1   Computational_Geometr

1.1   Geometry

```cpp
const double PI=atan2(0.0,-1.0);
template<typename T>
struct point{
  T x,y;
  point(){}
  point(const T&x,const T&y):x(x),y(y){}
  point operator+(const point &b)const{
    return point(x+b.x,y+b.y); }
  point operator-(const point &b)const{
    return point(x-b.x,y-b.y); }
  point operator*(const T &b)const{
    return point(x*b,y*b); }
  point operator/(const T &b)const{
    return point(x/b,y/b); }
  bool operator==(const point &b)const{
    return x==b.x&&y==b.y; }
  T dot(const point &b)const{
    return x*b.x+y*b.y; }
  T cross(const point &b)const{
    return x*b.y-y*b.x; }
  point normal()const{//求法向量
    return point(-y,x); }
  T abs2()const{//向量長度的平方
    return dot(*this); }
  T rad(const point &b)const{//兩向量的弧度
    return fabs(atan2(fabs(cross(b)),dot(b))); }
  T getA()const{//對x軸的弧度
    T A=atan2(y,x);//超過180度會變負的
    if(A<=-PI/2)A+=PI*2;
    return A;
  }
};
template<typename T>
struct line{
  line(){}
  point<T> p1,p2;
  T a,b,c;//ax+by+c=0
  line(const point<T>&x,const point<T>&y):p1
    (x),p2(y){}
  void pton(){//轉成一般式
    a=p1.y-p2.y;
    b=p2.x-p1.x;
    c=-a*p1.x-b*p1.y;
  }
  T ori(const point<T> &p)const{//點和有向直
    線的關係, >0左邊, =0在線上<0右邊
    return (p2-p1).cross(p-p1);
  }
  T btw(const point<T> &p)const{//點投影落在
    線段上<=0
    return (p1-p).dot(p2-p);
  }
  bool point_on_segment(const point<T>&p)
    const{//點是否在線段上
    return ori(p)==0&&btw(p)<=0;
  }
```

```cpp
  T dis2(const point<T> &p,bool is_segment
    =0)const{//點跟直線/線段的距離平方
    point<T> v=p2-p1,v1=p-p1;
    if(is_segment){
      point<T> v2=p-p2;
      if(v.dot(v1)<=0)return v1.abs2();
      if(v.dot(v2)>=0)return v2.abs2();
    }
    T tmp=v.cross(v1);
    return tmp*tmp/v.abs2();
  }
  T seg_dis2(const line<T> &l)const{//兩線段
    距離平方
    return min({dis2(l.p1,1),dis2(l.p2,1),l.
      dis2(p1,1),l.dis2(p2,1)});
  }
  point<T> projection(const point<T> &p)
    const{//點對直線的投影
    point<T> n=(p2-p1).normal();
    return p-n*(p-p1).dot(n)/n.abs2();
  }
  point<T> mirror(const point<T> &p)const{
    //點對直線的鏡射, 要先呼叫pton轉成一般式
    point<T> R;
    T d=a*a+b*b;
    R.x=(b*b*p.x-a*a*p.x-2*a*b*p.y-2*a*c)/d;
    R.y=(a*a*p.y-b*b*p.y-2*a*b*p.x-2*b*c)/d;
    return R;
  }
  bool equal(const line &l)const{//直線相等
    return ori(l.p1)==0&&ori(l.p2)==0;
  }
  bool parallel(const line &l)const{
    return (p1-p2).cross(l.p1-l.p2)==0;
  }
  bool cross_seg(const line &l)const{
    return (p2-p1).cross(l.p1-p1)*(p2-p1).
      cross(l.p2-p1)<=0;//直線是否交線段
  }
  int line_intersect(const line &l)const{//
    直線相交情況, -1無限多點, 1交於一點、0
    不相交
    return parallel(l)?(ori(l.p1)==0?-1:0)
      :1;
  }
  int seg_intersect(const line &l)const{
    T c1=ori(l.p1), c2=ori(l.p2);
    T c3=l.ori(p1), c4=l.ori(p2);
    if(c1==0&&c2==0){//共線
      bool b1=btw(l.p1)>=0,b2=btw(l.p2)>=0;
      T a3=l.btw(p1),a4=l.btw(p2);
      if(b1&&b2&&a3>=0&&a4>=0) return 2;
      if(b1&&b2&&a3>=0&&a4>=0) return 3;
      if(b1&&b2&&a3>=0&&a4>=0) return 0;
      return -1;//無限交點
    }else if(c1*c2<=0&&c3*c4<=0)return 1;
    return 0;//不相交
  }
  point<T> line_intersection(const line &l)
    const{/*直線交點*/
    point<T> a=p2-p1,b=l.p2-l.p1,s=l.p1-p1;
    //if(a.cross(b)==0)return INF;
    return p1+a*(s.cross(b)/a.cross(b));
  }
```

```cpp
  point<T> seg_intersection(const line &l)
    const{//線段交點
    int res=seg_intersect(l);
    if(res<=0) assert(0);
    if(res==2) return p1;
    if(res==3) return p2;
    return line_intersection(l);
  }
};
template<typename T>
struct polygon{
  polygon(){}
  vector<point<T> > p;//逆時針順序
  T area()const{//面積
    T ans=0;
    for(int i=p.size()-1,j=0;j<(int)p.size()
      ;i=j++)
      ans+=p[i].cross(p[j]);
    return ans/2;
  }
  point<T> center_of_mass()const{//重心
    T cx=0,cy=0,w=0;
    for(int i=p.size()-1,j=0;j<(int)p.size()
      ;i=j++){
      T a=p[i].cross(p[j]);
      cx+=(p[i].x+p[j].x)*a;
      cy+=(p[i].y+p[j].y)*a;
      w+=a;
    }
    return point<T>(cx/3/w,cy/3/w);
  }
  char ahas(const point<T>& t)const{//點是否
    在簡單多邊形內, 是的話回傳1、在邊上回
    傳-1、否則回傳0
    bool c=0;
    for(int i=0,j=p.size()-1;i<p.size();j=i
      ++)
      if(line<T>(p[i],p[j]).point_on_segment
        (t))return -1;
      else if((p[i].y>t.y)!=(p[j].y>t.y)&&
        t.x<(p[j].x-p[i].x)*(t.y-p[i].y)/(p[j
        ].y-p[i].y)+p[i].x)
        c=!c;
    return c;
  }
  char point_in_convex(const point<T>&x)
    const{
    int l=1,r=(int)p.size()-2;
    while(l<=r){//點是否在凸多邊形內, 是的話
      回傳1、在邊上回傳-1、否則回傳0
      int mid=(l+r)/2;
      T a1=(p[mid]-p[0]).cross(x-p[0]);
      T a2=(p[mid+1]-p[0]).cross(x-p[0]);
      if(a1>=0&&a2<=0){
        T res=(p[mid+1]-p[mid]).cross(x-p[
          mid]);
        return res>0?1:(res>=0?-1:0);
      }else if(a1<0)r=mid-1;
      else l=mid+1;
    }
    return 0;
  }
  vector<T> getA()const{//凸包邊對x軸的夾角
    vector<T>res;//一定是遞增的
```

```cpp
    for(size_t i=0;i<p.size();++i)
      res.push_back((p[(i+1)%p.size()]-p[i])
        .getA());
    return res;
  }
  bool line_intersect(const vector<T>&A,
    const line<T> &l)const{//O(logN)
    int f1=upper_bound(A.begin(),A.end(),(l.
      p1-l.p2).getA())-A.begin();
    int f2=upper_bound(A.begin(),A.end(),(l.
      p2-l.p1).getA())-A.begin();
    return l.cross_seg(line<T>(p[f1],p[f2]))
      ;
  }
  polygon cut(const line<T> &l)const{//凸包
    對直線切割, 得到直線L左側的凸包
    polygon ans;
    for(int n=p.size(),i=n-1,j=0;j<n;i=j++){
      if(l.ori(p[i])>=0){
        ans.p.push_back(p[i]);
        if(l.ori(p[j])<0)
          ans.p.push_back(l.
            line_intersection(line<T>(p[i
            ],p[j])));
      }else if(l.ori(p[j])>0)
        ans.p.push_back(l.line_intersection(
          line<T>(p[i],p[j])));
    }
    return ans;
  }
  static bool graham_cmp(const point<T>& a,
    const point<T>& b){//凸包排序函數
    return (a.x<b.x)||(a.x==b.x&&a.y<b.y);
  }
  void graham(vector<point<T> > &s){//凸包
    sort(s.begin(),s.end(),graham_cmp);
    p.resize(s.size()+1);
    int m=0;
    for(size_t i=0;i<s.size();++i){
      while(m>=2&&(p[m-1]-p[m-2]).cross(s[i
        ]-p[m-2])<=0)--m;
      p[m++]=s[i];
    }
    for(int i=s.size()-2,t=m+1;i>=0;--i){
      while(m>=t&&(p[m-1]-p[m-2]).cross(s[i
        ]-p[m-2])<=0)--m;
      p[m++]=s[i];
    }
    if(s.size()>1)--m;
    p.resize(m);
  }
  T diam(){//直徑
    int n=p.size(),t=1;
    T ans=0;p.push_back(p[0]);
    for(int i=0;i<n;i++){
      point<T> now=p[i+1]-p[i];
      while(now.cross(p[t+1]-p[i])>now.cross
        (p[t]-p[i]))t=(t+1)%n;
      ans=max(ans,(p[i]-p[t]).abs2());
    }
    return p.pop_back(),ans;
  }
  T min_cover_rectangle(){//最小覆蓋矩形
    int n=p.size(),t=1,r=1,l;
```

```cpp
212    if(n<3)return 0;//也可以做最小周長矩形
213    T ans=1e99;p.push_back(p[0]);
214    for(int i=0;i<n;i++){
215      point<T> now=p[i+1]-p[i];
216      while(now.cross(p[t+1]-p[i])>now.cross
           (p[t]-p[i]))t=(t+1)%n;
217      while(now.dot(p[r+1]-p[i])>now.dot(p[r
           ]-p[i]))r=(r+1)%n;
218      if(!i)l=r;
219      while(now.dot(p[l+1]-p[i])<=now.dot(p[
           l]-p[i]))l=(l+1)%n;
220      T d=now.abs2();
221      T tmp=now.cross(p[t]-p[i])*(now.dot(p[
           r]-p[i])-now.dot(p[l]-p[i]))/d;
222      ans=min(ans,tmp);
223    }
224    return p.pop_back(),ans;
225  }
226  T max_triangle(){//最大內接三角形
227    int n=p.size(),a=1,b=2;
228    if(n<3)return 0;
229    T ans=0,tmp;p.push_back(p[0]);
230    for(int i=0;i<n;++i){
231      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;
232      ans=max(ans,tmp);
233      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;
234      ans=max(ans,tmp);
235    }
236    return p.pop_back(),ans/2;
237  }
238  T dis2(polygon &pl){//凸包最近距離平方
239    vector<point<T> > &P=p,&Q=pl.p;
240    int n=P.size(),m=Q.size(),l=0,r=0;
241  for(int i=0;i<n;++i)if(P[i].y<P[l].y)l=i;
242  for(int i=0;i<m;++i)if(Q[i].y<Q[r].y)r=i;
243    P.push_back(P[0]),Q.push_back(Q[0]);
244    T ans=1e99;
245    for(int i=0;i<n;++i){
246      while((P[l]-P[l+1]).cross(Q[r+1]-Q[r])
           <0)r=(r+1)%m;
247      ans=min(ans,line<T>(P[l],P[l+1]).
           seg_dis2(line<T>(Q[r],Q[r+1])));
248      l=(l+1)%n;
249    }
250    return P.pop_back(),Q.pop_back(),ans;
251  }
252  static char sign(const point<T>&t){
253    return (t.y==0?t.x:t.y)<0;
254  }
255  static bool angle_cmp(const line<T>& A,
         const line<T>& B){
256    point<T> a=A.p2-A.p1,b=B.p2-B.p1;
257    return sign(a)<sign(b)||(sign(a)==sign(b
         )&&a.cross(b)>0);
258  }
259  int halfplane_intersection(vector<line<T>
       > &s){//半平面交
260    sort(s.begin(),s.end(),angle_cmp);//線段
         左側為該線段半平面
261    int L,R,n=s.size();
262    vector<point<T> > px(n);
263    vector<line<T> > q(n);
264    q[L=R=0]=s[0];
265    for(int i=1;i<n;++i){
266      while(L<R&&s[i].ori(px[R-1])<=0)--R;
267      while(L<R&&s[i].ori(px[L])<=0)++L;
268      q[++R]=s[i];
269      if(q[R].parallel(q[R-1])){
270        --R;
271        if(q[R].ori(s[i].p1)>0)q[R]=s[i];
272      }
273      if(L<R)px[R-1]=q[R-1].
           line_intersection(q[R]);
274    }
275    while(L<R&&q[L].ori(px[R-1])<=0)--R;
276    p.clear();
277    if(R-L<=1)return 0;
278    px[R]=q[R].line_intersection(q[L]);
279    for(int i=L;i<=R;++i)p.push_back(px[i]);
280    return R-L+1;
281  }
282 };
283 template<typename T>
284 struct triangle{
285   point<T> a,b,c;
286   triangle(){}
287   triangle(const point<T> &a,const point<T>
         &b,const point<T> &c):a(a),b(b),c(c){}
288   T area(){
289     T t=(b-a).cross(c-a)/2;
290     return t>0?t:-t;
291   }
292   point<T> barycenter()const{//重心
293     return (a+b+c)/3;
294   }
295   point<T> circumcenter()const{//外心
296     static line<T> u,v;
297     u.p1=(a+b)/2;
298     u.p2=point<T>(u.p1.x-a.y+b.y,u.p1.y+a.x-
           b.x);
299     v.p1=(a+c)/2;
300     v.p2=point<T>(v.p1.x-a.y+c.y,v.p1.y+a.x-
           c.x);
301     return u.line_intersection(v);
302   }
303   point<T> incenter()const{//內心
304     T A=sqrt((b-c).abs2()),B=sqrt((a-c).abs2
           ()),C=sqrt((a-b).abs2());
305     return point<T>(A*a.x+B*b.x+C*c.x,A*a.y+
           B*b.y+C*c.y)/(A+B+C);
306   }
307   point<T> perpencenter()const{//垂心
308     return barycenter()*3-circumcenter()*2;
309   }
310 };
311 template<typename T>
312 struct point3D{
313   T x,y,z;
314   point3D(){}
315   point3D(const T&x,const T&y,const T&z):x(x
         ),y(y),z(z){}
316   point3D operator+(const point3D &b)const{
317     return point3D(x+b.x,y+b.y,z+b.z);}
318   point3D operator-(const point3D &b)const{
319     return point3D(x-b.x,y-b.y,z-b.z);}
320   point3D operator*(const T &b)const{
321     return point3D(x*b,y*b,z*b);}
322   point3D operator/(const T &b)const{
323     return point3D(x/b,y/b,z/b);}
324   bool operator==(const point3D &b)const{
325     return x==b.x&&y==b.y&&z==b.z;}
326   T dot(const point3D &b)const{
327     return x*b.x+y*b.y+z*b.z;}
328   point3D cross(const point3D &b)const{
329     return point3D(y*b.z-z*b.y,z*b.x-x*b.z,x
         *b.y-y*b.x);}
330   T abs2()const{//向量長度的平方
331     return dot(*this);}
332   T area2(const point3D &b)const{//和b、原點
         圍成面積的平方
333     return cross(b).abs2()/4;}
334 };
335 template<typename T>
336 struct line3D{
337   point3D<T> p1,p2;
338   line3D(){}
339   line3D(const point3D<T> &p1,const point3D<
         T> &p2):p1(p1),p2(p2){}
340   T dis2(const point3D<T> &p,bool is_segment
         =0)const{//點跟直線/線段的距離平方
341     point3D<T> v=p2-p1,v1=p-p1;
342     if(is_segment){
343       point3D<T> v2=p-p2;
344       if(v.dot(v1)<=0)return v1.abs2();
345       if(v.dot(v2)>=0)return v2.abs2();
346     }
347     point3D<T> tmp=v.cross(v1);
348     return tmp.abs2()/v.abs2();
349   }
350   pair<point3D<T>,point3D<T> > closest_pair(
         const line3D<T> &l)const{
351     point3D<T> v1=(p1-p2),v2=(l.p1-l.p2);
352     point3D<T> N=v1.cross(v2),ab(p1-l.p1);
353     //if(N.abs2()==0)return NULL;平行或重合
354     T tmp=N.dot(ab),ans=tmp*tmp/N.abs2();//
         最近點對距離
355     point3D<T> d1=p2-p1,d2=l.p2-l.p1,D=d1.
         cross(d2),G=l.p1-p1;
356     T t1=(G.cross(d2)).dot(D)/D.abs2();
357     T t2=(G.cross(d1)).dot(D)/D.abs2();
358     return make_pair(p1+d1*t1,l.p1+d2*t2);
359   }
360   bool same_side(const point3D<T> &a,const
         point3D<T> &b)const{
361     return (p2-p1).cross(a-p1).dot((p2-p1).
         cross(b-p1))>0;
362   }
363 };
364 template<typename T>
365 struct plane{
366   point3D<T> p0,n;//平面上的點和法向量
367   plane(){}
368   plane(const point3D<T> &p0,const point3D<T
         > &n):p0(p0),n(n){}
369   T dis2(const point3D<T> &p)const{//點到平
         面距離的平方
370     T tmp=(p-p0).dot(n);
371     return tmp*tmp/n.abs2();
372   }
373   point3D<T> projection(const point3D<T> &p)
         const{
374     return p-n*(p-p0).dot(n)/n.abs2();
375   }
376   point3D<T> line_intersection(const line3D<
         T> &l)const{
377     T tmp=n.dot(l.p2-l.p1);//等於0表示平行或
         重合該直線
378     return l.p1+(l.p2-l.p1)*(n.dot(p0-l.p1)/
         tmp);
379   }
380   line3D<T> plane_intersection(const plane &
         pl)const{
381     point3D<T> e=n.cross(pl.n),v=n.cross(e);
382     T tmp=pl.n.dot(v);//等於0表示平行或重合
         該平面
383     point3D<T> q=p0+(v*(pl.n.dot(pl.p0-p0))/
         tmp);
384     return line3D<T>(q,q+e);
385   }
386 };
387 template<typename T>
388 struct triangle3D{
389   point3D<T> a,b,c;
390   triangle3D(){}
391   triangle3D(const point3D<T> &a,const
         point3D<T> &b,const point3D<T> &c):a(a
         ),b(b),c(c){}
392   bool point_in(const point3D<T> &p)const{//
         點在該平面上的投影在三角形中
393     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);
394   }
395 };
396 template<typename T>
397 struct tetrahedron{//四面體
398   point3D<T> a,b,c,d;
399   tetrahedron(){}
400   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){}
401   T volume6()const{//體積的六倍
402     return (d-a).dot((b-a).cross(c-a));
403   }
404   point3D<T> centroid()const{
405     return (a+b+c+d)/4;
406   }
407   bool point_in(const point3D<T> &p)const{
408     return triangle3D<T>(a,b,c).point_in(p)
           &&triangle3D<T>(c,d,a).point_in(p);
409   }
410 };
411 template<typename T>
412 struct convexhull3D{
413   static const int MAXN=1005;
414   struct face{
415     int a,b,c;
416     face(int a,int b,int c):a(a),b(b),c(c){}
417   };
418   vector<point3D<T>> pt;
419   vector<face> ans;
420   int fid[MAXN][MAXN];
```

```
421  void build(){
422    int n=pt.size();
423    ans.clear();
424    memset(fid,0,sizeof(fid));
425    ans.emplace_back(0,1,2);//注意不能共線
426    ans.emplace_back(2,1,0);
427    int ftop = 0;
428    for(int i=3, ftop=1; i<n; ++i,++ftop){
429      vector<face> next;
430      for(auto &f:ans){
431        T d=(pt[i]-pt[f.a]).dot((pt[f.b]-pt[
             f.a]).cross(pt[f.c]-pt[f.a]));
432        if(d<=0) next.push_back(f);
433        int ff=0;
434        if(d>0) ff=ftop;
435        else if(d<0) ff=-ftop;
436        fid[f.a][f.b]=fid[f.b][f.c]=fid[f.c
             ][f.a]=ff;
437      }
438      for(auto &f:ans){
439        if(fid[f.a][f.b]>0 && fid[f.a][f.b
             ]!=fid[f.b][f.a])
440          next.emplace_back(f.a,f.b,i);
441        if(fid[f.b][f.c]>0 && fid[f.b][f.c
             ]!=fid[f.c][f.b])
442          next.emplace_back(f.b,f.c,i);
443        if(fid[f.c][f.a]>0 && fid[f.c][f.a
             ]!=fid[f.a][f.c])
444          next.emplace_back(f.c,f.a,i);
445      }
446      ans=next;
447    }
448  }
449  point3D<T> centroid()const{
450    point3D<T> res(0,0,0);
451    T vol=0;
452    for(auto &f:ans){
453      T tmp=pt[f.a].dot(pt[f.b].cross(pt[f.c
             ]));
454      res=res+(pt[f.a]+pt[f.b]+pt[f.c])*tmp;
455      vol+=tmp;
456    }
457    return res/(vol*4);
458  }
459 };
```

1.2 SmallestCircle

```
1  using PT=point<T>; using CPT=const PT;
2  PT circumcenter(CPT &a,CPT &b,CPT &c){
3    PT u=b-a, v=c-a;
4    T c1=u.abs2()/2,c2=v.abs2()/2;
5    T d=u.cross(v);
6    return PT(a.x+(v.y*c1-u.y*c2)/d,a.y+(u.x*
          c2-v.x*c1)/d);
7  }
8  void solve(PT p[],int n,PT &c,T &r2){
9    random_shuffle(p,p+n);
10   c=p[0]; r2=0; // c,r2 = 圓心,半徑平方
11   for(int i=1;i<n;i++)if((p[i]-c).abs2()>r2){
12     c=p[i]; r2=0;
13     for(int j=0;j<i;j++)if((p[j]-c).abs2()>r2){
```

```
14       c.x=(p[i].x+p[j].x)/2;
15       c.y=(p[i].y+p[j].y)/2;
16       r2=(p[j]-c).abs2();
17       for(int k=0;k<j;k++)if((p[k]-c).abs2()>r2){
18         c=circumcenter(p[i],p[j],p[k]);
19         r2=(p[i]-c).abs2();
20       }
21     }
22   }
23 }
```

1.3 最近點對

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

2 Data_Structure

2.1 DLX

```
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);
65   return 0;
66 }
```

```
67 void dfs2(int d){//for最小重複覆蓋問題
68   if(d+h()>=ansd)return;
69   if(!R[0]){ansd=d;ans=anst;return;}
70   int c=R[0];
71   DFOR(i,R,0)if(S[i]<S[c])c=i;
72   DFOR(i,D,c){
73     anst.push_back(row[i]);
74     remove2(i);
75     DFOR(j,R,i)remove2(j),--S[col[j]];
76     dfs2(d+1);
77     anst.pop_back();
78     DFOR(j,L,i)restore2(j),++S[col[j]];
79     restore2(i);
80   }
81 }
82 bool exact_cover(){//解精確覆蓋問題
83   return ans.clear(), dfs(0);
84 }
85 void min_cover(){//解最小重複覆蓋問題
86   anst.clear();//暫存用，答案還是存在ans裡
87   dfs2(0);
88 }
89 #undef DFOR
90 };
```

2.2 Dynamic_KD_tree

```
1  template<typename T,size_t kd>//有kd個維度
2  struct kd_tree{
3    struct point{
4      T d[kd];
5      T dist(const point &x)const{
6        T ret=0;
7        for(size_t i=0;i<kd;++i)ret+=std::abs(
              d[i]-x.d[i]);
8        return ret;
9      }
10     bool operator==(const point &p){
11       for(size_t i=0;i<kd;++i)
12         if(d[i]!=p.d[i])return 0;
13       return 1;
14     }
15     bool operator<(const point &b)const{
16       return d[0]<b.d[0];
17     }
18   };
19 private:
20   struct node{
21     node *l,*r;
22     point pid;
23     int s;
24     node(const point &p):l(0),r(0),pid(p),s
              (1){}
25     ~node(){delete l,delete r;}
26     void up(){s=(l?l->s:0)+1+(r?r->s:0);}
27   }*root;
28   const double alpha,loga;
29   const T INF;//記得要給INF，表示極大值
30   int maxn;
31   struct __cmp{
32     int sort_id;
```

```cpp
    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);
    }
    cmp.sort_id=k;
    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_t

```cpp
struct node{//kd樹代替高維線段樹
  node *l,*r;
  point pid,mi,ma;
  int s, data;
  node(const point &p,int d):l(0),r(0),pid(p
      ),mi(p),ma(p),s(1),data(d),dmin(d),
      dmax(d){}
  void up(){
    mi=ma=pid;
    s=1;
    if(l){
      for(int i=0;i<kd;++i){
        mi.d[i]=min(mi.d[i],l->mi.d[i]);
        ma.d[i]=max(ma.d[i],l->ma.d[i]);
      }
      s+=l->s;
    }
    if(r){
      for(int i=0;i<kd;++i){
        mi.d[i]=min(mi.d[i],r->mi.d[i]);
        ma.d[i]=max(ma.d[i],r->ma.d[i]);
      }
      s+=r->s;
    }
  }
  void up2(){/*其他懶惰標記向上更新*/}
  void down(){/*其他懶惰標記下推*/}
}*root;
//檢查區間包含用的函數
bool range_include(node *o,const point &L,
    const point &R){
  for(int i=0;i<kd;++i)
    if(L.d[i]>o->ma.d[i]||R.d[i]<o->mi.d[i])
        return 0;
}//(L,R)區間有和o的區間有交集就回傳true
  return 1;
}
bool range_in_range(node *o,const point &L,
    const point &R){
  for(int i=0;i<kd;++i)
    if(L.d[i]>o->mi.d[i]||o->ma.d[i]>R.d[i])
        return 0;
}//(L,R)區間完全包含o的區間就回傳true
  return 1;
}
bool point_in_range(node *o,const point &L,
    const point &R){
  for(int i=0;i<kd;++i)
    if(L.d[i]>o->pid.d[i]||R.d[i]<o->pid.d[i
        ])return 0;
}//(L,R)區間完全包含o->pid這個點就回傳true
  return 1;
}
//單點修改，以單點改值為例
void update(node *u,const point &x,int data,
    int k=0){
  if(!u)return;
  u->down();
  if(u->pid==x){
    u->data=data;
    u->up2();
    return;
  }
  cmp.sort_id=k;
  update(cmp(x,u->pid)?u->l:u->r,x,data,(k
      +1)%kd);
  u->up2();
}
//區間修改
void update(node *o,const point &L,const
    point &R,int data){
  if(!o)return;
  o->down();
  if(range_in_range(o,L,R)){
    //區間懶惰標記修改
    o->down();
```

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

2.4   reference_point

```
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;}
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

```
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

```
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

```
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   Flow

3.1   dinic

```
1 template<typename T>
2 struct DINIC{
3   static const int MAXN=105;
4   static const T INF=INT_MAX;
5   int n, LV[MAXN], cur[MAXN];
6   struct edge{
7     int v,pre;
8     T cap,r;
9     edge(int v,int pre,T cap):v(v),pre(pre),
         cap(cap),r(cap){}
10  };
11  int g[MAXN];
12  vector<edge> e;
13  void init(int _n){
14    memset(g,-1,sizeof(int)*((n=_n)+1));
15    e.clear();
16  }
17  void add_edge(int u,int v,T cap,bool
         directed=false){
18    e.push_back(edge(v,g[u],cap));
19    g[u]=e.size()-1;
20    e.push_back(edge(u,g[v],directed?0:cap))
         ;
21    g[v]=e.size()-1;
22  }
23  int bfs(int s,int t){
24    memset(LV,0,sizeof(int)*(n+1));
25    memcpy(cur,g,sizeof(int)*(n+1));
26    queue<int> q;
27    q.push(s);
28    LV[s]=1;
29    while(q.size()){
30      int u=q.front();q.pop();
31      for(int i=g[u];~i;i=e[i].pre){
32        if(!LV[e[i].v]&&e[i].r){
33          LV[e[i].v]=LV[u]+1;
34          q.push(e[i].v);
35          if(e[i].v==t)return 1;
36        }
37      }
38    }
39    return 0;
40  }
41  T dfs(int u,int t,T CF=INF){
42    if(u==t)return CF;
43    T df;
44    for(int &i=cur[u];~i;i=e[i].pre){
45      if(LV[e[i].v]==LV[u]+1&&e[i].r){
46        if(df=dfs(e[i].v,t,min(CF,e[i].r))){
47          e[i].r-=df;
48          e[i^1].r+=df;
49          return df;
50        }
51      }
52    }
53    return LV[u]=0;
54  }
55  T dinic(int s,int t,bool clean=true){
56    if(clean)for(size_t i=0;i<e.size();++i)
```

```
57      e[i].r=e[i].cap;
58    T ans=0, f=0;
59    while(bfs(s,t))while(f=dfs(s,t))ans+=f;
60    return ans;
61  }
62 };
```

3.2   ISAP_with_cut

```
1 template<typename T>
2 struct ISAP{
3   static const int MAXN=105;
4   static const T INF=INT_MAX;
5   int n;//點數
6   int d[MAXN],gap[MAXN],cur[MAXN];
7   struct edge{
8     int v,pre;
9     T cap,r;
10    edge(int v,int pre,T cap):v(v),pre(pre),
         cap(cap),r(cap){}
11  };
12  int g[MAXN];
13  vector<edge> e;
14  void init(int _n){
15    memset(g,-1,sizeof(int)*((n=_n)+1));
16    e.clear();
17  }
18  void add_edge(int u,int v,T cap,bool
         directed=false){
19    e.push_back(edge(v,g[u],cap));
20    g[u]=e.size()-1;
21    e.push_back(edge(u,g[v],directed?0:cap))
         ;
22    g[v]=e.size()-1;
23  }
24  T dfs(int u,int s,int t,T CF=INF){
25    if(u==t)return CF;
26    T tf=CF,df;
27    for(int &i=cur[u];~i;i=e[i].pre){
28      if(e[i].r&&d[u]==d[e[i].v]+1){
29        df=dfs(e[i].v,s,t,min(tf,e[i].r));
30        e[i].r-=df;
31        e[i^1].r+=df;
32        if(!(tf-=df)||d[s]==n)return CF-tf;
33      }
34    }
35    int mh=n;
36    for(int i=cur[u]=g[u];~i;i=e[i].pre){
37      if(e[i].r&&d[e[i].v]<mh)mh=d[e[i].v];
38    }
39    if(!--gap[d[u]])d[s]=n;
40    else ++gap[d[u]=++mh];
41    return CF-tf;
42  }
43  T isap(int s,int t,bool clean=true){
44    memset(d,0,sizeof(int)*(n+1));
45    memset(gap,0,sizeof(int)*(n+1));
46    memcpy(cur,g,sizeof(int)*(n+1));
47    if(clean) for(size_t i=0;i<e.size();++i)
48      e[i].r=e[i].cap;
49    T MF=0;
50    for(gap[0]=n;d[s]<n;)MF+=dfs(s,s,t);
51    return MF;
```

```cpp
52      }
53      vector<int> cut_e;//最小割邊集
54      bool vis[MAXN];
55      void dfs_cut(int u){
56          vis[u]=1;//表示u屬於source的最小割集
57          for(int i=g[u];~i;i=e[i].pre)
58              if(e[i].r>0&&!vis[e[i].v])dfs_cut(e[i
                    ].v);
59      }
60      T min_cut(int s,int t){
61          T ans=isap(s,t);
62          memset(vis,0,sizeof(bool)*(n+1));
63          dfs_cut(s), cut_e.clear();
64          for(int u=0;u<=n;++u)if(vis[u])
65              for(int i=g[u];~i;i=e[i].pre)
66                  if(!vis[e[i].v])cut_e.push_back(i);
67          return ans;
68      }
69  };
```

3.3 MinCostMaxFlow

```cpp
1   template<typename TP>
2   struct MCMF{
3       static const int MAXN=440;
4       static const TP INF=999999999;
5       struct edge{
6           int v,pre;
7           TP r,cost;
8           edge(int v,int pre,TP r,TP cost):v(v),
                pre(pre),r(r),cost(cost){}
9       };
10      int n,S,T;
11      TP dis[MAXN],PIS,ans;
12      bool vis[MAXN];
13      vector<edge> e;
14      int g[MAXN];
15      void init(int _n){
16          memset(g,-1,sizeof(int)*((n=_n)+1));
17          e.clear();
18      }
19      void add_edge(int u,int v,TP r,TP cost,
            bool directed=false){
20          e.push_back(edge(v,g[u],r,cost));
21          g[u]=e.size()-1;
22          e.push_back(
23              edge(u,g[v],directed?0:r,-cost));
24          g[v]=e.size()-1;
25      }
26      TP augment(int u,TP CF){
27          if(u==T||!CF)return ans+=PIS*CF,CF;
28          vis[u]=1;
29          TP r=CF,d;
30          for(int i=g[u];~i;i=e[i].pre){
31              if(e[i].r&&!e[i].cost&&!vis[e[i].v]){
32                  d=augment(e[i].v,min(r,e[i].r));
33                  e[i].r-=d;
34                  e[i^1].r+=d;
35                  if(!(r-=d))break;
36              }
37          }
38          return CF-r;
```

```cpp
39      }
40      bool modlabel(){
41          for(int u=0;u<=n;++u)dis[u]=INF;
42          static deque<int>q;
43          dis[T]=0,q.push_back(T);
44          while(q.size()){
45              int u=q.front();q.pop_front();
46              TP dt;
47              for(int i=g[u];~i;i=e[i].pre){
48                  if(e[i^1].r&&(dt=dis[u]-e[i].cost)<
                        dis[e[i].v]){
49                      if((dis[e[i].v]=dt)<=dis[q.size()?
                            q.front():S]){
50                          q.push_front(e[i].v);
51                      }else q.push_back(e[i].v);
52                  }
53              }
54          }
55          for(int u=0;u<=n;++u)
56              for(int i=g[u];~i;i=e[i].pre)
57                  e[i].cost+=dis[e[i].v]-dis[u];
58          return PIS+=dis[S], dis[S]<INF;
59      }
60      TP mincost(int s,int t){
61          S=s,T=t;
62          PIS=ans=0;
63          while(modlabel()){
64              do memset(vis,0,sizeof(bool)*(n+1));
65              while(augment(S,INF));
66          }return ans;
67      }
68  };
```

4 Graph

4.1 Augmenting_Path

```cpp
1   #define MAXN1 505
2   #define MAXN2 505
3   int n1,n2;//n1個點 連向n2個點
4   int match[MAXN2];//屬於n2的點匹配了哪個點
5   vector<int > g[MAXN1];//圖
6   bool vis[MAXN2];//是否走訪過
7   bool dfs(int u){
8       for(size_t i=0;i<g[u].size();++i){
9           int v=g[u][i];
10          if(vis[v])continue;
11          vis[v]=1;
12          if(match[v]==-1||dfs(match[v]))
13              return match[v]=u, 1;
14      }
15      return 0;
16  }
17  inline int max_match(){
18      int ans=0;
19      memset(match,-1,sizeof(int)*n2);
20      for(int i=0;i<n1;++i){
21          memset(vis,0,sizeof(bool)*n2);
22          if(dfs(i))++ans;
23      }
```

```cpp
24      return ans;
25  }
```

4.2 Augmenting_Path_multiple

```cpp
1   #define MAXN1 1005
2   #define MAXN2 505
3   int n1,n2;//n1個點 連向n2個點，其中n2個點可以
        匹配很多邊
4   vector<int> g[MAXN1];//圖
5   int c[MAXN2];//每個屬於n2點 最多可以接受幾條
        匹配邊
6   vector<int> match_list[MAXN2];//每個屬於n2的
        點匹配了那些點
7   bool vis[MAXN2];//是否走訪過
8   bool dfs(int u){
9       for(size_t i=0;i<g[u].size();++i){
10          int v=g[u][i];
11          if(vis[v])continue;
12          vis[v]=true;
13          if((int)match_list[v].size()<c[v]){
14              return match_list[v].push_back(u),
                    true;
15          }else{
16              for(size_t j=0;j<match_list[v].size()
                    ;++j){
17                  int next_u=match_list[v][j];
18                  if(dfs(next_u))
19                      return match_list[v][j]=u, true;
20              }
21          }
22      }
23      return false;
24  }
25  int max_match(){
26      for(int i=0;i<n2;++i)match_list[i].clear()
            ;
27      int cnt=0;
28      for(int u=0;u<n1;++u){
29          memset(vis,0,sizeof(bool)*n2);
30          if(dfs(u))++cnt;
31      }
32      return cnt;
33  }
```

4.3 blossom_matching

```cpp
1   #define MAXN 505
2   vector<int>g[MAXN];
3   int pa[MAXN],match[MAXN],st[MAXN],S[MAXN],v[
        MAXN];
4   int t,n;
5   int lca(int x,int y){
6       for(++t;;swap(x,y)){
7           if(x==0)continue;
8           if(v[x]==t)return x;
9           v[x]=t;
10          x=st[pa[match[x]]];
```

```cpp
11      }
12  }
13  #define qpush(x) q.push(x),S[x]=0
14  void flower(int x,int y,int l,queue<int> &q)
        {
15      while(st[x]!=l){
16          pa[x]=y;
17          if(S[y=match[x]]==1)qpush(y);
18          st[x]=st[y]=l, x=pa[y];
19      }
20  }
21  bool bfs(int x){
22      for(int i=1;i<=n;++i)st[i]=i;
23      memset(S+1,-1,sizeof(int)*n);
24      queue<int>q; qpush(x);
25      while(q.size()){
26          x=q.front(),q.pop();
27          for(size_t i=0;i<g[x].size();++i){
28              int y=g[x][i];
29              if(S[y]==-1){
30                  pa[y]=x,S[y]=1;
31                  if(!match[y]){
32                      for(int lst;x;y=lst,x=pa[y])
33                          lst=match[x],match[x]=y,match[y
                            ]=x;
34                      return 1;
35                  }
36                  qpush(match[y]);
37              }else if(!S[y]&&st[y]!=st[x]){
38                  int l=lca(y,x);
39                  flower(y,x,l,q),flower(x,y,l,q);
40              }
41          }
42      }
43      return 0;
44  }
45  int blossom(){
46      int ans=0;
47      for(int i=1;i<=n;++i)
48          if(!match[i]&&bfs(i))++ans;
49      return ans;
50  }
```

4.4 graphISO

```cpp
1   const int MAXN=1005,K=30;//K要夠大
2   const long long A=3,B=11,C=2,D=19,P=0
        xdefaced;
3   long long f[K+1][MAXN];
4   vector<int> g[MAXN],rg[MAXN];
5   int n;
6   void init(){
7       for(int i=0;i<n;++i){
8           f[0][i]=1;
9           g[i].clear(), rg[i].clear();
10      }
11  }
12  void add_edge(int u,int v){
13      g[u].push_back(v), rg[v].push_back(u);
14  }
15  long long point_hash(int u){//O(N)
16      for(int t=1;t<=K;++t){
17          for(int i=0;i<n;++i){
```

```
18    f[t][i]=f[t-1][i]*A%P;
19    for(int j:g[i])f[t][i]=(f[t][i]+f[t
         -1][j]*B%P)%P;
20    for(int j:rg[i])f[t][i]=(f[t][i]+f[t
         -1][j]*C%P)%P;
21    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
22    f[t][i]%=P;
23    }
24    }
25    return f[K][u];
26 }
27 vector<long long> graph_hash(){
28    vector<long long> ans;
29    for(int i=0;i<n;++i)ans.push_back(
         point_hash(i));//O(N^2)
30    sort(ans.begin(),ans.end());
31    return ans;
32 }
```

4.5 KM

```
1  #define MAXN 405
2  #define INF 0x3f3f3f3f
3  int n;// 1-base，0表示沒有匹配
4  int g[MAXN][MAXN],lx[MAXN],ly[MAXN],pa[MAXN
      ],slack_y[MAXN];
5  int match_y[MAXN],match_x[MAXN];
6  bool vx[MAXN],vy[MAXN];
7  void augment(int y){
8     for(int x,z;y;y=z){
9        x=pa[y],z=match_x[x];
10       match_y[y]=x,match_x[x]=y;
11    }
12 }
13 void bfs(int st){
14    for(int i=1;i<=n;++i)slack_y[i]=INF,vx[i]=
         vy[i]=0;
15    queue<int> q;q.push(st);
16    for(;;){
17       while(q.size()){
18          int x=q.front();q.pop();
19          vx[x]=1;
20          for(int y=1;y<=n;++y)if(!vy[y]){
21             int t=lx[x]+ly[y]-g[x][y];
22             if(t==0){
23                pa[y]=x;
24                if(!match_y[y]){augment(y);return
                     ;}
25                vy[y]=1,q.push(match_y[y]);
26             }else if(slack_y[y]>t)pa[y]=x,
                  slack_y[y]=t;
27          }
28       }
29       int cut=INF;
30       for(int y=1;y<=n;++y){
31          if(!vy[y]&&cut>slack_y[y])cut=slack_y[
             y];
32       }
33       for(int j=1;j<=n;++j){
34          if(vx[j])lx[j]-=cut;
```

```
35          if(vy[j])ly[j]+=cut;
36          else slack_y[j]-=cut;
37       }
38       for(int y=1;y<=n;++y){
39          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 }
```

4.6 MaximumClique

```
1  struct MaxClique{
2     static const int MAXN=105;
3     int N,ans;
4     int g[MAXN][MAXN],dp[MAXN],stk[MAXN][MAXN
         ];
5     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 };
```

4.7 MinimumMeanCycle

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

4.8 Rectilinear_MST

```
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;//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 }
```

4.9 treeISO

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

4.10 一般圖最小權完美匹配

```
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;
```

4.11 全局最小割

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

4.12 平面圖判定

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

4.13 弦圖完美消除序列

```
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  };
```

4.14 最小斯坦納樹 DP

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

4.15   最小樹形圖 __ 朱劉

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

```
    return a;
  }
  void add_edge(int u,int v,T w){
    if(u!=v)pq[v]=merge(pq[v],&(mem[m++]=
        node(u,v,w)));
  }
  int find(int x,int *st){
    return st[x]==x?x:st[x]=find(st[x],st);
  }
  T build(int root,int n){
    T ans=0;int N=n,all=n;
    for(int i=1;i<=N;++i){
      if(i==root||!pq[i])continue;
      while(pq[i]){
        pq[i]->down(),E[i]=pq[i];
        pq[i]=merge(pq[i]->l,pq[i]->r);
        if(find(E[i]->u,id)!=find(i,id))
          break;
      }
      if(find(E[i]->u,id)==find(i,id))
        continue;
      ans+=E[i]->w;
      if(find(E[i]->u,st)==find(i,st)){
        if(pq[i])pq[i]->tag-=E[i]->w;
        pq[++N]=pq[i];id[N]=N;
        for(int u=find(E[i]->u,id);u!=i;u=
            find(E[u]->u,id)){
          if(pq[u])pq[u]->tag-=E[u]->w;
          id[find(u,id)]=N;
          pq[N]=merge(pq[N],pq[u]);
        }
        st[N]=find(i,st);
        id[find(i,id)]=N;
      }else st[find(i,st)]=find(E[i]->u,st)
          ,--all;
    }
    return all==1?ans:-INT_MAX;//圖不連通就
        無解
  }
};
```

4.16   穩定婚姻模板

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

```
        if ( 目前科系已額滿 ) {
            if ( 此考生成績比最低分數還高 ) {
                依加權後分數高低順序將考生id加入科系
                    錄取名單;
                Q.push(被踢出的考生);
            }
        }
    }
}
```

5   Linear_Programming

5.1   simplex

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

```
  for(int i = 1; i <= m; ++i)
    if(left[i] <= n) ans[left[i]] = a[i][0];
  ans[0] = -a[0][0];
  return ans;
}
```

5.2   最大密度子圖

```
typedef double T;//POJ 3155
const int MAXN=105;
struct edge{
  int u,v;
  T w;
  edge(int u=0,int v=0,T w=0):u(u),v(v),w(w)
      {}
};
vector<edge> E;
int n,m;// 1-base
T de[MAXN],pv[MAXN];//每個點的邊權和和點權(
    有些題目會給)
void init(){
  E.clear();
  for(int i=1;i<=n;++i)de[i]=pv[i]=0;
}
void add_edge(int u,int v,T w){
  E.push_back(edge(u,v,w));
  de[u]+=w,de[v]+=w;
}
T U;//二分搜的最大值
void get_U(){
  U=0;
  for(int i=1;i<=n;++i)U+=2*pv[i];
  for(size_t i=0;i<E.size();++i)U+=E[i].w;
}
ISAP<T> isap;//網路流
int s,t;//原匯點
void build(T L){
  isap.init(n+2);
  for(size_t i=0;i<E.size();++i)
    isap.add_edge(E[i].u,E[i].v,E[i].w);
  for(int v=1;v<=n;++v){
    isap.add_edge(s,v,U);
    isap.add_edge(v,t,U+2*L-de[v]-2*pv[v]);
  }
}
int main(){
  while(~scanf("%d%d",&n,&m)){
    if(!m){
      puts("1\n1");
      continue;
    }
    init();
    int u,v;
    for(int i=0;i<m;++i){
      scanf("%d%d",&u,&v);
      add_edge(u,v,1);
    }
    get_U();
    s=n+1,t=n+2;
    T l=0,r=U,k=1.0/(n*n);
    while(r-l>k){//二分搜最大值
      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 }
```

6 Number_Theory

6.1 basic

```
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)
10        phi[x]-=phi[i];
11 }
12 void all_divdown(const LL &n) {// all n/x
13    for(LL a=1;a<=n;a=n/(n/(a+1))){
14        // dosomething;
15    }
16 }
17 const int MAXPRIME = 1000000;
18 int iscom[MAXPRIME], prime[MAXPRIME],
       primecnt;
19 int phi[MAXPRIME], mu[MAXPRIME];
20 void sieve(void){
21    memset(iscom,0,sizeof(iscom));
22    primecnt = 0;
23    phi[1] = mu[1] = 1;
24    for(int i=2;i<MAXPRIME;++i) {
25        if(!iscom[i]) {
26            prime[primecnt++] = i;
27            mu[i] = -1;
28            phi[i] = i-1;
29        }
30        for(int j=0;j<primecnt;++j) {
31            int k = i * prime[j];
32            if(k>=MAXPRIME) break;
33            iscom[k] = prime[j];
34            if(i%prime[j]==0) {
35                mu[k] = 0;
36                phi[k] = phi[i] * prime[j];
37                break;
38            } 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;
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;
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)
172            break;
173        g1=g2;h1=h2;a1=a2;
174        p1=p2;p2=p;
175        q1=q2;q2=q;
176    }
177    System.out.println(p+" "+q);
178 }
```

6.2 bit_set

```
1  void sub_set(int S){
2    int sub=S;
3    do{
4        //對某集合的子集合的處理
5        sub=(sub-1)&S;
6    }while(sub!=S);
7  }
8  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;
13       comb=((comb&~y)/x>>1)|y;
14   }
15 }
```

6.3 cantor_expansion

```
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  }
```

6.4   FFT

```
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  };
```

6.5   find_real_root

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

```
4  }
5
6  double get(const vector<double>&coef, double
        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))) )
15       return lo;
15   if( !(sign_hi = sign(get(coef,hi))) )
16       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 }
```

6.6   FWT

```
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 }
```

6.7   LinearCongruence

```
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 }
```

6.8   Lucas

```
1  int mod_fact(int n,int &e){
2    e=0;
3    if(n==0)return 1;
4    int res=mod_fact(n/P,e);
5    e += n/P;
6    if((n/P)%2==0)return res*fact[n%P]%P;
7    return res*(P-fact[n%P])%P;
```

```
8  }
9  int Cmod(int n,int m){
10   int a1,a2,a3,e1,e2,e3;
11   a1=mod_fact(n,e1);
12   a2=mod_fact(m,e2);
13   a3=mod_fact(n-m,e3);
14   if(e1>e2+e3)return 0;
15   return a1*inv(a2*a3%P,P)%P;
16 }
```

6.9   Matrix

```
1  template<typename T>
2  struct Matrix{
3    using rt = std::vector<T>;
4    using mt = std::vector<rt>;
5    using matrix = Matrix<T>;
6    int r,c;
7    mt m;
8    Matrix(int r,int c):r(r),c(c),m(r,rt(c)){}
9    rt& operator[](int i){return m[i];}
10   matrix operator+(const matrix &a){
11     matrix rev(r,c);
12     for(int i=0;i<r;++i)
13       for(int j=0;j<c;++j)
14         rev[i][j]=m[i][j]+a.m[i][j];
15     return rev;
16   }
17   matrix operator-(const matrix &a){
18     matrix rev(r,c);
19     for(int i=0;i<r;++i)
20       for(int j=0;j<c;++j)
21         rev[i][j]=m[i][j]-a.m[i][j];
22     return rev;
23   }
24   matrix operator*(const matrix &a){
25     matrix rev(r,a.c);
26     matrix tmp(a.c,a.r);
27     for(int i=0;i<a.r;++i)
28       for(int j=0;j<a.c;++j)
29         tmp[j][i]=a.m[i][j];
30     for(int i=0;i<r;++i)
31       for(int j=0;j<a.c;++j)
32         for(int k=0;k<c;++k)
33           rev.m[i][j]+=m[i][k]*tmp[j][k];
34     return rev;
35   }
36   bool inverse(){
37     Matrix t(r,r+c);
38     for(int y=0;y<r;y++){
39       t.m[y][c+y] = 1;
40       for(int x=0;x<c;++x)
41         t.m[y][x]=m[y][x];
42     }
43     if( !t.gas() )
44       return false;
45     for(int y=0;y<r;y++)
46       for(int x=0;x<c;++x)
47         m[y][x]=t.m[y][c+x]/t.m[y][y];
48     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 };
```

6.10   MillerRobin

```
1  LL LLmul(LL a, LL b, const LL &mod) {
2      LL ans=0;
3      while(b) {
4          if(b&1) {
5              ans+=a;
6              if(ans>=mod) ans-=mod;
7          }
8          a<<=1, b>>=1;
9          if(a>=mod) a-=mod;
10     }
11     return ans;
12 }
13 LL mod_mul(LL a,LL b,LL m){
14     a%=m,b%=m;/* fast for m < 2^58 */
15     LL y=(LL)((double)a*b/m+0.5);
16     LL r=(a*b-y*m)%m;
17     return r<0?r+m:r;
18 }
19 template<typename T>
20 T pow(T a,T b,T mod){//a^b%mod
21     T ans=1;
22     for(;b;a=mod_mul(a,a,mod),b>>=1)
23         if(b&1)ans=mod_mul(ans,a,mod);
24     return ans;
25 }
26 int sprp[3]={2,7,61};//int範圍可解
27 int llsprp
        [7]={2,325,9375,28178,450775,9780504,
28 1795265022};//至少unsigned Long Long範圍
29 template<typename T>
30 bool isprime(T n,int *sprp,int num){
31     if(n==2)return 1;
```

```
32    if(n<2||n%2==0)return 0;
33    int t=0;
34    T u=n-1;
35    for(;u%2==0;++t)u>>=1;
36    for(int i=0;i<num;++i){
37        T a=sprp[i]%n;
38        if(a==0||a==1||a==n-1)continue;
39        T x=pow(a,u,n);
40        if(x==1||x==n-1)continue;
41        for(int j=0;j<t;++j){
42            x=mod_mul(x,x,n);
43            if(x==1)return 0;
44            if(x==n-1)break;
45        }
46        if(x==n-1)continue;
47        return 0;
48    }
49    return 1;
50 }
```

6.11   NTT

```
1  2615053605667*(2^18)+1,3
2  15*(2^27)+1,31
3  479*(2^21)+1,3
4  7*17*(2^23)+1,3
5  3*3*211*(2^19)+1,5
6  25*(2^22)+1,3
7  template<typename T,typename VT=vector<T> >
8  struct NTT{
9      const T P,G;
10     NTT(T p=(1<<23)*7*17+1,T g=3):P(p),G(g){}
11     unsigned bit_reverse(unsigned a,int len){
12         //look FFT.cpp
13     }
14     T pow_mod(T n,T k,T m){
15         T ans=1;
16         for(n=(n>=m?n%m:n);k;k>>=1){
17             if(k&1)ans=ans*n%m;
18             n=n*n%m;
19         }
20         return ans;
21     }
22     void ntt(bool is_inv,VT &in,VT &out,int N)
        {
23         int bitlen=__lg(N);
24         for(int i=0;i<N;++i)out[bit_reverse(i,
            bitlen)]=in[i];
25         for(int step=2,id=1;step<=N;step<<=1,++
            id){
26             T wn=pow_mod(G,(P-1)>>id,P),wi=1,u,t;
27             const int mh=step>>1;
28             for(int i=0;i<mh;++i){
29                 for(int j=i;j<N;j+=step){
30                     u=out[j],t=wi*out[j+mh]%P;
31                     out[j]=u+t;
32                     out[j+mh]=u-t;
33                     if(out[j]>=P)out[j]-=P;
34                     if(out[j+mh]<0)out[j+mh]+=P;
35                 }
36                 wi=wi*wn%P;
37             }
38         }
```

```
39         if(is_inv){
40             for(int i=1;i<N/2;++i)swap(out[i],out[
                N-i]);
41             T invn=pow_mod(N,P-2,P);
42             for(int i=0;i<N;++i)out[i]=out[i]*invn
                %P;
43         }
44     }
45 };
```

6.12   Simpson

```
1  double simpson(double a,double b){
2      double c=a+(b-a)/2;
3      return (F(a)+4*F(c)+F(b))*(b-a)/6;
4  }
5  double asr(double a,double b,double eps,
        double A){
6      double c=a+(b-a)/2;
7      double L=simpson(a,c),R=simpson(c,b);
8      if( abs(L+R-A)<15*eps )
9          return L+R+(L+R-A)/15.0;
10     return asr(a,c,eps/2,L)+asr(c,b,eps/2,R);
11 }
12 double asr(double a,double b,double eps){
13     return asr(a,b,eps,simpson(a,b));
14 }
```

6.13   外星模運算

```
1  //a[0]^(a[1]^a[2]^...)
2  #define maxn 1000000
3  int euler[maxn+5];
4  bool is_prime[maxn+5];
5  void init_euler(){
6      is_prime[1]=1;//一不是質數
7      for(int i=1;i<=maxn;i++)euler[i]=i;
8      for(int i=2;i<=maxn;i++){
9          if(!is_prime[i]){//是質數
10             euler[i]--;
11             for(int j=i<<1;j<=maxn;j+=i){
12                 is_prime[j]=1;
13                 euler[j]=euler[j]/i*(i-1);
14             }
15         }
16     }
17 }
18 LL pow(LL a,LL b,LL mod){//a^b%mod
19     LL ans=1;
20     for(;b;a=a*a%mod,b>>=1)
21         if(b&1)ans=ans*a%mod;
22     return ans;
23 }
24 bool isless(LL *a,int n,int k){
25     if(*a==1)return k>1;
26     if(--n==0)return *a<k;
27     int next=0;
28     for(LL b=1;b<k;++next)
29         b*=*a;
```

```
30     return isless(a+1,n,next);
31 }
32 LL high_pow(LL *a,int n,LL mod){
33     if(*a==1||--n==0)return *a%mod;
34     int k=0,r=euler[mod];
35     for(LL tma=1;tma!=pow(*a,k+r,mod);++k)
36         tma=tma*(*a)%mod;
37     if(isless(a+1,n,k))return pow(*a,high_pow(
        a+1,n,k),mod);
38     int tmd=high_pow(a+1,n,r), t=(tmd-k+r)%r;
39     return pow(*a,k+t,mod);
40 }
41 LL a[1000005];
42 int t,mod;
43 int main(){
44     init_euler();
45     scanf("%d",&t);
46     #define n 4
47     while(t--){
48         for(int i=0;i<n;++i)scanf("%lld",&a[i]);
49         scanf("%d",&mod);
50         printf("%lld\n",high_pow(a,n,mod));
51     }
52     return 0;
53 }
```

6.14   數位統計

```
1  ll d[65], dp[65][2];//up區間是不是完整
2  ll dfs(int p,bool is8,bool up){
3      if(!p)return 1; // 回傳0是不是答案
4      if(!up&&~dp[p][is8])return dp[p][is8];
5      int mx = up?d[p]:9;//可以用的有那些
6      ll ans=0;
7      for(int i=0;i<=mx;++i){
8          if( is8&&i==7 )continue;
9          ans += dfs(p-1,i==8,up&&i==mx);
10     }
11     if(!up)dp[p][is8]=ans;
12     return ans;
13 }
14 ll f(ll N){
15     int k=0;
16     while(N){ // 把數字先分解到陣列
17         d[++k] = N%10;
18         N/=10;
19     }
20     return dfs(k,false,true);
21 }
```

6.15   質因數分解

```
1  LL func(const LL n,const LL mod,const int c)
        {
2      return (LLmul(n,n,mod)+c+mod)%mod;
3  }
4
5  LL pollorrho(const LL n, const int c) {//循
        環節長度
```

```
6    LL a=1, b=1;
7    a=func(a,n,c)%n;
8    b=func(b,n,c)%n; b=func(b,n,c)%n;
9    while(gcd(abs(a-b),n)==1) {
10     a=func(a,n,c)%n;
11     b=func(b,n,c)%n; b=func(b,n,c)%n;
12   }
13   return gcd(abs(a-b),n);
14 }
15
16 void prefactor(LL &n, vector<LL> &v) {
17   for(int i=0;i<12;++i) {
18     while(n%prime[i]==0) {
19       v.push_back(prime[i]);
20       n/=prime[i];
21     }
22   }
23 }
24
25 void smallfactor(LL n, vector<LL> &v) {
26   if(n<MAXPRIME) {
27     while(isp[(int)n]) {
28       v.push_back(isp[(int)n]);
29       n/=isp[(int)n];
30     }
31     v.push_back(n);
32   } else {
33     for(int i=0;i<primecnt&&prime[i]*prime[i]<=n;++i) {
34       while(n%prime[i]==0) {
35         v.push_back(prime[i]);
36         n/=prime[i];
37       }
38     }
39     if(n!=1) v.push_back(n);
40   }
41 }
42
43 void comfactor(const LL &n, vector<LL> &v) {
44   if(n<1e9) {
45     smallfactor(n,v);
46     return;
47   }
48   if(Isprime(n)) {
49     v.push_back(n);
50     return;
51   }
52   LL d;
53   for(int c=3;;++c) {
54     d = pollorrho(n,c);
55     if(d!=n) break;
56   }
57   comfactor(d,v);
58   comfactor(n/d,v);
59 }
60
61 void Factor(const LL &x, vector<LL> &v) {
62   LL n = x;
63   if(n==1) { puts("Factor 1"); return; }
64   prefactor(n,v);
65   if(n==1) return;
66   comfactor(n,v);
67   sort(v.begin(),v.end());
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 }
```

7 String

7.1 AC 自動機

```
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+1;++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     }
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;
86     /*把戳記vt+=1，只要vt沒溢位，所有S[p].vis==vt就會變成false 這種利用vt的方法可以O(1)歸零vis陣列*/
87     for(int i=0;s[i];++i){
88       id=s[i]-L;
89       while(!S[p].next[id]&&p)p=S[p].fail;
90       if(!S[p].next[id])continue;
91       p=S[p].next[id];
92       if(S[p].ed&&S[p].vis!=vt){
93         S[p].vis=vt;
94         ans+=S[p].ed;
95       }
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 };
```

7.2 hash

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

7.3 KMP

```
1  /*產生fail function*/
2  void kmp_fail(char *s,int len,int *fail){
3    int id=-1;
4    fail[0]=-1;
5    for(int i=1;i<len;++i){
6      while(~id&&s[id+1]!=s[i])id=fail[id];
7      if(s[id+1]==s[i])++id;
8      fail[i]=id;
9    }
10 }
11 /*以字串B匹配字串A，傳回匹配成功的數量(用B的fail)*/
12 int kmp_match(char *A,int lenA,char *B,int lenB,int *fail){
```

```
13    int id=-1,ans=0;
14    for(int i=0;i<lenA;++i){
15        while(~id&&B[id+1]!=A[i])id=fail[id];
16        if(B[id+1]==A[i])++id;
17        if(id==lenB-1){/*匹配成功*/
18            ++ans, id=fail[id];
19        }
20    }
21    return ans;
22 }
```

7.4 manacher

```
1  //原字串: asdsasdsa
2  //先把字串變成這樣: @#a#s#d#s#a#s#d#s#a#
3  void manacher(char *s,int len,int *z){
4      int l=0,r=0;
5      for(int i=1;i<len;++i){
6          z[i]=r>i?min(z[2*l-i],r-i):1;
7          while(s[i+z[i]]==s[i-z[i]])++z[i];
8          if(z[i]+i>r)r=z[i]+i,l=i;
9      }//ans = max(z)-1
10 }
```

7.5 minimal_string_rotation

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

7.6 reverseBWT

```
1  const int MAXN = 305, MAXC = 'Z';
2  int ranks[MAXN], tots[MAXC], first[MAXC];
3  void rankBWT(const string &bw){
4      memset(ranks,0,sizeof(int)*bw.size());
5      memset(tots,0,sizeof(tots));
6      for(size_t i=0;i<bw.size();++i)
7          ranks[i] = tots[int(bw[i])]++;
8  }
9  void firstCol(){
10     memset(first,0,sizeof(first));
11     int totc = 0;
12     for(int c='A';c<='Z';++c){
13         if(!tots[c]) continue;
```

```
14         first[c] = totc;
15         totc += tots[c];
16     }
17 }
18 string reverseBwt(string bw,int begin){
19     rankBWT(bw), firstCol();
20     int i = begin; //原字串最後一個元素的位置
21     string res;
22     do{
23         char c = bw[i];
24         res = c + res;
25         i = first[int(c)] + ranks[i];
26     }while( i != begin );
27     return res;
28 }
```

7.7 suffix_array_lcp

```
1  #define radix_sort(x,y){\
2      for(i=0;i<A;++i)c[i]=0;\
3      for(i=0;i<n;++i)c[x[y[i]]]++;\
4      for(i=1;i<A;++i)c[i]+=c[i-1];\
5      for(i=n-1;~i;--i)sa[--c[x[y[i]]]]=y[i];\
6  }
7  #define AC(r,a,b)\
8      r[a]!=r[b]||a+k>=n||r[a+k]!=r[b+k]
9  void suffix_array(const char *s,int n,int *
       sa,int *rank,int *tmp,int *c){
10     int A='z'+1,i,k,id=0;
11     for(i=0;i<n;++i)rank[tmp[i]=s[i];
12     radix_sort(rank,tmp);
13     for(k=1;id<n-1;k<<=1){
14         for(id=0,i=n-k;i<n;++i)tmp[id++]=i;
15         for(i=0;i<n;++i)
16             if(sa[i]>=k)tmp[id++]=sa[i]-k;
17         radix_sort(rank,tmp);
18         swap(rank,tmp);
19         for(rank[sa[0]]=id=0,i=1;i<n;++i)
20             rank[sa[i]]=id+=AC(tmp,sa[i-1],sa[i]);
21         A=id+1;
22     }
23 }
24 //h:高度數組 sa:後綴數組 rank:排名
25 void suffix_array_lcp(const char *s,int len,
       int *h,int *sa,int *rank){
26     for(int i=0;i<len;++i)rank[sa[i]]=i;
27     for(int i=0,k=0;i<len;++i){
28         if(rank[i]==0)continue;
29         if(k)--k;
30         while(s[i+k]==s[sa[rank[i]-1]+k])++k;
31         h[rank[i]]=k;
32     }
33     h[0]=0;// h[k]=lcp(sa[k],sa[k-1]);
34 }
```

7.8 Z

```
1  void z_alg(char *s,int len,int *z){
2      int l=0,r=0;
```

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

8 Tarjan

8.1 dominator_tree

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

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

8.2 tnfshb017_2_sat

```
1  #include<bits/stdc++.h>
2  using namespace std;
3  #define MAXN 8001
4  #define MAXN2 MAXN*4
5  #define n(X) ((X)+2*N)
6  vector<int>  v[MAXN2], rv[MAXN2], vis_t;
7  int N,M;
8  void addedge(int s,int e){
9      v[s].push_back(e);
10     rv[e].push_back(s);
11 }
12 int scc[MAXN2];
13 bool vis[MAXN2]={false};
14 void dfs(vector<int> *uv,int n,int k=-1){
15     vis[n]=true;
16     for(int i=0;i<uv[n].size();++i)
17         if(!vis[uv[n][i]])
18             dfs(uv,uv[n][i],k);
19     if(uv==v)vis_t.push_back(n);
20     scc[n]=k;
21 }
22 void solve(){
23     for(int i=1;i<=N;++i){
24         if(!vis[i])dfs(v,i);
25         if(!vis[n(i)])dfs(v,n(i));
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<=M;++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 }
```

8.3 橋連通分量

```
1  #define N 1005
2  struct edge{
3    int u,v;
4    bool is_bridge;
5    edge(int u=0,int v=0):u(u),v(v),is_bridge
         (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 }
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 }
```

8.4 雙連通分量 & 割點

```
1  #define N 1005
2  vector<int> G[N];// 1-base
3  vector<int> bcc[N];//存每塊雙連通分量的點
4  int low[N],vis[N],Time;
5  int bcc_id[N],bcc_cnt;// 1-base
6  bool is_cut[N];//是否為割點
7  int st[N],top;
8  void dfs(int u,int pa=-1){//u當前點，pa父親
9    int 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       }
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 }
```

9 Tree_problem

9.1 HeavyLight

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

9.2 LCA

```
1  const int MAXN=100000; // 1-base
2  const int MLG=17; //Log2(MAXN)+1;
3  int pa[MLG+1][MAXN+5];
4  int dep[MAXN+5];
5  vector<int> G[MAXN+5];
6  void dfs(int x,int p=0){//dfs(root);
7    pa[0][x]=p;
```

```
8    for(int i=0;i<=MLG;++i)
9      pa[i+1][x]=pa[i][pa[i][x]];
10   for(auto &i:G[x]){
11     if(i==p)continue;
12     dep[i]=dep[x]+1;
13     dfs(i,x);
14   }
15 }
16 inline int jump(int x,int d){
17   for(int i=0;i<=MLG;++i)
18     if((d>>i)&1) x=pa[i][x];
19   return x;
20 }
21 inline int find_lca(int a,int b){
22   if(dep[a]>dep[b])swap(a,b);
23   b=jump(b,dep[b]-dep[a]);
24   if(a==b)return a;
25   for(int i=MLG;i>=0;--i){
26     if(pa[i][a]!=pa[i][b]){
27       a=pa[i][a];
28       b=pa[i][b];
29     }
30   }
31   return pa[0][a];
32 }
```

9.3 link_cut_tree

```
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> nd;
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 nd[nd[x].pa].ch[0]!=x&&nd[nd[x].pa
           ].ch[1]!=x;
11 }
12 void down(int x){//懶惰標記下推
13   if(nd[x].rev){
14     if(nd[x].ch[0])nd[nd[x].ch[0]].rev^=1;
15     if(nd[x].ch[1])nd[nd[x].ch[1]].rev^=1;
16     swap(nd[x].ch[0],nd[x].ch[1]);
17     nd[x].rev=0;
18   }
19 }
20 void push_down(int x){//所有祖先懶惰標記下推
21   if(!isroot(x))push_down(nd[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=nd[x].pa,z=nd[y].pa,d=(nd[y].ch[1]==
        x);
27   nd[x].pa=z;
28   if(!isroot(y))nd[z].ch[nd[z].ch[1]==y]=x;
29   nd[y].ch[d]=nd[x].ch[d^1];
```

```
30    nd[nd[y].ch[d]].pa=y;
31    nd[y].pa=x,nd[x].ch[d^1]=y;
32    up(y),up(x);
33  }
34  void splay(int x){//將x伸展到splay tree的根
35    push_down(x);
36    while(!isroot(x)){
37      int y=nd[x].pa;
38      if(!isroot(y)){
39        int z=nd[y].pa;
40        if((nd[z].ch[0]==y)^(nd[y].ch[0]==x))
41          rotate(y);
42        else rotate(x);
43      }
44      rotate(x);
45    }
46  }
47  int access(int x){
48    int last=0;
49    while(x){
50      splay(x);
51      nd[x].ch[1]=last;
52      up(x);
53      last=x;
54      x=nd[x].pa;
55    }
56    return last;//access後splay tree的根
57  }
58  void access(int x,bool is=0){//is=0就是一般
       的access
59    int last=0;
60    while(x){
61      splay(x);
62      if(is&&!nd[x].pa){
63        //printf("%d\n",max(nd[last].ma,nd[nd[
            x].ch[1]].ma));
64      }
65      nd[x].ch[1]=last;
66      up(x);
67      last=x;
68      x=nd[x].pa;
69    }
70  }
71  void query_edge(int u,int v){
72    access(u);
73    access(v,1);
74  }
75  void make_root(int x){
76    access(x),splay(x);
77    nd[x].rev^=1;
78  }
79  void make_root(int x){
80    nd[access(x)].rev^=1;
81    splay(x);
82  }
83  void cut(int x,int y){
84    make_root(x);
85    access(y);
86    splay(y);
87    nd[y].ch[0]=0;
88    nd[x].pa=0;
89  }
90  void cut_parents(int x){
91    access(x);
92    splay(x);
```

```
92    nd[nd[x].ch[0]].pa=0;
93    nd[x].ch[0]=0;
94  }
95  void link(int x,int y){
96    make_root(x);
97    nd[x].pa=y;
98  }
99  int find_root(int x){
100   x=access(x);
101   while(nd[x].ch[0])x=nd[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 nd[lca].data+nd[nd[lca].ch[1]].
            sum
118   }else{
119     //return nd[lca].data+nd[nd[lca].ch[1]].
            sum+nd[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
130 void bfs(int root){
131   //在建構的時候把每個點都設成一個splay tree
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(auto P:G[u]){
139       int v=P.first;
140       if(v!=pa[u]){
141         pa[v]=u;
142         nd[v].pa=u;
143         nd[v].data=e[P.second].w;
144         edge_node[P.second]=v;
145         up(v);
146         q.push(v);
147       }
148     }
149   }
150 }
151 void change(int x,int b){
```

```
152   splay(x);
153   //nd[x].data=b;
154   up(x);
155 }
```

9.4 POJ_tree

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

10 default

10.1 debug

```
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
```

10.2 ext

```
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);
```

10.3 IncStack

```
1  //Magic
2  #pragma GCC optimize "Ofast"
3  //stack resize,change esp to rsp if 64-bit
        system
4  asm("mov %0,%%esp\n" ::"g"(mem+10000000));
5  -Wl,--stack,214748364 -trigraphs
6  //linux stack resize
7  #include<sys/resource.h>
8  void increase_stack(){
9    const rlim_t ks=64*1024*1024;
10   struct rlimit rl;
11   int res=getrlimit(RLIMIT_STACK,&rl);
12   if(!res&&rl.rlim_cur<ks){
13     rl.rlim_cur=ks;
14     res=setrlimit(RLIMIT_STACK,&rl);
15   }
16 }
```

10.4 input

```
1  inline int read(){
2    int x=0; bool f=0; char c=getchar();
3    while(ch<'0'||'9'<ch)f|=ch=='-',ch=getchar
        ();
4    while('0'<=ch&&ch<='9')x=x*10-'0'+ch,ch=
        getchar();
5    return f?-x:x;
6  }
7  // #!/bin/bash
8  // g++ -std=c++11 -O2 -Wall -Wextra -Wno-
        unused-result -DDEBUG $1 && ./a.out
9  //  -fsanitize=address -fsanitize=undefined
        -fsanitize=return
```

11 language

11.1 CNF

```
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 }
```

12 other

12.1 WhatDay

```
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  }
```

12.2 上下最大正方形

```
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);
23 }
```

12.3 最大矩形

```
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 }
```

13 zformula

13.1 formula

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，$E \leq 3V - 6$
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. $\gamma = 0.57721566490153286060651209008240243104215$
5. 格雷碼 $= n \oplus (n >> 1)$
6. $SG(A + B) = SG(A) \oplus SG(B)$
7. 選轉矩陣 $M(\theta) = \begin{pmatrix} cos\theta & -sin\theta \\ sin\theta & cos\theta \end{pmatrix}$

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) $B_{n+1} = \sum_{k=0}^{n} C_k^n B_k$
   (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 8×3^2, 180(邊) 有 6×3^3, 全部 $\frac{1}{24}(3^6 + 6 \times 3^3 + 3 \times 3^4 + 8 \times 3^2 + 6 \times 3^3) = 57$

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

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    public static void main(String args[]){
         throws FileNotFoundException,
         IOException
9      Scanner sc = new Scanner(new FileReader(
         "a.in"));
10     PrintWriter pw = new PrintWriter(new
         FileWriter("a.out"));
11     int n,m;
12     n=sc.nextInt();//读入下一个INT
13     m=sc.nextInt();
14
15     for(ci=1; ci<=c; ++ci){
16       pw.println("Case #"+ci+": easy for
           output");
17     }
18
19     pw.close();//关闭流并释放, 这个很重要,
         否则是没有输出的
20     sc.close();//关闭流并释放
21   }
22 }
```

13.2.2 优先队列

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

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

14

14.1 ganadoQuote

```
1  ¡Allí está!
2  ¡Un forastero!
3  ¡Agarrenlo!
4  ¡Os voy a romper a pedazos!
5  ¡Cógelo!
6  ¡Te voy a hacer picadillo!
7  ¡Te voy a matar!
8  ¡Míralo, está herido!
9  ¡Sos cerdo!
10 ¿Dónde estás?
11 ¡Detrás de tí, imbécil!
12 ¡No dejes que se escape!
13 ¡Basta, hijo de puta!
14 Lord Saddler...
15
16 ¡Mátalo!
17 ¡Allí está!
```

18 Morir es vivir.
19 Síííí, ¡Quiero matar!
20 Muere, muere, muere....
21 Cerebros,cerebros,cerebros...
22 Cógedlo, cógedlo, cógedlo...
23 Lord Saddler...
24 Dieciséis.
25
26 ¡Va por él!
27 ¡Muérete!
28 ¡Cógelo!
29 ¡Te voy a matar!
30 ¡Bloqueale el paso!
31 ¡Te cogí!
32 ¡No dejes que se escape!

34 ¿Qué carajo estás haciendo aquí? ¡Lárgate, cabrón!
35 Hay un rumor de que hay un extranjero entre nosotros.
36 Nuestro jefe se encargará de la rata.
37 Su "Las Plagas" es mucho mejor que la nuestra.
38 Tienes razón, es un hombre.
39 Usa los músculos.
40 Se vuelve loco!
41 ¡Hey, acá!
42 ¡Por aquí!
43 ¡El Gigante!
44 ¡Del Lago!
45 ¡Cógelo!
46 ¡Cógenlo!
47 ¡Allí!
48 ¡Rápido!
49 ¡Empieza a rezar!
50 ¡Mátenlos!
51 ¡Te voy a romper en pedazos!
52 ¡La campana!
53 Ya es hora de rezar.
54 Tenemos que irnos.
55 ¡Maldita sea, mierda!
56 ¡Ya es hora de aplastar!
57 ¡Mierda!
58 ¡Puedes correr, pero no te puedes esconder!
59 ¡Sos cerdo!
60 ¡Está en la trampa!
61 ¡Ah, que madre!
62 ¡Vámonos!
63 ¡Ándale!
64 ¡Cabrón!
65 ¡Coño!
66 ¡Agárrenlo!
67 Cógerlo, Cógerlo...
68 ¡Allí está, mátalo!
69 ¡No dejas que se escape de la isla vivo!
70 ¡Hasta luego!
71 ¡Rápido, es un intruso!

14.2

1 /************************
2 L'Internationale,
3   Sera le genre humain.

17 Вставай, проклятьем заклеймённый,
18 Весь мир голодных и рабов!
19 Кипит наш разум возмущённый
20 И в смертный бой вести готов.
21 Весь мир насилья мы разрушим
22 До основанья, а затем
23 Мы наш, мы новый мир построим, —
24 Кто был ничем, тот станет всем.

26 Chorus
27 Это есть наш последний
28 И решительный бой;
29 С Интернационалом
30 Воспрянет род людской!

32 Никто не даст нам избавленья:
33 Ни бог, ни царь и не герой!
34 Добьёмся мы освобожденья
35 Своею собственной рукой.
36 Чтоб свергнуть гнёт рукой умелой,
37 Отвоевать своё добро, —
38 Вздувайте горн и куйте смело,
39 Пока железо горячо!

41 Chorus

43 Довольно кровь сосать, вампиры,
44 Тюрьмой, налогом, нищетой!
45 У вас — вся власть, все блага мира,
46 А наше право — звук пустой !
47 Мы жизнь построим по-иному —
48 И вот наш лозунг боевой:
49 Вся власть народу трудовому!
50 А дармоедов всех долой!

52 Chorus

54 Презренны вы в своём богатстве,
55 Угля и стали короли!
56 Вы ваши троны, тунеядцы,
57 На наших спинах возвели.
58 Заводы, фабрики, палаты —
59 Всё нашим создано трудом.
60 Пора! Мы требуем возврата
61 Того, что взято грабежом.

63 Chorus

65 Довольно королям в угоду
66 Дурманить нас в чаду войны!
67 Война тиранам! Мир Народу!
68 Бастуйте, армии сыны!
69 Когда ж тираны нас заставят

70 В бою геройски пасть за них —
71 Убийцы, в вас тогда направим
72 Мы жерла пушек боевых!

74 Chorus

76 Лишь мы, работники всемирной
77 Великой армии труда,
78 Владеть землёй имеем право,
79 Но паразиты — никогда!
80 И если гром великий грянет
81 Над сворой псов и палачей, —
82 Для нас всё так же солнце станет
83 Сиять огнём своих лучей.

85 Chorus

14.3 保佑

佛祖保佑　　永無BUG

神獸保佑 永無BUG!

元首保佑 永無BUG

神獸保佑 永無BUG

ACM ICPC Team Reference - Made in Abyss

Contents