计算几何

1.Andrew凸包求周长

```
struct P{
    double x,y;
    bool operator<(const P&p)const{
        return y==p.y?x<p.x:y<p.y;
    }
}a[N],s[N];
int top,n;
double cross(P a,P b,P c){
    return (b.x-a.x)*(c.y-a.y)-(c.x-a.x)*(b.y-a.y);
}
double dis(P a,P b){
    return (a.x-b.x)*(a.x-b.x)+(a.y-b.y)*(a.y-b.y);
}
double andrew(){
    sort(a+1,a+n+1);
    s[top=1]=a[1];
    for(int i=2;i<=n;){
        if(top>1&&cross(s[top-1],a[i],s[top])>=0) --top;
        else s[++top]=a[i++];
    }
    int tmp=top;
    for(int i=n-1;i;){
        if(top>tmp&&cross(s[top-1],a[i],s[top])>=0) --top;
        else s[++top]=a[i--];
    }
    if(n>1) top--;//因为多了一个1号结点 要去掉
    double ans=0;for(int i=2;i<=top;i++) ans+=sqrt(dis(s[i],s[i-1]));
    if(top>1) ans+=sqrt(dis(s[1],s[top]));return ans;
}
int main(){
    scanf("%d",&n);
    for(int i=1;i<=n;i++) scanf("%lf%lf",&a[i].x,&a[i].y);
    sort(a+1,a+n+1);
    printf("%.2f\n",andrew());
    return 0;
}
```

2.graham求凸包周长

```
struct P{
    double x,y;
}a[N],s[N];
int top,n;
double cross(P a,P b,P c){
    return (b.x-a.x)*(c.y-a.y)-(c.x-a.x)*(b.y-a.y);
}
double dis(P a,P b){
    return (a.x-b.x)*(a.x-b.x)+(a.y-b.y)*(a.y-b.y);
}
bool cmp(P u,P v){
    double t=cross(u,v,a[1]);
    return t>0||(t==0&&dis(u,a[1])<dis(v,a[1]));
}
double graham(){
    s[top=1]=a[1];
    for(int i=2;i<=n;){
        if(top>1&&cross(s[top-1],a[i],s[top])>=0) top--;
        else s[++top]=a[i++];
    }
    double ans=0;
    for(int i=2;i<=top;i++) ans+=sqrt(dis(s[i],s[i-1]));
    if(top>1) ans+=sqrt(dis(s[top],s[1]));return ans;
}
int main(){
    scanf("%d",&n);
    for(int i=1;i<=n;i++) scanf("%lf%lf",&a[i].x,&a[i].y);
    int id=1;
    for(int i=2;i<=n;i++)
        if(a[i].x<a[id].x||(a[i].x==a[id].x&&a[i].y<a[id].y)) id=i;
    swap(a[1],a[id]);sort(a+2,a+n+1,cmp);
    printf("%.0f\n",graham());
    return 0;
}
```

数据结构

1.树状数组BIT

```
struct BIT{
    #define lowbit(x) x&(-x)
    #define il inline
    ll s[N];
    int n;
    il void upd(int x,int v){
        while(x<=n){
            s[x]+=v;x+=lowbit(x);
        }return;
    }
    il ll que(int x){
        ll ans=0;
        while(x){
            ans+=s[x];x-=lowbit(x);
        }return ans;
    }
}T;
```

2.ST表

```
#define il inline
struct RMQ{
    int f[N][20],b[N];//pay attention init f[i][0]
    il void init(int n){
        for(int i=2;i<=n;i++)    //pre deal with
            b[i]=b[i>>1]+1;
        for(int j=1;j<=b[n];j++)
            for(int i=1;i+(1<<j)-1<=n;i++)
                f[i][j]=max(f[i][j-1],f[i+(1<<j-1)][j-1]);
    }
    il int que(int l,int r){
        int x=b[r-l+1];
        return max(f[l][x],f[r-(1<<x)+1][x]);
    }
}rm;
```

3.Treap(size)

```
#define il inline
int R;
struct Treap{ //fhq treap (split by rank wtihout rotation)
    int w[N],rd[N],sz[N],son[N][2],cnt;
    bool lz[N];
    il void re(int x){sz[x]=sz[son[x][0]]+sz[son[x][1]]+1;}
    il int newnode(int x){w[++cnt]=x,sz[cnt]=1,rd[cnt]=rand();return cnt;}
    il void down(int x){//do something
        if(lz[x]){
            swap(son[x][0],son[x][1]);
            lz[son[x][0]]^=1;
            lz[son[x][1]]^=1;
            lz[x]=0;
        }}
    int merge(int x,int y){if(!x||!y) return x+y;down(x),down(y);
        if(rd[x]<rd[y])
            {son[x][1]=merge(son[x][1],y);re(x);return x;}
        son[y][0]=merge(x,son[y][0]);re(y);return y;
    }
    void split(int i,int k,int &x,int &y){if(!i) {x=y=0;return;}down(i);
        if(sz[son[i][0]]<k) x=i,split(son[i][1],k-sz[son[i][0]]-1,son[i][1],y);
        else y=i,split(son[i][0],k,x,son[i][0]);
        re(i);
    }
    void fun(int x){ //inorder traversal
        if(!x) return;down(x);
        fun(son[x][0]);printf("%d ",w[x]);fun(son[x][1]);
    }
}T;
```

4.Treap(value)

```
#define il inline
int R;
struct Treap{ //fhq treap (split by rank wtihout rotation)
    int w[N],rd[N],sz[N],son[N][2],cnt;
    bool lz[N];
    il void re(int x){sz[x]=sz[son[x][0]]+sz[son[x][1]]+1;}
    il int newnode(int x){w[++cnt]=x,sz[cnt]=1,rd[cnt]=rand();return cnt;}
    int merge(int x,int y){if(!x||!y) return x+y;
        if(rd[x]<rd[y])
            {son[x][1]=merge(son[x][1],y);re(x);return x;}
        son[y][0]=merge(x,son[y][0]);re(y);return y;
    }
    void split(int i,int k,int &x,int &y){if(!i) {x=y=0;return;}
        if(w[i]<=k) x=i,split(son[i][1],k,son[i][1],y);
        else y=i,split(son[i][0],k,x,son[i][0]);
        re(i);}
    void ins(int a){
        int x,y;
        split(R,a,x,y);
        R=merge(merge(x,newnode(a)),y);
    }
    void del(int a){
        int x,y,z;
        split(R,a,x,z);
        split(x,a-1,x,y);
        y=merge(son[y][0],son[y][1]);
        R=merge(merge(x,y),z);
    }
    int kth(int x,int k){
        while(1){
            if(k<=sz[son[x][0]]) x=son[x][0];
            else if(k==sz[son[x][0]]+1) return w[x];
            else k-=sz[son[x][0]]+1,x=son[x][1];
        }
    }
    int pre(int a){
        int x,y;
        split(R,a-1,x,y);
        int ans=kth(x,sz[x]);
        R=merge(x,y);
        return ans;
    }
    int suf(int a){
        int x,y;
        split(R,a,x,y);
        int ans=kth(y,1);
        R=merge(x,y);
        return ans;
    }
}T;
```

5.Treap(有旋)

```
//fhq treap (having rotation [split by value] )
int sum=0,R=0;//sum (sum of node) R (root index)
int size[N],v[N],num[N],rd[N],son[N][2];
inline void pushup(int p){
    size[p]=size[son[p][0]]+size[son[p][1]]+num[p];
}
inline void rotate(int &p,int d){   //d=0 (left rotate)
    int k=son[p][d^1];
    son[p][d^1]=son[k][d];
    son[k][d]=p;
    pushup(p),pushup(k),p=k;
}
void ins(int &p,int x){
    if (!p){
        p=++sum,size[p]=num[p]=1;
        v[p]=x,rd[p]=rand();
        return;
    }
    if (v[p]==x){
        num[p]++,size[p]++;return;
    }
    int d=(x>v[p]);
    ins(son[p][d],x);
    if (rd[p]<rd[son[p][d]]) rotate(p,d^1);//Max Heap
    pushup(p);
}
void del(int &p,int x){
    if (!p) return;
    if(x!=v[p]) del(son[p][x>v[p]],x);
    else{
        if (!son[p][0] && !son[p][1]){
            num[p]--,size[p]--;
            if (!num[p]) p=0;
        }
        else if (son[p][0] && !son[p][1]){
            rotate(p,1);
            del(son[p][1],x);
        }
        else if (!son[p][0] && son[p][1]){
            rotate(p,0);
            del(son[p][0],x);
        }
        else if (son[p][0] && son[p][1]){
            int d=(rd[son[p][0]]>rd[son[p][1]]);
            rotate(p,d);
            del(son[p][d],x);
        }
    }
    pushup(p);
}
int rk(int p,int x){
    if (!p) return 0;
    if (v[p]==x) return size[son[p][0]]+1;
    if (v[p]<x) return size[son[p][0]]+num[p]+rk(son[p][1],x);
    if (v[p]>x) return rk(son[p][0],x);
}
int find(int p,int x){
    if (!p) return 0;
    if (size[son[p][0]]>=x) return find(son[p][0],x);
    else if (size[son[p][0]]+num[p]<x)
        return find(son[p][1],x-num[p]-size[son[p][0]]);
    else return v[p];
}
int pre(int p,int x){
    if (!p) return -inf;
    if (v[p]>=x) return pre(son[p][0],x);
    else return max(v[p],pre(son[p][1],x));
}
int suc(int p,int x){
    if (!p) return inf;
    if (v[p]<=x) return suc(son[p][1],x);
    else return min(v[p],suc(son[p][0],x));
}
```

6.Trie

```
#define il inline
const int li=20;
struct Trie{
    int son[N*li][2],sz[N*li],cnt;
    il void ins(int x){
        int p=0;
        for(int i=li;~i;i--){
            int &s=son[p][x>>i&1];
            if(!s) s=++cnt;
            p=s;
            sz[p]++;
        }
    }
    il void del(int x){
        int p=0;
        for(int i=li;~i;i--){
            int &s=son[p][x>>i&1];
            p=s;
            if(!--sz[p]) s=0;
        }
    }
    il int que(int x){
        int p=0,ans=0;
        for(int i=li;~i;i--){
            int s=x>>i&1;
            if(son[p][!s]) ans+=1<<i,p=son[p][!s];
            else p=son[p][s];
        }
        return ans;
    }
}T;
```

7.二维BIT

```
struct bit{
    #define lowbit(x) x&(-x)
    #define il inline
    int c[N][N],n;
    il void ins(int x,int y,int w){
        for(int i=x;i<=n;i+=lowbit(i))
            for(int j=y;j<=n;j+=lowbit(j))
                c[i][j]+=w;
    }
    il int que(int x,int y){
        int s=0;
        for(int i=x;i;i-=lowbit(i))
            for(int j=y;j;j-=lowbit(j))
                s+=c[i][j];
        return s;
    }
    il int subq(int a,int b,int x,int y){
        return que(x,y)-que(a-1,y)-que(x,b-1)+que(a-1,b-1);
    }
}T;
```

8.高精度

```
struct num{
    int len,c[N];
    num(){} //no parameter init
    num(int x){ mst(c,0);//init by number
    if(!x) len=1;
    else{len=0;while(x) c[++len]=x%10,x/=10;}
    }
    num operator +(const num &b)const{  //高精加
        num ans(0);
        int l=max(len,b.len),p=0;
        ans.len=l;
    for(int i=1;i<=l;i++){
         ans.c[i]=c[i]+b.c[i]+p;
         p=ans.c[i]/10;
         ans.c[i]%=10;
    }
    if(p) ans.c[++ans.len]=p;
    return ans;
    }
    num operator *(const num &b)const{  //高精乘高精
        num ans(0);
    int l=len+b.len-1,p=0;
    for(int i=1;i<=len;i++)
        for(int j=1;j<=b.len;j++)
            ans.c[i+j-1]+=c[i]*b.c[j];
    for(int i=1;i<=l;i++){
        ans.c[i+1]+=ans.c[i]/10;
        ans.c[i]%=10;
    }
    if(ans.c[l+1]) l++;ans.len=l;
    return ans;
    }
    num operator *(int &x){ //高精乘低精
        int p=0;
     for(int i=1;i<=len;i++){
          c[i]=c[i]*x+p;
          p=c[i]/10;
          c[i]%=10;
     }
     while(p){
        c[++len]=p;
        p/=10;
        c[len]%=10;
     }
     return *this;
    }
};
num fac(int n){  //Factorial: n!
     num a(1);
     int p[N];mst(p,0);
     for(int i=1;i<=n;i++){
        for(int j=1;j<=a.len;j++){
             a.c[j]=i*a.c[j]+p[j];p[j]=0;
```

```
              if(a.c[j]>9){
                  p[j+1]=a.c[j]/10;
                  a.c[j]%=10;
                  if(j==a.len) a.len++;
               }
          }
      }
      return a;
}
num sub(num a,num b){  //高精减
    int f=0;
    if(a.len<b.len) swap(a,b),f=1;
    else if(a.len==b.len){
        for(int i=1;i<=a.len;i++)
            if(a.c[i]<b.c[i]){
                f=1;
                swap(a,b);
                break;
            }
    }
    for(int i=1;i<=a.len;i++){
        int x=a.c[i]-b.c[i];
        if(x<0) x+=10,a.c[i+1]--;
        a.c[i]=x;
    }
    if(f) printf("-");
    int j=a.len;
    while(!a.c[j]&&j>1) j--;
    a.len=j;
    return a;
}
void div(num &a,int x){  //高精除低精度.(向下取整)
    int p=0;
    for(int i=a.len;i>=1;i--){
        p=p*10+a.c[i];
        a.c[i]=p/x;
        p%=x;
    }
    while(!a.c[a.len]&&a.len>1) a.len--;
}
bool jg(num a,num b){ //高精度比较
    if(a.len!=b.len) return a.len>b.len;
    else for(int i=1;i<=a.len;i++)
                if(a.c[i]!=b.c[i])
                    return a.c[i]>b.c[i];
}
num A(int n,int m){ //A(n,m) 高精度排列
    if(n<m) return num(0);
    else {num ans(1);
        for(int i=n-m+1;i<=n;i++) ans=ans*i;
        return ans;
    }
}
void Print(num a){
    while(!a.c[a.len]&&a.len>1) a.len--;
    for(int i=a.len;i;i--)
        printf("%d",a.c[i]);
    printf("\n");
```

```
}
```

9.线段树

```
//区间修改 区间求和
#define il inline
#define lx x<<1
#define rx x<<1|1
#define len(x) (a[x].r-a[x].l+1)
struct node{
    int l,r,lz;
    ll s;
}a[N<<2];
il void re(int x){
    a[x].s=a[lx].s+a[rx].s;
}
il void ptg(int x,int y){
    a[x].lz+=y,a[x].s+=1LL*len(x)*y;
}
il void pd(int x){
    if(a[x].lz){
        ptg(lx,a[x].lz),ptg(rx,a[x].lz);
        a[x].lz=0;
    }
}
il void bud(int x,int l,int r){
    a[x].l=l,a[x].r=r;
    if(l==r){
        return;
    }
    int m=(l+r)>>1;bud(lx,l,m),bud(rx,m+1,r);
    re(x);
}
il void upd(int x,int l,int r,int val){
    if(a[x].l>=l&&a[x].r<=r){
        ptg(x,val);return;
    }
    pd(x);
    int m=(a[x].l+a[x].r)>>1;
    if(l<=m) upd(lx,l,r,val);
    if(r>m) upd(rx,l,r,val);
    re(x);
}
il ll que(int x,int l,int r){
    if(a[x].l>=l&&a[x].r<=r) return a[x].s;
    pd(x);
    int m=(a[x].l+a[x].r)>>1;ll ans=0;
    if(l<=m) ans+=que(lx,l,r);
    if(r>m) ans+=que(rx,l,r);
    return ans;
}
```

数论

$$(S):\quad \begin{cases} x \equiv a_1 & \pmod{m_1} \\ x \equiv a_2 & \pmod{m_2} \\ \quad\vdots \\ x \equiv a_n & \pmod{m_n} \end{cases}$$

1.CRT中国剩余定理

```
void exgcd(ll a,ll b,ll &x,ll &y){
    //ax+by=1
    if(!b){
        x=1,y=0;
        return;
    }
    exgcd(b,a%b,y,x);
    y-=(a/b)*x;
}
ll m[N],a[N];//m[i] modulus (relatively prime) a[i] remainder
ll crt(int n){ //Chinese Remainder Theorem  (nlogM)
    ll M=1;//product of modulus
    for(int i=1;i<=n;i++)
        scanf("%lld%lld",&m[i],&a[i]),M*=m[i];
    ll ans=0;
    for(int i=1;i<=n;i++){
        ll x,y;
        ll u=M/m[i];// Inverse
        exgcd(u,m[i],x,y);// ux+m[i]y=1
        ans=(ans+1LL*a[i]*u*x%M)%M; //sum of a[i]*u*x
    }
    return (ans%M+M)%M;//Minimum positve integer
}
```

2.EXCRT

```
ll exgcd(ll a,ll b,ll &x,ll &y){
    //ax+by=1
    if(!b){
        x=1,y=0;
        return a;
    }
    ll g=exgcd(b,a%b,y,x);
    y-=(a/b)*x;
    return g;
}
ll qmul(ll a,ll b,ll m){
    ll s=0;
    while(b){
        if(b&1) s=(s+a)%m;
        a=(a+a)%m;
        b>>=1;
    }
    return s;
}
ll m[N],a[N];//m[i] modulus (relatively prime) a[i] remainder
ll excrt(int n){ //Chinese Remainder Theorem  (nlogM)
    ll M=1,ans=0;//product of modulus
    for(int i=1;i<=n;i++){
        ll x,y,c=(a[i]-ans%m[i]+m[i])%m[i];
        ll g=exgcd(M,m[i],x,y);
        if(c%g) return -1;//no answer
        ll g1=m[i]/g,g2=c/g;
        //x=qmul(x,g2,g1);  prevent (long long) overflow
        x=x*g2%g1;
        ans+=x*M;
        M*=g1;
        ans%=M;
        //ans=(ans%M+M)%M;
    }
    return (ans%M+M)%M;//Minimum positve integer
}
```

3.Lucas

```
ll fac[N];
int p;
void Init(int p){
    fac[0]=1;
    for(int i=1;i<p;i++) fac[i]=fac[i-1]*i%p;
}
ll ksm(ll a,ll n,int m=mod){
    ll ans=1;
    while(n){
        if(n&1) ans=ans*a%m;
        a=a*a%m;
        n>>=1;
    }
    return ans;
}
ll C(ll n,ll m){
    if(n<m) return 0;
    return fac[n]*ksm(fac[m]*fac[n-m]%p,p-2,p)%p;
}
ll Lucas(ll n,ll m){
    if(!m) return 1;
    return C(n%p,m%p)*Lucas(n/p,m/p)%p;
}
```

4.EXLucas

求解：$C_n^m \pmod p$, p可为非质数。

时间复杂度：$O(\sqrt{p}+\sum p_i^{c_i})$

```
//C(n,m)%p   p can be nor-prime
#define il inline
#define rep(i,a,b) for(ll i=a;i<=b;++i)
ll mod;
il void exgcd(ll a,ll b,ll &x,ll &y){
    if(!b) {x=1,y=0;return;}exgcd(b,a%b,y,x),y-=a/b*x;return;
}
il ll inv(ll n,ll p){ll x,y;exgcd(n,p,x,y);return (x+p)%p;}
il ll ksm(ll a,ll n,ll m=mod){
    ll s=1;for(;n;n>>=1,a=a*a%m) if(n&1) s=s*a%m;return s;
}
il ll crt(int n,ll *a,ll *m){
    ll M=1,s=0;
    rep(i,1,n) M*=m[i];
    rep(i,1,n){
        ll w=M/m[i];s=(s+a[i]*w%mod*inv(w,m[i])%mod)%mod;
    }return (s+mod)%mod;
}
il ll cal(ll n,ll q,ll qk){
    if(!n) return 1;ll s=1;
    rep(i,1,qk) if(i%q) s=s*i%qk;s=ksm(s,n/qk,qk);
    rep(i,n/qk*qk+1,n) if(i%q) s=s*(i%qk)%qk;
    return s*cal(n/q,q,qk)%qk;
}
il ll mutiLucas(ll n,ll m,ll q,ll qk){
    int cnt=0;ll inv1=inv(cal(m,q,qk),qk),inv2=inv(cal(n-m,q,qk),qk);
    for(ll i=n;i;i/=q) cnt+=i/q;
    for(ll i=m;i;i/=q) cnt-=i/q;
    for(ll i=n-m;i;i/=q) cnt-=i/q;
    return ksm(q,cnt,qk)*cal(n,q,qk)%qk*inv1%qk*inv2%qk;
}
il ll exLucas(ll n,ll m,ll p){
    int c=0;ll qk[20],a[20];
    for(ll i=2;i*i<=p;++i)
        if(p%i==0){
            qk[++c]=1;while(p%i==0) qk[c]*=i,p/=i;
            a[c]=mutiLucas(n,m,i,qk[c]);
        }
    if(p>1) qk[++c]=p,a[c]=mutiLucas(n,m,p,p);return crt(c,a,qk);
}
```

4.FFT

```
#define rgi register int
const double PI=acos(-1);
struct CP{
    CP (double x=0,double y=0):x(x),y(y){}
    double x,y;
    CP operator +(CP const &B)const {return CP(x+B.x,y+B.y);}
    CP operator -(CP const &B)const {return CP(x-B.x,y-B.y);}
    CP operator *(CP const &B)const {return CP(x*B.x-y*B.y,x*B.y+y*B.x);}
}A[N],B[N],C[N];
int tr[N],n,m,li;
void fft(CP *f,bool flag){//flag=1 DFT flag=0 IDFT
     for(rgi i=0;i<li;i++) if(i<tr[i]) swap(f[i],f[tr[i]]);//蝴蝶变换
     for(rgi p=2,len=1;p<=li;p<<=1,len<<=1){ //枚举区间长度
        CP tG(cos(2*PI/p),sin(2*PI/p));if(!flag) tG.y*=-1;//p次单位根
        for(rgi k=0;k<li;k+=p){ //枚举起点
            CP buf(1,0);
            for(rgi l=k;l<k+len;l++){ //合并区间
                CP tt=buf*f[len+l];
                f[len+l]=f[l]-tt;
                f[l]=f[l]+tt;
                buf=buf*tG;
            }
        }
     }
     if(!flag) for(rgi i=0;i<li;i++) f[i].x/=li;
}
```

5.高斯消元

```
//Gauss1
int n;
double a[N][N];
void Gauss(){ //O(n^3)
    rep(i,1,n){
        int mx=i;
        rep(j,i+1,n){
            if(fabs(a[j][i])>fabs(a[mx][i])) mx=j;
        }
        if(i!=mx)swap(a[i],a[mx]);
        if(!a[i][i]) return (void)puts("No Solution");
        rep(j,1,n){
            if(j!=i){
                double x=a[j][i]/a[i][i];
                rep(k,i+1,n+1)
                    a[j][k]-=x*a[i][k];
            }
        }
    }
    rep(i,1,n) printf("%.2f\n",a[i][n+1]/a[i][i]);
}
//Gauss2
int n;
const double eps=1e-6;
double a[N][N],ans[N];
void Gauss(){ //O(n^3)
    rep(i,1,n){
        int mx=i;
        rep(j,i+1,n){
            if(fabs(a[j][i])>fabs(a[mx][i])) mx=j;
        }
        if(mx!=i) swap(a[mx],a[i]);
        if(!a[i][i]) return (void)puts("No Solution");
        double x=a[i][i];
        rep(j,i,n+1) a[i][j]/=x;
        rep(j,i+1,n){
            x=a[j][i];
            rep(k,i,n+1) a[j][k]-=a[i][k]*x;
        }
    }
    ans[n]=a[n][n+1];
    per(i,n-1,1){   //回代
        ans[i]=a[i][n+1];
        rep(j,i+1,n)
            ans[i]-=a[i][j]*ans[j];
    }
    rep(i,1,n) printf("%.2f\n",ans[i]);
}
//Gauss2
int n;
const double eps=1e-6;
```

```
double a[N][N],ans[N];
int Gauss(){ //O(n^3)
    int r=1;
    rep(i,1,n){
        int mx=r;
        rep(j,r+1,n){
            if(fabs(a[j][i])>fabs(a[mx][i])) mx=j;
        }
        if(fabs(a[mx][i])<eps) continue;
        if(mx!=r) swap(a[mx],a[r]);
        double x=a[r][i];
        rep(j,i,n+1) a[r][j]/=x;
        rep(j,r+1,n){
            x=a[j][i];
            rep(k,i,n+1) a[j][k]-=a[r][k]*x;
        }
        r++;
    }
    ans[n]=a[n][n+1];
    per(i,n-1,1){   //回代
        ans[i]=a[i][n+1];
        rep(j,i+1,n)
            ans[i]-=a[i][j]*ans[j];
    }
    if(r<n+1){
        rep(i,r,n) if(fabs(ans[i])>eps) return -1; //无解
        return 0; //多解 (n+1-r) 个自由元.
    }
    return 1; //唯一解.
}
```

6.inv逆元

```
ll inv[N],fac[N],finv[N];
ll C(ll n,ll m){
    return fac[n]*finv[n-m]%mod*finv[m]%mod;
}
ll p=mod;
void init(int n){
    fac[0]=fac[1]=inv[1]=finv[1]=finv[0]=1;
    for(int i=2;i<=n;i++)
    fac[i]=fac[i-1]*i%mod,inv[i]=(p-p/i)*inv[p%i]%p,finv[i]=finv[i-1]*inv[i]%p;
}
```

7.Inv_matrix矩阵的逆

```
int n;
ll a[N][N];
ll ksm(ll a,ll n,ll m=mod){
    ll ans=1;
    while(n){
        if(n&1) ans=ans*a%m;
        a=a*a%m;
        n>>=1;
    }
    return ans;
}
void Inv_Matrix(){  //O(n^3)
    int m=n<<1;
    rep(i,1,n){
        int mx=i;
        rep(j,i+1,n){
            if(a[j][i]>a[mx][i]) mx=j;
        }
        if(i!=mx) swap(a[i],a[mx]);
        if(!a[i][i]) return (void)puts("No Solution");
        ll x=ksm(a[i][i],mod-2);
        rep(j,1,n){
            if(j!=i){
                ll y=a[j][i]*x%mod;
                rep(k,i+1,m)
                    (a[j][k]-=y*a[i][k]%mod)%=mod;
            }
        }
        rep(j,1,m)  a[i][j]=a[i][j]*x%mod;
    }
    rep(i,1,n){
        rep(j,n+1,m) printf("%lld ",(a[i][j]%mod+mod)%mod);
    printf("\n");
    }
}
int main(){
    scanf("%d",&n);
    rep(i,1,n){
        rep(j,1,n) scanf("%lld",&a[i][j]);
        a[i][i+n]=1;
    }
    Inv_Matrix();
    return 0;
}
```

8.拉格朗日插值法

```
//Lagrange Interpolation
#define il inline
il ll ksm(ll a,ll n,ll m=mod){ll s=1;while(n){if(n&1)
s=s*a%m;a=a*a%m;n>>=1;}return s;}
struct LR{
    ll fac[N+5],facinv[N+5],inv[mod+5];
    il ll Inv(ll n){return ksm(n,mod-2);}
    il void init(){ //预处理阶乘和阶乘逆元,逆元.
        fac[0]=inv[0]=inv[1]=1;for(int i=1;i<=N;i++) fac[i]=fac[i-1]*i%mod;
        facinv[N]=Inv(fac[N]);
        for(int i=N-1;~i;i--) facinv[i]=facinv[i+1]*(i+1)%mod;
        for(int i=2;i<mod+5;i++)
            inv[i]=(mod-mod/i)*inv[mod%i]%mod;
    }
    il ll cal(ll *x,ll *y,int n,ll k){  //离散点n个点[0,n-1] x[i],y[i] 插f(k)
        ll s=0;
        for(int i=0;i<=n;i++){
            ll p=y[i]%mod,q=1;
            for(int j=0;j<=n;j++){
                if(i==j) continue;
                p=p*(k-x[j])%mod;
                q=q*(x[i]-x[j])%mod;
            }
            s=(s+p*Inv(q)%mod)%mod;
        }return (s%mod+mod)%mod;
    }
    il ll inpo(ll *f,int n,ll x){   //给定 连续i属于[0,n] f(i) 拉插f(x)
        ll c=1,p,s=0;
        for(int i=0;i<=n;i++) c=c*(x-i)%mod;
        for(int i=0;i<=n;i++){
            p=c*inv[x-i]%mod*facinv[n-i]%mod*facinv[i]%mod;
            if((n-i)&1)  s=(s-p*f[i]%mod)%mod;
            else s=(s+p*f[i]%mod)%mod;
        }
        return (s%mod+mod)%mod;
    }
}L;
```

9.矩阵快速幂

```
struct mtx{
    int a[95][95];
    int r,c;
    mtx(int rr,int cc,int p=0){
        r=rr,c=cc;
        mst(a,0);
        if(p==1) for(int i=1;i<=rr;i++) a[i][i]=1;
    }
    mtx operator *(const mtx &b)const{
        mtx m(r,b.c);
            for(int i=1;i<=r;i++)
            for(int j=1;j<=m.c;j++)
                for(int k=1;k<=c;k++)
            m.a[i][j]=(m.a[i][j]+a[i][k]*b.a[k][j])%mod;
        return m;
    }
    mtx operator ^(int n)const{
        mtx ans(r,c,1),x=*this;
        while(n){
        if(n&1) ans=ans*x;
        x=x*x;
        n>>=1;
        }
        return ans;
    }
};
```

10.辛普森积分

```
#define il inline
#define db double
il db f(db x){
    // set function
}
il db simpson(db l,db r){
    db m=(l+r)/2;
    return (f(l)+f(r)+4*f(m))*(r-l)/6;
}
il db asr(db l,db r,db eps,db ans){
    db m=(l+r)/2;
    db ls=simpson(l,m),rs=simpson(m,r);
    if(fabs(ls+rs-ans)<=eps*15) return ls+rs+(ls+rs-ans)/15;
    return asr(l,m,eps/2,ls)+asr(m,r,eps/2,rs);
}
```

11.扩欧

```
void exgcd(int a,int b,int &x,int &y){
    //ax+by=1
    if(!b){
        x=1,y=0;
        return;
    }
    exgcd(b,a%b,y,x);
    y-=(a/b)*x;
}
```

12.欧拉降幂

```
//欧拉降幂
#define mod(a,b) a<b?a:a%b+b
/*O(n) 预处理欧拉函数
int phi[N],p[N],cnt;
bitset<N>vis;
void ss(int n){
    phi[1]=1;
    for(int i=2;i<=n;i++){
        if(!vis[i]) p[++cnt]=i,phi[i]=i-1;
        for(int j=1;j<=cnt&&i*p[j]<=n;j++){
            vis[i*p[j]]=1;
            if(i%p[j]==0){
                phi[i*p[j]]=phi[i]*p[j];
                break;
            }phi[i*p[j]]=phi[i]*phi[p[j]];
        }
    }
}*/
//记忆化+根号n 计算欧拉函数.
unordered_map<ll,ll>mp;
ll phi(ll n){
    if(mp[n]) return mp[n];
    ll s=n,nn=n;
    for(ll i=2;i*i<=n;i++){
        if(n%i==0){
            s-=s/i;
            while(n%i==0) n/=i;
        }
    }if(n>1) s-=s/n;return mp[nn]=s;
}
ll ksm(ll a,ll n,ll m){
    ll s=1;
    a=mod(a,m);
    while(n){
        if(n&1) s=mod(s*a,m);
        a=mod(a*a,m);
        n>>=1;
    }return s;
}
// 幂塔函数 (2^2)^2...  n次
int fun(int a, int n,int m) {
    if (n==1||m==1) return mod(a,m);
    return ksm(a,fun(a,n-1,phi(m)),m);
}
//计算 ( (a[l]) ^ a[l+1] ) ^ a[l+2] ..... 最外层是a[r]
//O(log^2p)
ll solve(int l,int r,int m){
   if(l==r||m==1) return mod(a[l],m);
   return ksm(a[l],solve(l+1,r,phi(m)),m);
}
```

13.组合数

```
// Combinatorics
#define il inline
#define rgi register int
ll ksm(ll a,ll n,ll m=mod){
    ll s=1;while(n){
        if(n&1) s=s*a%mod;
        a=a*a%mod;
        n>>=1;
    }return s;
}
struct Comb{
    int n;
    vector<ll>fac,invfac;
    Comb(int n):n(n){
        fac.assign(n+1,0),invfac.assign(n+1,0);
        fac[0]=1;
        for(rgi i=1;i<=n;i++) fac[i]=fac[i-1]*i%mod;
        invfac[n]=inv(fac[n]);
        for(int i=n-1;~i;i--) invfac[i]=invfac[i+1]*(i+1)%mod;
    }
    il ll C(int n,int k){
        if(n<0||k<0||n<k) return 0LL;
        return fac[n]*(invfac[k]*invfac[n-k]%mod)%mod;
    }
    il inv(ll n){
        return ksm(n,mod-2);
    }
};
```

14.最小圆覆盖

```
struct node{
    double x,y;
    node(const double x=0,const double y=0):x(x),y(y){}
    node operator +(const node&a)const{return node(x+a.x,y+a.y);}
    node operator -(const node&a)const{return node(x-a.x,y-a.y);}
    node operator * (const double&a)const{return node(x*a, y*a);}
    node operator / (const double&a)const{return node(x/a, y/a);}
    const double len()const{return sqrt(x*x+y*y);}//len^2
    node norm() const {return *this/len();}
    node rt90() {return node(y, -x);}
};
double dot(const node &a,const node &b){
    return a.x*b.x+a.y*b.y;
}
double cross(const node &a,const node &b){
    return a.x*b.y-b.x*a.y;
}
node lcp(const node& p0, const node& v0, const node& p1, const node& v1){
    //crossover point of two lines
    //line consists of (point and direction vector)
    double t=cross(p1-p0,v1)/cross(v0,v1);
    return p0+v0*t;
}

node circle(const node& a, const node& b, const node& c)
{
    return lcp((a+b)/2,(b-a).rt90(),(a+c)/2,(c-a).rt90());
}
int n;
node a[N];
double r;
void Welzl(){
    node o;r=0;//o: center of a circle   r: radius
    random_shuffle(a+1,a+n+1);
    for(int i=1;i<=n;i++){
        if((a[i]-o).len() > r){
            o=a[i],r=0;
            for(int j=1;j<i;j++){
                if((a[j]-o).len()>r){
                    o=(a[i]+a[j])/2,r=(a[j]-o).len();
                    for(int k=1;k<j;k++)
                        if((a[k]-o).len()>r)
                  o = circle(a[i],a[j],a[k]),r=(a[k]-o).len();
                }
            }
        }
    }
    printf("%.10f\n%.10f %.10f\n",r, o.x, o.y);
}
```

图论

1.dijkstra最短路

```
int h[N],cnt,d[N];
struct edge{
    int to,nt,w;
}e[M<<1];
void add(int u,int v,int w){
    e[++cnt]={v,h[u],w},h[u]=cnt;
}
bitset<N>vis;
void dij(int st,int ed){
    priority_queue<PII>q;mst(d,0x3f),d[st]=0,q.push({0,st});
    while(!q.empty()){
        int u=q.top().se;q.pop();
        if(vis[u]) continue;vis[u]=1;
        for(int i=h[u];i;i=e[i].nt){
            int v=e[i].to,w=e[i].w;
            if(!vis[v]&&d[v]>d[u]+w){
                d[v]=d[u]+w;
                q.push({-d[v],v});
            }
        }
    }
}
```

2.dinic

```
//Dinic O(n^2m)
int n,m,st,ed;
int id(int x,int y){
    return (x-1)*m+y;
}
struct edge{
    int to,nt,w;
}e[M];
int h[N],cur[N],cnt=1,dep[N];
void add(int u,int v,int w){
    e[++cnt]={v,h[u],w},h[u]=cnt;
    e[++cnt]={u,h[v],0},h[v]=cnt;
}
int dfs(int u,int c){   //search for augment path
    if(u==ed) return c;
    int res=c;
    for(int &i=cur[u];i;i=e[i].nt){
        int v=e[i].to,w=e[i].w;
        if(w&&dep[v]==dep[u]+1){
            int now=dfs(v,min(res,w));
            if(!now) dep[v]=1;
            else e[i].w-=now,e[i^1].w+=now,res-=now;
        }
        if(!res) return c;
    }return c-res;
}
bool bfs(){     //layer the graph
    queue<int>q;q.push(st);mst(dep,0),dep[st]=1;
    while(!q.empty()){
        int u=q.front();q.pop();cur[u]=h[u];
        for(int i=h[u];i;i=e[i].nt){
            int v=e[i].to,w=e[i].w;
```

```
                if(w&&!dep[v]) dep[v]=dep[u]+1,q.push(v);
            }
        }return dep[ed];
}
ll dinic(){
    ll s=0;
    while(bfs()) s+=dfs(st,inf);
    return s;
}
```

3.并查集dsu

```
int s[N];
int find(int x){return x==s[x]?x:s[x]=find(s[x]);}
void Init(int n){
    for(int i=1;i<=n;i++) s[i]=i;
}
```

4.KM

是一种在二分图上求解最大权完美匹配的算法，用邻接矩阵存图即可。相比费用流较高的复杂度，km算法有着更为优秀的效率，但局限性在于只能做匹配，而不像费用流那样灵活可变。

时间复杂度：$O(n^3)$

```
const int N=407,inf=1e9+7;
struct KM{      //KM algorithm
    #define il inline
    ll a[N][N],hl[N],hr[N],slk[N];
    int n,fl[N],fr[N],vl[N],vr[N],pre[N],q[N],ql,qr;
    il int check(int i){
        if(vl[i]=1,~fl[i])return vr[q[qr++]=fl[i]]=1;
        while(~i)swap(i,fr[fl[i]=pre[i]]);
        return 0;
    }
    void bfs(int s){    //search augments
        for(int i=0;i<n;i++) vl[i]=vr[i]=0,slk[i]=inf;
        for(vr[q[ql=0]=s]=qr=1;;){
            for(ll d; ql<qr;)
                for(int i=0,j=q[ql++]; i<n; ++i)
                    if(!vl[i]&&slk[i]>=(d=hl[i]+hr[j]-a[i][j]))
                        if(pre[i]=j,d)slk[i]=d;
                        else if(!check(i))return;
            ll d=inf;
            for(int i=0; i<n; ++i)
                if(!vl[i]&&d>slk[i])d=slk[i];
            for(int i=0; i<n; ++i){
                if(vl[i])hl[i]+=d;
                else slk[i]-=d;
                if(vr[i])hr[i]-=d;
            }
            for(int i=0; i<n; ++i)
                if(!vl[i]&&!slk[i]&&!check(i))return;
        }
    }
    il void match(){
        for(int i=0;i<n;i++) fl[i]=fr[i]=-1,hr[i]=0;
        for(int i=0; i<n; ++i)hl[i]=*max_element(a[i],a[i]+n);
        for(int j=0; j<n; ++j)bfs(j);
    }
    il ll que(){ //max value ans
    ll ans=0;
    for(int i=0; i<n; ++i)ans+=a[i][fl[i]];return ans;
    }
    il void fun(int n){
    //Print match point,if i stary by 0 : modfiy i [0,n-1]
        for(int i=1;i<=n;i++) printf("%d ",a[i][fl[i]]?fl[i]:0);printf("\n");
    }
}km;
```

5.MCMF

```
int cnt=1,h[N],flow[N],dis[N],vis[N],n,m,st,ed;
ll mc,mf;
int id(int x,int y){
    return (x-1)*n+y;
}
queue<int>q;
struct edge{
    int to,nt,f,w;//f:flow ,w:cost
}e[M];
struct Pre{
    int i,u;
}pre[N];//记录前驱结点和边的信息，便于更新残余网络，建立反边.(反悔和贪心的思想)
void add(int u,int v,int f,int w){
    e[++cnt]={v,h[u],f,w},h[u]=cnt;
    e[++cnt]={u,h[v],0,-w},h[v]=cnt;
}
bool spfa(){// 跑spfa
    mst(dis,0x3f),mst(flow,0x3f),mst(vis,0);    //初始化.
    q.push(st),vis[st]=1,dis[st]=0,pre[ed].u=-1;//预处理
    while(!q.empty()){
        int u=q.front();q.pop();vis[u]=0;
        for(int i=h[u];i;i=e[i].nt){
            int v=e[i].to,f=e[i].f,w=e[i].w;
            if(f>0&&dis[v]>dis[u]+w){
                dis[v]=dis[u]+w;
                pre[v].u=u,pre[v].i=i;
                flow[v]=min(flow[u],f);
                if(!vis[v]) vis[v]=1,q.push(v);
            }
        }
    }
    return pre[ed].u!=-1;
}
void MCMF(){    //MIncost Maxflow
    while(spfa()){
        int u=ed,x=flow[ed];
        mf+=x;
        mc+=x*dis[u];
        while(u!=st){
            e[pre[u].i].f-=x;
            e[pre[u].i^1].f+=x;
            u=pre[u].u;
        }
    }
}
```

6.spfa

```
int h[N],cnt,d[N];
struct edge{
    int to,nt,w;
}e[M<<1];
void add(int u,int v,int w){
    e[++cnt]={v,h[u],w},h[u]=cnt;
}
bitset<N>vis;
void spfa(int st,int ed){
    queue<int>q;mst(d,0x3f),d[st]=0,vis[st]=1;q.push(st);
    while(!q.empty()){
        int u=q.front();q.pop();vis[u]=0;
        for(int i=h[u];i;i=e[i].nt){
            int v=e[i].to,w=e[i].w;
            if(d[v]>d[u]+w){
                d[v]=d[u]+w;
                if(!vis[v]) q.push(v),vis[v]=1;
            }
        }
    }
}
```

7.Tarjan

```
//tarjan O(n+m)
int n,m,h[N],cnt;
int dfn[N],low[N],id,vis[N];
int col[N],num[N],sum;
//bool cut[N]; judge cut point
stack<int>s;
struct edge{
    int to,nt;
}e[M];
void add(int u,int v){
    e[++cnt]={v,h[u]},h[u]=cnt;
}
void dfs(int u){
    low[u]=dfn[u]=++id;vis[u]=1;
    s.push(u);
    //int son=0;
    for(int i=h[u];i;i=e[i].nt){
        int v=e[i].to;
        if(!dfn[v]){
            dfs(v);
            low[u]=min(low[u],low[v]);
            //if(low[v]>=dfn[u]) cut[u]=1;
        }else if(vis[v]) low[u]=min(low[u],dfn[v]);
    }
    if(dfn[u]==low[u]){ //shrink point
        col[u]=++sum;vis[u]=0;
        while(s.top()!=u){
            col[s.top()]=sum;vis[s.top()]=0;
            s.pop();
        }
        s.pop();
    }
    //if(u==rt&&son==1) cut[u]=0;
}
```

8.倍增LCA

```
//倍增LCA
int n;
struct edge{
    int to,nt;
}e[N<<1];
int h[N],cnt,f[N][21],dep[N];
void add(int u,int v){
    e[++cnt]={v,h[u]},h[u]=cnt;
    e[++cnt]={u,h[v]},h[v]=cnt;
}
void dfs(int u){
    for(int i=h[u];i;i=e[i].nt){
        int v=e[i].to;
        if(v!=f[u][0]){
            f[v][0]=u;
            dep[v]=dep[u]+1;
            dfs(v);
        }
    }
}
void init(){
    for(int j=1;j<=20;j++)
        for(int i=1;i<=n;i++)
            f[i][j]=f[f[i][j-1]][j-1];
}
int lca(int u,int v){
    if(dep[u]<dep[v]) swap(u,v);
    int delta=dep[u]-dep[v];
    for(int i=0;i<=20;i++)
        if(delta>>i&1) u=f[u][i];
    if(u==v) return u;
    for(int i=20;~i;i--){
        if(f[u][i]!=f[v][i]) u=f[u][i],v=f[v][i];
    }
    return f[u][0];
}
```

9.点分治

```
//点分治模板
#define hh(u,i) for(int i=h[u];i;i=e[i].nt)
int h[N],cnt,n,k,vis[N];
int mx[N],sz[N],rt,sum,b[N],tmp[N],tot;
ll ans;
struct edge{
    int to,nt;
}e[N<<1];
inline void add(int u,int v){
    e[++cnt]={v,h[u]},h[u]=cnt;e[++cnt]={u,h[v]},h[v]=cnt;
}
void grt(int u,int fa){
    mx[u]=0,sz[u]=1;hh(u,i){int v=e[i].to;if(v==fa||vis[v]) continue;
    grt(v,u),mx[u]=max(mx[u],sz[v]),sz[u]+=sz[v];
    }mx[u]=max(mx[u],sum-sz[u]);if(mx[u]<mx[rt]) rt=u;
}
void dfs(int u,int fa,int d){
    if(d<=k){
        ans+=b[k-d];
        tmp[d]++;
    }
    for(int i=h[u];i;i=e[i].nt){
        int v=e[i].to;
        if(v==fa||vis[v]) continue;
        dfs(v,u,d+1);
    }
}
void cal(int u){
    for(int i=h[u];i;i=e[i].nt){
        int v=e[i].to;
        if(vis[v]) continue;
        mst(tmp,0);
        dfs(v,u,1);
        for(int j=1;j<=k;j++) b[j]+=tmp[j];
    }for(int j=1;j<=k;j++) b[j]=0;
}
void solve(int u){
    vis[u]=b[0]=1,cal(u);hh(u,i){int v=e[i].to;if(vis[v]) continue;
    mx[rt=0]=sum=sz[v],grt(v,u),grt(rt,0),solve(rt);
    }
}
// init: sum=mx[rt=0]=n,grt(1,0),solve(rt);
```

10.树剖

```
//树剖
struct edge{
    int to,nt;
}e[N<<1];
int h[N],cnt;
void add(int u,int v){
    e[++cnt]={v,h[u]},h[u]=cnt;
    e[++cnt]={u,h[v]},h[v]=cnt;
}
int sz[N],fa[N],son[N],top[N],dfn[N],dep[N];
void dfs(int u,int f){
    sz[u]=1,fa[u]=f,dep[u]=dep[f]+1;
    for(int i=h[u];i;i=e[i].nt){
        int v=e[i].to;
        if(v==f) continue;
        dfs(v,u);
        sz[u]+=sz[v];
        if(sz[son[u]]<sz[v]) son[u]=v;
    }
}
int id;
void dfs1(int u,int tp){
    top[u]=tp,dfn[u]=++id;w[id]=val[u];
    if(son[u]) dfs1(son[u],tp);
    for(int i=h[u];i;i=e[i].nt){
        int v=e[i].to;
        if(v!=fa[u]&&v!=son[u]) dfs1(v,v);
    }
}
int upd_c(int x,int y,int z){   //chain update
     while(top[x]!=top[y]){
        if(dep[top[x]]<dep[top[y]]) swap(x,y);
        upd(1,dfn[top[x]],dfn[x],z);
        x=fa[top[x]];
     }
     if(dep[x]<dep[y]) swap(x,y);
     upd(1,dfn[y],dfn[x],z);
}
int que_c(int x,int y){     //chain query
    int ans=0;
    while(top[x]!=top[y]){
        if(dep[top[x]]<dep[top[y]]) swap(x,y);
        ans=ans+que(1,dfn[top[x]],dfn[x]);
        x=fa[top[x]];
     }
     if(dep[x]<dep[y]) swap(x,y);
     ans=ans+que(1,dfn[y],dfn[x]);
     return ans;
}
void upd_s(int x,int y){        //substree query
    upd(1,dfn[x],dfn[x]+sz[x]-1,y);
}
int que_s(int x){       //substree query
    return que(1,dfn[x],dfn[x]+sz[x]-1);
}
```

11.匈牙利

```
struct node{
    int to,nt;
}e[M];
int cnt,h[N];
void add(int u,int v){
    e[++cnt]={v,h[u]},h[u]=cnt;
}
int vis[N],mh[N];
bool find(int u){
    for(int i=h[u];i;i=e[i].nt){
        if(vis[e[i].to]) continue;
        int v=e[i].to;
        vis[v]=1;
        if(!mh[v]||find(mh[v])){
            mh[v]=u;
            return true;
        }
    }
    return false;
}
```

12.一般图最大权匹配

```
#include <bits/stdc++.h>
#define DIST(e) (lab[e.u] + lab[e.v] - g[e.u][e.v].w * 2)
using namespace std;
typedef long long ll;
const int N = 1023, INF = 1e9;
struct Edge {
    int u, v, w;
} g[N][N];
int n, m, n_x, lab[N], match[N], slack[N], st[N], pa[N], flower_from[N][N],
S[N], vis[N];
vector<int> flower[N];
deque<int> q;
void update_slack(int u, int x) {
    if (!slack[x] || DIST(g[u][x]) < DIST(g[slack[x]][x]))
        slack[x] = u;
}
void set_slack(int x) {
    slack[x] = 0;
    for (int u = 1; u <= n; ++u)
        if (g[u][x].w > 0 && st[u] != x && S[st[u]] == 0)
            update_slack(u, x);
}
void q_push(int x) {
    if (x <= n)
        return q.push_back(x);
    for (int i = 0; i < flower[x].size(); i++) q_push(flower[x][i]);
}
void set_st(int x, int b) {
    st[x] = b;
    if (x <= n)
        return;
    for (int i = 0; i < flower[x].size(); ++i) set_st(flower[x][i], b);
}
int get_pr(int b, int xr) {
    int pr = find(flower[b].begin(), flower[b].end(), xr) - flower[b].begin();
    if (pr % 2 == 1) {
        reverse(flower[b].begin() + 1, flower[b].end());
        return (int)flower[b].size() - pr;
    } else
        return pr;
}
void set_match(int u, int v) {
    match[u] = g[u][v].v;
    if (u <= n)
        return;
    Edge e = g[u][v];
    int xr = flower_from[u][e.u], pr = get_pr(u, xr);
    for (int i = 0; i < pr; ++i) set_match(flower[u][i], flower[u][i ^ 1]);
    set_match(xr, v);
    rotate(flower[u].begin(), flower[u].begin() + pr, flower[u].end());
}
void augment(int u, int v) {
    int xnv = st[match[u]];
    set_match(u, v);
    if (!xnv)
        return;
    set_match(xnv, st[pa[xnv]]);
    augment(st[pa[xnv]], xnv);
}
int get_lca(int u, int v) {
    static int t = 0;
    for (++t; u || v; swap(u, v)) {
        if (u == 0)
            continue;
        if (vis[u] == t)
            return u;
        vis[u] = t;
        u = st[match[u]];
        if (u)
            u = st[pa[u]];
    }
    return 0;
}
void add_blossom(int u, int lca, int v) {
    int b = n + 1;
    while (b <= n_x && st[b]) ++b;
    if (b > n_x)
        ++n_x;
    lab[b] = 0, S[b] = 0;
    match[b] = match[lca];
    flower[b].clear();
    flower[b].push_back(lca);
    for (int x = u, y; x != lca; x = st[pa[y]])
        flower[b].push_back(x), flower[b].push_back(y = st[match[x]]),
q_push(y);
    reverse(flower[b].begin() + 1, flower[b].end());
    for (int x = v, y; x != lca; x = st[pa[y]])
        flower[b].push_back(x), flower[b].push_back(y = st[match[x]]),
q_push(y);
    set_st(b, b);
    for (int x = 1; x <= n_x; ++x) g[b][x].w = g[x][b].w = 0;
    for (int x = 1; x <= n; ++x) flower_from[b][x] = 0;
    for (int i = 0; i < flower[b].size(); ++i) {
        int xs = flower[b][i];
        for (int x = 1; x <= n_x; ++x)
            if (g[b][x].w == 0 || DIST(g[xs][x]) < DIST(g[b][x]))
                g[b][x] = g[xs][x], g[x][b] = g[x][xs];
        for (int x = 1; x <= n; ++x)
            if (flower_from[xs][x])
                flower_from[b][x] = xs;
    }
    set_slack(b);
}
void expand_blossom(int b) {
    for (int i = 0; i < flower[b].size(); ++i) set_st(flower[b][i], flower[b]
[i]);
```

```
    int xr = flower_from[b][g[b][pa[b]].u], pr = get_pr(b, xr);
    for (int i = 0; i < pr; i += 2) {
        int xs = flower[b][i], xns = flower[b][i + 1];
        pa[xs] = g[xns][xs].u;
        S[xs] = 1, S[xns] = 0;
        slack[xs] = 0, set_slack(xns);
        q_push(xns);
    }
    S[xr] = 1, pa[xr] = pa[b];
    for (int i = pr + 1; i < flower[b].size(); ++i) {
        int xs = flower[b][i];
        S[xs] = -1, set_slack(xs);
    }
    st[b] = 0;
}
bool on_found_Edge(const Edge &e) {
    int u = st[e.u], v = st[e.v];
    if (S[v] == -1) {
        pa[v] = e.u, S[v] = 1;
        int nu = st[match[v]];
        slack[v] = slack[nu] = 0;
        S[nu] = 0, q_push(nu);
    } else if (S[v] == 0) {
        int lca = get_lca(u, v);
        if (!lca)
            return augment(u, v), augment(v, u), 1;
        else
            add_blossom(u, lca, v);
    }
    return 0;
}
bool matching() {
    fill(S, S + n_x + 1, -1), fill(slack, slack + n_x + 1, 0);
    q.clear();
    for (int x = 1; x <= n_x; ++x)
        if (st[x] == x && !match[x])
            pa[x] = 0, S[x] = 0, q_push(x);
    if (q.empty())
        return 0;
    for (;;) {
        while (q.size()) {
            int u = q.front();
            q.pop_front();
            if (S[st[u]] == 1)
                continue;
            for (int v = 1; v <= n; ++v)
                if (g[u][v].w > 0 && st[u] != st[v]) {
                    if (DIST(g[u][v]) == 0) {
                        if (on_found_Edge(g[u][v]))
                            return 1;
                    } else
                        update_slack(u, st[v]);
                }
        }
        int d = INF;
        for (int b = n + 1; b <= n_x; ++b)
            if (st[b] == b && S[b] == 1)
                d = min(d, lab[b] / 2);
        for (int x = 1; x <= n_x; ++x)
            if (st[x] == x && slack[x]) {
                if (S[x] == -1)
                    d = min(d, DIST(g[slack[x]][x]));
                else if (S[x] == 0)
                    d = min(d, DIST(g[slack[x]][x]) / 2);
            }
        for (int u = 1; u <= n; ++u) {
            if (S[st[u]] == 0) {
                if (lab[u] <= d)
                    return 0;
                lab[u] -= d;
            } else if (S[st[u]] == 1)
                lab[u] += d;
        }
        for (int b = n + 1; b <= n_x; ++b)
            if (st[b] == b) {
                if (S[st[b]] == 0)
                    lab[b] += d * 2;
                else if (S[st[b]] == 1)
                    lab[b] -= d * 2;
            }
        q.clear();
        for (int x = 1; x <= n_x; ++x)
            if (st[x] == x && slack[x] && st[slack[x]] != x && DIST(g[slack[x]]
[x]) == 0)
                if (on_found_Edge(g[slack[x]][x]))
                    return 1;
        for (int b = n + 1; b <= n_x; ++b)
            if (st[b] == b && S[b] == 1 && lab[b] == 0)
                expand_blossom(b);
    }
    return 0;
}
pair<ll, int> weight_blossom() {
    fill(match, match + n + 1, 0);
    n_x = n;
    int n_matches = 0;
    ll tot_weight = 0;
    for (int u = 0; u <= n; ++u) st[u] = u, flower[u].clear();
    int w_max = 0;
    for (int u = 1; u <= n; ++u)
        for (int v = 1; v <= n; ++v) {
            flower_from[u][v] = (u == v ? u : 0);
            w_max = max(w_max, g[u][v].w);
        }
    for (int u = 1; u <= n; ++u) lab[u] = w_max;
    while (matching()) ++n_matches;
    for (int u = 1; u <= n; ++u)
        if (match[u] && match[u] < u)
            tot_weight += g[u][match[u]].w;
    return make_pair(tot_weight, n_matches);
}
int main() {
    cin >> n >> m;
    for (int u = 1; u <= n; ++u)
        for (int v = 1; v <= n; ++v) g[u][v] = Edge{ u, v, 0 };
    for (int i = 0, u, v, w; i < m; ++i) {
        cin >> u >> v >> w;
        g[u][v].w = g[v][u].w = w;
    }
    cout << weight_blossom().first << '\n';
    for (int u = 1; u <= n; ++u) cout << match[u] << ' ';
}
```

13.最小割树

```
//Dinic O(n^2m)
int n,m,st,ed;
int id(int x,int y){
    return (x-1)*n+y;
}
struct edge{
    int to,nt,w;
}e[M<<2],g[M<<2];
int h[N],cur[N],cnt=1,dep[N],cnt1,h1[N];
void add(int u,int v,int w){
    e[++cnt]={v,h[u],w},h[u]=cnt;
    e[++cnt]={u,h[v],0},h[v]=cnt;
}
void add1(int u,int v,int w){
    g[++cnt1]={v,h1[u],w},h1[u]=cnt1;
    g[++cnt1]={u,h1[v],w},h1[v]=cnt1;
}
int dfs(int u,int c){   //search for augment path
    if(u==ed) return c;
    int res=c;
    for(int &i=cur[u];i;i=e[i].nt){
        int v=e[i].to,w=e[i].w;
        if(w&&dep[v]==dep[u]+1){
            int now=dfs(v,min(res,w));
            if(!now) dep[v]=1;
            else e[i].w-=now,e[i^1].w+=now,res-=now;
        }
        if(!res) return c;
    }return c-res;
}
bool bfs(){     //layer the graph
    queue<int>q;q.push(st);mst(dep,0),dep[st]=1;
    while(!q.empty()){
        int u=q.front();q.pop();cur[u]=h[u];
        for(int i=h[u];i;i=e[i].nt){
            int v=e[i].to,w=e[i].w;
            if(w&&!dep[v]) dep[v]=dep[u]+1,q.push(v);
        }
    }return dep[ed];
}
ll dinic(){
    for(int i=2;i<=cnt;i+=2) e[i].w=(e[i].w+e[i^1].w),e[i^1].w=0;
    ll s=0;
    while(bfs()) s+=dfs(st,inf);
    return s;
}
int t[N],t1[N],t2[N];
```

```
void divi(int l,int r){
    if(l==r) return;// 递归终止条件: 集合的大小为1.
    st=t[l],ed=t[l+1]; //任意选择集合中的两个结点st,ed
    ll w=dinic(); //求最小割w
    add1(st,ed,w);// 最小割树建立边权edge(st,ed,w)
    int c1=0,c2=0;//分治解决两个集合.
    for(int i=l;i<=r;i++)
        if(dep[t[i]]) t1[++c1]=t[i]; //集合1,st可以访问到的结点/
        else t2[++c2]=t[i]; //st访问不到的结点.
    for(int i=l;i<l+c1;i++) t[i]=t1[i-l+1];
    for(int i=l+c1;i<=r;i++) t[i]=t2[i-l-c1+1];
    divi(l,l+c1-1);//分治
    divi(l+c1,r);
}
//倍增LCA
int f[N][21];
int mn[N][21];
void dfs(int u){
    for(int i=h1[u];i;i=g[i].nt){
        int v=g[i].to;
        if(v!=f[u][0]){
            f[v][0]=u;
            mn[v][0]=g[i].w;
            dep[v]=dep[u]+1;
            dfs(v);
        }
    }
}
void init(){
    for(int j=1;j<=10;j++)
        for(int i=1;i<=n;i++)
        {
            f[i][j]=f[f[i][j-1]][j-1];
            mn[i][j]=min(mn[i][j-1],mn[f[i][j-1]][j-1]);
        }
}
int query(int u,int v){
    if(dep[u]<dep[v]) swap(u,v);
    int delta=dep[u]-dep[v];
    int ans=inf;
    for(int i=0;i<=10;i++)
        if(delta>>i&1){
            ans=min(ans,mn[u][i]);
             u=f[u][i];
        }
    if(u==v) return ans;
    for(int i=10;~i;i--){
        if(f[u][i]!=f[v][i]){
            ans=min(ans,mn[u][i]);
            ans=min(ans,mn[v][i]);
            u=f[u][i],v=f[v][i];
        }
    }
    ans=min(ans,mn[u][0]);ans=min(ans,mn[v][0]);
    return ans;
}
int main(){
    scanf("%d%d",&n,&m);
    for(int i=1;i<=n;i++) t[i]=i;
    for(int i=1,u,v,w;i<=m;i++){
        scanf("%d%d%d",&u,&v,&w);
        add(u,v,w);
        add(v,u,w);
    }
    divi(1,n);
    mst(dep,0);
    dfs(1);
    init();
    return 0;
}
```

字符串

1.exkmp

```
int e[N],nt[N];
void Gnt(string a,int *nt){ //match itself
    int k=0,p=0,n=SZ(a);
    nt[0]=n;//nt[i] lcp
    for(int i=1;i<n;i++){
        if(i+nt[i-k]<p) nt[i]=nt[i-k];
        else {
            if(i>=p) p=i;
            while(p<n&&a[p]==a[p-i]) p++;
            nt[i]=p-i;
            k=i;
        }
    }
}
void Exd(string a,string b,int *e,int *nt){ // lcp of a[i]'s suffix in b
    int k=0,p=0,n=SZ(a),m=SZ(b);
    Gnt(b,nt);
    for(int i=0;i<n;i++){
        if(i+nt[i-k]<p) e[i]=nt[i-k];
        else {
            if(i>=p) p=i;
            while(p<n&&p-i<m&&a[p]==b[p-i]) p++;
            e[i]=p-i;
            k=i;
        }
    }
}
int cycle(string a){ //minimum cycle
    int n=a.size();
    Gnt(a,nt);
    int t=n;
    for(int i=1;i<n;i++)
        if(i+nt[i]==n){
            t=n%i?n:i;
            break;
        }
    return t;
}
```

2.kmp

```
// kmp
int p[N];
void Gnt(char *a,int n){
    int j=0;
    p[0]=p[1]=0;
    for(int i=2;i<=n;i++){
        while(j&&a[i]!=a[j+1]) j=p[j];
        if(a[i]==a[j+1]) j++;
        p[i]=j;
    }
}
void kmp(char *a,char *b){
    int n=strlen(a+1),m=strlen(b+1);
    for(int i=1,j=0;i<=n;i++){
        while(j&&a[i]!=b[j+1]) j=p[j];
        if(a[i]==b[j+1]) j++;
        if(j==m){
            printf("%d\n",i-m+1);
            j=p[j];
        }
    }
}
void Gnt(string &a){
    int n=SZ(a),j=0;
    p[0]=0;
    for(int i=1;i<n;i++){
        while(j&&a[i]!=a[j]) j=p[j-1];
        if(a[i]==a[j]) j++;
        p[i]=j;
    }
}
void kmp(string &a,string &b){
    Gnt(b);
    int n=SZ(a),m=SZ(b);
    for(int i=0,j=0;i<n;i++){
        while(j&&a[i]!=b[j]) j=p[j-1];
        if(a[i]==b[j]) j++;
        if(j==m){
            printf("%d\n",i-m+2);
            j=p[j-1];
        }
    }
}
int cycle(string &a){
    int n=SZ(a);
    return n-p[n-1];// n % len ==0  is complete
}
```

3.哈希

```
int base=131;
```

```
struct Hash{
    int n;
    ull p[N],h[N];
    char a[N];
    void init(){
        n=strlen(a+1);
        p[0]=1;
        for(int i=1;i<=n;i++)
        h[i]=h[i-1]*base+a[i]-'a'+1,p[i]=p[i-1]*base;
    }
    ull get(int l,int r){
        //check(l<=r)
        return h[r]-h[l-1]*p[r-l+1];
    }
}s,t;
```

杂项

1.Cantor展开

```
int n,q,a[N];
ll fac[N];
void init(int n){
    fac[0]=1;for(int i=1;i<=n;i++) fac[i]=fac[i-1]*i;
}
#define lowbit(x) x&(-x)
ll s[N];
void upd(int x,int v){
    while(x<=n) s[x]+=v,x+=lowbit(x);return;
}
ll que(int x){
    ll ans=0;while(x) ans+=s[x],x-=lowbit(x);return ans;
}
ll Cantor(){
    for(int i=1;i<=n;i++) scanf("%d",&a[i]),s[i]=0;
    ll ans=1;
    for(int i=n;i;i--) ans=(ans+fac[n-i]*que(a[i]-1)),upd(a[i],1);
    return ans;
}
void NCantor(ll x){
    x--;
    /*  可用权值线段树 (找全局第k小) 优化
    #define lx x<<1
#define rx x<<1|1
#define len(x) (a[x].r-a[x].l+1)
struct node{
    int l,r;
    ll s;
}a[N<<2];
inline void re(int x){
    a[x].s=a[lx].s+a[rx].s;
}
void build(int x,int l,int r){
    a[x].l=l,a[x].r=r,a[x].s=1;
    if(l==r) return;
    int mid=(l+r)>>1;
    build(lx,l,mid);
    build(rx,mid+1,r);
    re(x);
}
int query(int x,int k){
    a[x].s--;
    if(a[x].l==a[x].r) return a[x].l;
    if(a[lx].s>=k) return query(lx,k);
    else return query(rx,k-a[lx].s);
}
*/
    int vis[22]={};
    for(int i=1;i<=n;i++){
        ll y=x/fac[n-i];
        for(int j=1;j<=n;j++)
            if(!vis[j]){
                if(!y) {printf("%d ",j),vis[j]=1;break;}
                y--;
            }
        x%=fac[n-i];
    }printf("\n");
}
```

2.HashMap

```
struct HashMap{//replace unordered_map
    //  (mod<N) mod is prime
    // usually 100003(10^5) 200003
    struct node{
        int k,v,nt;//<key,value>
    }e[N];
    int h[N],cnt;
    void ins(int x){
        int u=x%mod;
        for(int i=h[u];i;i=e[i].nt)
            if(e[i].k==x) {e[i].v++;return;}
        e[++cnt]={x,1,h[u]},h[u]=cnt;
    }
    int find(int x){
        int u=x%mod;
        for(int i=h[u];i;i=e[i].nt) if(e[i].k==x) return e[i].v;
        return 0;
    }
}mp;
```

3.快读快写

```
//if have char input #define should cancel
#define getchar()(p1==p2&&(p2=(p1=buf)+fread(buf,1,1<<21,stdin),p1==p2)?
EOF:*p1++)
char buf[1<<21],*p1=buf,*p2=buf;
template <typename T>
inline T& read(T& r) {
    r = 0; bool w = 0; char ch = getchar();
    while(ch < '0' || ch > '9') w = ch == '-' ? 1 : 0, ch = getchar();
    while(ch >= '0' && ch <= '9') r = r * 10 + (ch ^ 48), ch = getchar();
    return r = w ? -r : r;
}

template<class T>
inline void write(T x)
{
    if(!x)putchar('0');if(x<0)x=-x,putchar('-');
    static int sta[25];int tot=0;
    while(x)sta[tot++]=x%10,x/=10;
    while(tot)putchar(sta[--tot]+48);
}
```

4.离散化

```
void Discret(int *a,int *b,int n){
    //离散化
    sort(b+1,b+n+1);
    int cnt=unique(b+1,b+n+1)-b-1;//(b+1) start position
    for(int i=1;i<=n;i++){
        a[i]=lower_bound(b+1,b+cnt+1,a[i])-b;
    }
}
```