

<div> <div>Basic Module</div> <div> <pre> #include<iostream> #define fr(i,a,b) for(i=a;i<=b;+i) using namespace std; int main(){ } Bidirectional BFS #include<iostream> #include<string> #define fr(i,a,b) for(i=a;i<=b;+i) using namespace std; struct State{ }; const int maxh=1<<20; int u[maxh]; State ht[maxh],queue[2][maxh/2]; int p[2],q[2],found[2]; int cal(State r){ } bool same(State a,State b){ } int hash(State r){ int i=cal(r); while(u[i]&&!same(r,ht[i])) i=(i+1)&(maxh-1); ht[i]=r; return i; } void add(State s,int tag){ int w=hash(s); if(u[w]==0 !(tag*2-1)*u[w]<0){ queue[tag][++q[tag]]=s; if(u[w]){ found[tag]=q[tag]; found[1-tag]=abs(u[w]); } else u[w]=q[tag]*(tag*2-1); } } void extend(State r,int tag){ } void BFS2(State s,State t){ memset(u,q[0]=q[1]=found[0]=found[1]=0,sizeof(u)); p[0]=p[1]=1; add(s,0); add(t,1); for(int tag=0;p[0]<=q[0]&&p[1]<=q[1]&&found[0]==0;tag=1-tag) for(int k=q[tag];p[tag]<=k&&found[0]==0;p[tag]++) extend(queue[tag][p[tag]],tag); } int main(){}</pre> </div> <div> <div>Big Number</div> <div> <pre> #include<iostream> #define fr(i,a,b) for(i=a;i<=b;+i) const int maxsize=5002; const int base=1000; using namespace std; struct BigNumber{ int x[maxsize],size,i,j; BigNumber(int t):size(0){ memset(x,0,sizeof(x)); for(;t/=base) x[size++]=t%base; } BigNumber():size(0){ memset(x,0,sizeof(x)); } BigNumber(string s){ memset(x,0,sizeof(x)); int j=1,k=0; for(i=s.size()-1;i>=0;i--){ x[k]+=(s[i]-'0')*j; j*=10; if(j>=base){ j=1; k++; } } } void print(){ int i=size?size-1:0; printf("%d",x[i]); while(i--){ for(j=10;j*x[i]<base&&j<base;j*=10) printf("0"); printf("%d",x[i]); } } BigNumber operator+(BigNumber r){ if(r.size<size) r.size=size; fr(i,0,r.size-1){ r.x[i]+=x[i]; if(r.x[i]>=base){ r.x[i]-=base; r.x[i+1]++; } } r.resize(r.size+1); return r; } BigNumber operator-(BigNumber r){ BigNumber tmp; fr(i,0,size-1){ tmp.x[i]+=x[i]-r.x[i]; if(tmp.x[i]<0){ tmp.x[i]+=base; tmp.x[i+1]--; } } tmp.resize(size); return tmp; } BigNumber operator*(int r){ int i=0,k=0; BigNumber tmp; while(i<size k){ k+=x[i]*r; tmp.x[i++]=k%base; k/=base; } tmp.size=i; return tmp; } BigNumber operator*(BigNumber r){ BigNumber tmp; fr(i,0,size-1) fr(j,0,r.size-1) tmp.x[i+j]+=x[i]*r.x[j]; fr(i,0,r.size+size-1){ tmp.x[i+1]+=tmp.x[i]/base; tmp.x[i]%=base; } tmp.resize(r.size+size); return tmp; } BigNumber operator/(int r){ int i,k=0; BigNumber tmp; for(i=size-1;i>=0;i--){ k=k*base+x[i]; tmp.x[i]=k/r; k%=r; } tmp.resize(size); return tmp; } BigNumber operator/(BigNumber r){ int i,j,k,key,temp,len=r.size,para=len<10?len:10; double a=r.toDouble(len-1,para)-1e-8,b; BigNumber ans,tmp=*this; for(i=size-len;i>=0;i--){ key=int(toDouble(i+len,para)*base/a); if(key) fr(j,0,len-1){ </pre> </div> </div> </div>	<div> <div></div> <div> <pre> x[i+j]-=r.x[j]*key; x[i+j+1]-=temp=(base-1-x[i+j])/base; x[i+j]+=temp*base; } do{ for(k=len;k>=0;k--) if(x[k+i]!=r.x[k]) break; if(k>=0&&x[k+i]<r.x[k]) break; key++; fr(j,0,len-1){ x[i+j]-=r.x[j]; if(x[i+j]<0){ x[i+j]+=base; x[i+j+1]--; } } }while(true); ans.x[i]=key; } memmove(x,tmp.x,sizeof(x)); ans.resize(size); return ans; } void resize(int r){ size=r; while(size&&x[size-1]==0) size--; } bool operator==(BigNumber r){ if(r.size!=size) return false; fr(i,0,size-1) if(r.x[i]!=x[i]) return false; return true; } bool operator>(BigNumber r){ if(r.size!=size) return size>r.size; for(i=size-1;i>=0;i--) if(r.x[i]!=x[i]) return x[i]>r.x[i]; return false; } double toDouble(int a,int limit){ double ans=0,value=1; int i; for(i=a;i>=0&&i>a-limit;i--,value/=base) ans+=value*x[i]; return ans; } } struct BigInteger{ int sign; BigNumber v; BigInteger(int r):sign((r>=0)*2-1),v(abs(r)){} BigInteger operator+(BigInteger r){ if(r.sign==sign) r.v=r.v+v; else if(r.v>v) r.v=r.v-v; else{ r.v=v-r.v; r.sign=sign; } return r; } BigInteger operator-(BigInteger r){ r.sign=-r.sign; return operator+(r); } BigInteger operator*(BigInteger r){ r.sign*=sign; r.v=v*r.v; return r; } BigInteger operator/(BigInteger r){ </pre> </div> </div>
--	---

```

        r.sign*=sign;
        r.v=v/r.v;
        return r;
    }

    void print(){
        if(sign<0&&v.size)
            printf("-");
        v.print();
    }
}

int main(){

Dijkstra + Heap
#include<iostream>
#define fr(i,a,b) for(i=a;i<=b;i++)
using namespace std;
const int maxn=100002;
const int maxm=1000002;
const int limit=1000000000;
template<class T>
struct DijkstraHeap{
    int
g[maxn],num[maxm*2],next[maxm*2],w[maxn],where[maxn],
tt;

    T v[maxm*2],d[maxn];
    DijkstraHeap(){
        clear();
    }

    void clear(){
        memset(g,0,sizeof(g));
        tt=0;
    }

    void add(int a,int b,T c){
        num[++tt]=b;
        v[tt]=c;
        next[tt]=g[a];
        g[a]=tt;
    }

    void add2(int a,int b,T c){
        add(a,b,c);
        add(b,c,a);
    }

    void chg(int a,int b){
        swap(w[a],w[b]);
        where[w[a]]=a;
        where[w[b]]=b;
    }

    void up(int r){
        while(r>1&&d[w[r]]<d[w[r>>1]]){
            chg(r,r>>1);
            r>>=1;
        }
    }

    void down(int r,int l){
        int j=r+r;
        while(j<=l){
            if(j<l&&d[w[j+1]]<d[w[j]])
                j++;
            if(d[w[j]]>=d[w[r]])
                break;
            chg(r,j);
            r=j;
            j=r+r;
        }
    }

    T get(int s,int t,int n){
        int i,k,l=n+1;
        fr(i,0,n)
            d[i]=limit;
        d[s]=0;
        fr(i,0,n)
            where[w[i+1]=i]=i+1;
        up(where[s]);
        while(w[1]!=t&&l){
            k=w[1];
            chg(1,l);
            down(1,--l);
            for(i=g[k];i!=next[i]){
                if(d[k]+v[i]<d[num[i]]){
                    d[num[i]]=d[k]+v[i];
                    up(where[num[i]]);
                }
            }
            return d[t];
        }
    };

    int main(){
Disjoint Set
#include<iostream>
#define fr(i,a,b) for(i=a;i<=b;i++)
using namespace std;
const int maxn=1000002;
struct DisjointSet{
    int f[maxn];
    DisjointSet(){
        clear();
    }

    void clear(){
        memset(f,255,sizeof(f));
    }

    void connect(int a,int b){
        int fa=find(a),fb=find(b);
        if(fa!=fb)
            f[fa]=fb;
    }

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

    int find(int r){
        if(f[r]<0)
            return r;
        return f[r]=find(f[r]);
    }
};

Eulerian Path
#include<iostream>
#include<vector>
#define fr(i,a,b) for(i=a;i<=b;i++)
using namespace std;
const int maxn=100002;
const int maxm=1000002;
const int mm=maxm*2+1;
struct EulerianPath1{
    int
a,b,next[maxn],num[maxn],g[maxn],tt,in[maxn],out[maxn];
    bool used[mm],u[maxn];
    vector<int> ans;
    EulerianPath1(){
        clear();
    }

    void clear(){
        memset(g,tt=0,sizeof(g));
        memset(in,0,sizeof(in));
        memset(out,0,sizeof(out));
    }

    void add(int a,int b){
        out[a]++;
        in[b]++;
        num[++tt]=b;
        next[tt]=g[a];
        g[a]=tt;
    }

    void dfs(int r){
        u[r]=true;
        for(int i=g[r];i!=next[i]){
            if(used[abs(i)])
                continue;
            used[abs(i)]=true;
            dfs(num[i]);
        }
        ans.push_back(r);
    }

    vector<int> get(int a,int b){
        int i,st=a,tot=0;
        ans.clear();
        fr(i,a,b){
            tot+=dv[i]&1;
            if(dv[i])
                st=i;
            if(tot>2)
                return ans;
        }

        memset(u,0,sizeof(u));
        memset(used,0,sizeof(used));
        dfs(st);
        fr(i,a,b)
            if(!u[i])
                ans.clear();
        return ans;
    }
};

Extended BFS
#include<iostream>
#define fr(i,a,b) for(i=a;i<=b;i++)
using namespace std;
struct State{
};
const int maxh=1<<20;
bool u[maxh];
State ht[maxh],queue[maxh/2];
int p,q,found;
int cal(State r){
}

bool same(State a,State b){
    tot+=abs(in[i]-out[i]);
    if(out[i]>in[i])
        st=i;
    if(tot>2)
        return ans;
}

    struct EulerianPath2{
        int
a,b,buf0[mm],buf1[mm],*next,*num,g[maxn],tt,dv[maxn];
        bool used[mm],u[maxn];
        vector<int> ans;
        EulerianPath2():next(buf0+maxm),num(buf1+maxm){
            clear();
        }

        void clear(){
            memset(g,tt=0,sizeof(g));
            memset(dv,0,sizeof(dv));
        }

        void add(int a,int b){
            dv[a]++;
            dv[b]++;
            num[++tt]=b;
            next[tt]=g[a];
            g[a]=tt;
            num[-tt]=a;
            next[-tt]=g[b];
            g[b]=-tt;
        }

        void dfs(int r){
            u[r]=true;
            for(int i=g[r];i!=next[i]){
                if(used[abs(i)])
                    continue;
                used[abs(i)]=true;
                dfs(num[i]);
            }
            ans.push_back(r);
        }

        vector<int> get(int a,int b){
            int i,st=a,tot=0;
            ans.clear();
            fr(i,a,b){
                tot+=dv[i]&1;
                if(dv[i])
                    st=i;
                if(tot>2)
                    return ans;
            }

            memset(u,0,sizeof(u));
            memset(used,0,sizeof(used));
            dfs(st);
            fr(i,a,b)
                if(!u[i])
                    ans.clear();
            return ans;
        }
    };
};

```

```
}
int hash(State r){
    int i=cal(r);
    while(u[i]&&!same(r,ht[i]))
        i=(i+1)&(maxh-1);
    ht[i]=r;
    return i;
}
bool isFinal(State s){
}
void add(State s){
    int w=hash(s);
    if(u[w])
        return;
    u[w]=true;
    queue[++q]=s;
    if(isFinal(s))
        found=q;
}
void extend(State s){
}
int BFS(State s){
    memset(u,found=q=0,sizeof(u));
    p=1;
    for(add(s);p<=q&&found==0;p++)
        extend(queue[p]);
    return found;
}
FFT
#include<iostream>
#include<cmath>
#define fr(i,a,b) for(i=a;i<=b;i++)
using namespace std;
const int maxn=1<<20;
template<class T>
struct Complex{
    T a,b;
    Complex(T x=0,T y=0):a(x),b(y){}
    Complex operator+(Complex r){
        return Complex(a+r.a,b+r.b);
    }
    Complex operator-(Complex r){
        return Complex(a-r.a,b-r.b);
    }
    Complex operator*(Complex r){
        return Complex(a*r.a-b*r.b,a*r.b+b*r.a);
    }
};
typedef Complex<double> Cb;
void transform(Cb *a,Cb *y,int k,int ty){
    int i,j,where,m,tmp,st;
    fr(i,0,(1<<k)-1){
        where=0;
        fr(j,0,k-1)
            where=where*2+((i>>j)&1);
        y[where]=a[i]*(ty==1?1:1.0/(1<<k));
    }
    fr(i,1,k){
        m=1<<i;
        Cb wm=Cb(cos(ty*2*acos(-1)/m),sin(ty*2*acos(-1)/m));
        for(st=0;st<(1<<k);st+=m){
            Cb w=1;
            fr(j,0,(m>>1)-1){
                Cb t=w*y[st+j+(m>>1)],u=y[st+j];
                y[st+j]=u+t;
                y[st+j+(m>>1)]=u-t;
                w=w*wm;
            }
        }
    }
}
void FFT(Cb *a,int n,Cb *b,int m,Cb *c,int &k){
    for(k=0;(1<<k)<n+m;k++);
    transform(a,c,k,1);
    transform(b,a,k,1);
    for(int i=0;i<(1<<k);i++)
        a[i]=a[i]*c[i];
    transform(a,c,k,-1);
}
```

Geometry

```
#include<iostream>
#include<vector>
#include<cmath>
#include<list>
#define fr(i,a,b) for(i=a;i<=b;i++)
#define x first
#define y second
#define point(a,b) make_pair(a,b)
using namespace std;
typedef pair<double,double> point;
typedef vector<point> polygon;
const double zero=1e-8;
const double limit=1e8;
const double pi=acos(-1);
point operator+(point a,point b){
    return make_pair(a.x+b.x,a.y+b.y);
}
point operator-(point a,point b){
    return make_pair(a.x-b.x,a.y-b.y);
}
point operator*(point a,double b){
    return make_pair(a.x*b,a.y*b);
}
point operator/(point a,double b){
    return make_pair(a.x/b,a.y/b);
}
double area2(point a,point b,point c){
    return (b.x-a.x)*(c.y-a.y)-(c.x-a.x)*(b.y-a.y);
}
double dis(point a,point b){
    return sqrt((a.x-b.x)*(a.x-b.x)+(a.y-b.y)*(a.y-b.y));
}
ostream& operator<<(ostream& os,point r){
    os<<r.x<<" "<<r.y<<" ";
    return os;
}
void print(polygon p){
    int i;
    cout<<p.size()<<" ";
    fr(i,0,(int)p.size()-1)
        cout<<[" "<<p[i].x<<" "<<p[i].y<<" "];
    cout<<endl;
}
point poi(point a,point b,point c,point d){
    double s1=area2(a,b,c),s2=area2(a,b,d);
    return make_pair((c.x*s2-d.x*s1)/(s2-s1),(c.y*s2-
d.y*s1)/(s2-s1));
}
bool onseg(point a,point b,point c){
    return dis(a,c)+dis(b,c)-dis(a,b)<zero;
}
point rotate(point a,point b,double angle){
    return make_pair((b.x-a.x)*cos(angle)-(b.y-
a.y)*sin(angle)+a.x,(b.x-a.x)*sin(angle)+(b.y-
a.y)*cos(angle)+a.y);
}
bool intersect(point &a,point &b,point &c,point &d){
    return area2(a,b,c)*area2(a,b,d)<-
zero&&area2(c,d,a)*area2(c,d,b)<-zero;
}
double newy(point a,point b,double x){
    return (x-a.x)*(b.y-a.y)/(b.x-a.x)+a.y;
}
bool inside(polygon p,point a){
    double angle=rand();
    point b=make_pair(cos(angle)*1e20,sin(angle)*1e20);
    int i,tot=0;
    fr(i,0,(int)p.size()-1)
        if(onseg(p[i],p[(i+1)%p.size()],a))
            return true;
    fr(i,0,(int)p.size()-1)
        tot+=intersect(p[i],p[(i+1)%p.size()],a,b);
    return tot&1;
}
double cal(point a,point b,point c,point d){
    if(a.x>b.x)
        return -cal(b,a,c,d);
    if(c.x>d.x)
        return -cal(a,b,d,c);
    if(min(b.x,d.x)-max(a.x,c.x)<zero)
        return 0;
    a.x<c.x?a=make_pair(c.x,newy(a,b,c.x)):c=make_pair(a.x,
newy(c,d,a.x));
    b.x>d.x?b=make_pair(d.x,newy(a,b,d.x)):d=make_pair(b.x,
newy(c,d,b.x));
    point e=area2(a,b,c)*area2(a,b,d)>-
zero?(a.y<c.y?a:c):poi(a,b,c,d);
    return ((min(a.y,c.y)+e.y)*(e.x-
a.x)+(min(b.y,d.y)+e.y)*(b.x-e.x))*0.5;
}
double intersection(polygon a,polygon b){
    int i,j;
    double s=0,tmp;
    fr(i,0,(int)a.size()-1)
        fr(j,0,(int)b.size()-1)
            s+=tmp=cal(a[i],a[(i+1)%a.size()],b[j],b[(j+1)%b.size()]);
    return s;
}
double intersection(point a,point b,polygon p){
    vector<point> list;
    int i,j;
    double sum=0;
    list.push_back(a);
    list.push_back(b);
    fr(i,0,(int)p.size()-1){
        if(onseg(a,b,p[i]))
            list.push_back(p[i]);
        if(intersect(a,b,p[i],p[(i+1)%p.size()]))
            list.push_back(poi(a,b,p[i],p[(i+1)%p.size()]));
    }
    fr(i,1,(int)list.size()-1)
        fr(j,i+1,(int)list.size()-1)
            if(dis(a,list[i])>dis(a,list[j]))
                swap(list[i],list[j]);
    fr(i,0,(int)list.size()-2)
        if(inside(p,make_pair((list[i].x+list[i+1].x)/2,(list[i].y+list[i+1].y)/2)))
            sum+=dis(list[i],list[i+1]);
    return sum;
}
double area(polygon p){
    if(p.size()==0)
        return 0;
    int i,len=p.size();
    double s=p[len-1].x*p[0].y-p[0].x*p[len-1].y;
    fr(i,0,len-2)
        s+=p[i].x*p[i+1].y-p[i+1].x*p[i].y;
    return s*0.5;
}
bool cmp(point a,point b){
    return a.x*b.y-b.x*a.y>0;
}
polygon convexHull(polygon p){
    if(p.size()<3)
        return p;
    int i,m=1,n=0;
    fr(i,1,(int)p.size()-1)
        if(p[i].x<p[0].x || fabs(p[i].x-
p[0].x)<zero&&p[i].y<p[0].y)
            swap(p[0],p[i]);
    fr(i,1,(int)p.size()-1)
        if(fabs(p[i].x-p[0].x)+fabs(p[i].y-p[0].y)>zero)
            p[++n]=make_pair(p[i].x-p[0].x,p[i].y-p[0].y);
    sort(p.begin()+1,p.begin()+n+1,cmp);
    fr(i,1,n)
        p[i]=make_pair(p[i].x+p[0].x,p[i].y+p[0].y);
    fr(i,2,n){
        if(onseg(p[m-1],p[m],p[i]))
            continue;
        while(m&&area2(p[m-1],p[m],p[i])<zero)
            m--;
        p[++m]=p[i];
    }
    p.resize(m+1);
    return p;
}
```

```
}
#define lp list<point>
vector<double> slope;
bool cmp2(int a,int b){
    return slope[a]<slope[b];
}
lp::iterator next(lp::iterator now,lp &p){
    return ++now==p.end()?p.begin():now;
}
lp::iterator back(lp::iterator now,lp &p){
    return now==p.begin()?--p.end():--now;
}
polygon halfPlaneIntersection(point *a,point *b,int n){
    lp p;
    polygon ans;
    vector<int> w;
    int i,j;
    slope.clear();
    fr(i,0,n-1){
        slope.push_back(atan2(b[i].y-a[i].y,b[i].x-a[i].x));
        w.push_back(i);
    }
    sort(w.begin(),w.end(),cmp2);
    p.push_back(make_pair(-limit,-limit));
    p.push_back(make_pair(limit,-limit));
    p.push_back(make_pair(limit,limit));
    p.push_back(make_pair(-limit,limit));
    lp::iterator now=p.begin(),i0,i1,tmp;
    fr(i,0,n-1){
        point c=a[w[i]],d=b[w[i]];
        while(area2(c,d,*now)>area2(c,d,*next(now,p)))
            now=next(now,p);
        while(area2(c,d,*now)>area2(c,d,*back(now,p)))
            now=back(now,p);
        if(area2(c,d,*now)>=zero)
            continue;
        i0=i1=now;
        while(area2(c,d,*back(i0,p))<zero){
            i0=back(i0,p);
            if(i0==now)
                return ans;
        }
        while(area2(c,d,*next(i1,p))<zero)
            i1=next(i1,p);
        p.insert(i0,poi(*i0,*back(i0,p),c,d));
        *i1=poi(*i1,*next(i1,p),c,d);
        for(now=i0;now!=i1;now=tmp){
            tmp=next(now,p);
            p.erase(now);
        }
    }
    for(now=p.begin();now!=p.end();now++)
        ans.push_back(*now);
    return ans;
}
void midperpendicular(point a,point b,point &c,point &d){
    c=(a+b)/2;
    d=point(-(b.y-c.y),b.x-c.x)+c;
}
point circleCenter(point a,point b,point c){
    point ab0,ab1,ac0,ac1;
    midperpendicular(a,b,ab0,ab1);
    midperpendicular(a,c,ac0,ac1);
    return poi(ab0,ab1,ac0,ac1);
}
int main(){
Hash
#include<iostream>
#define fr(i,a,b) for(i=a;i<=b;i++)
#define HASH int
using namespace std;
const int maxh=1<<20;
bool u[maxh];
int ht[maxh];
int cal(HASH r){
    return r&(maxh-1);
}
bool same(HASH a,HASH b){
    return a==b;
}
int hash(HASH r){
    int i=cal(r);
    while(u[i]&&!same(r,ht[i]))
        i=(i+1)&(maxh-1);
    ht[i]=r;
    return i;
}
Linear Programming
/*说明： 本来变量都应放在 class 里面的，但是由于在里面
开大内存会 RE，所以暂时先放外面。N[0]代表 N 中的元素
个数，B[0]代表 B 中的元素个数。 读入格式（在文件名为
inputName 的文件中读入）： 首先两个数 n,m，表示未知
数的数量和约束的数量。 接下来一行 n 个数，为目标函
数的系数。 然后 m 行，每行 m+1 个数，表示一个约
束。前 m 个数是系数，最后一个为常数项。 输出格式
（在文件名为 outputName 的文件中输出）： 如果无解，
只有一行"Infeasible"。 如果解可以无穷大，只有一行
"Unbounded"。否则，第一行为最大的目标函数值，接下
来是每个未知数的值。*/
#include <string>
#include <cmath>
#include <iostream>
using namespace std;
const double eps=1e-10;
const int maxn=2002;
const int oo=19890709;
struct LinearProgramming{
    double
    A[maxn][maxn],tA[maxn][maxn],b[maxn],tb[maxn],c[maxn],tc
    [maxn],v;
    int N[maxn],B[maxn],n,m;
    void set(){
        for(int i=1; i<=n; i++){
            scanf("%lf", &c[i]);
            c[i]*=m;
        }
        for(int i=1; i<=m; i++){
            for(int j=1; j<=n; j++)
                scanf("%lf", &A[n+i][j]);
            scanf("%lf", &b[n+i]);
        }
    }
    void pivot(int l, int e){
        tb[e] = b[l]/A[l][e];
        tA[e][l] = 1/A[l][e];
        for(int i=1; i<=N[0]; i++)
            if (N[i] != e)
                tA[e][N[i]] = A[l][N[i]]/A[l][e];
        for(int i=1; i<=B[0]; i++){
            tb[B[i]] = b[B[i]]-A[B[i]][e]*tb[e];
            tA[B[i]][l] = -A[B[i]][e]*tA[e][l];
            for(int j=1; j<=N[0]; j++)
                if (N[j] != e)
                    tA[B[i]][N[j]] = A[B[i]][N[j]]-
tA[e][N[j]]*A[B[i]][e];
        }
        v += tb[e]*c[e];
        tc[l] = -tA[e][l]*c[e];
        for(int i=1; i<=N[0]; i++)
            if (N[i] != e)
                tc[N[i]] = c[N[i]]-tA[e][N[i]]*c[e];
        for(int i=1; i<=N[0]; i++)
            if (N[i] == e) N[i] = l;
        for(int i=1; i<=B[0]; i++)
            if (B[i] == l) B[i] = e;
        for(int i=1; i<=B[0]; i++){
            for(int j=1; j<=N[0]; j++)
                A[B[i]][N[j]] = tA[B[i]][N[j]];
            b[B[i]] = tb[B[i]];
        }
        for(int i=1; i<=N[0]; i++)
            c[N[i]] = tc[N[i]];
    }
    bool opt()//false stands for unbounded
        while (true){
            int l, e;
            double maxUp = -1;//不能是 0！
for(int ie=1; i<=N[0]; ie++){
            int te = N[ie];
            if (c[te] <= eps) continue;//eps or 0
            double delta = oo;
            int tl = maxn;
            for(int i=1; i<=B[0]; i++)
                if (A[B[i]][te] > eps){//eps or 0
                    double temp = b[B[i]]/A[B[i]][te];
                    if (delta == oo || temp < delta ||
                        delta = temp;
                        tl = B[i];
                    }
                }
            if (tl == maxn) return false;
            if (delta*c[te] > maxUp){
                maxUp = delta*c[te];
                l = tl;
                e = te;
            }
        }
        if (maxUp == -1) break;
        pivot(l, e);
    }
    return true;
}
void delete0(){
    int p;
    for(p=1; p<=B[0]; p++)
        if (B[p] == 0) break;
    if (p <= B[0]) pivot(0, N[1]);
    for(p=1; p<=N[0]; p++)
        if (N[p] == 0) break;
    for(int i=p; i<N[0]; i++)
        N[i] = N[i+1];
    N[0]-=;
}
bool initialize(){
    N[0] = B[0] = 0;
    for(int i=1; i<=n; i++)
        N[++N[0]] = i;
    for(int i=1; i<=m; i++)
        B[++B[0]] = n+i;
    v = 0;
    int l = B[1];
    for(int i=2; i<=B[0]; i++)
        if (b[B[i]] < b[l])
            l = B[i];
    if (b[l] >= 0) return true;
    double origC[maxn];
    memcpy(origC, c, sizeof(double)*(n+m+1));
    N[++N[0]] = 0;
    for(int i=1; i<=B[0]; i++)
        A[B[i]][0] = -1;
    memset(c, 0, sizeof(double)*(n+m+1));
    c[0] = -1;
    pivot(l, 0);
    opt();//unbounded
    if (v < -eps) return false;//eps
    delete0();
    memcpy(c, origC, sizeof(double)*(n+m+1));
    bool inB[maxn];
    memset(inB, false, sizeof(bool)*(n+m+1));
    for(int i=1; i<=B[0]; i++)
        inB[B[i]] = true;
    for(int i=1; i<=n+m; i++)
        if (inB[i] && c[i] != 0){
            v += c[i]*b[i];
            for(int j=1; j<=N[0]; j++)
                c[N[j]] -= A[i][N[j]]*c[i];
            c[i] = 0;
        }
    return true;
}
double get(){
    set();
    if (!initialize()){
        printf("Infeasible\n");
        return 0;
    }
}
```

```

        if (!opt()){
            printf("Unbounded\n");
            return 0;
        }
        return v;
        bool inN[maxn];
        memset(inN, false, sizeof(bool)*(n+m+1));
        for(int i=1; i<=N[0]; i++)
            inN[N[i]] = true;
        for(int i=1; i<=n; i++)
            if (inN[i]) printf("x%d = %lf\n", i, 0.0);
            else printf("x%d = %lf\n", i, b[i]);
    }
};
LinearProgramming g;
int main(){
    while(cin>>g.n>>g.m)
        printf("Nasa can spend %d taka.\n", (int)ceil(g.get()));
}

```

Max Flow

```

#include<iostream>
using namespace std;
const int maxn=100002;
const int maxm=1000002;
const int mm=maxm*2+1;
const int limit=1000000000;
const double zero=1e-8;
template<class T>
struct MaxFlow{
    int buf3[mm],buf4[mm],g[maxn],la[maxn],*num,*next,tt;
    T buf1[mm],buf2[mm],*x,*y,rec,sum,u[maxn];
    MaxFlow():x(buf1+maxm),y(buf2+maxm),num(buf3+max
m),next(buf4+maxm){
        clear();
    }
    void clear(){
        memset(g,tt=0,sizeof(g));
    }
    void add(int a,int b,T c){
        next[++tt]=g[a];
        next[-tt]=g[b];
        num[g[a]=tt]=b;
        num[g[b]=-tt]=a;
        x[tt]=c;
        x[-tt]=0;//x[-tt]=-c if undirected
    }
    bool dfs(int r,int t,T mi){
        if(r==t){
            rec=mi;
            return true;
        }
        u[r]=sum;
        int i=la[r];
        do{
            if(x[i]-
y[i]>zero&&u[num[i]]!=sum&&dfs(num[i],t,min(mi,x[i]-y[i]))){
                y[-i]=(y[la[r]=i]+=rec);
                return true;
            }
            i=(next[i]?next[i]:g[r]);
        }while(i!=a[r]);
        return false;
    }
    T get(int s,int t){
        memset(u,255,sizeof(u));
        memmove(la,g,sizeof(g));
        memset(y-tt,0,(tt*2+1)*sizeof(T));
        sum=0;
        while(dfs(s,t,limit))
            sum+=rec;
        return sum;
    }
};
MaxFlow<int> g;
int main(){
}

```

Max Flow Max Cost

```

#include<iostream>
#define fr(i,a,b) for(i=a;i<=b;i++)

```

```

#define nxt(r) (r==n?0:r+1)
using namespace std;
const int maxn=100002;
const int maxm=1000002;
const int mm=maxm*2+1;
const int limit=1000000000;
const double zero=1e-8;
template<class T1,class T2>
struct MaxFlowMaxCost{
    int
    buf3[mm],buf4[mm],g[maxn],*num,*next,tt,queue[maxn],f[m
axn],i,j,k,p,q;
    T1 buf1[mm],buf2[mm],*x,*y;
    T2 buf5[mm],*v,d[maxn];
    bool u[maxn];
    MaxFlowMaxCost():x(buf1+maxm),y(buf2+maxm),num(b
uf3+maxm),next(buf4+maxm),v(buf5+maxm){
        clear();
    }
    void clear(){
        memset(g,tt=0,sizeof(g));
    }
    void add(int a,int b,T1 c,T2 d){
        next[++tt]=g[a];
        next[-tt]=g[b];
        num[g[a]=tt]=b;
        num[g[b]=-tt]=a;
        x[tt]=c;
        v[-tt]=-(v[tt]=d);
        x[-tt]=0;
    }
    void get(int s,int t,int n,T1 &maxflow,T2 &maxcost){
        maxflow=maxcost=0;
        memset(y-tt,0,(tt*2+1)*sizeof(T1));
        do{
            fr(i,0,n)
                d[i]=-limit;
            memset(u,d[s]=0,n+1);

            for(u[queue[p=q=0]=s]=true;nxt(q)!=p;p=nxt(p)){
                u[j=queue[p]]=false;
                for(i=g[j];i!=next[i])
                    if(x[i]-
y[i]>zero&&d[k=num[i]]<d[j]+v[i]){
                        d[k]=d[j]+v[f[k]=i];
                        if(!u[k])
                            u[queue[q=nxt(q)]=k]=true;
                    }
            }
            if(d[t]>-limit){
                T1 mi=limit;
                for(i=t;i!=s;i=num[-f[i]])
                    mi=min(mi,x[f[i]]-y[f[i]]);
                for(i=t;i!=s;i=num[-f[i]])
                    y[-f[i]]=(y[f[i]]+=mi);
                maxcost+=mi*d[t];
                maxflow+=mi;
            }
        }while(d[t]>-limit);
    }
};
int main(){
}

```

Max Flow Min Cost

```

#include<iostream>
#define fr(i,a,b) for(i=a;i<=b;i++)
#define nxt(r) (r==n?0:r+1)
using namespace std;
const int maxn=100002;
const int maxm=1000002;
const int mm=maxm*2+1;
const int limit=1000000000;
const double zero=1e-8;
template<class T1,class T2>
struct MaxFlowMinCost{
    int
    buf3[mm],buf4[mm],g[maxn],*num,*next,tt,queue[maxn],f[m
axn],i,j,k,p,q;
    T1 buf1[mm],buf2[mm],*x,*y;
    T2 buf5[mm],*v,d[maxn];

```

```

bool u[maxn];
    MaxFlowMinCost():x(buf1+maxm),y(buf2+maxm),num(b
uf3+maxm),next(buf4+maxm),v(buf5+maxm){
        clear();
    }
    void clear(){
        memset(g,tt=0,sizeof(g));
    }
    void add(int a,int b,T1 c,T2 d){
        next[++tt]=g[a];
        next[-tt]=g[b];
        num[g[a]=tt]=b;
        num[g[b]=-tt]=a;
        v[-tt]=-(v[tt]=d);
        x[tt]=c;
        x[-tt]=0;
    }
    void get(int s,int t,int n,T1 &maxflow,T2 &mincost){
        maxflow=mincost=0;
        memset(y-tt,0,(tt*2+1)*sizeof(T1));
        do{
            fr(i,0,n)
                d[i]=limit;
            memset(u,d[s]=0,n+1);

            for(u[queue[p=q=0]=s]=true;nxt(q)!=p;p=nxt(p)){
                u[j=queue[p]]=false;
                for(i=g[j];i!=next[i])
                    if(x[i]-
y[i]>zero&&d[k=num[i]]>d[j]+v[i]){
                        d[k]=d[j]+v[f[k]=i];
                        if(!u[k])
                            u[queue[q=nxt(q)]=k]=true;
                    }
            }
            if(d[t]<limit){
                T1 mi=limit;
                for(i=t;i!=s;i=num[-f[i]])
                    mi=min(mi,x[f[i]]-y[f[i]]);
                for(i=t;i!=s;i=num[-f[i]])
                    y[-f[i]]=(y[f[i]]+=mi);
                mincost+=mi*d[t];
                maxflow+=mi;
            }
        }while(d[t]<limit);
    }
};
int main(){
}

```

Number Theory

```

#include<iostream>
template<class T>
T extGCD(T a,T b,T &x,T &y){
    if(b==0){
        x=1;
        y=0;
        return a;
    }
    T tmp=extGCD(b,a%b,y,x);
    y=-a/b*x;
    return tmp;
}
template<class T>
T extMod(T a,T b){
    T tmp=a%b;
    return tmp<0?tmp+b:tmp;
}
int main(){
}

```

Prime Heap

```

#include<iostream>
#define fr(i,a,b) for(i=a;i<=b;i++)
using namespace std;
const int maxn=100002;
const int maxm=1000002;
const int limit=1000000000;
template<class T>
struct PrimeHeap{
    int
    g[maxn],num[maxn],next[maxn],w[maxn],where[maxn],tt;

```

```
T v[maxn],d[maxn];
PrimeHeap(){
    clear();
}

void clear(){
    memset(g,tt=0,sizeof(g));
}

void add(int a,int b,T c){
    num[+tt]=b;
    v[tt]=c;
    next[tt]=g[a];
    g[a]=tt;
}

void add2(int a,int b,T c){
    add(a,b,c);
    add(b,a,c);
}

void chg(int a,int b){
    swap(w[a],w[b]);
    where[w[a]]=a;
    where[w[b]]=b;
}

void up(int r){
    while(r>1&&d[w[r]]<d[w[r>>1]]){
        chg(r,r>>1);
        r>>=1;
    }
}

void down(int r,int l){
    int j=r+r;
    while(j<=l){
        if(j<l&&d[w[j+1]]<d[w[j]])
            j++;
        if(d[w[j]]>=d[w[r]])
            break;
        chg(r,j);
        r=j;
        j=r+r;
    }
}

T get(int s,int n){
    int i,k,l=n+1;
    T ans=0;
    fr(i,0,n)
        d[i]=limit;
    d[s]=0;
    fr(i,0,n)
        where[w[i+1]]=i+1;
    for(i=i;i--){
        up(i);
        while(d[w[1]]<limit&&l){
            ans+=d[k=w[1]];
            chg(1,l);
            down(1,--l);
            for(i=g[k];i=next[i])
                if(v[i]<d[num[i]]&&where[num[i]]<=l){
                    d[num[i]]=v[i];
                    up(where[num[i]]);
                }
        }
        return ans;
    }
};

int main(){}
```

Size Balanced Tree

```
#include<iostream>
#define sz(t) (t==NULL?0:t->s)
using namespace std;
struct Node{
    int key,s;
    Node *left,*right;
    Node(int a:s(1),left(NULL),right(NULL),key(a)){
};

int cmp(Node *a,Node *b){
    return a->key-b->key;
}

void update(Node *t){
    if(!t)
        return;
    t->s=sz(t->left)+sz(t->right)+1;
}
```

```
void rr(Node *&t,Node *k){
    t->left=k->right;
    k->right=t;
    update(t);
    update(k);
    t=k;
}

void lr(Node *&t,Node *k){
    t->right=k->left;
    k->left=t;
    update(t);
    update(k);
    t=k;
}

void insert(Node *&t,Node *k){
    if(t==NULL)
        t=k;
    else
        if(cmp(k,t)<0){
            insert(t->left,k);
            if(sz(t->left->left)>sz(t->right))
                rr(t,t->left);
        }
        else{
            insert(t->right,k);
            if(sz(t->right->right)>sz(t->left))
                lr(t,t->right);
        }
    update(t);
}

Node findmin(Node *t){
    while(t->left!=NULL)
        t=t->left;
    return *t;
}

void remove(Node *&t,Node *k){
    if(t==NULL)
        return;
    if(cmp(k,t)==0)
        if(t->right==NULL){
            k=t->left;
            delete t;
            t=k;
        }
        else{
            Node tmp=findmin(t->right);
            remove(t->right,&tmp);
            tmp.left=t->left;
            tmp.right=t->right;
            *t=tmp;
        }
    else
        if(cmp(k,t)<0)
            remove(t->left,k);
        else
            remove(t->right,k);
    update(t);
}

void preorder(Node *t){
    if(t==NULL)
        return;
    preorder(t->left);
    printf("%d ",t->key);
    preorder(t->right);
}

int main(){}
```

Sort

```
#include<iostream>
#define fr(i,a,b) for(i=a;i<=b;i++)
using namespace std;
void qsort(int a,int b,int x[]){
    if(a>=b)
        return;
    int i=a,j=b,k=x[(a+b)>>1];
    do{
        while(x[j]>k)
            j--;
        while(x[i]<k)
            i++;
        swap(x[i++],x[j--]);
    }while(i<=j);
    qsort(a,j,x);
    qsort(i,b,x);
}

long long mergeSort(int a,int b,int list[]){
    if(a>=b)
        return 0;
    int m=(a+b)>>1,i=a,j=m+1,k,*tmp=new int[b-a+1]+a;
    long long tot=mergeSort(a,m,list)+mergeSort(m+1,b,list);
    fr(k,a,b)
        if(j>b || i<=m&&list[i]<=list[j]){
            tmp[k]=list[i++];
            tot+=j-m-1;
        }
        else
            tmp[k]=list[j++];
    memmove(list+a,tmp+a,(b-a+1)*sizeof(list[0]));
    delete tmp;
    return tot;
}

int main(){}
```

Suffix Sort

```
#include<iostream>
#define fr(i,a,b) for(i=a;i<=b;i++)
using namespace std;
const int maxn=200002;
int
sa[maxn],rank[maxn],nrank[maxn],nsa[maxn],cnt[maxn],x[maxn],h[maxn];
void suffixSort(int *x,int n,int limit){
    int i,k;
    memset(cnt,0,sizeof(cnt));
    fr(i,0,n-1)
        cnt[x[i]]++;
    fr(i,1,limit)
        cnt[i]+=cnt[i-1];
    for(i=n-1;i>=0;i--)
        sa[--cnt[x[i]]]=i;
    rank[sa[0]]=0;
    fr(i,1,n-1)
        rank[sa[i]]=rank[sa[i-1]]+(x[sa[i-1]]!=x[sa[i]]);
    for(k=1;k<n;k*=2){
        fr(i,0,n-1)
            cnt[rank[sa[i]]]=i+1;
        for(i=n-1;i>=0;i--)
            if(sa[i]-k>=0)
                nsa[--cnt[rank[sa[i]-k]]]=sa[i]-k;
        fr(i,n-k,n-1)
            nsa[--cnt[rank[i]]]=i;
        nrank[sa[0]]=0;
        rank[n]=-1;
        fr(i,1,n-1)
            nrank[nsa[i]]=nrank[nsa[i-1]]+(rank[nsa[i]]!=rank[nsa[i-1]]+k);
        memmove(rank,nrank,n*4);
        memmove(sa,nsa,n*4);
    }
}

void setHeight(int n){
    int i,j,k=0;
    fr(i,0,n-1)
        if(rank[i]){
            k=k?k-1:0;
            j=sa[rank[i]-1];
            while(i+k<n&&j+k<n&&x[j+k]==x[i+k])
                k++;
            h[rank[i]]=k;
        }
        else
            h[0]=k=0;
}

int main(){}
```

Two SAT

```
#include<iostream>
```

```
#include <stdio>
#include <vector>
#include <algorithm>
using namespace std;
const int N = 2100;//最大人数
class SAT{
private:
    vector<int> g[N]; //原图边连接情况
    int n, m, h[N], id[N]; //id[]表示原图中每个点都属于哪个强连通分量
    int cnt, scnt, dfn[N], low[N], cur[N];
    int stack[N], top, est[N], etop;
    vector<int> tree[N]; //有向无环图的边连接情况(新图)
    vector<int> contain[N]; //新图中每个点都包含原图中的哪些点
    void dfs(int);
    void tsDfs(int);
    void topologicalSort();
    void colorDfs(int);
    void color();
    void tagAnswer();
    void printAnswer();
    void getOneAnswer();
    void buildGraph();
public:
    void scR();
    bool build();
    bool judge();
    void solve();
};

/*函数 build 是对原图初始化(根据实际输入情况做相应的更改)*/
bool SAT::build(){
    if (scanf("%d %d",&n,&m)==EOF)
        return false;
    n*=2;
    for (int i=0;i<n;i++)
        g[i].clear();
    for (int i=0;i<m;i++) {
        int a,b,c,d;
        scanf("%d %d",&a,&b);//a, b 是互斥的。a 所在组的另一点为 c, b 所在组的另一点为 d。原图中插入边 a->d 和边 b->c。
        if(a>0){
            a=(a-1)*2;
            c=a+1;
        }
        else{
            c=(-a-1)*2;
            a=c+1;
        }
        if(b>0){
            b=(b-1)*2;
            d=b+1;
        }
        else{
            d=(-b-1)*2;
            b=d+1;
        }
        g[c].push_back(b);
        g[d].push_back(a);
    }
    return true;
}

void SAT::dfs(int src){
    etop = top = 0;
    stack[top++] = src;
    while(top != 0){
        int c = stack[top-1];
        if(dfn[c] == -1){
            h[c] = dfn[c] = low[c] = cnt++;
            est[etop++] = c;
        }
        for(; cur[c] >= 0; cur[c]--){
            int no = g[c][cur[c]];
            if(dfn[no] == -1){
                stack[top++] = no;
                break;
            }
        }
    }
}

h[c] <?= low[no];
}
if(cur[c] >= 0)
    continue;
top--;
int k;
if(h[c] != low[c]){
    low[c] = h[c];
    continue;
}
do{
    k = est[--etop];
    id[k] = scnt; low[k] = N;
}while(k != c);
scnt++;
}
}

/*函数 scR 和 dfs 是求原图的强连通分量*/
void SAT::scR(){
    memset(dfn, -1, sizeof(dfn));
    cnt = scnt = 0;
    for(int i = 0; i < n; i++){
        cur[i] = g[i].size()-1;
        contain[i].clear();
    }
    for(int i = 0; i < n; i++)
        if(dfn[i] == -1)
            dfs(i);

    /*统计每个强连通分量都包含哪些点，为后面求可行方案做准备。如果不求可行解，可注释掉。*/
    for (int i=0;i<n;i++)
        contain[id[i]].push_back(i);
}

/*函数 judge 判断是否能找出一个可行方案出来*/
bool SAT::judge(){
    for (int i=0;i<n;i+=2)
        if (id[i]==id[i+1])
            return false;
    return true;
}

/*函数 buildGraph 把每个强连通分量作为一个点，重新构图。(缩点，得到的是一个有向无环图)
用的是链接表的形式，可能有很多重边。可以加一些预处理消除重边。s*/
void SAT::buildGraph(){
    for (int i=0;i<scnt;i++)
        tree[i].clear();
    for (int i=0;i<n;i++)
        for (int j=0;j<g[i].size();j++){
            int a=id[i];
            int b=id[g[i][j]];
            if (a!=b)
                tree[b].push_back(a);
        }
}

void SAT::tsDfs(int k){
    dfn[k]=cnt++;
    for (int i=0;i<tree[k].size();i++){
        int w=tree[k][i];
        if (dfn[w]==-1)
            tsDfs(w);
    }
    low[scnt++]=k;
}

/*函数 topologicalSort 和 tsDfs 是对新图进行拓扑排序，排序后的结果存在 low 数组中 */
void SAT::topologicalSort(){
    for (int i=0;i<scnt;i++){
        dfn[i]=-1;
        low[i]=-1;
    }
    int nn=scnt;
    cnt=scnt=0;
    for (int i=0;i<nn;i++)
        if (dfn[i]==-1)
            tsDfs(i);
}

void SAT::colorDfs(int k){
    dfn[k]=2;
    for (int i=0;i<tree[k].size();i++){
        int w=tree[k][i];
        if (dfn[w]==-1)
            colorDfs(w);
    }
}

/*函数 color 和 colorDfs 是对新图进行着色，新图中着色为 1 的点组成一组可行解*/
void SAT::color(){
    for (int i=0;i<scnt;i++)
        dfn[i]=-1;
    for (int i=scnt-1;i>=0;i--){
        if (dfn[low[i]]==-1){
            /*新图中 low[i]着色为 1 后，它的矛盾点应标记为 2*/
            int a=contain[low[i]][0];//在原图中找一点属于强连通分量 low[i]的点 a，点 a 所属组的另一点 b 所属的强连通分量 id[b]一定是 low[i]矛盾点。
            int b;
            if (a%2==0)
                b=a+1;
            else
                b=a-1;
            dfn[low[i]]=1;
            if (dfn[id[b]]==-1)
                colorDfs(id[b]); //由于依赖关系，有 id[b]能达的点都是 low[i]的矛盾点
        }
    }

    /*函数 tagAnswer 由新图对原图的点进行标记，得到原图的可行解*/
    void SAT::tagAnswer(){
        for (int i=0;i<n;i++)
            low[i]=-1;
        for (int i=0;i<scnt;i++)
            if (dfn[i]==1)//i 为新图中可行解包含的点，那么原图中强连通分量属于 i 的点都是原图中可行解的点
                for (int j=0;j<contain[i].size();j++)
                    low[contain[i][j]]=1;
    }

    /*函数 printAnswer 打印原图的可行解*/
    void SAT::printAnswer(){
        for (int i=0,j=0;i<n;i++,j++){
            if (low[i]==1){
                printf("%d",i);
                if (j!=n/2-1)
                    printf(" ");
                else
                    printf("\n");
            }
        }

        /*函数 getOneAnswer 得到原图的一组可行解*/
        void SAT::getOneAnswer(){
            buildGraph();
            topologicalSort();
            color();
            tagAnswer();
            printAnswer();
        }

        /*函数 solve 可根据实际要求，进行更改输出*/
        void SAT::solve(){
            scR();
            if (judge()){
                printf("1\n");
                getOneAnswer();
            }
            else
                printf("0\n");
        }
    }

    int main(){
        SAT sat;
        while (sat.build())
            sat.solve();
    }
}
```