1 Computational Geometry

1.1 Geometry

```cpp
const double PI=atan2(0.0,-1.0);
template<typename T>
struct point{
  T x,y;
  point(){}
  point(const T&x,const T&y):x(x),y(y){}
  point operator+(const point &b)const{
    return point(x+b.x,y+b.y); }
  point operator-(const point &b)const{
    return point(x-b.x,y-b.y); }
  point operator*(const T &b)const{
    return point(x*b,y*b); }
  point operator/(const T &b)const{
    return point(x/b,y/b); }
  bool operator==(const point &b)const{
    return x==b.x&&y==b.y; }
  T dot(const point &b)const{
    return x*b.x+y*b.y; }
  T cross(const point &b)const{
    return x*b.y-y*b.x; }
  point normal()const{//求法向量
    return point(-y,x); }
  T abs2()const{//向量長度的平方
    return dot(*this); }
  T rad(const point &b)const{//兩向量的弧度
return fabs(atan2(fabs(cross(b)),dot(b))); }
  T getA()const{//對x軸的弧度
    T A=atan2(y,x);//超過180度會變負的
    if(A<=-PI/2)A+=PI*2;
    return A;
  }
};
template<typename T>
struct line{
  line(){}
  point<T> p1,p2;
  T a,b,c;//ax+by+c=0
  line(const point<T>&x,const point<T>&y):p1
      (x),p2(y){}
  void pton(){//轉成一般式
    a=p1.y-p2.y;
    b=p2.x-p1.x;
    c=-a*p1.x-b*p1.y;
  }
  T ori(const point<T> &p)const{//點和有向直
      線的關係，>0左邊、=0在線上<0右邊
    return (p2-p1).cross(p-p1);
  }
  T btw(const point<T> &p)const{//點投影落在
      線段上<=0
    return (p1-p).dot(p2-p);
  }
  bool point_on_segment(const point<T>&p)
      const{//點是否在線段上
    return ori(p)==0&&btw(p)<=0;
  }
  T dis2(const point<T> &p,bool is_segment
      =0)const{//點跟直線/線段的距離平方
    point<T> v=p2-p1,v1=p-p1;
    if(is_segment){
      point<T> v2=p-p2;
      if(v.dot(v1)<=0)return v1.abs2();
      if(v.dot(v2)>=0)return v2.abs2();
    }
    T tmp=v.cross(v1);
    return tmp*tmp/v.abs2();
  }
  T seg_dis2(const line<T> &l)const{//兩線段
      距離平方
    return min({dis2(l.p1,1),dis2(l.p2,1),l.
        dis2(p1,1),l.dis2(p2,1)});
  }
  point<T> projection(const point<T> &p)
      const{//點對直線的投影
    point<T> n=(p2-p1).normal();
    return p-n*(p-p1).dot(n)/n.abs2();
  }
  point<T> mirror(const point<T> &p)const{
    //點對直線的鏡射，要先呼叫pton轉成一般式
    point<T> R;
    T d=a*a+b*b;
    R.x=(b*b*p.x-a*a*p.x-2*a*b*p.y-2*a*c)/d;
    R.y=(a*a*p.y-b*b*p.y-2*a*b*p.x-2*b*c)/d;
    return R;
  }
  bool equal(const line &l)const{//直線相等
    return ori(l.p1)==0&&ori(l.p2)==0;
  }
  bool parallel(const line &l)const{
    return (p1-p2).cross(l.p1-l.p2)==0;
  }
  bool cross_seg(const line &l)const{
    return (p2-p1).cross(l.p1-p1)*(p2-p1).
        cross(l.p2-p1)<=0;//直線是否交線段
  }
  int line_intersect(const line &l)const{//
      直線相交情況，-1無限多點、1交於一點、0
      不相交
    return parallel(l)?(ori(l.p1)==0?-1:0)
        :1;
  }
  int seg_intersect(const line &l)const{
    T c1=ori(l.p1), c2=ori(l.p2);
    T c3=l.ori(p1), c4=l.ori(p2);
    if(c1==0&&c2==0){//共線
      bool b1=btw(l.p1)>=0,b2=btw(l.p2)>=0;
      T a3=l.btw(p1),a4=l.btw(p2);
      if(b1&&b2&&a3==0&&a4>=0) return 2;
      if(b1&&b2&&a3>=0&&a4==0) return 3;
      if(b1&&b2&&a3>=0&&a4>=0) return 0;
      return -1;//無限交點
    }else if(c1*c2<=0&&c3*c4<=0)return 1;
    return 0;//不相交
  }
  point<T> line_intersection(const line &l)
      const{/*直線交點*/
    point<T> a=p2-p1,b=l.p2-l.p1,s=l.p1-p1;
    //if(a.cross(b)==0)return INF;
    return p1+a*(s.cross(b)/a.cross(b));
  }
  point<T> seg_intersection(const line &l)
      const{//線段交點
    int res=seg_intersect(l);
    if(res<=0) assert(0);
    if(res==2) return p1;
    if(res==3) return p2;
    return line_intersection(l);
  }
};
template<typename T>
struct polygon{
  polygon(){}
  vector<point<T> > p;//逆時針順序
  T area()const{//面積
    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{//重心
    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{//點是否
      在簡單多邊形內，是的話回傳1、在邊上回
      傳-1、否則回傳0
    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){//點是否在凸多邊形內，是的話
        回傳1、在邊上回傳-1、否則回傳0
      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){
        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{//凸包邊對x軸的夾角
    vector<T>res;//一定是遞增的
    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{//凸包
      對直線切割，得到直線l左側的凸包
    polygon ans;
    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 monotone_chain_cmp(const point
      <T>& a,const point<T>& b){//凸包排序函
      數
    return (a.x<b.x)||(a.x==b.x&&a.y<b.y);
  }
  void monotone_chain(vector<point<T> > &s){
      //凸包
    sort(s.begin(),s.end(),
        monotone_chain_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 diam(){//直徑
    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(){//最小覆蓋矩形
```

```
    int n=p.size(),t=1,r=1,l;
    if(n<3)return 0;//也可以做最小周長矩形
    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){//凸包最近距離平方
    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){//半平面交
    sort(s.begin(),s.end(),angle_cmp);//線段左側為該線段半平面
    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{//重心
    return (a+b+c)/3;
  }
  point<T> circumcenter()const{//外心
    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{//內心
    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{//垂心
    return barycenter()*3-circumcenter()*2;
  }
};
template<typename T>
struct point3D{
  T x,y,z;
  point3D(){}
  point3D(const T&x,const T&y,const T&z):x(x),y(y),z(z){}
  point3D operator+(const point3D &b)const{
    return point3D(x+b.x,y+b.y,z+b.z);}
  point3D operator-(const point3D &b)const{
    return point3D(x-b.x,y-b.y,z-b.z);}
  point3D operator*(const T &b)const{
    return point3D(x*b,y*b,z*b);}
  point3D operator/(const T &b)const{
    return point3D(x/b,y/b,z/b);}
  bool operator==(const point3D &b)const{
    return x==b.x&&y==b.y&&z==b.z;}
  T dot(const point3D &b)const{
    return x*b.x+y*b.y+z*b.z;}
  point3D cross(const point3D &b)const{
    return point3D(y*b.z-z*b.y,z*b.x-x*b.z,x*b.y-y*b.x);}
  T abs2()const{//向量長度的平方
    return dot(*this);}
  T area2(const point3D &b)const{//和b、原點圍成面積的平方
    return cross(b).abs2()/4;}
};
template<typename T>
struct line3D{
  point3D<T> p1,p2;
  line3D(){}
  line3D(const point3D<T> &p1,const point3D<T> &p2):p1(p1),p2(p2){}
  T dis2(const point3D<T> &p,bool is_segment=0)const{//點跟直線/線段的距離平方
    point3D<T> v=p2-p1,v1=p-p1;
    if(is_segment){
      point3D<T> v2=p-p2;
      if(v.dot(v1)<=0)return v1.abs2();
      if(v.dot(v2)>=0)return v2.abs2();
    }
    point3D<T> tmp=v.cross(v1);
    return tmp.abs2()/v.abs2();
  }
  pair<point3D<T>,point3D<T> > closest_pair(const line3D<T> &l)const{
    point3D<T> v1=(p1-p2),v2=(l.p1-l.p2);
    point3D<T> N=v1.cross(v2),ab(p1-l.p1);
    //if(N.abs2()==0)return NULL;平行或重合
    T tmp=N.dot(ab),ans=tmp*tmp/N.abs2();//最近點對距離
    point3D<T> d1=p2-p1,d2=l.p2-l.p1,D=d1.cross(d2),G=l.p1-p1;
    T t1=(G.cross(d2)).dot(D)/D.abs2();
    T t2=(G.cross(d1)).dot(D)/D.abs2();
    return make_pair(p1+d1*t1,l.p1+d2*t2);
  }
  bool same_side(const point3D<T> &a,const point3D<T> &b)const{
    return (p2-p1).cross(a-p1).dot((p2-p1).cross(b-p1))>0;
  }
};
template<typename T>
struct plane{
  point3D<T> p0,n;//平面上的點和法向量
  plane(){}
  plane(const point3D<T> &p0,const point3D<T> &n):p0(p0),n(n){}
  T dis2(const point3D<T> &p)const{//點到平面距離的平方
    T tmp=(p-p0).dot(n);
    return tmp*tmp/n.abs2();
  }
  point3D<T> projection(const point3D<T> &p)const{
    return p-n*(p-p0).dot(n)/n.abs2();
  }
  point3D<T> line_intersection(const line3D<T> &l)const{
    T tmp=n.dot(l.p2-l.p1);//等於0表示平行或重合該平面
    return l.p1+(l.p2-l.p1)*(n.dot(p0-l.p1)/tmp);
  }
  line3D<T> plane_intersection(const plane &pl)const{
    point3D<T> e=n.cross(pl.n),v=n.cross(e);
    T tmp=pl.n.dot(v);//等於0表示平行或重合該平面
    point3D<T> q=p0+(v*(pl.n.dot(pl.p0-p0))/tmp);
    return line3D<T>(q,q+e);
  }
};
template<typename T>
struct triangle3D{
  point3D<T> a,b,c;
  triangle3D(){}
  triangle3D(const point3D<T> &a,const point3D<T> &b,const point3D<T> &c):a(a),b(b),c(c){}
  bool point_in(const point3D<T> &p)const{//點在該平面上的投影在三角形中
    return line3D<T>(b,c).same_side(p,a)&&
        line3D<T>(a,c).same_side(p,b)&&
        line3D<T>(a,b).same_side(p,c);
  }
};
template<typename T>
struct tetrahedron{//四面體
  point3D<T> a,b,c,d;
  tetrahedron(){}
  tetrahedron(const point3D<T> &a,const point3D<T> &b,const point3D<T> &c,
      const point3D<T> &d):a(a),b(b),c(c),d(d){}
  T volume6()const{//體積的六倍
    return (d-a).dot((b-a).cross(c-a));
  }
  point3D<T> centroid()const{
    return (a+b+c+d)/4;
  }
  bool point_in(const point3D<T> &p)const{
    return triangle3D<T>(a,b,c).point_in(p)
        &&triangle3D<T>(c,d,a).point_in(p);
  }
};
template<typename T>
struct convexhull3D{
  static const int MAXN=1005;
  struct face{
    int a,b,c;
    face(int a,int b,int c):a(a),b(b),c(c){}
  };
  vector<point3D<T>> pt;
  vector<face> ans;
  int fid[MAXN][MAXN];
  void build(){
    int n=pt.size();
    ans.clear();
    memset(fid,0,sizeof(fid));
    ans.emplace_back(0,1,2);//注意不能共線
    ans.emplace_back(2,1,0);
    int ftop = 0;
    for(int i=3, ftop=1; i<n; ++i,++ftop){
      vector<face> next;
      for(auto &f:ans){
        T d=(pt[i]-pt[f.a]).dot((pt[f.b]-pt[f.a]).cross(pt[f.c]-pt[f.a]));
        if(d<=0) next.push_back(f);
        int ff=0;
        if(d>0) ff=ftop;
        else if(d<0) ff=-ftop;
```

```
        fid[f.a][f.b]=fid[f.b][f.c]=fid[f.c][f.a]=ff;
      }
      for(auto &f:ans){
        if(fid[f.a][f.b]>0 && fid[f.a][f.b]!=fid[f.b][f.a])
          next.emplace_back(f.a,f.b,i);
        if(fid[f.b][f.c]>0 && fid[f.b][f.c]!=fid[f.c][f.b])
          next.emplace_back(f.b,f.c,i);
        if(fid[f.c][f.a]>0 && fid[f.c][f.a]!=fid[f.a][f.c])
          next.emplace_back(f.c,f.a,i);
      }
      ans=next;
    }
  }
  point3D<T> centroid()const{
    point3D<T> res(0,0,0);
    T vol=0;
    for(auto &f:ans){
      T tmp=pt[f.a].dot(pt[f.b].cross(pt[f.c]));
      res=res+(pt[f.a]+pt[f.b]+pt[f.c])*tmp;
      vol+=tmp;
    }
    return res/(vol*4);
  }
};
```

1.2 SmallestCircle

```
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 = 圓心,半徑平方
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();
      }
    }
  }
}
```

1.3 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());
}
```

2 Data Structure

2.1 segment tree

```
template <typename T>
class SegmentTree {
 private:
  std::vector<T> tree, lazy;
  std::vector<bool> updated;
  int n;

  void build(int node, int start, int end, const std::vector<T>& arr) {
    if (start == end) {
      tree[node] = arr[start];
    } else {
      int mid = (start + end) / 2;
      build(2 * node, start, mid, arr);
      build(2 * node + 1, mid + 1, end, arr);
      tree[node] = tree[2 * node] + tree[2 * node + 1];
    }
  }

  void updateRange(int node, int start, int end, int l, int r, T val) {
    if (updated[node]) {
      tree[node] = lazy[node] * (end - start + 1);
      if (start != end) {
        lazy[node * 2] = lazy[node];
        lazy[node * 2 + 1] = lazy[node];
        updated[node * 2] = updated[node];
        updated[node * 2 + 1] = updated[node];
      }
      updated[node] = false;
    }
    if (start > end or start > r or end < l)
      return;
    if (start >= l and end <= r) {
      tree[node] = val * (end - start + 1);
      if (start != end) {
        lazy[node * 2] = val;
        lazy[node * 2 + 1] = val;
        updated[node * 2] = true;
        updated[node * 2 + 1] = true;
      }
      return;
    }
    int mid = (start + end) / 2;
    updateRange(node * 2, start, mid, l, r, val);
    updateRange(node * 2 + 1, mid + 1, end, l, r, val);
    tree[node] = tree[node * 2] + tree[node * 2 + 1];
  }

  T queryRange(int node, int start, int end, int l, int r) {
    if (start > end or start > r or end < l)
      return 0;
    if (updated[node]) {
      tree[node] = lazy[node] * (end - start + 1);
      if (start != end) {
        lazy[node * 2] = lazy[node];
        lazy[node * 2 + 1] = lazy[node];
        updated[node * 2] = updated[node];
        updated[node * 2 + 1] = updated[node];
      }
      updated[node] = false;
    }
    if (start >= l and end <= r) return tree[node];
    int mid = (start + end) / 2;
    T p1 = queryRange(node * 2, start, mid, l, r);
    T p2 = queryRange(node * 2 + 1, mid + 1, end, l, r);
    return p1 + p2;
  }

 public:
  SegmentTree(const std::vector<T>& arr) {
    n = arr.size();
    tree.resize(4 * n);
    lazy.resize(4 * n);
    updated.resize(4 * n, false);
    build(1, 0, n - 1, arr);
  }

  void updateRange(int l, int r, T val) { updateRange(1, 0, n - 1, l, r, val); }

  T queryRange(int l, int r) { return queryRange(1, 0, n - 1, l, r); }
};
```

2.2 DisjointSet

```
struct DisjointSetUnion {
  public:
    DisjointSetUnion()
      : _n(0) {}
    explicit DisjointSetUnion(int n)
      : _n(n)
      , parent_or_size(n, -1) {}

    int merge(int a, int b) {
      assert(0 <= a && a < _n);
      assert(0 <= b && b < _n);
      int x = leader(a), y = leader(b);
      if (x == y) return x;
      if (-parent_or_size[x] < -parent_or_size[y])
        std::swap(x, y);
      parent_or_size[x] += parent_or_size[y];
      parent_or_size[y] = x;
      return x;
    }

    bool same(int a, int b) {
      assert(0 <= a && a < _n);
      assert(0 <= b && b < _n);
      return leader(a) == leader(b);
    }

    int leader(int a) {
      assert(0 <= a && a < _n);
      if (parent_or_size[a] < 0) return a;
      return parent_or_size[a] = leader(parent_or_size[a]);
    }

    int size(int a) {
      assert(0 <= a && a < _n);
      return -parent_or_size[leader(a)];
    }

    std::vector<std::vector<int>> groups() {
      std::vector<int> leader_buf(_n), group_size(_n);
      for (int i = 0; i < _n; i++) {
        leader_buf[i] = leader(i);
        group_size[leader_buf[i]]++;
      }
      std::vector<std::vector<int>> result(_n);
      for (int i = 0; i < _n; i++) {
        result[i].reserve(group_size[i]);
      }
      for (int i = 0; i < _n; i++) {
        result[leader_buf[i]].push_back(i);
      }
      result.erase(
        std::remove_if(
        result.begin(), result.end(),
        [&](const std::vector<int>& v) {
        return v.empty();
        }),
        result.end());
      return result;
```

```
    }

  private:
    int _n;
    // root node: -1 * component size
    // otherwise: parent
    std::vector<int> parent_or_size;
};
```

2.3 undo disjoint set

```
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;
```

2.4 fenwick

```
namespace internal {
template <class T>
using to_unsigned_t = typename to_unsigned<T
    >::type;
}  // namespace internal

template <class T>
struct FenwickTree {
  using U = internal::to_unsigned_t<T>;

 public:
  FenwickTree() : _n(0) {}
  explicit FenwickTree(int n) : _n(n), data(
      n) {}

  void add(int p, T x) {
    assert(0 <= p && p < _n);
    p++;
    while (p <= _n) {
      data[p - 1] += U(x);
      p += p & -p;
    }
  }

  T sum(int l, int r) {
    assert(0 <= l && l <= r && r <= _n);
    return sum(r) - sum(l);
  }

 private:
  int _n;
  std::vector<U> data;

  U sum(int r) {
    U s = 0;
    while (r > 0) {
      s += data[r - 1];
      r -= r & -r;
    }
    return s;
  }
};
```

3 Flow

3.1 min cost flow

```
namespace internal {

template <class T>
struct simple_queue {
  std::vector<T> payload;
  int pos = 0;
  void reserve(int n) { payload.reserve(n);
      }
  int size() const { return int(payload.size
      ()) - pos; }
  bool empty() const { return pos == int(
      payload.size()); }
  void push(const T& t) { payload.push_back(
      t); }
  T& front() { return payload[pos]; }
  void clear() {
    payload.clear();
    pos = 0;
  }
  void pop() { pos++; }
};

template <class E>
struct csr {
  std::vector<int> start;
  std::vector<E> elist;
  explicit csr(int n, const std::vector<std
      ::pair<int, E>>& edges)
      : start(n + 1), elist(edges.size()) {
    for (auto e : edges) {
      start[e.first + 1]++;
    }
    for (int i = 1; i <= n; i++) {
      start[i] += start[i - 1];
    }
    auto counter = start;
    for (auto e : edges) {
      elist[counter[e.first]++] = e.second;
    }
  }
};

}  // namespace internal

template <class Cap, class Cost>
struct MinCostFlowGraph {
 public:
  MinCostFlowGraph() {}
  explicit MinCostFlowGraph(int n) : _n(n)
      {}

  int add_edge(int from, int to, Cap cap,
      Cost cost) {
    assert(0 <= from && from < _n);
    assert(0 <= to && to < _n);
    assert(0 <= cap);
    assert(0 <= cost);
    int m = int(_edges.size());
    _edges.push_back({from, to, cap, 0, cost
        });
    return m;
  }

  struct edge {
    int from, to;
    Cap cap, flow;
    Cost cost;
  };

  edge get_edge(int i) {
    int m = int(_edges.size());
    assert(0 <= i && i < m);
    return _edges[i];
  }
  std::vector<edge> edges() { return _edges;
      }

  std::pair<Cap, Cost> flow(int s, int t) {
    return flow(s, t, std::numeric_limits<
        Cap>::max());
  }
  std::pair<Cap, Cost> flow(int s, int t,
      Cap flow_limit) {
    return slope(s, t, flow_limit).back();
  }
  std::vector<std::pair<Cap, Cost>> slope(
      int s, int t) {
    return slope(s, t, std::numeric_limits<
        Cap>::max());
  }
  std::vector<std::pair<Cap, Cost>> slope(
      int s, int t, Cap flow_limit) {
    assert(0 <= s && s < _n);
    assert(0 <= t && t < _n);
    assert(s != t);

    int m = int(_edges.size());
    std::vector<int> edge_idx(m);

    auto g = [&]() {
      std::vector<int> degree(_n), redge_idx
          (m);
      std::vector<std::pair<int, _edge>>
          elist;
      elist.reserve(2 * m);
      for (int i = 0; i < m; i++) {
        auto e = _edges[i];
        edge_idx[i] = degree[e.from]++;
        redge_idx[i] = degree[e.to]++;
        elist.push_back({e.from, {e.to, -1,
            e.cap - e.flow, e.cost}});
        elist.push_back({e.to, {e.from, -1,
            e.flow, -e.cost}});
      }
      auto _g = internal::csr<_edge>(_n,
          elist);
      for (int i = 0; i < m; i++) {
        auto e = _edges[i];
        edge_idx[i] += _g.start[e.from];
        redge_idx[i] += _g.start[e.to];
        _g.elist[edge_idx[i]].rev =
            redge_idx[i];
        _g.elist[redge_idx[i]].rev =
            edge_idx[i];
      }
      return _g;
    }();

    auto result = slope(g, s, t, flow_limit)
        ;

    for (int i = 0; i < m; i++) {
      auto e = g.elist[edge_idx[i]];
      _edges[i].flow = _edges[i].cap - e.cap
          ;
    }

    return result;
  }

 private:
  int _n;
  std::vector<edge> _edges;

  // inside edge
  struct _edge {
    int to, rev;
    Cap cap;
    Cost cost;
  };

  std::vector<std::pair<Cap, Cost>> slope(
      internal::csr<_edge>& g, int s, int t,
       Cap flow_limit) {
    // variants (C = maxcost):
```

```
    // -(n-1)C <= dual[s] <= dual[i] <= dual[t] = 0
    // reduced cost (= e.cost + dual[e.from] - dual[e.to]) >= 0 for all edge

    // dual_dist[i] = (dual[i], dist[i])
    std::vector<std::pair<Cost, Cost>> dual_dist(_n);
    std::vector<int> prev_e(_n);
    std::vector<bool> vis(_n);
    struct Q {
      Cost key;
      int to;
      bool operator<(Q r) const { return key > r.key; }
    };
    std::vector<int> que_min;
    std::vector<Q> que;
    auto dual_ref = [&]() {
      for (int i = 0; i < _n; i++) {
        dual_dist[i].second = std::numeric_limits<Cost>::max();
      }
      std::fill(vis.begin(), vis.end(), false);
      que_min.clear();
      que.clear();

      // que[0..heap_r) was heapified
      size_t heap_r = 0;

      dual_dist[s].second = 0;
      que_min.push_back(s);
      while (!que_min.empty() || !que.empty()) {
        int v;
        if (!que_min.empty()) {
          v = que_min.back();
          que_min.pop_back();
        } else {
          while (heap_r < que.size()) {
            heap_r++;
            std::push_heap(que.begin(), que.begin() + heap_r);
          }
          v = que.front().to;
          std::pop_heap(que.begin(), que.end());
          que.pop_back();
          heap_r--;
        }
        if (vis[v]) continue;
        vis[v] = true;
        if (v == t) break;
        // dist[v] = shortest(s, v) + dual[s] - dual[v]
        // dist[v] >= 0 (all reduced cost are positive)
        // dist[v] <= (n-1)C
        Cost dual_v = dual_dist[v].first, dist_v = dual_dist[v].second;
        for (int i = g.start[v]; i < g.start[v + 1]; i++) {
          auto e = g.elist[i];
          if (!e.cap) continue;
          // |-dual[e.to] + dual[v]| <= (n-1)C
          // cost <= C - -(n-1)C + 0 = nC
          Cost cost = e.cost - dual_dist[e.to].first + dual_v;
          if (dual_dist[e.to].second - dist_v > cost) {
            Cost dist_to = dist_v + cost;
            dual_dist[e.to].second = dist_to;
            prev_e[e.to] = e.rev;
            if (dist_to == dist_v) {
              que_min.push_back(e.to);
            } else {
              que.push_back(Q{dist_to, e.to});
            }
          }
        }
      }
      if (!vis[t]) {
        return false;
      }

      for (int v = 0; v < _n; v++) {
        if (!vis[v]) continue;
        // dual[v] = dual[v] - dist[t] + dist[v]
        //         = dual[v] - (shortest(s, t) + dual[s] - dual[t]) +
        //           (shortest(s, v) + dual[s] - dual[v]) = - shortest(s, t) + dual[t] + shortest(s, v) = shortest(s, v) - shortest(s, t) >= 0 - (n-1)C
        dual_dist[v].first -= dual_dist[t].second - dual_dist[v].second;
      }
      return true;
    };
    Cap flow = 0;
    Cost cost = 0, prev_cost_per_flow = -1;
    std::vector<std::pair<Cap, Cost>> result = {{Cap(0), Cost(0)}};
    while (flow < flow_limit) {
      if (!dual_ref()) break;
      Cap c = flow_limit - flow;
      for (int v = t; v != s; v = g.elist[prev_e[v]].to) {
        c = std::min(c, g.elist[g.elist[prev_e[v]].rev].cap);
      }
      for (int v = t; v != s; v = g.elist[prev_e[v]].to) {
        auto& e = g.elist[prev_e[v]];
        e.cap += c;
        g.elist[e.rev].cap -= c;
      }
      Cost d = -dual_dist[s].first;
      flow += c;
      cost += c * d;
      if (prev_cost_per_flow == d) {
        result.pop_back();
      }
      result.push_back({flow, cost});
      prev_cost_per_flow = d;
    }
    return result;
  }
};
```

3.2 max flow

```
namespace internal {

template <class T>
struct simple_queue {
  std::vector<T> payload;
  int pos = 0;
  void reserve(int n) { payload.reserve(n); }
  int size() const { return int(payload.size()) - pos; }
  bool empty() const { return pos == int(payload.size()); }
  void push(const T& t) { payload.push_back(t); }
  T& front() { return payload[pos]; }
  void clear() {
    payload.clear();
    pos = 0;
  }
  void pop() { pos++; }
};

}  // namespace internal

template <class Cap>
struct MaxFlowGraph {
 public:
  MaxFlowGraph() : MaxFlowGraph(0) {}
  explicit MaxFlowGraph(int n) : _n(n), g(n) {}

  int add_edge(int from, int to, Cap cap) {
    assert(0 <= from && from < _n);
    assert(0 <= to && to < _n);
    assert(0 <= cap);
    int m = int(pos.size());
    pos.push_back({from, int(g[from].size())});
    int from_id = int(g[from].size());
    int to_id = int(g[to].size());
    if (from == to) to_id++;
    g[from].push_back(_edge{to, to_id, cap});
    g[to].push_back(_edge{from, from_id, 0});
    return m;
  }

  struct edge {
    int from, to;
    Cap cap, flow;
  };

  edge get_edge(int i) {
    int m = int(pos.size());
    assert(0 <= i && i < m);
    auto _e = g[pos[i].first][pos[i].second];
    auto _re = g[_e.to][_e.rev];
    return edge{pos[i].first, _e.to, _e.cap + _re.cap, _re.cap};
  }
  std::vector<edge> edges() {
    int m = int(pos.size());
    std::vector<edge> result;
    for (int i = 0; i < m; i++) {
      result.push_back(get_edge(i));
    }
    return result;
  }
  void change_edge(int i, Cap new_cap, Cap new_flow) {
    int m = int(pos.size());
    assert(0 <= i && i < m);
    assert(0 <= new_flow && new_flow <= new_cap);
    auto& _e = g[pos[i].first][pos[i].second];
    auto& _re = g[_e.to][_e.rev];
    _e.cap = new_cap - new_flow;
    _re.cap = new_flow;
  }

  Cap flow(int s, int t) { return flow(s, t, std::numeric_limits<Cap>::max()); }
  Cap flow(int s, int t, Cap flow_limit) {
    assert(0 <= s && s < _n);
    assert(0 <= t && t < _n);
    assert(s != t);

    std::vector<int> level(_n), iter(_n);
    internal::simple_queue<int> que;

    auto bfs = [&]() {
      std::fill(level.begin(), level.end(), -1);
      level[s] = 0;
      que.clear();
      que.push(s);
      while (!que.empty()) {
        int v = que.front();
        que.pop();
        for (auto e : g[v]) {
          if (e.cap == 0 || level[e.to] >= 0) continue;
          level[e.to] = level[v] + 1;
          if (e.to == t) return;
          que.push(e.to);
        }
      }
    };
    auto dfs = [&](auto self, int v, Cap up) {
      if (v == s) return up;
      Cap res = 0;
      int level_v = level[v];
      for (int& i = iter[v]; i < int(g[v].size()); i++) {
        _edge& e = g[v][i];
        if (level_v <= level[e.to] || g[e.to][e.rev].cap == 0) continue;
```

```
        Cap d = self(self, e.to, std::min(up - res, g[e.to][e.rev].cap));
        if (d <= 0) continue;
        g[v][i].cap += d;
        g[e.to][e.rev].cap -= d;
        res += d;
        if (res == up) return res;
      }
      level[v] = _n;
      return res;
    };

    Cap flow = 0;
    while (flow < flow_limit) {
      bfs();
      if (level[t] == -1) break;
      std::fill(iter.begin(), iter.end(), 0);
      Cap f = dfs(dfs, t, flow_limit - flow);
      if (!f) break;
      flow += f;
    }
    return flow;
  }

  std::vector<bool> min_cut(int s) {
    std::vector<bool> visited(_n);
    internal::simple_queue<int> que;
    que.push(s);
    while (!que.empty()) {
      int p = que.front();
      que.pop();
      visited[p] = true;
      for (auto e : g[p]) {
        if (e.cap && !visited[e.to]) {
          visited[e.to] = true;
          que.push(e.to);
        }
      }
    }
    return visited;
  }

 private:
  int _n;
  struct _edge {
    int to, rev;
    Cap cap;
  };
  std::vector<std::pair<int, int>> pos;
  std::vector<std::vector<_edge>> g;
};
```

4 Graph

4.1 tree centroid

```
// 这份代码默认节点编号从 1 开始，即 i ∈ [1, n]
int size[MAXN],  // 这个节点的「大小」(所有子树上节点数 + 该节点)
    weight[MAXN],  // 这个节点的「重量」，即所有子树「大小」的最大值
    centroid[2];  // 用于记录树的重心(存的是节点编号)

void GetCentroid(int cur, int fa) {  // cur 表示当前节点 (current)
  size[cur] = 1;
  weight[cur] = 0;
  for (int i = head[cur]; i != -1; i = e[i].nxt) {
    if (e[i].to != fa) {  // e[i].to 表示这条有向边所通向的节点。
      GetCentroid(e[i].to, cur);
      size[cur] += size[e[i].to];
      weight[cur] = max(weight[cur], size[e[i].to]);
    }
  }
  weight[cur] = max(weight[cur], n - size[cur]);
  if (weight[cur] <= n / 2) {  // 依照树的重心的定义统计
    centroid[centroid[0] != 0] = cur;
  }
}
```

4.2 tarjan

```
int dfn[N], low[N], dfncnt, s[N], in_stack[N], tp;
int scc[N], sc;  // 结点 i 所在 SCC 的编号
int sz[N];       // 强连通 i 的大小

void tarjan(int u) {
  low[u] = dfn[u] = ++dfncnt, s[++tp] = u, in_stack[u] = 1;
  for (int i = h[u]; i; i = e[i].nex) {
    const int &v = e[i].t;
    if (!dfn[v]) {
      tarjan(v);
      low[u] = min(low[u], low[v]);
    } else if (in_stack[v]) {
      low[u] = min(low[u], dfn[v]);
    }
  }
  if (dfn[u] == low[u]) {
    ++sc;
    while (s[tp] != u) {
      scc[s[tp]] = sc;
      sz[sc]++;
      in_stack[s[tp]] = 0;
      --tp;
    }
    scc[s[tp]] = sc;
    sz[sc]++;
    in_stack[s[tp]] = 0;
    --tp;
  }
}
```

4.3 heavy light decomposition

```
const int kMax = 1e5 + 5;
int n, m, r, kMod, op, x, y, z, cnt = 0;
int a[kMax] = {}, dep[kMax] = {}, fa[kMax] = {}, siz[kMax] = {},
    hson[kMax] = {};
int nid[kMax] = {}, na[kMax] = {}, ltop[kMax] = {};
int seg[kMax << 2] = {}, tag[kMax << 2] = {};
vector<int> g[kMax];

void dfs1(int u, int f) {
  dep[u] = dep[f] + 1;
  fa[u] = f;
  siz[u] = 1;
  for (auto v : g[u]) {
    if (v == f) continue;
    dfs1(v, u);
    siz[u] += siz[v];
    if (siz[v] > siz[hson[u]]) hson[u] = v;
  }
}

void dfs2(int u, int cur_ltop) {
  nid[u] = ++cnt;
  na[cnt] = a[u];
  ltop[u] = cur_ltop;
  if (hson[u]) dfs2(hson[u], cur_ltop);
  for (auto v : g[u]) {
    if (v == fa[u] || v == hson[u]) continue;
    dfs2(v, v);
  }
}

void build(int u, int l, int r) {
  if (l == r) {
    seg[u] = na[l] % kMod;
    return;
  }
  int mid = (l + r) >> 1;
  build(u << 1, l, mid);
  build(u << 1 | 1, mid + 1, r);
  seg[u] = (seg[u << 1] + seg[u << 1 | 1]) % kMod;
}

void push_down(int u, int l, int r) {
  if (tag[u]) {
    int mid = (l + r) >> 1;
    seg[u << 1] = (seg[u << 1] + tag[u] * (mid - l + 1)) % kMod;
    seg[u << 1 | 1] = (seg[u << 1 | 1] + tag[u] * (r - mid)) % kMod;
    tag[u << 1] = (tag[u << 1] + tag[u]) % kMod;
    tag[u << 1 | 1] = (tag[u << 1 | 1] + tag[u]) % kMod;
    tag[u] = 0;
  }
}

int query(int u, int l, int r, int ql, int qr) {
  if (ql <= l && r <= qr) return seg[u];
  push_down(u, l, r);
  int mid = (l + r) >> 1, res = 0;
  if (ql <= mid) res = (res + query(u << 1, l, mid, ql, qr)) % kMod;
  if (qr > mid) res = (res + query(u << 1 | 1, mid + 1, r, ql, qr)) % kMod;
  return res;
}

void update(int u, int l, int r, int ql, int qr, int k) {
  if (ql <= l && r <= qr) {
    seg[u] = (seg[u] + k * (r - l + 1)) % kMod;
    tag[u] = (tag[u] + k) % kMod;
    return;
  }
  push_down(u, l, r);
  int mid = (l + r) >> 1;
  if (ql <= mid) update(u << 1, l, mid, ql, qr, k);
  if (qr > mid) update(u << 1 | 1, mid + 1, r, ql, qr, k);
  seg[u] = (seg[u << 1] + seg[u << 1 | 1]) % kMod;
}

int query_path(int u, int v) {
  int res = 0;
  while (ltop[u] != ltop[v]) {
    if (dep[ltop[u]] < dep[ltop[v]]) swap(u, v);
    res = (res + query(1, 1, n, nid[ltop[u]], nid[u])) % kMod;
    u = fa[ltop[u]];
  }
  if (dep[u] > dep[v]) swap(u, v);
  res = (res + query(1, 1, n, nid[u], nid[v])) % kMod;
  return res;
}

void update_path(int u, int v, int k) {
  k %= kMod;
  while (ltop[u] != ltop[v]) {
    if (dep[ltop[u]] < dep[ltop[v]]) swap(u, v);
    update(1, 1, n, nid[ltop[u]], nid[u], k);
    u = fa[ltop[u]];
  }
  if (dep[u] > dep[v]) swap(u, v);
  update(1, 1, n, nid[u], nid[v], k);
}

int query_subtree(int u) { return query(1, 1, n, nid[u], nid[u] + siz[u] - 1); }

void update_subtree(int u, int k) {
```

```
  update(1, 1, n, nid[u], nid[u] + siz[u] - 1, k);
}

int32_t main() {
  cin.tie(nullptr)->sync_with_stdio(false);
  cin >> n >> m >> r >> kMod;
  for (int i = 1; i <= n; i++) cin >> a[i];
  for (int i = 1; i < n; i++) {
    cin >> x >> y;
    g[x].push_back(y);
    g[y].push_back(x);
  }

  dfs1(r, 0);
  dfs2(r, r);
  build(1, 1, n);

  while (m--) {
    cin >> op >> x;
    if (op == 1) {
      cin >> y >> z;
      update_path(x, y, z);
    } else if (op == 2) {
      cin >> y;
      cout << query_path(x, y) << '\n';
    } else if (op == 3) {
      cin >> z;
      update_subtree(x, z);
    } else {
      cout << query_subtree(x) << '\n';
    }
  }
  return 0;
}
```

5 Number Theory

5.1 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 log_mod(const LL &a, const LL &b, const LL &p) {
  // a ^ x = b ( mod p )
  int m=sqrt(p+.5), e=1;
  LL v=inv(modexp(a,m,p), p);
  map<LL,int> x;
  x[1]=0;
  for(int i=1;i<m;++i) {
    e = LLmul(e,a,p);
    if(!x.count(e)) x[e] = i;
  }
  for(int i=0;i<m;++i) {
    if(x.count(b)) return i*m + x[b];
    b = LLmul(b,v,p);
  }
  return -1;
}

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)%M;
    /*如果m[i]是質數，Euler(m[i])-1=m[i]-2，就不用算Euler了*/
  }
  return ans;
}

//java code
//求sqrt(N)的連分數
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);
}
```

5.2 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;
  }
}
```

5.3 FFT

```cpp
template<typename T,typename VT=vector<
    complex<T> > >
struct FFT{
  const T pi;
  FFT(const T pi=acos((T)-1)):pi(pi){}
  unsigned bit_reverse(unsigned a,int len){
a=((a&0x55555555U)<<1)|((a&0xAAAAAAAAU)>>1);
a=((a&0x33333333U)<<2)|((a&0xCCCCCCCCU)>>2);
a=((a&0x0F0F0F0FU)<<4)|((a&0xF0F0F0F0U)>>4);
a=((a&0x00FF00FFU)<<8)|((a&0xFF00FF00U)>>8);
a=((a&0x0000FFFFU)<<16)|((a&0xFFFF0000U)
    >>16);
    return a>>(32-len);
  }
  void fft(bool is_inv,VT &in,VT &out,int N)
      {
    int bitlen=__lg(N),num=is_inv?-1:1;
    for(int i=0;i<N;++i)out[bit_reverse(i,
        bitlen)]=in[i];
    for(int step=2;step<=N;step<<=1){
      const int mh=step>>1;
      for(int i=0;i<mh;++i){
        complex<T> wi=exp(complex<T>(0,i*num
            *pi/mh));
        for(int j=i;j<N;j+=step){
          int k=j+mh;
          complex<T> u=out[j],t=wi*out[k];
          out[j]=u+t;
          out[k]=u-t;
        }
      }
    }
    if(is_inv)for(int i=0;i<N;++i)out[i]/=N;
  }
};
```

5.4 質因數分解

```cpp
LL func(const LL n,const LL mod,const int c)
    {
  return (LLmul(n,n,mod)+c+mod)%mod;
}

LL pollorrho(const LL n, const int c) {//循
    環節長度
  LL a=1, b=1;
  a=func(a,n,c)%n;
  b=func(b,n,c)%n; b=func(b,n,c)%n;
  while(gcd(abs(a-b),n)==1) {
    a=func(a,n,c)%n;
    b=func(b,n,c)%n; b=func(b,n,c)%n;
  }
  return gcd(abs(a-b),n);
}

void prefactor(LL &n, vector<LL> &v) {
  for(int i=0;i<12;++i) {
    while(n%prime[i]==0) {
      v.push_back(prime[i]);
      n/=prime[i];
    }
  }
}

void smallfactor(LL n, vector<LL> &v) {
  if(n<MAXPRIME) {
    while(isp[(int)n]) {
      v.push_back(isp[(int)n]);
      n/=isp[(int)n];
    }
    v.push_back(n);
  } else {
    for(int i=0;i<primecnt&&prime[i]*prime[i
        ]<=n;++i) {
      while(n%prime[i]==0) {
        v.push_back(prime[i]);
        n/=prime[i];
      }
    }
    if(n!=1) v.push_back(n);
  }
}

void comfactor(const LL &n, vector<LL> &v) {
  if(n<1e9) {
    smallfactor(n,v);
    return;
  }
  if(Isprime(n)) {
    v.push_back(n);
    return;
  }
  LL d;
  for(int c=3;;++c) {
    d = pollorrho(n,c);
    if(d!=n) break;
  }
  comfactor(d,v);
  comfactor(n/d,v);
}

void Factor(const LL &x, vector<LL> &v) {
  LL n = x;
  if(n==1) { puts("Factor 1"); return; }
  prefactor(n,v);
  if(n==1) return;
  comfactor(n,v);
  sort(v.begin(),v.end());
}

void AllFactor(const LL &n,vector<LL> &v) {
  vector<LL> tmp;
  Factor(n,tmp);
  v.clear();
  v.push_back(1);
  int len;
  LL now=1;
  for(int i=0;i<tmp.size();++i) {
    if(i==0 || tmp[i]!=tmp[i-1]) {
      len = v.size();
      now = 1;
    }
    now*=tmp[i];
    for(int j=0;j<len;++j)
      v.push_back(v[j]*now);
  }
}
```

5.5 find real root

```cpp
// 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);
  // 視情況把答案 +eps，避免 -0
}
```

5.6 中國剩餘定理

```cpp
int exgcd(int a, int b, int &x, int &y) {
  int x1 = 1, x2 = 0, x3 = 0, x4 = 1;
  while (b != 0) {
    int c = a / b;
    std::tie(x1, x2, x3, x4, a, b) =
        std::make_tuple(x3, x4, x1 - x3 * c,
            x2 - x4 * c, b, a - b * c);
  }
  x = x1, y = x2;
  return a;
}
LL CRT(int k, LL* a, LL* r) {
  LL n = 1, ans = 0;
  for (int i = 1; i <= k; i++) n = n * r[i];
  for (int i = 1; i <= k; i++) {
    LL m = n / r[i], b, y;
    exgcd(m, r[i], b, y);  // b * m mod r[i]
        = 1
    ans = (ans + a[i] * m * b % n) % n;
  }
  return (ans % n + n) % n;
}
```

5.7 linear sieve

```cpp
const int N = 10000000;
vector<int> lp(N+1);
vector<int> pr;

for (int i=2; i <= N; ++i) {
    if (lp[i] == 0) {
        lp[i] = i;
        pr.push_back(i);
    }
    for (int j = 0; i * pr[j] <= N; ++j) {
        lp[i * pr[j]] = pr[j];
        if (pr[j] == lp[i]) {
            break;
        }
    }
}
```

5.8 Matrix

```cpp
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;
  }
};
```

5.9 數位統計

```
ll d[65], dp[65][2];//up區間是不是完整
ll dfs(int p,bool is8,bool up){
  if(!p)return 1; // 回傳0是不是答案
  if(!up&&~dp[p][is8])return dp[p][is8];
  int mx = up?d[p]:9;//可以用的有那些
  ll ans=0;
  for(int i=0;i<=mx;++i){
    if( is8&&i==7 )continue;
    ans += dfs(p-1,i==8,up&&i==mx);
  }
  if(!up)dp[p][is8]=ans;
  return ans;
}
ll f(ll N){
  int k=0;
  while(N){  // 把數字先分解到陣列
    d[++k] = N%10;
    N/=10;
  }
  return dfs(k,false,true);
}
```

6 String

6.1 manacher

```
//原字串: asdsasdsa
//先把字串變成這樣: @#a#s#d#s#a#s#d#s#a#
void manacher(char *s,int len,int *z){
  int l=0,r=0;
  for(int i=1;i<len;++i){
    z[i]=r>i?min(z[2*l-i],r-i):1;
    while(s[i+z[i]]==s[i-z[i]])++z[i];
    if(z[i]+i>r)r=z[i]+i,l=i;
  }//ans = max(z)-1
}
```

6.2 reverseBWT

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

6.3 suffix array lcp

```
#define radix_sort(x,y){\
  for(i=0;i<A;++i)c[i]=0;\
  for(i=0;i<n;++i)c[x[y[i]]]++;\
  for(i=1;i<A;++i)c[i]+=c[i-1];\
  for(i=n-1;~i;--i)sa[--c[x[y[i]]]]=y[i];\
}
#define AC(r,a,b)\
  r[a]!=r[b]||a+k>=n||r[a+k]!=r[b+k]
void suffix_array(const char *s,int n,int *
    sa,int *rank,int *tmp,int *c){
  int A='z'+1,i,k,id=0;
  for(i=0;i<n;++i)rank[tmp[i]=i]=s[i];
  radix_sort(rank,tmp);
  for(k=1;id<n-1;k<<=1){
    for(id=0,i=n-k;i<n;++i)tmp[id++]=i;
    for(i=0;i<n;++i)
      if(sa[i]>=k)tmp[id++]=sa[i]-k;
    radix_sort(rank,tmp);
    swap(rank,tmp);
    for(rank[sa[0]]=id=0,i=1;i<n;++i)
      rank[sa[i]]=id+=AC(tmp,sa[i-1],sa[i]);
    A=id+1;
  }
}
//h:高度數組 sa:後綴數組 rank:排名
void suffix_array_lcp(const char *s,int len,
    int *h,int *sa,int *rank){
  for(int i=0;i<len;++i)rank[sa[i]]=i;
  for(int i=0,k=0;i<len;++i){
    if(rank[i]==0)continue;
    if(k)--k;
    while(s[i+k]==s[sa[rank[i]-1]+k])++k;
    h[rank[i]]=k;
  }
  h[0]=0;// h[k]=lcp(sa[k],sa[k-1]);
}
```

6.4 KMP

```
/*產生fail function*/
void kmp_fail(char *s,int len,int *fail){
  int id=-1;
  fail[0]=-1;
  for(int i=1;i<len;++i){
    while(~id&&s[id+1]!=s[i])id=fail[id];
    if(s[id+1]==s[i])++id;
    fail[i]=id;
  }
}
/*以字串B匹配字串A，傳回匹配成功的數量(用B的
    fail)*/
int kmp_match(char *A,int lenA,char *B,int
    lenB,int *fail){
  int id=-1,ans=0;
  for(int i=0;i<lenA;++i){
    while(~id&&B[id+1]!=A[i])id=fail[id];
    if(B[id+1]==A[i])++id;
    if(id==lenB-1){/*匹配成功*/
      ++ans, id=fail[id];
    }
  }
  return ans;
}
```

6.5 hash

```
#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;
}
```

6.6 AC 自動機

```
template<char L='a',char R='z'>
class ac_automaton{
  struct joe{
    int next[R-L+1],fail,efl,ed,cnt_dp,vis;
    joe():ed(0),cnt_dp(0),vis(0){
      for(int i=0;i<=R-L;++i)next[i]=0;
    }
  };
public:
  std::vector<joe> S;
  std::vector<int> q;
  int qs,qe,vt;
  ac_automaton():S(1),qs(0),qe(0),vt(0){}
  void clear(){
```

```
    q.clear();
    S.resize(1);
    for(int i=0;i<=R-L;++i)S[0].next[i]=0;
    S[0].cnt_dp=S[0].vis=qs=qe=vt=0;
  }
  void insert(const char *s){
    int o=0;
    for(int i=0,id;s[i];++i){
      id=s[i]-L;
      if(!S[o].next[id]){
        S.push_back(joe());
        S[o].next[id]=S.size()-1;
      }
      o=S[o].next[id];
    }
    ++S[o].ed;
  }
  void build_fail(){
    S[0].fail=S[0].efl=-1;
    q.clear();
    q.push_back(0);
    ++qe;
    while(qs!=qe){
      int pa=q[qs++],id,t;
      for(int i=0;i<=R-L;++i){
        t=S[pa].next[i];
        if(!t)continue;
        id=S[pa].fail;
        while(~id&&!S[id].next[i])id=S[id].
            fail;
        S[t].fail=~id?S[id].next[i]:0;
        S[t].efl=S[S[t].fail].ed?S[t].fail:S
            [S[t].fail].efl;
        q.push_back(t);
        ++qe;
      }
    }
  }
  /*DP出每個前綴在字串s出現的次數並傳回所有
      字串被s匹配成功的次數O(N+M)*/
  int match_0(const char *s){
    int ans=0,id,p=0,i;
    for(i=0;s[i];++i){
      id=s[i]-L;
      while(!S[p].next[id]&&p)p=S[p].fail;
      if(!S[p].next[id])continue;
      p=S[p].next[id];
      ++S[p].cnt_dp;/*匹配成功則它所有後綴都
          可以被匹配(DP計算)*/
    }
    for(i=qe-1;i>=0;--i){
      ans+=S[q[i]].cnt_dp*S[q[i]].ed;
      if(~S[q[i]].fail)S[S[q[i]].fail].
          cnt_dp+=S[q[i]].cnt_dp;
    }
    return ans;
  }
  /*多串匹配走efl邊並傳回所有字串被s匹配成功
      的次數O(N*M^1.5)*/
  int match_1(const char *s)const{
    int ans=0,id,p=0,t;
    for(int i=0;s[i];++i){
      id=s[i]-L;
      while(!S[p].next[id]&&p)p=S[p].fail;
      if(!S[p].next[id])continue;
      p=S[p].next[id];
      if(S[p].ed)ans+=S[p].ed;
      for(t=S[p].efl;~t;t=S[t].efl){
        ans+=S[t].ed;/*因為都走efl邊所以保證
            匹配成功*/
      }
    }
    return ans;
  }
  /*枚舉(s的子字串nA)的所有相異字串各恰一次
      並傳回次數O(N*M^(1/3))*/
  int match_2(const char *s){
    int ans=0,id,p=0,t;
    ++vt;
    /*把戳記vt+=1，只要vt沒溢位，所有S[p].
        vis==vt就會變成false
    這種利用vt的方法可以O(1)歸零vis陣列*/
    for(int i=0;s[i];++i){
      id=s[i]-L;
      while(!S[p].next[id]&&p)p=S[p].fail;
      if(!S[p].next[id])continue;
      p=S[p].next[id];
      if(S[p].ed&&S[p].vis!=vt){
        S[p].vis=vt;
        ans+=S[p].ed;
      }
      for(t=S[p].efl;~t&&S[t].vis!=vt;t=S[t
          ].efl){
        S[t].vis=vt;
        ans+=S[t].ed;/*因為都走efl邊所以保證
            匹配成功*/
      }
    }
    return ans;
  }
  /*把AC自動機變成真的自動機*/
  void evolution(){
    for(qs=1;qs!=qe;){
      int p=q[qs++];
      for(int i=0;i<=R-L;++i)
        if(S[p].next[i]==0)S[p].next[i]=S[S[
            p].fail].next[i];
    }
  }
};
```

6.7 minimal string rotation

```
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
}
```

6.8 Z

```
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;
  }
}
```

7 Tarjan

7.1 dominator tree

```
struct dominator_tree{
  static const int MAXN=5005;
  int n;// 1-base
  vector<int> G[MAXN], rG[MAXN];
  int pa[MAXN], dfn[MAXN], id[MAXN], dfnCnt;
  int semi[MAXN], idom[MAXN], best[MAXN];
  vector<int> tree[MAXN]; // tree here
  void init(int _n){
    n = _n;
    for(int i=1; i<=n; ++i)
      G[i].clear(), rG[i].clear();
  }
  void add_edge(int u, int v){
    G[u].push_back(v);
    rG[v].push_back(u);
  }
  void dfs(int u){
    id[dfn[u]=++dfnCnt]=u;
    for(auto v:G[u]) if(!dfn[v])
      dfs(v),pa[dfn[v]]=dfn[u];
  }
  int find(int y,int x){
    if(y <= x) return y;
    int tmp = find(pa[y],x);
    if(semi[best[y]] > semi[best[pa[y]]])
      best[y] = best[pa[y]];
    return pa[y] = tmp;
  }
  void tarjan(int root){
    dfnCnt = 0;
    for(int i=1; i<=n; ++i){
      dfn[i] = idom[i] = 0;
      tree[i].clear();
      best[i] = semi[i] = i;
    }
    dfs(root);
    for(int i=dfnCnt; i>1; --i){
      int u = id[i];
      for(auto v:rG[u]) if(v=dfn[v]){
        find(v,i);
        semi[i]=min(semi[i],semi[best[v]]);
      }
      tree[semi[i]].push_back(i);
      for(auto v:tree[pa[i]]){
        find(v, pa[i]);
        idom[v] = semi[best[v]]==pa[i]
            ? pa[i] : best[v];
      }
      tree[pa[i]].clear();
    }
    for(int i=2; i<=dfnCnt; ++i){
      if(idom[i] != semi[i])
        idom[i] = idom[idom[i]];
      tree[id[idom[i]]].push_back(id[i]);
    }
  }
}dom;
```

7.2 橋連通分量

```
#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;
  }
}
```

7.3 雙連通分量 & 割點

```
#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;
}
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;
  }
}
```

7.4 tnfshb017 2 sat

```
#include<bits/stdc++.h>
using namespace std;
#define MAXN 8001
#define MAXN2 MAXN*4
#define n(X) ((X)+2*N)
vector<int>  v[MAXN2], rv[MAXN2], vis_t;
int N,M;
void addedge(int s,int e){
  v[s].push_back(e);
  rv[e].push_back(s);
}
int scc[MAXN2];
bool vis[MAXN2]={false};
void dfs(vector<int> *uv,int n,int k=-1){
  vis[n]=true;
  for(int i=0;i<uv[n].size();++i)
    if(!vis[uv[n][i]])
      dfs(uv,uv[n][i],k);
  if(uv==v)vis_t.push_back(n);
  scc[n]=k;
}
void solve(){
  for(int i=1;i<=N;++i){
    if(!vis[i])dfs(v,i);
    if(!vis[n(i)])dfs(v,n(i));
  }
  memset(vis,0,sizeof(vis));
  int c=0;
  for(int i=vis_t.size()-1;i>=0;--i)
    if(!vis[vis_t[i]])
      dfs(rv,vis_t[i],c++);
}
int main(){
  int a,b;
  scanf("%d%d",&N,&M);
  for(int i=1;i<=N;++i){
    // (A or B)&(!A & !B) A^B
    a=i*2-1;
    b=i*2;
    addedge(n(a),b);
    addedge(n(b),a);
    addedge(a,n(b));
    addedge(b,n(a));
  }
  while(M--){
    scanf("%d%d",&a,&b);
    a = a>0?a*2-1:-a*2;
    b = b>0?b*2-1:-b*2;
    // A or B
    addedge(n(a),b);
    addedge(n(b),a);
  }
  solve();
  bool check=true;
  for(int i=1;i<=2*N;++i)
    if(scc[i]==scc[n(i)])
      check=false;
  if(check){
    printf("%d\n",N);
    for(int i=1;i<=2*N;i+=2){
      if(scc[i]>scc[i+2*N]) putchar('+');
      else putchar('-');
    }
    puts("");
  }else puts("0");
  return 0;
}
```

8 other

8.1 上下最大正方形

```
void solve(int n,int a[],int b[]){// 1-base
  int ans=0;
  deque<int>da,db;
  for(int l=1,r=1;r<=n;++r){
    while(da.size()&&a[da.back()]>=a[r]){
      da.pop_back();
    }
    da.push_back(r);
    while(db.size()&&b[db.back()]>=b[r]){
      db.pop_back();
    }
    db.push_back(r);
    for(int d=a[da.front()]+b[db.front()];r-
        l+1>d;++l){
      if(da.front()==l)da.pop_front();
      if(db.front()==l)db.pop_front();
      if(da.size()&&db.size()){
        d=a[da.front()]+b[db.front()];
      }
    }
    ans=max(ans,r-l+1);
  }
  printf("%d\n",ans);
}
```

8.2 WhatDay

```
int whatday(int y,int m,int d){
  if(m<=2)m+=12,--y;
  if(y<1752||y==1752&&m<9||y==1752&&m==9&&d
      <3)
    return (d+2*m+3*(m+1)/5+y+y/4+5)%7;
  return (d+2*m+3*(m+1)/5+y+y/4-y/100+y/400)
      %7;
}
```

8.3 最大矩形

```
LL max_rectangle(vector<int> s){
  stack<pair<int,int > > st;
  st.push(make_pair(-1,0));
  s.push_back(0);
  LL ans=0;
  for(size_t i=0;i<s.size();++i){
    int h=s[i];
    pair<int,int > now=make_pair(h,i);
    while(h<st.top().first){
      now=st.top();
      st.pop();
      ans=max(ans,(LL)(i-now.second)*now.
          first);
    }
    if(h>st.top().first){
      st.push(make_pair(h,now.second));
    }
  }
  return ans;
}
```

9 zformula

9.1 formula

9.1.1 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

9.1.2 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) ans=$\sum_{W_v>0} W_v - flow(S, T)$
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：
   $l = 0, r = U, eps = 1/n^2$
   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
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排序
   (h) 區間圖染色: 用線段樹做

9.1.3 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 : $O\left(V^{1/2}E\right)$

9.1.4 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;
}
```

9.1.5 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 $M(\theta) = \begin{pmatrix} cos\theta & -sin\theta \\ sin\theta & cos\theta \end{pmatrix}$

9.1.6 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. $\sum_{i=1}^{n}\sum_{j=1}^{n} lcm(i,j) = n\sum_{d|n} d \times \phi(d)$

9.1.7 排組公式

1. k 卡特蘭 $\frac{C_n^{kn}}{n(k-1)+1} \cdot C_m^n = \frac{n!}{m!(n-m)!}$
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) $\sum_{k\le m}\binom{m+r}{k}x^k y^k = \sum_{k\le m}\binom{-r}{k}(-x)^k(x+y)^{m-k}$

9.1.8 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. $B_2 = 1/6, B_4 = -1/30, B_6 = 1/42, B_8 = -1/30, B_{10} = 5/66, B_{12} = -691/2730, B_{14} = 7/6, B_{16} = -3617/510, B_{18} = 43867/798, B_{20} = -174611/330,$

9.1.9 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 $\frac{1}{24}\left(3^6 + 6 \times 3^3 + 3 \times 3^4 + 8 \times 3^2 + 6 \times 3^3\right) = 57$

9.1.10 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)$M[i][i] = degree(V_i)$,$M[i][j] = -1$,if have $E(i,j)$,0 if no edge. delete any one row and col in A, $ans = det(A)$

ACM ICPC Team Reference - Angry Crow Takes Flight!

Contents

ACM ICPC Judge Test - Angry Crow Takes Flight!

C++ Resource Test

```cpp
#include <bits/stdc++.h>
using namespace std;

namespace system_test {

const size_t KB = 1024;
const size_t MB = KB * 1024;
const size_t GB = MB * 1024;

size_t block_size, bound;
void stack_size_dfs(size_t depth = 1) {
  if (depth >= bound)
    return;
  int8_t ptr[block_size]; // 若無法編譯將 block_size 改成常數
  memset(ptr, 'a', block_size);
  cout << depth << endl;
  stack_size_dfs(depth + 1);
}

void stack_size_and_runtime_error(size_t block_size, size_t bound = 1024) {
  system_test::block_size = block_size;
  system_test::bound = bound;
  stack_size_dfs();
}

double speed(int iter_num) {
  const int block_size = 1024;
  volatile int A[block_size];
  auto begin = chrono::high_resolution_clock::now();
  while (iter_num--)
    for (int j = 0; j < block_size; ++j)
      A[j] += j;
  auto end = chrono::high_resolution_clock::now();
  chrono::duration<double> diff = end - begin;
  return diff.count();
}

void runtime_error_1() {
  // Segmentation fault
  int *ptr = nullptr;
  *(ptr + 7122) = 7122;
}

void runtime_error_2() {
  // Segmentation fault
  int *ptr = (int *)memset;
  *ptr = 7122;
}

void runtime_error_3() {
  // munmap_chunk(): invalid pointer
  int *ptr = (int *)memset;
  delete ptr;
}

void runtime_error_4() {
  // free(): invalid pointer
  int *ptr = new int[7122];
  ptr += 1;
  delete[] ptr;
}

void runtime_error_5() {
  // maybe illegal instruction
  int a = 7122, b = 0;
  cout << (a / b) << endl;
}

void runtime_error_6() {
  // floating point exception
  volatile int a = 7122, b = 0;
  cout << (a / b) << endl;
}

void runtime_error_7() {
  // call to abort.
  assert(false);
}

} // namespace system_test

#include <sys/resource.h>
void print_stack_limit() { // only work in Linux
  struct rlimit l;
  getrlimit(RLIMIT_STACK, &l);
  cout << "stack_size = " << l.rlim_cur << " byte" << endl;
}
```