

模板整理 For ZJUPC2019

By zhoushendong

2019-04-13

数论

Min_25 篩

$$\sum_{i=1}^n f(i) = 1 + \sum_{\substack{2 \leq p \leq n \\ p \text{ 是质数}}} \sum_{\substack{2 \leq x \leq n \\ x \text{ 的最小质因子为 } p}} f(x) = 1 + \sum_{\substack{2 \leq p' \leq n, e \geq 1 \\ p \text{ 是质数}}} f(p^e) \left(1 + \sum_{\substack{2 \leq x \leq \lfloor \frac{n}{p^e} \rfloor \\ x \text{ 不含 } p \text{ 的质因子}}} f(x) \right)$$

实际上也就是枚举 i 的最小质因子。

而我们注意到，对于一个合数 n 来说，其最小质因子必然不会超过 \sqrt{n} ，因此：

$$\sum_{\substack{2 \leq p^e \leq n, e \geq 1 \\ p \text{ 是质数}}} f(p^e) \left(1 + \sum_{\substack{2 \leq x \leq \lfloor \frac{n}{p^e} \rfloor \\ x \text{ 不含 } p \text{ 的质因子}}} f(x) \right) = \sum_{\substack{2 \leq p \leq \sqrt{n}, 2 \leq p^e \leq n, e \geq 1 \\ p \text{ 是质数}}} f(p^e) \left(1 + \sum_{\substack{2 \leq x \leq \lfloor \frac{n}{p^e} \rfloor \\ x \text{ 不含 } p \text{ 的质因子}}} f(x) \right) + \sum_{\substack{\sqrt{n} < p \leq n \\ p \text{ 是质数}}} f(p)$$

令

$$g_{n,m} = \sum_{\substack{2 \leq x \leq n \\ x \text{ 不含 } \leq m \text{ 的质因子}}} f(x)$$

$$h_n = \sum_{\substack{2 \leq p \leq n \\ p \text{ 是质数}}} f(p)$$

$$g_{n,m} = \sum_{\substack{2 \leq x \leq n \\ x \text{ 不含 } \leq m \text{ 的质因子}}} f(x)$$

$$= \sum_{\substack{m < p \leq \sqrt{n}, p^e \leq n, e \geq 1 \\ p \text{ 是质数}}} f(p^e) \left([e > 1] + \sum_{\substack{2 \leq x \leq \lfloor \frac{n}{p^e} \rfloor \\ x \text{ 不含 } p \text{ 的质因子}}} f(x) \right) + \sum_{\substack{m < p \leq n \\ p \text{ 是质数}}} f(p)$$

$$= \sum_{\substack{m < p \leq \sqrt{n}, p^e \leq n, e \geq 1 \\ p \text{ 是质数}}} f(p^e) \left([e > 1] + g_{\lfloor \frac{n}{p^e} \rfloor, p} \right) + h_n - h_m$$

我们可以从埃氏筛法的角度理解：埃氏筛法的过程是每次枚举一个 p ，筛去所有不小于 p^2 的 p 的倍数。于是可以令 $h'_{i,j}$ 表示埃氏筛法枚举了前 i 个素数后，不超过 j 的所有还剩下的数的 s 次方之和。即

$$h'_{i,j} = \sum_{\substack{1 \leq p \leq j \\ p \text{ 是 } 1, \text{ 质数, 或没有 } \leq p \text{ 的质因子}}} p^s$$

考虑如何转移，不妨假设第 i 个质数为 p_i ，那么考虑埃氏筛法的过程，对于 $j \geq p_i^2$ 我们有：

$$h'_{i,j} = h'_{i-1,j} - p_i^s \left(h'_{i-1, \lfloor \frac{j}{p_i} \rfloor} - h'_{i-1, p_i-1} \right)$$

在其他情况下， $h_{i,j}$ 不变。

```
exCRT
bool CRT(LL w1, LL p1, LL w2, LL p2, LL &w, LL &p){
    LL x, y, z=w2-w1, g=ex_gcd(p1, p2, x, y);
    if (z%g) return 0;
    LL t=z/g;
    x=Mul(x, t, p2/g); p=p1/g*p2;
    w=((w1+Mul(x, p1, p))%p+p)%p;
    return 1;
}
BSGS
```

```
int BSGS(int A, int B, int P){
    int M=max((int)(0.8*sqrt(1.0*P)),1), AM=Pow(A, M, P);
    Map.clear();
    for (int b=0, pw=B; b<M; b++, pw=1LL*pw*A%P)
        Map.update(pw, b+1);
    for (int a=M, pw=AM; a-M<P; a+=M, pw=1LL*pw*AM%P){
        int v=Map.find(pw);
        if (v) return a-(v-1);
    }
    return -1;
}
```

```

}

int ExBSGS(int A,int B,int P){
    A%=P,B%=P;int k=0,v=1;
    while (1){
        int g=gcd(A,P);
        if (g==1)break;
        if (B%g)return -1;
        k++,B/=g,P/=g,v=1LL*v*(A/g)%P;
        if (v==B) return k;
    }
    if (P==1) return k;
    int M=max((int)sqrt(1.0*P),1),AM=Pow(A,M,P);
    Map.clear();
    for (int b=0,pw=B;b<M;b+=1,pw=1LL*pw*A%P)
        Map.update(pw,b+1);
    for (int a=M,pw=1LL*v*AM%P;a-M<P;a+=M,pw=1LL*pw*AM%P){
        int v=Map.find(pw);
        if (v) return a-(v-1)+k;
    }
    return -1;
}

Exgcd
int exgcd(int a,int b,int &x,int &y){
    if (!b){x=1,y=0;return a;}
    int res=exgcd(b,a%b,y,x);
    y-=(a/b)*x;
    return res;
}

Exlucas
LL Inv(LL X,LL mod){
    if (!X) return 0;
    LL a=X,b=mod,x,y;
    ex_gcd(a,b,x,y);
    return x=(x%b+b)%b;
}

LL ex_lucas(LL n,LL pi,LL pk){
    if (!n) return 1LL;
    LL ans=1;
    for (LL i=2;i<=pk;i++)
        if (i%pi)ans=ans*i%pk;
    ans=Pow(ans,n/pk,pk);
    for (LL i=2;i<=n%pk;i++)
        if (i%pi)ans=ans*i%pk;
    return ans*ex_lucas(n/pi,pi,pk)%pk;
}

LL C(LL n,LL m,LL pi,LL pk){
    if (m>n) return 0;
    LL a=ex_lucas(n,pi,pk),b=ex_lucas(m,pi,pk);
    LL c=ex_lucas(n-m,pi,pk),k=0,ans;
    for (LL i=n;i;i/=pi,k+=i);
    for (LL i=m;i;i/=pi,k-=i);
    for (LL i=n-m;i;i/=pi,k-=i);
    ans=a*Inv(b,pk)%pk*Inv(c,pk)%pk*Pow(pi,k,pk)%pk;
    return ans*(P/pk)%P*Inv(P/pk,pk)%P;
}

LL C(LL n,LL m){
    LL ans=0;
    for (int i=1;i<=cnt;i++)
        ans=(ans+C(n,m,px[i],Pow(px[i],py[i],P+1)))%P;
    return ans;
}

Pollard_Rho
#include <bits/stdc++.h>
using namespace std;
typedef long long LL;
typedef unsigned long long ULL;
typedef long double LD;
namespace Pollard_Rho{
    int prime[9]={2,3,5,7,11,13,17,19,23};
    ULL RR;int Pcnt;LL p[70];
    vector<LL> res;
}

```

```

LL R(LL mod){return (RR+=4179340454199820289LL)%mod;}
LL Mul(LL x,LL y,LL mod){
    LL d=(LL)floor((LD)x*y/mod+0.5);LL
    res=x*y-d*mod;
    if (res<0)res+=mod;return res;
}
LL Pow(LL x,LL y,LL mod){
    LL ans=1%mod;
    for (;y;y>>=1,x=Mul(x,x,mod))
        if (y&1)ans=Mul(ans,x,mod);
    return ans;
}
bool Miller_Rabin(LL n){
    if (n<=1) return 0;
    for (int i=0;i<9;i++)
        if (n==prime[i])return 1;
    LL d=n-1,int tmp=0;
    while (!(d&1))d>>=1,tmp++;
    for (int i=0;i<9;i++){
        LL x=Pow(prime[i],d,n),p=x;
        for (int j=1;j<=tmp;j++){
            x=Mul(x,x,n);
            if (x==1&&p!=1&&p!=n-1) return 0;
            p=x;
        }
        if (x!=1) return 0;
    }
    return 1;
}
LL f(LL x,LL c,LL mod){return (Mul(x,x,mod)+c)%mod;}
LL gcd(LL x,LL y){return y?gcd(y,x%y):x;}
LL Get_Factor(LL c,LL n){
    LL x=R(n),y=f(x,c,n),p=n;
    while (x!=y&&(p==n||p==1)){
        p=gcd(n,max(x-y,y-x));
        x=f(x,c,n);y=f(f(y,c,n),c,n);
    }
    return p;
}
void Pollard_Rho(LL n){
    if (n<=1) return;
    if (Miller_Rabin(n)){res.push_back(n);return;}
    while (1){
        LL v=Get_Factor(R(n-1)+1,n);
        if (v!=n&&v!=1){
            Pollard_Rho(v);Pollard_Rho(n/v);return;
        }
    }
}
void work(LL n){res.clear();Pollard_Rho(n);}

二次剩余
namespace Rem2{
    int INIT_TAG=0;int t,w;
    void init(){
        INIT_TAG=1;
        srand('C'+'L'+'Y'+'A'+'K'+'I'+'O'+'I');
    }
    pair<int,int> Mul_pii(pair<int,int> A,pair<int,int>
B){
        static int a,b;
        a=((LL)A.fi*B.fi+(LL)A.se*B.se%mod*w)%mod;
        b=((LL)A.fi*B.se+(LL)A.se*B.fi)%mod;
        return make_pair(a,b);
    }
    pair<int,int> Pow_pii(pair<int,int> x,int y){
        pair<int,int> ans=make_pair(1,0);
        for (;y;y>>=1,x=Mul_pii(x,x))
            if (y&1)ans=Mul_pii(ans,x);
        return ans;
    }
    int Sqrt(int x){
        if (!INIT_TAG)init();

```

```

if (x==0) return 0;
if (Pow(x,(mod-1)/2)!=1) return -1;
do {
    t=randint()%(mod-1)+1;
    w=((LL)t*t+mod-x)%mod;
} while (Pow(w,(mod-1)/2)==1);
pair <int,int>
res=Pow_pii(make_pair(t,1),(mod+1)/2);
return min(res.fi,mod-res.fi);
}
}

字符串

exKMP
void exKMP(char s[],char t[],int g[],int f[],int n,int m){
    int Max=0,f[0]=m;
    for (int i=1;i<m;i++){
        f[i]=max(0,min(f[i-Max],Max+f[Max]-i));
        while (i+f[i]<n&&t[f[i]]==t[i+f[i]])f[i]++;
        if (!Max||i+f[i]>Max+f[Max])Max=i;
    }
    Max=0;
    for (int i=0;i<n;i++){
        g[i]=max(0,min(f[i-Max],Max+g[Max]-i));
        while (i+g[i]<n&&t[g[i]]==t[i+g[i]])g[i]++;
        if (!Max||i+g[i]>Max+g[Max])Max=i;
    }
}
}

PAM
namespace PAM{
int len[N],Fail[N],Next[N][26];
int size[N];int cnt;
void init(){
    cnt=2;len[1]=-1,Fail[1]=1;
    len[2]=0,Fail[2]=1;clr(Next),clr(size);
}
void build(char *s,int n){
    init();s[0]='*';int x=1;
    for (int i=1;i<n;i++){
        while (s[i-len[x]-1]!=s[i])x=Fail[x];
        int c=s[i]-'a';
        if (Next[x][c])x=Next[x][c];
        else {
            int y=Next[x][c]=++cnt;
            len[y]=len[x]+2;
            if (len[y]==1)Fail[y]=2;
            else {
                x=Fail[x];
                while (s[i-len[x]-1]!=s[i])
                    x=Fail[x];
                Fail[y]=Next[x][c];
            }x=y;
        }size[x]++;
    }
    for (int i=cnt;i>=1;i--)
        size[Fail[i]]+=size[i];
}
}

SA
int SA[N],rank[N],tmp[N],height[N],tax[N];
int ST[N][20];
void Sort(int n,int m){
    for (int i=0;i<=m;i++)tax[i]=0;
    for (int i=1;i<=n;i++)tax[rank[i]]++;
    for (int i=1;i<=m;i++)tax[i]+=tax[i-1];
    for (int i=n;i>=1;i--)SA[tax[rank[tmp[i]]]-1]=tmp[i];
}
bool cmp(int rk[],int x,int y,int w){
    return rk[x]==rk[y]&&rk[x+w]==rk[y+w];
}

```

```

void Suffix_Array(int s[],int n){
    clr(SA),clr(rank),clr(tmp),clr(height),clr(tax);
    for (int i=1;i<=n;i++)rank[i]=s[i],tmp[i]=i;
    int m=234;Sort(n,m);
    for (int w=1,p=0;p<n;w<<1,m=p){p=0;
        for (int i=n-w+1;i<=n;i++)tmp[++p]=i;
        for (int i=1;i<=n;i++)
            if (SA[i]>w)tmp[++p]=SA[i]-w;
        Sort(n,m);swap(rank,tmp);rank[SA[1]]=p=1;
        for (int i=2;i<=n;i++)
            rank[SA[i]]=cmp(tmp,SA[i],SA[i-1],w)?p:+p;
    }
    for (int i=1,j,k=0;i<=n;height[rank[i++]]=k)
        for (k=max(k-1,0),j=SA[rank[i]-1];s[i+k]==s[j+k];k++)
            height[1]=0;
}
void Get_ST(int n){
    memset(ST,0,sizeof ST);
    for (int i=1;i<=n;i++){
        ST[i][0]=height[i];
        for (int j=1;j<20;j++){
            ST[i][j]=ST[i][j-1];
            if (i-(1<<(j-1))>0)
                ST[i][j]=min(ST[i][j],ST[i-(1<<(j-1))][j-1]);
        }
    }
}
int Query(int L,int R){
    int val=floor(log(R-L+1)/log(2));
    return min(ST[L+(1<<val)-1][val],ST[R][val]);
}
int LCP(int x,int y){
    x=rank[x],y=rank[y];
    return Query(min(x,y)+1,max(x,y));
}
}

SAM
struct Node{int Next[26],fa,Max;}t[N<<1];
int size,last;
void init(){size=last=1;}
void extend(int c){
    if (t[p].Next[c]&&t[p].Max+1==t[t[p].Next[c]].Max)
        return t[p].Next[c];
    int p=last,np=++size,q,nq;
    t[np].Max=t[p].Max+1;
    for (;p&&!t[p].Next[c];p=t[p].fa)
        t[p].Next[c]=np;
    if (!p)t[np].fa=1;
    else {
        q=t[p].Next[c];
        if (t[q].Max==t[p].Max+1)t[np].fa=q;
        else {
            nq=++size;
            t[nq]=t[q],t[nq].Max=t[p].Max+1;
            t[q].fa=t[np].fa=nq;
            for (;p&&t[p].Next[c]==q;p=t[p].fa)
                t[p].Next[c]=nq;
        }
    }
    last=np;
}
}

Manachar
int len[N];
void Manachar(char *s,int n){
    For(i,0,n+1)len[i]=0;
    int mx=1;
    For(i,1,n){
        len[i]=max(1,min(mx+len[mx]-i,len[mx*2-i]));
        while (s[i-len[i]]==s[i+len[i]])len[i]++;
        if (i+len[i]>mx+len[mx])mx=i;
    }
}

```

最小表示法

```

void Get_Min_Rep(char *s,int n){
    static char tmp[N];
#define S(x) s[(x)>n?(x)-n:(x)]
    int i=1,j=2,k=0;
    while (i<=n&&j<=n&&k<n){
        if (S(i+k)==S(j+k)) k++;
        else {
            (S(i+k)<S(j+k)?j:i)+=k+1;
            k=0;if (i==j)j++;
        }
    }
    int p=min(i,j);
    For(i,1,n)tmp[i]=S(p+i-1);
    For(i,1,n)s[i]=tmp[i];
    #undef S
}

```

图论

Dinic

```

struct Edge{
    int x,y,nxt,cap;
    Edge(){}
    Edge(int a,int b,int c,int d){
        x=a,y=b,cap=c,nxt=d;
    }
};

struct Network{
    static const int N=105,M=5005*2,INF=0x7FFFFFFF;
    Edge e[M],tmp[M];
    int n,S,T,fst[N],cur[N],cnt;
    int q[N],dis[N],head,tail;
    LL MaxFlow;
    void clear(int _n){
        n=_n,cnt=1;
        memset(fst,0,sizeof fst);
    }
    void add(int a,int b,int c){
        e[++cnt]=Edge(a,b,c,fst[a]),fst[a]=cnt;
        e[++cnt]=Edge(b,a,0,fst[b]),fst[b]=cnt;
    }
    void init(){
        for (int i=1;i<=n;i++)
            cur[i]=fst[i];
    }
    void init(int _S,int _T){
        S=_S,T=_T,MaxFlow=0,init();
    }
    int bfs(){
        memset(dis,0,sizeof dis);
        head=tail=0,q[++tail]=T,dis[T]=1;
        while (head<tail)
            for (int x=q[++head],i=fst[x];i;i=e[i].nxt)
                if (!dis[e[i].y]&&e[i^1].cap){
                    dis[q[++tail]]=e[i].y=dis[x]+1;
                    if (e[i].y==S)
                        return 1;
                }
        return (bool)dis[S];
    }
    int dfs(int x,int Flow){
        if (x==T||!Flow)
            return Flow;
        int now=Flow;
        for (int &i=cur[x];i;i=e[i].nxt){
            int y=e[i].y;
            if (dis[x]==dis[y]+1&&e[i].cap){
                int d=dfs(y,min(now,e[i].cap));
                e[i].cap-=d,e[i^1].cap+=d,now-=d;
                if (now==0)
                    break;
            }
        }
        return now;
    }
}

```

```

    }
    return Flow-now;
}
LL Dinic(){
    while (bfs())
        init(),MaxFlow+=dfs(S,INF);
    return MaxFlow;
}
LL Auto(int _S,int _T){
    init(_S,_T);
    return Dinic();
}
}g;
KM
const int N=405,INF=2e9;//UOJ#80
int n,m;
int g[N][N];
void Getg(){
    int c=read();
    while (c--){
        int x=read(),y=read(),z=read();
        g[x][y]=z;
    }
}
int match[N],visx[N],visy[N],ex[N],ey[N],Min[N];
int dfs(int x){
    visx[x]=1;
    for (int y=1;y<=m;y++)
        if (!visy[y]){
            int d=ex[x]+ey[y]-g[x][y];
            if (!d){
                visy[y]=1;
                if (!match[y]||dfs(match[y]))
                    return match[y]=x,1;
            }
            else Min[y]=min(Min[y],d);
        }
    return 0;
}
LL KM(){
    clr(match),clr(ex),clr(ey);
    for (int i=1;i<=n;i++)
        for (int j=1;j<=m;j++)
            ex[i]=max(ex[i],g[i][j]);
    for (int i=1;i<=n;i++){
        for (int j=1;j<=m;j++)Min[j]=INF;
        clr(visx),clr(visy);
        if (!dfs(i)){
            while (1){
                int d=INF,y=0;
                for (int j=1;j<=m;j++)
                    if (!visy[j])d=min(d,Min[j]);
                for (int j=1;j<=n;j++)if (visx[j])ex[j]-=d;
                for (int j=1;j<=m;j++)if (visy[j])ey[j]+=d;
                    else if (!(Min[j]-=d))y=j;
                if (!match[y])break;
                int x=match[y];visx[x]=visy[y]=1;
                for (int j=1;j<=m;j++)
                    Min[j]=min(Min[j],ex[x]+ey[j]-g[x][j]);
            }
            clr(visx),clr(visy);
        }
    }
    LL ans=0;
    for (int i=1;i<=n;i++)
        for (int j=1;j<=m;j++)
            ans+=ex[i];
    return ans;
}
int id[N];
int main(){
    n=read(),m=max(n,(int)read());
}

```

Templates For ZJUPC2019 --- by zhouchengdong

```
Getg();cout<<KM()<<endl;
for (int i=1;i<=m;i++)id[match[i]]=i;
for (int i=1;i<=n;i++)
    if (g[i][id[i]])cout<<id[i]<<" ";
    else cout<<0<<" ";
return 0;
}

SPFA 费用流
struct Edge{
    int x,y,c,nxt,cap;
    Edge(){}
    Edge(int a,int b,int _c,int d,int e){
        x=a,y=b,c=_c,cap=d,nxt=e;
    }
};

struct Network{
    static const int N=405,M=15005*2,INF=0x7FFFFFFF;
    Edge e[M];
    int n,S,T,fst[N],cur[N],cnt;
    int q[N],vis[N],head,tail;
    int MaxFlow,MinCost,dis[N];
    void clear(int _n){
        n=_n,cnt=1;memset(fst,0,sizeof fst);
    }
    void add(int a,int b,int c,int d){
        e[++cnt]=Edge(a,b,d,c,fst[a]),fst[a]=cnt;
        e[++cnt]=Edge(b,a,-d,0,fst[b]),fst[b]=cnt;
    }
    void init(){
        for (int i=1;i<=n;i++)cur[i]=fst[i];
    }
    void init(int _S,int _T){
        S=_S,T=_T,MaxFlow=MinCost=0,init();
    }
    int SPFA(){
        for (int i=1;i<=n;i++)dis[i]=INF;
        memset(vis,0,sizeof vis);
        head=tail=0;dis[q[++tail]=T]=0;
        while (head!=tail){
            if ((++head)>=n)head-=n;
            int x=q[head];vis[x]=0;
            for (int i=fst[x];i;i=e[i].nxt){
                int y=e[i].y;
                if (e[i^1].cap&&dis[x]-e[i].c<dis[y]){
                    dis[y]=dis[x]-e[i].c;
                    if (!vis[y]){
                        if ((++tail)>=n)tail-=n;
                        vis[q[tail]=y]=1;
                    }
                }
            }
            memset(vis,0,sizeof vis);
            return dis[S]<INF;
        }
        int dfs(int x,int Flow){
            if (x==T||!Flow)return Flow;
            vis[x]=1;int now=Flow;
            for (int &i=cur[x];i;i=e[i].nxt){
                int y=e[i].y;
                if (!vis[y]&&e[i].cap&&dis[x]-e[i].c==dis[y]){
                    int d=dfs(y,min(now,e[i].cap));
                    e[i].cap-=d,e[i^1].cap+=d;
                    if (!(now-=d))break;
                }
            }
            vis[x]=0;return Flow-now;
        }
        void Dinic(){
            while (SPFA()){
                init();int now=dfs(S,INF);
                MaxFlow+=now,MinCost+=now*dis[S];
            }
        }
        void MCMF(int &_MinCost,int &_MaxFlow){
            Dinic(),_MinCost=MinCost,_MaxFlow=MaxFlow;
        }
        void Auto(int _S,int _T,int &_MinCost,int &_MaxFlow){
            init(_S,_T),MCMF(_MinCost,_MaxFlow);
        }
    }
}g;
```

上下界网络流

```
struct LU_Network{
    static const int N=50010;
    static const LL INF=1LL<<50;
    Network g;
    int n,S,T;
    LL in[N],Sum;
    void clear(int _n){
        memset(in,0,sizeof in);
        n=_n,S=n+1,T=n+2,Sum=0,g.clear(T);
    }
    void add(int x,int y,int L,int U){
        g.add(x,y,U-L),in[x]-=L,in[y]+=L;
    }
    void build(){
        for (int i=1;i<=n;i++)
            if (in[i]>0)
                g.add(S,i,in[i]),Sum+=in[i];
            else if (in[i]<0)
                g.add(i,T,-in[i]);
    }
    bool CanFlow(){
        build();
        return g.Auto(S,T)>=Sum;
    }
    int cur,c1;
    bool CanFlow(int s,int t){
        build();
        c1=g.cnt;
        g.add(t,s,INF>>1);
        cur=g.cnt;
        return g.Auto(S,T)>=Sum;
    }
    LL MaxFlow(int s,int t){
        if (!CanFlow(s,t))
            return -1;
        return g.Auto(s,t);
    }
    LL MinFlow(int s,int t){
        if (!CanFlow(s,t))
            return -1;
        LL now=g.e[g.cnt].cap;
        g.e[g.cnt].cap=g.e[g.cnt^1].cap=0;
        return now-g.Auto(t,s);
    }
}g;
```

多项式

FFT

```
const int N=1<<20;const double PI=acos(-1.0);
struct C{
    double r,i;
    C(){}
    C(double a,double b){r=a,i=b;}
    C operator + (C x){return C(r+x.r,i+x.i);}
    C operator - (C x){return C(r-x.r,i-x.i);}
    C operator * (C x){return C(r*x.r-i*x.i,r*x.i+i*x.r);}
}a[N],b[N],w[N];
int A,B,n,L,R[N];
void FFT(C a[],int n){
    for (int i=0;i<n;i++)
        if (R[i]>i)swap(a[R[i]],a[i]);
    for (int t=n>>1,d=1;d<n;d<=1,t>>=1)
        for (int i=0;i<n;i+=(d<<1))
            for (int j=0;j<d;j++){
                C tmp=w[t*j]*a[i+j+d];
                a[i+j+d]=a[i+j]-tmp;
                a[i+j]=a[i+j]+tmp;
            }
    }//double Eps : 1e-14~1e-15
int main(){
    for (n=1,L=0;n<=A+B;n<=1,L++);
}
```

Templates For ZJUPC2019 --- by zhoushendong

```
for (int i=0;i<n;i++){  
    R[i]=(R[i]>>1)>>1)|((i&1)<<(L-1));  
    w[i]=C(cos(2.0*i*PI/n),sin(2.0*i*PI/n));  
}  
FFT(a,n),FFT(b,n);  
for (int i=0;i<n;i++)  
    a[i]=a[i]*b[i],w[i].i=-w[i].i;  
FFT(a,n);  
for (int i=0;i<n;i++)a[i].r/=n;  
}  
  
// Mul for mod 1e9+7  
C a0[N],a1[N],b0[N],b1[N],A[N],B[N],C[N];  
vector <int> Mul(vector <int> a,vector <int> b){  
    static vector <int> ans;  
    ans.clear();  
    int n,d;  
    for (n=1,d=0;n<a.size()+b.size();n<<=1,d++);  
    for (int i=0;i<n;i++){  
        R[i]=(R[i]>>1)>>1)|((i&1)<<(d-1));  
        w[i]=Comp(cos(pi*2/n*i),sin(pi*2/n*i));  
    }  
    for (int i=0;i<n;i++)  
        a0[i]=a1[i]=b0[i]=b1[i]=Comp(0,0);  
    for (int i=0;i<a.size();i++)  
        a0[i].r=a[i]&32767,a1[i].r=a[i]>>15;  
    for (int i=0;i<b.size();i++)  
        b0[i].r=b[i]&32767,b1[i].r=b[i]>>15;  
    FFT(a0,n),FFT(a1,n),FFT(b0,n),FFT(b1,n);  
    for (int i=0;i<n;i++){  
        A[i]=a0[i]*b0[i];  
        B[i]=a1[i]*b1[i];  
        C[i]=a0[i]*b1[i]+a1[i]*b0[i];  
        w[i].i=-w[i].i;  
    }  
    FFT(A,n),FFT(B,n),FFT(C,n);  
    for (int i=0;i<n;i++){  
        int v=0;  
        Add(v,(LL)(A[i].r/n+0.5)%mod);  
        Add(v,(((LL)(B[i].r/n+0.5)%mod)<<30)%mod);  
        Add(v,(((LL)(C[i].r/n+0.5)%mod)<<15)%mod);  
        ans.pb(v);  
    }  
    while (!ans.empty()&&!ans.back())  
        ans.pop_back();  
    return ans;  
}  
Poly  
Poly operator / (Poly A,Poly B){//Divide  
int n=A.size(),m=B.size();  
reverse(A.v.begin(),A.v.end());  
reverse(B.v.begin(),B.v.end());  
int k=n-m+1;  
if (k<0)  
    return Poly(0);  
while (A.size()>k)A.pop_back();  
while (B.size()>k)B.pop_back();  
A=A*Inverse(B,k);  
while (A.size()>k)A.pop_back();  
reverse(A.v.begin(),A.v.end());  
return A;  
}  
Poly operator % (Poly A,Poly B){//Modulo  
while (!A.empty()&&!A.back())A.pop_back();  
while (!B.empty()&&!B.back())B.pop_back();  
A=A-A/B*B;  
while (A.size()>=B.size())A.pop_back();  
while (!A.empty()&&!A.back())A.pop_back();  
return A;  
}  
Poly Inverse(Poly a,int n){  
    static Poly A,B;  
    while (!a.empty()&&!a.back())a.pop_back();  
    if (a.empty())return a;  
    A.clear(),B.clear();  
    B.push_back(a[0]);  
    A.push_back(Pow(B[0],mod-2));  
    for (int t=1;t<n;){  
        for (int i=t;i<min(a.size(),(t<<1));i++)  
            B.push_back(a[i]);  
        t<<=1;A=A*2-Poly(Fast :: MulInv(A.v,B.v));  
        while (A.size()>t)A.pop_back();  
    }  
    while (A.size()>n)A.pop_back();  
    return A;  
}  
Poly Sqrt(Poly a,int n){  
    static Poly A,B;  
    while (!a.empty()&&!a.back())a.pop_back();  
    if (a.empty())return a;  
    A.clear(),B.clear();  
    B.push_back(a[0]);  
    A.push_back(Rem2 :: Sqrt(B[0]));  
    for (int t=1;t<n;){  
        for (int i=t;i<min(a.size(),(t<<1));i++)  
            B.push_back(a[i]);  
        t<<=1;A+=B*Inverse(A,t);  
        while (A.size()>t)A.pop_back();  
        A*=499122177;  
    }  
    if (A[0]>mod-A[0])  
        for (int i=0;i<A.size();i++)  
            A[i]=(mod-A[i])%mod;  
    while (A.size()>n)A.pop_back();  
    return A;  
}  
Poly Ln(Poly a,int n){  
    while (!a.empty()&&!a.back())a.pop_back();  
    if (a.empty()||a[0]!=1)return a;  
    a=Integral(Derivation(a)*Inverse(a,n));  
    while (a.size()>n)a.pop_back();  
    return a;  
}  
Poly Exp(Poly a,int n){  
    static Poly A,B;  
    while (!a.empty()&&!a.back())a.pop_back();  
    if (a.empty())return Poly(1);  
    if (a[0]!=0)return a;  
    A.clear(),B.clear();  
    B.push_back(1);A.push_back(a[0]);  
    for (int t=1;t<n;){  
        for (int i=t;i<min(a.size(),(t<<1));i++)  
            A.push_back(a[i]);  
        t<<=1;B=B*(Poly(1)+A-Ln(B,t));  
        while (B.size()>t)B.pop_back();  
    }  
    while (B.size()>n)B.pop_back();  
    return B;  
}  
Poly PolyPow(Poly x,int y,int n){  
    static Poly A,B;  
    int k0=0,kc,ivkc;  
    while (!x.empty()&&!x.back())x.pop_back();  
    if (x.empty())return x;  
    while (k0<x.size()&&x[k0]==0)k0++;  
    kc=x[k0],ivkc=Pow(kc,mod-2);A.clear();  
    for (int i=k0;i<x.size();i++)  
        A.push_back((int)((LL)x[i]*ivkc%mod));  
    A=Exp(Ln(A,n)*y,n);B.clear();  
    if ((LL)k0*y>=n)return B;  
    kc=Pow(kc,y),k0*=y;  
    for (int i=0;i<k0;i++)B.push_back(0);  
    for (int i=0;i<min(A.size(),n-k0);i++)  
        B.push_back((int)((LL)A[i]*kc%mod));
```

```

    while (B.size()>n)B.pop_back();
    return B;
}

namespace Qiuzhi{
    Poly P[N<<2],f[N<<2],M;
    vector <int> x,y,int n;
    void GetP(int rt,int L,int R){
        if (L==R){
            P[rt].clear();
            P[rt].push_back((mod-x[L])%mod);
            P[rt].push_back(1);
            return;
        }
        int mid=(L+R)>>1,ls=rt<<1,rs=ls|1;
        GetP(ls,L,mid);GetP(rs,mid+1,R);
        P[rt]=P[ls]*P[rs];
    }
    void qiuzhi(int rt,int L,int R){
        if (f[rt].empty())f[rt].push_back(0);
        if (L==R)return (void)(y[L]=f[rt][0]);
        int mid=(L+R)>>1,ls=rt<<1,rs=ls|1;
        f[ls]=f[rt]%P[ls];f[rs]=f[rt]%P[rs];
        qiuzhi(ls,L,mid);qiuzhi(rs,mid+1,R);
    }
    vector <int> Get_Val(vector <int> A,Poly F){
        n=A.size();x.clear(),y.clear();
        for (int i=0;i<n;i++)
            x.push_back(A[i]),y.push_back(0);
        GetP(1,0,n-1);f[1]=F;
        qiuzhi(1,0,n-1);
        return y;
    }
}
namespace Chazhi{
    Poly P[N<<2],M;
    vector <int> x,y,int n;
    void GetP(int rt,int L,int R){
        if (L==R){
            P[rt].clear();
            P[rt].push_back((mod-x[L])%mod);
            P[rt].push_back(1);
            return;
        }
        int mid=(L+R)>>1,ls=rt<<1,rs=ls|1;
        GetP(ls,L,mid);GetP(rs,mid+1,R);
        P[rt]=P[ls]*P[rs];
    }
    Poly chazhi(int rt,int L,int R){
        if (L==R)return Poly(y[L]);
        int mid=(L+R)>>1,ls=rt<<1,rs=ls|1;
        return
chazhi(ls,L,mid)*P[rs]+chazhi(rs,mid+1,R)*P[ls];
    }
    Poly Get_Poly(vector <int> A,vector <int> B){
        n=A.size();x=A;int Product=1;
        GetP(1,0,n-1);M=Derivation(P[1]);
        y=Qiuzhi :: Get_Val(A,M);
        for (int i=0;i<y.size();i++)
            y[i]=(LL)B[i]*Pow(y[i],mod-2)%mod;
        return chazhi(1,0,n-1);
    }
}

```

第一类斯特林数

```

namespace str{
    vector <int> rpow[22];
    //attention: rpow[i].size() = pow(2, n) + 1
    //using: Fac[i] = i!, Inv[i] = 1/Fac[i]; mxd = pow(2,
n) + 1
    vector <int> Get_Add(vector <int> f,int n,int v){
        vector <int> g,h;g.clear();
        for (int i=-n;i<=0;i++)
            g.pb((LL)Pow(v,-i)*Inv[-i]%mod);

```

```

        for (int i=0;i<=n;i++)
            f[i]=(LL)f[i]*Fac[i]%mod;
        h=poly::Mul(f,g);
        while (h.size()<n*2+1)h.pb(0);
        g.clear();
        for (int i=0;i<=n;i++)
            g.pb((LL)h[i+n]*Inv[i]%mod);
        return g;
    }
    vector <int> Get_rpow(int x){
        if (!rpow[x].empty())return rpow[x];
        if (x==0){
            rpow[x].pb(0),rpow[x].pb(1);
            return rpow[x];
        }
        int n=1<<(x-1);
        vector <int> f=Get_rpow(x-1);
        rpow[x]=poly::Mul(f,Get_Add(f,n,n));
        while (rpow[x].size()>n*2+1)
            rpow[x].pop_back();
        while (rpow[x].size()<n*2+1)
            rpow[x].pb(0);
        return rpow[x];
    }
    void Get_s1(int n,int res[]){
        static vector <int> ans;
        if (n==0)return (void)(res[0]=1);
        ans.clear();ans.pb(1);
        for (int i=0;i<20;i++)
            if (n>>i&1)
                ans=poly::Mul(Get_rpow(i),
Get_Add(ans,(int)ans.size()-1,1<<i));
        for (int i=0;i<=n;i++)res[i]=ans[i];
    }
}

```

杂项

Fread

```

namespace IO{
    const int Len=1<<21;
    char Ibuf[Len+1],*Is=Ibuf,*It=Ibuf;
    char gc(){
        if (Is==It){
            It=(Is=Ibuf)+fread(Ibuf,1,Len,stdin);
            if (Is==It)
                return EOF;
        }
        return *Is++;
    }
    char Obuf[Len+1],*Ot=Obuf;
    void pc(char ch){
        if (Ot==Obuf+Len)
            fwrite(Obuf,1,Len,stdout),Ot=Obuf;
        *Ot++=ch;
    }
    void flush(){
        fwrite(Obuf,1,Ot-Obuf,stdout);
        Ot=Obuf;
    }
}

```

DLX

```

const int N=100+5,M=100+5,S=N*M;
struct DLX{
    int n,m,cnt;
    int x[S],y[S],L[S],R[S],U[S],D[S];
    int C[M],anscnt,ans[N];
    void init(int c){
        memset(x,0,sizeof x),memset(y,0,sizeof y);
        memset(L,0,sizeof L),memset(R,0,sizeof R);
        memset(U,0,sizeof U),memset(D,0,sizeof D);

```

```

    memset(C,0,sizeof C),memset(ans,0,sizeof ans);
    ansCnt=0,m=c;
    for (int i=0;i<=m;i++)
        L[i]=i-1,R[i]=i+1,U[i]=D[i]=i;
    L[0]=m,R[m]=0,cnt=m;
}
void link(int i,int j){
    cnt++;x[cnt]=i;y[cnt]=j;
    L[cnt]=cnt-1;R[cnt]=cnt+1;
    D[cnt]=j;D[U[j]]=cnt;
    U[cnt]=U[j];U[j]=cnt;C[j]++;
}
void Delete(int k){
    L[R[k]]=L[k];R[L[k]]=R[k];
    for (int i=D[k];i!=k;i=D[i])
        for (int j=R[i];j!=i;j=R[j]){
            U[D[j]]=U[j];D[U[j]]=D[j];C[y[j]]--;
        }
}
void Reset(int k){
    L[R[k]]=k;R[L[k]]=k;
    for (int i=U[k];i!=k;i=U[i])
        for (int j=L[i];j!=i;j=L[j]){
            U[D[j]]=j;D[U[j]]=j;C[y[j]]++;
        }
}
bool solve(){
    if (R[0]==0) return true;
    ansCnt++;int k=R[0];
    for (int i=R[k];i!=0;i=R[i])
        if (C[i]<C[k]) k=i;
    Delete(k);
    for (int i=D[k];i!=k;i=D[i]){
        ans[ansCnt]=x[i];
        for (int j=R[i];j!=i;j=R[j])
            Delete(y[j]);
        if (solve()) return true;
        for (int j=L[i];j!=i;j=L[j])
            Reset(y[j]);
    }
    Reset(k);ansCnt--;
    return false;
}
}dlx;

```

```

Hash_List
struct hash_map{
    static const int Ti=233,mod=1<<16;
    int cnt,k[mod+1],v[mod+1],nxt[mod+1],fst[mod+1];
    int Hash(int x){int v=x&(mod-1);return v==0?mod:v;}
    void clear(){cnt=0;memset(fst,0,sizeof fst);}
    void update(int x,int a){
        int y=Hash(x);
        for (int p=fst[y];p;p=nxt[p])
            if (k[p]==x)
                return (void)(v[p]=a);
        k[++cnt]=x,nxt[cnt]=fst[y],fst[y]=cnt,v[cnt]=a;
    }
    int find(int x){
        int y=Hash(x);
        for (int p=fst[y];p;p=nxt[p])
            if (k[p]==x)
                return v[p];
        return 0;
    }
    int &operator [] (int x){
        int y=Hash(x);
        for (int p=fst[y];p;p=nxt[p])
            if (k[p]==x)
                return v[p];
        k[++cnt]=x,nxt[cnt]=fst[y],fst[y]=cnt;
        return v[cnt]=0;
    }
}

```

}

}Map;

全局平衡二叉树

```

int build_bin(int pre,int L,int R){
    if (L>R)
        return 0;
    int x=L,mid=L+1,r=R,ckv=(sz[L-1]+sz[R]+1)>>1;
    while (l<=r){
        mid=(l+r)>>1;
        if (sz[mid]<=ckv)
            l=mid+1,x=mid;
        else
            r=mid-1;
    }
    mid=x,fa[x=tmp[x]]=pre;
    Son[x][0]=build_bin(x,L,mid-1);
    Son[x][1]=build_bin(x,mid+1,R);
    update(x);
    return x;
}
int build(int pre,int x){
    for (int i=x;i;i=son[i])
        for (auto y : e[i])
            if (y!=son[i]&&y!=fa[i])
                build(i,y);
    int n=tmp[0]=sz[0]=0;
    for (int i=x;i;i=son[i]){
        tmp[++n]=i;
        sz[n]=sz[n-1]+size[i]-size[son[i]];
    }
    return build_bin(pre,1,n);
}

```

最小圆覆盖

```

Point Rotate(Point A,double B){
    return
    Point(A.x*cos(B)-A.y*sin(B),A.x*sin(B)+A.y*cos(B));
}
Point Center(Point A,Point B,Point C){
    Point a=(A+B)/2,b=(A+C)/2;
    Point u=Rotate(B-A,pi/2),v=Rotate(C-A,pi/2);
    if (Dcmp(cross(u,v))==0){
        if (Dcmp(Dis(A,C),Dis(A,B)+Dis(B,C))==0) return
        (A+C)/2;
        if (Dcmp(Dis(A,B),Dis(A,C)+Dis(B,C))==0) return
        (A+B)/2;
        if (Dcmp(Dis(B,C),Dis(A,B)+Dis(A,C))==0) return
        (B+C)/2;
    }
    return a+u*cross(v,a-b)/cross(u,v);
}
int n;
int main(){
    srand(233);n=read();
    for (int i=1;i<=n;i++)
        p[i].x=read(),p[i].y=read();
    Point c=p[1];double r=0;
    random_shuffle(p+1,p+n+1);
    for (int i=2;i<=n;i++){
        if (Dcmp(Dis(p[i],c),r)<=0) continue;
        c=p[i],r=0;
        for (int j=1;j<i;j++){
            if (Dcmp(Dis(p[j],c),r)<=0) continue;
            c=(p[i]+p[j])/2,r=Dis(p[i],c);
            for (int k=1;k<j;k++){
                if (Dcmp(Dis(p[k],c),r)<=0) continue;
                c=Center(p[i],p[j],p[k]);
                r=Dis(p[i],c);
            }
        }
        printf("%.6lf %.6lf %.6lf\n",r,c.x,c.y);
        return 0;
    }
}

```