1 Computational_Geometry

1.1 Geometry.cpp

```cpp
const double PI=atan2(0.0,-1.0);
template<typename T>
struct point{
  T x,y;
  point(){}
  point(const T&x,const T&y):x(x),y(y){}
  point operator+(const point &b)const{
    return point(x+b.x,y+b.y);}
  point operator-(const point &b)const{
    return point(x-b.x,y-b.y);}
  point operator*(const T &b)const{
    return point(x*b,y*b);}
  point operator/(const T &b)const{
    return point(x/b,y/b);}
  bool operator==(const point &b)const{
    return x==b.x&&y==b.y;}
  T dot(const point &b)const{
    return x*b.x+y*b.y;}
  T cross(const point &b)const{
    return x*b.y-y*b.x;}
  point normal()const{//求法向量
    return point(-y,x);}
  T abs2()const{//向量長度的平方
    return dot(*this);
  }
  T rad(const point &b)const{//兩向量的弧度
    return fabs(atan2(fabs(cross(b)),dot(b))
      );
  }
  T getA()const{//對x軸的弧度
    T A=atan2(y,x);//超過180度會變負的
    if(A<=-PI/2)A+=PI*2;
    return A;
  }
};
template<typename T>
struct line{
  line(){}
  point<T> p1,p2;
  T a,b,c;//ax+by+c=0
  line(const point<T>&x,const point<T>&y):p1
    (x),p2(y){}
  void pton(){//轉成一般式
    a=p1.y-p2.y;
    b=p2.x-p1.x;
    c=-a*p1.x-b*p1.y;
  }
  T 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();
  }
  T seg_dis2(const line<T> &l)const{//兩線段
    距離平方
    return min({dis2(l.p1,1),dis2(l.p2,1),l.
      dis2(p1,1),l.dis2(p2,1)});
  }
  point<T> projection(const point<T> &p)
    const{//點對直線的投影
    point<T> n=(p2-p1).normal();
    return p-n*(p-p1).dot(n)/n.abs2();
  }
  point<T> mirror(const point<T> &p)const{//
    點對直線的鏡射
    //要先呼叫pton轉成一般式
    point<T> 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-p1)*(p2-p1).
      cross(l.p2-p1)<=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{//面積
    T ans=0;
    for(int i=p.size()-1,j=0;j<(int)p.size()
      ;i=j++)
      ans+=p[i].cross(p[j]);
    return ans/2;
  }
  point<T> center_of_mass()const{//重心
    T cx=0,cy=0,w=0;
    for(int i=p.size()-1,j=0;j<(int)p.size()
      ;i=j++){
      T a=p[i].cross(p[j]);
      cx+=(p[i].x+p[j].x)*a;
      cy+=(p[i].y+p[j].y)*a;
      w+=a;
    }
    return point<T>(cx/3/w,cy/3/w);
  }
  char ahas(const point<T>& t)const{//點是否
    在簡單多邊形內，是的話回傳1、在邊上回
    傳-1、否則回傳0
    bool c=0;
    for(int i=0,j=p.size()-1;i<p.size();j=i
      ++)
      if(line<T>(p[i],p[j]).point_on_segment
        (t))return -1;
      else if((p[i].y>t.y)!=(p[j].y>t.y)&&
        t.x<(p[j].x-p[i].x)*(t.y-p[i].y)/(p[j
        ].y-p[i].y)+p[i].x)
        c=!c;
    return c;
  }
  char point_in_convex(const point<T>&x)
    const{
    int l=1,r=(int)p.size()-2;
    while(l<=r){//點是否在凸多邊形內，是的話
      回傳1、在邊上回傳-1、否則回傳0
      int mid=(l+r)/2;
      T a1=(p[mid]-p[0]).cross(x-p[0]);
      T a2=(p[mid+1]-p[0]).cross(x-p[0]);
      if(a1>=0&&a2<=0){
        T res=(p[mid+1]-p[mid]).cross(x-p[
          mid]);
        return res>0?1:(res>=0?-1:0);
      }else if(a1<0)r=mid-1;
      else l=mid+1;
    }
    return 0;
  }
  vector<T> getA()const{//凸包邊對x軸的夾角
    vector<T>res;//一定是遞增的
    for(size_t i=0;i<p.size();++i)
      res.push_back((p[(i+1)%p.size()]-p[i])
        .getA());
    return res;
  }
  bool line_intersect(const vector<T>&A,
    const line<T> &l)const{//O(logN)
    int f1=upper_bound(A.begin(),A.end(),(l.
      p1-l.p2).getA())-A.begin();
    int f2=upper_bound(A.begin(),A.end(),(l.
      p2-l.p1).getA())-A.begin();
    return l.cross_seg(line<T>(p[f1],p[f2]))
      ;
  }
  polygon cut(const line<T> &l)const{//凸包
    對直線切割，得到直線L左側的凸包
    polygon ans;
    for(int n=p.size(),i=n-1,j=0;j<n;i=j++){
      if(l.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];
```

```cpp
198    }
199    for(int i=s.size()-2,t=m+1;i>=0;--i){
200        while(m>=t&&(p[m-1]-p[m-2]).cross(s[i]-p[m-2])<=0)--m;
201        p[m++]=s[i];
202    }
203    if(s.size()>1)--m;
204    p.resize(m);
205 }
206 T diam(){//直徑
207    int n=p.size(),t=1;
208    T ans=0;p.push_back(p[0]);
209    for(int i=0;i<n;i++){
210        point<T> now=p[i+1]-p[i];
211        while(now.cross(p[t+1]-p[i])>now.cross(p[t]-p[i]))t=(t+1)%n;
212        ans=max(ans,max((p[i]-p[t]).abs2(),(p[i+1]-p[t+1]).abs2()));
213    }
214    return p.pop_back(),ans;
215 }
216 T min_cover_rectangle(){//最小覆蓋矩形
217    int n=p.size(),t=1,r=1,l;
218    if(n<3)return 0;//也可以做最小周長矩形
219    T ans=1e99;p.push_back(p[0]);
220    for(int i=0;i<n;i++){
221        point<T> now=p[i+1]-p[i];
222        while(now.cross(p[t+1]-p[i])>now.cross(p[t]-p[i]))t=(t+1)%n;
223        while(now.dot(p[r+1]-p[i])>now.dot(p[r]-p[i]))r=(r+1)%n;
224        if(!i)l=r;
225        while(now.dot(p[l+1]-p[i])<=now.dot(p[l]-p[i]))l=(l+1)%n;
226        T d=now.abs2();
227        T tmp=now.cross(p[t]-p[i])*(now.dot(p[r]-p[i])-now.dot(p[l]-p[i]))/d;
228        ans=min(ans,tmp);
229    }
230    return p.pop_back(),ans;
231 }
232 T max_triangle(){//最大內接三角形
233    int n=p.size(),a=1,b=2;
234    if(n<3)return 0;
235    T ans=0,tmp;p.push_back(p[0]);
236    for(int i=0;i<n;++i){
237        while((p[a]-p[i]).cross(p[b+1]-p[i])>(tmp=(p[a]-p[i]).cross(p[b]-p[i])))b=(b+1)%n;
238        ans=max(ans,tmp);
239        while((p[a+1]-p[i]).cross(p[b]-p[i])>(tmp=(p[a]-p[i]).cross(p[b]-p[i])))a=(a+1)%n;
240        ans=max(ans,tmp);
241    }
242    return p.pop_back(),ans/2;
243 }
244 T dis2(polygon &pl){//凸包最近距離平方
245    vector<point<T> > &P=p,&Q=pl.p;
246    int n=P.size(),m=Q.size(),l=0,r=0;
247    for(int i=0;i<n;++i)if(P[i].y<P[l].y)l=i;
248    for(int i=0;i<m;++i)if(Q[i].y<Q[r].y)r=i;
249    P.push_back(P[0]),Q.push_back(Q[0]);
250    T ans=1e99;
251    for(int i=0;i<n;++i){
252        while((P[l]-P[l+1]).cross(Q[r+1]-Q[r])<0)r=(r+1)%m;
253        ans=min(ans,line<T>(P[l],P[l+1]).seg_dis2(line<T>(Q[r],Q[r+1])));
254        l=(l+1)%n;
255    }
256    return P.pop_back(),Q.pop_back(),ans;
257 }
258 static char sign(const point<T>&t){
259    return (t.y==0?t.x:t.y)<0;
260 }
261 static bool angle_cmp(const line<T>& A, const line<T>& B){
262    point<T> a=A.p2-A.p1,b=B.p2-B.p1;
263    return sign(a)<sign(b)||(sign(a)==sign(b)&&a.cross(b)>0);
264 }
265 int halfplane_intersection(vector<line<T> > &s){//半平面交
266    sort(s.begin(),s.end(),angle_cmp);//線段左側為該線段半平面
267    int L,R,n=s.size();
268    vector<point<T> > px(n);
269    vector<line<T> > q(n);
270    q[L=R=0]=s[0];
271    for(int i=1;i<n;++i){
272        while(L<R&&s[i].cross(px[R-1])<=0)--R;
273        while(L<R&&s[i].cross(px[L])<=0)++L;
274        q[++R]=s[i];
275        if(q[R].parallel(q[R-1])){
276            --R;
277            if(q[R].cross(s[i].p1)>0)q[R]=s[i];
278        }
279        if(L<R)px[R-1]=q[R-1].line_intersection(q[R]);
280    }
281    while(L<R&&q[L].cross(px[R-1])<=0)--R;
282    p.clear();
283    if(R-L<=1)return 0;
284    px[R]=q[R].line_intersection(q[L]);
285    for(int i=L;i<=R;++i)p.push_back(px[i]);
286    return R-L+1;
287 }
288 };
289 template<typename T>
290 struct triangle{
291    point<T> a,b,c;
292    triangle(){}
293    triangle(const point<T> &a,const point<T> &b,const point<T> &c):a(a),b(b),c(c){}
294    T area()const{
295        T t=(b-a).cross(c-a)/2;
296        return t>0?t:-t;
297    }
298    point<T> barycenter()const{//重心
299        return (a+b+c)/3;
300    }
301    point<T> circumcenter()const{//外心
302        static line<T> u,v;
303        u.p1=(a+b)/2;
304        u.p2=point<T>(u.p1.x-a.y+b.y,u.p1.y+a.x-b.x);
305        v.p1=(a+c)/2;
306        v.p2=point<T>(v.p1.x-a.y+c.y,v.p1.y+a.x-c.x);
307        return u.line_intersection(v);
308    }
309    point<T> incenter()const{//內心
310        T A=sqrt((b-c).abs2()),B=sqrt((a-c).abs2()),C=sqrt((a-b).abs2());
311        return point<T>(A*a.x+B*b.x+C*c.x,A*a.y+B*b.y+C*c.y)/(A+B+C);
312    }
313    point<T> perpencenter()const{//垂心
314        return barycenter()*3-circumcenter()*2;
315    }
316 };
317 template<typename T>
318 struct point3D{
319    T x,y,z;
320    point3D(){}
321    point3D(const T&x,const T&y,const T&z):x(x),y(y),z(z){}
322    point3D operator+(const point3D &b)const{
323        return point3D(x+b.x,y+b.y,z+b.z);}
324    point3D operator-(const point3D &b)const{
325        return point3D(x-b.x,y-b.y,z-b.z);}
326    point3D operator*(const T &b)const{
327        return point3D(x*b,y*b,z*b);}
328    point3D operator/(const T &b)const{
329        return point3D(x/b,y/b,z/b);}
330    bool operator==(const point3D &b)const{
331        return x==b.x&&y==b.y&&z==b.z;}
332    T dot(const point3D &b)const{
333        return x*b.x+y*b.y+z*b.z;}
334    point3D cross(const point3D &b)const{
335        return point3D(y*b.z-z*b.y,z*b.x-x*b.z,x*b.y-y*b.x);}
336    T abs2()const{//向量長度的平方
337        return dot(*this);}
338    T area2(const point3D &b)const{//和b、原點圍成面積的平方
339        return cross(b).abs2()/4;}
340 };
341 template<typename T>
342 struct line3D{
343    point3D<T> p1,p2;
344    line3D(){}
345    line3D(const point3D<T> &p1,const point3D<T> &p2):p1(p1),p2(p2){}
346    T dis2(const point3D<T> &p,bool is_segment=0)const{//點跟直線/線段的距離平方
347        point3D<T> v=p2-p1,v1=p-p1;
348        if(is_segment){
349            point3D<T> v2=p-p2;
350            if(v.dot(v1)<=0)return v1.abs2();
351            if(v.dot(v2)>=0)return v2.abs2();
352        }
353        point3D<T> tmp=v.cross(v1);
354        return tmp.abs2()/v.abs2();
355    }
356    pair<point3D<T>,point3D<T> > closest_pair(const line3D<T> &l)const{
357        point3D<T> v1=(p1-p2),v2=(l.p1-l.p2);
358        point3D<T> N=v1.cross(v2),ab(p1-l.p1);
359        //if(N.abs2()==0)return NULL;平行或重合
360        T tmp=N.dot(ab),ans=tmp*tmp/N.abs2();//最近點對距離
361        point3D<T> d1=p2-p1,d2=l.p2-l.p1,D=d1.cross(d2);
362        T t1=((l.p1-p1).cross(d2)).dot(D)/D.abs2();
363        T t2=((l.p1-p1).cross(d1)).dot(D)/D.abs2();
364        return make_pair(p1+d1*t1,l.p1+d2*t2);
365    }
366    bool same_side(const point3D<T> &a,const point3D<T> &b){
367        return (p2-p1).cross(a-p1).dot((p2-p1).cross(b-p1))>0;
368    }
369 };
370 template<typename T>
371 struct plane{
372    point3D<T> p0,n;//平面上的點和法向量
373    plane(){}
374    plane(const point3D<T> &p0,const point3D<T> &n):p0(p0),n(n){}
375    T dis2(const point3D<T> &p)const{//點到平面距離的平方
376        T tmp=(p-p0).dot(n);
377        return tmp*tmp/n.abs2();
378    }
379    point3D<T> projection(const point3D<T> &p)const{
380        return p-n*(p-p0).dot(n)/n.abs2();
381    }
382    point3D<T> line_intersection(const line3D<T> &l)const{
383        T tmp=n.dot(l.p2-l.p1);//等於0表示平行或重合該直線
384        return l.p1+(l.p2-l.p1)*(n.dot(p0-l.p1)/tmp);
385    }
386    line3D<T> plane_intersection(const plane &pl)const{
387        point3D<T> e=n.cross(pl.n),v=n.cross(e);
388        T tmp=pl.n.dot(v);//等於0表示平行或重合該平面
389        point3D<T> q=p0+(v*(pl.n.dot(pl.p0-p0))/tmp);
390        return line3D<T>(q,q+e);
391    }
392 };
393 template<typename T>
394 struct triangle3D{
395    point3D<T> a,b,c;
396    triangle3D(){}
397    triangle3D(const point3D<T> &a,const point3D<T> &b,const point3D<T> &c):a(a),b(b),c(c){}
398    bool point_in(const point3D<T> &p)const{//點在該平面上的投影在三角形中
399        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);
400    }
401 };
402 template<typename T>
403 struct tetrahedron{//四面體
```

```
404    point3D<T> a,b,c,d;
405    tetrahedron(){}
406    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){}
407    T volume6()const{//體積的六倍
408        return (d-a).dot((b-a).cross(c-a));
409    }
410    point3D<T> centroid()const{
411        return (a+b+c+d)/4;
412    }
413    bool point_in(const point3D<T> &p)const{
414        return triangle3D<T>(a,b,c).point_in(p)
               &&triangle3D<T>(c,d,a).point_in(p);
415    }
416  };
417  template<typename T>
418  struct convexhull3D{
419      static const int MAXN=1005;
420      struct face{
421          int a,b,c;
422          face(int a,int b,int c):a(a),b(b),c(c){}
423      };
424      vector<point3D<T>> pt;
425      vector<face> ans;
426      int fid[MAXN][MAXN];
427      void build(){
428          int n=pt.size();
429          ans.clear();
430          memset(fid,0,sizeof(fid));
431          ans.emplace_back(0,1,2);
432          ans.emplace_back(2,1,0);
433          int ftop = 0;
434          for(int i=3, ftop=1; i<n; ++i,++ftop){
435              vector<face> next;
436              for(auto &f:ans){
437                  T d=(pt[i]-pt[f.a]).dot((pt[f.b]-pt[
                       f.a]).cross(pt[f.c]-pt[f.a]));
438                  if(d<=0) next.push_back(f);
439                  int ff=0;
440                  if(d>0) ff=ftop;
441                  else if(d<0) ff=-ftop;
442                  fid[f.a][f.b]=fid[f.b][f.c]=fid[f.c
                       ][f.a]=ff;
443              }
444              for(auto &f:ans){
445                  if(fid[f.a][f.b]>0 && fid[f.a][f.b
                       ]!=fid[f.b][f.a])
446                      next.emplace_back(f.a,f.b,i);
447                  if(fid[f.b][f.c]>0 && fid[f.b][f.c
                       ]!=fid[f.c][f.b])
448                      next.emplace_back(f.b,f.c,i);
449                  if(fid[f.c][f.a]>0 && fid[f.c][f.a
                       ]!=fid[f.a][f.c])
450                      next.emplace_back(f.c,f.a,i);
451              }
452              ans=next;
453          }
454      }
455      point3D<T> centroid()const{
456          point3D<T> res(0,0,0);
457          T vol=0;
458          for(auto &f:ans){
459              T tmp=pt[f.a].dot(pt[f.b].cross(pt[f.c
                   ]));
460              res=res+(pt[f.a]+pt[f.b]+pt[f.c])*tmp;
461              vol+=tmp;
462          }
463          return res/(vol*4);
464      }
465  };
```

1.2   SmallestCircle.cpp

```
1   #include"Geometry.cpp"
2   struct Circle{
3       typedef point<double> p;
4       typedef const point<double> cp;
5       p x;
6       double r2;
7       bool incircle(cp &c)const{return (x-c).
            abs2()<=r2;}
8   };
9
10  Circle TwoPointCircle(Circle::cp &a, Circle
        ::cp &b) {
11      Circle::p m=(a+b)/2;
12      return (Circle){m,(a-m).abs2()};
13  }
14
15  Circle outcircle(Circle::p a, Circle::p b,
        Circle::p c) {
16      if(TwoPointCircle(a,b).incircle(c))
            return TwoPointCircle(a,b);
17      if(TwoPointCircle(b,c).incircle(a))
            return TwoPointCircle(b,c);
18      if(TwoPointCircle(c,a).incircle(b))
            return TwoPointCircle(c,a);
19      Circle::p ret;
20      double a1=b.x-a.x, b1=b.y-a.y, c1=(a1*a1
            +b1*b1)/2;
21      double a2=c.x-a.x, b2=c.y-a.y, c2=(a2*a2
            +b2*b2)/2;
22      double d = a1*b2 - a2*b1;
23      ret.x=a.x+(c1*b2-c2*b1)/d;
24      ret.y=a.y+(a1*c2-a2*c1)/d;
25      return (Circle){ret,(ret-a).abs2()};
26  }
27  //rand required
28  Circle SmallestCircle(std::vector<Circle::p>
        &p){
29      int n=p.size();
30      if(n==1) return (Circle){p[0],0.0};
31      if(n==2) return TwoPointCircle(p[0],p
            [1]);
32      random_shuffle(p.begin(),p.end());
33      Circle c = {p[0],0.0};
34      for(int i=0;i<n;++i){
35          if(c.incircle(p[i])) continue;
36          c=Circle{p[i],0.0};
37          for(int j=0;j<i;++j){
38              if(c.incircle(p[j])) continue;
39              c=TwoPointCircle(p[i],p[j]);
40              for(int k=0;k<j;++k){
41                  if(c.incircle(p[k]))
                        continue;
42                  c=outcircle(p[i],p[j],p[k]);
43              }
44          }
45      }
46      return c;
47  }
```

1.3   最近點對.cpp

```
1   #define INF LLONG_MAX
2   template<typename T>
3   T closest_pair(vector<point<T> >&v,vector<
        point<T> >&t,int l,int r){
4       T dis=INF, tmd;
5       if(l>=r)return dis;
6       int mid=(l+r)/2;
7       if((tmd=closest_pair(v,t,l,mid))<dis)dis=
            tmd;
8       if((tmd=closest_pair(v,t,mid+1,r))<dis)dis
            =tmd;
9       t.clear();
10      for(int i=l;i<=r;++i)
11          if((v[i].x-v[mid].x)*(v[i].x-v[mid].x)<
                dis)t.push_back(v[i]);
12      sort(t.begin(),t.end(),point<T>::y_cmp);//
            如果用merge_sort的方式可以到O(n)
13      for(size_t i=0;i<t.size();++i)
14          for(size_t j=1;j<=3&&i+j<t.size();++j)
15              if((tmd=(t[i]-t[i+j]).abs2())<dis)dis=
                    tmd;
16      return dis;
17  }
18  template<typename T>
19  inline T closest_pair(vector<point<T> > &v){
20      vector<point<T> >t;
21      sort(v.begin(),v.end(),point<T>::x_cmp);
22      return closest_pair(v,t,0,v.size()-1);//最
            近點對距離
23  }
```

2   Data_Structure

2.1   DLX.cpp

```
1   const int MAXN=4100, MAXM=1030, MAXND=16390;
2   struct DLX{
3       int n,m,sz,ansd;//高是n，寬是m的稀疏矩陣
4       int S[MAXM],H[MAXN];
5       int row[MAXND],col[MAXND];//每個節點代表的
            列跟行
6       int L[MAXND],R[MAXND],U[MAXND],D[MAXND];
7       vector<int> ans,anst;
8       void init(int _n,int _m){
9           n=_n,m=_m;
10          for(int i=0;i<=m;++i){
11              U[i]=D[i]=i,L[i]=i-1,R[i]=i+1;
12              S[i]=0;
```

```
13          }
14          R[m]=0,L[0]=m;
15          sz=m,ansd=INT_MAX;//ansd存最優解的個數
16          for(int i=1;i<=n;++i)H[i]=-1;
17      }
18      void add(int r,int c){
19          ++S[col[++sz]=c];
20          row[sz]=r;
21          D[sz]=D[c],U[D[c]]=sz,U[sz]=c,D[c]=sz;
22          if(H[r]<0)H[r]=L[sz]=R[sz]=sz;
23          else R[sz]=R[H[r]],L[R[H[r]]]=sz,L[sz]=H
                [r],R[H[r]]=sz;
24      }
25  #define DFOR(i,A,s) for(int i=A[s];i!=s;i=
        A[i])
26      void remove(int c){//刪除第c行和所有當前覆
            蓋到第c行的列
27          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
28          DFOR(i,D,c)DFOR(j,R,i){U[D[j]]=U[j],D[U[
                j]]=D[j],--S[col[j]];}
29      }
30      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
31          DFOR(i,U,c)DFOR(j,L,i){++S[col[j]],U[D[j
                ]]=j,D[U[j]]=j;}
32          L[R[c]]=c,R[L[c]]=c;
33      }
34      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
35          DFOR(i,D,nd)L[R[i]]=L[i],R[L[i]]=R[i];
36      }
37      void restore2(int nd){//刪除nd所在的行當前
            所有點，為remove2的逆操作
38          DFOR(i,U,nd)L[R[i]]=R[L[i]]=i;
39      }
40      bool vis[MAXM];
41      int h(){//估價函數 for IDA*
42          int res=0;
43          memset(vis,0,sizeof(vis));
44          DFOR(i,R,0)if(!vis[i]){
45              vis[i]=1;
46              ++res;
47              DFOR(j,D,i)DFOR(k,R,j)vis[col[k]]=1;
48          }
49          return res;
50      }
51      bool dfs(int d){//for精確覆蓋問題
52          if(d+h()>=ansd)return 0;//找最佳解用，找
                任意解可以刪掉
53          if(!R[0]){ansd=d;return 1;}
54          int c=R[0];
55          DFOR(i,R,0)if(S[i]<S[c])c=i;
56          remove(c);
57          DFOR(i,D,c){
58              ans.push_back(row[i]);
59              DFOR(j,R,i)remove(col[j]);
60              if(dfs(d+1))return 1;
61              ans.pop_back();
62              DFOR(j,L,i)restore(col[j]);
63          }
64          restore(c);
```

```cpp
    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
    return ans.clear(), dfs(0);
  }
  void min_cover(){//解最小重複覆蓋問題
    anst.clear();//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>//有kd個維度
struct kd_tree{
  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;//記得要給INF，表示極大值
  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) return u=new node(x), 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;
        return u=0, 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);
    }
    --u->s;
    if(erase(cmp(x,u->pid)?u->l:u->r,(k+1)%
        kd,x))
      return --u->s, 1;
    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;//回傳離x第k近的點的距離
  }
  const std::vector<point> &range(const
      point&mi,const point&ma){
    in_range.clear();
    range(root,0,mi,ma);
    return in_range;//回傳介於mi到ma之間的點
        vector
  }
  int size(){return root?root->s:0;}
};
```

2.3 kd_tree_replace_segment_tr

```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;
```

```cpp
11    if(l){
12      for(int i=0;i<kd;++i){
13        mi.d[i]=min(mi.d[i],l->mi.d[i]);
14        ma.d[i]=max(ma.d[i],l->ma.d[i]);
15      }
16      s+=l->s;
17    }
18    if(r){
19      for(int i=0;i<kd;++i){
20        mi.d[i]=min(mi.d[i],r->mi.d[i]);
21        ma.d[i]=max(ma.d[i],r->ma.d[i]);
22      }
23      s+=r->s;
24    }
25  }
26  void up2(){
27    //其他懶惰標記向上更新
28  }
29  void down(){
30    //其他懶惰標記下推
31  }
32 }*root;
33
34 /*檢查區間包含用的函數*/
35 inline bool range_include(node *o,const
      point &L,const point &R){
36   for(int i=0;i<kd;++i){
37     if(L.d[i]>o->ma.d[i]||R.d[i]<o->mi.d[i])
38       return 0;
     }//只要(L,R)區間有和o的區間有交集就回傳
         true
39   return 1;
40 }
41 inline bool range_in_range(node *o,const
      point &L,const point &R){
42   for(int i=0;i<kd;++i){
43     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
45   return 1;
46 }
47 inline bool point_in_range(node *o,const
      point &L,const point &R){
48   for(int i=0;i<kd;++i){
49     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
51   return 1;
52 }
53
54 /*單點修改，以單點改值為例*/
55 void update(node *u,const point &x,int data,
      int k=0){
56   if(!u)return;
57   u->down();
58   if(u->pid==x){
59     u->data=data;
60     u->up2();
61     return;
62   }
63   cmp.sort_id=k;
64   update(cmp(x,u->pid)?u->l:u->r,x,data,(k
      +1)%kd);
```

```cpp
65    u->up2();
66 }
67
68 /*區間修改*/
69 void update(node *o,const point &L,const
      point &R,int data){
70   if(!o)return;
71   o->down();
72   if(range_in_range(o,L,R)){
73     //區間懶惰標記修改
74     o->down();
75     return;
76   }
77   if(point_in_range(o,L,R)){
78     //這個點在(L,R)區間，但是他的左右子樹不
           一定在區間中
79     //單點懶惰標記修改
80   }
81   if(o->l&&range_include(o->l,L,R))update(o
      ->l,L,R,data);
82   if(o->r&&range_include(o->r,L,R))update(o
      ->r,L,R,data);
83   o->up2();
84 }
85
86 /*區間查詢，以總和為例*/
87 int query(node *o,const point &L,const point
      &R){
88   if(!o)return 0;
89   o->down();
90   if(range_in_range(o,L,R))return o->sum;
91   int ans=0;
92   if(point_in_range(o,L,R))ans+=o->data;
93   if(o->l&&range_include(o->l,L,R))ans+=
        query(o->l,L,R);
94   if(o->r&&range_include(o->r,L,R))ans+=
        query(o->r,L,R);
95   return ans;
96 }
```

2.4 reference_point.cpp

```cpp
1  template<typename T>
2  struct _RefC{
3    T data;
4    int ref;
5    _RefC(const T&d=0):data(d),ref(0){}
6  };
7  template<typename T>
8  struct _rp{
9    _RefC<T> *p;
10   T *operator->(){return &p->data;}
11   T &operator*(){return p->data;}
12   operator _RefC<T>*(){return p;}
13   _rp &operator=(const _rp &t){
14     if(p&&!--p->ref)delete p;
15     p=t.p,p&&++p->ref;
16     return *this;
17   }
18   _rp(_RefC<T> *t=0):p(t){p&&++p->ref;}
19   _rp(const _rp &t):p(t.p){p&&++p->ref;}
20   ~_rp(){if(p&&!--p->ref)delete p;}
```

```cpp
21 };
22 template<typename T>
23 inline _rp<T> new_rp(const T&nd){
24   return _rp<T>(new _RefC<T>(nd));
25 }
```

2.5 skew_heap.cpp

```cpp
1 node *merge(node *a,node *b){
2   if(!a||!b) return a?a:b;
3   if(b->data<a->data) swap(a,b);
4   swap(a->l,a->r);
5   a->l=merge(b,a->l);
6   return a;
7 }
```

2.6 undo_disjoint_set.cpp

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

2.7 整體二分.cpp

```cpp
1  void totBS(int L, int R, vector<Item> M){
2    if(Q.empty()) return; //維護全域B陣列
3    if(L==R) 整個M的答案=r, return;
4    int mid = (L+R)/2;
5    vector<Item> mL, mR;
6    do_modify_B_with_divide(mid,M);
7    //讓B陣列在遞迴的時候只會保留[L~mid]的資訊
8    undo_modify_B(mid,M);
9    totBS(L,mid,mL);
10   totBS(mid+1,R,mR);
11 }
```

3 default

3.1 debug.cpp

```cpp
1  //volatile
2  #ifdef DEBUG
3  #define dbg(...) {\
4    fprintf(stderr,"%s - %d : (%s) = ",
         __PRETTY_FUNCTION__,__LINE__,#
         __VA_ARGS__);\
5    _DO(__VA_ARGS__);\
6  }
7  template<typename I> void _DO(I&&x){cerr<<x
     <<endl;}
8  template<typename I,typename...T> void _DO(I
     &&x,T&&...tail){cerr<<x<<", ";_DO(tail
     ...);}
9  #else
10 #define dbg(...)
11 #endif
```

3.2 ext.cpp

```cpp
1  #include<bits/extc++.h>
2  #include<ext/pd_ds/assoc_container.hpp>
3  #include<ext/pd_ds/tree_policy.hpp>
4  using namespace __gnu_cxx;
5  using namespace __gnu_pbds;
6  template<typename T>
7  using pbds_set = tree<T,null_type,less<T>,
       rb_tree_tag,
       tree_order_statistics_node_update>;
8  template<typename T,typename U>
9  using pbds_map = tree<T,U,less<T>,
       rb_tree_tag,
       tree_order_statistics_node_update>;
10 using heap = __gnu_pbds::priority_queue<int
     >;
11 //s.find_by_order(1);//0 base
12 //s.order_of_key(1);
```

3.3 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));
-Wl,--stack,214748364 -trigraphs
//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);
  }
}
```

3.4 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;
}
// #!/bin/bash
// g++ -std=c++11 -O2 -Wall -Wextra -Wno-
    unused-result -DDEBUG $1 && ./a.out
//  -fsanitize=address -fsanitize=undefined
    -fsanitize=return
```

4 Flow

4.1 dinic.cpp

```cpp
template<typename T>
struct DINIC{
  static const int MAXN=105;
  static const T INF=INT_MAX;
  int n, level[MAXN], cur[MAXN];
  struct edge{
    int v,pre;
    T cap,flow,r;
    edge(int v,int pre,T cap):v(v),pre(pre),
        cap(cap),flow(0),r(cap){}
  };
  int g[MAXN];
  vector<edge> e;
  void init(int _n){
    memset(g,-1,sizeof(int)*((n=_n)+1));
    e.clear();
  }
```

```cpp
  void add_edge(int u,int v,T 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;
  }
  int bfs(int s,int t){
    memset(level,0,sizeof(int)*(n+1));
    memcpy(cur,g,sizeof(int)*(n+1));
    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;
  }
  T dfs(int u,int t,T cur_flow=INF){
    if(u==t)return cur_flow;
    T df;
    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,min(cur_flow,e[i
            ].r))){
          e[i].flow+=df;
          e[i^1].flow-=df;
          e[i].r-=df;
          e[i^1].r+=df;
          return df;
        }
      }
    }
    return level[u]=0;
  }
  T 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;
      }
    }
    T ans=0, mf=0;
    while(bfs(s,t))while(mf=dfs(s,t))ans+=mf
        ;
    return ans;
  }
};
```

4.2 ISAP_with_cut.cpp

```cpp
template<typename T>
struct ISAP{
  static const int MAXN=105;
  static const T INF=INT_MAX;
```

```cpp
  int n;//點數
  int d[MAXN],gap[MAXN],cur[MAXN];
  struct edge{
    int v,pre;
    T cap,flow,r;
    edge(int v,int pre,T cap):v(v),pre(pre),
        cap(cap),flow(0),r(cap){}
  };
  int g[MAXN];
  vector<edge> e;
  void init(int _n){
    memset(g,-1,sizeof(int)*((n=_n)+1));
    e.clear();
  }
  void add_edge(int u,int v,T 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;
  }
  T dfs(int u,int s,int t,T cur_flow=INF){
    if(u==t)return cur_flow;
    T 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,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 mh=n;
    for(int i=cur[u]=g[u];~i;i=e[i].pre){
      if(e[i].r&&d[e[i].v]<mh)mh=d[e[i].v];
    }
    if(!--gap[d[u]])d[s]=n;
    else ++gap[d[u]=++mh];
    return cur_flow-tf;
  }
  T 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;
    }
    T max_flow=0;
    for(gap[0]=n;d[s]<n;)max_flow+=dfs(s,s,t
        );
    return max_flow;
  }
  vector<int> cut_e;//最小割邊集
  bool vis[MAXN];
  void dfs_cut(int u){
    vis[u]=1;//表示u屬於source的最小割集
    for(int i=g[u];~i;i=e[i].pre)
      if(e[i].flow<e[i].cap&&!vis[e[i].v])
        dfs_cut(e[i].v);
```

```cpp
  }
  T min_cut(int s,int t){
    T ans=isap(s,t);
    memset(vis,0,sizeof(bool)*(n+1));
    dfs_cut(s), cut_e.clear();
    for(int u=0;u<=n;++u)
      if(vis[u])for(int i=g[u];~i;i=e[i].pre
          )
        if(!vis[e[i].v])cut_e.push_back(i);
    return ans;
  }
};
```

4.3 MinCostMaxFlow.cpp

```cpp
template<typename _T>
struct MCMF{
  static const int MAXN=440;
  static const _T INF=999999999;
  struct edge{
    int v,pre;
    _T cap,cost;
    edge(int v,int pre,_T cap,_T cost):v(v),
        pre(pre),cap(cap),cost(cost){}
  };
  int n,S,T;
  _T dis[MAXN],piS,ans;
  bool vis[MAXN];
  vector<edge> e;
  int g[MAXN];
  void init(int _n){
    memset(g,-1,sizeof(int)*((n=_n)+1));
    e.clear();
  }
  void add_edge(int u,int v,_T cap,_T cost,
      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;
  }
  _T augment(int u,_T cur_flow){
    if(u==T||!cur_flow)return ans+=piS*
        cur_flow,cur_flow;
    vis[u]=1;
    _T 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,min(r,e[i].cap));
        e[i].cap-=d;
        e[i^1].cap+=d;
        if(!(r-=d))break;
      }
    }
    return cur_flow-r;
  }
  bool modlabel(){
    for(int u=0;u<=n;++u)dis[u]=INF;
    static deque<int>q;
    dis[T]=0,q.push_back(T);
    while(q.size()){
```

```
44    int u=q.front();q.pop_front();
45    _T dt;
46    for(int i=g[u];~i;i=e[i].pre){
47      if(e[i^1].cap&&(dt=dis[u]-e[i].cost)
            <dis[e[i].v]){
48        if((dis[e[i].v]=dt)<=dis[q.size()?
              q.front():S]){
49          q.push_front(e[i].v);
50        }else q.push_back(e[i].v);
51      }
52    }
53  }
54  for(int u=0;u<=n;++u)
55    for(int i=g[u];~i;i=e[i].pre)
56      e[i].cost+=dis[e[i].v]-dis[u];
57  return piS+=dis[S], dis[S]<INF;
58  }
59  _T mincost(int s,int t){
60    S=s,T=t;
61    piS=ans=0;
62    while(modlabel()){
63      do memset(vis,0,sizeof(bool)*(n+1));
64      while(augment(S,INF));
65    }return ans;
66  }
67 };
```

5 Graph

5.1 Augmenting_Path.cpp

```cpp
#define MAXN1 505
#define MAXN2 505
int n1,n2;//n1個點連向n2個點
int match[MAXN2];//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]))
      return match[v]=u, 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;
}
```

5.2 Augmenting_Path_multiple.

```cpp
#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]){
      return match_list[v].push_back(u),
            true;
    }else{
      for(size_t j=0;j<match_list[v].size()
          ;++j){
        int next_u=match_list[v][j];
        if(dfs(next_u))
          return match_list[v][j]=u, true;
      }
    }
  }
  return false;
}
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;
}
```

5.3 blossom_matching.cpp

```cpp
#define MAXN 505
vector<int>g[MAXN];
int pa[MAXN],match[MAXN],st[MAXN],S[MAXN],v[
    MAXN];
int t,n;
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
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];
  }
}
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;
}
int blossom(){
  int ans=0;
  for(int i=1;i<=n;++i)
    if(!match[i]&&bfs(i))++ans;
  return ans;
}
```

5.4 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;
void init(){
  for(int i=0;i<n;++i){
    f[0][i]=1;
    g[i].clear(), rg[i].clear();
  }
}
void add_edge(int u,int v){
  g[u].push_back(v), rg[v].push_back(u);
}
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];
}
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;
}
```

5.5 KM.cpp

```cpp
#define MAXN 405
#define INF 0x3f3f3f3f
int n;// 1-base，0表示沒有匹配
int g[MAXN][MAXN],lx[MAXN],ly[MAXN],pa[MAXN
    ],slack_y[MAXN];
int match_y[MAXN],match_x[MAXN];
bool vx[MAXN],vy[MAXN];
void augment(int y){
  for(int x,z;y;y=z){
    x=pa[y],z=match_x[x];
    match_y[y]=x,match_x[x]=y;
  }
}
void bfs(int st){
  for(int i=1;i<=n;++i)slack_y[i]=INF,vx[i]=
      vy[i]=0;
  queue<int> q;q.push(st);
  for(;;){
    while(q.size()){
      int x=q.front();q.pop();
      vx[x]=1;
      for(int y=1;y<=n;++y)if(!vy[y]){
        int t=lx[x]+ly[y]-g[x][y];
        if(t==0){
          pa[y]=x;
          if(!match_y[y]){augment(y);return
              ;}
          vy[y]=1,q.push(match_y[y]);
        }else if(slack_y[y]>t)pa[y]=x,
            slack_y[y]=t;
      }
    }
    int cut=INF;
    for(int y=1;y<=n;++y){
      if(!vy[y]&&cut>slack_y[y])cut=slack_y[
          y];
    }
    for(int j=1;j<=n;++j){
      if(vx[j])lx[j]-=cut;
      if(vy[j])ly[j]+=cut;
      else slack_y[j]-=cut;
    }
    for(int y=1;y<=n;++y){
      if(!vy[y]&&slack_y[y]==0){
```

```
40        if(!match_y[y]){augment(y);return;}
41        vy[y]=1,q.push(match_y[y]);
42      }
43    }
44  }
45 }
46 long long KM(){
47   memset(match_y,0,sizeof(int)*(n+1));
48   memset(ly,0,sizeof(int)*(n+1));
49   for(int x=1;x<=n;++x){
50     lx[x]=-INF;
51     for(int y=1;y<=n;++y)
52       lx[x]=max(lx[x],g[x][y]);
53   }
54   for(int x=1;x<=n;++x)bfs(x);
55   long long ans=0;
56   for(int y=1;y<=n;++y)ans+=g[match_y[y]][y
       ];
57   return ans;
58 }
```

5.6 MaximumClique.cpp

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

```
38        if(g[u][v])stk[1][ns++]=v;
39      dfs(ns,1),dp[u]=ans;
40    }
41    return ans;
42  }
43 };
```

5.7 MinimumMeanCycle.cpp

```
1 #include<cstdint>//for DBL_MAX
2 int dp[maxN+1][maxN+1];
3 double mnc(int n){
4   int u,v,w;
5   const int inf=0x7f7f7f7f;
6   memset(dp,0x7f,sizeof(dp));
7   memset(dp[0],0,sizeof(dp[0]));
8   for(int i=0;i<n;++i){
9     for(auto e:E){
10      tie(u,v,w)=e;
11      if(dp[i][u]!=inf)
12        dp[i+1][v]=min(dp[i+1][v],dp[i][u]+w);
13    }
14  }
15  double res = DBL_MAX;
16  for(int i=1;i<=n;++i){
17    double val = DBL_MIN;
18    for(int j=0;j<n;++j)
19      val=max(val,double(dp[n][i]-dp[i][j
        ])/(n-j));
20    res=min(res,val);
21  }
22  return res;
23 }
```

5.8 Minimum_General_Weighte

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

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

5.9 Rectilinear_MST.cpp

```
1 //平面曼哈頓最小生成樹構造圖(去除非必要邊)
2 #define T int
3 #define INF 0x3f3f3f3f
4 struct point{
5   T x,y;
6   int id;//每個點的編號都要不一樣,從0開始編
      號
7   point(){}
8   T dist(const point &p)const{
9     return abs(x-p.x)+abs(y-p.y);
10  }
```

```
11 };
12 bool cmpx(const point &a,const point &b){
13   return a.x<b.x||(a.x==b.x&&a.y<b.y);
14 }
15 struct edge{
16   int u,v;
17   T cost;
18   edge(int u,int v,T c):u(u),v(v),cost(c){}
19   bool operator<(const edge&e)const{
20     return cost<e.cost;
21   }
22 };
23 struct bit_node{
24   T mi;
25   int id;
26   bit_node(const T&mi=INF,int id=-1):mi(mi),
       id(id){}
27 };
28 vector<bit_node> bit;
29 void bit_update(int i,const T&data,int id){
30   for(;i;i-=i&(-i)){
31     if(data<bit[i].mi)bit[i]=bit_node(data,
         id);
32   }
33 }
34 int bit_find(int i,int m){
35   bit_node x;
36   for(;i<=m;i+=i&(-i)) if(bit[i].mi<x.mi)x=
       bit[i];
37   return x.id;
38 }
39 vector<edge> build_graph(int n,point p[]){
40   vector<edge> e;//edge for MST
41   for(int dir=0;dir<4;++dir){//4種座標變換
42     if(dir%2) for(int i=0;i<n;++i) swap(p[i
         ].x,p[i].y);
43     else if(dir==2) for(int i=0;i<n;++i) p[i
         ].x=-p[i].x;
44     sort(p,p+n,cmpx);
45     vector<T> ga(n), gb;
46     for(int i=0;i<n;++i)ga[i]=p[i].y-p[i].x;
47     gb=ga, sort(gb.begin(),gb.end());
48     gb.erase(unique(gb.begin(),gb.end()),gb.
         end());
49     int m=gb.size();
50     bit=vector<bit_node>(m+1);
51     for(int i=n-1;i>=0;--i){
52       int pos=lower_bound(gb.begin(),gb.end
           (),ga[i])-gb.begin()+1;
53       int ans=bit_find(pos,m);
54       if(~ans)e.push_back(edge(p[i].id,p[ans
           ].id,p[i].dist(p[ans])));
55       bit_update(pos,p[i].x+p[i].y,i);
56     }
57   }
58   return e;
59 }
```

5.10 treeISO.cpp

```
1 const int MAXN=100005;
2 const long long X=12327,P=0xdefaced;
3 vector<int> g[MAXN];
```

```cpp
 4 bool vis[MAXN];
 5 long long dfs(int u){//hash ver
 6   vis[u]=1;
 7   vector<long long> tmp;
 8   for(auto v:g[u])if(!vis[v])tmp.PB(dfs(v));
 9   if(tmp.empty())return 177;
10   long long ret=4931;
11   sort(tmp.begin(),tmp.end());
12   for(auto v:tmp)ret=((ret*X)^v)%P;
13   return ret;
14 }
15 //-------------------------
16 string dfs(int x,int p){
17   vector<string> c;
18   for(int y:g[x])
19     if(y!=p)c.emplace_back(dfs(y,x));
20   sort(c.begin(),c.end());
21   string ret("(");
22   for(auto &s:c)ret+=s;
23   ret+=")";
24   return ret;
25 }
```

5.11 全局最小割.cpp

```cpp
 1 const int INF=0x3f3f3f3f;
 2 template<typename T>
 3 struct stoer_wagner{// 0-base
 4   static const int MAXN=150;
 5   T g[MAXN][MAXN],dis[MAXN];
 6   int nd[MAXN],n,s,t;
 7   void init(int _n){
 8     n=_n;
 9     for(int i=0;i<n;++i)
10       for(int j=0;j<n;++j)g[i][j]=0;
11   }
12   void add_edge(int u,int v,T w){
13     g[u][v]=g[v][u]+=w;
14   }
15   T min_cut(){
16     T ans=INF;
17     for(int i=0;i<n;++i)nd[i]=i;
18     for(int ind,tn=n;tn>1;--tn){
19       for(int i=1;i<tn;++i)dis[nd[i]]=0;
20       for(int i=1;i<tn;++i){
21         ind=i;
22         for(int j=i;j<tn;++j){
23           dis[nd[j]]+=g[nd[i-1]][nd[j]];
24           if(dis[nd[ind]]<dis[nd[j]])ind=j;
25         }
26         swap(nd[ind],nd[i]);
27       }
28       if(ans>dis[nd[ind]])ans=dis[t=nd[ind
           ]],s=nd[ind-1];
29       for(int i=0;i<tn;++i)
30         g[nd[ind-1]][nd[i]]=g[nd[i]][nd[ind
             -1]]+=g[nd[i]][nd[ind]];
31     }
32     return ans;
33   }
34 };
```

5.12 平面圖判定.cpp

```cpp
 1 static const int MAXN = 20;
 2 struct Edge{
 3   int u, v;
 4   Edge(int s, int d) : u(s), v(d) {}
 5 };
 6 bool isK33(int n, int degree[]){
 7   int t = 0, z = 0;
 8   for(int i=0;i<n;++i){
 9     if(degree[i] == 3)++t;
10     else if(degree[i] == 0)++z;
11     else return false;
12   }
13   return t == 6 && t + z == n;
14 }
15 bool isK5(int n, int degree[]){
16   int f = 0, z = 0;
17   for(int i=0;i<n;++i){
18     if(degree[i] == 4)++f;
19     else if(degree[i] == 0)++z;
20     else return false;
21   }
22   return f == 5 && f + z == n;
23 }
24 // it judge a given graph is Homeomorphic
       with K33 or K5
25 bool isHomeomorphic(bool G[MAXN][MAXN],
       const int n){
26   for(;;){
27     int cnt = 0;
28     for(int i=0;i<n;++i){
29       vector<Edge> E;
30       for(int j=0;j<n&&E.size()<3;++j)
31         if(G[i][j] && i != j)
32           E.push_back(Edge(i, j));
33       if(E.size() == 1){
34         G[i][E[0].v] = G[E[0].v][i] = false;
35       }else if(E.size() == 2){
36         G[i][E[0].v] = G[E[0].v][i] = false;
37         G[i][E[1].v] = G[E[1].v][i] = false;
38         G[E[0].v][E[1].v] = G[E[1].v][E[0].v
             ] = true;
39         ++cnt;
40       }
41     }
42     if(cnt == 0)break;
43   }
44   static int degree[MAXN];
45   fill(degree, degree + n, 0);
46   for(int i=0;i<n;++i){
47     for(int j=i+1; j<n; ++j){
48       if(!G[i][j])continue;
49       ++degree[i];
50       ++degree[j];
51     }
52   }
53   return !(isK33(n, degree) || isK5(n,
       degree));
54 }
```

5.13 弦圖完美消除序列.cpp

```cpp
 1 struct chordal{
 2   static const int MAXN=1005;
 3   int n;// 0-base
 4   vector<int>G[MAXN];
 5   int rank[MAXN],label[MAXN];
 6   bool mark[MAXN];
 7   void init(int _n){n=_n;
 8     for(int i=0;i<n;++i)G[i].clear();
 9   }
10   void add_edge(int u,int v){
11     G[u].push_back(v);
12     G[v].push_back(u);
13   }
14   vector<int> MCS(){
15     memset(rank,-1,sizeof(int)*n);
16     memset(label,0,sizeof(int)*n);
17     priority_queue<pair<int,int> > pq;
18     for(int i=0;i<n;++i)pq.push(make_pair(0,
         i));
19     for(int i=n-1;i>=0;--i)for(;;){
20       int u=pq.top().second;pq.pop();
21       if(~rank[u])continue;
22       rank[u]=i;
23       for(auto v:G[u])if(rank[v]==-1){
24         pq.push(make_pair(++label[v],v));
25       }
26       break;
27     }
28     vector<int> res(n);
29     for(int i=0;i<n;++i)res[rank[i]]=i;
30     return res;
31   }
32   bool check(vector<int> ord){//弦圖判定
33     for(int i=0;i<n;++i)rank[ord[i]]=i;
34     memset(mark,0,sizeof(bool)*n);
35     for(int i=0;i<n;++i){
36       vector<pair<int,int> > tmp;
37       for(auto u:G[ord[i]])if(!mark[u])
38         tmp.push_back(make_pair(rank[u],u));
39       sort(tmp.begin(),tmp.end());
40       if(tmp.size()){
41         int u=tmp[0].second;
42         set<int> S;
43         for(auto v:G[u])S.insert(v);
44         for(size_t j=1;j<tmp.size();++j)
45           if(!S.count(tmp[j].second))return
               0;
46       }
47       mark[ord[i]]=1;
48     }
49     return 1;
50   }
51 };
```

5.14 最小斯坦納樹 DP.cpp

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

5.15 最小樹形圖 __ 朱劉.cpp

```cpp
 1 template<typename T>
 2 struct zhu_liu{
 3   static const int MAXN=110,MAXM=10005;
 4   struct node{
 5     int u,v;
 6     T w,tag;
 7     node *l,*r;
 8     node(int u=0,int v=0,T w=0):u(u),v(v),w(
         w),tag(0),l(0),r(0){}
 9     void down(){
10       w+=tag;
11       if(l)l->tag+=tag;
12       if(r)r->tag+=tag;
13       tag=0;
14     }
15   }mem[MAXM];//靜態記憶體
16   node *pq[MAXN*2],*E[MAXN*2];
17   int st[MAXN*2],id[MAXN*2],m;
18   void init(int n){
19     for(int i=1;i<=n;++i)
20       pq[i]=E[i]=0, st[i]=id[i]=i;
21     m=0;
22   }
23   node *merge(node *a,node *b){//skew heap
24     if(!a||!b)return a?a:b;
25     a->down(),b->down();
26     if(b->w<a->w)return merge(b,a);
27     swap(a->l,a->r);
28     a->l=merge(b,a->l);
29     return a;
30   }
31   void add_edge(int u,int v,T w){
32     if(u!=v)pq[v]=merge(pq[v],&(mem[m++]=
         node(u,v,w)));
33   }
34   int find(int x,int *st){
```

```cpp
35    return st[x]==x?x:st[x]=find(st[x],st);
36  }
37  T build(int root,int n){
38    T ans=0;int N=n,all=n;
39    for(int i=1;i<=N;++i){
40      if(i==root||!pq[i])continue;
41      while(pq[i]){
42        pq[i]->down(),E[i]=pq[i];
43        pq[i]=merge(pq[i]->l,pq[i]->r);
44        if(find(E[i]->u,id)!=find(i,id))
            break;
45      }
46      if(find(E[i]->u,id)==find(i,id))
          continue;
47      ans+=E[i]->w;
48      if(find(E[i]->u,st)==find(i,st)){
49        if(pq[i])pq[i]->tag-=E[i]->w;
50        pq[++N]=pq[i];id[N]=N;
51        for(int u=find(E[i]->u,id);u!=i;u=
            find(E[u]->u,id)){
52          if(pq[u])pq[u]->tag-=E[u]->w;
53          id[find(u,id)]=N;
54          pq[N]=merge(pq[N],pq[u]);
55        }
56        st[N]=find(i,st);
57        id[find(i,id)]=N;
58      }else st[find(i,st)]=find(E[i]->u,st)
            ,--all;
59    }
60    return all==1?ans:-INT_MAX;//圖不連通就
          無解
61  }
62  };
```

5.16 穩定婚姻模板.cpp

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

6 language

6.1 CNF.cpp

```cpp
1  #define MAXN 55
2  struct CNF{
3    int s,x,y;//s->xy | s->x, if y==-1
4    int cost;
5    CNF(){}
6    CNF(int s,int x,int y,int c):s(s),x(x),y(y
       ),cost(c){}
7  };
8  int state;//規則數量
9  map<char,int> rule;//每個字元對應到的規則，
       小寫字母為終端字符
10 vector<CNF> cnf;
11 void init(){
12   state=0;
13   rule.clear();
14   cnf.clear();
15 }
16 void add_to_cnf(char s,const string &p,int
       cost){
17   //加入一個s -> <p>的文法，代價為cost
18   if(rule.find(s)==rule.end())rule[s]=state
         ++;
19   for(auto c:p)if(rule.find(c)==rule.end())
         rule[c]=state++;
20   if(p.size()==1){
21     cnf.push_back(CNF(rule[s],rule[p[0]],-1,
           cost));
22   }else{
23     int left=rule[s];
24     int sz=p.size();
25     for(int i=0;i<sz-2;++i){
26       cnf.push_back(CNF(left,rule[p[i]],
             state,0));
27       left=state++;
28     }
29     cnf.push_back(CNF(left,rule[p[sz-2]],
           rule[p[sz-1]],cost));
30   }
31 }
32 vector<long long> dp[MAXN][MAXN];
33 vector<bool> neg_INF[MAXN][MAXN];//如果花費
       是負的可能會有無限小的情形
34 void relax(int l,int r,const CNF &c,long
       long cost,bool neg_c=0){
35   if(!neg_INF[l][r][c.s]&&(neg_INF[l][r][c.x
         ]||cost<dp[l][r][c.s])){
36     if(neg_c||neg_INF[l][r][c.x]){
37       dp[l][r][c.s]=0;
38       neg_INF[l][r][c.s]=true;
39     }else dp[l][r][c.s]=cost;
40   }
41 }
42 void bellman(int l,int r,int n){
43   for(int k=1;k<=state;++k)
44     for(auto c:cnf)
45       if(c.y==-1)relax(l,r,c,dp[l][r][c.x]+c
             .cost,k==n);
46 }
47 void cyk(const vector<int> &tok){
48   for(int i=0;i<(int)tok.size();++i){
49     for(int j=0;j<(int)tok.size();++j){
50       dp[i][j]=vector<long long>(state+1,
             INT_MAX);
51       neg_INF[i][j]=vector<bool>(state+1,
             false);
52     }
53     dp[i][i][tok[i]]=0;
54     bellman(i,i,tok.size());
55   }
56   for(int r=1;r<(int)tok.size();++r){
57     for(int l=r-1;l>=0;--l){
58       for(int k=l;k<r;++k)
59         for(auto c:cnf)
60           if(~c.y)relax(l,r,c,dp[l][k][c.x]+
                 dp[k+1][r][c.y]+c.cost);
61       bellman(l,r,tok.size());
62     }
63   }
64 }
```

7 Linear_Programming

7.1 最大密度子圖.cpp

```cpp
1  typedef double T;//POJ 3155
2  const int MAXN=105;
3  struct edge{
4    int u,v;
5    T w;
6    edge(int u=0,int v=0,T w=0):u(u),v(v),w(w)
       {}
7  };
8  vector<edge> E;
9  int n,m;// 1-base
10 T de[MAXN],pv[MAXN];//每個點的邊權和和點權(
       有些題目會給)
11 void init(){
12   E.clear();
13   for(int i=1;i<=n;++i)de[i]=pv[i]=0;
14 }
15 void add_edge(int u,int v,T w){
16   E.push_back(edge(u,v,w));
17   de[u]+=w,de[v]+=w;
18 }
19 T U;//二分搜的最大值
20 void get_U(){
21   U=0;
22   for(int i=1;i<=n;++i)U+=2*pv[i];
23   for(size_t i=0;i<E.size();++i)U+=E[i].w;
24 }
25 ISAP<T> isap;//網路流
26 int s,t;//原匯點
27 void build(T L){
28   isap.init(n+2);
29   for(size_t i=0;i<E.size();++i)
30     isap.add_edge(E[i].u,E[i].v,E[i].w);
31   for(int v=1;v<=n;++v){
32     isap.add_edge(s,v,U);
33     isap.add_edge(v,t,U+2*L-de[v]-2*pv[v]);
34   }
35 }
36 int main(){
37   while(~scanf("%d%d",&n,&m)){
38     if(!m){
39       puts("1\n1");
40       continue;
41     }
42     init();
43     int u,v;
44     for(int i=0;i<m;++i){
45       scanf("%d%d",&u,&v);
46       add_edge(u,v,1);
47     }
48     get_U();
49     s=n+1,t=n+2;
50     T l=0,r=U,k=1.0/(n*n);
51     while(r-l>k){//二分搜最大值
52       T mid=(l+r)/2;
53       build(mid);
54       T res=(U*n-isap.isap(s,t))/2;
55       if(res>0)l=mid;
56       else r=mid;
57     }
58     build(l);
59     isap.min_cut(s,t);
60     vector<int> ans;
61     for(int i=1;i<=n;++i)
62       if(isap.vis[i])ans.push_back(i);
63     printf("%d\n",ans.size());
64     for(size_t i=0;i<ans.size();++i)
65       printf("%d\n",ans[i]);
66   }
67   return 0;
68 }
```

8 Number_Theory

8.1 basic.cpp

```cpp
1  template<typename T>
2  void gcd(const T &a,const T &b,T &d,T &x,T &
       y){
3    if(!b) d=a,x=1,y=0;
4    else gcd(b,a%b,d,y,x), y-=x*(a/b);
5  }
6  long long int phi[N+1];
7  void phiTable(){
8    for(int i=1;i<=N;i++)phi[i]=i;
9    for(int i=1;i<=N;i++)for(x=i*2;x<=N;x+=i)
         phi[x]-=phi[i];
10 }
11 void all_divdown(const LL &n) {// all n/x
12   for(LL a=1;a<=n;a=n/(n/(a+1))){
```

```cpp
13      // dosomething;
14    }
15  }
16  const int MAXPRIME = 1000000;
17  int iscom[MAXPRIME], prime[MAXPRIME],
       primecnt;
18  int phi[MAXPRIME], mu[MAXPRIME];
19  void sieve(void){
20    memset(iscom,0,sizeof(iscom));
21    primecnt = 0;
22    phi[1] = mu[1] = 1;
23    for(int i=2;i<MAXPRIME;++i) {
24      if(!iscom[i]) {
25        prime[primecnt++] = i;
26        mu[i] = -1;
27        phi[i] = i-1;
28      }
29      for(int j=0;j<primecnt;++j) {
30        int k = i * prime[j];
31        if(k>=MAXPRIME) break;
32        iscom[k] = prime[j];
33        if(i%prime[j]==0) {
34          mu[k] = 0;
35          phi[k] = phi[i] * prime[j];
36          break;
37        } else {
38          mu[k] = -mu[i];
39          phi[k] = phi[i] * (prime[j]-1);
40        }
41      }
42    }
43  }
44
45  bool g_test(const LL &g, const LL &p, const
       vector<LL> &v) {
46    for(int i=0;i<v.size();++i)
47      if(modexp(g,(p-1)/v[i],p)==1)
48        return false;
49    return true;
50  }
51  LL primitive_root(const LL &p) {
52    if(p==2) return 1;
53    vector<LL> v;
54    Factor(p-1,v);
55    v.erase(unique(v.begin(), v.end()), v.end
         ());
56    for(LL g=2;g<p;++g)
57      if(g_test(g,p,v))
58        return g;
59    puts("primitive_root NOT FOUND");
60    return -1;
61  }
62  int Legendre(const LL &a, const LL &p) {
63      return modexp(a%p,(p-1)/2,p); }
64  LL inv(const LL &a, const LL &n) {
65    LL d,x,y;
66    gcd(a,n,d,x,y);
67    return d==1 ? (x+n)%n : -1;
68  }
69
70  int inv[maxN];
71  LL invtable(int n,LL P){
72    inv[1]=1;
73    for(int i=2;i<n;++i)
74      inv[i]=(P-(P/i))*inv[P%i]%P;
```

```cpp
75  }
76
77  LL log_mod(const LL &a, const LL &b, const
       LL &p) {
78    // a ^ x = b ( mod p )
79    int m=sqrt(p+.5), e=1;
80    LL v=inv(modexp(a,m,p), p);
81    map<LL,int> x;
82    x[1]=0;
83    for(int i=1;i<m;++i) {
84      e = LLmul(e,a,p);
85      if(!x.count(e)) x[e] = i;
86    }
87    for(int i=0;i<m;++i) {
88      if(x.count(b)) return i*m + x[b];
89      b = LLmul(b,v,p);
90    }
91    return -1;
92  }
93
94  LL Tonelli_Shanks(const LL &n, const LL &p)
       {
95    // x^2 = n ( mod p )
96    if(n==0) return 0;
97    if(Legendre(n,p)!=1) while(1) { puts("SQRT
           ROOT does not exist"); }
98    int S = 0;
99    LL Q = p-1;
100   while( !(Q&1) ) { Q>>=1; ++S; }
101   if(S==1) return modexp(n%p,(p+1)/4,p);
102   LL z = 2;
103   for(;Legendre(z,p)!=-1;++z)
104   LL c = modexp(z,Q,p);
105   LL R = modexp(n%p,(Q+1)/2,p), t = modexp(n
         %p,Q,p);
106   int M = S;
107   while(1) {
108     if(t==1) return R;
109     LL b = modexp(c,1L<<(M-i-1),p);
110     R = LLmul(R,b,p);
111     t = LLmul( LLmul(b,b,p), t, p);
112     c = LLmul(b,b,p);
113     M = i;
114   }
115   return -1;
116 }
117
118 template<typename T>
119 T Euler(T n){
120   T ans=n;
121   for(T i=2;i*i<=n;++i){
122     if(n%i==0){
123       ans=ans/i*(i-1);
124       while(n%i==0)n/=i;
125     }
126   }
127   if(n>1)ans=ans/n*(n-1);
128   return ans;
129 }
130
131 //Chinese_remainder_theorem
132 template<typename T>
133 T pow_mod(T n,T k,T m){
134   T ans=1;
135   for(n=(n>=m?n%m:n);k;k>>=1){
136     if(k&1)ans=ans*n%m;
```

```cpp
137     n=n*n%m;
138   }
139   return ans;
140 }
141 template<typename T>
142 T crt(vector<T> &m,vector<T> &a){
143   T M=1,tM,ans=0;
144   for(int i=0;i<(int)m.size();++i)M*=m[i];
145   for(int i=0;i<(int)a.size();++i){
146     tM=M/m[i];
147     ans=(ans+(a[i]*tM%M)*pow_mod(tM,Euler(m[
           i])-1,m[i])%M)%M;
148     /*如果m[i]是質數，Euler(m[i])-1=m[i]-2，
             就不用算Euler了*/
149   }
150   return ans;
151 }
152
153 //java code
154 //求sqrt(N)的連分數
155 public static void Pell(int n){
156   BigInteger N,p1,p2,q1,q2,a0,a1,a2,g1,g2,h1
         ,h2,p,q;
157   g1=q2=p1=BigInteger.ZERO;
158   h1=q1=p2=BigInteger.ONE;
159   a0=a1=BigInteger.valueOf((int)Math.sqrt
         (1.0*n));
160   BigInteger ans=a0.multiply(a0);
161   if(ans.equals(BigInteger.valueOf(n))){
162     System.out.println("No solution!");
163     return ;
164   }
165   while(true){
166     g2=a1.multiply(h1).substract(g1);
167     h2=N.substract(g2.pow(2)).divide(h1);
168     a2=g2.add(a0).divide(h2);
169     p=a1.multiply(p2).add(p1);
170     q=a1.multiply(q2).add(q1);
171     if(p.pow(2).substract(N.multiply(q.pow
           (2))).compareTo(BigInteger.ONE)==0)
         break;
172     g1=g2;h1=h2;a1=a2;
173     p1=p2;p2=p;
174     q1=q2;q2=q;
175   }
176   System.out.println(p+" "+q);
177 }
```

8.2   bit__set.cpp

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

```cpp
13     comb=((comb&~y)/x>>1)|y;
14   }
15 }
```

8.3   cantor__expansion.cpp

```cpp
1  int factorial[MAXN];
2  void init(){
3    factorial[0]=1;
4    for(int i=1;i<=MAXN;++i)factorial[i]=
         factorial[i-1]*i;
5  }
6  int encode(const vector<int> &s){
7    int n=s.size(),res=0;
8    for(int i=0;i<n;++i){
9      int t=0;
10     for(int j=i+1;j<n;++j)
11       if(s[j]<s[i])++t;
12     res+=t*factorial[n-i-1];
13   }
14   return res;
15 }
16 vector<int> decode(int a,int n){
17   vector<int> res;
18   vector<bool> vis(n,0);
19   for(int i=n-1;i>=0;--i){
20     int t=a/factorial[i],j;
21     for(j=0;j<n;++j)
22       if(!vis[j]){
23         if(t==0)break;
24         --t;
25       }
26     res.push_back(j);
27     vis[j]=1;
28     a%=factorial[i];
29   }
30   return res;
31 }
```

8.4   FFT.cpp

```cpp
1  template<typename T,typename VT=vector<
       complex<T> > >
2  struct FFT{
3    const T pi;
4    FFT(const T pi=acos((T)-1)):pi(pi){}
5    unsigned bit_reverse(unsigned a,int len){
6      a=((a&0x55555555U)<<1)|((a&0xAAAAAAAAU)
           >>1);
7      a=((a&0x33333333U)<<2)|((a&0xCCCCCCCCU)
           >>2);
8      a=((a&0x0F0F0F0FU)<<4)|((a&0xF0F0F0F0U)
           >>4);
9      a=((a&0x00FF00FFU)<<8)|((a&0xFF00FF00U)
           >>8);
10     a=((a&0x0000FFFFU)<<16)|((a&0xFFFF0000U)
           >>16);
11     return a>>(32-len);
12   }
```

```
13   void fft(bool is_inv,VT &in,VT &out,int N)
          {
14     int bitlen=__lg(N),num=is_inv?-1:1;
15     for(int i=0;i<N;++i)out[bit_reverse(i,
            bitlen)]=in[i];
16     for(int step=2;step<=N;step<<=1){
17       const int mh=step>>1;
18       for(int i=0;i<mh;++i){
19         complex<T> wi=exp(complex<T>(0,i*num
              *pi/mh));
20         for(int j=i;j<N;j+=step){
21           int k=j+mh;
22           complex<T> u=out[j],t=wi*out[k];
23           out[j]=u+t;
24           out[k]=u-t;
25         }
26       }
27     }
28     if(is_inv)for(int i=0;i<N;++i)out[i]/=N;
29   }
30 };
```

8.5 find_real_root.cpp

```
1  // an*x^n + ... + a1x + a0 = 0;
2  int sign(double x){
3    return x < -eps ? -1 : x > eps;
4  }
5
6  double get(const vector<double>&coef, double
        x){
7    double e = 1, s = 0;
8    for(auto i : coef) s += i*e, e *= x;
9    return s;
10 }
11
12 double find(const vector<double>&coef, int n
        , double lo, double hi){
13   double sign_lo, sign_hi;
14   if( !(sign_lo = sign(get(coef,lo))) )
        return lo;
15   if( !(sign_hi = sign(get(coef,hi))) )
        return hi;
16   if(sign_lo * sign_hi > 0) return INF;
17   for(int stp = 0; stp < 100 && hi - lo >
        eps; ++stp){
18     double m = (lo+hi)/2.0;
19     int sign_mid = sign(get(coef,m));
20     if(!sign_mid) return m;
21     if(sign_lo*sign_mid < 0) hi = m;
22     else lo = m;
23   }
24   return (lo+hi)/2.0;
25 }
26
27 vector<double> cal(vector<double>coef, int n
        ){
28   vector<double>res;
29   if(n == 1){
30     if(sign(coef[1])) res.pb(-coef[0]/coef
          [1]);
31     return res;
32   }
```

```
33   vector<double>dcoef(n);
34   for(int i = 0; i < n; ++i) dcoef[i] = coef
        [i+1]*(i+1);
35   vector<double>droot = cal(dcoef, n-1);
36   droot.insert(droot.begin(), -INF);
37   droot.pb(INF);
38   for(int i = 0; i+1 < droot.size(); ++i){
39     double tmp = find(coef, n, droot[i],
          droot[i+1]);
40     if(tmp < INF) res.pb(tmp);
41   }
42   return res;
43 }
44
45 int main () {
46   vector<double>ve;
47   vector<double>ans = cal(ve, n);
48   // 視情況把答案 +eps，避免 -0
49 }
```

8.6 FWT.cpp

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

8.7 LinearCongruence.cpp

```
1  pair<LL,LL> LinearCongruence(LL a[],LL b[],
        LL m[],int n) {
2    // a[i]*x = b[i] ( mod m[i] )
3    for(int i=0;i<n;++i) {
4      LL x, y, d = extgcd(a[i],m[i],x,y);
```

```
5      if(b[i]%d!=0) return make_pair(-1LL,0LL)
          ;
6      m[i] /= d;
7      b[i] = LLmul(b[i]/d,x,m[i]);
8    }
9    LL lastb = b[0], lastm = m[0];
10   for(int i=1;i<n;++i) {
11     LL x, y, d = extgcd(m[i],lastm,x,y);
12     if((lastb-b[i])%d!=0) return make_pair
          (-1LL,0LL);
13     lastb = LLmul((lastb-b[i])/d,x,(lastm/d
          )*m[i];
14     lastm = (lastm/d)*m[i];
15     lastb = (lastb+b[i])%lastm;
16   }
17   return make_pair(lastb<0?lastb+lastm:lastb
        ,lastm);
18 }
```

8.8 Lucas.cpp

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

8.9 Matrix.cpp

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

```
21         rev[i][j]=m[i][j]-a.m[i][j];
22     return rev;
23   }
24   matrix operator*(const matrix &a){
25     matrix rev(r,a.c);
26     matrix tmp(a.c,a.r);
27     for(int i=0;i<a.r;++i)
28       for(int j=0;j<a.c;++j)
29         tmp[j][i]=a.m[i][j];
30     for(int i=0;i<r;++i)
31       for(int j=0;j<a.c;++j)
32         for(int k=0;k<c;++k)
33           rev.m[i][j]+=m[i][k]*tmp[j][k];
34     return rev;
35   }
36   bool inverse(){
37     Matrix t(r,r+c);
38     for(int y=0;y<r;y++){
39       t.m[y][c+y] = 1;
40       for(int x=0;x<c;++x)
41         t.m[y][x]=m[y][x];
42     }
43     if( !t.gas() )
44       return false;
45     for(int y=0;y<r;y++)
46       for(int x=0;x<c;++x)
47         m[y][x]=t.m[y][c+x]/t.m[y][y];
48     return true;
49   }
50   T gas(){
51     vector<T> lazy(r,1);
52     bool sign=false;
53     for(int i=0;i<r;++i){
54       if( m[i][i]==0 ){
55         int j=i+1;
56         while(j<r&&!m[j][i])j++;
57         if(j==r)continue;
58         m[i].swap(m[j]);
59         sign=!sign;
60       }
61       for(int j=0;j<r;++j){
62         if(i==j)continue;
63         lazy[j]=lazy[j]*m[i][i];
64         T mx=m[j][i];
65         for(int k=0;k<c;++k)
66           m[j][k]=m[j][k]*m[i][i]-m[i][k]*mx
                ;
67       }
68     }
69     T det=sign?-1:1;
70     for(int i=0;i<r;++i){
71       det = det*m[i][i];
72       det = det/lazy[i];
73       for(auto &j:m[i])j/=lazy[i];
74     }
75     return det;
76   }
77 };
```

8.10 MillerRobin.cpp

```
1  LL LLmul(LL a, LL b, const LL &mod) {
2    LL ans=0;
```

```cpp
    while(b) {
        if(b&1) {
            ans+=a;
            if(ans>=mod) ans-=mod;
        }
        a<<=1, b>>=1;
        if(a>=mod) a-=mod;
    }
    return ans;
}
LL mod_mul(LL a,LL b,LL m){
    a%=m,b%=m;/* fast for m < 2^58 */
    LL y=(LL)((double)a*b/m+0.5);
    LL r=(a*b-y*m)%m;
    return r<0?r+m:r;
}
template<typename T>
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,
1795265022};//至少unsigned long long範圍
template<typename T>
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;
}
```

8.11 NTT.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=vector<T> >
struct NTT{
    const T P,G;
    NTT(T p=(1<<23)*7*17+1,T g=3):P(p),G(g){}
```

```cpp
unsigned bit_reverse(unsigned a,int len){
    //look FFT.cpp
}
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=__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)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;
    }
  }
};
```

8.12 Simpson.cpp

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

8.13 外星模運算.cpp

```cpp
//a[0]^(a[1]^a[2]^...)
#define maxn 1000000
int euler[maxn+5];
bool is_prime[maxn+5];
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);
            }
        }
    }
}
LL pow(LL a,LL b,LL mod){//a^b%mod
    LL ans=1;
    for(;b;a=a*a%mod,b>>=1)
        if(b&1)ans=ans*a%mod;
    return ans;
}
bool isless(LL *a,int n,int k){
    if(*a==1)return k>1;
    if(--n==0)return *a<k;
    int next=0;
    for(LL b=1;b<k;++next)
        b*=*a;
    return isless(a+1,n,next);
}
LL high_pow(LL *a,int n,LL mod){
    if(*a==1||--n==0)return *a%mod;
    int k=0,r=euler[mod];
    for(LL 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), t=(tmd-k+r)%r;
    return pow(*a,k+t,mod);
}
LL 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;
}
```

8.14 質因數分解.cpp

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

```cpp
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());
```

```cpp
68 }
69
70 void AllFactor(const LL &n,vector<LL> &v) {
71   vector<LL> tmp;
72   Factor(n,tmp);
73   v.clear();
74   v.push_back(1);
75   int len;
76   LL now=1;
77   for(int i=0;i<tmp.size();++i) {
78     if(i==0 || tmp[i]!=tmp[i-1]) {
79       len = v.size();
80       now = 1;
81     }
82     now*=tmp[i];
83     for(int j=0;j<len;++j)
84       v.push_back(v[j]*now);
85   }
86 }
```

9 other

9.1 WhatDay.cpp

```cpp
1 int whatday(int y,int m,int d){
2   if(m<=2)m+=12,--y;
3   if(y<1752||y==1752&&m<9||y==1752&&m==9&&d
       <3)
4     return (d+2*m+3*(m+1)/5+y+y/4+5)%7;
5   return (d+2*m+3*(m+1)/5+y+y/4-y/100+y/400)
       %7;
6 }
```

9.2 上下最大正方形.cpp

```cpp
1 void solve(int n,int a[],int b[]){// 1-base
2   int ans=0;
3   deque<int>da,db;
4   for(int l=1,r=1;r<=n;++r){
5     while(da.size()&&a[da.back()]>=a[r]){
6       da.pop_back();
7     }
8     da.push_back(r);
9     while(db.size()&&b[db.back()]>=b[r]){
10      db.pop_back();
11    }
12    db.push_back(r);
13    for(int d=a[da.front()]+b[db.front()];r-
        l+1>d;++l){
14      if(da.front()==l)da.pop_front();
15      if(db.front()==l)db.pop_front();
16      if(da.size()&&db.size()){
17        d=a[da.front()]+b[db.front()];
18      }
19    }
20    ans=max(ans,r-l+1);
21  }
22  printf("%d\n",ans);
```

```cpp
23 }
```

9.3 最大矩形.cpp

```cpp
1 LL max_rectangle(vector<int> s){
2   stack<pair<int,int> > st;
3   st.push(make_pair(-1,0));
4   s.push_back(0);
5   LL ans=0;
6   for(size_t i=0;i<s.size();++i){
7     int h=s[i];
8     pair<int,int > now=make_pair(h,i);
9     while(h<st.top().first){
10      now=st.top();
11      st.pop();
12      ans=max(ans,(LL)(i-now.second)*now.
          first);
13    }
14    if(h>st.top().first){
15      st.push(make_pair(h,now.second));
16    }
17  }
18  return ans;
19 }
```

10 String

10.1 AC 自動機.cpp

```cpp
1 template<char L='a',char R='z'>
2 class ac_automaton{
3   struct joe{
4     int next[R-L+1],fail,efl,ed,cnt_dp,vis;
5     joe():ed(0),cnt_dp(0),vis(0){
6       for(int i=0;i<=R-L;++i)next[i]=0;
7     }
8   };
9 public:
10  std::vector<joe> S;
11  std::vector<int> q;
12  int qs,qe,vt;
13  ac_automaton():S(1),qs(0),qe(0),vt(0){}
14  void clear(){
15    q.clear();
16    S.resize(1);
17    for(int i=0;i<=R-L;++i)S[0].next[i]=0;
18    S[0].cnt_dp=S[0].vis=qs=qe=vt=0;
19  }
20  void insert(const char *s){
21    int o=0;
22    for(int i=0,id;s[i];++i){
23      id=s[i]-L;
24      if(!S[o].next[id]){
25        S.push_back(joe());
26        S[o].next[id]=S.size()-1;
27      }
28      o=S[o].next[id];
29    }
```

```cpp
30    ++S[o].ed;
31  }
32  void build_fail(){
33    S[0].fail=S[0].efl=-1;
34    q.clear();
35    q.push_back(0);
36    ++qe;
37    while(qs!=qe){
38      int pa=q[qs++],id,t;
39      for(int i=0;i<=R-L;++i){
40        t=S[pa].next[i];
41        if(!t)continue;
42        id=S[pa].fail;
43        while(~id&&!S[id].next[i])id=S[id].
            fail;
44        S[t].fail=~id?S[id].next[i]:0;
45        S[t].efl=S[S[t].fail].ed?S[t].fail:S
            [S[t].fail].efl;
46        q.push_back(t);
47        ++qe;
48      }
49    }
50  }
51  /*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)*/
52  int match_0(const char *s){
53    int ans=0,id,p=0,i;
54    for(i=0;s[i];++i){
55      id=s[i]-L;
56      while(!S[p].next[id]&&p)p=S[p].fail;
57      if(!S[p].next[id])continue;
58      p=S[p].next[id];
59      ++S[p].cnt_dp;/*匹配成功則它所有後綴都
          可以被匹配(DP計算)*/
60    }
61    for(i=qe-1;i>=0;--i){
62      ans+=S[q[i]].cnt_dp*S[q[i]].ed;
63      if(~S[q[i]].fail)S[S[q[i]].fail].
          cnt_dp+=S[q[i]].cnt_dp;
64    }
65    return ans;
66  }
67  /*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)*/
68  int match_1(const char *s)const{
69    int ans=0,id,p=0,t;
70    for(int i=0;s[i];++i){
71      id=s[i]-L;
72      while(!S[p].next[id]&&p)p=S[p].fail;
73      if(!S[p].next[id])continue;
74      p=S[p].next[id];
75      if(S[p].ed)ans+=S[p].ed;
76      for(t=S[p].efl;~t;t=S[t].efl){
77        ans+=S[t].ed;/*因為都走efl邊所以保證
            匹配成功*/
78      }
79    }
80    return ans;
81  }
82  /*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))*/
83  int match_2(const char *s){
84    int ans=0,id,p=0,t;
85    ++vt;
```

```cpp
86    /*把戳記vt+=1，只要vt沒溢位，所有S[p].
        vis==vt就會變成false
87    這種利用vt的方法可以O(1)歸零vis陣列*/
88    for(int i=0;s[i];++i){
89      id=s[i]-L;
90      while(!S[p].next[id]&&p)p=S[p].fail;
91      if(!S[p].next[id])continue;
92      p=S[p].next[id];
93      if(S[p].ed&&S[p].vis!=vt){
94        S[p].vis=vt;
95        ans+=S[p].ed;
96      }
97      for(t=S[p].efl;~t&&S[t].vis!=vt;t=S[t
          ].efl){
98        S[t].vis=vt;
99        ans+=S[t].ed;/*因為都走efl邊所以保證
            匹配成功*/
100     }
101   }
102   return ans;
103 }
104 /*把AC自動機變成真的自動機*/
105 void evolution(){
106   for(qs=1;qs!=qe;){
107     int p=q[qs++];
108     for(int i=0;i<=R-L;++i)
109       if(S[p].next[i]==0)S[p].next[i]=S[S[
            p].fail].next[i];
110   }
111 }
112 };
```

10.2 hash.cpp

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

10.3 KMP.cpp

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

10.4 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;
  }
}
```

10.5 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
}
```

10.6 reverseBWT.cpp

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

10.7 suffix_array_lcp.cpp

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

```cpp
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;
}
```

10.8 Z.cpp

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

11 Tarjan

11.1 dominator_tree.cpp

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

11.2 tnfshb017_2_sat.cpp

```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], rv[MAXN2], vis_t;
int N,M;
void addedge(int s,int e){
  v[s].push_back(e);
  rv[e].push_back(s);
}
int scc[MAXN2];
bool vis[MAXN2]={false};
void dfs(vector<int> *uv,int n,int k=-1){
  vis[n]=true;
  for(int i=0;i<uv[n].size();++i)
    if(!vis[uv[n][i]])
      dfs(uv,uv[n][i],k);
  if(uv==v)vis_t.push_back(n);
  scc[n]=k;
}
void solve(){
  for(int i=1;i<=N;++i)
    if(!vis[i])dfs(v,i);
    if(!vis[n(i)])dfs(v,n(i));
```

```cpp
26      }
27      memset(vis,0,sizeof(vis));
28      int c=0;
29      for(int i=vis_t.size()-1;i>=0;--i)
30          if(!vis[vis_t[i]])
31              dfs(rv,vis_t[i],c++);
32  }
33  int main(){
34      int a,b;
35      scanf("%d%d",&N,&M);
36      for(int i=1;i<=N;++i){
37          // (A or B)&(!A & !B) A^B
38          a=i*2-1;
39          b=i*2;
40          addedge(n(a),b);
41          addedge(n(b),a);
42          addedge(a,n(b));
43          addedge(b,n(a));
44      }
45      while(M--){
46          scanf("%d%d",&a,&b);
47          a = a>0?a*2-1:-a*2;
48          b = b>0?b*2-1:-b*2;
49          // A or B
50          addedge(n(a),b);
51          addedge(n(b),a);
52      }
53      solve();
54      bool check=true;
55      for(int i=1;i<=2*N;++i)
56          if(scc[i]==scc[n(i)])
57              check=false;
58      if(check){
59          printf("%d\n",N);
60          for(int i=1;i<=2*N;i+=2){
61              if(scc[i]>scc[i+2*N]) putchar('+');
62              else putchar('-');
63          }
64          puts("");
65      }else puts("0");
66      return 0;
67  }
```

11.3 橋連通分量.cpp

```cpp
1   #define N 1005
2   struct edge{
3       int u,v;
4       bool is_bridge;
5       edge(int u=0,int v=0):u(u),v(v),is_bridge
            (0){}
6   };
7   vector<edge> E;
8   vector<int> G[N];// 1-base
9   int low[N],vis[N],Time;
10  int bcc_id[N],bridge_cnt,bcc_cnt;// 1-base
11  int st[N],top;//BCC用
12  inline void add_edge(int u,int v){
13      G[u].push_back(E.size());
14      E.push_back(edge(u,v));
15      G[v].push_back(E.size());
16      E.push_back(edge(v,u));
17  }
```

```cpp
18  void dfs(int u,int re=-1){//u當前點，re為u連
            接前一個點的邊
19      int v;
20      low[u]=vis[u]=++Time;
21      st[top++]=u;
22      for(size_t i=0;i<G[u].size();++i){
23          int e=G[u][i];v=E[e].v;
24          if(!vis[v]){
25              dfs(v,e^1);//e^1反向邊
26              low[u]=min(low[u],low[v]);
27              if(vis[u]<low[v]){
28                  E[e].is_bridge=E[e^1].is_bridge=1;
29                  ++bridge_cnt;
30              }
31          }else if(vis[v]<vis[u]&&e!=re)
32              low[u]=min(low[u],vis[v]);
33      }
34      if(vis[u]==low[u]){//處理BCC
35          ++bcc_cnt;// 1-base
36          do bcc_id[v=st[--top]]=bcc_cnt;//每個點
                所在的BCC
37          while(v!=u);
38      }
39  }
40  inline void bcc_init(int n){
41      Time=bcc_cnt=bridge_cnt=top=0;
42      E.clear();
43      for(int i=1;i<=n;++i){
44          G[i].clear();
45          vis[i]=bcc_id[i]=0;
46      }
47  }
```

11.4 雙連通分量 & 割點.cpp

```cpp
1   #define N 1005
2   vector<int> G[N];// 1-base
3   vector<int> bcc[N];//存每塊雙連通分量的點
4   int low[N],vis[N],Time;
5   int bcc_id[N],bcc_cnt;// 1-base
6   bool is_cut[N];//是否為割點
7   int st[N],top;
8   void dfs(int u,int pa=-1){//u當前點，pa父親
9       int v,child=0;
10      low[u]=vis[u]=++Time;
11      st[top++]=u;
12      for(size_t i=0;i<G[u].size();++i){
13          if(!vis[v=G[u][i]]){
14              dfs(v,u),++child;
15              low[u]=min(low[u],low[v]);
16              if(vis[u]<=low[v]){
17                  is_cut[u]=1;
18                  bcc[++bcc_cnt].clear();
19                  int t;
20                  do{
21                      bcc_id[t=st[--top]]=bcc_cnt;
22                      bcc[bcc_cnt].push_back(t);
23                  }while(t!=v);
24                  bcc_id[u]=bcc_cnt;
25                  bcc[bcc_cnt].push_back(u);
26              }
```

```cpp
27          }else if(vis[v]<vis[u]&&v!=pa)//反向邊
28              low[u]=min(low[u],vis[v]);
29      }
30      if(pa==-1&&child<2)is_cut[u]=0;//u是dfs樹
            的根要特判
31  }
32  inline void bcc_init(int n){
33      Time=bcc_cnt=top=0;
34      for(int i=1;i<=n;++i){
35          G[i].clear();
36          is_cut[i]=vis[i]=bcc_id[i]=0;
37      }
38  }
```

12 Tree_problem

12.1 HeavyLight.cpp

```cpp
1   #include<vector>
2   #define MAXN 100005
3   int siz[MAXN],max_son[MAXN],pa[MAXN],dep[
        MAXN];
4   int link_top[MAXN],link[MAXN],cnt;
5   vector<int> G[MAXN];
6   void find_max_son(int u){
7       siz[u]=1;
8       max_son[u]=-1;
9       for(auto v:G[u]){
10          if(v==pa[u])continue;
11          pa[v]=u;
12          dep[v]=dep[u]+1;
13          find_max_son(v);
14          if(max_son[u]==-1||siz[v]>siz[max_son[u
                ]])max_son[u]=v;
15          siz[u]+=siz[v];
16      }
17  }
18  void build_link(int u,int top){
19      link[u]=++cnt;
20      link_top[u]=top;
21      if(max_son[u]==-1)return;
22      build_link(max_son[u],top);
23      for(auto v:G[u]){
24          if(v==max_son[u]||v==pa[u])continue;
25          build_link(v,v);
26      }
27  }
28  int find_lca(int a,int b){
29      //求LCA，可以在過程中對區間進行處理
30      int ta=link_top[a],tb=link_top[b];
31      while(ta!=tb){
32          if(dep[ta]<dep[tb]){
33              swap(ta,tb);
34              swap(a,b);
35          }
36          //這裡可以對a所在的鏈做區間處理
37          //區間為(link[ta],link[a])
38          ta=link_top[a=pa[ta]];
39      }
```

```cpp
40      //最後a,b會在同一條鏈，若a!=b還要再進行一
            次區間處理
41      return dep[a]<dep[b]?a:b;
42  }
```

12.2 LCA.cpp

```cpp
1   #define MAXN 100000
2   #define MAX_LOG 17
3   int pa[MAX_LOG+1][MAXN+5];
4   int dep[MAXN+5];
5   vector<int> G[MAXN+5];
6   void dfs(int x,int p){//dfs(1,-1)
7       pa[0][x]=p;
8       for(int i=0;i+1<MAX_LOG;++i)pa[i+1][x]=pa[
            i][pa[i][x]];
9       for(auto &i:G[x]){
10          if(i==p)continue;
11          dep[i]=dep[x]+1;
12          dfs(i,x);
13      }
14  }
15  inline int jump(int x,int d){
16      for(int i=0;i<d;++i)if((x>>i)&1)x=pa[k][x];
17      return x;
18  }
19  inline int find_lca(int a,int b){
20      if(dep[a]>dep[b])swap(a,b);
21      b=jump(b,dep[b]-dep[a]);
22      if(a==b)return a;
23      for(int i=MAX_LOG;i>=0;--i){
24          if(pa[i][a]!=pa[i][b]){
25              a=pa[i][a];
26              b=pa[i][b];
27          }
28      }
29      return pa[0][a];
30  }
```

12.3 link_cut_tree.cpp

```cpp
1   struct splay_tree{
2       int ch[2],pa;//子節點跟父母
3       bool rev;//反轉的懶惰標記
4       splay_tree():pa(0),rev(0){ch[0]=ch[1]=0;}
5   };
6   vector<splay_tree> node;
7   //有的時候用vector會TLE，要注意
8   //這邊以node[0]作為null節點
9   bool isroot(int x){//判斷是否為這棵splay
        tree的根
10      return node[node[x].pa].ch[0]!=x&&node[
            node[x].pa].ch[1]!=x;
11  }
12  void down(int x){//懶惰標記下推
13      if(node[x].rev){
14          if(node[x].ch[0])node[node[x].ch[0]].rev
                ^=1;
```

```cpp
15    if(node[x].ch[1])node[node[x].ch[1]].rev
          ^=1;
16    std::swap(node[x].ch[0],node[x].ch[1]);
17    node[x].rev^=1;
18  }
19 }
20 void push_down(int x){//將所有祖先的懶惰標記
        下推
21   if(!isroot(x))push_down(node[x].pa);
22   down(x);
23 }
24 void up(int x){}//將子節點的資訊向上更新
25 void rotate(int x){//旋轉，會自行判斷轉的方
        向
26   int y=node[x].pa,z=node[y].pa,d=(node[y].
        ch[1]==x);
27   node[x].pa=z;
28   if(!isroot(y))node[z].ch[node[z].ch[1]==y
        ]=x;
29   node[y].ch[d]=node[x].ch[d^1];
30   node[node[y].ch[d]].pa=y;
31   node[y].pa=x,node[x].ch[d^1]=y;
32   up(y),up(x);
33 }
34 void splay(int x){//將節點x伸展到所在splay
        tree的根
35   push_down(x);
36   while(!isroot(x)){
37     int y=node[x].pa;
38     if(!isroot(y)){
39       int z=node[y].pa;
40       if((node[z].ch[0]==y)^(node[y].ch[0]==
            x))rotate(y);
41       else rotate(x);
42     }
43     rotate(x);
44   }
45 }
46 int access(int x){
47   int last=0;
48   while(x){
49     splay(x);
50     node[x].ch[1]=last;
51     up(x);
52     last=x;
53     x=node[x].pa;
54   }
55   return last;//回傳access後splay tree的根
56 }
57 void access(int x,bool is=0){//is=0就是一般
        的access
58   int last=0;
59   while(x){
60     splay(x);
61     if(is&&!node[x].pa){
62       //printf("%d\n",max(node[last].ma,node
            [node[x].ch[1]].ma));
63     }
64     node[x].ch[1]=last;
65     up(x);
66     last=x;
67     x=node[x].pa;
68   }
69 }
```

```cpp
70 void query_edge(int u,int v){
71   access(u);
72   access(v,1);
73 }
74 void make_root(int x){
75   access(x),splay(x);
76   node[x].rev^=1;
77 }
78 void make_root(int x){
79   node[access(x)].rev^=1;
80   splay(x);
81 }
82 void cut(int x,int y){
83   make_root(x);
84   access(y);
85   splay(y);
86   node[y].ch[0]=0;
87   node[x].pa=0;
88 }
89 void cut_parents(int x){
90   access(x);
91   splay(x);
92   node[node[x].ch[0]].pa=0;
93   node[x].ch[0]=0;
94 }
95 void link(int x,int y){
96   make_root(x);
97   node[x].pa=y;
98 }
99 int find_root(int x){
100   x=access(x);
101   while(node[x].ch[0])x=node[x].ch[0];
102   splay(x);
103   return x;
104 }
105 int query(int u,int v){
106 //傳回uv路徑splay tree的根結點
107 //這種寫法無法求LCA
108   make_root(u);
109   return access(v);
110 }
111 int query_lca(int u,int v){
112 //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
113   access(u);
114   int lca=access(v);
115   splay(u);
116   if(u==lca){
117     //return node[lca].data+node[node[lca].
            ch[1]].sum
118   }else{
119     //return node[lca].data+node[node[lca].
            ch[1]].sum+node[u].sum
120   }
121 }
122 struct EDGE{
123   int a,b,w;
124 }e[10005];
125 int n;
126 vector<pair<int ,int > >G[10005];
127 //first表示子節點，second表示邊的編號
128 int pa[10005],edge_node[10005];
129 //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
```

```cpp
130 void bfs(int root){
131 //在建構的時候把每個點都設成一個splay tree，
        不會壞掉
132   queue<int > q;
133   for(int i=1;i<=n;++i)pa[i]=0;
134   q.push(root);
135   while(q.size()){
136     int u=q.front();
137     q.pop();
138     for(int i=0;i<(int)G[u].size();++i){
139       int v=G[u][i].first;
140       if(v!=pa[u]){
141         pa[v]=u;
142         node[v].pa=u;
143         node[v].data=e[G[u][i].second].w;
144         edge_node[G[u][i].second]=v;
145         up(v);
146         q.push(v);
147       }
148     }
149   }
150 }
151 void change(int x,int b){
152   splay(x);
153   //node[x].data=b;
154   up(x);
155 }
```

12.4 POJ__tree.cpp

```cpp
1 #include<bits/stdc++.h>
2 using namespace std;
3 #define MAXN 10005
4 int n,k;
5 vector<pair<int,int> >g[MAXN];
6 int size[MAXN];
7 bool vis[MAXN];
8 inline void init(){
9   for(int i=0;i<=n;++i){
10     g[i].clear();
11     vis[i]=0;
12   }
13 }
14 void get_dis(vector<int> &dis,int u,int pa,
      int d){
15   dis.push_back(d);
16   for(size_t i=0;i<g[u].size();++i){
17     int v=g[u][i].first,w=g[u][i].second;
18     if(v!=pa&&!vis[v])get_dis(dis,v,u,d+w);
19   }
20 }
21 vector<int> dis;//這東西如果放在函數裡會TLE
22 int cal(int u,int d){
23   dis.clear();
24   get_dis(dis,u,-1,d);
25   sort(dis.begin(),dis.end());
26   int l=0,r=dis.size()-1,res=0;
27   while(l<r){
28     while(l<r&&dis[l]+dis[r]>k)--r;
29     res+=r-(l++);
30   }
31   return res;
```

```cpp
32 }
33 pair<int,int> tree_centroid(int u,int pa,
      const int sz){
34   size[u]=1;//找樹重心，second是重心
35   pair<int,int> res(INT_MAX,-1);
36   int ma=0;
37   for(size_t i=0;i<g[u].size();++i){
38     int v=g[u][i].first;
39     if(v==pa||vis[v])continue;
40     res=min(res,tree_centroid(v,u,sz));
41     size[u]+=size[v];
42     ma=max(ma,size[v]);
43   }
44   ma=max(ma,sz-size[u]);
45   return min(res,make_pair(ma,u));
46 }
47 int tree_DC(int u,int sz){
48   int center=tree_centroid(u,-1,sz).second;
49   int ans=cal(center,0);
50   vis[center]=1;
51   for(size_t i=0;i<g[center].size();++i){
52     int v=g[center][i].first,w=g[center][i].
          second;
53     if(vis[v])continue;
54     ans-=cal(v,w);
55     ans+=tree_DC(v,size[v]);
56   }
57   return ans;
58 }
59 int main(){
60   while(scanf("%d%d",&n,&k),n||k){
61     init();
62     for(int i=1;i<n;++i){
63       int u,v,w;
64       scanf("%d%d%d",&u,&v,&w);
65       g[u].push_back(make_pair(v,w));
66       g[v].push_back(make_pair(u,w));
67     }
68     printf("%d\n",tree_DC(1,n));
69   }
70   return 0;
71 }
```

13 zformula

13.1 formula.tex

13.1.1 Pick 公式

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

13.1.2 圖論

1. $V - E + F = 2$
2. 對於平面圖，$F = E - V + n + 1$，n 是連通分量
3. 對於平面圖，$E < 3V - 6$
4. 對於連通圖 G，最大獨立點集的大小設為 I(G)，最大匹配大小設為 M(G)，最小點覆蓋設為 Cv(G)，最小邊覆蓋設為 Ce(G)。對於任意連通圖：

(a) $I(G) + Cv(G) = |V|$
(b) $M(G) + Ce(G) = |V|$

5. 對於連通二分圖：

(a) $I(G) = Cv(G)$
(b) $M(G) = Ce(G)$

6. 最大權閉合圖：

(a) $C(u, V) = \infty, (u, v) \in E$
(b) $C(S, v) = W_v, W_v > 0$
(c) $C(v, T) = -W_v, W_v < 0$

7. 最大密度子圖：

(a) $C(u, v) = 1, (u, v) \in E$
(b) $C(S, v) = U_v, v \in V$
(c) $C(v, T) = U + 2g - d_v, v \in V$

8. 弦圖：

(a) 完美消除序列從後往前依次給每個點染色，給每個點染上可以染的最小顏色
(b) 最大團大小 = 色數
(c) 最大獨立集：完美消除序列從前往後能選就選
(d) 最小團覆蓋：最大獨立集的點和他延伸的邊構成
(e) 區間圖是弦圖
(f) 區間圖的完美消除序列：將區間按造又端點由小到大排序
(g) 區間圖染色：用線段樹做

```
1  double l=0,=m,stop=1.0/n/n;
2  while(r-l>=stop){
3    double(mid);
4    if((n*m-sol.maxFlow(s,t))/2>eps)l=mid;
5    else r=mid;
6  }
7  build(l);
8  sol.maxFlow(s,t);
9  vector<int> ans;
10 for(int i=1;i<=n;++i)
11   if(sol.vis[i])ans.push_back(i);
```

13.1.3　學長公式

1. $\sum_{d|n} \phi(n) = n$
2. $g(n) = \sum_{d|n} f(d) => f(n) = \sum_{d|n} \mu(d) \times g(n/d)$
3. Harmonic series $H_n = \ln(n) + \gamma + 1/(2n) - 1/(12n^2) + 1/(120n^4)$
4. $\gamma = 0.5772156649015328606065120900824024310421 5$
5. 格雷碼 = $n \oplus (n >> 1)$
6. $SG(A + B) = SG(A) \oplus SG(B)$
7. 選轉矩陣 $M(\theta) = \begin{pmatrix} cos\theta & -sin\theta \\ sin\theta & cos\theta \end{pmatrix}$

13.1.4　基本數論

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

13.1.5　排組公式

1. k 卡特蘭 $\frac{C_n^{kn}}{n(k-1)+1} \cdot C_m^n = \frac{n!}{m!(n-m)!}$
2. $H(n, m) \cong x_1 + x_2 \ldots + x_n = k, num = C_k^{n+k-1}$
3. Stirling number of 2nd, n 人分 k 組方法數目

(a) $S(0, 0) = S(n, n) = 1$
(b) $S(n, 0) = 0$
(c) $S(n, k) = kS(n - 1, k) + S(n - 1, k - 1)$

4. Bell number, n 人分任意多組方法數目

(a) $B_0 = 1$
(b) $B_n = \sum_{i=0}^{n} S(n, i)$
(c) $B_{n+1} = \sum_{k=0}^{n} C_n^k B_k$
(d) $B_{p+n} \equiv B_n + B_{n+1} mod p$, p is prime
(e) $B_{p^m+n} \equiv mB_n + B_{n+1} mod p$, p is prime
(f) From $B_0 : 1, 1, 2, 5, 15, 52,$ $203, 877, 4140, 21147, 115975$

5. Derangement, 錯排, 沒有人在自己位置上

(a) $D_n = n!(1 - \frac{1}{1!} + \frac{1}{2!} - \frac{1}{3!} \ldots + (-1)^n \frac{1}{n!})$
(b) $D_n = (n - 1)(D_{n-1} + D_{n-2}), D_0 = 1, D_1 = 0$
(c) From $D_0 : 1, 0, 1, 2, 9, 44,$ $265, 1854, 14833, 133496$

6. Binomial Equality

(a) $\sum_k \binom{r}{m+k}\binom{s}{n-k} = \binom{r+s}{m+n}$
(b) $\sum_k \binom{l}{m+k}\binom{s}{n+k} = \binom{l+s}{l-m+n}$
(c) $\sum_k \binom{l}{m+k}\binom{s+k}{n}(-1)^k = (-1)^{l+m}\binom{s-m}{n-l}$
(d) $\sum_{k \le l} \binom{l-k}{m}\binom{s}{k-n}(-1)^k = (-1)^{l+m}\binom{s-m-1}{l-n-m}$
(e) $\sum_{0 \le k \le l} \binom{l-k}{m}\binom{q+k}{n} = \binom{l+q+1}{m+n+1}$
(f) $\binom{r}{k} = (-1)^k \binom{k-r-1}{k}$
(g) $\binom{r}{m}\binom{m}{k} = \binom{r}{k}\binom{r-k}{m-k}$
(h) $\sum_{k \le n} \binom{r+k}{k} = \binom{r+n+1}{n}$
(i) $\sum_{0 \le k \le n} \binom{k}{m} = \binom{n+1}{m+1}$
(j) $\sum_{k \le m} \binom{m+r}{k} x^k y^k = \sum_{k \le m} \binom{-r}{k}(-x)^k(x+y)^{m-k}$

13.1.6　冪次, 冪次和

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

13.1.7　Burnside's lemma

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

13.1.8　Count on a tree

1. Rooted tree: $s_{n+1} = \frac{1}{n}\sum_{i=1}^{n}(i \times a_i \times \sum_{j=1}^{\lfloor n/i \rfloor} a_{n+1-i \times j})$
2. Unrooted tree:

(a) Odd: $a_n - \sum_{i=1}^{n/2} a_i a_{n-i}$
(b) Even: $Odd + \frac{1}{2}a_{n/2}(a_{n/2} + 1)$

3. Spanning Tree

(a) 完全圖 $n^n - 2$
(b) 一般圖 (Kirchhoff's theorem)$M[i][i] = degree(V_i), M[i][j] = -1$,if have $E(i, j)$,0 if no edge. delete any one row and col in A, $ans = det(A)$

13.2　java.tex

13.2.1　文件操作

```
1  import java.io.*;
2  import java.util.*;
3  import java.math.*;
4  import java.text.*;
5
6  public class Main
7  {
8
9    public static void main(String args[])
         throws FileNotFoundException,
         IOException
10   {
       Scanner sc = new Scanner(new FileReader(
         "a.in"));
       PrintWriter pw = new PrintWriter(new
         FileWriter("a.out"));
       int n,m;
       n=sc.nextInt();//读入下一个INT
       m=sc.nextInt();

       for(ci=1; ci<=c; ++ci)
       {
         pw.println("Case #"+ci+": easy for
           output");
       }
```
```
22     pw.close();//关闭流并释放，这个很重要，
         否则是没有输出的
23     sc.close();//关闭流并释放
24   }
25 }
```

13.2.2　优先队列

```
1  PriorityQueue queue = new PriorityQueue( 1,
       new Comparator()
2  {
3    public int compare( Point a, Point b )
4    {
5    if( a.x < b.x || a.x == b.x && a.y < b.y )
6      return -1;
7    else if( a.x == b.x && a.y == b.y )
8      return 0;
9    else
10     return 1;
11   }
12 });
```

13.2.3　Map

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

13.2.4　sort

```
1  static class cmp implements Comparator
2  {
3    public int compare(Object o1,Object o2)
4    {
5    BigInteger b1=(BigInteger)o1;
6    BigInteger b2=(BigInteger)o2;
7    return b1.compareTo(b2);
8    }
9  }
10 public static void main(String[] args)
       throws IOException
11 {
12   Scanner cin = new Scanner(System.in);
13   int n;
14   n=cin.nextInt();
15   BigInteger[] seg = new BigInteger[n];
16   for (int i=0;i<n;i++)
17     seg[i]=cin.nextBigInteger();
18   Arrays.sort(seg,new cmp());
19 }
```

ACM ICPC Team Reference - NTHU Jinkela

Contents