項 a0 之因數, Q 必為領導係數 an 之因數
2. 滿足 ceil(n/i)=k 之最大 i:
   (a) INF, if k=1
   (b) n/(k-1)-1, else if k-1 整除 n
   (c) n/(k-1), else
3. 滿足 floor(n/i)=k 之最大 i: floor(n/k)
4. 尤拉函數: phi(n)=n 乘上所有 (1-1/p)，對 n 之所有質因數 p
5. 尤拉定理: $a^phi(n) = 1(modn)$, a,n 互質
6. 尤拉降冪: $a^b = a^{bmodphi(n)+phi(n)}(modn), b > phi(n)$, 不必互質
7. 次方同餘定理: $a^kmodp = (amodp)^{(kmodp-1)}$ p 是質數
8. Modulo inverse: inv[i] = - floor(p / i) * inv[p mod i] (mod p)
9. 中國剩餘定理: x=Ai(mod mi), mi 互質, Mi=所有 m 的乘積/ mi, Ti=Mi^-1(mod mi), 則 x=sigma(Mi*Ti*Ai)(mod M)
10. 枚舉擴展歐幾里得之解: 若 x0,y0 為 a*x+b*y = k 之一組解，則 x=x0+t*b/gcd(a,b), y=y0+t*a/gcd(a,b) 亦為解，t 為整數

11. Sigma{i : gcd(i,n) = 1 and i in [1, n]} = n*phi(n)/2 for n > 1
12. $Sigma\{i * r^i : iin[1,n]\} = (n * r^(n+1) - r * (r^n - 1)/(r-1))/(r-1)$
13. 投擲正面機率 p 之硬幣 n 次，正面偶數次機率: $0.5 + 0.5 * (1-2p)^n$
14. 分式拆分: (a - b)/(ab) = 1/b - 1/a
15. 最大獨立集: 點的集合，其內點不相鄰
16. 最小點覆蓋: 點的集合，所有邊都被覆蓋
17. 最大匹配: 邊的集合，其內邊不共用點
18. 最小邊覆蓋: 邊的集合，所有點都被覆蓋
19. 最大獨立集 + 最小點覆蓋 =V(數值)
20. 最大匹配 + 最小邊覆蓋 =V(數值)
21. 最大匹配 = 最大流 (directed, 二分圖)
22. 最大匹配 = 最小點覆蓋 (二分圖)
23. 最小點覆蓋 + 最小邊覆蓋 =V(數值，二分圖)
24. 二分圖帶權最小點覆蓋 = 對左邊的點 v 連 cap(src,v)=w(v) 之邊，右邊每個 v 連 cap(v,tgt)=w(v) 之邊，每條邊 (u,v) 連 cap(u,v)=INF，皆有向，最大流即為所求。
25. 一般圖帶權最小邊覆蓋 = (將原圖每個 w(u,v) 改為 w'(u,v)=c(u)+c(v)-w(u,v))，所求為新圖之最大權匹配 +sigma{c(v)}，c(v) 為點 v 連到的最小 edge 權重。
26. 一矩陣 A 所有 eigen value 之合 = 對角線合
27. 一矩陣 A 所有 eigen value 之積 =det(A)
28. 三角形 ABC, 對邊長 abc:
29. area=sqrt(s(s-a)(s-b)(s-b)), s= 周長/2
30. a/sinA = b/sinB = c/sinC = 2R, R 為外接圓半徑
31. 內接圓半徑 =2*area/(a+b+c)
32. 外接圓半徑 =abc/4*area
33. 球缺體積, h 為高, 且 h <= R: $PI * h^2 * (R - h/3)$
34. 枚舉 submask: for (int s=m; s; s=(s-1)&m) // Take care of ZERO after loop
35. 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，$E \le 3V - 6$
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. 不相交環覆蓋: 每個 v 拆 vin, vout, 存在 iff. 二分完美匹配存在, 最小邊權環覆蓋 = 最小完美匹配
6. vertex disjoint DAG path cover (蓋住所有點): 每個 v 拆 vin, vout, 原圖 |V|-| 最大二分匹配 | 即為所求
7. 可相交 DAG path cover: 每個 v 對他能走到的所有點 u 連一條邊, 轉為 disjoint. (轉換後所有中途點毋須存在)
8. max anti-chain over partial order (最大 subset 任兩人不可比較): 建出 partial order 的 transitive closure, disjoint DAG path cover 即為所求.
9. 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) ans=$\sum_{W_v>0} W_v - flow(S,T)$
10. 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：
   $l = 0, r = U, eps = 1/n^2$
   if($(U \times |V| - flow(S,T))/2 > 0$) $l = mid$
   else $r = mid$
   (h) ans=$min\_cut(S,T)$
   (i) $|E| = 0$ 要特殊判斷
11. 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排序
   (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

$x_i = \{0,1\}$，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

```
binary_search(){
  while(r-l>eps){
    g=(l+r)/2;
    for(i:所有元素)D[i]=B[i]-g*C[i];//D(i,g)
    找出一組合法x[i]使f(g)最大;
    if(f(g)>0) l=g;
    else r=g;
  }
  Ans = r;
}
Dinkelbach(){
  g=任意狀態(通常設為0);
  do{
    Ans=g;
    for(i:所有元素)D[i]=B[i]-g*C[i];//D(i,g)
    找出一組合法x[i]使f(g)最大;
    p=0,q=0;
    for(i:所有元素)
      if(x[i])p+=B[i],q+=C[i];
    g=p/q;//更新解，注意q=0的情況
  }while(abs(Ans-g)>EPS);
  return Ans;
}
```

11.1.6 學長公式

1. $\sum_{d|n} \phi(n) = n$
2. $g(n) = \sum_{d|n} f(d) => f(n) = \sum_{d|n} \mu(d) \times g(n/d)$
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. $\sum_{i=1}^{n} \sum_{j=1}^{m}$ 互質數量 $= \sum \mu(d) \left\lfloor \frac{n}{d} \right\rfloor \left\lfloor \frac{m}{d} \right\rfloor$
4. Useful strate: Enumerate (i,j) having common factor d then inclusion exclusion. (for all pair gcd sum / number of coprime pairs)
5. Useful strate: For each i, first pick a smaller j for a coprime pair (i,j), then used to form pairs with larger gcd. (for all pair lcm / gcd sum)

11.1.8 排組公式

1. k 卡特蘭 $\frac{C_n^{kn}}{n(k-1)+1}$，$C_m^n = \frac{n!}{m!(n-m)!}$
2. $H(n,m) \cong x_1 + x_2 \ldots + x_n = k, num = C_k^{n+k-1}$
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) $B_{p+n} \equiv B_n + B_{n+1} modp$, p is prime
   (e) $B_{p^m+n} \equiv mB_n + B_{n+1} modp$, 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) $D_n = n!(1 - \frac{1}{1!} + \frac{1}{2!} - \frac{1}{3!} \ldots + (-1)^n \frac{1}{n!})$
   (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) $\sum_k \binom{l}{m+k}\binom{s+k}{n}(-1)^k = (-1)^{l+m}\binom{s-m}{n-l}$
   (d) $\sum_{k\le l} \binom{l-k}{m}\binom{s}{k-n}(-1)^k = (-1)^{l+m}\binom{s-m-1}{l-n-m}$
   (e) $\sum_{0\le k\le l} \binom{l-k}{m}\binom{q+k}{n} = \binom{l+q+1}{m+n+1}$
   (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) $\sum_{k\le n} \binom{r+k}{k} = \binom{r+n+1}{n}$
   (i) $\sum_{0\le k\le n} \binom{k}{m} = \binom{n+1}{m+1}$
   (j) $\sum_{k\le m} \binom{m+r}{k} x^k y^k = \sum_{k\le m} \binom{-r}{k}(-x)^k (x+y)^{m-k}$

11.1.9 冪次, 冪次和

1. $a^b \% P = a^{b\%\varphi(p)+\varphi(p)}, b \ge \varphi(p)$
2. $1^3 + 2^3 + 3^3 + \ldots + n^3 = \frac{n^4}{4} + \frac{n^3}{2} + \frac{n^2}{4}$
3. $1^4 + 2^4 + 3^4 + \ldots + n^4 = \frac{n^5}{5} + \frac{n^4}{2} + \frac{n^3}{3} - \frac{n}{30}$
4. $1^5 + 2^5 + 3^5 + \ldots + n^5 = \frac{n^6}{6} + \frac{n^5}{2} + \frac{5n^4}{12} - \frac{n^2}{12}$
5. $0^k + 1^k + 2^k + \ldots + n^k = P(k), P(k) = \frac{(n+1)^{k+1} - \sum_{i=0}^{k-1} C_i^{k+1} P(i)}{k+1}, P(0) = n+1$
6. $\sum_{k=0}^{m-1} k^n = \frac{1}{n+1} \sum_{k=0}^{n} C_k^{n+1} B_k m^{n+1-k}$
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 3×3^4，120(角) 有 8×3^2，180(邊) 有 6×3^3，全部 $\frac{1}{24}\left(3^6+6\times 3^3+3\times 3^4+8\times 3^2+6\times 3^3\right) = 57$

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 (for n labeled vertices)
   (a) 完全圖 $n^n - 2$
   (b) 完全二分圖 $K_{n,m}$: $m^{n-1}\times n^{m-1}$
   (c) 一般圖 (Kirchhoff's theorem)$M[i][i] = degree(V_i)$,$M[i][j] = -1$,if have $E(i,j)$,0 if no edge. delete any one row and col in A, $ans = det(A)$

11.1.12 Horrible bugs

1. int 開成 bool 導致計算出錯或其他型別開錯導致 cin 出錯
2. cmp 寫成非嚴格偏序
3. 該開 multiset 不小心開成 set
4. 你以為 sort 只要排一維, 其實兩維都要排
5. 分成多個地方 output, 忘記設定 precision 或沒 return
6. 把 N 向上補成 2 的倍數或改動常數, 但是 N 會用在別的地方

l, r ，題目沒有說 l <= r 之類的

7. 填入無限大或負數之類的湊成整數倍, 結果被拿來當 array id
8. Any unsigned BUG?
9. 再把題目看一次
10. 感覺都沒錯, 生一些有相同物的 case 或邊界條件

11.2 java

11.2.1 文件操作

```java
import java.io.*;
import java.util.*;
import java.math.*;
import java.text.*;

public class Main{

  public static void main(String args[]){
      throws FileNotFoundException,
      IOException
    Scanner sc = new Scanner(new FileReader(
        "a.in"));
    PrintWriter pw = new PrintWriter(new
        FileWriter("a.out"));
    int n,m;
    n=sc.nextInt();//读入下一个INT
    m=sc.nextInt();

    for(ci=1; ci<=c; ++ci){
      pw.println("Case #"+ci+": easy for
          output");
    }

    pw.close();//关闭流并释放，这个很重要，
        否则是没有输出的
    sc.close();//关闭流并释放
  }
}
```

11.2.2 优先队列

```java
PriorityQueue queue = new PriorityQueue( 1,
    new Comparator(){
  public int compare( Point a, Point b ){
  if( a.x < b.x || a.x == b.x && a.y < b.y )
    return -1;
  else if( a.x == b.x && a.y == b.y )
    return 0;
  else return 1;
  }
});
```

11.2.3 Map

```java
Map map = new HashMap();
map.put("sa","dd");
String str = map.get("sa").toString;

for(Object obj : map.keySet()){
  Object value = map.get(obj );
}
```

11.2.4 sort

```java
static class cmp implements Comparator{
  public int compare(Object o1,Object o2){
  BigInteger b1=(BigInteger)o1;
  BigInteger b2=(BigInteger)o2;
  return b1.compareTo(b2);
  }
}
public static void main(String[] args)
    throws IOException{
  Scanner cin = new Scanner(System.in);
  int n;
  n=cin.nextInt();
  BigInteger[] seg = new BigInteger[n];
  for (int i=0;i<n;i++)
  seg[i]=cin.nextBigInteger();
  Arrays.sort(seg,new cmp());
}
```

11.2.5 utility

```java
BigInteger x,y,z; z=x.divide(y); // multiply
    , subtract, add, mod, z=x.negate()
Arrays.sort(arr, 0, size);
BigInteger dp[][] = new BigInteger[n][n];
Math.min(x, y) // Math.max
Integer.toString(5);
x=BigInteger.valueOf(5);
while (fin.hasNext()) x = fin.nextBigInteger
    ();
```

ACM ICPC Team Reference - PolarSheep

Contents