1 Computational Geometry

1.1 Geometry

```
const double PI=atan2(0.0,-1.0);
template<typename T>
struct point{
  T x,y;
  point(){}
  point(const T&x,const T&y):x(x),y(y){}
  point operator+(const point &b)const{
    return point(x+b.x,y+b.y); }
  point operator-(const point &b)const{
    return point(x-b.x,y-b.y); }
  point operator*(const T &b)const{
    return point(x*b,y*b); }
  point operator/(const T &b)const{
    return point(x/b,y/b); }
  bool operator==(const point &b)const{
    return x==b.x&&y==b.y; }
  T dot(const point &b)const{
    return x*b.x+y*b.y; }
  T cross(const point &b)const{
    return x*b.y-y*b.x; }
  point normal()const{//求法向量
    return point(-y,x); }
  T abs2()const{//向量長度的平方
    return dot(*this); }
  T rad(const point &b)const{//兩向量的弧度
return fabs(atan2(fabs(cross(b)),dot(b))); }
  T getA()const{//對x軸的弧度
    T A=atan2(y,x);//超過180度會變負的
    if(A<=-PI/2)A+=PI*2;
    return A;
  }
};
template<typename T>
struct line{
  line(){}
  point<T> p1,p2;
  T a,b,c;//ax+by+c=0
  line(const point<T>&x,const point<T>&y):p1
      (x),p2(y){}
  void pton(){//轉成一般式
    a=p1.y-p2.y;
    b=p2.x-p1.x;
    c=-a*p1.x-b*p1.y;
  }
  T ori(const point<T> &p)const{//點和有向直
      線的關係，>0左邊、=0在線上<0右邊
    return (p2-p1).cross(p-p1);
  }
  T btw(const point<T> &p)const{//點投影落在
      線段上<=0
    return (p1-p).dot(p2-p);
  }
  bool point_on_segment(const point<T>&p)
      const{//點是否在線段上
    return ori(p)==0&&btw(p)<=0;
  }
  T dis2(const point<T> &p,bool is_segment
      =0)const{//點跟直線/線段的距離平方
    point<T> v=p2-p1,v1=p-p1;
    if(is_segment){
      point<T> v2=p-p2;
      if(v.dot(v1)<=0)return v1.abs2();
      if(v.dot(v2)>=0)return v2.abs2();
    }
    T tmp=v.cross(v1);
    return tmp*tmp/v.abs2();
  }
  T seg_dis2(const line<T> &l)const{//兩線段
      距離平方
    return min({dis2(l.p1,1),dis2(l.p2,1),l.
        dis2(p1,1),l.dis2(p2,1)});
  }
  point<T> projection(const point<T> &p)
      const{//點對直線的投影
    point<T> n=(p2-p1).normal();
    return p-n*(p-p1).dot(n)/n.abs2();
  }
  point<T> mirror(const point<T> &p)const{
    //點對直線的鏡射，要先呼叫pton轉成一般式
    point<T> R;
    T d=a*a+b*b;
    R.x=(b*b*p.x-a*a*p.x-2*a*b*p.y-2*a*c)/d;
    R.y=(a*a*p.y-b*b*p.y-2*a*b*p.x-2*b*c)/d;
    return R;
  }
  bool equal(const line &l)const{//直線相等
    return ori(l.p1)==0&&ori(l.p2)==0;
  }
  bool parallel(const line &l)const{
    return (p1-p2).cross(l.p1-l.p2)==0;
  }
  bool cross_seg(const line &l)const{
    return (p2-p1).cross(l.p1-p1)*(p2-p1).
        cross(l.p2-p1)<=0;//直線是否交線段
  }
  int line_intersect(const line &l)const{//
      直線相交情況，-1無限多點、1交於一點、0
      不相交
    return parallel(l)?(ori(l.p1)==0?-1:0)
        :1;
  }
  int seg_intersect(const line &l)const{
    T c1=ori(l.p1), c2=ori(l.p2);
    T c3=l.ori(p1), c4=l.ori(p2);
    if(c1==0&&c2==0){//共線
      bool b1=btw(l.p1)>=0,b2=btw(l.p2)>=0;
      T a3=l.btw(p1),a4=l.btw(p2);
      if(b1&&b2&&a3==0&&a4>=0) return 2;
      if(b1&&b2&&a3>=0&&a4==0) return 3;
      if(b1&&b2&&a3>=0&&a4>=0) return 0;
      return -1;//無限交點
    }else if(c1*c2<=0&&c3*c4<=0)return 1;
    return 0;//不相交
  }
  point<T> line_intersection(const line &l)
      const{/*直線交點*/
    point<T> a=p2-p1,b=l.p2-l.p1,s=l.p1-p1;
    //if(a.cross(b)==0)return INF;
    return p1+a*(s.cross(b)/a.cross(b));
  }
  point<T> seg_intersection(const line &l)
      const{//線段交點
    int res=seg_intersect(l);
    if(res<=0) assert(0);
    if(res==2) return p1;
    if(res==3) return p2;
    return line_intersection(l);
  }
};
template<typename T>
struct polygon{
  polygon(){}
  vector<point<T> > p;//逆時針順序
  T area()const{//面積
    T ans=0;
    for(int i=p.size()-1,j=0;j<(int)p.size()
        ;i=j++)
      ans+=p[i].cross(p[j]);
    return ans/2;
  }
  point<T> center_of_mass()const{//重心
    T cx=0,cy=0,w=0;
    for(int i=p.size()-1,j=0;j<(int)p.size()
        ;i=j++){
      T a=p[i].cross(p[j]);
      cx+=(p[i].x+p[j].x)*a;
      cy+=(p[i].y+p[j].y)*a;
      w+=a;
    }
    return point<T>(cx/3/w,cy/3/w);
  }
  char ahas(const point<T>& t)const{//點是否
      在簡單多邊形內，是的話回傳1、在邊上回
      傳-1、否則回傳0
    bool c=0;
    for(int i=0,j=p.size()-1;i<p.size();j=i
        ++)
      if(line<T>(p[i],p[j]).point_on_segment
          (t))return -1;
      else if((p[i].y>t.y)!=(p[j].y>t.y)&&
          t.x<(p[j].x-p[i].x)*(t.y-p[i].y)/(p[j
          ].y-p[i].y)+p[i].x)
        c=!c;
    return c;
  }
  char point_in_convex(const point<T>&x)
      const{
    int l=1,r=(int)p.size()-2;
    while(l<=r){//點是否在凸多邊形內，是的話
        回傳1、在邊上回傳-1、否則回傳0
      int mid=(l+r)/2;
      T a1=(p[mid]-p[0]).cross(x-p[0]);
      T a2=(p[mid+1]-p[0]).cross(x-p[0]);
      if(a1>=0&&a2<=0){
        T res=(p[mid+1]-p[mid]).cross(x-p[
            mid]);
        return res>0?1:(res>=0?-1:0);
      }else if(a1<0)r=mid-1;
      else l=mid+1;
    }
    return 0;
  }
  vector<T> getA()const{//凸包邊對x軸的夾角
    vector<T>res;//一定是遞增的
    for(size_t i=0;i<p.size();++i)
      res.push_back((p[(i+1)%p.size()]-p[i])
          .getA());
    return res;
  }
  bool line_intersect(const vector<T>&A,
      const line<T> &l)const{//O(logN)
    int f1=upper_bound(A.begin(),A.end(),(l.
        p1-l.p2).getA())-A.begin();
    int f2=upper_bound(A.begin(),A.end(),(l.
        p2-l.p1).getA())-A.begin();
    return l.cross_seg(line<T>(p[f1],p[f2]))
        ;
  }
  polygon cut(const line<T> &l)const{//凸包
      對直線切割，得到直線l左側的凸包
    polygon ans;
    for(int n=p.size(),i=n-1,j=0;j<n;i=j++){
      if(l.ori(p[i])>=0){
        ans.p.push_back(p[i]);
        if(l.ori(p[j])<0)
          ans.p.push_back(l.
              line_intersection(line<T>(p[i
              ],p[j])));
      }else if(l.ori(p[j])>0)
        ans.p.push_back(l.line_intersection(
            line<T>(p[i],p[j])));
    }
    return ans;
  }
  static bool monotone_chain_cmp(const point
      <T>& a,const point<T>& b){//凸包排序函
      數
    return (a.x<b.x)||(a.x==b.x&&a.y<b.y);
  }
  void monotone_chain(vector<point<T> > &s){
      //凸包
    sort(s.begin(),s.end(),
        monotone_chain_cmp);
    p.resize(s.size()+1);
    int m=0;
    for(size_t i=0;i<s.size();++i){
      while(m>=2&&(p[m-1]-p[m-2]).cross(s[i
          ]-p[m-2])<=0)--m;
      p[m++]=s[i];
    }
    for(int i=s.size()-2,t=m+1;i>=0;--i){
      while(m>=t&&(p[m-1]-p[m-2]).cross(s[i
          ]-p[m-2])<=0)--m;
      p[m++]=s[i];
    }
    if(s.size()>1)--m;
    p.resize(m);
  }
  T diam(){//直徑
    int n=p.size(),t=1;
    T ans=0;p.push_back(p[0]);
    for(int i=0;i<n;i++){
      point<T> now=p[i+1]-p[i];
      while(now.cross(p[t+1]-p[i])>now.cross
          (p[t]-p[i]))t=(t+1)%n;
      ans=max(ans,(p[i]-p[t]).abs2());
    }
    return p.pop_back(),ans;
  }
  T min_cover_rectangle(){//最小覆蓋矩形
```

```
    int n=p.size(),t=1,r=1,l;
    if(n<3)return 0;//也可以做最小周長矩形
    T ans=1e99;p.push_back(p[0]);
    for(int i=0;i<n;i++){
      point<T> now=p[i+1]-p[i];
      while(now.cross(p[t+1]-p[i])>now.cross
           (p[t]-p[i]))t=(t+1)%n;
      while(now.dot(p[r+1]-p[i])>now.dot(p[r
           ]-p[i]))r=(r+1)%n;
      if(!i)l=r;
      while(now.dot(p[l+1]-p[i])<=now.dot(p[
           l]-p[i]))l=(l+1)%n;
      T d=now.abs2();
      T tmp=now.cross(p[t]-p[i])*(now.dot(p[
           r]-p[i])-now.dot(p[l]-p[i]))/d;
      ans=min(ans,tmp);
    }
    return p.pop_back(),ans;
  }
  T dis2(polygon &pl){//凸包最近距離平方
    vector<point<T> > &P=p,&Q=pl.p;
    int n=P.size(),m=Q.size(),l=0,r=0;
  for(int i=0;i<n;++i)if(P[i].y<P[l].y)l=i;
  for(int i=0;i<m;++i)if(Q[i].y<Q[r].y)r=i;
    P.push_back(P[0]),Q.push_back(Q[0]);
    T ans=1e99;
    for(int i=0;i<n;++i){
      while((P[l]-P[l+1]).cross(Q[r+1]-Q[r])
           <0)r=(r+1)%m;
      ans=min(ans,line<T>(P[l],P[l+1]).
           seg_dis2(line<T>(Q[r],Q[r+1])));
      l=(l+1)%n;
    }
    return P.pop_back(),Q.pop_back(),ans;
  }
  static char sign(const point<T>&t){
    return (t.y==0?t.x:t.y)<0;
  }
  static bool angle_cmp(const line<T>& A,
       const line<T>& B){
    point<T> a=A.p2-A.p1,b=B.p2-B.p1;
    return sign(a)<sign(b)||(sign(a)==sign(b
         )&&a.cross(b)>0);
  }
  int halfplane_intersection(vector<line<T>
       > &s){//半平面交
    sort(s.begin(),s.end(),angle_cmp);//線段
         左側為該線段半平面
    int L,R,n=s.size();
    vector<point<T> > px(n);
    vector<line<T> > q(n);
    q[L=R=0]=s[0];
    for(int i=1;i<n;++i){
      while(L<R&&s[i].ori(px[R-1])<=0)--R;
      while(L<R&&s[i].ori(px[L])<=0)++L;
      q[++R]=s[i];
      if(q[R].parallel(q[R-1])){
        --R;
        if(q[R].ori(s[i].p1)>0)q[R]=s[i];
      }
      if(L<R)px[R-1]=q[R-1].
           line_intersection(q[R]);
    }
    while(L<R&&q[L].ori(px[R-1])<=0)--R;
    p.clear();
    if(R-L<=1)return 0;
    px[R]=q[R].line_intersection(q[L]);
    for(int i=L;i<=R;++i)p.push_back(px[i]);
    return R-L+1;
  }
};
template<typename T>
struct triangle{
  point<T> a,b,c;
  triangle(){}
  triangle(const point<T> &a,const point<T>
       &b,const point<T> &c):a(a),b(b),c(c){}
  T area()const{
    T t=(b-a).cross(c-a)/2;
    return t>0?t:-t;
  }
  point<T> barycenter()const{//重心
    return (a+b+c)/3;
  }
  point<T> circumcenter()const{//外心
    static line<T> u,v;
    u.p1=(a+b)/2;
    u.p2=point<T>(u.p1.x-a.y+b.y,u.p1.y+a.x-
         b.x);
    v.p1=(a+c)/2;
    v.p2=point<T>(v.p1.x-a.y+c.y,v.p1.y+a.x-
         c.x);
    return u.line_intersection(v);
  }
  point<T> incenter()const{//內心
    T A=sqrt((b-c).abs2()),B=sqrt((a-c).abs2
         ()),C=sqrt((a-b).abs2());
    return point<T>(A*a.x+B*b.x+C*c.x,A*a.y+
         B*b.y+C*c.y)/(A+B+C);
  }
  point<T> perpencenter()const{//垂心
    return barycenter()*3-circumcenter()*2;
  }
};
template<typename T>
struct point3D{
  T x,y,z;
  point3D(){}
  point3D(const T&x,const T&y,const T&z):x(x
       ),y(y),z(z){}
  point3D operator+(const point3D &b)const{
    return point3D(x+b.x,y+b.y,z+b.z);}
  point3D operator-(const point3D &b)const{
    return point3D(x-b.x,y-b.y,z-b.z);}
  point3D operator*(const T &b)const{
    return point3D(x*b,y*b,z*b);}
  point3D operator/(const T &b)const{
    return point3D(x/b,y/b,z/b);}
  bool operator==(const point3D &b)const{
    return x==b.x&&y==b.y&&z==b.z;}
  T dot(const point3D &b)const{
    return x*b.x+y*b.y+z*b.z;}
  point3D cross(const point3D &b)const{
    return point3D(y*b.z-z*b.y,z*b.x-x*b.z,x
         *b.y-y*b.x);}
  T abs2()const{//向量長度的平方
    return dot(*this);}
  T area2(const point3D &b)const{//和b、原點
       圍成面積的平方
    return cross(b).abs2()/4;}
};
template<typename T>
struct line3D{
  point3D<T> p1,p2;
  line3D(){}
  line3D(const point3D<T> &p1,const point3D<
       T> &p2):p1(p1),p2(p2){}
  T dis2(const point3D<T> &p,bool is_segment
       =0)const{//點跟直線/線段的距離平方
    point3D<T> v=p2-p1,v1=p-p1;
    if(is_segment){
      point3D<T> v2=p-p2;
      if(v.dot(v1)<=0)return v1.abs2();
      if(v.dot(v2)>=0)return v2.abs2();
    }
    point3D<T> tmp=v.cross(v1);
    return tmp.abs2()/v.abs2();
  }
  pair<point3D<T>,point3D<T> > closest_pair(
       const line3D<T> &l)const{
    point3D<T> v1=(p1-p2),v2=(l.p1-l.p2);
    point3D<T> N=v1.cross(v2),ab(p1-l.p1);
    //if(N.abs2()==0)return NULL;平行或重合
    T tmp=N.dot(ab),ans=tmp*tmp/N.abs2();//
         最近點對距離
    point3D<T> d1=p2-p1,d2=l.p2-l.p1,D=d1.
         cross(d2),G=l.p1-p1;
    T t1=(G.cross(d2)).dot(D)/D.abs2();
    T t2=(G.cross(d1)).dot(D)/D.abs2();
    return make_pair(p1+d1*t1,l.p1+d2*t2);
  }
  bool same_side(const point3D<T> &a,const
       point3D<T> &b)const{
    return (p2-p1).cross(a-p1).dot((p2-p1).
         cross(b-p1))>0;
  }
};
template<typename T>
struct plane{
  point3D<T> p0,n;//平面上的點和法向量
  plane(){}
  plane(const point3D<T> &p0,const point3D<T
       > &n):p0(p0),n(n){}
  T dis2(const point3D<T> &p)const{//點到平
       面距離的平方
    T tmp=(p-p0).dot(n);
    return tmp*tmp/n.abs2();
  }
  point3D<T> projection(const point3D<T> &p)
       const{
    return p-n*(p-p0).dot(n)/n.abs2();
  }
  point3D<T> line_intersection(const line3D<
       T> &l)const{
    T tmp=n.dot(l.p2-l.p1);//等於0表示平行或
         重合該平面
    return l.p1+(l.p2-l.p1)*(n.dot(p0-l.p1)/
         tmp);
  }
  line3D<T> plane_intersection(const plane &
       pl)const{
    point3D<T> e=n.cross(pl.n),v=n.cross(e);
    T tmp=pl.n.dot(v);//等於0表示平行或重合
         該平面
    point3D<T> q=p0+(v*(pl.n.dot(pl.p0-p0))/
         tmp);
    return line3D<T>(q,q+e);
  }
};
template<typename T>
struct triangle3D{
  point3D<T> a,b,c;
  triangle3D(){}
  triangle3D(const point3D<T> &a,const
       point3D<T> &b,const point3D<T> &c):a(a
       ),b(b),c(c){}
  bool point_in(const point3D<T> &p)const{//
       點在該平面上的投影在三角形中
    return line3D<T>(b,c).same_side(p,a)&&
         line3D<T>(a,c).same_side(p,b)&&
         line3D<T>(a,b).same_side(p,c);
  }
};
template<typename T>
struct tetrahedron{//四面體
  point3D<T> a,b,c,d;
  tetrahedron(){}
  tetrahedron(const point3D<T> &a,const
       point3D<T> &b,const point3D<T> &c,
       const point3D<T> &d):a(a),b(b),c(c),d(
       d){}
  T volume6()const{//體積的六倍
    return (d-a).dot((b-a).cross(c-a));
  }
  point3D<T> centroid()const{
    return (a+b+c+d)/4;
  }
  bool point_in(const point3D<T> &p)const{
    return triangle3D<T>(a,b,c).point_in(p)
         &&triangle3D<T>(c,d,a).point_in(p);
  }
};
template<typename T>
struct convexhull3D{
  static const int MAXN=1005;
  struct face{
    int a,b,c;
    face(int a,int b,int c):a(a),b(b),c(c){}
  };
  vector<point3D<T>> pt;
  vector<face> ans;
  int fid[MAXN][MAXN];
  void build(){
    int n=pt.size();
    ans.clear();
    memset(fid,0,sizeof(fid));
    ans.emplace_back(0,1,2);//注意不能共線
    ans.emplace_back(2,1,0);
    int ftop = 0;
    for(int i=3, ftop=1; i<n; ++i,++ftop){
      vector<face> next;
      for(auto &f:ans){
        T d=(pt[i]-pt[f.a]).dot((pt[f.b]-pt[
             f.a]).cross(pt[f.c]-pt[f.a]));
        if(d<=0) next.push_back(f);
        int ff=0;
        if(d>0) ff=ftop;
        else if(d<0) ff=-ftop;
```

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

1.2 SmallestCircle

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

1.3 最近點對

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

2 Data Structure

2.1 undo disjoint set

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

2.2 DLX

```
const int MAXN=4100, MAXM=1030, MAXND=16390;
struct DLX{
  int n,m,sz,ansd;//高是n，寬是m的稀疏矩陣
  int S[MAXM],H[MAXN];
  int row[MAXND],col[MAXND];//每個節點代表的列跟行
  int L[MAXND],R[MAXND],U[MAXND],D[MAXND];
  vector<int> ans,anst;
  void init(int _n,int _m){
    n=_n,m=_m;
    for(int i=0;i<=m;++i){
      U[i]=D[i]=i,L[i]=i-1,R[i]=i+1;
      S[i]=0;
    }
    R[m]=0,L[0]=m;
    sz=m,ansd=INT_MAX;//ansd存最優解的個數
    for(int i=1;i<=n;++i)H[i]=-1;
  }
  void add(int r,int c){
    ++S[col[++sz]=c];
    row[sz]=r;
    D[sz]=D[c],U[D[c]]=sz,U[sz]=c,D[c]=sz;
    if(H[r]<0)H[r]=L[sz]=R[sz]=sz;
    else R[sz]=R[H[r]],L[R[H[r]]]=sz,L[sz]=H[r],R[H[r]]=sz;
  }
  #define DFOR(i,A,s) for(int i=A[s];i!=s;i=A[i])
  void remove(int c){//刪除第c行和所有當前覆蓋到第c行的列
    L[R[c]]=L[c],R[L[c]]=R[c];//這裡刪除第c行，若有些行不需要處理可以在開始時呼叫他
    DFOR(i,D,c)DFOR(j,R,i){U[D[j]]=U[j],D[U[j]]=D[j],--S[col[j]];}
  }
  void restore(int c){//恢復第c行和所有當前覆蓋到第c行的列，remove的逆操作
    DFOR(i,U,c)DFOR(j,L,i){++S[col[j]],U[D[j]]=j,D[U[j]]=j;}
    L[R[c]]=c,R[L[c]]=c;
  }
  void remove2(int nd){//刪除nd所在的行當前所有點(包括虛擬節點)，只保留nd
    DFOR(i,D,nd)L[R[i]]=L[i],R[L[i]]=R[i];
  }
  void restore2(int nd){//刪除nd所在的行當前所有點，為remove2的逆操作
    DFOR(i,U,nd)L[R[i]]=R[L[i]]=i;
  }
  bool vis[MAXM];
  int h(){//估價函數 for IDA*
    int res=0;
    memset(vis,0,sizeof(vis));
    DFOR(i,R,0)if(!vis[i]){
      vis[i]=1;
      ++res;
      DFOR(j,D,i)DFOR(k,R,j)vis[col[k]]=1;
    }
    return res;
  }
  bool dfs(int d){//for精確覆蓋問題
    if(d+h()>=ansd)return 0;//找最佳解用，找任意解可以刪掉
    if(!R[0]){ansd=d;return 1;}
    int c=R[0];
    DFOR(i,R,0)if(S[i]<S[c])c=i;
    remove(c);
    DFOR(i,D,c){
      ans.push_back(row[i]);
      DFOR(j,R,i)remove(col[j]);
      if(dfs(d+1))return 1;
      ans.pop_back();
      DFOR(j,L,i)restore(col[j]);
    }
    restore(c);
    return 0;
  }
  void dfs2(int d){//for最小重複覆蓋問題
    if(d+h()>=ansd)return;
    if(!R[0]){ansd=d;ans=anst;return;}
    int c=R[0];
    DFOR(i,R,0)if(S[i]<S[c])c=i;
    DFOR(i,D,c){
      anst.push_back(row[i]);
      remove2(i);
      DFOR(j,R,i)remove2(j),--S[col[j]];
      dfs2(d+1);
      anst.pop_back();
      DFOR(j,L,i)restore2(j),++S[col[j]];
      restore2(i);
    }
  }
  bool exact_cover(){//解精確覆蓋問題
    return ans.clear(), dfs(0);
  }
  void min_cover(){//解最小重複覆蓋問題
    anst.clear();//暫存用，答案還是存在ans裡
    dfs2(0);
  }
  #undef DFOR
};
```

3 Flow

3.1 Gomory Hu

```
//最小割樹+求任兩點間最小割
//0-base, root=0
LL e[MAXN][MAXN]; //任兩點間最小割
int p[MAXN]; //parent
ISAP D; // original graph
void gomory_hu(){
```

```
  fill(p, p+n, 0);
  fill(e[0], e[n], INF);
  for( int s = 1; s < n; ++s ) {
    int t = p[s];
    ISAP F = D;
    LL tmp = F.min_cut(s, t);
    for( int i = 1; i < s; ++i )
      e[s][i] = e[i][s] = min(tmp, e[t][i]);
    for( int i = s+1; i <= n; ++i )
      if( p[i] == t && F.vis[i] ) p[i] = s;
  }
}
```

3.2 ISAP with cut

```
template<typename T>
struct ISAP{
  static const int MAXN=105;
  static const T INF=INT_MAX;
  int n;//點數
  int d[MAXN],gap[MAXN],cur[MAXN];
  struct edge{
    int v,pre;
    T cap,r;
    edge(int v,int pre,T cap):v(v),pre(pre),
        cap(cap),r(cap){}
  };
  int g[MAXN];
  vector<edge> e;
  void init(int _n){
    memset(g,-1,sizeof(int)*((n=_n)+1));
    e.clear();
  }
  void add_edge(int u,int v,T cap,bool
      directed=false){
    e.push_back(edge(v,g[u],cap));
    g[u]=e.size()-1;
    e.push_back(edge(u,g[v],directed?0:cap))
        ;
    g[v]=e.size()-1;
  }
  T dfs(int u,int s,int t,T CF=INF){
    if(u==t)return CF;
    T tf=CF,df;
    for(int &i=cur[u];~i;i=e[i].pre){
      if(e[i].r&&d[u]==d[e[i].v]+1){
        df=dfs(e[i].v,s,t,min(tf,e[i].r));
        e[i].r-=df;
        e[i^1].r+=df;
        if(!(tf-=df)||d[s]==n)return CF-tf;
      }
    }
    int mh=n;
    for(int i=cur[u]=g[u];~i;i=e[i].pre){
      if(e[i].r&&d[e[i].v]<mh)mh=d[e[i].v];
    }
    if(!--gap[d[u]])d[s]=n;
    else ++gap[d[u]=++mh];
    return CF-tf;
  }
  T isap(int s,int t,bool clean=true){
    memset(d,0,sizeof(int)*(n+1));
    memset(gap,0,sizeof(int)*(n+1));
    memcpy(cur,g,sizeof(int)*(n+1));
    if(clean) for(size_t i=0;i<e.size();++i)
      e[i].r=e[i].cap;
    T MF=0;
    for(gap[0]=n;d[s]<n;)MF+=dfs(s,s,t);
    return MF;
  }
  vector<int> cut_e;//最小割邊集
  bool vis[MAXN];
  void dfs_cut(int u){
    vis[u]=1;//表示u屬於source的最小割集
    for(int i=g[u];~i;i=e[i].pre)
      if(e[i].r>0&&!vis[e[i].v])dfs_cut(e[i
          ].v);
  }
  T min_cut(int s,int t){
    T ans=isap(s,t);
    memset(vis,0,sizeof(bool)*(n+1));
    dfs_cut(s), cut_e.clear();
    for(int u=0;u<=n;++u)if(vis[u])
      for(int i=g[u];~i;i=e[i].pre)
        if(!vis[e[i].v])cut_e.push_back(i);
    return ans;
  }
};
```

3.3 maxFlow 拷貝

```
#include<bits/stdc++.h>
using namespace std;
typedef int Graph[55][55];
Graph C, R; //Capacity, Flow, Remain
int parent[55], flow[55];
int n, m;
int bfs(int s, int t){
  memset(parent, 0, sizeof(parent));
  memset(flow, 0, sizeof(flow));
  queue<int> q;
  q.push(s);
  parent[s] = s;
  flow[s] = INT_MAX;
  int current;
  while(!q.empty()){
    current = q.front();
    q.pop();
    for(int i=1;i<=n;i++){
      if(R[current][i] > 0 && parent[i] ==
          0){
        parent[i] = current;
        flow[i] = min(R[current][i], flow[
            current]);
        if(i == t){
          return flow[t];
        }
        q.push(i);
      }
    }
  }
  return 0;
}
void fordFulkerson(int s, int t){
  memcpy(R, C, sizeof(C));
  while(1){
    int f = bfs(1,2);
    if(f == 0)break;
    int s = parent[2], t = 2;
    while(s != t){
      R[s][t] -= f;
      t = s;
      s = parent[t];
    }
  }

  for(int i=1;i<=n;i++){
    if(flow[i] > 0){
      for(int j=1;j<=n;j++){
        if(flow[j] == 0 && C[i][j] != 0){
          cout<<i<<" "<<j<<endl;
        }
      }
    }
  }
  cout<<endl;
}
int main(){
  while(cin>>n>>m){
    if(n == 0 && m == 0)break;
    memset(R,0, sizeof(R));
    memset(C,0,sizeof(C));
    for(int i=0;i<m;i++){
      int a, b, c;
      cin>>a>>b>>c;
      C[a][b] = c;
      C[b][a] = c;
    }
    fordFulkerson(1,2);
  }
  return 0;
}
```

3.4 MinCostMaxFlow

```
template<typename TP>
struct MCMF{
  static const int MAXN=440;
  static const TP INF=999999999;
  struct edge{
    int v,pre;
    TP r,cost;
    edge(int v,int pre,TP r,TP cost):v(v),
        pre(pre),r(r),cost(cost){}
  };
  int n,S,T;
  TP dis[MAXN],PIS,ans;
  bool vis[MAXN];
  vector<edge> e;
  int g[MAXN];
  void init(int _n){
    memset(g,-1,sizeof(int)*((n=_n)+1));
    e.clear();
  }
  void add_edge(int u,int v,TP r,TP cost,
      bool directed=false){
    e.push_back(edge(v,g[u],r,cost));
    g[u]=e.size()-1;
    e.push_back(
    edge(u,g[v],directed?0:r,-cost));
    g[v]=e.size()-1;
  }
  TP augment(int u,TP CF){
    if(u==T||!CF)return ans+=PIS*CF,CF;
    vis[u]=1;
    TP r=CF,d;
    for(int i=g[u];~i;i=e[i].pre){
      if(e[i].r&&!e[i].cost&&!vis[e[i].v]){
        d=augment(e[i].v,min(r,e[i].r));
        e[i].r-=d;
        e[i^1].r+=d;
        if(!(r-=d))break;
      }
    }
    return CF-r;
  }
  bool modlabel(){
    for(int u=0;u<=n;++u)dis[u]=INF;
    static deque<int>q;
    dis[T]=0,q.push_back(T);
    while(q.size()){
      int u=q.front();q.pop_front();
      TP dt;
      for(int i=g[u];~i;i=e[i].pre){
        if(e[i^1].r&&(dt=dis[u]-e[i].cost)<
            dis[e[i].v]){
          if((dis[e[i].v]=dt)<=dis[q.size()?
              q.front():S]){
            q.push_front(e[i].v);
          }else q.push_back(e[i].v);
        }
      }
    }
    for(int u=0;u<=n;++u)
      for(int i=g[u];~i;i=e[i].pre)
        e[i].cost+=dis[e[i].v]-dis[u];
    return PIS+=dis[S], dis[S]<INF;
  }
  TP mincost(int s,int t){
    S=s,T=t;
    PIS=ans=0;
    while(modlabel()){
      do memset(vis,0,sizeof(bool)*(n+1));
      while(augment(S,INF));
    }return ans;
  }
};
```

3.5 dinic

```
template<typename T>
struct DINIC{
  static const int MAXN=105;
  static const T INF=INT_MAX;
  int n, LV[MAXN], cur[MAXN];
  struct edge{
    int v,pre;
    T cap,r;
    edge(int v,int pre,T cap):v(v),pre(pre),
        cap(cap),r(cap){}
  };
```

```
  int g[MAXN];
  vector<edge> e;
  void init(int _n){
    memset(g,-1,sizeof(int)*((n=_n)+1));
    e.clear();
  }
  void add_edge(int u,int v,T cap,bool directed=false){
    e.push_back(edge(v,g[u],cap));
    g[u]=e.size()-1;
    e.push_back(edge(u,g[v],directed?0:cap));
    g[v]=e.size()-1;
  }
  int bfs(int s,int t){
    memset(LV,0,sizeof(int)*(n+1));
    memcpy(cur,g,sizeof(int)*(n+1));
    queue<int> q;
    q.push(s);
    LV[s]=1;
    while(q.size()){
      int u=q.front();q.pop();
      for(int i=g[u];~i;i=e[i].pre){
        if(!LV[e[i].v]&&e[i].r){
          LV[e[i].v]=LV[u]+1;
          q.push(e[i].v);
          if(e[i].v==t)return 1;
        }
      }
    }
    return 0;
  }
  T dfs(int u,int t,T CF=INF){
    if(u==t)return CF;
    T df;
    for(int &i=cur[u];~i;i=e[i].pre){
      if(LV[e[i].v]==LV[u]+1&&e[i].r){
        if(df=dfs(e[i].v,t,min(CF,e[i].r))){
          e[i].r-=df;
          e[i^1].r+=df;
          return df;
        }
      }
    }
    return LV[u]=0;
  }
  T dinic(int s,int t,bool clean=true){
    if(clean)for(size_t i=0;i<e.size();++i)
      e[i].r=e[i].cap;
    T ans=0, f=0;
    while(bfs(s,t))while(f=dfs(s,t))ans+=f;
    return ans;
  }
};
```

4 Graph

4.1 Dijkstra

```
// Dijkstra's Algorithm in C++

#include <iostream>
#include <vector>

#define INT_MAX 10000000

using namespace std;

void DijkstrasTest();

int main() {
  DijkstrasTest();
  return 0;
}

class Node;
class Edge;

void Dijkstras();
vector<Node*>* AdjacentRemainingNodes(Node* node);
Node* ExtractSmallest(vector<Node*>& nodes);
int Distance(Node* node1, Node* node2);
bool Contains(vector<Node*>& nodes, Node* node);
void PrintShortestRouteTo(Node* destination);

vector<Node*> nodes;
vector<Edge*> edges;

class Node {
  public:
  Node(char id)
    : id(id), previous(NULL),
      distanceFromStart(INT_MAX) {
    nodes.push_back(this);
  }

  public:
  char id;
  Node* previous;
  int distanceFromStart;
};

class Edge {
  public:
  Edge(Node* node1, Node* node2, int distance)
    : node1(node1), node2(node2), distance(distance) {
    edges.push_back(this);
  }
  bool Connects(Node* node1, Node* node2) {
    return (
      (node1 == this->node1 &&
       node2 == this->node2) ||
      (node1 == this->node2 &&
       node2 == this->node1));
  }

  public:
  Node* node1;
  Node* node2;
  int distance;
};

//////////////////
void DijkstrasTest() {
  Node* a = new Node('a');
  Node* b = new Node('b');
  Node* c = new Node('c');
  Node* d = new Node('d');
  Node* e = new Node('e');
  Node* f = new Node('f');
  Node* g = new Node('g');

  Edge* e1 = new Edge(a, c, 1);
  Edge* e2 = new Edge(a, d, 2);
  Edge* e3 = new Edge(b, c, 2);
  Edge* e4 = new Edge(c, d, 1);
  Edge* e5 = new Edge(b, f, 3);
  Edge* e6 = new Edge(c, e, 3);
  Edge* e7 = new Edge(e, f, 2);
  Edge* e8 = new Edge(d, g, 1);
  Edge* e9 = new Edge(g, f, 1);

  a->distanceFromStart = 0;  // set start node
  Dijkstras();
  PrintShortestRouteTo(f);
}

//////////////////

void Dijkstras() {
  while (nodes.size() > 0) {
    Node* smallest = ExtractSmallest(nodes);
    vector<Node*>* adjacentNodes =
      AdjacentRemainingNodes(smallest);

    const int size = adjacentNodes->size();
    for (int i = 0; i < size; ++i) {
      Node* adjacent = adjacentNodes->at(i);
      int distance = Distance(smallest, adjacent) +
          smallest->distanceFromStart;

      if (distance < adjacent->distanceFromStart) {
        adjacent->distanceFromStart = distance;
        adjacent->previous = smallest;
      }
    }
    delete adjacentNodes;
  }
}

// Find the node with the smallest distance,
// remove it, and return it.
Node* ExtractSmallest(vector<Node*>& nodes) {
  int size = nodes.size();
  if (size == 0) return NULL;
  int smallestPosition = 0;
  Node* smallest = nodes.at(0);
  for (int i = 1; i < size; ++i) {
    Node* current = nodes.at(i);
    if (current->distanceFromStart <
      smallest->distanceFromStart) {
      smallest = current;
      smallestPosition = i;
    }
  }
  nodes.erase(nodes.begin() + smallestPosition);
  return smallest;
}

// Return all nodes adjacent to 'node' which are still
// in the 'nodes' collection.
vector<Node*>* AdjacentRemainingNodes(Node* node) {
  vector<Node*>* adjacentNodes = new vector<Node*>();
  const int size = edges.size();
  for (int i = 0; i < size; ++i) {
    Edge* edge = edges.at(i);
    Node* adjacent = NULL;
    if (edge->node1 == node) {
      adjacent = edge->node2;
    } else if (edge->node2 == node) {
      adjacent = edge->node1;
    }
    if (adjacent && Contains(nodes, adjacent)) {
      adjacentNodes->push_back(adjacent);
    }
  }
  return adjacentNodes;
}

// Return distance between two connected nodes
int Distance(Node* node1, Node* node2) {
  const int size = edges.size();
  for (int i = 0; i < size; ++i) {
    Edge* edge = edges.at(i);
    if (edge->Connects(node1, node2)) {
      return edge->distance;
    }
  }
  return -1;  // should never happen
}

// Does the 'nodes' vector contain 'node'
bool Contains(vector<Node*>& nodes, Node* node) {
  const int size = nodes.size();
  for (int i = 0; i < size; ++i) {
    if (node == nodes.at(i)) {
      return true;
    }
  }
  return false;
}

//////////////////

void PrintShortestRouteTo(Node* destination) {
  Node* previous = destination;
  cout << "Distance from start: "
     << destination->distanceFromStart << endl;
  while (previous) {
    cout << previous->id << " ";
```

```
    previous = previous->previous;
  }
  cout << endl;
}

// these two not needed
vector<Edge*>* AdjacentEdges(vector<Edge*>& Edges, Node* node);
void RemoveEdge(vector<Edge*>& Edges, Edge* edge);

vector<Edge*>* AdjacentEdges(vector<Edge*>& edges, Node* node) {
  vector<Edge*>* adjacentEdges = new vector<Edge*>();

  const int size = edges.size();
  for (int i = 0; i < size; ++i) {
    Edge* edge = edges.at(i);
    if (edge->node1 == node) {
      cout << "adjacent: " << edge->node2->id << endl;
      adjacentEdges->push_back(edge);
    } else if (edge->node2 == node) {
      cout << "adjacent: " << edge->node1->id << endl;
      adjacentEdges->push_back(edge);
    }
  }
  return adjacentEdges;
}

void RemoveEdge(vector<Edge*>& edges, Edge* edge) {
  vector<Edge*>::iterator it;
  for (it = edges.begin(); it < edges.end(); ++it) {
    if (*it == edge) {
      edges.erase(it);
      return;
    }
  }
}
```

4.2 FloydWarshall

```
// Floyd-Warshall
// by. MiohitoKiri5474
#include<bits/stdc++.h>
#define maxN 5005
typedef long long LL;

using namespace std;

LL graph[maxN][maxN];

int main(){
  ios::sync_with_stdio ( false );
  cin.tie ( 0 );
  cout.tie ( 0 );

  int n, m, u, v, w, q;
  cin >> n >> m;

  memset ( graph, 0x3f3f3f, sizeof graph );
  while ( m-- ){
    cin >> u >> v >> w;
    graph[u][v] = graph[v][u] = w;
  }

  for ( int i = 0 ; i < n ; i++ )
    graph[1][1] = 0;

  for ( int k = 0 ; k < n ; k++ )
    for ( int i = 0 ; i < n ; i++ )
      for ( int j = 0 ; j < n ; j++ )
        if ( graph[i][j] > graph[i][k] + graph[k][j] )
          graph[i][j] = graph[i][k] + graph[k][j];

  cin >> q;
  while ( q-- ){
    cin >> u >> v;
    cout << graph[u][v] << '\n';
  }
}
```

4.3 全局最小割

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

4.4 dsu

```
int p[100005], rk[100005];
int find(int n){
  if(p[n] == n)return n;
  return find(p[n]);
}
void union_set(int x, int y){
  int rx = find(x);
  int ry = find(y);
  if(rx != ry){
    if(rk[rx] > rk[ry]){
      p[ry] = rx;
    }
    else if(rk[ry] > rk[rx]){
      p[rx] = ry;
    }
    else {
      p[rx] = ry;
      rk[ry] += 1;
    }
  }
}
```

4.5 spfa

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

using namespace std;

#define maxN 100005
#define pb push_back
typedef pair < int, int > pii;

vector < pii > edges[maxN];
int dis[maxN];
bool inQueue[maxN]; // 紀錄是否已經在queue中

inline void SPFA ( int start ){
  memset ( dis, 0x3f3f3f, sizeof dis );
  dis[start]=0;
  queue < int > q;
  q.push ( start );
  inQueue[start] = true;
  while ( !q.empty() ){
    int now = q.front();
    q.pop();
    inQueue[now] = false; // 紀錄已經取出
    random_shuffle ( edges.begin(), edges.end() );//打亂順序 有些測資會故意讓算法爆掉
    for ( auto i: edges[now] ){ // 跑過所有可以被now連結到的點
      if ( dis[i.first] > dis[now] + i.second ){
        dis[i.first] = dis[now] + i.second;
        if ( !inQueue[i.first] ){
          // 如果點沒有在queue中，再加入queue，並記錄已經在queue中
          inQueue[i.first] = true;
          q.push ( i.first );
        }
      }
    }
  }
}
```

4.6 kruskal

```
// Kruskal
// by. MiohitoKiri5474
#include<bits/stdc++.h>

using namespace std;

#define maxN 10005
#define pb push_back
typedef pair < int, int > pii;

int dis[maxN];

inline void init ( void ){
  for ( int i = 0 ; i < maxN ; i++ )
    dis[i] = i;
}

inline int find ( int a ){
  return dis[a] == a ? a : dis[a] = find ( dis[a] );
}

inline void Union ( int a, int b ){
  dis[find ( a )] = find ( b );
}

inline bool same ( int a, int b ){
  return find ( a ) == find ( b );
}

struct node{
  int u, v, w;
};

inline bool cmp ( node a, node b ){
  return a.w < b.w;
}

vector < node > edges;
vector < pii > mst[maxN];

inline void Kruskal ( void ){
  sort ( edges.begin(), edges.end(), cmp );
  for ( auto i: edges ){ // C++ 11寫法，不懂再來問
    if ( same ( i.u, i.v ) )
      continue;
    Union ( i.u, i.v );
    mst[i.u].pb ( pii ( i.v, i.w ) );
    mst[i.v].pb ( pii ( i.u, i.w ) );
  }
}
```

4.7 bellmanford

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

using namespace std;

#define maxN 100005
#define pb push_back
typedef pair < int, int > pii;

vector < pii > edges[maxN];
int dis[maxN];

inline void BellmanFord ( int start ){
  memset ( dis, 0x3f3f3f, sizeof dis );
    dis[start]=0;
  queue < int > q;
  q.push ( start );
  while ( !q.empty() ){
    int now = q.front();
    q.pop();
    for ( auto i: edges[now] ){
      if ( dis[i.first] > dis[now] + i.second ){ // 檢查是否能更新
        q.push ( i.first );
        dis[i.first] = dis[now] + i.second;
      }
    }
  }
}
```

4.8 kruskalMST

```cpp
// C++ program for Kruskal's algorithm to find Minimum
// Spanning Tree of a given connected, undirected and
// weighted graph
#include<bits/stdc++.h>
using namespace std;

// Creating shortcut for an integer pair
typedef pair<int, int> iPair;

// Structure to represent a graph
struct Graph
{
  int V, E;
  vector< pair<int, iPair> > edges;

  // Constructor
  Graph(int V, int E)
  {
    this->V = V;
    this->E = E;
  }

  // Utility function to add an edge
  void addEdge(int u, int v, int w)
  {
    edges.push_back({w, {u, v}});
  }

  // Function to find MST using Kruskal's
  // MST algorithm
  int kruskalMST();
};

// To represent Disjoint Sets
struct DisjointSets
{
  int *parent, *rnk;
  int n;

  // Constructor.
  DisjointSets(int n)
  {
    // Allocate memory
    this->n = n;
    parent = new int[n+1];
    rnk = new int[n+1];

    // Initially, all vertices are in
    // different sets and have rank 0.
    for (int i = 0; i <= n; i++)
    {
      rnk[i] = 0;

      //every element is parent of itself
      parent[i] = i;
    }
  }

  // Find the parent of a node 'u'
  // Path Compression
  int find(int u)
  {
    /* Make the parent of the nodes in the path
    from u--> parent[u] point to parent[u] */
    if (u != parent[u])
      parent[u] = find(parent[u]);
    return parent[u];
  }

  // Union by rank
  void merge(int x, int y)
  {
    x = find(x), y = find(y);

    /* Make tree with smaller height
    a subtree of the other tree */
    if (rnk[x] > rnk[y])
      parent[y] = x;
    else // If rnk[x] <= rnk[y]
      parent[x] = y;

    if (rnk[x] == rnk[y])
      rnk[y]++;
  }
};

/* Functions returns weight of the MST*/

int Graph::kruskalMST()
{
  int mst_wt = 0; // Initialize result

  // Sort edges in increasing order on basis of cost
  sort(edges.begin(), edges.end());

  // Create disjoint sets
  DisjointSets ds(V);

  // Iterate through all sorted edges
  vector< pair<int, iPair> >::iterator it;
  for (it=edges.begin(); it!=edges.end(); it++)
  {
    int u = it->second.first;
    int v = it->second.second;

    int set_u = ds.find(u);
    int set_v = ds.find(v);

    // Check if the selected edge is creating
    // a cycle or not (Cycle is created if u
    // and v belong to same set)
    if (set_u != set_v)
    {
      // Current edge will be in the MST
      // so print it
      cout << u << " - " << v << endl;

      // Update MST weight
      mst_wt += it->first;

      // Merge two sets
      ds.merge(set_u, set_v);
    }
  }

  return mst_wt;
}

// Driver program to test above functions
int main()
{
  /* Let us create above shown weighted
  and undirected graph */
  int V = 9, E = 14;
  Graph g(V, E);

  // making above shown graph
  g.addEdge(0, 1, 4);
  g.addEdge(0, 7, 8);
  g.addEdge(1, 2, 8);
  g.addEdge(1, 7, 11);
  g.addEdge(2, 3, 7);
  g.addEdge(2, 8, 2);
  g.addEdge(2, 5, 4);
  g.addEdge(3, 4, 9);
  g.addEdge(3, 5, 14);
  g.addEdge(4, 5, 10);
  g.addEdge(5, 6, 2);
  g.addEdge(6, 7, 1);
  g.addEdge(6, 8, 6);
  g.addEdge(7, 8, 7);

  cout << "Edges of MST are \n";
  int mst_wt = g.kruskalMST();

  cout << "\nWeight of MST is " << mst_wt;

  return 0;
}
```

4.9 PrimeMST

```cpp
// Prim
// by. MiohitoKiri5474
#include<bits/stdc++.h>

using namespace std;

#define pb push_back
#define F first
#define S second
typedef pair < int, int > pii;
typedef pair < int, pii > pipii;
const int maxN = 100005;

int dis[maxN], sz[maxN];

inline void init ( void ){
  for ( int i = 0 ; i < maxN ; i++ )
    dis[i] = i, sz[i] = 1;
}

int find ( int n ){
  return dis[n] == n ? n : dis[n] = find ( dis[n] );
}

inline void Union ( int a, int b ){
  a = find ( a ), b = find ( b );
  dis[a] = b;
  sz[b] += sz[a];
}

inline bool same ( int a, int b ){
  return find ( a ) == find ( b );
}

vector < pii > edges[maxN], mst[maxN];
bool pushed[maxN];

inline void Pirm ( int n ){
  priority_queue < pipii, vector < pipii >, greater < pipii > > pq;
  for ( auto i: edges[0] )
    pq.push ( pipii ( i.S, pii ( i.F, 0 ) ) );
  pushed[0] = true;
  init();
  while ( sz[find ( 0 )] != n ){
    pipii top = pq.top();
    pq.pop();
    while ( same ( 0, top.S.F ) ){
      top = pq.top();
      pq.pop();
    }
    int u = top.S.F, v = top.S.S;

```

```
    mst[u].pb ( pii ( v, top.F ) );
    mst[v].pb ( pii ( u, top.F ) );
    Union ( u, v );

    if ( !pushed[u] ){
      for ( auto i: edges[u] )
        pq.push ( pipii ( i.S, pii ( i.F, u
            ) ) );
      pushed[u] = true;
    }
    if ( !pushed[v] ){
      for ( auto i: edges[v] )
        pq.push ( pipii ( i.S, pii ( i.F, v
            ) ) );
      pushed[v] = true;
    }
  }
}
```

4.10 treeISO

```
const int MAXN=100005;
const long long X=12327,P=0xdefaced;
vector<int> g[MAXN];
bool vis[MAXN];
long long dfs(int u){//hash ver
  vis[u]=1;
  vector<long long> tmp;
  for(auto v:g[u])if(!vis[v])tmp.PB(dfs(v));
  if(tmp.empty())return 177;
  long long ret=4931;
  sort(tmp.begin(),tmp.end());
  for(auto v:tmp)ret=((ret*X)^v)%P;
  return ret;
}
//----------------------------
string dfs(int x,int p){
  vector<string> c;
  for(int y:g[x])
    if(y!=p)c.emplace_back(dfs(y,x));
  sort(c.begin(),c.end());
  string ret("(");
  for(auto &s:c)ret+=s;
  ret+=")";
  return ret;
}
```

4.11 MaximumClique

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

5 Linear Programming

5.1 simplex

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

6 Number Theory

6.1 bit set

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

6.2 FFT

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

6.3 質因數分解

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

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

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

void smallfactor(LL n, vector<LL> &v) {
  if(n<MAXPRIME) {
```

```
    while(isp[(int)n]) {
      v.push_back(isp[(int)n]);
      n/=isp[(int)n];
    }
    v.push_back(n);
  } else {
    for(int i=0;i<primecnt&&prime[i]*prime[i
        ]<=n;++i) {
      while(n%prime[i]==0) {
        v.push_back(prime[i]);
        n/=prime[i];
      }
    }
    if(n!=1) v.push_back(n);
  }
}

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

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

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

6.4 Matrix

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

7 String

7.1 reverseBWT

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

7.2 suffix array lcp

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

7.3 kmp

```
#include<bits/stdc++.h>
using namespace std;
//#define int long long
#define IOS ios::sync_with_stdio(0);cin.tie
    (0);
vector<int> v(1005);
void build_next(string T){
  v[0] = 0;
  int pre_len = 0;
  int i = 1;
  while(i<T.length()){
    if(T[pre_len] == T[i]){
      pre_len++;
      v.push_back(pre_len);
      i++;
    } else {
      if(pre_len > 0){
        pre_len = v[pre_len-1];
      }
      if(pre_len == 0){
        v.push_back(0);
        i++;
      }
    }
  }
}
bool KMP(string S, string T){
  build_next(T);
  int i=0,j=0;
  while(i<S.length()){
    if(S[i] == T[j]){
```

```
      i++;
      j++;
    } else if(j>0){
      j = v[j-1];
    } else {
      i++;
    }

    if(j == T.length()){
      return 1;
    }
  }
  return 0;
}


int32_t main(){
  int k;
  cin>>k;
  while(k--){
    int q;
    string s;
    cin>>s;
    cin>>q;
    while(q--){
      string t;
      cin>>t;
      if(KMP(s,t) == 1)cout<<"y"<<endl;
      else cout<<"n"<<endl;

    }
  }
  return 0;
}
```

7.4 hash

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

7.5 AC 自動機

```
template<char L='a',char R='z'>
class ac_automaton{
  struct joe{
    int next[R-L+1],fail,efl,ed,cnt_dp,vis;
    joe():ed(0),cnt_dp(0),vis(0){
      for(int i=0;i<=R-L;++i)next[i]=0;
    }
  };
public:
  std::vector<joe> S;
  std::vector<int> q;
  int qs,qe,vt;
  ac_automaton():S(1),qs(0),qe(0),vt(0){}
  void clear(){
    q.clear();
    S.resize(1);
    for(int i=0;i<=R-L;++i)S[0].next[i]=0;
    S[0].cnt_dp=S[0].vis=qs=qe=vt=0;
  }
  void insert(const char *s){
    int o=0;
    for(int i=0,id;s[i];++i){
      id=s[i]-L;
      if(!S[o].next[id]){
        S.push_back(joe());
        S[o].next[id]=S.size()-1;
      }
      o=S[o].next[id];
    }
    ++S[o].ed;
  }
  void build_fail(){
    S[0].fail=S[0].efl=-1;
    q.clear();
    q.push_back(0);
    ++qe;
    while(qs!=qe){
      int pa=q[qs++],id,t;
      for(int i=0;i<=R-L;++i){
        t=S[pa].next[i];
        if(!t)continue;
        id=S[pa].fail;
        while(~id&&!S[id].next[i])id=S[id].
            fail;
        S[t].fail=~id?S[id].next[i]:0;
        S[t].efl=S[S[t].fail].ed?S[t].fail:S
            [S[t].fail].efl;
        q.push_back(t);
        ++qe;
      }
    }
  }
  /*DP出每個前綴在字串s出現的次數並傳回所有
      字串被s匹配成功的次數O(N+M)*/
  int match_0(const char *s){
    int ans=0,id,p=0,i;
    for(i=0;s[i];++i){
      id=s[i]-L;
      while(!S[p].next[id]&&p)p=S[p].fail;
      if(!S[p].next[id])continue;
      p=S[p].next[id];
      ++S[p].cnt_dp;/*匹配成功則它所有後綴都
          可以被匹配(DP計算)*/
    }
    for(i=qe-1;i>=0;--i){
      ans+=S[q[i]].cnt_dp*S[q[i]].ed;
      if(~S[q[i]].fail)S[S[q[i]].fail].
          cnt_dp+=S[q[i]].cnt_dp;
    }
    return ans;
  }
  /*多串匹配走efl邊並傳回所有字串被s匹配成功
      的次數O(N*M^1.5)*/
  int match_1(const char *s)const{
    int ans=0,id,p=0,t;
    for(int i=0;s[i];++i){
      id=s[i]-L;
      while(!S[p].next[id]&&p)p=S[p].fail;
      if(!S[p].next[id])continue;
      p=S[p].next[id];
      if(S[p].ed)ans+=S[p].ed;
      for(t=S[p].efl;~t;t=S[t].efl){
        ans+=S[t].ed;/*因為都走efl邊所以保證
            匹配成功*/
      }
    }
    return ans;
  }
  /*枚舉(s的子字串nA)的所有相異字串各恰一次
      並傳回次數O(N*M^(1/3))*/
  int match_2(const char *s){
    int ans=0,id,p=0,t;
    ++vt;
    /*把戳記vt+=1，只要vt沒溢位，所有S[p].
        vis==vt就會變成false
    這種利用vt的方法可以O(1)歸零vis陣列*/
    for(int i=0;s[i];++i){
      id=s[i]-L;
      while(!S[p].next[id]&&p)p=S[p].fail;
      if(!S[p].next[id])continue;
      p=S[p].next[id];
      if(S[p].ed&&S[p].vis!=vt){
        S[p].vis=vt;
        ans+=S[p].ed;
      }
      for(t=S[p].efl;~t&&S[t].vis!=vt;t=S[t
          ].efl){
        S[t].vis=vt;
        ans+=S[t].ed;/*因為都走efl邊所以保證
            匹配成功*/
      }
    }
    return ans;
  }
  /*把AC自動機變成真的自動機*/
  void evolution(){
    for(qs=1;qs!=qe;){
      int p=q[qs++];
      for(int i=0;i<=R-L;++i)
        if(S[p].next[i]==0)S[p].next[i]=S[S[
            p].fail].next[i];
    }
  }
};
```

7.6 minimal string rotation

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

7.7 Z

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

8 Tarjan

8.1 橋連通分量

```
#define N 1005
struct edge{
  int u,v;
  bool is_bridge;
  edge(int u=0,int v=0):u(u),v(v),is_bridge
      (0){}
};
vector<edge> E;
vector<int> G[N];// 1-base
int low[N],vis[N],Time;
int bcc_id[N],bridge_cnt,bcc_cnt;// 1-base
int st[N],top;//BCC用
void add_edge(int u,int v){
  G[u].push_back(E.size());
  E.emplace_back(u,v);
  G[v].push_back(E.size());
  E.emplace_back(v,u);
}
void dfs(int u,int re=-1){//u當前點，re為u連
    接前一個點的邊
  int v;
  low[u]=vis[u]=++Time;
  st[top++]=u;
  for(int e:G[u]){
```

```
    v=E[e].v;
    if(!vis[v]){
      dfs(v,e^1);//e^1反向邊
      low[u]=min(low[u],low[v]);
      if(vis[u]<low[v]){
        E[e].is_bridge=E[e^1].is_bridge=1;
        ++bridge_cnt;
      }
    }else if(vis[v]<vis[u]&&e!=re)
      low[u]=min(low[u],vis[v]);
  }
  if(vis[u]==low[u]){//處理BCC
    ++bcc_cnt;// 1-base
    do bcc_id[v=st[--top]]=bcc_cnt;//每個點所在的BCC
    while(v!=u);
  }
}
void bcc_init(int n){
  Time=bcc_cnt=bridge_cnt=top=0;
  E.clear();
  for(int i=1;i<=n;++i){
    G[i].clear();
    vis[i]=bcc_id[i]=0;
  }
}
```

9 Tree Problem

9.1 LCA

```
const int MAXN=100000; // 1-base
const int MLG=17; //log2(MAXN)+1;
int pa[MLG+2][MAXN+5];
int dep[MAXN+5];
vector<int> G[MAXN+5];
void dfs(int x,int p=0){//dfs(root);
  pa[0][x]=p;
  for(int i=0;i<=MLG;++i)
    pa[i+1][x]=pa[i][pa[i][x]];
  for(auto &i:G[x]){
    if(i==p)continue;
    dep[i]=dep[x]+1;
    dfs(i,x);
  }
}
inline int jump(int x,int d){
  for(int i=0;i<=MLG;++i)
    if((d>>i)&1) x=pa[i][x];
  return x;
}
inline int find_lca(int a,int b){
  if(dep[a]>dep[b])swap(a,b);
  b=jump(b,dep[b]-dep[a]);
  if(a==b)return a;
  for(int i=MLG;i>=0;--i){
    if(pa[i][a]!=pa[i][b]){
      a=pa[i][a];
      b=pa[i][b];
    }
  }
  return pa[0][a];
}
```

9.2 link cut tree

```
struct splay_tree{
  int ch[2],pa;//子節點跟父母
  bool rev;//反轉的懶惰標記
  splay_tree():pa(0),rev(0){ch[0]=ch[1]=0;}
};
vector<splay_tree> nd;
//有的時候用vector會TLE，要注意
//這邊以node[0]作為null節點
bool isroot(int x){//判斷是否為這棵splay tree的根
  return nd[nd[x].pa].ch[0]!=x&&nd[nd[x].pa].ch[1]!=x;
}
void down(int x){//懶惰標記下推
  if(nd[x].rev){
    if(nd[x].ch[0])nd[nd[x].ch[0]].rev^=1;
    if(nd[x].ch[1])nd[nd[x].ch[1]].rev^=1;
    swap(nd[x].ch[0],nd[x].ch[1]);
    nd[x].rev=0;
  }
}
void push_down(int x){//所有祖先懶惰標記下推
  if(!isroot(x))push_down(nd[x].pa);
  down(x);
}
void up(int x){}//將子節點的資訊向上更新
void rotate(int x){//旋轉，會自行判斷轉的方向
  int y=nd[x].pa,z=nd[y].pa,d=(nd[y].ch[1]==x);
  nd[x].pa=z;
  if(!isroot(y))nd[z].ch[nd[z].ch[1]==y]=x;
  nd[y].ch[d]=nd[x].ch[d^1];
  nd[nd[y].ch[d]].pa=y;
  nd[y].pa=x,nd[x].ch[d^1]=y;
  up(y),up(x);
}
void splay(int x){//將x伸展到splay tree的根
  push_down(x);
  while(!isroot(x)){
    int y=nd[x].pa;
    if(!isroot(y)){
      int z=nd[y].pa;
      if((nd[z].ch[0]==y)^(nd[y].ch[0]==x))
        rotate(y);
      else rotate(x);
    }
    rotate(x);
  }
}
int access(int x){
  int last=0;
  while(x){
    splay(x);
    nd[x].ch[1]=last;
    up(x);
    last=x;
    x=nd[x].pa;
  }
  return last;//access後splay tree的根
}
void access(int x,bool is=0){//is=0就是一般的access
  int last=0;
  while(x){
    splay(x);
    if(is&&!nd[x].pa){
      //printf("%d\n",max(nd[last].ma,nd[nd[x].ch[1]].ma));
    }
    nd[x].ch[1]=last;
    up(x);
    last=x;
    x=nd[x].pa;
  }
}
void query_edge(int u,int v){
  access(u);
  access(v,1);
}
void make_root(int x){
  access(x),splay(x);
  nd[x].rev^=1;
}
void make_root(int x){
  nd[access(x)].rev^=1;
  splay(x);
}
void cut(int x,int y){
  make_root(x);
  access(y);
  splay(y);
  nd[y].ch[0]=0;
  nd[x].pa=0;
}
void cut_parents(int x){
  access(x);
  splay(x);
  nd[nd[x].ch[0]].pa=0;
  nd[x].ch[0]=0;
}
void link(int x,int y){
  make_root(x);
  nd[x].pa=y;
}
int find_root(int x){
  x=access(x);
  while(nd[x].ch[0])x=nd[x].ch[0];
  splay(x);
  return x;
}
int query(int u,int v){
//傳回uv路徑splay tree的根結點
//這種寫法無法求LCA
  make_root(u);
  return access(v);
}
int query_lca(int u,int v){
//假設求鏈上點權的總和，sum是子樹的權重和，data是節點的權重
  access(u);
  int lca=access(v);
  splay(u);
  if(u==lca){
    //return nd[lca].data+nd[nd[lca].ch[1]].sum
  }else{
    //return nd[lca].data+nd[nd[lca].ch[1]].sum+nd[u].sum
  }
}
struct EDGE{
  int a,b,w;
}e[10005];
int n;
vector<pair<int,int>> G[10005];
//first表示子節點，second表示邊的編號
int pa[10005],edge_node[10005];
//pa是父母節點，暫存用的，edge_node是每個編被存在哪個點裡面的陣列
void bfs(int root){
//在建構的時候把每個點都設成一個splay tree
  queue<int > q;
  for(int i=1;i<=n;++i)pa[i]=0;
  q.push(root);
  while(q.size()){
    int u=q.front();
    q.pop();
    for(auto P:G[u]){
      int v=P.first;
      if(v!=pa[u]){
        pa[v]=u;
        nd[v].pa=u;
        nd[v].data=e[P.second].w;
        edge_node[P.second]=v;
        up(v);
        q.push(v);
      }
    }
  }
}
void change(int x,int b){
  splay(x);
  //nd[x].data=b;
  up(x);
}
```

10 other

10.1 上下最大正方形

```
void solve(int n,int a[],int b[]){// 1-base
  int ans=0;
  deque<int>da,db;
  for(int l=1,r=1;r<=n;++r){
    while(da.size()&&a[da.back()]>=a[r]){
      da.pop_back();
    }
```

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

10.2 WhatDay

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

10.3 最大矩形

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

11 zformula

11.1 formula

11.1.1 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

11.1.2 圖論

1. 對於平面圖，$F = E - V + C + 1$，C 是連通分量數
2. 對於平面圖，$E \leq 3V - 6$
3. 對於連通圖 G，最大獨立點集的大小設為 I(G)，最大匹配大小設為 M(G)，最小點覆蓋設為 Cv(G)，最小邊覆蓋設為 Ce(G)。對於任意連通圖：
   (a) $I(G) + Cv(G) = |V|$
   (b) $M(G) + Ce(G) = |V|$
4. 對於連通二分圖：
   (a) $I(G) = Cv(G)$
   (b) $M(G) = Ce(G)$
5. 最大權閉合圖：
   (a) $C(u, v) = \infty, (u, v) \in E$
   (b) $C(S, v) = W_v, W_v > 0$
   (c) $C(v, T) = -W_v, W_v < 0$
   (d) ans=$\sum_{W_v>0} W_v - flow(S, T)$
6. 最大密度子圖：
   (a) 求 $max\left(\frac{W_e+W_v}{|V'|}\right), e \in E', v \in V'$
   (b) $U = \sum_{v\in V} 2W_v + \sum_{e\in E} W_e$
   (c) $C(u, v) = W_{(u,v)}, (u, v) \in E$，雙向邊
   (d) $C(S, v) = U, v \in V$
   (e) $D_u = \sum_{(u,v)\in E} W_{(u,v)}$
   (f) $C(v, T) = U + 2g - D_v - 2W_v, v \in V$
   (g) 二分搜 g：
   $l = 0, r = U, eps = 1/n^2$
   if$((U \times |V| - flow(S, T))/2 > 0)\, l = mid$
   else $r = mid$
   (h) ans=$min\_cut(S, T)$
   (i) $|E| = 0$ 要特殊判斷
7. 弦圖：
   (a) 點數大於 3 的環都要有一條弦
   (b) 完美消除序列從後往前依次給每個點染色，給每個點染上可以染的最小顏色
   (c) 最大團大小 = 色數
   (d) 最大獨立集: 完美消除序列從前往後能選就選
   (e) 最小團覆蓋: 最大獨立集的點和他延伸的邊構成
   (f) 區間圖是弦圖
   (g) 區間圖的完美消除序列: 將區間按造又端點由小到大排序
   (h) 區間圖染色: 用線段樹做

11.1.3 dinic 特殊圖複雜度

1. 單位流：$O\left(min\left(V^{3/2}, E^{1/2}\right) E\right)$
2. 二分圖：$O\left(V^{1/2}E\right)$

11.1.4 0-1 分數規劃

$x_i = \{0, 1\}$，x_i 可能會有其他限制，求 $max\left(\frac{\sum B_i x_i}{\sum C_i x_i}\right)$

1. $D(i, g) = B_i - g \times C_i$
2. $f(g) = \sum D(i, g)x_i$
3. $f(g) = 0$ 時 g 為最佳解，$f(g) < 0$ 沒有意義
4. 因為 $f(g)$ 單調可以二分搜 g
5. 或用 Dinkelbach 通常比較快

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

11.1.5 學長公式

1. $\sum_{d|n} \phi(n) = n$
2. $g(n) = \sum_{d|n} f(d) \Rightarrow f(n) = \sum_{d|n} \mu(d) \times g(n/d)$
3. Harmonic series $H_n = \ln(n) + \gamma + 1/(2n) - 1/(12n^2) + 1/(120n^4)$
4. $\gamma = 0.57721566490153286060651209008240243104215$
5. 格雷碼 $= n \oplus (n >> 1)$
6. $SG(A + B) = SG(A) \oplus SG(B)$
7. 選轉矩陣 $M(\theta) = \begin{pmatrix} cos\theta & -sin\theta \\ sin\theta & cos\theta \end{pmatrix}$

11.1.6 基本數論

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

11.1.7 排組公式

1. k 卡特蘭 $\frac{C_n^{kn}}{n(k-1)+1} \cdot C_m^n = \frac{n!}{m!(n-m)!}$
2. $H(n, m) \cong x_1 + x_2 \ldots + x_n = k, num = C_k^{n+k-1}$
3. Stirling number of 2^{nd},n 人分 k 組方法數目
   (a) $S(0, 0) = S(n, n) = 1$
   (b) $S(n, 0) = 0$
   (c) $S(n, k) = kS(n - 1, k) + S(n - 1, k - 1)$
4. Bell number,n 人分任意多組方法數目
   (a) $B_0 = 1$
   (b) $B_n = \sum_{i=0}^{n} S(n, i)$
   (c) $B_{n+1} = \sum_{k=0}^{n} C_k^n B_k$
   (d) $B_{p+n} \equiv B_n + B_{n+1} mod p$, 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) $\sum_{k\leq l} \binom{l-k}{m}\binom{s}{k-n}(-1)^k = (-1)^{l+m}\binom{s-m-1}{l-n-m}$
   (e) $\sum_{0\leq k\leq l} \binom{l-k}{m}\binom{q+k}{n} = \binom{l+q+1}{m+n+1}$
   (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) $\sum_{k\leq n} \binom{r+k}{k} = \binom{r+n+1}{n}$
   (i) $\sum_{0\leq k\leq n} \binom{k}{m} = \binom{n+1}{m+1}$
   (j) $\sum_{k\leq m} \binom{m+r}{k}x^k y^k = \sum_{k\leq m} \binom{-r}{k}(-x)^k(x + y)^{m-k}$

11.1.8 冪次, 冪次和

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

11.1.9 Burnside's lemma

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

11.1.10 Count on a tree

1. Rooted tree: $s_{n+1} = \frac{1}{n}\sum_{i=1}^{n}(i\times a_i\times \sum_{j=1}^{\lfloor n/i\rfloor} a_{n+1-i\times j})$
2. Unrooted tree:
   (a) Odd:$a_n - \sum_{i=1}^{n/2} a_i a_{n-i}$
   (b) Even:$Odd + \frac{1}{2}a_{n/2}(a_{n/2}+1)$
3. Spanning Tree
   (a) 完全圖 n^n-2
   (b) 一般圖 (Kirchhoff's theorem)$M[i][i] = degree(V_i)$,$M[i][j] = -1$,if have $E(i,j)$,0 if no edge. delete any one row and col in A, $ans = det(A)$

12 Интернационал

12.1 保佑

```
//                  _oo0oo_
//                 o8888888o
//                 88" . "88
//                 (| -_- |)
//                 0\  =  /0
//               ___/`---'\___
//             .' \\|     |// '.
//            / \\|||  :  |||// \
//           / _||||| -:- |||||- \
//          |   | \\\  -  /// |   |
//          | \_|  ''\---/''  |_/ |
//          \  .-\__  '-'  ___/-. /
//        ___'. .'  /--.--\  `. .'___
//     ."" '<  `.___\_<|>_/___.' >' "".
//    | | :  `- \`.;`\ _ /`;.`/ - ` : | |
//    \  \ `_.   \_ __\ /__ _/   .-` /  /
// ===`-.____`.___ \_____/___.-`___.-'===
//                 `=---='
//
// ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
//         佛祖保佑          永無BUG
```

```
#     ┌─┐       ┌─┐
#  ┌──┘ └───────┘ └──┐
#  │                 │
#  │       ──        │
#  │   ─┬┘       └┬─  │
#  │                 │
#  │       ─┴─       │
#  │                 │
#  └───┐         ┌───┘
#      │         │
#      │         │
#      │         │
#      │         └─────────────┐
#      │                       │
#      │                       ├─┐
#      │                       │ │
#      │                       └─┘
#      └──┐  ┌─────────┬──┐  ┌──┘
#         │ ─┤ ─┤       │ ─┤ ─┤
#         └──┴──┘       └──┴──┘
#                神獸保佑 永無BUG!
```

```
// ##             ###################
// ##             ##
// ##             ##
// ##             ##
// ##             ##
// ##             ##
// ##             ##
// ##             ##
// ###################################
//                ##                ##
//                ##                ##
//                ##                ##
//                ##                ##
//                ##                ##
//                ##                ##
//                ##                ##
// #################              ##
//
//        元首保佑 永無BUG
```

```
//
//                    __
// _____ __ __      < @ \      _  __ _____
// \____\\_\\_)    /**|    (_//_//____/
//  \____\(_)\*\/****\/*/(_)/____/
//   \____\_/_\*\****/*/_\_/____/
//         u/u/u|****|u\u\u
//          u/u/|****|\u\u
//              |*||*|
//              |  |
//             /+--+\
//            || 卐 ||
//             \\==//
//
//          神獸保佑 永無BUG
```

ACM ICPC Team Reference - NCUnknown

Contents

ACM ICPC Judge Test - NCUnknown

C++ Resource Test

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

namespace system_test {

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

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

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

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

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

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

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

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

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

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

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

} // namespace system_test

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