1 Computational_Geometry

1.1 formula.tex

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

1.2 Geometry.cpp

```cpp
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(
    夾角)*/
    return fabs(atan2(fabs(cross(b)),dot(b))
      );
  }
};
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 cross(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);
  }
  bool point_on_segment(const point<T>&p)
    const{/*點是否線段上*/
    return cross(p)==0&&(p1-p).dot(p2-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();
  }
  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> ans;
    T d=a*a+b*b;
    ans.x=(b*b*p.x-a*a*p.x-2*a*b*p.y-2*a*c)/
      d;
    ans.y=(a*a*p.y-b*b*p.y-2*a*b*p.x-2*b*c)/
      d;
    return ans;
  }
  bool equal(const line &l)const{/*直線相等
    */
    return cross(l.p1)==0&&cross(l.p2)==0;
  }
  bool parallel(const line &l)const{/*直線平
    行*/
    return (p1-p2).cross(l.p1-l.p2)==0;
  }
  bool cross_seg(const line &l)const{/*直線
    是否交線段*/
    return (p2-p1).cross(l.p1)*(p2-p1).cross
      (l.p2)<=0;
  }
  char 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)?(cross(l.p1)==0?-1:0)
      :1;
  }
  char seg_intersect(const line &l)const{/*
    線段相交情況，-1無限多點、1交於一點、0
    不相交*/
    T c1=(p2-p1).cross(l.p1-p1);
    T c2=(p2-p1).cross(l.p2-p1);
    T c3=(l.p2-l.p1).cross(p1-l.p1);
    T c4=(l.p2-l.p1).cross(p2-l.p1);
    if(c1==0&&c2==0){
      if(p1==l.p1&&(p2-p1).dot(l.p2)<=0)
        return 1;
      if(p1==l.p2&&(p2-p1).dot(l.p1)<=0)
        return 1;
      if(p2==l.p1&&(p1-p2).dot(l.p2)<=0)
        return 1;
      if(p2==l.p2&&(p1-p2).dot(l.p1)<=0)
        return 1;
      return -1;
    }else if(c1*c2<=0&&c3*c4<=0)return 1;
    return 0;
  }
  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*/
    T c1=(p2-p1).cross(l.p1-p1);
    T c2=(p2-p1).cross(l.p2-p1);
    T c3=(l.p2-l.p1).cross(p1-l.p1);
    T c4=(l.p2-l.p1).cross(p2-l.p1);
    if(c1==0&&c2==0){
      if(p1==l.p1&&(p2-p1).dot(l.p2)<=0)
        return p1;
      if(p1==l.p2&&(p2-p1).dot(l.p1)<=0)
        return p1;
      if(p2==l.p1&&(p1-p2).dot(l.p2)<=0)
        return p2;
      if(p2==l.p2&&(p1-p2).dot(l.p1)<=0)
        return p2;
    }else if(c1*c2<=0&&c3*c4<=0)return
      line_intersection(l);
    //return INF;
  }
};
template<typename T>
struct polygon{
  polygon(){}
  vector<point<T> > p;//逆時針順序
  T area()const{/*多邊形面積*/
    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{/*多邊形重
    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;
  }
  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.cross(p[i])>=0){
        ans.p.push_back(p[i]);
        if(l.cross(p[j])<0)
          ans.p.push_back(l.
            line_intersection(line<T>(p[i
            ],p[j])));
      }else if(l.cross(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){
    return (a.x<b.x)||(a.x==b.x&&a.y<b.y);/*
      凸包排序函數*/
  }
  void graham(vector<point<T> > &s){/*凸包*/
    sort(s.begin(),s.end(),graham_cmp);
    p.resize(s.size()+1);
    int m=0;
    for(int i=0;i<(int)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);
  }
  inline static char sign(const point<T>&t){
    return (t.y==0?t.x:t.y)<0;
```

```
  }
  inline 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].cross(px[R-1])<=0)--R;
      while(L<R&&s[i].cross(px[L])<=0)++L;
      q[++R]=s[i];
      if(q[R].parallel(q[R-1])){
        --R;
        if(q[R].cross(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].cross(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，用到根號*/
    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);
    T t1=((l.p1-p1).cross(d2)).dot(D)/D.abs2();
    T t2=((l.p1-p1).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=105;
  struct face{
    int a,b,c;
    bool use;
    face(){}
    face(int a,int b,int c):a(a),b(b),c(c),use(1){}
  };
  vector<point3D<T> > pt;
  vector<face> fc;
  int fid[MAXN][MAXN];
  static bool point_cmp(const point3D<T> &a,const point3D<T> &b){
    return a.x<b.x||(a.x==b.x&&(a.y<b.y||(a.y==b.y&&a.z<b.z)));
  }
  bool outside(int p,int a,int b,int c)const{
    return tetrahedron<T>(pt[a],pt[b],pt[c],pt[p]).volume6()<0;
  }
  bool outside(int p,int f)const{return outside(p,fc[f].a,fc[f].b,fc[f].c);}
  void add_face(int a,int b,int c,int p){
    if(outside(p,a,b,c))fid[c][b]=fid[b][a]=fid[a][c]=fc.size(),fc.push_back(face(c,b,a));
    else fid[a][b]=fid[b][c]=fid[c][a]=fc.size(),fc.push_back(face(a,b,c));
  }
  bool dfs(int p,int f){
    if(!fc[f].use)return true;
    if(outside(p,f)){
      int a=fc[f].a,b=fc[f].b,c=fc[f].c;
      fc[f].use=false;
      if(!dfs(p,fid[b][a]))add_face(p,a,b,c);
      if(!dfs(p,fid[c][b]))add_face(p,b,c,a);
      if(!dfs(p,fid[a][c]))add_face(p,c,a,b);
      return true;
    }else return false;
  }
  void build(){
    bool ok=false;
    fc.clear();
    sort(pt.begin(),pt.end(),point_cmp);
```

```
    pt.resize(unique(pt.begin(),pt.end())-pt.begin());
    for(size_t i=2;i<pt.size();++i){
      if((pt[0]-pt[i]).area2(pt[1]-pt[i])!=0){
        ok=true;
        swap(pt[i],pt[2]);
        break;
      }
    }
    if(!ok)return;
    ok=false;
    for(size_t i=3;i<pt.size();++i){
      if(tetrahedron<T>(pt[0],pt[1],pt[2],pt[i]).volume6()!=0){
        ok=true;
        swap(pt[i],pt[3]);
        break;
      }
    }
    if(!ok)return;
    for(int i=0;i<4;++i)add_face(i,(i+1)%4,(i+2)%4,(i+3)%4);
    for(size_t i=4;i<pt.size();++i){
      for(int j=fc.size()-1;j>=0;--j){
        if(outside(i,j)){
          dfs(i,j);
          break;
        }
      }
    }
    size_t sz=0;
    for(size_t i=0;i<fc.size();++i)if(fc[i].use)fc[sz++]=fc[i];
    fc.resize(sz);
  }
  point3D<T> centroid()const{
    point3D<T> res(0,0,0);
    T vol=0;
    for(size_t i=0;i<fc.size();++i){
      T tmp=pt[fc[i].a].dot(pt[fc[i].b].cross(pt[fc[i].c]));
      res=res+(pt[fc[i].a]+pt[fc[i].b]+pt[fc[i].c])*tmp;
      vol+=tmp;
    }
    return res/(vol*4);
  }
};
```

1.3 SmallestCircle.cpp

```
#include"Geometry.cpp"
#include<vector>
struct Circle{
    typedef point<double> p;
    typedef const point<double> cp;
    p x;
    double r2;
    bool incircle(cp &c)const{return (x-c).abs2()<=r2;}
};

Circle TwoPointCircle(Circle::cp &a, Circle::cp &b) {
    Circle::p m=(a+b)/2;
    return (Circle){m,(a-m).abs2()};
}

Circle outcircle(Circle::p a, Circle::p b, Circle::p c) {
    if(TwoPointCircle(a,b).incircle(c))
        return TwoPointCircle(a,b);
    if(TwoPointCircle(b,c).incircle(a))
        return TwoPointCircle(b,c);
    if(TwoPointCircle(c,a).incircle(b))
        return TwoPointCircle(c,a);
    Circle::p ret;
    double a1=b.x-a.x, b1=b.y-a.y, c1=(a1*a1+b1*b1)/2;
    double a2=c.x-a.x, b2=c.y-a.y, c2=(a2*a2+b2*b2)/2;
    double d = a1*b2 - a2*b1;
    ret.x=a.x+(c1*b2-c2*b1)/d;
    ret.y=a.y+(a1*c2-a2*c1)/d;
    return (Circle){ret,(ret-a).abs2()};
}
//rand required
Circle SmallestCircle(std::vector<Circle::p> &p){
    int n=p.size();
    if(n==1) return (Circle){p[0],0.0};
    if(n==2) return TwoPointCircle(p[0],p[1]);
    random_shuffle(p.begin(),p.end());
    Circle c = {p[0],0.0};
    for(int i=0;i<n;++i){
        if(c.incircle(p[i])) continue;
        c=Circle{p[i],0.0};
        for(int j=0;j<i;++j){
            if(c.incircle(p[j])) continue;
            c=TwoPointCircle(p[i],p[j]);
            for(int k=0;k<j;++k){
                if(c.incircle(p[k]))
                    continue;
                c=outcircle(p[i],p[j],p[k]);
            }
        }
    }
    return c;
}
```

1.4 最近點對.cpp

```
#define INF LLONG_MAX/*預設是long long最大值*/
template<typename T>
T closest_pair(vector<point<T> >&v,vector<point<T> >&t,int l,int r){
  T dis=INF,tmd;
  if(l>=r)return dis;
  int mid=(l+r)/2;
  if((tmd=closest_pair(v,t,l,mid))<dis)dis=tmd;
  if((tmd=closest_pair(v,t,mid+1,r))<dis)dis=tmd;
  t.clear();
  for(int i=l;i<=r;++i)
    if((v[i].x-v[mid].x)*(v[i].x-v[mid].x)<dis)t.push_back(v[i]);
  sort(t.begin(),t.end(),point<T>::y_cmp);/*如果用merge_sort的方式可以O(n)*/
  for(int i=0;i<(int)t.size();++i)
    for(int j=1;j<=3&&i+j<(int)t.size();++j)
      if((tmd=(t[i]-t[i+j]).abs2())<dis)dis=tmd;
  return dis;
}
template<typename T>
inline T closest_pair(vector<point<T> > &v){
  vector<point<T> >t;
  sort(v.begin(),v.end(),point<T>::x_cmp);
  return closest_pair(v,t,0,v.size()-1);/*最近點對距離*/
}
```

1.5 浮點數誤差模板.cpp

```
const double EPS=1e-9;
struct Double{
  double d;
  Double(double d=0):d(d){}
  bool operator <(const Double &b)const{return d-b.d<-EPS;}
  bool operator >(const Double &b)const{return d-b.d>EPS;}
  bool operator ==(const Double &b)const{return fabs(d-b.d)<=EPS;}
  bool operator !=(const Double &b)const{return fabs(d-b.d)>EPS;}
  bool operator <=(const Double &b)const{return d-b.d<=EPS;}
  bool operator >=(const Double &b)const{return d-b.d>=-EPS;}
  operator double()const{return d;}
};
```

2 Data_Structure

2.1 DLX.cpp

```
#define MAXN 4100
#define MAXM 1030
#define MAXND 16390
struct DLX{
  int n,m,sz,ansd;//高是n，寬是m的稀疏矩陣
  int S[MAXM],H[MAXN];
  int row[MAXND],col[MAXND];//每個節點代表的列跟行
  int L[MAXND],R[MAXND],U[MAXND],D[MAXND];
  vector<int> ans,anst;
  void init(int _n,int _m){
    n=_n,m=_m;
    for(int i=0;i<=m;++i){
      U[i]=D[i]=i,L[i]=i-1,R[i]=i+1;
      S[i]=0;
    }
    R[m]=0,L[0]=m;
    sz=m,ansd=INT_MAX;//ansd存最優解的個數
    for(int i=1;i<=n;++i)H[i]=-1;
  }
  void add(int r,int c){
    ++S[col[++sz]=c];
    row[sz]=r;
    D[sz]=D[c],U[D[c]]=sz,U[sz]=c,D[c]=sz;
    if(H[r]<0)H[r]=L[sz]=R[sz]=sz;
    else R[sz]=R[H[r]],L[R[H[r]]]=sz,L[sz]=H[r],R[H[r]]=sz;
  }
  #define DFOR(i,A,s) for(int i=A[s];i!=s;i=A[i])
  void remove(int c){//刪除第c行和所有當前覆蓋到第c行的列
    L[R[c]]=L[c],R[L[c]]=R[c];//這裡刪除第c行，若有些行不需要處理可以在開始時呼叫他
    DFOR(i,D,c)DFOR(j,R,i){U[D[j]]=U[j],D[U[j]]=D[j],--S[col[j]];}
  }
  void restore(int c){//恢復第c行和所有當前覆蓋到第c行的列，remove的逆操作
    DFOR(i,U,c)DFOR(j,L,i){++S[col[j]],U[D[j]]=j,D[U[j]]=j;}
    L[R[c]]=c,R[L[c]]=c;
  }
  void remove2(int nd){//刪除nd所在的行當前所有點(包括虛擬節點)，只保留nd
    DFOR(i,D,nd)L[R[i]]=L[i],R[L[i]]=R[i];
  }
  void restore2(int nd){//刪除nd所在的行當前所有點，為remove2的逆操作
    DFOR(i,U,nd)L[R[i]]=R[L[i]]=i;
  }
  bool vis[MAXM];
  int h(){//估價函數 for IDA*
    int res=0;
    memset(vis,0,sizeof(vis));
    DFOR(i,R,0)if(!vis[i]){
      vis[i]=1;
      ++res;
      DFOR(j,D,i)DFOR(k,R,j)vis[col[k]]=1;
    }
    return res;
  }
  bool dfs(int d){//for精確覆蓋問題
    if(d+h()>=ansd)return 0;//找最佳解用，找任意解可以刪掉
    if(!R[0]){ansd=d;return 1;}
    int c=R[0];
    DFOR(i,R,0)if(S[i]<S[c])c=i;
    remove(c);
    DFOR(i,D,c){
      ans.push_back(row[i]);
      DFOR(j,R,i)remove(col[j]);
      if(dfs(d+1))return 1;
      ans.pop_back();
```

```cpp
      DFOR(j,L,i)restore(col[j]);
    }
    restore(c);
    return 0;
  }
  void dfs2(int d){//for最小重複覆蓋問題
    if(d+h()>=ansd)return;
    if(!R[0]){ansd=d;ans=anst;return;}
    int c=R[0];
    DFOR(i,R,0)if(S[i]<S[c])c=i;
    DFOR(i,D,c){
      anst.push_back(row[i]);
      remove2(i);
      DFOR(j,R,i)remove2(j),--S[col[j]];
      dfs2(d+1);
      anst.pop_back();
      DFOR(j,L,i)restore2(j),++S[col[j]];
      restore2(i);
    }
  }
  bool exact_cover(){//解精確覆蓋問題
    ans.clear();//答案存在ans裡
    return dfs(0);
  }
  void min_cover(){//解最小覆蓋問題
    anst.clear();//這只是暫存用，答案還是存在ans裡
    dfs2(0);
  }
  #undef DFOR
};
```

2.2 Dynamic_KD_tree.cpp

```cpp
template<typename T,size_t kd>//kdª??³´XƒŸ?«?
class kd_tree{
  public:
    struct point{
      T d[kd];
      T dist(const point &x)const{
        T ret=0;
        for(size_t i=0;i<kd;++i)ret+=std::abs(d[i]-x.d[i]);
        return ret;
      }
      bool operator==(const point &p){
        for(size_t i=0;i<kd;++i)
          if(d[i]!=p.d[i])return 0;
        return 1;
      }
      bool operator<(const point &b)const{
        return d[0]<b.d[0];
      }
    };
  private:
    struct node{
      node *l,*r;
      point pid;
      int s;
      node(const point &p):l(0),r(0),pid(p),s(1){}
      ~node(){delete l,delete r;}
      void up(){s=(l?l->s:0)+1+(r?r->s:0);}
    }*root;
    const double alpha,loga;
    const T INF;//°O±onµ¹INF¡Aª??¥¤j?
    int maxn;
    struct __cmp{
      int sort_id;
      bool operator()(const node*x,const node*y)const{
        return operator()(x->pid,y->pid);
      }
      bool operator()(const point &x,const point &y)const{
        if(x.d[sort_id]!=y.d[sort_id])
          return x.d[sort_id]<y.d[sort_id];
        for(size_t i=0;i<kd;++i)
          if(x.d[i]!=y.d[i])return x.d[i]<y.d[i];
        return 0;
      }
    }cmp;
    int size(node *o){return o?o->s:0;}
    std::vector<node*> A;
    node* build(int k,int l,int r){
      if(l>r)return 0;
      if(k==kd)k=0;
      int mid=(l+r)/2;
      cmp.sort_id=k;
      std::nth_element(A.begin()+l,A.begin()+mid,A.begin()+r+1,cmp);
      node *ret=A[mid];
      ret->l=build(k+1,l,mid-1);
      ret->r=build(k+1,mid+1,r);
      ret->up();
      return ret;
    }
    bool isbad(node*o){
      return size(o->l)>alpha*o->s||size(o->r)>alpha*o->s;
    }
    void flatten(node *u,typename std::vector<node*>::iterator &it){
      if(!u)return;
      flatten(u->l,it);
      *it=u;
      flatten(u->r,++it);
    }
    void rebuild(node*&u,int k){
      if((int)A.size()<u->s)A.resize(u->s);
      typename std::vector<node*>::iterator it=A.begin();
      flatten(u,it);
      u=build(k,0,u->s-1);
    }
    bool insert(node*&u,int k,const point &x,int dep){
      if(!u){
        u=new node(x);
        return dep<=0;
      }
      ++u->s;
      cmp.sort_id=k;
      if(insert(cmp(x,u->pid)?u->l:u->r,(k+1)%kd,x,dep-1)){
        if(!isbad(u))return 1;
        rebuild(u,k);
      }
      return 0;
    }
    node *findmin(node*o,int k){
      if(!o)return 0;
      if(cmp.sort_id==k)return o->l?findmin(o->l,(k+1)%kd):o;
      node *l=findmin(o->l,(k+1)%kd);
      node *r=findmin(o->r,(k+1)%kd);
      if(l&&!r)return cmp(l,o)?l:o;
      if(!l&&r)return cmp(r,o)?r:o;
      if(!l&&!r)return o;
      if(cmp(l,r))return cmp(l,o)?l:o;
      return cmp(r,o)?r:o;
    }
    bool erase(node *&u,int k,const point &x){
      if(!u)return 0;
      if(u->pid==x){
        if(u->r);
        else if(u->l){
          u->r=u->l;
          u->l=0;
        }else{
          delete u;
          u=0;
          return 1;
        }
        --u->s;
        cmp.sort_id=k;
        u->pid=findmin(u->r,(k+1)%kd)->pid;
        return erase(u->r,(k+1)%kd,u->pid);
      }
      cmp.sort_id=k;
      if(erase(cmp(x,u->pid)?u->l:u->r,(k+1)%kd,x)){
        --u->s;return 1;
      }else return 0;
    }
    T heuristic(const T h[])const{
      T ret=0;
      for(size_t i=0;i<kd;++i)ret+=h[i];
      return ret;
    }
    int qM;
    std::priority_queue<std::pair<T,point > >pQ;
    void nearest(node *u,int k,const point &x,T *h,T &mndist){
      if(u==0||heuristic(h)>=mndist)return;
      T dist=u->pid.dist(x),old=h[k];
      /*mndist=std::min(mndist,dist);*/
      if(dist<mndist){
        pQ.push(std::make_pair(dist,u->pid));
        if((int)pQ.size()==qM+1)
          mndist=pQ.top().first,pQ.pop();
      }
      if(x.d[k]<u->pid.d[k]){
        nearest(u->l,(k+1)%kd,x,h,mndist);
        h[k]=std::abs(x.d[k]-u->pid.d[k]);
        nearest(u->r,(k+1)%kd,x,h,mndist);
      }else{
        nearest(u->r,(k+1)%kd,x,h,mndist);
        h[k]=std::abs(x.d[k]-u->pid.d[k]);
        nearest(u->l,(k+1)%kd,x,h,mndist);
      }
      h[k]=old;
    }
    std::vector<point>in_range;
    void range(node *u,int k,const point&mi,const point&ma){
      if(!u)return;
      bool is=1;
      for(int i=0;i<kd;++i)
        if(u->pid.d[i]<mi.d[i]||ma.d[i]<u->pid.d[i]){
          is=0;break;
        }
      if(is)in_range.push_back(u->pid);
      if(mi.d[k]<=u->pid.d[k])range(u->l,(k+1)%kd,mi,ma);
      if(ma.d[k]>=u->pid.d[k])range(u->r,(k+1)%kd,mi,ma);
    }
  public:
    kd_tree(const T &INF,double a=0.75):root(0),alpha(a),loga(log2(1.0/a)),INF(INF),maxn(1){}
    ~kd_tree(){delete root;}
    void clear(){delete root,root=0,maxn=1;}
    void build(int n,const point *p){
      delete root,A.resize(maxn=n);
      for(int i=0;i<n;++i)A[i]=new node(p[i]);
      root=build(0,0,n-1);
    }
    void insert(const point &x){
      insert(root,0,x,__lg(size(root))/loga);
      if(root->s>maxn)maxn=root->s;
    }
    bool erase(const point &p){
      bool d=erase(root,0,p);
      if(root&&root->s<alpha*maxn)rebuild();
      return d;
    }
    void rebuild(){
      if(root)rebuild(root,0);
      maxn=root->s;
    }
    T nearest(const point &x,int k){
      qM=k;
      T mndist=INF,h[kd]={};
      nearest(root,0,x,h,mndist);
      mndist=pQ.top().first;
      pQ=std::priority_queue<std::pair<T,point > >();
      return mndist;/*¦^¶???x²?kª??Iª º¶Z??*/
    }
    const std::vector<point> &range(const point&mi,const point&ma){
      in_range.clear();
      range(root,0,mi,ma);
      return in_range;/*¦^¶G¶©?mi¨?ma¤§¶¡ªº?Ivector*/
    }
    int size(){return root?root->s:0;}
};
```

2.3 kd_tree_replace_segment_tree.cpp

```cpp
/*kd樹代替高維線段樹*/
struct node{
  node *l,*r;
  point pid,mi,ma;
  int s;
  int data;
  node(const point &p,int d):l(0),r(0),pid(p
      ),mi(p),ma(p),s(1),data(d),dmin(d),
      dmax(d){}
  void up(){
    mi=ma=pid;
    s=1;
    if(l){
      for(int i=0;i<kd;++i){
        mi.d[i]=min(mi.d[i],l->mi.d[i]);
        ma.d[i]=max(ma.d[i],l->ma.d[i]);
      }
      s+=l->s;
    }
    if(r){
      for(int i=0;i<kd;++i){
        mi.d[i]=min(mi.d[i],r->mi.d[i]);
        ma.d[i]=max(ma.d[i],r->ma.d[i]);
      }
      s+=r->s;
    }
  }
  void up2(){
    //其他懶惰標記向上更新
  }
  void down(){
    //其他懶惰標記下推
  }
}*root;

/*檢查區間包含用的函數*/
inline bool range_include(node *o,const
    point &L,const point &R){
  for(int i=0;i<kd;++i){
    if(L.d[i]>o->ma.d[i]||R.d[i]<o->mi.d[i])
        return 0;
  }//只要(L,R)區間有和o的區間有交集就回傳
      true
  return 1;
}
inline bool range_in_range(node *o,const
    point &L,const point &R){
  for(int i=0;i<kd;++i){
    if(L.d[i]>o->mi.d[i]||o->ma.d[i]>R.d[i])
        return 0;
  }//如果(L,R)區間完全包含o的區間就回傳true
  return 1;
}
inline bool point_in_range(node *o,const
    point &L,const point &R){
  for(int i=0;i<kd;++i){
    if(L.d[i]>o->pid.d[i]||R.d[i]<o->pid.d[i
        ])return 0;
  }//如果(L,R)區間完全包含o->pid這個點就回傳
      true
  return 1;
}
/*單點修改，以單點改值為例*/
void update(node *u,const point &x,int data,
    int k=0){
  if(!u)return;
  u->down();
  if(u->pid==x){
    u->data=data;
    u->up2();
    return;
  }
  cmp.sort_id=k;
  update(cmp(x,u->pid)?u->l:u->r,x,data,(k
      +1)%kd);
  u->up2();
}

/*區間修改*/
void update(node *o,const point &L,const
    point &R,int data){
  if(!o)return;
  o->down();
  if(range_in_range(o,L,R)){
    //區間懶惰標記修改
    o->down();
    return;
  }
  if(point_in_range(o,L,R)){
    //這個點在(L,R)區間，但是他的左右子樹不
        一定在區間中
    //單點懶惰標記修改
  }
  if(o->l&&range_include(o->l,L,R))update(o
      ->l,L,R,data);
  if(o->r&&range_include(o->r,L,R))update(o
      ->r,L,R,data);
  o->up2();
}

/*區間查詢，以總和為例*/
int query(node *o,const point &L,const point
    &R){
  if(!o)return 0;
  o->down();
  if(range_in_range(o,L,R))return o->sum;
  int ans=0;
  if(point_in_range(o,L,R))ans+=o->data;
  if(o->l&&range_include(o->l,L,R))ans+=
      query(o->l,L,R);
  if(o->r&&range_include(o->r,L,R))ans+=
      query(o->r,L,R);
  return ans;
}
```

2.4 persistent_segment_tree.cpp

```cpp
#include<bits/stdc++.h>//POJ 2104
using namespace std;
struct node{
  int l,r;
  int data;
  node(int l,int r,int d):l(l),r(r),data(d)
      {}
};
vector<node> nds;
inline void up(int o,int l,int r){
  nds[o].data=nds[l].data+nds[r].data;
}
inline int new_node(int l,int r,int d){
  nds.push_back(node(l,r,d));
  return nds.size()-1;
}
inline int new_node(const node &nd){
  nds.push_back(nd);
  return nds.size()-1;
}
int build_tree(int l,int r){
  int nd=new_node(-1,-1,0);
  if(l==r)return nd;
  int mid=(l+r)/2;
  int L=build_tree(l,mid);//執行時vector會被
      重構
  int R=build_tree(mid+1,r);//一定要這樣寫
  nds[nd].l=L;
  nds[nd].r=R;
  //up(nd,L,R);
  return nd;
}
int insert(int l,int r,int rt,int x,int d){
  if(x<l||r<x)return rt;
  int nd=new_node(nds[rt]);
  if(l==r&&l==x)nds[nd].data+=d;
  else{
    int mid=(l+r)/2;
    int L=insert(l,mid,nds[nd].l,x,d);
    int R=insert(mid+1,r,nds[nd].r,x,d);
    nds[nd].l=L;
    nds[nd].r=R;
    up(nd,L,R);
  }
  return nd;
}
inline int cal(int L,int R){
  return nds[R].data-nds[L].data;
}
int find(int l,int r,int L,int R,int k){
  if(l==r)return l;
  int mid=(l+r)/2;
  int add=cal(nds[L].l,nds[R].l);
  if(k<=add)return find(l,mid,nds[L].l,nds[R
      ].l,k);
  return find(mid+1,r,nds[L].r,nds[R].r,k-
      add);
}
int n,m;
int s[100005];
int root[100005];
int main(){
  while(~scanf("%d%d",&n,&m)){
    nds.clear();
    vector<int> lsh;
    for(int i=1;i<=n;++i){
      scanf("%d",&s[i]);
      lsh.push_back(s[i]);
    }
    sort(lsh.begin(),lsh.end());
    lsh.resize(unique(lsh.begin(),lsh.end())
        -lsh.begin());
    int N=(int)lsh.size()-1;
    root[0]=build_tree(0,N);
    for(int i=1;i<=n;++i){
      s[i]=lower_bound(lsh.begin(),lsh.end()
          ,s[i])-lsh.begin();
      root[i]=insert(0,N,root[i-1],s[i],1);
    }
    while(m--){
      int a,b,k;
      scanf("%d%d%d",&a,&b,&k);
      int res=find(0,N,root[a-1],root[b],k);
      printf("%d\n",lsh[res]);
    }
  }
  return 0;
}
```

2.5 reference_point.cpp

```cpp
#include<bits/stdc++.h>
using namespace std;
template<typename T>
struct _RefCounter{
  T data;
  int ref;
  _RefCounter(const T&d=0):data(d),ref(0){}
};
template<typename T>
struct reference_pointer{
  _RefCounter<T> *p;
  T *operator->(){return &(*p).data;}
  T &operator*(){return p->data;}
  operator int(){return(int)(long long)p;}
  reference_pointer&operator=(const
      reference_pointer &t){
    if(p&&--(*p).ref==0)delete p;
    p=t.p;
    p&&++(*p).ref;
    return*this;
  }
  reference_pointer(_RefCounter<T> *t=0):p(t
      ){
    p&&++(*p).ref;
  }
  reference_pointer(const reference_pointer
      &t):p(t.p){
    p&&++(*p).ref;
  }
  ~reference_pointer(){
    if(p&&--(*p).ref==0)delete p;
  }
};
template<typename T>
inline const reference_pointer<T>
    new_reference(const T&nd){
  return reference_pointer<T>(new
      _RefCounter<T>(nd));
}
struct P{
  int a,b;
  P(int A,int B):a(A),b(B){}
```

```
}p(2,3);
int main(){
  reference_pointer<int >b=new_reference(int
      (5));
  reference_pointer<int >a=new_reference(*b)
      ;
  reference_pointer<P >c=new_reference(p);
  return 0;
}
```

2.6 skew_heap.cpp

```
template<typename T,typename _Compare=std::
    less<T> >
class skew_heap{
  private:
    struct node{
      T data;
      node *l,*r;
      node(const T&d):data(d),l(0),r(0){}
      ~node(){delete l,delete r;}
    }*root;
    int _size;
    _Compare cmp;
    node *merge(node *a,node *b){
      if(!a||!b)return a?a:b;
      if(cmp(a->data,b->data))return merge(b
          ,a);
      node *t=a->r;
      a->r=a->l;
      a->l=merge(b,t);
      return a;
    }
  public:
    skew_heap():root(0),_size(0){}
    ~skew_heap(){delete root;}
    void clear(){delete root,root=0,_size
        =0;}
    void join(skew_heap &o){
      root=merge(root,o.root);
      o.root=0;
      _size+=o._size;
      o._size=0;
    }
    void swap(skew_heap &o){
      node *t=root;
      root=o.root;
      o.root=t;
      int st=_size;
      _size=o._size;
      o._size=st;
    }
    void push(const T&data){
      _size++;
      root=merge(root,new node(data));
    }
    void pop(){
      if(_size)_size--;
      node *tmd=merge(root->l,root->r);
      root->l=root->r=0;
      delete root;
      root=tmd;
    }
    const T& top(){return root->data;}
    int size(){return _size;}
    bool empty(){return !_size;}
};
```

2.7 split_merge.cpp

```
void split(node *o,node *&a,node *&b,int k){
  if(!o)a=b=0;
  else{
    //o=new node(*o);
    o->down();
    if(k<=size(o->l)){
      b=o;
      split(o->l,a,b->l,k);
    }else{
      a=o;
      split(o->r,a->r,b,k-size(o->l)-1);
    }
    o->up();
  }
}
node *merge(node *a,node *b){
  if(!a||!b)return a?a:b;
  static int x;
  if(x++%(a->s+b->s)<a->s){
    //a=new node(*a);
    a->down();
    a->r=merge(a->r,b);
    a->up();
    return a;
  }else{
    //b=new node(*b);
    b->down();
    b->l=merge(a,b->l);
    b->up();
    return b;
  }
}
```

2.8 treap.cpp

```
template<typename T>
class treap{
  private:
    struct node{
      T data;
      unsigned fix;
      int s;
      node *ch[2];
      node(const T&d):data(d),s(1){}
      node():s(0){ch[0]=ch[1]=this;}
    }*nil,*root;
    unsigned x;
    unsigned ran(){return x=x*0xdefaced+1;}
    void rotate(node *&a,bool d){
      node *b=a;
      a=a->ch[!d];
      a->s=b->s;
      b->ch[!d]=a->ch[d];
      a->ch[d]=b;
      b->s=b->ch[0]->s+b->ch[1]->s+1;
    }
    void insert(node *&o,const T &data){
      if(!o->s){
        o=new node(data),o->fix=ran();
        o->ch[0]=o->ch[1]=nil;
      }else{
        o->s++;
        bool d=o->data<data;
        insert(o->ch[d],data);
        if(o->ch[d]->fix>o->fix)rotate(o,!d)
            ;
      }
    }
    node *merge(node *a,node *b){
      if(!a->s||!b->s)return a->s?a:b;
      if(a->fix>b->fix){
        a->ch[1]=merge(a->ch[1],b);
        a->s=a->ch[0]->s+a->ch[1]->s+1;
        return a;
      }else{
        b->ch[0]=merge(a,b->ch[0]);
        b->s=b->ch[0]->s+b->ch[1]->s+1;
        return b;
      }
    }
    bool erase(node *&o,const T &data){
      if(!o->s)return 0;
      if(o->data==data){
        node *t=o;
        o=merge(o->ch[0],o->ch[1]);
        delete t;
        return 1;
      }
      if(erase(o->ch[o->data<data],data)){
        o->s--;return 1;
      }else return 0;
    }
    void clear(node *&o){
      if(o->s)clear(o->ch[0]),clear(o->ch
          [1]),delete o;
    }
  public:
    treap(unsigned s=20150119):nil(new node)
        ,root(nil),x(s){}
    ~treap(){clear(root),delete nil;}
    void clear(){clear(root),root=nil;}
    void insert(const T &data){
      insert(root,data);
    }
    bool erase(const T &data){
      return erase(root,data);
    }
    bool find(const T&data){
      for(node *o=root;o->s;)
      if(o->data==data)return 1;
      else o=o->ch[o->data<data];
      return 0;
    }
    int rank(const T&data){
      int cnt=0;
      for(node *o=root;o->s;)
      if(o->data<data)cnt+=o->ch[0]->s+1,o=o
          ->ch[1];
      else o=o->ch[0];
      return cnt;
    }
    const T&kth(int k){
      for(node *o=root;;)
      if(k<=o->ch[0]->s)o=o->ch[0];
      else if(k==o->ch[0]->s+1)return o->
          data;
      else k-=o->ch[0]->s+1,o=o->ch[1];
    }
    const T&operator[](int k){
      return kth(k);
    }
    const T&preorder(const T&data){
      node *x=root,*y=0;
      while(x->s)
      if(x->data<data)y=x,x=x->ch[1];
      else x=x->ch[0];
      if(y)return y->data;
      return data;
    }
    const T&successor(const T&data){
      node *x=root,*y=0;
      while(x->s)
      if(data<x->data)y=x,x=x->ch[0];
      else x=x->ch[1];
      if(y)return y->data;
      return data;
    }
    int size(){return root->s;}
};
```

2.9 操作分治.cpp

```
void dq(int l,int r){
  if(l==r)return;
  int mid=(l+r)/2;
  dq(l,mid);
  處理[l,mid]的操作對[mid+1,r]的影響
  dq(mid+1,r);
}
```

2.10 整體二分.cpp

```
void BS(int l,int r,vector<Item> &vs){
  //答案該<l會有的已經做完了
  if(l==r)整個vs的答案=l;//??????
  int mid=(l+r)/2;
  do_thing(l,mid);//做答案<=mid會做的事
  vector<Item> left=vs裡滿足的;
  vector<Item> right=vs-left;
  undo_thing(l,mid);
  BS(l,mid,left);
  do_thing(l,mid);
  BS(mid+1,r,right);//??????
}
```

3 default

3.1 debug.cpp

```cpp
#ifdef Jinkela
#define debug(...) {\
  fprintf(stderr,"%s - %d : (%s) = ",
       __PRETTY_FUNCTION__,__LINE__,#
       __VA_ARGS__);\
  _DO(__VA_ARGS__);\
}
template<typename I> void _DO(I&&x){cerr<<x
     <<endl;}
template<typename I,typename...T> void _DO(I
     &&x,T&&...tail){cerr<<x<<", ";_DO(tail
     ...);}
#else
#define debug(...)
#endif
```

3.2 IncStack.cpp

```cpp
//Magic
#pragma GCC optimize "Ofast"
//stack resize,change esp to rsp if 64-bit
     system
asm("mov %0,%%esp\n" ::"g"(mem+10000000));
//linux stack resize
#include<sys/resource.h>
void increase_stack(){
  const rlim_t ks=64*1024*1024;
  struct rlimit rl;
  int res=getrlimit(RLIMIT_STACK,&rl);
  if(!res&&rl.rlim_cur<ks){
    rl.rlim_cur=ks;
    res=setrlimit(RLIMIT_STACK,&rl);
  }
}
```

3.3 input.cpp

```cpp
inline int read(){
    int x=0; bool f=0; char c=getchar();
    while(ch<'0'||'9'<ch)f|=ch=='-',ch=
         getchar();
    while('0'<=ch&&ch<='9')x=x*10-'0'+ch,ch=
         getchar();
    return f?-x:x;
}
//g++ -std=c++11 -O2 -Wall -Wextra -Wno-
     unused-variable $1 && ./a.out
```

4 Flow

4.1 dinic.cpp

```cpp
#define MAXN 105
#define INF INT_MAX
int n;/*number of nodes*/
int level[MAXN],cur[MAXN];/*layer, current
     arc*/
struct edge{
  int v,pre;
  long long cap,flow,r;
  edge(int v,int pre,long long cap):v(v),pre
       (pre),cap(cap),flow(0),r(cap){}
};
int g[MAXN];
std::vector<edge> e;
inline void init(){
  memset(g,-1,sizeof(int)*(n+1));
  e.clear();
}
inline void add_edge(int u,int v,long long
     cap,bool directed=false){
  e.push_back(edge(v,g[u],cap));
  g[u]=e.size()-1;
  e.push_back(edge(u,g[v],directed?0:cap));
  g[v]=e.size()-1;
}
inline int bfs(int s,int t){
  memset(level,0,sizeof(int)*(n+1));
  memcpy(cur,g,sizeof(int)*(n+1));
  std::queue<int >q;
  q.push(s);
  level[s]=1;
  while(q.size()){
    int u=q.front();q.pop();
    for(int i=g[u];~i;i=e[i].pre){
      if(!level[e[i].v]&&e[i].r){
        level[e[i].v]=level[u]+1;
        q.push(e[i].v);
        if(e[i].v==t)return 1;
      }
    }
  }
  return 0;
}
long long dfs(int u,int t,long long cur_flow
     =INF){
  if(u==t||!cur_flow)return cur_flow;
  long long df,tf=0;
  for(int &i=cur[u];~i;i=e[i].pre){
    if(level[e[i].v]==level[u]+1&&e[i].r){
      if(df=dfs(e[i].v,t,std::min(cur_flow,e
           [i].r))){
        e[i].flow+=df;
        e[i^1].flow-=df;
        e[i].r-=df;
        e[i^1].r+=df;
        tf+=df;
        if(!(cur_flow-=df))break;
      }
    }
  }
  if(!df)level[u]=0;
  return tf;
}
inline long long dinic(int s,int t,bool
     clean=true){
  if(clean){
    for(size_t i=0;i<e.size();++i){
      e[i].flow=0;
      e[i].r=e[i].cap;
    }
  }
  long long ans=0;
  while(bfs(s,t))ans+=dfs(s,t);
  return ans;
}
```

4.2 ISAP.cpp

```cpp
#define MAXN 105
#define INF INT_MAX
int n;/*點數*/
int d[MAXN],gap[MAXN],cur[MAXN];
/*層次、gap[i]=層次為i的點之個數、當前弧優化
     */
struct edge{
  int v,pre;
  long long cap,flow,r;
  edge(int v,int pre,long long cap):v(v),pre
       (pre),cap(cap),flow(0),r(cap){}
};
int g[MAXN];
std::vector<edge> e;
inline void init(){
  memset(g,-1,sizeof(int)*(n+1));
  e.clear();
}
inline void add_edge(int u,int v,long long
     cap,bool directed=false){
  e.push_back(edge(v,g[u],cap));
  g[u]=e.size()-1;
  e.push_back(edge(u,g[v],directed?0:cap));
  g[v]=e.size()-1;
}
long long dfs(int u,int s,int t,long long
     cur_flow=INF){
  if(u==t)return cur_flow;
  long long tf=cur_flow,df;
  for(int &i=cur[u];~i;i=e[i].pre){
    if(e[i].r&&d[u]==d[e[i].v]+1){
      df=dfs(e[i].v,s,t,std::min(tf,e[i].r))
          ;
      e[i].flow+=df;
      e[i^1].flow-=df;
      e[i].r-=df;
      e[i^1].r+=df;
      if(!(tf-=df)||d[s]==n)return cur_flow-
           tf;
    }
  }
  int minh=n;
  for(int i=cur[u]=g[u];~i;i=e[i].pre){
    if(e[i].r&&d[e[i].v]<minh)minh=d[e[i].v
         ];
  }
  if(!--gap[d[u]])d[s]=n;
  else ++gap[d[u]=++minh];
  return cur_flow-tf;
}
inline long long isap(int s,int t,bool clean
     =true){
  memset(d,0,sizeof(int)*(n+1));
  memset(gap,0,sizeof(int)*(n+1));
  memcpy(cur,g,sizeof(int)*(n+1));
  if(clean){
    for(size_t i=0;i<e.size();++i){
      e[i].flow=0;
      e[i].r=e[i].cap;
    }
  }
  long long max_flow=0;
  for(gap[0]=n;d[s]<n;)max_flow+=dfs(s,s,t);
  return max_flow;
}
```

4.3 MinCostMaxFlow.cpp

```cpp
#define MAXN 440
#define INF 999999999
struct edge{
  int v,pre;
  int cap,cost;
  edge(int v,int pre,int cap,int cost):v(v),
       pre(pre),cap(cap),cost(cost){}
};
int n,S,T;
int dis[MAXN],piS,ans;
bool vis[MAXN];
std::vector<edge> e;
int g[MAXN];
inline void init(){
  memset(g,-1,sizeof(int)*n);
  e.clear();
}
inline void add_edge(int u,int v,int cost,
     int cap,bool directed=false){
  e.push_back(edge(v,g[u],cap,cost));
  g[u]=e.size()-1;
  e.push_back(edge(u,g[v],directed?0:cap,-
       cost));
  g[v]=e.size()-1;
}
int augment(int u,int cur_flow){
  if(u==T||!cur_flow)return ans+=piS*
       cur_flow,cur_flow;
  vis[u]=1;
  int r=cur_flow,d;
  for(int i=g[u];~i;i=e[i].pre){
    if(e[i].cap&&!e[i].cost&&!vis[e[i].v]){
      d=augment(e[i].v,std::min(r,e[i].cap))
          ;
      e[i].cap-=d;
      e[i^1].cap+=d;
      if(!(r-=d))break;
    }
  }
  return cur_flow-r;
```

```
}
inline bool modlabel(){
  for(int i=0;i<n;++i)dis[i]=INF;
  dis[T]=0;
  static std::deque<int>q;
  q.push_back(T);
  while(q.size()){
    int u=q.front();
    q.pop_front();
    int dt;
    for(int i=g[u];~i;i=e[i].pre){
      if(e[i^1].cap&&(dt=dis[u]-e[i].cost)<
          dis[e[i].v]){
        if((dis[e[i].v]=dt)<=dis[q.size()?q.
            front():S]){
          q.push_front(e[i].v);
        }else q.push_back(e[i].v);
      }
    }
  }
  for(int u=0;u<n;++u){
    for(int i=g[u];~i;i=e[i].pre){
      e[i].cost+=dis[e[i].v]-dis[u];
    }
  }
  piS+=dis[S];
  return dis[S]<INF;
}
inline int mincost(){
  piS=ans=0;
  while(modlabel()){
    do memset(vis,0,sizeof(bool)*n);
    while(augment(S,INF));
  }
  return ans;
}
```

5 Graph

5.1 Arborescence_EV.cpp

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

struct node {
    int from, to, cost;
    node(int from=0,int to=0,int cost=0):
        from(from),to(to),cost(cost){};
} edge[M];

int m, n, m, c;
int far[N], In[N], ID[N], vis[N];

bool MST(int cost,int n,int root)
{
    long long int ans=0;
    while(true)
    {
        for(int i=0;i<n;++i) IN[i].first =
            INF;
        for(int i=0;i<m;++i)
            if(edge[i].from!=edge[i].to)
                IN[edge[i].to] = min(IN[edge
                    [i].to],make_pair(edge[i
                    ].cost,edge[i].from));
        for(int i=0;i<n;++i)
            if(i!=root && IN[i].first==INF)
                return false; // NO
                    Arborescence

        int cntnode = 0;
        memset(ID,-1,sizeof(ID));
        memset(vis,-1,sizeof(vis));
        In[root] = 0;
        for(int i=0;i<n;++i) ans += IN[i].
            first;
        for(int i=0;i<n;++i) {
            int x;
            for(x=i;vis[x]!=i&&ID[x]==-1&&x
                !=root;x=IN[x].second)
                vis[x] = i;
            if(ID[x]==-1 && x!=root) {
                for(int i=IN[x].second;u!=x;
                    u=IN[u].second)
                    ID[u] = cntnode;
                ++cntnode;
            }
        }
        if(cntnode==0)  break; // END

        for(int i=0;i<n;++i)
            if(ID[i]==-1)
                ID[i] = cntnode++;

        for(int i=0;i<m;++i) {
            int v = edge[i].to;
            edge[i].from = ID[edge[i].from];
            edge[i].to = ID[edge[i].to];
            if(edge[i].from!=edge[i].to)
                edge[i].cost -= IN[edge[i].
                    to].first;
        }
        n=cntnode;
        root=ID[root];
    }
    return ans<=cost;
}
```

5.2 Augmenting_Path.cpp

```
#define MAXN1 505
#define MAXN2 505
int n1,n2;/*n1個點連向n2個點*/
int match[MAXN2];/*每個屬於n2的點匹配了哪個
    點*/
vector<int > g[MAXN1];/*圖*/
bool vis[MAXN2];/*是否走訪過*/
bool dfs(int u){
  for(size_t i=0;i<g[u].size();++i){
    int v=g[u][i];
    if(vis[v])continue;
    vis[v]=1;
    if(match[v]==-1||dfs(match[v])){
      match[v]=u;
      return 1;
    }
  }
  return 0;
}
inline int max_match(){
  int ans=0;
  memset(match,-1,sizeof(int)*n2);
  for(int i=0;i<n1;++i){
    memset(vis,0,sizeof(bool)*n2);
    if(dfs(i))++ans;
  }
  return ans;
}
```

5.3 Augmenting_Path_multiple.c

```
#define MAXN1 1005
#define MAXN2 505
int n1,n2;//n1個點連向n2個點，其中n2個點可以
    匹配很多邊
vector<int > g[MAXN1];//圖
int c[MAXN2];//每個屬於n2點最多可以接受幾條
    匹配邊
vector<int> match_list[MAXN2];//每個屬於n2的
    點匹配了那些點
bool vis[MAXN2];//是否走訪過
bool dfs(int u){
  for(size_t i=0;i<g[u].size();++i){
    int v=g[u][i];
    if(vis[v])continue;
    vis[v]=true;
    if((int)match_list[v].size()<c[v]){
      match_list[v].push_back(u);
      return true;
    }else{
      for(size_t j=0;j<match_list[v].size()
          ;++j){
        int next_u=match_list[v][j];
        if(dfs(next_u)){
          match_list[v][j]=u;
          return true;
        }
      }
    }
  }
  return false;
}
inline int max_match(){
  for(int i=0;i<n2;++i)match_list[i].clear()
      ;
  int cnt=0;
  for(int u=0;u<n1;++u){
    memset(vis,0,sizeof(bool)*n2);
    if(dfs(u))++cnt;
  }
  return cnt;
}
```

5.4 blossom_matching.cpp

```
#define MAXN 505
vector<int>g[MAXN];
int pa[MAXN],match[MAXN],st[MAXN],S[MAXN],v[
    MAXN];
int t,n;
inline int lca(int x,int y){
  for(++t;;swap(x,y)){
    if(x==0)continue;
    if(v[x]==t)return x;
    v[x]=t;
    x=st[pa[match[x]]];
  }
}
#define qpush(x) q.push(x),S[x]=0
inline void flower(int x,int y,int l,queue<
    int> &q){
  while(st[x]!=l){
    pa[x]=y;
    if(S[y=match[x]]==1)qpush(y);
    st[x]=st[y]=l,x=pa[y];
  }
}
inline bool bfs(int x){
  for(int i=1;i<=n;++i)st[i]=i;
  memset(S+1,-1,sizeof(int)*n);
  queue<int>q;qpush(x);
  while(q.size()){
    x=q.front(),q.pop();
    for(size_t i=0;i<g[x].size();++i){
      int y=g[x][i];
      if(S[y]==-1){
        pa[y]=x,S[y]=1;
        if(!match[y]){
          for(int lst;x;y=lst,x=pa[y])
            lst=match[x],match[x]=y,match[y
                ]=x;
          return 1;
        }
        qpush(match[y]);
      }else if(!S[y]&&st[y]!=st[x]){
        int l=lca(y,x);
        flower(y,x,l,q),flower(x,y,l,q);
      }
    }
  }
  return 0;
}
inline int blossom(){
  int ans=0;
  for(int i=1;i<=n;++i)
    if(!match[i]&&bfs(i))++ans;
  return ans;
}
```

5.5 formula.tex

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 I(G)+Cv(G)=|V| M(G)+Ce(G)=|V|
對於連通二分圖 I(G)=Cv(G) M(G)=Ce(G)
最大團 = 補圖的最大獨立集

5.6 graphISO.cpp

```cpp
const int MAXN=1005,K=30;//K要夠大
const long long A=3,B=11,C=2,D=19,P=0
    xdefaced;
long long f[K+1][MAXN];
vector<int> g[MAXN],rg[MAXN];
int n;
inline void init(){
  for(int i=0;i<n;++i){
    f[0][i]=1;
    g[i].clear();
    rg[i].clear();
  }
}
inline void add_edge(int u,int v){
  g[u].push_back(v);
  rg[v].push_back(u);
}
inline long long point_hash(int u){//O(N)
  for(int t=1;t<=K;++t){
    for(int i=0;i<n;++i){
      f[t][i]=f[t-1][i]*A%P;
      for(int j:g[i])f[t][i]=(f[t][i]+f[t
          -1][j]*B%P)%P;
      for(int j:rg[i])f[t][i]=(f[t][i]+f[t
          -1][j]*C%P)%P;
      if(i==u)f[t][i]+=D;//如果圖太大的話，
          把這行刪掉，執行一次後f[K]就會是所
          有點的答案
      f[t][i]%=P;
    }
  }
  return f[K][u];
}
inline vector<long long> graph_hash(){
  vector<long long> ans;
  for(int i=0;i<n;++i)ans.push_back(
      point_hash(i));//O(N^2)
  sort(ans.begin(),ans.end());
  return ans;
}
```

5.7 KM.cpp

```cpp
#define MAXN 100
int n;
int g[MAXN][MAXN],lx[MAXN],ly[MAXN],slack_y[
    MAXN];
int match_y[MAXN];
bool vx[MAXN],vy[MAXN];//要保證g是完全二分圖
bool dfs(int x,bool adjust=1){//DFS找增廣
    路，is=1表示要交換邊
  if(vx[x])return 0;
  vx[x]=1;
  for(int y=0;y<n;++y){
    if(vy[y])continue;
    int t=lx[x]+ly[y]-g[x][y];
    if(t==0){
      vy[y]=1;
      if(match_y[y]==-1||dfs(match_y[y],
          adjust)){
        if(adjust)match_y[y]=x;
        return 1;
      }
    }else if(slack_y[y]>t)slack_y[y]=t;
  }
  return 0;
}
inline int km(){
  memset(ly,0,sizeof(int)*n);
  memset(match_y,-1,sizeof(int)*n);
  for(int x=0;x<n;++x){
    lx[x]=0;
    for(int y=0;y<n;++y){
      lx[x]=max(lx[x],g[x][y]);
    }
  }
  for(int x=0;x<n;++x){
    for(int y=0;y<n;++y)slack_y[y]=INT_MAX;
    memset(vx,0,sizeof(bool)*n);
    memset(vy,0,sizeof(bool)*n);
    if(dfs(x))continue;
    bool flag=1;
    while(flag){
      int cut=INT_MAX;
      for(int y=0;y<n;++y){
        if(!vy[y]&&cut>slack_y[y])cut=
            slack_y[y];
      }
      for(int j=0;j<n;++j){
        if(vx[j])lx[j]-=cut;
        if(vy[j])ly[j]+=cut;
        else slack_y[j]-=cut;
      }
      for(int y=0;y<n;++y){
        if(!vy[y]&&slack_y[y]==0){
          vy[y]=1;
          if(match_y[y]==-1||dfs(match_y[y
              ],0)){
            flag=0;//測試成功，有增廣路
            break;
          }
        }
      }
    }
    memset(vx,0,sizeof(bool)*n);
    memset(vy,0,sizeof(bool)*n);
    dfs(x);//最後要記得將邊翻反轉
  }
  int ans=0;
  for(int y=0;y<n;++y)ans+=g[match_y[y]][y];
  return ans;
}
```

5.8 MaximumClique.cpp

```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;
  }
};
```

5.9 Minimum_General_Weighted_M

```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++)
      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;
```

5.10 Rectilinear_Steiner_tree.c

```cpp
//平面曼哈頓最小生成樹構造圖(去除非必要邊)
#include<vector>
#include<algorithm>
#define T int
```

```cpp
#define INF 0x3f3f3f3f
struct point{
  T x,y;
  int id;//每個點的編號都要不一樣，從0開始編號
  point(){}
  T dist(const point &p)const{
    return std::abs(x-p.x)+std::abs(y-p.y);
  }
};
inline bool cmpx(const point &a,const point &b){
  return a.x<b.x||(a.x==b.x&&a.y<b.y);
}
struct edge{
  int u,v;
  T cost;
  edge(int u,int v,const T&c):u(u),v(v),cost(c){}
  bool operator<(const edge&e)const{
    return cost<e.cost;
  }
};
struct bit_node{
  T mi;
  int id;
  bit_node(const T&mi=INF,int id=-1):mi(mi),id(id){}
};
std::vector<bit_node> bit;
inline void bit_update(int i,const T&data,int id){
  for(;i;i-=i&(-i)){
    if(data<bit[i].mi)bit[i]=bit_node(data,id);
  }
}
inline int bit_find(int i,int m){
  bit_node x;
  for(;i<=m;i+=i&(-i)){
    if(bit[i].mi<x.mi)x=bit[i];
  }
  return x.id;
}
inline std::vector<edge> build_graph(int n,point p[]){
  std::vector<edge> e;//回傳的邊就可以用來求最小生成樹
  for(int dir=0;dir<4;++dir){//4種座標變換
    if(dir%2){
      for(int i=0;i<n;++i)std::swap(p[i].x,p[i].y);
    }else if(dir==2){
      for(int i=0;i<n;++i)p[i].x=-p[i].x;
    }
    std::sort(p,p+n,cmpx);
    std::vector<T>ga(n),gb;
    for(int i=0;i<n;++i)ga[i]=p[i].y-p[i].x;
    gb=ga;
    std::sort(gb.begin(),gb.end());
    gb.resize(std::unique(gb.begin(),gb.end())-gb.begin());
    int m=gb.size();
    bit=std::vector<bit_node>(m+1);
    for(int i=n-1;i>=0;--i){
      int pos=std::lower_bound(gb.begin(),gb.end(),ga[i])-gb.begin()+1;
      int ans=bit_find(pos,m);
      if(~ans)e.push_back(edge(p[i].id,p[ans].id,p[i].dist(p[ans])));
      bit_update(pos,p[i].x+p[i].y,i);
    }
  }
  return e;
}
```

5.11 treeISO.cpp

```cpp
const int MAXN=100005;
const long long X=12327,P=0xdefaced;
vector<int> g[MAXN];
bool vis[MAXN];
long long dfs(int u){
  vis[u]=1;
  vector<long long> tmp;
  for(auto v:g[u])if(!vis[v])tmp.push_back(dfs(v));
  if(tmp.empty())return 177;
  long long ret=4931;
  sort(tmp.begin(),tmp.end());
  for(auto v:tmp)ret=((ret*X)^v)%P;
  return ret;
}
```

5.12 一般圖最大權匹配.cpp

```cpp
#include<bits/stdc++.h>
using namespace std;
#define INF INT_MAX
#define MAXN 400
struct edge{
  int u,v,w;
  edge(){}
  edge(int u,int v,int w):u(u),v(v),w(w){}
};
int n,n_x;
edge g[MAXN*2+1][MAXN*2+1];
int lab[MAXN*2+1];
int match[MAXN*2+1],slack[MAXN*2+1],st[MAXN*2+1],pa[MAXN*2+1];
int flower_from[MAXN*2+1][MAXN+1],S[MAXN*2+1],vis[MAXN*2+1];
vector<int> flower[MAXN*2+1];
queue<int> q;
inline int e_delta(const edge &e){ // does not work inside blossoms
  return lab[e.u]+lab[e.v]-g[e.u][e.v].w*2;
}
inline void update_slack(int u,int x){
  if(!slack[x]||e_delta(g[u][x])<e_delta(g[slack[x]][x]))slack[x]=u;
}
inline void set_slack(int x){
  slack[x]=0;
  for(int u=1;u<=n;++u)
    if(g[u][x].w>0&&st[u]!=x&&S[st[u]]==0)update_slack(u,x);
}
void q_push(int x){
  if(x<=n)q.push(x);
  else for(size_t i=0;i<flower[x].size();i++)q_push(flower[x][i]);
}
inline void set_st(int x,int b){
  st[x]=b;
  if(x>n)for(size_t i=0;i<flower[x].size();++i)
    set_st(flower[x][i],b);
}
inline int get_pr(int b,int xr){
  int pr=find(flower[b].begin(),flower[b].end(),xr)-flower[b].begin();
  if(pr%2==1){//檢查他在前一層圖是奇點還是偶點
    reverse(flower[b].begin()+1,flower[b].end());
    return (int)flower[b].size()-pr;
  }else return pr;
}
inline void set_match(int u,int v){
  match[u]=g[u][v].v;
  if(u>n){
    edge e=g[u][v];
    int xr=flower_from[u][e.u],pr=get_pr(u,xr);
    for(int i=0;i<pr;++i)set_match(flower[u][i],flower[u][i^1]);
    set_match(xr,v);
    rotate(flower[u].begin(),flower[u].begin()+pr,flower[u].end());
  }
}
inline void augment(int u,int v){
  for(;;){
    int xnv=st[match[u]];
    set_match(u,v);
    if(!xnv)return;
    set_match(xnv,st[pa[xnv]]);
    u=st[pa[xnv]],v=xnv;
  }
}
inline int get_lca(int u,int v){
  static int t=0;
  for(++t;u||v;swap(u,v)){
    if(u==0)continue;
    if(vis[u]==t)return u;
    vis[u]=t;//這種方法可以不用清空v陣列
    u=st[match[u]];
    if(u)u=st[pa[u]];
  }
  return 0;
}
inline void add_blossom(int u,int lca,int v){
  int b=n+1;
  while(b<=n_x&&st[b])++b;
  if(b>n_x)++n_x;
  lab[b]=0,S[b]=0;
  match[b]=match[lca];
  flower[b].clear();
  flower[b].push_back(lca);
  for(int x=u,y;x!=lca;x=st[pa[y]])
    flower[b].push_back(x),flower[b].push_back(y=st[match[x]]),q_push(y);
  reverse(flower[b].begin()+1,flower[b].end());
  for(int x=v,y;x!=lca;x=st[pa[y]])
    flower[b].push_back(x),flower[b].push_back(y=st[match[x]]),q_push(y);
  set_st(b,b);
  for(int x=1;x<=n_x;++x)g[b][x].w=g[x][b].w=0;
  for(int x=1;x<=n;++x)flower_from[b][x]=0;
  for(size_t i=0;i<flower[b].size();++i){
    int xs=flower[b][i];
    for(int x=1;x<=n_x;++x)
      if(g[b][x].w==0||e_delta(g[xs][x])<e_delta(g[b][x]))
        g[b][x]=g[xs][x],g[x][b]=g[x][xs];
    for(int x=1;x<=n;++x)
      if(flower_from[xs][x])flower_from[b][x]=xs;
  }
  set_slack(b);
}
inline void expand_blossom(int b){ // S[b] == 1
  for(size_t i=0;i<flower[b].size();++i)
    set_st(flower[b][i],flower[b][i]);
  int xr=flower_from[b][g[b][pa[b]].u],pr=get_pr(b,xr);
  for(int i=0;i<pr;i+=2){
    int xs=flower[b][i],xns=flower[b][i+1];
    pa[xs]=g[xns][xs].u;
    S[xs]=1,S[xns]=0;
    slack[xs]=0,set_slack(xns);
    q_push(xns);
  }
  S[xr]=1,pa[xr]=pa[b];
  for(size_t i=pr+1;i<flower[b].size();++i){
    int xs=flower[b][i];
    S[xs]=-1,set_slack(xs);
  }
  st[b]=0;
}
inline bool on_found_edge(const edge &e){
  int u=st[e.u],v=st[e.v];
  if(S[v]==-1){
    pa[v]=e.u,S[v]=1;
    int nu=st[match[v]];
    slack[v]=slack[nu]=0;
    S[nu]=0,q_push(nu);
  }else if(S[v]==0){
    int lca=get_lca(u,v);
    if(!lca){
      augment(u,v),augment(v,u);
      return true;
    }else add_blossom(u,lca,v);
  }
  return false;
}
inline bool matching(){
  memset(S+1,-1,sizeof(int)*n_x);
  memset(slack+1,0,sizeof(int)*n_x);
  q=queue<int>();
  for(int x=1;x<=n_x;++x)
```

```
    if(st[x]==x&&!match[x])pa[x]=0,S[x]=0,q_push(x);
  if(q.empty())return false;
  for(;;){
    while(q.size()){
      int u=q.front();q.pop();
      if(S[st[u]]==1)continue;
      for(int v=1;v<=n;++v)
        if(g[u][v].w>0&&st[u]!=st[v]){
          if(e_delta(g[u][v])==0){
            if(on_found_edge(g[u][v]))return true;
          }else update_slack(u,st[v]);
        }
    }
    int d=INF;
    for(int b=n+1;b<=n_x;++b)
      if(st[b]==b&&S[b]==1)d=min(d,lab[b]/2);
    for(int x=1;x<=n_x;++x)
      if(st[x]==x&&slack[x]){
        if(S[x]==-1)d=min(d,e_delta(g[slack[x]][x]));
        else if(S[x]==0)d=min(d,e_delta(g[slack[x]][x])/2);
      }
    for(int u=1;u<=n;++u){
      if(S[st[u]]==0){
        if(lab[u]<=d)return 0;
        lab[u]-=d;
      }else if(S[st[u]]==1)lab[u]+=d;
    }
    for(int b=n+1;b<=n_x;++b)
      if(st[b]==b){
        if(S[st[b]]==0)lab[b]+=d*2;
        else if(S[st[b]]==1)lab[b]-=d*2;
      }
    q=queue<int>();
    for(int x=1;x<=n_x;++x)
      if(st[x]==x&&slack[x]&&st[slack[x]]!=x&&e_delta(g[slack[x]][x])==0)
        if(on_found_edge(g[slack[x]][x]))return true;
    for(int b=n+1;b<=n_x;++b)
      if(st[b]==b&&S[b]==1&&lab[b]==0)expand_blossom(b);
  }
  return false;
}
inline pair<long long,int> weight_blossom(){
  memset(match+1,0,sizeof(int)*n);
  n_x=n;
  int n_matches=0;
  long long tot_weight=0;
  for(int u=0;u<=n;++u)st[u]=u,flower[u].clear();
  int w_max=0;
  for(int u=1;u<=n;++u)
    for(int v=1;v<=n;++v){
      flower_from[u][v]=(u==v?u:0);
      w_max=max(w_max,g[u][v].w);
    }
  for(int u=1;u<=n;++u)lab[u]=w_max;
  while(matching())++n_matches;
  for(int u=1;u<=n;++u)
    if(match[u]&&match[u]<u)
      tot_weight+=g[u][match[u]].w;
  return make_pair(tot_weight,n_matches);
}
inline void init_weight_graph(){
  for(int u=1;u<=n;++u)
    for(int v=1;v<=n;++v)
      g[u][v]=edge(u,v,0);
}
int main(){
  int m;
  scanf("%d%d",&n,&m);
  init_weight_graph();
  for(int i=0;i<m;++i){
    int u,v,w;
    scanf("%d%d%d",&u,&v,&w);
    g[u][v].w=g[v][u].w=w;
  }
  printf("%lld\n",weight_blossom().first);
  for(int u=1;u<=n;++u)printf("%d ",match[u]);puts("");
  return 0;
}
```

6 language

6.1 CNF.cpp

```
#define MAXN 55
struct CNF{
  int s,x,y;//s->xy | s->x, if y==-1
  int cost;
  CNF(){}
  CNF(int s,int x,int y,int c):s(s),x(x),y(y),cost(c){}
};
int state;//規則數量
map<char,int> rule;//每個字元對應到的規則，小寫字母為終端字符
vector<CNF> cnf;
inline void init(){
  state=0;
  rule.clear();
  cnf.clear();
}
inline void add_to_cnf(char s,const string &p,int cost){
  //加入一個s -> <p>的文法，代價為cost
  if(rule.find(s)==rule.end())rule[s]=state++;
  for(auto c:p)if(rule.find(c)==rule.end())rule[c]=state++;
  if(p.size()==1){
    cnf.push_back(CNF(rule[s],rule[p[0]],-1,cost));
  }else{
    int left=rule[s];
    int sz=p.size();
    for(int i=0;i<sz-2;++i){
      cnf.push_back(CNF(left,rule[p[i]],state,0));
      left=state++;
    }
    cnf.push_back(CNF(left,rule[p[sz-2]],rule[p[sz-1]],cost));
  }
}
vector<long long> dp[MAXN][MAXN];
vector<bool> neg_INF[MAXN][MAXN];//如果花費是負的可能會有無限小的情形
inline void relax(int l,int r,const CNF &c,long long cost,bool neg_c=0){
  if(!neg_INF[l][r][c.s]&&(neg_INF[l][r][c.x]||cost<dp[l][r][c.s])){
    if(neg_c||neg_INF[l][r][c.x]){
      dp[l][r][c.s]=0;
      neg_INF[l][r][c.s]=true;
    }else dp[l][r][c.s]=cost;
  }
}
inline void bellman(int l,int r,int n){
  for(int k=1;k<=state;++k)
    for(auto c:cnf)
      if(c.y==-1)relax(l,r,c,dp[l][r][c.x]+c.cost,k==n);
}
inline void cyk(const vector<int> &tok){
  for(int i=0;i<(int)tok.size();++i){
    for(int j=0;j<(int)tok.size();++j){
      dp[i][j]=vector<long long>(state+1,INT_MAX);
      neg_INF[i][j]=vector<bool>(state+1,false);
    }
    dp[i][i][tok[i]]=0;
    bellman(i,i,tok.size());
  }
  for(int r=1;r<(int)tok.size();++r){
    for(int l=r-1;l>=0;--l){
      for(int k=l;k<r;++k)
        for(auto c:cnf)
          if(~c.y)relax(l,r,c,dp[l][k][c.x]+dp[k+1][r][c.y]+c.cost);
      bellman(l,r,tok.size());
    }
  }
}
```

6.2 earley.cpp

```
struct Rule{
  vector<vector<Rule*> > p;
  void add(const vector<Rule*> &l){
    p.push_back(l);
  }
};
map<string,Rule*> NameRule;
map<Rule*,string> RuleName;
inline void init_Rule(){
  for(auto r:RuleName)delete r.first;
  RuleName.clear();
  NameRule.clear();
}
inline Rule *add_rule(const string &s){
  if(NameRule.find(s)!=NameRule.end())return NameRule[s];
  Rule *r=new Rule();
  RuleName[r]=s;
  NameRule[s]=r;
  return r;
}
typedef vector<Rule*> production;
struct State{
  Rule *r;
  int rid,dot_id,start,end;
  State(Rule *r,int rid,int dot,int start):r(r),rid(rid),dot_id(dot),start(start),end(-1){}
  State(Rule *r=0,int col=0):r(r),rid(-1),dot_id(-1),start(-1),end(col){}
  bool completed()const{
    return rid==-1||dot_id>=(int)r->p[rid].size();
  }
  Rule *next_term()const{
    if(completed())return 0;
    return r->p[rid][dot_id];
  }
  bool operator<(const State& b)const{
    if(start!=b.start)return start<b.start;
    if(dot_id!=b.dot_id)return dot_id<b.dot_id;
    if(r!=b.r)return r<b.r;
    return rid<b.rid;
  }
  void print()const{
    cout<<RuleName[r]<<"->";
    if(rid!=-1)for(size_t i=0;;++i){
      if((int)i==dot_id)cout<<" "<<"$";
      if(i>=r->p[rid].size())break;
      cout<<" "<<RuleName[r->p[rid][i]];
    }
    cout<<" "<<"["<<start<<","<<end<<"]"<<endl;
  }
};
struct Column{
  Rule *term;
  string value;
  vector<State> s;
  map<State,set<pair<State,State> > > div;
  //div比較像一棵 左兄右子的樹
  Column(Rule *r,const string &s):term(r),value(s){}
  Column(){}
  bool add(const State &st,int col){
    if(div.find(st)==div.end()){
      div[st];
      s.push_back(st);
      s.back().end=col;
      return true;
    }else return false;
  }
};
inline vector<Column> lexer(string text){
  //tokenize，要自己寫，以下為範例
  //他會把 input stream 變成 token stream，就是(terminal,value)pair
```

```
  vector<Column> token;
  replace(text.begin(),text.end(),',',' ');
  stringstream ss(text);
  while(ss>>text){
    if(text=="a"||text=="of")continue;
    if(text=="list"){
      token.push_back(Column(NameRule["("],"("));
    }else if(text=="and"){
      token.push_back(Column(NameRule[")"],")"));
    }else token.push_back(Column(NameRule["T"],text));
  }
  return token;
}
vector<Column> table;
inline void predict(int col,Rule *rul){
  for(size_t i=0;i<rul->p.size();++i){
    table[col].add(State(rul,i,0,col),col);
  }
}
inline void scan(int col,const State &s,Rule *r){
  if(r!=table[col].term)return;
  State ns(s.r,s.rid,s.dot_id+1,s.start);
  table[col].add(ns,col);
  table[col].div[ns].insert(make_pair(s,State(r,col)));
}
inline void complete(int col,const State &s){
  for(size_t i=0;i<table[s.start].s.size();++i){
    State &st=table[s.start].s[i];
    Rule *term=st.next_term();
    if(!term||term->p.size()==0)continue;
    if(term==s.r){
      State nst(st.r,st.rid,st.dot_id+1,st.start);
      table[col].add(nst,col);
      table[col].div[nst].insert(make_pair(st,s));
    }
  }
}
inline pair<bool,State> parse(Rule *GAMMA,const vector<Column > &token){
  table.resize(token.size()+1);
  for(size_t i=0;i<token.size();++i)table[i+1]=Column(token[i]);
  table[0]=Column();
  table[0].add(State(GAMMA,0,0,0),0);
  for(size_t i=0;i<table.size();++i){
    for(size_t j=0;j<table[i].s.size();++j){
      State state=table[i].s[j];
      if(state.completed())complete(i,state);
      else{
        Rule *term=state.next_term();
        if(term->p.size())predict(i,term);
        else if(i+1<table.size())scan(i+1,state,term);
      }
    }
  }
  for(size_t i=0;i<table.back().s.size();++i){
    if(table.back().s[i].r==GAMMA&&table.back().s[i].completed()){
      return make_pair(true,table.back().s[i]);
    }
  }
  return make_pair(false,State(0,-1));
}
struct node{//語法樹的節點
  State s;
  vector<vector<node*> > child;//vector<node*>.size()>1表示ambiguous
  node(const State &s):s(s){}
  node(){}
};
struct State_end_cmp{
  bool operator()(const State &a,const State &b)const{
    return a.end<b.end||(a.end==b.end&&a<b);
  }
};
map<State,node*,State_end_cmp> cache;
vector<node*> node_set;
inline void init_cache(){
  for(auto d:node_set)delete d;
  cache.clear();
  node_set.clear();
}
void _build_tree(const State &s,node *pa,bool amb=0){
  if(cache.find(s)!=cache.end()){
    pa->child.push_back(vector<node*>(1,cache[s]));
    return;
  }
  node *o;
  if(s.completed()){
    o=new node(s);
    if(amb)pa->child.back().push_back(o);
    else pa->child.push_back(vector<node*>(1,o));
  }else o=pa->child.back().back();
  amb=0;
  for(auto div:table[s.end].div[s]){
    if(!amb)_build_tree(div.first,pa);
    _build_tree(div.second,o,amb);
    amb=1;
  }
  if(s.completed())cache[s]=o;
}
inline node *build_tree(const State &s){
  init_cache();
  node o;
  _build_tree(s,&o);
  assert(o.child.size()==1);
  assert(o.child.back().size()==1);
  return o.child.back().back();
}
void print_tree(node *o,int dep=0){
  cout<<string(dep,' '),o->s.print();
  for(auto div:o->child){
    for(auto nd:div){
      print_tree(nd,dep+2);
    }
  }
}
//開始寫code:以下為加入語法的範例
inline Rule *get_my_Rule(){
  Rule *S=add_rule("S"),*E=add_rule("E"),*L=add_rule("L");
  Rule *list=add_rule("("),*AND=add_rule(")"),*T=add_rule("T");
  S->add({list,E});
  S->add({list,L});
  L->add({E,L});
  L->add({E,AND,E});
  E->add({T});
  E->add({S});
  Rule *GAMMA=add_rule("GAMMA");//一定要有gamma rule當作是最上層的語法
  GAMMA->add({S});
  return GAMMA;
}
```

7 Number_Theory

7.1 basic.cpp

```
typedef long long int LL;
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);
}

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;
}

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;
}
```

7.2 bit_set.cpp

```
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;
  }
}
```

7.3 cantor_expansion.cpp

```
#include<bits/stdc++.h>
using namespace std;
#define MAXN 11
int factorial[MAXN];
inline void init(){
  factorial[0]=1;
  for(int i=1;i<=MAXN;++i)factorial[i]=
      factorial[i-1]*i;
}
inline int encode(const std::vector<int> &s)
    {
  int n=s.size(),res=0;
  for(int i=0;i<n;++i){
    int t=0;
    for(int j=i+1;j<n;++j)
      if(s[j]<s[i])++t;
    res+=t*factorial[n-i-1];
  }
  return res;
}
inline std::vector<int> decode(int a,int n){
  std::vector<int> res;
  std::vector<bool> vis(n,0);
  for(int i=n-1;i>=0;--i){
    int t=a/factorial[i],j;
    for(j=0;j<n;++j)
      if(!vis[j]){
        if(t==0)break;
        --t;
      }
    res.push_back(j);
    vis[j]=1;
    a%=factorial[i];
  }
  return res;
}
int main(){
  init();
  vector<int> p={0,1,2,3,4,5,6,7,8};
  for(int i=0;i<factorial[9];++i){
    vector<int> s=decode(i,9);
    if(s!=p)puts("XX");
    next_permutation(p.begin(),p.end());
  }
  return 0;
}
```

7.4 Chinese_remainder_theorem.

```
#include<bits/stdc++.h>
using namespace std;
#ifndef CHINESE_REMAINDER_THEOREM
#define CHINESE_REMAINDER_THEOREM
template<typename T>
inline 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;
}
template<typename T>
inline 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>
inline T crt(std::vector<T> &m,std::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;
}
#endif
int n;
vector<long long> a,m;
int main(){
  while(~scanf("%d",&n)){
    for(int i=0;i<n;++i){
      long long x,y;
      scanf("%lld%lld",&x,&y);
      m.push_back(x);
      a.push_back(y);
    }
    long long ans=crt(m,a);
    printf("%lld\n",ans);
    for(int i=0;i<n;++i)printf("%lld %lld\n"
        ,m[i],ans%m[i]);
    m.clear();
    a.clear();
  }
  return 0;
}
/*
4
199 198
200 199
201 197
137 88
2
265163 465
66546165 7122
5
379 46
853 852
971 777
659 128
281 256
4
6359 1
4877 5
1627 6
8941 7122
*/
```

7.5 enumerate.cpp

```
void all_divdown(const LL &n) { // all n/x
    for(LL a=1;a<=n;a=n/(n/(a+1))) {
        // dosomething;
    }
}
```

7.6 eulerphi.cpp

```
int eulerPhi(int n){
  int m = sqrt(n+0.5);
  int res=n;
  for(int i=2; i<=m; i++){
    if(n%i==0){
      res = res*(i-1)/i;
      while(n%i==0)n/=i;
    }
  }
  if(n>1) res = res*(n-1)/n;
  return res;
}

vector<int> phiTable(int n){
  vector<int>phi(n+1, 0);
  phi[1] = 1;
  for(int i=2; i<=n; i++) if(!phi[i])
    for(int j=i; j<=n; j+=i){
      if(!phi[j])phi[j] = j;
      phi[j] = phi[j]*(i-1)/i;
    }
  return phi;
}
```

7.7 Factor.cpp

```
LL LLmul(LL a, LL b, const LL &mod) {
    LL ans=0;
    while(b) {
        if(b&1) {
            ans+=a;
            if(ans>=mod) ans-=mod;
        }
        a<<=1, b>>=1;
        if(a>=mod) a-=mod;
    }
    return ans;
}
inline long long mod_mul(long long a,long
     long b,long long m){
  a%=m,b%=m;
  long long y=(long long)((double)a*b/m+0.5)
      ;/* fast for m < 2^58 */
  long long r=(a*b-y*m)%m;
  return r<0?r+m:r;
}
template<typename T>
inline T pow(T a,T b,T mod){//a^b%mod
  T ans=1;
  for(;b;a=mod_mul(a,a,mod),b>>=1)
    if(b&1)ans=mod_mul(ans,a,mod);
  return ans;
}
int sprp[3]={2,7,61};//int範圍可解
int llsprp
    [7]={2,325,9375,28178,450775,9780504,1795265
    //至少unsigned long long範圍
template<typename T>
inline bool isprime(T n,int *sprp,int num){
  if(n==2)return 1;
  if(n<2||n%2==0)return 0;
  int t=0;
  T u=n-1;
  for(;u%2==0;++t)u>>=1;
  for(int i=0;i<num;++i){
    T a=sprp[i]%n;
    if(a==0||a==1||a==n-1)continue;
    T x=pow(a,u,n);
    if(x==1||x==n-1)continue;
```

```
    for(int j=0;j<t;++j){
      x=mod_mul(x,x,n);
      if(x==1)return 0;
      if(x==n-1)break;
    }
    if(x==n-1)continue;
    return 0;
  }
  return 1;
}

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);
    }
}
```

7.8 FFT.cpp

```
template<typename T,typename VT=std::vector<std::complex<T> > >
struct FFT{
  const T pi;
  FFT(const T pi=acos((T)-1)):pi(pi){}
  unsigned int bit_reverse(unsigned int 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=std::__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){
        std::complex<T> wi=exp(std::complex<T>(0,i*num*pi/mh));
        for(int j=i;j<N;j+=step){
          int k=j+mh;
          std::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;
  }
};
```

7.9 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
}
```

7.10 formula.tex

$\sum_{d|n} phi(n) = n$
$\sum_{d|n} mu(n) = (n == 1)$
$g(n) = \sum_{d|n} f(d) => f(n) = \sum_{d|n} mu(d) * g(n/d)$
$Catalannumber : (2n)!/n!/n!/(n+1)$
$HarmonicseriesH_n = ln(n) + gamma + 1/(2n) - 1/(12nn) + 1/(120nnnn)$
$gamma = 0.57721566490153286060651209008240243104215$
$i - thgraycode : i^{(i>>1)}$
$SG(A+B) = SG(A) \oplus SG(B)$

7.11 Gauss_Elimination.cpp

```
const int MAX = 300;
const double EPS = 1e-8;

double mat[MAX][MAX];
void Gauss(int n) {
  for(int i=0; i<n; i++) {
    bool ok = 0;
    for(int j=i; j<n; j++) {
      if(fabs(mat[j][i]) > EPS) {
        swap(mat[j], mat[i]);
        ok = 1;
        break;
      }
    }
    if(!ok) continue;

    double fs = mat[i][i];
    for(int j=i+1; j<n; j++) {
      double r = mat[j][i] / fs;
      for(int k=i; k<n; k++) {
        mat[j][k] -= mat[i][k] * r;
      }
    }
  }
}
```

7.12 LinearCongruence.cpp

```cpp
pair<LL,LL> LinearCongruence(LL a[],LL b[],
     LL m[],int n) {
    // a[i]*x = b[i] ( mod m[i] )
    for(int i=0;i<n;++i) {
        LL x, y, d = extgcd(a[i],m[i],x,y);
        if(b[i]%d!=0) return make_pair(-1LL
             ,0LL);
        m[i] /= d;
        b[i] = LLmul(b[i]/d,x,m[i]);
    }
    LL lastb = b[0], lastm = m[0];
    for(int i=1;i<n;++i) {
        LL x, y, d = extgcd(m[i],lastm,x,y);
        if((lastb-b[i])%d!=0) return
             make_pair(-1LL,0LL);
        lastb = LLmul((lastb-b[i])/d,x,(
             lastm/d))*m[i];
        lastm = (lastm/d)*m[i];
        lastb = (lastb+b[i])%lastm;
    }
    return make_pair(lastb<0?lastb+lastm:
         lastb,lastm);
}
```

7.13 Lucas.cpp

```cpp
int mod_fact(int n,int &e){
    e=0;
    if(n==0)return 1;
    // (n/p)! % p
    int res=mod_fact(n/P,e);
    e += n/P;
    if( (n/P) %2 == 0){// = 1
        return res*fact[n%P]%P;
    }
    // = -1
    return res*(P-fact[n%P])%P;
}
int extGCD(int a,int b,int &x,int &y){
    int d=a;
    if(b!=0){
        d=extGCD(b,a%b,y,x);
        y-= (a/b)*x;
    }else{
        x=1;y=0;
    }
    return d;
}
int modInverse(int n){
    int x,y;
    extGCD(n,P,x,y);
    return (P+x%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*modInverse(a2*a3%P)%P;
}
```

7.14 NTT.cpp

```cpp
2615053605667*(2^18)+1,3
15*(2^27)+1,31
479*(2^21)+1,3
7*17*(2^23)+1,3
3*3*211*(2^19)+1,5
25*(2^22)+1,3
template<typename T,typename VT=std::vector<
     T> >
struct NTT{
  const T P,G;
  NTT(T p=(1<<23)*7*17+1,T g=3):P(p),G(g){}
  unsigned int bit_reverse(unsigned int 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);
  }
  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;
  }
  void ntt(bool is_inv,VT &in,VT &out,int N)
       {
    int bitlen=std::__lg(N);
    for(int i=0;i<N;++i)out[bit_reverse(i,
         bitlen)]=in[i];
    for(int step=2,id=1;step<=N;step<<=1,++
         id){
      T wn=pow_mod(G,(P-1)>>id,P),wi=1,u,t;
      const int mh=step>>1;
      for(int i=0;i<mh;++i){
        for(int j=i;j<N;j+=step){
          u=out[j],t=wi*out[j+mh]%P;
          out[j]=u+t;
          out[j+mh]=u-t;
          if(out[j]>=P)out[j]-=P;
          if(out[j+mh]<0)out[j+mh]+=P;
        }
        wi=wi*wn%P;
      }
    }
    if(is_inv){
      for(int i=1;i<N/2;++i)std::swap(out[i
           ],out[N-i]);
      T invn=pow_mod(N,P-2,P);
      for(int i=0;i<N;++i)out[i]=out[i]*invn
           %P;
    }
  }
};
```

7.15 random.cpp

```cpp
inline int random_int(){
  static int seed=20160424;
  return seed+=(seed<<16)+0x1db3d743;
}
inline long long random_long_long(){
  static long long seed=20160424;
  return seed+=(seed<<32)+0xdb3d742c265539d;
}
```

7.16 外星模運算.cpp

```cpp
//a[0]^(a[1]^a[2]^...)
#include<bits/stdc++.h>
using namespace std;
#define maxn 1000000
int euler[maxn+5];
bool is_prime[maxn+5];
inline void init_euler(){
  is_prime[1]=1;//一不是質數
  for(int i=1;i<=maxn;i++)euler[i]=i;
  for(int i=2;i<=maxn;i++){
    if(!is_prime[i]){//是質數
      euler[i]--;
      for(int j=i<<1;j<=maxn;j+=i){
        is_prime[j]=1;
        euler[j]=euler[j]/i*(i-1);
      }
    }
  }
}
inline long long pow(long long a,long long b
     ,long long mod){//a^b%mod
  long long ans=1;
  for(;b;a=a*a%mod,b>>=1)
    if(b&1)ans=ans*a%mod;
  return ans;
}
bool isless(long long *a,int n,int k){
  if(*a==1)return k>1;
  if(--n==0)return *a<k;
  int next=0;
  for(long long b=1;b<k;++next)
    b*=*a;
  return isless(a+1,n,next);
}
long long high_pow(long long *a,int n,long
     long mod){
  if(*a==1||--n==0)return *a%mod;
  int k=0,r=euler[mod];
  for(long long tma=1;tma!=pow(*a,k+r,mod)
       ;++k)
    tma=tma*(*a)%mod;
  if(isless(a+1,n,k))return pow(*a,high_pow(
       a+1,n,k),mod);
  int tmd=high_pow(a+1,n,r);
  int t=(tmd-k+r)%r;
  return pow(*a,k+t,mod);
}
long long a[1000005];
int t,mod;
int main(){
  init_euler();
  scanf("%d",&t);
  #define n 4
  while(t--){
    for(int i=0;i<n;++i)scanf("%lld",&a[i]);
    scanf("%d",&mod);
    printf("%lld\n",high_pow(a,n,mod));
  }
  return 0;
}
```

7.17 模運算模板.cpp

```cpp
template<typename T,long long mod>
struct mod_t{//mod只能是質數
  T data;
  mod_t(){}
  mod_t(const T &d):data((d%mod+mod)%mod){}
  mod_t pow(T b)const{
    mod_t ans(1);
    for(mod_t now=*this;b;now=now*now,b/=2)
      if(b%2)ans=ans*now;
    return ans;
  }
  mod_t operator-(int)const{
    return mod_t(mod-data);
  }
  mod_t operator+(const mod_t &b)const{
    return mod_t((data+b.data)%mod);
  }
  mod_t operator-(const mod_t &b)const{
    return mod_t((data-b.data+mod)%mod);
  }
  mod_t operator*(const mod_t &b)const{
    return mod_t((data*b.data)%mod);
  }
  mod_t operator/(const mod_t &b)const{
    return *this*b.pow(mod-2);//*this *
         Inverse(b)
  }
  operator T()const{return data;}
  friend istream &operator>>(istream &i,
       mod_t &b){
    T d;
    i>>d;
    b=mod_t(d);
    return i;
  }
};
```

8 String

8.1 AC 自動機.cpp

```cpp
template<char L='a',char R='z'>
class ac_automaton{
  private:
    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];
      }
    }
};
```

8.2 hash.cpp

```cpp
#define MAXN 1000000
#define prime_mod 1073676287
/*prime_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)%prime_mod*/
inline void hash_init(int len,T prime=0 xdefaced){
  h_base[0]=1;
  for(int i=1;i<=len;++i){
    h[i]=(h[i-1]*prime+s[i-1])%prime_mod;
    h_base[i]=(h_base[i-1]*prime)%prime_mod;
  }
}
inline 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])%prime_mod+prime_mod)%prime_mod;
}
```

8.3 KMP.cpp

```cpp
/*產生fail function*/
inline 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)*/
inline 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;
}
```

8.4 manacher.cpp

```cpp
//原字串: asdsasdsa
//先把字串變成這樣: @a#s#d#s#a#s#d#s#a#
inline 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;
  }
}
```

8.5 minimal_string_rotation.cpp

```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);//傳回最小循環表示法起始位置
}
```

8.6 suffix_array_lcp.cpp

```cpp
#define radix_sort(x,y){\
  for(i=0;i<A;++i)c[i]=0;\
  for(i=0;i<len;++i)c[x[y[i]]]++;\
  for(i=1;i<A;++i)c[i]+=c[i-1];\
  for(i=len-1;i>=0;--i)sa[--c[x[y[i]]]]=y[i];\
}
void suffix_array(const char *s,int len,int *sa,int *rank,int *tmp,int *c){
  int A='z'+1,i,k,id,*t;
  for(i=0;i<len;++i){
    tmp[i]=i;
    rank[i]=s[i];
  }
  radix_sort(rank,tmp);
  for(k=1;id<len-1;k<<=1){
    id=0;
    for(i=len-k;i<len;++i)tmp[id++]=i;
    for(i=0;i<len;++i){
      if(sa[i]>=k)tmp[id++]=sa[i]-k;
    }
    radix_sort(rank,tmp);
    t=rank;rank=tmp;tmp=t;
    id=0;
    rank[sa[0]]=0;
    for(i=1;i<len;++i){
      if(tmp[sa[i-1]]!=tmp[sa[i]]||sa[i-1]+k>=len||tmp[sa[i-1]+k]!=tmp[sa[i]+k])++id;
```

```
      rank[sa[i]]=id;
    }
    A=id+1;
  }
}
#undef radix_sort
//h:高度數組 sa:後綴數組 rank:排名
inline 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;
}
```

8.7 Z.cpp

```
inline 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;
  }
}
```

9 Tarjan

9.1 dominator_tree.cpp

```
struct dominator_tree{
  static const int MAXN=5005;
  int n;// 1-base
  vector<int> suc[MAXN],pre[MAXN];//存圖和反
      向圖
  int fa[MAXN],dfn[MAXN],id[MAXN],Time;//for
      dfs
  int semi[MAXN],idom[MAXN];
  int anc[MAXN],best[MAXN];//disjoint set
  vector<int> dom[MAXN];//dominator_tree存這
      裡
  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;
```

9.2 tnfshb017_2_sat.cpp

```
#include<bits/stdc++.h>
using namespace std;
#define MAXN 8001
#define MAXN2 MAXN*4
#define n(X) ((X)+2*N)
vector<int>  v[MAXN2];
vector<int> rv[MAXN2];
vector<int>  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('-');
        }
        putchar('\n');
    }else puts("0");
    return 0;
}
```

9.3 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
inline void add_edge(int u,int v){
  G[u].push_back(E.size());
  E.push_back(edge(u,v));
  G[v].push_back(E.size());
  E.push_back(edge(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(size_t i=0;i<G[u].size();++i){
    int e=G[u][i];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);
  }
}
inline 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;
  }
}
```

9.4 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，割點的bcc_id沒
    意義
int st[N],top;
void dfs(int u,int pa=-1){//u當前點，pa父親
  int v,child=0;
  low[u]=vis[u]=++Time;
  st[top++]=u;
```

```
  for(size_t i=0;i<G[u].size();++i){
    if(!vis[v=G[u][i]]){
      dfs(v,u),++child;
      low[u]=min(low[u],low[v]);
      if(vis[u]<=low[v]){
        is_cut[u]=1;
        bcc[++bcc_cnt].clear();
        int t;
        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]);
  }
  if(pa==-1&&child<2)is_cut[u]=0;//u是dfs樹
      的根要特判
}
inline 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;
  }
}
```

10 Tree_problem

10.1 HeavyLight.cpp

```
#include<vector>
#define MAXN 100005
typedef std::vector<int >::iterator VIT;
int siz[MAXN],max_son[MAXN],pa[MAXN],dep[
    MAXN];
/*節點大小、大小最大的孩子、父母節點、深度*/
int link_top[MAXN],link[MAXN],cnt;
/*每個點所在鍊的鏈頭、樹鏈剖分的DFS序、時間
    戳*/
std::vector<int >G[MAXN];/*用vector存樹*/
void find_max_son(int x){
  siz[x]=1;
  max_son[x]=-1;
  for(VIT i=G[x].begin();i!=G[x].end();++i){
    if(*i==pa[x])continue;
    pa[*i]=x;
    dep[*i]=dep[x]+1;
    find_max_son(*i);
    if(max_son[x]==-1||siz[*i]>siz[max_son[x
        ]])max_son[x]=*i;
    siz[x]+=siz[*i];
  }
}
void build_link(int x,int top){
  link[x]=++cnt;/*記錄x點的時間戳*/
  link_top[x]=top;
  if(max_son[x]==-1)return;
  build_link(max_son[x],top);/*優先走訪最大
      孩子*/
  for(VIT i=G[x].begin();i!=G[x].end();++i){
    if(*i==max_son[x]||*i==pa[x])continue;
    build_link(*i,*i);
  }
}
inline int find_lca(int a,int b){
  /*求LCA，可以在過程中對區間進行處理*/
  int ta=link_top[a],tb=link_top[b];
  while(ta!=tb){
    if(dep[ta]<dep[tb]){
      std::swap(ta,tb);
      std::swap(a,b);
    }
    //這裡可以對a所在的鏈做區間處理
    //區間為(link[ta],link[a])
    ta=link_top[a=pa[ta]];
  }
  /*最後a,b會在同一條鏈，若a!=b還要在進行一
      次區間處理*/
  return dep[a]<dep[b]?a:b;
}
```

10.2 LCA.cpp

```
#define MAXN 100000
#define MAX_LOG 17
int pa[MAX_LOG+1][MAXN+5];
int dep[MAXN+5];
vector<int>G[MAXN+5];
void dfs(int x,int p){//dfs(1,-1);
  pa[0][x]=p;
  for(int i=0;i+1<MAX_LOG;++i)pa[i+1][x]=pa[
      i][pa[i][x]];
  for(auto &i:G[x]){
    if(i==p)continue;
    dep[i]=dep[x]+1;
    dfs(i,x);
  }
}
inline int jump(int x,int d){
  for(int i=0;i<d;++i)if((x>>i)&1)x=pa[k][x
      ];
  return x;
}
inline int find_lca(int a,int b){
  if(dep[a]>dep[b])swap(a,b);
  b=jump(b,dep[b]-dep[a]);
  if(a==b)return a;
  for(int i=MAX_LOG;i>=0;--i){
    if(pa[i][a]!=pa[i][b]){
      a=pa[i][a];
      b=pa[i][b];
    }
  }
  return pa[0][a];
}
```

10.3 link_cut_tree.cpp

```
#include<vector>
struct splay_tree{
  int ch[2],pa;/*子節點跟父母*/
  bool rev;/*反轉的懶惰標記*/
  splay_tree():pa(0),rev(0){ch[0]=ch[1]=0;}
};
std::vector<splay_tree> node;
/*
有的時候用vector會TLE，要注意
這邊以node[0]作為null節點
*/
inline bool isroot(int x){/*判斷是否為這棵
    splay tree的根*/
  return node[node[x].pa].ch[0]!=x&&node[
      node[x].pa].ch[1]!=x;
}
inline void down(int x){/*懶惰標記下推*/
  if(node[x].rev){
    if(node[x].ch[0])node[node[x].ch[0]].rev
        ^=1;
    if(node[x].ch[1])node[node[x].ch[1]].rev
        ^=1;
    std::swap(node[x].ch[0],node[x].ch[1]);
    node[x].rev^=1;
  }
}
void push_down(int x){/*將所有祖先的懶惰標記
    下推*/
  if(!isroot(x))push_down(node[x].pa);
  down(x);
}
inline void up(int x){}/*將子節點的資訊向上
    更新*/
inline void rotate(int x){/*旋轉，會自行判斷
    轉的方向*/
  int y=node[x].pa,z=node[y].pa,d=(node[y].
      ch[1]==x);
  node[x].pa=z;
  if(!isroot(y))node[z].ch[node[z].ch[1]==y
      ]=x;
  node[y].ch[d]=node[x].ch[d^1];
  node[node[y].ch[d]].pa=y;
  node[y].pa=x,node[x].ch[d^1]=y;
  up(y);
  up(x);
}
inline void splay(int x){/*將節點x伸展到所在
    splay tree的根*/
  push_down(x);
  while(!isroot(x)){
    int y=node[x].pa;
    if(!isroot(y)){
      int z=node[y].pa;
      if((node[z].ch[0]==y)^(node[y].ch[0]==
          x))rotate(y);
      else rotate(x);
    }
    rotate(x);
  }
}
inline int access(int x){
  int last=0;
  while(x){
    splay(x);
    node[x].ch[1]=last;
    up(x);
    last=x;
    x=node[x].pa;
  }
  return last;/*回傳access後splay tree的根*/
}
inline void access(int x,bool is=0){/*is=0就
    是一般的access*/
  int last=0;
  while(x){
    splay(x);
    if(is&&!node[x].pa){
      //printf("%d\n",max(node[last].ma,node
          [node[x].ch[1]].ma));
    }
    node[x].ch[1]=last;
    up(x);
    last=x;
    x=node[x].pa;
  }
}
inline void query_edge(int u,int v){
  access(u);
  access(v,1);
}
inline void make_root(int x){
  access(x),splay(x);
  node[x].rev^=1;
}
inline void make_root(int x){
  node[access(x)].rev^=1;
  splay(x);
}
inline void cut(int x,int y){
  make_root(x);
  access(y);
  splay(y);
  node[y].ch[0]=0;
  node[x].pa=0;
}
inline void cut_parents(int x){
  access(x);
  splay(x);
  node[node[x].ch[0]].pa=0;
  node[x].ch[0]=0;
}
inline void link(int x,int y){
  make_root(x);
  node[x].pa=y;
}
inline int find_root(int x){
  x=access(x);
  while(node[x].ch[0])x=node[x].ch[0];
  splay(x);
  return x;
}
inline int query(int u,int v){
/*
傳回uv路徑splay tree的根結點
這種寫法無法求LCA
*/
```

```
  make_root(u);
  return access(v);
}
inline int query_lca(int u,int v){
/*假設求鏈上點權的總和，sum是子樹的權重和，
    data是節點的權重*/
  access(u);
  int lca=access(v);
  splay(u);
  if(u==lca){
    //return node[lca].data+node[node[lca].
        ch[1]].sum;
  }else{
    //return node[lca].data+node[node[lca].
        ch[1]].sum+node[u].sum;
  }
}
struct EDGE{
  int a,b,w;
}e[10005];
int n;
std::vector<std::pair<int ,int > >G[10005];
/*first表示子節點，second表示邊的編號*/
int pa[10005],edge_node[10005];
/*pa是父母節點，暫存用的，edge_node是每個編
    被存在哪個點裡面的陣列*/
inline void bfs(int root){
/*在建構的時候把每個點都設成一個splay tree，
    不會壞掉*/
  std::queue<int > q;
  for(int i=1;i<=n;++i)pa[i]=0;
  q.push(root);
  while(q.size()){
    int u=q.front();
    q.pop();
    for(int i=0;i<(int)G[u].size();++i){
      int v=G[u][i].first;
      if(v!=pa[u]){
        pa[v]=u;
        node[v].pa=u;
        node[v].data=e[G[u][i].second].w;
        edge_node[G[u][i].second]=v;
        up(v);
        q.push(v);
      }
    }
  }
}
inline void change(int x,int b){
  splay(x);
  //node[x].data=b;
  up(x);
}
```

10.4 POJ_tree.cpp

```
#include<bits/stdc++.h>
using namespace std;
#define MAXN 10005
int n,k;
vector<pair<int,int> >g[MAXN];
int size[MAXN];
bool vis[MAXN];
inline void init(){
  for(int i=0;i<=n;++i){
    g[i].clear();
    vis[i]=0;
  }
}
void get_dis(vector<int> &dis,int u,int pa,
    int d){
  dis.push_back(d);
  for(size_t i=0;i<g[u].size();++i){
    int v=g[u][i].first,w=g[u][i].second;
    if(v!=pa&&!vis[v])get_dis(dis,v,u,d+w);
  }
}
vector<int> dis;//這東西如果放在函數裡會TLE
int cal(int u,int d){
  dis.clear();
  get_dis(dis,u,-1,d);
  sort(dis.begin(),dis.end());
  int l=0,r=dis.size()-1,res=0;
  while(l<r){
    while(l<r&&dis[l]+dis[r]>k)--r;
    res+=r-(l++);
  }
  return res;
}
pair<int,int> tree_centroid(int u,int pa,
    const int sz){
  size[u]=1;//找樹重心，second是重心
  pair<int,int> res(INT_MAX,-1);
  int ma=0;
  for(size_t i=0;i<g[u].size();++i){
    int v=g[u][i].first;
    if(v==pa||vis[v])continue;
    res=min(res,tree_centroid(v,u,sz));
    size[u]+=size[v];
    ma=max(ma,size[v]);
  }
  ma=max(ma,sz-size[u]);
  return min(res,make_pair(ma,u));
}
int tree_DC(int u,int sz){
  int center=tree_centroid(u,-1,sz).second;
  int ans=cal(center,0);
  vis[center]=1;
  for(size_t i=0;i<g[center].size();++i){
    int v=g[center][i].first,w=g[center][i].
        second;
    if(vis[v])continue;
    ans-=cal(v,w);
    ans+=tree_DC(v,size[v]);
  }
  return ans;
}
int main(){
  while(scanf("%d%d",&n,&k),n||k){
    init();
    for(int i=1;i<n;++i){
      int u,v,w;
      scanf("%d%d%d",&u,&v,&w);
      g[u].push_back(make_pair(v,w));
      g[v].push_back(make_pair(u,w));
    }
    printf("%d\n",tree_DC(1,n));
  }
  return 0;
}
```

ACM ICPC Team Reference - NTHU Jinkela

Contents

1 Computational_Geometry **1**
- 1.1 formula.tex . . . 1
- 1.2 Geometry.cpp . . . 1
- 1.3 SmallestCircle.cpp . . . 3
- 1.4 最近點對.cpp . . . 3
- 1.5 浮點數誤差模板.cpp . . . 3

2 Data_Structure **3**
- 2.1 DLX.cpp . . . 3
- 2.2 Dynamic_KD_tree.cpp . . . 4
- 2.3 kd_tree_replace_segment_tree.cpp . . . 5
- 2.4 persistent_segment_tree.cpp . . . 5
- 2.5 reference_point.cpp . . . 5
- 2.6 skew_heap.cpp . . . 6
- 2.7 split_merge.cpp . . . 6
- 2.8 treap.cpp . . . 6
- 2.9 操作分治.cpp . . . 6
- 2.10 整體二分.cpp . . . 6

3 default **7**
- 3.1 debug.cpp . . . 7
- 3.2 IncStack.cpp . . . 7
- 3.3 input.cpp . . . 7

4 Flow **7**
- 4.1 dinic.cpp . . . 7
- 4.2 ISAP.cpp . . . 7
- 4.3 MinCostMaxFlow.cpp . . . 7

5 Graph **8**
- 5.1 Arborescence_EV.cpp . . . 8
- 5.2 Augmenting_Path.cpp . . . 8
- 5.3 Augmenting_Path_multiple.cpp . . . 8
- 5.4 blossom_matching.cpp . . . 8
- 5.5 formula.tex . . . 8
- 5.6 graphISO.cpp . . . 9
- 5.7 KM.cpp . . . 9
- 5.8 MaximumClique.cpp . . . 9
- 5.9 Minimum_General_Weighted_Matching.cpp . . . 9
- 5.10 Rectilinear_Steiner_tree.cpp . . . 9
- 5.11 treeISO.cpp . . . 10
- 5.12 一般圖最大權匹配.cpp . . . 10

6 language **11**
- 6.1 CNF.cpp . . . 11
- 6.2 earley.cpp . . . 11

7 Number_Theory **12**
- 7.1 basic.cpp . . . 12
- 7.2 bit_set.cpp . . . 13
- 7.3 cantor_expansion.cpp . . . 13
- 7.4 Chinese_remainder_theorem.cpp . . . 13
- 7.5 enumerate.cpp . . . 13
- 7.6 eulerphi.cpp . . . 13
- 7.7 Factor.cpp . . . 13
- 7.8 FFT.cpp . . . 14
- 7.9 find_real_root.cpp . . . 14
- 7.10 formula.tex . . . 14
- 7.11 Gauss_Elimination.cpp . . . 14
- 7.12 LinearCongruence.cpp . . . 14
- 7.13 Lucas.cpp . . . 15
- 7.14 NTT.cpp . . . 15
- 7.15 random.cpp . . . 15
- 7.16 外星模運算.cpp . . . 15
- 7.17 模運算模板.cpp . . . 15

8 String **15**
- 8.1 AC 自動機.cpp . . . 15
- 8.2 hash.cpp . . . 16
- 8.3 KMP.cpp . . . 16
- 8.4 manacher.cpp . . . 16
- 8.5 minimal_string_rotation.cpp . . . 16
- 8.6 suffix_array_lcp.cpp . . . 16
- 8.7 Z.cpp . . . 17

9 Tarjan **17**
- 9.1 dominator_tree.cpp . . . 17
- 9.2 tnfshb017_2_sat.cpp . . . 17
- 9.3 橋連通分量.cpp . . . 17
- 9.4 雙連通分量 & 割點.cpp . . . 17

10 Tree_problem **18**
- 10.1 HeavyLight.cpp . . . 18
- 10.2 LCA.cpp . . . 18
- 10.3 link_cut_tree.cpp . . . 18
- 10.4 POJ_tree.cpp . . . 19