Spear of Longinus

Shanghai Jiao Tong University

Contents

数论

扩展欧几里德算法

```
LL exgcd(LL a,LL b,LL &x,LL &y){
    if(!b){
        x=1;y=0;return a;
    }else{
        LL d=exgcd(b,a%b,x,y);
        LL t=x;x=y;y=t-a/b*y;
        return d;
    }
}
```

中国剩余定理

```
LL china(int n,int *a,int *m){
    LL M=1,d,x=0,y;
    for(int i=0;i<n;i++)
        M*=m[i];
    for(int i=0;i<n;i++){
        LL w=M/m[i];
        d=exgcd(m[i],w,d,y);
        y=(y%M+M)%M;
        x=(x+y*w%M*a[i])%M;
    }
    while(x<0)x+=M;
    return x;
}
```

中国剩余定理 2

```
//merge Ax=B and ax=b to A'x=B'
void merge(LL &A,LL &B,LL a,LL b){
    LL x,y;
    sol(A,-a,b-B,x,y);
    A=lcm(A,a);
    B=(a*y+b)%A;
    B=(B+A)%A;
}
```

扩展小步大步

```
LL solve2(LL a,LL b,LL p){
    //a^x=b (mod p)
    b%=p;
    LL e=1%p;
    for(int i=0;i<100;i++){
        if(e==b)return i;
        e=e*a%p;
    }
    int r=0;
    while(gcd(a,p)!=1){
        LL d=gcd(a,p);
        if(b%d)return -1;
        p/=d;b/=d;b=b*inv(a/d,p);
        r++;
    }LL res=BSGS(a,b,p);
    if(res==-1)return -1;
    return res+r;
}
```

卢卡斯定理

```
LL Lucas(LL n,LL m,LL p){
    LL ans=1;
    while(n&&m){
        LL a=n%p,b=m%p;
        if(a<b)return 0;
        ans=(ans*C(a,b,p))%p;
        n/=p;m/=p;
    }return ans%p;
}
```

Miller Rabin 素数测试

```
const int BASE[12] = {2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37};
bool check(long long n,int base) {
    long long n2=n-1,res;
    int s=0;
    while(n2%2==0) n2>>=1,s++;
    res=pw(base,n2,n);
    if((res==1)||(res==n-1)) return 1;
    while(s--) {
        res=mul(res,res,n);
        if(res==n-1) return 1;
    }
    return 0; // n is not a strong pseudo prime
}
bool isprime(const long long &n) {
    if(n==2)
        return true;
    if(n<2 || n%2==0)
        return false;
    for(int i=0;i<12&&BASE[i]<n;i++){
        if(!check(n,BASE[i]))
            return false;
    }
    return true;
}
```

Pollard Rho 大数分解

时间复杂度：$\mathcal{O}(n^{1/4})$

```
LL prho(LL n,LL c){
    LL i=1,k=2,x=rand()%(n-1)+1,y=x;
    while(1){
        i++;x=(x*x%n+c)%n;
        LL d=__gcd((y-x+n)%n,n);
        if(d>1&&d<n)return d;
        if(y==x)return n;
        if(i==k)y=x,k<<=1;
    }
}
void factor(LL n,vector<LL>&fat){
    if(n==1)return;
    if(isprime(n)){
        fat.push_back(n);
        return;
    }LL p=n;
    while(p>=n)p=prho(p,rand()%(n-1)+1);
    factor(p,fat);
    factor(n/p,fat);
}
```

快速数论变换 (zky)

返回结果：

$$c_i = \sum_{0\le j\le i} a_j \cdot b_{i-j}(mod)\ (0 \le i < n)$$

使用说明：$magic$ 是 mod 的原根

时间复杂度：$\mathcal{O}(nlogn)$

```
/*
{(mod,G)}={(81788929,7),(101711873,3),(167772161,3)
          ,(377487361,7),(998244353,3),(1224736769,3)
          ,(1300234241,3),(1484783617,5)}
*/
int mo=998244353,G=3;
void NTT(int a[],int n,int f){
    for(register int i=0;i<n;i++)
        if(i<rev[i])
            swap(a[i],a[rev[i]]);
    for (register int i=2;i<=n;i<<=1){
        static int exp[maxn];
        exp[0]=1;exp[1]=pw(G,(mo-1)/i);
        if(f==-1)exp[1]=pw(exp[1],mo-2);
        for(register int k=2;k<(i>>1);k++)
            exp[k]=1LL*exp[k-1]*exp[1]%mo;
        for(register int j=0;j<n;j+=i){
            for(register int k=0;k<(i>>1);k++){
                register int &pA=a[j+k],&pB=a[j+k+(i>>1)];
                register int A=pA,B=1LL*pB*exp[k]%mo;
                pA=(A+B)%mo;
                pB=(A-B+mo)%mo;
            }
        }
    }
    if(f==-1){
        int rv=pw(n,mo-2)%mo;
        for(int i=0;i<n;i++)
            a[i]=1LL*a[i]*rv%mo;
    }
}
void mul(int m,int a[],int b[],int c[]){
    int n=1,len=0;
    while(n<m)n<<=1,len++;
    for (int i=1;i<n;i++)
        rev[i]=(rev[i>>1]>>1)|((i&1)<<(len-1));
    NTT(a,n,1);
    NTT(b,n,1);
    for(int i=0;i<n;i++)
        c[i]=1LL*a[i]*b[i]%mo;
    NTT(c,n,-1);
}
```

原根

```
vector<LL>fct;
bool check(LL x,LL g){
    for(int i=0;i<fct.size();i++)
        if(pw(g,(x-1)/fct[i],x)==1)
            return 0;
    return 1;
}
LL findrt(LL x){
    LL tmp=x-1;
    for(int i=2;i*i<=tmp;i++){
        if(tmp%i==0){
            fct.push_back(i);
            while(tmp%i==0)tmp/=i;
        }
    }if(tmp>1)fct.push_back(tmp);
    // x is 1,2,4,p^n,2p^n
    // x has phi(phi(x)) primitive roots
    for(int i=2;i<int(1e9);i++)if(check(x,i))
        return i;
    return -1;
}
const int BASE[12] = {2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37};
bool check(long long n,int base) {
    long long n2=n-1,res;
    int s=0;
    while(n2%2==0) n2>>=1,s++;
    res=pw(base,n2,n);
    if((res==1)||(res==n-1)) return 1;
    while(s--) {
        res=mul(res,res,n);
        if(res==n-1) return 1;
    }
    return 0; // n is not a strong pseudo prime
}
bool isprime(const long long &n) {
    if(n==2)
        return true;
    if(n<2 || n%2==0)
        return false;
    for(int i=0;i<12&&BASE[i]<n;i++){
        if(!check(n,BASE[i]))
            return false;
    }
    return true;
}
```

线性递推

```
//已知 a_0, a_1, ..., a_{m-1}\\
    a_n = c_0 * a_{n-m} + ... + c_{m-1} * a_{n-1}\\
    求 a_n = v_0 * a_0 + v_1 * a_1 + ... + v_{m-1} * a_{m-1}\\

void linear_recurrence(long long n, int m, int a[], int c[], int p) {
    long long v[M] = {1 % p}, u[M << 1], msk = !!n;
    for(long long i(n); i > 1; i >>= 1) {
        msk <<= 1;
    }
    for(long long x(0); msk; msk >>= 1, x <<= 1) {
        fill_n(u, m << 1, 0);
        int b(!!(n & msk));
        x |= b;
        if(x < m) {
            u[x] = 1 % p;
        }else {
            for(int i(0); i < m; i++) {
                for(int j(0), t(i + b); j < m; j++, t++) {
                    u[t] = (u[t] + v[i] * v[j]) % p;
                }
            }
            for(int i((m << 1) - 1); i >= m; i--) {
                for(int j(0), t(i - m); j < m; j++, t++) {
                    u[t] = (u[t] + c[j] * u[i]) % p;
                }
            }
        }
        copy(u, u + m, v);
    }
    //a[n] = v[0] * a[0] + v[1] * a[1] + ... + v[m - 1] * a[m - 1].
    for(int i(m); i < 2 * m; i++) {
        a[i] = 0;
        for(int j(0); j < m; j++) {
            a[i] = (a[i] + (long long)c[j] * a[i + j - m]) % p;
        }
    }
    for(int j(0); j < m; j++) {
        b[j] = 0;
        for(int i(0); i < m; i++) {
            b[j] = (b[j] + v[i] * a[i + j]) % p;
        }
```

```
    }
    for(int j(0); j < m; j++) {
        a[j] = b[j];
    }
}
```

直线下整点个数

返回结果：

$$\sum_{0\le i<n}\lfloor\frac{a+b\cdot i}{m}\rfloor$$

```
//calc \sum_{i=0}^{n-1} [(a+bi)/m]
// n,a,b,m >0
LL solve(LL n,LL a,LL b,LL m){
    if(b==0)
        return n*(a/m);
    if(a>=m || b>=m)
        return n*(a/m)+(n-1)*n/2*(b/m)+solve(n,a%m,b%m,m);
    return solve((a+b*n)/m,(a+b*n)%m,m,b);
}
```

数值

高斯消元

```
void Gauss(){
    int r,k;
    for(int i=0;i<n;i++){
        r=i;
        for(int j=i+1;j<n;j++)
            if(fabs(A[j][i])>fabs(A[r][i]))r=j;
        if(r!=i)for(int j=0;j<=n;j++)swap(A[i][j],A[r][j]);
        for(int k=i+1;k<n;k++){
            double f=A[k][i]/A[i][i];
            for(int j=i;j<=n;j++)A[k][j]-=f*A[i][j];
        }
    }
    for(int i=n-1;i>=0;i--){
        for(int j=i+1;j<n;j++)
            A[i][n]-=A[j][n]*A[i][j];
        A[i][n]/=A[i][i];
    }
    for(int i=0;i<n-1;i++)
        cout<<fixed<<setprecision(3)<<A[i][n]<<" ";
    cout<<fixed<<setprecision(3)<<A[n-1][n];
}
bool Gauss(){
    for(int i=1;i<=n;i++){
        int r=0;
        for(int j=i;j<=m;j++)
        if(a[j][i]){r=j;break;}
        if(!r)return 0;
        ans=max(ans,r);
        swap(a[i],a[r]);
        for(int j=i+1;j<=m;j++)
        if(a[j][i])a[j]^=a[i];
    }for(int i=n;i>=1;i--){
        for(int j=i+1;j<=n;j++)if(a[i][j])
        a[i][n+1]=a[i][n+1]^a[j][n+1];
    }return 1;
}
LL Gauss(){
    for(int i=0;i<n;i++)for(int j=0;j<n;j++)A[i][j]%=m;
    for(int i=0;i<n;i++)for(int j=0;j<n;j++)A[i][j]=(A[i][j]+m)%m;
    LL ans=n%2?-1:1;
    for(int i=0;i<n;i++){
        for(int j=i+1;j<n;j++){
            while(A[j][i]){
                LL t=A[i][i]/A[j][i];
                for(int k=0;k<n;k++)
                A[i][k]=(A[i][k]-A[j][k]*t%m+m)%m;
                swap(A[i],A[j]);
                ans=-ans;
            }
        }ans=ans*A[i][i]%m;
    }return (ans%m+m)%m;
}
int Gauss(){//求秩
    int r,now=-1;
    int ans=0;
    for(int i = 0; i <n; i++){
        r = now + 1;
        for(int j = now + 1; j < m; j++)
            if(fabs(A[j][i]) > fabs(A[r][i]))
                r = j;
        if (!sgn(A[r][i])) continue;
        ans++;
        now++;
        if(r != now)
            for(int j = 0; j < n; j++)
                swap(A[r][j], A[now][j]);

        for(int k = now + 1; k < m; k++){
            double t = A[k][i] / A[now][i];
            for(int j = 0; j < n; j++){
                A[k][j] -= t * A[now][j];
            }
        }
    }
    return ans;
}
```

快速傅立叶变换

返回结果：

$$c_i=\sum_{0\le j\le i}a_j\cdot b_{i-j}\ (0\le i<n)$$

时间复杂度：$\mathcal{O}(nlogn)$

```
typedef complex<double> cp;
const double pi = acos(-1);
void FFT(vector<cp>&num,int len,int ty){
    for(int i=1,j=0;i<len-1;i++){
        for(int k=len;j^=k>>=1,~j&k;);
        if(i<j)
            swap(num[i],num[j]);
    }
    for(int h=0;(1<<h)<len;h++){
        int step=1<<h,step2=step<<1;
        cp w0(cos(2.0*pi/step2),ty*sin(2.0*pi/step2));
        for(int i=0;i<len;i+=step2){
            cp w(1,0);
            for(int j=0;j<step;j++){
                cp &x=num[i+j+step];
                cp &y=num[i+j];
                cp d=w*x;
                x=y-d;
                y=y+d;
                w=w*w0;
            }
        }
    }
```

```
    if(ty==-1)
        for(int i=0;i<len;i++)
            num[i]=cp(num[i].real()/(double)len,num[i].imag());
}
vector<cp> mul(vector<cp>a,vector<cp>b){
    int len=a.size()+b.size();
    while((len&-len)!=len)len++;
    while(a.size()<len)a.push_back(cp(0,0));
    while(b.size()<len)b.push_back(cp(0,0));
    FFT(a,len,1);
    FFT(b,len,1);
    vector<cp>ans(len);
    for(int i=0;i<len;i++)
        ans[i]=a[i]*b[i];
    FFT(ans,len,-1);
    return ans;
}
```

单纯形法求解线性规划

返回结果：

$$max\{c_{1\times m}\cdot x_{m\times 1} \mid x_{m\times 1}\geq 0_{m\times 1}, a_{n\times m}\cdot x_{m\times 1}\leq b_{n\times 1}\}$$

```
namespace LP{
    const int maxn=233;
    double a[maxn][maxn];
    int Ans[maxn],pt[maxn];
    int n,m;
    void pivot(int l,int i){
        double t;
        swap(Ans[l+n],Ans[i]);
        t=-a[l][i];
        a[l][i]=-1;
        for(int j=0;j<=n;j++)a[l][j]/=t;
        for(int j=0;j<=m;j++){
            if(a[j][i]&&j!=l){
                t=a[j][i];
                a[j][i]=0;
                for(int k=0;k<=n;k++)a[j][k]+=t*a[l][k];
            }
        }
    }
    vector<double> solve(vector<vector<double>
      ↪ >A,vector<double>B,vector<double>C){
        n=C.size();
        m=B.size();
        for(int i=0;i<C.size();i++)
            a[0][i+1]=C[i];
        for(int i=0;i<B.size();i++)
            a[i+1][0]=B[i];

        for(int i=0;i<m;i++)
            for(int j=0;j<n;j++)
                a[i+1][j+1]=-A[i][j];

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

        double t;
        for(;;){
            int l=0;t=-eps;
            for(int j=1;j<=m;j++)if(a[j][0]<t)t=a[l=j][0];
            if(!l)break;
            int i=0;
            for(int j=1;j<=n;j++)if(a[l][j]>eps){i=j;break;}
            if(!i){
                puts("Infeasible");
                return vector<double>();
            }
            pivot(l,i);
        }
        for(;;){
            int i=0;t=eps;
            for(int j=1;j<=n;j++)if(a[0][j]>t)t=a[0][i=j];
            if(!i)break;
            int l=0;
            t=1e30;
            for(int j=1;j<=m;j++)if(a[j][i]<-eps){
                double tmp;
                tmp=-a[j][0]/a[j][i];
                if(t>tmp)t=tmp,l=j;
            }
            if(!l){
                puts("Unbounded");
                return vector<double>();
            }
            pivot(l,i);
        }
        vector<double>x;
        for(int i=n+1;i<=n+m;i++)pt[Ans[i]]=i-n;
        for(int i=1;i<=n;i++)x.push_back(pt[i]?a[pt[i]][0]:0);
        return x;
    }
}
```

自适应辛普森

```
double area(const double &left, const double &right) {
    double mid = (left + right) / 2;
    return (right - left) * (calc(left) + 4 * calc(mid) + calc(right)) / 6;
}

double simpson(const double &left, const double &right,
               const double &eps, const double &area_sum) {
    double mid = (left + right) / 2;
    double area_left = area(left, mid);
    double area_right = area(mid, right);
    double area_total = area_left + area_right;
    if (std::abs(area_total - area_sum) < 15 * eps) {
        return area_total + (area_total - area_sum) / 15;
    }
    return simpson(left, mid, eps / 2, area_left)
         + simpson(mid, right, eps / 2, area_right);
}

double simpson(const double &left, const double &right, const double &eps) {
    return simpson(left, right, eps, area(left, right));
}
```

多项式求根

```
const double eps=1e-12;
double a[10][10];
typedef vector<double> vd;
int sgn(double x) { return x < -eps ? -1 : x > eps; }
double mypow(double x,int num){
    double ans=1.0;
    for(int i=1;i<=num;++i)ans*=x;
    return ans;
}
double f(int n,double x){
    double ans=0;
    for(int i=n;i>=0;--i)ans+=a[n][i]*mypow(x,i);
    return ans;
}
double getRoot(int n,double l,double r){
```

```
    if(sgn(f(n,l))==0)return l;
    if(sgn(f(n,r))==0)return r;
    double temp;
    if(sgn(f(n,l))>0)temp=-1;else temp=1;
    double m;
    for(int i=1;i<=10000;++i){
        m=(l+r)/2;
        double mid=f(n,m);
        if(sgn(mid)==0){
            return m;
        }
        if(mid*temp<0)l=m;else r=m;
    }
    return (l+r)/2;
}
vd did(int n){
    vd ret;
    if(n==1){
        ret.push_back(-1e10);
        ret.push_back(-a[n][0]/a[n][1]);
        ret.push_back(1e10);
        return ret;
    }
    vd mid=did(n-1);
    ret.push_back(-1e10);
    for(int i=0;i+1<mid.size();++i){
        int t1=sgn(f(n,mid[i])),t2=sgn(f(n,mid[i+1]));
        if(t1*t2>0)continue;
        ret.push_back(getRoot(n,mid[i],mid[i+1]));
    }
    ret.push_back(1e10);
    return ret;
}
int main(){
    int n; scanf("%d",&n);
    for(int i=n;i>=0;--i){
        scanf("%lf",&a[n][i]);
    }
    for(int i=n-1;i>=0;--i)
        for(int j=0;j<=i;++j)a[i][j]=a[i+1][j+1]*(j+1);
    vd ans=did(n);
    sort(ans.begin(),ans.end());
    for(int i=1;i+1<ans.size();++i)printf("%.10f\n",ans[i]);
    return 0;
}
```

数据结构

平衡的二叉查找树

Treap

```
#include<bits/stdc++.h>
using namespace std;
const int maxn=1e5+5;
#define sz(x) (x?x->siz:0)
struct Treap{
    struct node{
        int key,val;
        int siz,s;
        node *c[2];
        node(int v=0){
            val=v;
            key=rand();
            siz=1,s=1;
            c[0]=c[1]=0;
        }
        void rz(){siz=s;if(c[0])siz+=c[0]->siz;if(c[1])siz+=c[1]->siz;}
    }pool[maxn],*cur,*root;
    Treap(){cur=pool;}
    node* newnode(int val){return *cur=node(val),cur++;}
    void rot(node *&t,int d){
        if(!t->c[d])t=t->c[!d];
        else{
            node *p=t->c[d];t->c[d]=p->c[!d];
            p->c[!d]=t;t->rz();p->rz();t=p;
        }
    }
    void insert(node *&t,int x){
        if(!t){t=newnode(x);return;}
        if(t->val==x){t->s++;t->siz++;return;}
        insert(t->c[x>t->val],x);
        if(t->key<t->c[x>t->val]->key)
            rot(t,x>t->val);
        else t->rz();
    }
    void del(node *&t,int x){
        if(!t)return;
        if(t->val==x){
            if(t->s>1){t->s--;t->siz--;return;}
            if(!t->c[0]||!t->c[1]){
                if(!t->c[0])t=t->c[1];
                else t=t->c[0];
                return;
            }
            int d=t->c[0]->key<t->c[1]->key;
            rot(t,d);
            del(t,x);
            return;
        }
        del(t->c[x>t->val],x);
        t->rz();
    }
    int pre(node *t,int x){
        if(!t)return INT_MIN;
        int ans=pre(t->c[x>t->val],x);
        if(t->val<x)ans=max(ans,t->val);
        return ans;
    }
    int nxt(node *t,int x){
        if(!t)return INT_MAX;
        int ans=nxt(t->c[x>=t->val],x);
        if(t->val>x)ans=min(ans,t->val);
        return ans;
    }
    int rank(node *t,int x){
        if(!t)return 0;
        if(t->val==x)return sz(t->c[0]);
        if(t->val<x)return sz(t->c[0])+t->s+rank(t->c[1],x);
        if(t->val>x)return rank(t->c[0],x);
    }
    int kth(node *t,int x){
        if(sz(t->c[0])>=x)return kth(t->c[0],x);
        if(sz(t->c[0])+t->s>=x)return t->val;
        return kth(t->c[1],x-t->s-sz(t->c[0]));
    }
}T;
```

坚固的数据结构

坚固的平衡树

```
#define sz(x) (x?x->siz:0)
struct node{
    int siz,key;
    LL val,sum;
    LL mu,a,d;
    node *c[2],*f;
```

```
    void split(int ned,node *&p,node *&q);
    node* rz(){
        sum=val;siz=1;
        if(c[0])sum+=c[0]->sum,siz+=c[0]->siz;
        if(c[1])sum+=c[1]->sum,siz+=c[1]->siz;
        return this;
    }
    void make(LL _mu,LL _a,LL _d){
        sum=sum*_mu+_a*siz+_d*siz*(siz-1)/2;
        val=val*_mu+_a+_d*sz(c[0]);
        mu*=_mu;a=a*_mu+_a;d=d*_mu+_d;
    }
    void pd(){
        if(mu==1&&a==0&&d==0)return;
        if(c[0])c[0]->make(mu,a,d);
        if(c[1])c[1]->make(mu,a+d+d*sz(c[0]),d);
        mu=1;a=d=0;
    }
    node(){mu=1;}
}nd[maxn*2],*root;
node *merge(node *p,node *q){
    if(!p||!q)return p?p->rz():(q?q->rz():0);
    p->pd();q->pd();
    if(p->key<q->key){
        p->c[1]=merge(p->c[1],q);
        return p->rz();
    }else{
        q->c[0]=merge(p,q->c[0]);
        return q->rz();
    }
}
void node::split(int ned,node *&p,node *&q){
    if(!ned){p=0;q=this;return;}
    if(ned==siz){p=this;q=0;return;}
    pd();
    if(sz(c[0])>=ned){
        c[0]->split(ned,p,q);c[0]=0;rz();
        q=merge(q,this);
    }else{
        c[1]->split(ned-sz(c[0])-1,p,q);c[1]=0;rz();
        p=merge(this,p);
    }
}
int main(){
    for(int i=1;i<=n;i++){
        nd[i].val=in();
        nd[i].key=rand();
        nd[i].rz();
        root=merge(root,nd+i);
    }
}
```

坚固的字符串

坚固的左偏树

```
int Merge(int x, int y){
  if (x == 0 || y == 0) return x + y;
  if (Heap[x].Key < Heap[y].Key) swap(x, y);
  Heap[x].Ri = Merge(Heap[x].Ri, y);
  if (Heap[Heap[x].Le].Dis < Heap[Heap[x].Ri].Dis) swap(Heap[x].Le, Heap[x].Ri);
  if (Heap[x].Ri == 0) Heap[x].Dis = 0;
  else Heap[x].Dis = Heap[Heap[x].Ri].Dis + 1;
  return x;
}

for (int i = 0; i <= n; i++){
    Heap[i].Le = Heap[i].Ri = 0;
    Heap[i].Dis = 0;
    Heap[i].Key = Cost[i];
}
Heap[0].Dis = -1;
```

树上的魔术师

Link Cut Tree(zky)

```
struct LCT{
    struct node{
        bool rev;
        int mx,val;
        node *f,*c[2];
        bool d(){return this==f->c[1];}
        bool rt(){return !f||(f->c[0]!=this&&f->c[1]!=this);}
        void sets(node *x,int d){pd();if(x)x->f=this;c[d]=x;rz();}
        void makerv(){rev^=1;swap(c[0],c[1]);}
        void pd(){
            if(rev){
                if(c[0])c[0]->makerv();
                if(c[1])c[1]->makerv();
                rev=0;
            }
        }
        void rz(){
            mx=val;
            if(c[0])mx=max(mx,c[0]->mx);
            if(c[1])mx=max(mx,c[1]->mx);
        }
    }nd[int(1e4)+1];
    void rot(node *x){
        node *y=x->f;if(!y->rt())y->f->pd();
        y->pd();x->pd();bool d=x->d();
        y->sets(x->c[!d],d);
        if(y->rt())x->f=y->f;
        else y->f->sets(x,y->d());
        x->sets(y,!d);
    }
    void splay(node *x){
        while(!x->rt())
            if(x->f->rt())rot(x);
            else if(x->d()==x->f->d())rot(x->f),rot(x);
            else rot(x),rot(x);
    }
    node* access(node *x){
        node *y=0;
        for(;x;x=x->f){
            splay(x);
            x->sets(y,1);y=x;
        }return y;
    }
    void makert(node *x){
        access(x)->makerv();
        splay(x);
    }
    void link(node *x,node *y){
        makert(x);
        x->f=y;
        access(x);
    }
    void cut(node *x,node *y){
        makert(x);access(y);splay(y);
        y->c[0]=x->f=0;
        y->rz();
    }
    void link(int x,int y){link(nd+x,nd+y);}
```

```
    void cut(int x,int y){cut(nd+x,nd+y);}
}T;
```

Link Cut Tree(Splay)

```
struct node{
    bool Rev;
    int c[2], fa;
}T[N];
inline void Rev(int x){
    if (!x) return;
    swap(T[x].c[0], T[x].c[1]);
    T[x].Rev ^= 1;
}
inline void Lazy_Down(int x){
    if (!x) return;
    if (T[x].Rev) Rev(T[x].c[0]), Rev(T[x].c[1]), T[x].Rev = 0;
}
void Rotate(int x, int c){
    int y = T[x].c[c];
    int z = T[y].c[1 - c];
    if (T[x].fa){
        if (T[T[x].fa].c[0] == x) T[T[x].fa].c[0] = y;
        else T[T[x].fa].c[1] = y;
    }
    T[z].fa = x; T[x].c[c] = z;
    T[y].fa = T[x].fa; T[x].fa = y; T[y].c[1 - c] = x;
    //Update(x);
    //Update(y);
}
int stack[N], fx[N];
void Splay(int x){
    int top = 0;
    for (int  u = x; u; u = T[u].fa)
        stack[++top] = u;
    for (int i = top; i >= 1; i--)
        Lazy_Down(stack[i]);
    for (int i = 2; i <= top; i++)
        if (T[stack[i]].c[0] == stack[i - 1]) fx[i] = 0;
        else fx[i] = 1;
    for (int i = 2; i <= top; i += 2){
        if (i == top) Rotate(stack[i], fx[i]);
        else {
            if (fx[i] == fx[i + 1]){
                Rotate(stack[i + 1], fx[i + 1]);
                Rotate(stack[i], fx[i]);
            } else {
                Rotate(stack[i], fx[i]);
                Rotate(stack[i + 1], fx[i + 1]);
            }
        }
    }
    if (x != stack[top]) Par[x] = Par[stack[top]], Par[stack[top]] = 0;
    //if (fa == 0) Root = x;
}
inline int Access(int u){
    int Nxt = 0;
    while (u){
        Splay(u);
        if (T[u].c[1]){
            T[T[u].c[1]].fa = 0;
            Par[T[u].c[1]] = u;
        }
        T[u].c[1] = Nxt;
        if (Nxt){
            T[Nxt].fa = u;
            Par[Nxt] = 0;
        }
        //Update(u)
        Nxt = u;
        u = Par[u];
    }
    return Nxt;
}
inline void Root(int u){
    Access(u);
    Splay(u);
    Rev(u);
}
inline void Link(int u, int v){
    Root(u);
    Par[u] = v;
}
inline void Cut(int u, int v){
    Access(u);
    Splay(v);
    if (Par[v] != u){
        swap(u, v);
        Access(u);
        Splay(v);
    }
    Par[v] = 0;
}
inline int Find_Root(int x){
    Access(x);
    Splay(x);
    int y = x;
    while (T[y].c[0]){
        Lazy_Down(y);
        y = T[y].c[0];
    }
    return y;
}
```

可持久化线段树

```
struct node1 {
    int L, R, Lson, Rson, Sum;
} tree[N * 40];
int root[N], a[N], b[N];
int tot, n, m;
int Real[N];
int Same(int x) {
    ++tot;
    tree[tot] = tree[x];
    return tot;
}
int build(int L, int R) {
    ++tot;
    tree[tot].L = L;
    tree[tot].R = R;
    tree[tot].Lson = tree[tot].Rson = tree[tot].Sum = 0;
    if (L == R) return tot;
    int s = tot;
    int mid = (L + R) >> 1;
    tree[s].Lson = build(L, mid);
    tree[s].Rson = build(mid + 1, R);
    return s;
}
int Ask(int Lst, int Cur, int L, int R, int k) {
    if (L == R) return L;
    int Mid = (L + R) >> 1;
    int Left = tree[tree[Cur].Lson].Sum - tree[tree[Lst].Lson].Sum;
    if (Left >= k) return Ask(tree[Lst].Lson, tree[Cur].Lson, L, Mid, k);
    k -= Left;
    return Ask(tree[Lst].Rson, tree[Cur].Rson, Mid + 1, R, k);
```

```
}
int Add(int Lst, int pos) {
    int root = Same(Lst);
    tree[root].Sum++;
    if (tree[root].L == tree[root].R) return root;
    int mid = (tree[root].L + tree[root].R) >> 1;
    if (pos <= mid) tree[root].Lson = Add(tree[root].Lson, pos);
    else tree[root].Rson = Add(tree[root].Rson, pos);
    return root;
}
int main() {
    scanf("%d%d", &n, &m);
    int up = 0;
    for (int i = 1; i <= n; i++){
        scanf("%d", &a[i]);
        b[i] = a[i];
    }
    sort(b + 1, b + n + 1);
    up = unique(b + 1, b + n + 1) - b - 1;
    for (int i = 1; i <= n; i++){
        int tmp = lower_bound(b + 1, b + up + 1, a[i]) - b;
        Real[tmp] = a[i];
        a[i] = tmp;
    }
    tot = 0;
    root[0] = build(1, up);
    for (int i = 1; i <= n; i++){
        root[i] = Add(root[i - 1], a[i]);
    }
    for (int i = 1; i <= m; i++){
        int u, v, w;
        scanf("%d%d%d", &u, &v, &w);
        printf("%d\n", Real[Ask(root[u - 1], root[v], 1, up, w)]);
    }
    return 0;
}
```

k-d 树

```
long long norm(const long long &x) {
    //    For manhattan distance
    return std::abs(x);
    //    For euclid distance
    return x * x;
}

struct Point {
    int x, y, id;

    const int& operator [] (int index) const {
        if (index == 0) {
            return x;
        } else {
            return y;
        }
    }

    friend long long dist(const Point &a, const Point &b) {
        long long result = 0;
        for (int i = 0; i < 2; ++i) {
            result += norm(a[i] - b[i]);
        }
        return result;
    }
} point[N];

struct Rectangle {
    int min[2], max[2];

    Rectangle() {
        min[0] = min[1] = INT_MAX;
        max[0] = max[1] = INT_MIN;
    }

    void add(const Point &p) {
        for (int i = 0; i < 2; ++i) {
            min[i] = std::min(min[i], p[i]);
            max[i] = std::max(max[i], p[i]);
        }
    }

    long long dist(const Point &p) {
        long long result = 0;
        for (int i = 0; i < 2; ++i) {
            //    For minimum distance
            result += norm(std::min(std::max(p[i], min[i]), max[i]) - p[i]);
            //    For maximum distance
            result += std::max(norm(max[i] - p[i]), norm(min[i] - p[i]));
        }
        return result;
    }
};

struct Node {
    Point seperator;
    Rectangle rectangle;
    int child[2];

    void reset(const Point &p) {
        seperator = p;
        rectangle = Rectangle();
        rectangle.add(p);
        child[0] = child[1] = 0;
    }
} tree[N << 1];

int size, pivot;

bool compare(const Point &a, const Point &b) {
    if (a[pivot] != b[pivot]) {
        return a[pivot] < b[pivot];
    }
    return a.id < b.id;
}

int build(int l, int r, int type = 1) {
    pivot = type;
    if (l >= r) {
        return 0;
    }
    int x = ++size;
    int mid = l + r >> 1;
    std::nth_element(point + l, point + mid, point + r, compare);
    tree[x].reset(point[mid]);
    for (int i = l; i < r; ++i) {
        tree[x].rectangle.add(point[i]);
    }
    tree[x].child[0] = build(l, mid, type ^ 1);
    tree[x].child[1] = build(mid + 1, r, type ^ 1);
    return x;
}

int insert(int x, const Point &p, int type = 1) {
    pivot = type;
    if (x == 0) {
        tree[++size].reset(p);
        return size;
    }
    tree[x].rectangle.add(p);
```

```
    if (compare(p, tree[x].seperator)) {
        tree[x].child[0] = insert(tree[x].child[0], p, type ^ 1);
    } else {
        tree[x].child[1] = insert(tree[x].child[1], p, type ^ 1);
    }
    return x;
}

//    For minimum distance
void query(int x, const Point &p, std::pair<long long, int> &answer, int type =
  ↪ 1) {
    pivot = type;
    if (x == 0 || tree[x].rectangle.dist(p) > answer.first) {
        return;
    }
    answer = std::min(answer,
             std::make_pair(dist(tree[x].seperator, p), tree[x].seperator.id));
    if (compare(p, tree[x].seperator)) {
        query(tree[x].child[0], p, answer, type ^ 1);
        query(tree[x].child[1], p, answer, type ^ 1);
    } else {
        query(tree[x].child[1], p, answer, type ^ 1);
        query(tree[x].child[0], p, answer, type ^ 1);
    }
}

std::priority_queue<std::pair<long long, int> > answer;

void query(int x, const Point &p, int k, int type = 1) {
    pivot = type;
    if (x == 0 ||
        (int)answer.size() == k && tree[x].rectangle.dist(p) >
          ↪ answer.top().first) {
        return;
    }
    answer.push(std::make_pair(dist(tree[x].seperator, p),
      ↪ tree[x].seperator.id));
    if ((int)answer.size() > k) {
        answer.pop();
    }
    if (compare(p, tree[x].seperator)) {
        query(tree[x].child[0], p, k, type ^ 1);
        query(tree[x].child[1], p, k, type ^ 1);
    } else {
        query(tree[x].child[1], p, k, type ^ 1);
        query(tree[x].child[0], p, k, type ^ 1);
    }
}
```

莫队算法

```
struct node{
    int l, r, id;
    friend bool operator < (const node &a, const node &b){
        if (a.l / Block == b.l / Block) return a.r < b.r;
        return a.l / Block < b.l / Block;
    }
}q[N];
Block = int(sqrt(n));
for (int i = 1; i <= m; i++){
    scanf("%d%d", &q[i].l, &q[i].r);
    q[i].id = i;
}
sort(q + 1, q + 1 + m);
Cur = a[1]; /// Hints: adjust by yourself
Le = Ri = 1;
for (int i = 1; i <= m; i++){
    while (q[i].r > Ri) Ri++, ChangeRi(1, Le, Ri);
    while (q[i].l > Le) ChangeLe(-1, Le, Ri), Le++;
    while (q[i].l < Le) Le--, ChangeLe(1, Le, Ri);
    while (q[i].r < Ri) ChangeRi(-1, Le, Ri), Ri--;
    Ans[q[i].id] = Cur;
}
```

树状数组 kth

```
int find(int k){
    int cnt=0,ans=0;
    for(int i=22;i>=0;i--){
        ans+=(1<<i);
        if(ans>n || cnt+d[ans]>=k)ans-=(1<<i);
        else cnt+=d[ans];
    }
    return ans+1;
}
```

虚树

```
int a[maxn*2],sta[maxn*2];
int top=0,k;
void build(){
    top=0;
    sort(a,a+k,bydfn);
    k=unique(a,a+k)-a;
    sta[top++]=1;_n=k;
    for(int i=0;i<k;i++){
        int LCA=lca(a[i],sta[top-1]);
        while(dep[LCA]<dep[sta[top-1]]){
            if(dep[LCA]>=dep[sta[top-2]]){
                add_edge(LCA,sta[--top]);
                if(sta[top-1]!=LCA)sta[top++]=LCA;
                break;
            }add_edge(sta[top-2],sta[top-1]);top--;
        }if(sta[top-1]!=a[i])sta[top++]=a[i];
    }
    while(top>1)
        add_edge(sta[top-2],sta[top-1]),top--;
    for(int i=0;i<k;i++)inr[a[i]]=1;
}
```

点分治 (zky)

```
int siz[maxn],f[maxn],dep[maxn],cant[maxn],root,All,d[maxn];
void makert(int u,int fa){
    siz[u]=1;f[u]=0;
    for(int i=0;i<G[u].size();i++){
        edge e=G[u][i];
        if(e.v!=fa&&!cant[e.v]){
            dep[e.v]=dep[u]+1;
            makert(e.v,u);
            siz[u]+=siz[e.v];
            f[u]=max(f[u],siz[e.v]);
        }
    }f[u]=max(f[u],All-f[u]);
    if(f[root]>f[u])root=u;
}
void dfs(int u,int fa){
    //Gain data
    for(int i=0;i<G[u].size();i++){
        edge e=G[u][i];
        if(e.v==fa||cant[e.v])continue;
        d[e.v]=d[u]+e.w;
        dfs(e.v,u);
```

```
    }
}
void calc(int u){
    d[u]=0;
    for(int i=0;i<G[u].size();i++){
        edge e=G[u][i];
        if(cant[e.v])continue;
        d[e.v]=e.w;
        dfs(e.v,u);

    }
}
void solve(int u){
    calc(u);cant[u]=1;
    for(int i=0;i<G[u].size();i++){
        edge e=G[u][i];
        if(cant[e.v])continue;
        All=siz[e.v];
        f[root=0]=n+1;
        makert(e.v,0);
        solve(root);
    }
}
All=n
f[root=0]=n+1;
makert(1,1);
solve(root);
```

元芳树

```
void tarjan(int u){
    dfn[u]=low[u]=++tot;
    for(int i=0;i<G[u].size();i++){
        edge e=G[u][i];
        if(dfn[e.v])
            low[u]=min(low[u],dfn[e.v]);
        else{
            S.push(e);
            tarjan(e.v);
            if(low[e.v]==dfn[u]){

                if(S.top()==e){
                    fa[e.v][0]=u;
                    fw[e.v]=e.w;
                    S.pop();
                    continue;
                }

                Rcnt++;
                edge ed;
                do{
                    ed=S.top();S.pop();
                    ring[Rcnt].push_back(ed);
                }while(ed!=e);
                reverse(ring[Rcnt].begin(),ring[Rcnt].end());
                int last=ring[Rcnt].back().v;
                ring[Rcnt].push_back((edge){last,u,Mw[pack(last,u)]});
            }
            low[u]=min(low[u],low[e.v]);
        }
    }
}
void up(int u){
    if(dep[u]||u==1)return ;
    if(fa[u][0])up(fa[u][0]);
    dep[u]=dep[fa[u][0]]+1;
    fw[u]+=fw[fa[u][0]];
}
void build(){
    S.push((edge){0,1,0});
    tarjan(1);

    for(int i=1;i<=Rcnt;i++){
        rlen[i]=0;
        sum[i].resize(ring[i].size());
        dis[i].resize(ring[i].size());
        for(int j=0;j<ring[i].size();j++){
            rlen[i]+=ring[i][j].w;
            ind[i].push_back(make_pair(ring[i][j].u,j));
        }
        sum[i][0]=0;
        fw[i+n]=0;
        fa[i+n][0]=ring[i][0].u;
        for(int j=1;j<ring[i].size();j++){
            sum[i][j]=sum[i][j-1]+ring[i][j-1].w;
            dis[i][j]=min(sum[i][j],rlen[i]-sum[i][j]);
            fw[ring[i][j].u]=dis[i][j];
            fa[ring[i][j].u][0]=i+n;
        }
        sort(ind[i].begin(),ind[i].end());
    }

    for(int i=1;i<=n+Rcnt;i++)
        up(i);

    for(int j=1;j<BIT;j++)
    for(int i=1;i<=n+Rcnt;i++)if(fa[i][j-1])
        fa[i][j]=fa[fa[i][j-1]][j-1];

}
pair<int,int>second_lca;
int lca(int u,int v){
    if(dep[u]<dep[v])swap(u,v);
    int d=dep[u]-dep[v];
    for(int i=0;i<BIT;i++)if(d>>i&1)
        u=fa[u][i];
    if(u==v)return u;
    for(int i=BIT-1;i>=0;i--)if(fa[u][i]!=fa[v][i]){
        u=fa[u][i];
        v=fa[v][i];
    }
    second_lca=make_pair(u,v);
    return fa[u][0];
}
```

图论

点双连通分量 (lyx)

```
///求割点, 割点向每个点双连通分量连边
void Dfs(int x, int lst){
    dfn[x] = ++dfc;
    low[x] = dfn[x];
    stack[++cnt] = x;
    int son = 0;

    for (int i = g[x]; i; i = nxt[i]){
        if (!dfn[adj[i]]){
            ++son;
            Dfs(adj[i], i);
            low[x] = min(low[x], low[adj[i]]);
            if (low[adj[i]] >= dfn[x]){
                int Tmp;
                iscut[x] = 1;
                ++block;
                E[x].push_back(block + n);
```

```
                do{
                    Tmp = stack[cnt --];
                    belong[Tmp] = block + n;

                    E[Tmp].push_back(block + n);
                }while (Tmp != adj[i]);
            }
        }
        else
        if ((i  ^ lst) != 1) low[x] = min(low[x], dfn[adj[i]]);
    }
    if (x == Root && son == 1) iscut[x] = 0, belong[x] = E[x][0];
    if (x == Root && son == 0){
        ++block;
        belong[x] = block + n;
    }
}
    tot = 1;//!!!!!!!!!!!!!!!!!!!!!!!!!!!!!1
    block = 0;
    cnt = 0;
    dfc = 0;
    for (int i = 1; i <= n; i++)
        if (dfn[i] == 0){
            Root = i;
            Dfs(i, 0);
        }
```

2-SAT 问题 (强连通分量)

```
int stamp, comps, top;
int dfn[N], low[N], comp[N], stack[N];

void add(int x, int a, int y, int b) {
    edge[x << 1 | a].push_back(y << 1 | b);
}

void tarjan(int x) {
    dfn[x] = low[x] = ++stamp;
    stack[top++] = x;
    for (int i = 0; i < (int)edge[x].size(); ++i) {
        int y = edge[x][i];
        if (!dfn[y]) {
            tarjan(y);
            low[x] = std::min(low[x], low[y]);
        } else if (!comp[y]) {
            low[x] = std::min(low[x], dfn[y]);
        }
    }
    if (low[x] == dfn[x]) {
        comps++;
        do {
            int y = stack[--top];
            comp[y] = comps;
        } while (stack[top] != x);
    }
}

bool solve() {
    int counter = n + n + 1;
    stamp = top = comps = 0;
    std::fill(dfn, dfn + counter, 0);
    std::fill(comp, comp + counter, 0);
    for (int i = 0; i < counter; ++i) {
        if (!dfn[i]) {
            tarjan(i);
        }
    }
    for (int i = 0; i < n; ++i) {
        if (comp[i << 1] == comp[i << 1 | 1]) {
            return false;
        }
        answer[i] = (comp[i << 1 | 1] < comp[i << 1]);
    }
    return true;
}
```

二分图最大匹配

Hungary 算法

时间复杂度：$\mathcal{O}(V \cdot E)$

```
vector<int>G[maxn];
int Link[maxn],vis[maxn],T;
bool find(int x){
    for(int i=0;i<G[x].size();i++){
        int v=G[x][i];
        if(vis[v]==T)continue;
        vis[v]=T;
        if(!Link[v]||find(Link[v])){
            Link[v]=x;
            return 1;
        }
    }return 0;
}
int Hungarian(int n){
    int ans=0;
    memset(Link,0,sizeof Link);
    for(int i=1;i<=n;i++){
        T++;
        ans+=find(i);
    }return ans;
}
```

Hopcroft Karp 算法

时间复杂度：$\mathcal{O}(\sqrt{V} \cdot E)$

```
int matchx[N], matchy[N], level[N];

bool dfs(int x) {
    for (int i = 0; i < (int)edge[x].size(); ++i) {
        int y = edge[x][i];
        int w = matchy[y];
        if (w == -1 || level[x] + 1 == level[w] && dfs(w)) {
            matchx[x] = y;
            matchy[y] = x;
            return true;
        }
    }
    level[x] = -1;
    return false;
}

int solve() {
    std::fill(matchx, matchx + n, -1);
    std::fill(matchy, matchy + m, -1);
    for (int answer = 0; ; ) {
        std::vector<int> queue;
        for (int i = 0; i < n; ++i) {
            if (matchx[i] == -1) {
                level[i] = 0;
                queue.push_back(i);
            } else {
                level[i] = -1;
            }
```

```
        }
        for (int head = 0; head < (int)queue.size(); ++head) {
            int x = queue[head];
            for (int i = 0; i < (int)edge[x].size(); ++i) {
                int y = edge[x][i];
                int w = matchy[y];
                if (w != -1 && level[w] < 0) {
                    level[w] = level[x] + 1;
                    queue.push_back(w);
                }
            }
        }
        int delta = 0;
        for (int i = 0; i < n; ++i) {
            if (matchx[i] == -1 && dfs(i)) {
                delta++;
            }
        }
        if (delta == 0) {
            return answer;
        } else {
            answer += delta;
        }
    }
}
```

二分图最大权匹配

时间复杂度：$\mathcal{O}(V^4)$

```
int labelx[N], labely[N], match[N], slack[N];
bool visitx[N], visity[N];

bool dfs(int x) {
    visitx[x] = true;
    for (int y = 0; y < n; ++y) {
        if (visity[y]) {
            continue;
        }
        int delta = labelx[x] + labely[y] - graph[x][y];
        if (delta == 0) {
            visity[y] = true;
            if (match[y] == -1 || dfs(match[y])) {
                match[y] = x;
                return true;
            }
        } else {
            slack[y] = std::min(slack[y], delta);
        }
    }
    return false;
}

int solve() {
    for (int i = 0; i < n; ++i) {
        match[i] = -1;
        labelx[i] = INT_MIN;
        labely[i] = 0;
        for (int j = 0; j < n; ++j) {
            labelx[i] = std::max(labelx[i], graph[i][j]);
        }
    }
    for (int i = 0; i < n; ++i) {
        while (true) {
            std::fill(visitx, visitx + n, 0);
            std::fill(visity, visity + n, 0);
            for (int j = 0; j < n; ++j) {
                slack[j] = INT_MAX;
            }
            if (dfs(i)) {
                break;
            }
            int delta = INT_MAX;
            for (int j = 0; j < n; ++j) {
                if (!visity[j]) {
                    delta = std::min(delta, slack[j]);
                }
            }
            for (int j = 0; j < n; ++j) {
                if (visitx[j]) {
                    labelx[j] -= delta;
                }
                if (visity[j]) {
                    labely[j] += delta;
                } else {
                    slack[j] -= delta;
                }
            }
        }
    }
    int answer = 0;
    for (int i = 0; i < n; ++i) {
        answer += graph[match[i]][i];
    }
    return answer;
}
```

最大流 (dinic)

时间复杂度：$\mathcal{O}(V^2 \cdot E)$

```
struct edge{int u,v,cap,flow;};
vector<edge>edges;
vector<int>G[maxn];
int s,t;
int cur[maxn],d[maxn];
void add(int u,int v,int cap){
    edges.push_back((edge){u,v,cap,0});
    G[u].push_back(edges.size()-1);
    edges.push_back((edge){v,u,0,0});
    G[v].push_back(edges.size()-1);
}
bool bfs(){
    static int vis[maxn];
    memset(vis,0,sizeof vis);vis[s]=1;
    queue<int>q;q.push(s);d[s]=0;
    while(!q.empty()){
        int u=q.front();q.pop();
        for(int i=0;i<G[u].size();i++){
            edge e=edges[G[u][i]];if(vis[e.v]||e.cap==e.flow)continue;
            d[e.v]=d[u]+1;vis[e.v]=1;q.push(e.v);
        }
    }return vis[t];
}
int dfs(int u,int a){
    if(u==t||!a)return a;
    int flow=0,f;
    for(int &i=cur[u];i<G[u].size();i++){
        edge e=edges[G[u][i]];
        if(d[e.v]==d[u]+1&&(f=dfs(e.v,min(a,e.cap-e.flow)))>0){
            edges[G[u][i]].flow+=f;
            edges[G[u][i]^1].flow-=f;
            flow+=f;a-=f;if(!a)break;
        }
```

```
    }return flow;
}
int dinic(){
    int flow=0,x;
    while(bfs()){
        memset(cur,0,sizeof cur);
        while(x=dfs(s,INT_MAX)){
            flow+=x;
            memset(cur,0,sizeof cur);
        }
    }return flow;
}
```

最大流 (sap)

时间复杂度：$\mathcal{O}(V^2 \cdot E)$

```
int g[T], adj[M], nxt[M], f[M];
int cnt[T], dist[T], cur[T], fa[T], dat[T];
void Ins(int x, int y, int ff, int rf){
    adj[++tot] = y; nxt[tot] = g[x]; g[x] = tot; f[tot] = ff;
    adj[++tot] = x; nxt[tot] = g[y]; g[y] = tot; f[tot] = rf;
}
int sap(int s, int t){
    int x, sum;
    for (int i = 1; i <= t; i++){
        dist[i] = 1;
        cur[i] = g[i];
        fa[i] = 0;
        dat[i] = 0;
        cnt[i] = 0;
    }
    cnt[0] = 1; cnt[1] = t - 1;
    dist[t] = 0;
    dat[s] = INF;
    x = s;
    sum = 0;
    while (1){
        int p;
        for (p = cur[x]; p; p = nxt[p]){
            if (f[p] > 0 && dist[adj[p]] == dist[x] - 1) break;
        }
        if (p > 0){
            cur[x] = p;
            fa[adj[p]] = p;
            dat[adj[p]] = min(dat[x], f[p]);
            x = adj[p];
            if (x == t){
                sum += dat[x];
                while (x != s){
                    f[fa[x]] -= dat[t];
                    f[fa[x] ^ 1] += dat[t];
                    x = adj[fa[x] ^ 1];
                }
            }
        } else {
            cnt[dist[x]] --;
            if (cnt[dist[x]] == 0) return sum;
            dist[x] = t + 1;
            for (int p = g[x]; p; p = nxt[p]){
                if (f[p] > 0 && dist[adj[p]] + 1 < dist[x]){
                    dist[x] = dist[adj[p]] + 1;
                    cur[x] = p;
                }
            }
            cnt[dist[x]]++;
            if (dist[s] > t) return sum;
            if (x != s) x = adj[fa[x] ^ 1];
        }
    }
}
/*
tot = 1
edges' id start from 2
remember to clean g
t is the number of points
*/
```

上下界网络流

$B(u,v)$ 表示边 (u,v) 流量的下界，$C(u,v)$ 表示边 (u,v) 流量的上界，$F(u,v)$ 表示边 (u,v) 的流量。设 $G(u,v) = F(u,v) - B(u,v)$，显然有

$$0 \le G(u,v) \le C(u,v) - B(u,v)$$

无源汇的上下界可行流

建立超级源点 S^* 和超级汇点 T^*，对于原图每条边 (u,v) 在新网络中连如下三条边：$S^* \to v$，容量为 $B(u,v)$；$u \to T^*$，容量为 $B(u,v)$；$u \to v$，容量为 $C(u,v) - B(u,v)$。最后求新网络的最大流，判断从超级源点 S^* 出发的边是否都满流即可，边 (u,v) 的最终解中的实际流量为 $G(u,v) + B(u,v)$。

有源汇的上下界可行流

从汇点 T 到源点 S 连一条上界为 ∞，下界为 0 的边。按照**无源汇的上下界可行流**一样做即可，流量即为 $T \to S$ 边上的流量。

有源汇的上下界最大流

1. 在**有源汇的上下界可行流**中，从汇点 T 到源点 S 的边改为连一条上界为 ∞，下届为 x 的边。x 满足二分性质，找到最大的 x 使得新网络存在**无源汇的上下界可行流**即为原图的最大流。

2. 从汇点 T 到源点 S 连一条上界为 ∞，下界为 0 的边，变成无源汇的网络。按照**无源汇的上下界可行流**的方法，建立超级源点 S^* 和超级汇点 T^*，求一遍 $S^* \to T^*$ 的最大流，再将从汇点 T 到源点 S 的这条边拆掉，求一次 $S \to T$ 的最大流即可。

有源汇的上下界最小流

1. 在**有源汇的上下界可行流**中，从汇点 T 到源点 S 的边改为连一条上界为 x，下界为 0 的边。x 满足二分性质，找到最小的 x 使得新网络存在**无源汇的上下界可行流**即为原图的最小流。

2. 按照**无源汇的上下界可行流**的方法，建立超级源点 S^* 与超级汇点 T^*，求一遍 $S^* \to T^*$ 的最大流，但是注意这一次不加上汇点 T 到源点 S 的这条边，即不使之改为无源汇的网络去求解。求完后，再加上那条汇点 T 到源点 S 上界 ∞ 的边。因为这条边下界为 0，所以 S^*，T^* 无影响，再直接求一次 $S^* \to T^*$ 的最大流。若超级源点 S^* 出发的边全部满流，则 $T \to S$ 边上的流量即为原图的最小流，否则无解。

最小费用最大流

稀疏图

时间复杂度：$\mathcal{O}(V \cdot E^2)$

```
struct EdgeList {
    int size;
    int last[N];
    int succ[M], other[M], flow[M], cost[M];
    void clear(int n) {
        size = 0;
        std::fill(last, last + n, -1);
    }
    void add(int x, int y, int c, int w) {
        succ[size] = last[x];
        last[x] = size;
        other[size] = y;
        flow[size] = c;
        cost[size++] = w;
    }
} e;

int n, source, target;
int prev[N];

void add(int x, int y, int c, int w) {
```

```cpp
    e.add(x, y, c, w);
    e.add(y, x, 0, -w);
}

bool augment() {
    static int dist[N], occur[N];
    std::vector<int> queue;
    std::fill(dist, dist + n, INT_MAX);
    std::fill(occur, occur + n, 0);
    dist[source] = 0;
    occur[source] = true;
    queue.push_back(source);
    for (int head = 0; head < (int)queue.size(); ++head) {
        int x = queue[head];
        for (int i = e.last[x]; ~i; i = e.succ[i]) {
            int y = e.other[i];
            if (e.flow[i] && dist[y] > dist[x] + e.cost[i]) {
                dist[y] = dist[x] + e.cost[i];
                prev[y] = i;
                if (!occur[y]) {
                    occur[y] = true;
                    queue.push_back(y);
                }
            }
        }
        occur[x] = false;
    }
    return dist[target] < INT_MAX;
}

std::pair<int, int> solve() {
    std::pair<int, int> answer = std::make_pair(0, 0);
    while (augment()) {
        int number = INT_MAX;
        for (int i = target; i != source; i = e.other[prev[i] ^ 1]) {
            number = std::min(number, e.flow[prev[i]]);
        }
        answer.first += number;
        for (int i = target; i != source; i = e.other[prev[i] ^ 1]) {
            e.flow[prev[i]] -= number;
            e.flow[prev[i] ^ 1] += number;
            answer.second += number * e.cost[prev[i]];
        }
    }
    return answer;
}
```

稠密图

使用条件：费用非负

时间复杂度：$\mathcal{O}(V \cdot E^2)$

```cpp
struct EdgeList {
    int size;
    int last[N];
    int succ[M], other[M], flow[M], cost[M];
    void clear(int n) {
        size = 0;
        std::fill(last, last + n, -1);
    }
    void add(int x, int y, int c, int w) {
        succ[size] = last[x];
        last[x] = size;
        other[size] = y;
        flow[size] = c;
        cost[size++] = w;
    }
} e;

int n, source, target, flow, cost;
int slack[N], dist[N];
bool visit[N];

void add(int x, int y, int c, int w) {
    e.add(x, y, c, w);
    e.add(y, x, 0, -w);
}

bool relabel() {
    int delta = INT_MAX;
    for (int i = 0; i < n; ++i) {
        if (!visit[i]) {
            delta = std::min(delta, slack[i]);
        }
        slack[i] = INT_MAX;
    }
    if (delta == INT_MAX) {
        return true;
    }
    for (int i = 0; i < n; ++i) {
        if (visit[i]) {
            dist[i] += delta;
        }
    }
    return false;
}

int dfs(int x, int answer) {
    if (x == target) {
        flow += answer;
        cost += answer * (dist[source] - dist[target]);
        return answer;
    }
    visit[x] = true;
    int delta = answer;
    for (int i = e.last[x]; ~i; i = e.succ[i]) {
        int y = e.other[i];
        if (e.flow[i] > 0 && !visit[y]) {
            if (dist[y] + e.cost[i] == dist[x]) {
                int number = dfs(y, std::min(e.flow[i], delta));
                e.flow[i] -= number;
                e.flow[i ^ 1] += number;
                delta -= number;
                if (delta == 0) {
                    dist[x] = INT_MIN;
                    return answer;
                }
            } else {
                slack[y] = std::min(slack[y], dist[y] + e.cost[i] - dist[x]);
            }
        }
    }
    return answer - delta;
}

std::pair<int, int> solve() {
    flow = cost = 0;
    std::fill(dist, dist + n, 0);
    do {
        do {
            fill(visit, visit + n, 0);
        } while (dfs(source, INT_MAX));
    } while (!relabel());
    return std::make_pair(flow, cost);
}
```

一般图最大匹配

时间复杂度：$\mathcal{O}(V^3)$

```
int match[N], belong[N], next[N], mark[N], visit[N];
std::vector<int> queue;

int find(int x) {
    if (belong[x] != x) {
        belong[x] = find(belong[x]);
    }
    return belong[x];
}

void merge(int x, int y) {
    x = find(x);
    y = find(y);
    if (x != y) {
        belong[x] = y;
    }
}

int lca(int x, int y) {
    static int stamp = 0;
    stamp++;
    while (true) {
        if (x != -1) {
            x = find(x);
            if (visit[x] == stamp) {
                return x;
            }
            visit[x] = stamp;
            if (match[x] != -1) {
                x = next[match[x]];
            } else {
                x = -1;
            }
        }
        std::swap(x, y);
    }
}

void group(int a, int p) {
    while (a != p) {
        int b = match[a], c = next[b];
        if (find(c) != p) {
            next[c] = b;
        }
        if (mark[b] == 2) {
            mark[b] = 1;
            queue.push_back(b);
        }
        if (mark[c] == 2) {
            mark[c] = 1;
            queue.push_back(c);
        }
        merge(a, b);
        merge(b, c);
        a = c;
    }
}

void augment(int source) {
    queue.clear();
    for (int i = 0; i < n; ++i) {
        next[i] = visit[i] = -1;
        belong[i] = i;
        mark[i] = 0;
    }
    mark[source] = 1;
    queue.push_back(source);
    for (int head = 0; head < (int)queue.size() && match[source] == -1; ++head) {
        int x = queue[head];
        for (int i = 0; i < (int)edge[x].size(); ++i) {
            int y = edge[x][i];
            if (match[x] == y || find(x) == find(y) || mark[y] == 2) {
                continue;
            }
            if (mark[y] == 1) {
                int r = lca(x, y);
                if (find(x) != r) {
                    next[x] = y;
                }
                if (find(y) != r) {
                    next[y] = x;
                }
                group(x, r);
                group(y, r);
            } else if (match[y] == -1) {
                next[y] = x;
                for (int u = y; u != -1; ) {
                    int v = next[u];
                    int mv = match[v];
                    match[v] = u;
                    match[u] = v;
                    u = mv;
                }
                break;
            } else {
                next[y] = x;
                mark[y] = 2;
                mark[match[y]] = 1;
                queue.push_back(match[y]);
            }
        }
    }
}

int solve() {
    std::fill(match, match + n, -1);
    for (int i = 0; i < n; ++i) {
        if (match[i] == -1) {
            augment(i);
        }
    }
    int answer = 0;
    for (int i = 0; i < n; ++i) {
        answer += (match[i] != -1);
    }
    return answer;
}
```

无向图全局最小割

时间复杂度：$\mathcal{O}(V^3)$

注意事项：处理重边时，应该对边权累加

```
int node[N], dist[N];
bool visit[N];

int solve(int n) {
    int answer = INT_MAX;
    for (int i = 0; i < n; ++i) {
        node[i] = i;
    }
    while (n > 1) {
```

```cpp
        int max = 1;
        for (int i = 0; i < n; ++i) {
            dist[node[i]] = graph[node[0]][node[i]];
            if (dist[node[i]] > dist[node[max]]) {
                max = i;
            }
        }
        int prev = 0;
        memset(visit, 0, sizeof(visit));
        visit[node[0]] = true;
        for (int i = 1; i < n; ++i) {
            if (i == n - 1) {
                answer = std::min(answer, dist[node[max]]);
                for (int k = 0; k < n; ++k) {
                    graph[node[k]][node[prev]] =
                       (graph[node[prev]][node[k]] += graph[node[k]][node[max]]);
                }
                node[max] = node[--n];
            }
            visit[node[max]] = true;
            prev = max;
            max = -1;
            for (int j = 1; j < n; ++j) {
                if (!visit[node[j]]) {
                    dist[node[j]] += graph[node[prev]][node[j]];
                    if (max == -1 || dist[node[max]] < dist[node[j]]) {
                        max = j;
                    }
                }
            }
        }
    }
    return answer;
}
```

哈密尔顿回路（ORE 性质的图）

ORE 性质：

$$\forall x, y \in V \wedge (x, y) \notin E \quad s.t. \quad deg_x + deg_y \geq n$$

返回结果：从顶点 1 出发的一个哈密尔顿回路

使用条件：$n \geq 3$

```cpp
int left[N], right[N], next[N], last[N];

void cover(int x) {
    left[right[x]] = left[x];
    right[left[x]] = right[x];
}

int adjacent(int x) {
    for (int i = right[0]; i <= n; i = right[i]) {
        if (graph[x][i]) {
            return i;
        }
    }
    return 0;
}

std::vector<int> solve() {
    for (int i = 1; i <= n; ++i) {
        left[i] = i - 1;
        right[i] = i + 1;
    }
    int head, tail;
    for (int i = 2; i <= n; ++i) {
        if (graph[1][i]) {
            head = 1;
            tail = i;
            cover(head);
            cover(tail);
            next[head] = tail;
            break;
        }
    }
    while (true) {
        int x;
        while (x = adjacent(head)) {
            next[x] = head;
            head = x;
            cover(head);
        }
        while (x = adjacent(tail)) {
            next[tail] = x;
            tail = x;
            cover(tail);
        }
        if (!graph[head][tail]) {
            for (int i = head, j; i != tail; i = next[i]) {
                if (graph[head][next[i]] && graph[tail][i]) {
                    for (j = head; j != i; j = next[j]) {
                        last[next[j]] = j;
                    }
                    j = next[head];
                    next[head] = next[i];
                    next[tail] = i;
                    tail = j;
                    for (j = i; j != head; j = last[j]) {
                        next[j] = last[j];
                    }
                    break;
                }
            }
        }
        next[tail] = head;
        if (right[0] > n) {
            break;
        }
        for (int i = head; i != tail; i = next[i]) {
            if (adjacent(i)) {
                head = next[i];
                tail = i;
                next[tail] = 0;
                break;
            }
        }
    }
    std::vector<int> answer;
    for (int i = head; ; i = next[i]) {
        if (i == 1) {
            answer.push_back(i);
            for (int j = next[i]; j != i; j = next[j]) {
                answer.push_back(j);
            }
            answer.push_back(i);
            break;
        }
        if (i == tail) {
            break;
        }
    }
    return answer;
}
```

必经点树

```
vector<int>G[maxn],rG[maxn],dom[maxn];
int n,m;
int dfn[maxn],rdfn[maxn],dfs_c,semi[maxn],idom[maxn],fa[maxn];
struct ufsets{
    int fa[maxn],best[maxn];
    int find(int x){
        if(fa[x]==x)
            return x;
        int f=find(fa[x]);
        if(dfn[semi[best[x]]]>dfn[semi[best[fa[x]]]])
            best[x]=best[fa[x]];
        fa[x]=f;
        return f;
    }
    int getbest(int x){
        find(x);
        return best[x];
    }
    void init(){
        for(int i=1;i<=n;i++)
            fa[i]=best[i]=i;
    }
}uf;
void init(){
    uf.init();
    for(int i=1;i<=n;i++){
        semi[i]=i;
        idom[i]=0;
        fa[i]=0;
        dfn[i]=rdfn[i]=0;
    }
    dfs_c=0;
}
void dfs(int u){
    dfn[u]=++dfs_c;
    rdfn[dfn[u]]=u;
    for(int i=0;i<G[u].size();i++){
        int v=G[u][i];
        if(!dfn[v]){
            fa[v]=u;
            dfs(v);
        }
    }
}

void tarjan(){
    for(int i=n;i>1;i--){
        int tmp=1e9;
        int y=rdfn[i];
        for(int i=0;i<rG[y].size();i++){
            int x=rG[y][i];
            tmp=min(tmp,dfn[semi[uf.getbest(x)]]);
        }
        semi[y]=rdfn[tmp];
        int x=fa[y];
        dom[semi[y]].push_back(y);
        uf.fa[y]=x;
        for(int i=0;i<dom[x].size();i++){
            int z=dom[x][i];
            if(dfn[semi[uf.getbest(z)]]<dfn[x])
                idom[z]=uf.getbest(z);
            else
                idom[z]=semi[z];
        }
        dom[x].clear();
    }
    semi[rdfn[1]]=1;
    for(int i=2;i<=n;i++){
        int x=rdfn[i];
        if(idom[x]!=semi[x])
            idom[x]=idom[idom[x]];

    }
    idom[rdfn[1]]=0;
}
init();
dfs(1);
tarjan();
```

字符串

模式匹配

KMP 算法

```
void build(char *pattern) {
    int length = (int)strlen(pattern + 1);
    fail[0] = -1;
    for (int i = 1, j; i <= length; ++i) {
        for (j = fail[i - 1]; j != -1 && pattern[i] != pattern[j + 1]; j = fail[j]);
        fail[i] = j + 1;
    }
}

void solve(char *text, char *pattern) {
    int length = (int)strlen(text + 1);
    for (int i = 1, j; i <= length; ++i) {
       for (j = match[i - 1]; j != -1 && text[i] != pattern[j + 1]; j = fail[j]);
        match[i] = j + 1;
    }
}
//Hint: 1 - Base
```

扩展 KMP 算法

返回结果：

$$next_i = lcp(text, text_{i...n-1})$$

```
void solve(char *text, int length, int *next) {
    int j = 0, k = 1;
    for (; j + 1 < length && text[j] == text[j + 1]; j++);
    next[0] = length - 1;
    next[1] = j;
    for (int i = 2; i < length; ++i) {
        int far = k + next[k] - 1;
        if (next[i - k] < far - i + 1) {
            next[i] = next[i - k];
        } else {
            j = std::max(far - i + 1, 0);
            for (; i + j < length && text[j] == text[i + j]; j++);
            next[i] = j;
            k = i;
        }
    }
}
/// 0 - Base
```

AC 自动机

```
struct Node{
    int Next[30], fail, mark;
}Tree[N];

void Init(){
    memset(Tree, 0, sizeof Tree);
```

```
    cnt = 1;

    for (int i = 1; i <= n; i++){
        char c;
        int now = 1;
        scanf("%s", s + 1);
        int Length = strlen(s + 1);
        for (int j = 1; j <= Length; j++){
            c = s[j];
            if (Tree[now].Next[c - 'a']) now = Tree[now].Next[c - 'a']; else
                Tree[now].Next[c - 'a'] = ++ cnt, now = cnt;
        }
    }
}

void Build_Ac(){
    int en = 0;
    Q[0] = 1;
    for (int fi = 0; fi <= en; fi++){
        int now = Q[fi];
        for (int next = 0; next < 26; next++)
            if (Tree[now].Next[next])
            {
                int k = Tree[now].Next[next];
                if (now == 1) Tree[k].fail = 1; else
                {
                    int h = Tree[now].fail;
                    while (h && !Tree[h].Next[next]) h = Tree[h].fail;
                    if (!h) Tree[k].fail = 1;
                    else Tree[k].fail = Tree[h].Next[next];
                }
                Q[++ en] = k;
            }
    }
}

/// Hints : when not match , fail = 1
```

后缀三姐妹

后缀数组

```
struct Sa{
    int heap[N],s[N],sa[N],r[N],tr[N],sec[N],m,cnt;
    int h[19][N];
    void Prep(){
        for (int i=1; i<=m; i++) heap[i]=0;
        for (int i=1; i<=n; i++) heap[s[i]]++;
        for (int i=2; i<=m; i++) heap[i]+=heap[i-1];
        for (int i=n; i>=1; i--) sa[heap[s[i]]--]=i;
        r[sa[1]]=1; cnt=1;
        for (int i=2; i<=n; i++){
            if (s[sa[i]]!=s[sa[i-1]]) cnt++;
            r[sa[i]]=cnt;
        }
        m=cnt;
    }
    void Suffix(){
        int j=1;
        while (cnt<n){
            cnt=0;
            for (int i=n-j+1; i<=n; i++) sec[++cnt]=i;
            for (int i=1; i<=n; i++) if (sa[i]>j)sec[++cnt]=sa[i]-j;
            for (int i=1; i<=n; i++) tr[i]=r[sec[i]];
            for (int i=1; i<=m; i++) heap[i]=0;
            for (int i=1; i<=n; i++) heap[tr[i]]++;
            for (int i=2; i<=m; i++) heap[i]+=heap[i-1];
            for (int i=n; i>=1; i--) sa[heap[tr[i]]--]=sec[i];
            tr[sa[1]]=1; cnt=1;
            for (int i=2; i<=n; i++){
                if ((r[sa[i]]!=r[sa[i-1]]) || (r[sa[i]+j]!=r[sa[i-1]+j])) cnt++;
                tr[sa[i]]=cnt;
            }
            for (int i=1; i<=n; i++) r[i]=tr[i];
            m=cnt; j=j+j;
        }
    }
    void Calc(){
        int k=0;
        for (int i=1; i<=n; i++){
            if (r[i]==1) continue;
            int j=sa[r[i]-1];
            while ((i+k<=n) && (j+k<=n) && (s[i+k]==s[j+k])) k++;
            h[0][r[i]]=k;
            if (k) k--;
        }
        for (int i=1; i<19; i++)
            for (int j=1; j+(1 << i)-1<=n; j++)
                h[i][j]=min(h[i-1][j],h[i-1][j + (1 << (i - 1)) + 1]);
    }
    int Query(int L,int R){
        L=r[L], R=r[R];
        if (L>R) swap(L,R);
        L++;
        int l0 = Lg[R-L+1];
        return min(h[l0][L],h[l0][R-(1 << l0)+1]);
    }
    void Work(){
        Prep(); Suffix(); Calc();
    }
}P,S;/// Hints : 1 - Base
```

后缀数组 (dc3)

```
//`DC3 待排序的字符串放在 r 数组中, 从 r[0] 到 r[n-1], 长度为 n, 且最大值小于 m.`
//`约定除 r[n-1] 外所有的 r[i] 都大于 0, r[n-1]=0。`
//`函数结束后, 结果放在 sa 数组中, 从 sa[0] 到 sa[n-1]。`
//`r 必须开长度乘 3`
#define maxn 10000
#define F(x) ((x)/3+((x)%3==1?0:tb))
#define G(x) ((x)<tb?(x)*3+1:((x)-tb)*3+2)

int wa[maxn],wb[maxn],wv[maxn],wss[maxn];
int s[maxn*3],sa[maxn*3];
int c0(int *r,int a,int b)
{
    return r[a]==r[b]&&r[a+1]==r[b+1]&&r[a+2]==r[b+2];
}
int c12(int k,int *r,int a,int b)
{
    if(k==2) return r[a]<r[b]||r[a]==r[b]&&c12(1,r,a+1,b+1);
    else return r[a]<r[b]||r[a]==r[b]&&wv[a+1]<wv[b+1];
}
void sort(int *r,int *a,int *b,int n,int m)
{
    int i;
    for(i=0;i<n;i++) wv[i]=r[a[i]];
    for(i=0;i<m;i++) wss[i]=0;
    for(i=0;i<n;i++) wss[wv[i]]++;
    for(i=1;i<m;i++) wss[i]+=wss[i-1];
    for(i=n-1;i>=0;i--) b[--wss[wv[i]]]=a[i];
}
void dc3(int *r,int *sa,int n,int m)
{
    int i,j,*rn=r+n,*san=sa+n,ta=0,tb=(n+1)/3,tbc=0,p;
    r[n]=r[n+1]=0;
```

```
    for(i=0;i<n;i++)
        if(i%3!=0) wa[tbc++]=i;
    sort(r+2,wa,wb,tbc,m);
    sort(r+1,wb,wa,tbc,m);
    sort(r,wa,wb,tbc,m);
    for(p=1,rn[F(wb[0])]=0,i=1;i<tbc;i++)
        rn[F(wb[i])]=c0(r,wb[i-1],wb[i])?p-1:p++;
    if (p<tbc) dc3(rn,san,tbc,p);
    else for (i=0;i<tbc;i++) san[rn[i]]=i;
    for (i=0;i<tbc;i++)
        if(san[i]<tb) wb[ta++]=san[i]*3;
    if(n%3==1) wb[ta++]=n-1;
    sort(r,wb,wa,ta,m);
    for(i=0;i<tbc;i++)
        wv[wb[i]=G(san[i])]=i;
    for(i=0,j=0,p=0;i<ta && j<tbc;p++)
        sa[p]=c12(wb[j]%3,r,wa[i],wb[j])?wa[i++]:wb[j++];
    for(;i<ta;p++) sa[p]=wa[i++];
    for(;j<tbc;p++) sa[p]=wb[j++];
}

int main(){
    int n,m=0;
    scanf("%d",&n);
    for (int i=0;i<n;i++) scanf("%d",&s[i]),s[i]++,m=max(s[i]+1,m);
    printf("%d\n",m);
    s[n++]=0;
    dc3(s,sa,n,m);
    for (int i=0;i<n;i++) printf("%d ",sa[i]);printf("\n");
}
```

后缀自动机

多串 LCS　　对一个串建后缀自动机，其他串在上面匹配，因为是求所有串的公共子串，所以每个点记录每个串最长匹配长度的最小值，最后找到所有点中最长的一个即可。一个注意事项就是，当走到一个点时，还要更新它的 parent 树上的祖先的匹配长度，数组开两倍啦啦啦！

各长度字串出现次数最大值　　给一个字符串 S，令 F(x) 表示 S 的所有长度为 x 的子串中，出现次数的最大值。　　构建字符串的自动机，对于每个节点，right 集合大小就是出现次数，maxs 就是它代表的最长长度，那么我们用 |right(x)| 去更新 f[maxs[x]] 的值，最后从大到小用 f[i] 去更新 f[i-1] 的值即可

```
struct Node{
    int len, fail;
    int To[30];
}T[N];
int Lst, Root, tot, ans;
char s[N];
int Len[N], Ans[N], Ord[N];
void Add(int x, int l){
    int Nt = ++tot, p = Lst;
    T[Nt].len = l;
    for (;p && !T[p].To[x]; p = T[p].fail) T[p].To[x] = Nt;
    if (!p) T[Nt].fail = Root; else
    if (T[T[p].To[x]].len == T[p].len + 1) T[Nt].fail = T[p].To[x];
    else{
        int q = ++tot, qt = T[p].To[x];
        T[q] = T[qt];
        T[q].len = T[p].len + 1;
        T[qt].fail = T[Nt].fail = q;
        for (;p && T[p].To[x] == qt; p = T[p].fail) T[p].To[x] = q;
    }
    Lst = Nt;
}
bool cmp(int a, int b){
    return T[a].len < T[b].len;
}
int main(){
    scanf("%s", s + 1);
    int n = strlen(s + 1);
    ans = n;
    Root = tot = Lst = 1;
    for (int i = 1; i <= n; i++)
        Add(s[i] - 'a' + 1, i);
    for (int i = 1; i <= tot; i++)
        Ord[i] = i;
    sort(Ord + 1, Ord + tot + 1, cmp);
    for (int i = 1; i <= tot; i++)
        Ans[i] = T[i].len;
    bool flag = 0;
    while (scanf("%s", s + 1) != EOF){
        flag = 1;
        int n = strlen(s + 1);
        int p = Root, len = 0;
        for (int i = 1; i <= tot; i++) Len[i] = 0;
        for (int i = 1; i <= n; i++){
            int x = s[i] - 'a' + 1;
            if (T[p].To[x]) len++, p = T[p].To[x];
            else {
                while (p && !T[p].To[x]) p = T[p].fail;
                if (!p) p = Root, len = 0;
                else len = T[p].len + 1, p = T[p].To[x];
            }
            Len[p] = max(Len[p], len);
        }
        for (int i = tot; i >= 1; i--){
            int Cur = Ord[i];
            Ans[Cur] = min(Ans[Cur], Len[Cur]);
            if (Len[Cur] && T[Cur].fail)
                Len[T[Cur].fail] = T[T[Cur].fail].len;
        }
    }
    if (flag){
        ans = 0;
        for (int i = 1; i <= tot; i++){
            ans = max(ans, Ans[i]);
        }
    }
    printf("%d\n", ans);
    return 0;
}
```

回文三兄弟

马拉车

```
void Manacher(){
    R[1] = 1;
    for (int i = 2, j = 1; i <= length; i++){
        if (j + R[j] <= i){
            R[i] = 0;
        } else {
            R[i] = min(R[j * 2 - i], j + R[j] - i);
        }
        while (i - R[i] >= 1 && i + R[i] <= length
            && text[i - R[i]] == text[i + R[i]]){
            R[i]++;
        }
        if (i + R[i] > j + R[j]){
            j = i;
        }
    }
}
    length = 0;
    int n = strlen(s + 1);
    for (int i = 1; i <= n; i++){
        text[++length] = '*';
        text[++length] = s[i];
```

```
    }
    text[++length] = '*';
/// Hints: 1 - Base
```

回文自动机 (zky)

```
struct PAM{
    int tot,last,str[maxn],nxt[maxn][26],n;
    int len[maxn],suf[maxn],cnt[maxn];
    int newnode(int l){
        len[tot]=l;
        return tot++;
    }
    void init(){
        tot=0;
        newnode(0);// tree0 is node 0
        newnode(-1);// tree-1 is node 1
        str[0]=-1;
        suf[0]=1;
    }
    int find(int x){
        while(str[n-len[x]-1]!=str[n])x=suf[x];
        return x;
    }
    void add(int c){
        str[++n]=c;
        int u=find(last);
        if(!nxt[u][c]){
            int v=newnode(len[u]+2);
            suf[v]=nxt[find(suf[u])][c];
            nxt[u][c]=v;
        }last=nxt[u][c];
        cnt[last]++;
    }
    void count(){
        for(int i=tot-1;i>=0;i--)cnt[suf[i]]+=cnt[i];
    }
}P;
int main(){
    P.init();
    for(int i=0;i<n;i++)
        P.add(s[i]-'a');
    P.count();
```

循环串最小表示

```
string sol(char *s){
    int n=strlen(s);
    int i=0,j=1,k=0,p;
    while(i<n&&j<n&&k<n){
        int t=s[(i+k)%n]-s[(j+k)%n];
        if(t==0)k++;
        else if(t<0)j+=k+1,k=0;
        else i+=k+1,k=0;
        if(i==j)j++;
    }p=min(i,j);
    string S;
    for(int i=p;i<p+n;i++)S.push_back(s[i%n]);
    return S;
}
```

计算几何

二维基础

点类

```
struct P{
    double x,y;
    P turn90(){return P(-y,x);}
};
double det(P a,P b,P c){
    return (b-a)*(c-a);
}
P intersect(L l1,L l2){
    double s1=det(l1.a,l1.b,l2.a);
    double s2=det(l1.a,l1.b,l2.b);
    return (l2.a*s2-l2.b*s1)/(s2-s1);
}
P project(P p,L l){
    return l.a+l.v()*((p-l.a)^l.v())/l.v().len2();
}
double dis(P p,L l){
    return fabs((p-l.a)*l.v())/l.v().len();
}
```

圆类

```
struct C{
    P o;
    double r;
    C(){}
    C(P _o,double _r):o(_o),r(_r){}
};
// 求圆与直线的交点
//turn90() P(-y,x)
double fix(double x){return sgn(x)?x:0;}
bool intersect(C a, L l, P &p1, P &p2) {
    double x = ((l.a - a.o)^ (l.b - l.a)),
        y = (l.b - l.a).len2(),
        d = x * x - y * ((l.a - a.o).len2() - a.r * a.r);
    if (sgn(d) < 0) return false;
    d = max(d, 0.0);
    P p = l.a - ((l.b - l.a) * (x / y)), delta = (l.b - l.a) * (sqrt(d) / y);
    p1 = p + delta, p2 = p - delta;
    return true;
}
// 求圆与圆的交点，注意调用前要先判定重圆
bool intersect(C a, C b, P &p1, P &p2) {
    double s1 = (a.o - b.o).len();
    if (sgn(s1 - a.r - b.r) > 0 || sgn(s1 - fabs(a.r - b.r)) < 0) return false;
    double s2 = (a.r * a.r - b.r * b.r) / s1;
    double aa = (s1 + s2) * 0.5, bb = (s1 - s2) * 0.5;
    P o = (b.o - a.o) * (aa / (aa + bb)) + a.o;
    P delta = (b.o - a.o).norm().turn90() * sqrt(fix(a.r * a.r - aa * aa));
    p1 = o + delta, p2 = o - delta;
    return true;
}
// 求点到圆的切点，按关于点的顺时针方向返回两个点
bool tang(const C &c, const P &p0, P &p1, P &p2) {
    double x = (p0 - c.o).len2(), d = x - c.r * c.r;
    if (d < eps) return false; // 点在圆上认为没有切点
    P p = (p0 - c.o) * (c.r * c.r / x);
    P delta = ((p0 - c.o) * (-c.r * sqrt(d) / x)).turn90();
    p1 = c.o + p + delta;
    p2 = c.o + p - delta;
    return true;
}
// 求圆到圆的外共切线，按关于 c1.o 的顺时针方向返回两条线
vector<L> extan(const C &c1, const C &c2) {
    vector<L> ret;
    if (sgn(c1.r - c2.r) == 0) {
        P dir = c2.o - c1.o;
        dir = (dir * (c1.r / dir.len())).turn90();
        ret.push_back(L(c1.o + dir, c2.o + dir));
```

```cpp
        ret.push_back(L(c1.o - dir, c2.o - dir));
    } else {
        P p = (c1.o * -c2.r + c2.o * c1.r) / (c1.r - c2.r);
        P p1, p2, q1, q2;
        if (tang(c1, p, p1, p2) && tang(c2, p, q1, q2)) {
            if (c1.r < c2.r) swap(p1, p2), swap(q1, q2);
            ret.push_back(L(p1, q1));
            ret.push_back(L(p2, q2));
        }
    }
    return ret;
}
// 求圆到圆的内共切线, 按关于 c1.o 的顺时针方向返回两条线
vector<L> intan(const C &c1, const C &c2) {
    vector<L> ret;
    P p = (c1.o * c2.r + c2.o * c1.r) / (c1.r + c2.r);
    P p1, p2, q1, q2;
    if (tang(c1, p, p1, p2) && tang(c2, p, q1, q2)) { // 两圆相切认为没有切线
        ret.push_back(L(p1, q1));
        ret.push_back(L(p2, q2));
    }
    return ret;
}
```

凸包

```cpp
vector<P> convex(vector<P>p){
    sort(p.begin(),p.end());
    vector<P>ans,S;
    for(int i=0;i<p.size();i++){
        while(S.size()>=2
                && sgn(det(S[S.size()-2],S.back(),p[i]))<=0)
                    S.pop_back();
        S.push_back(p[i]);
    }//dw
    ans=S;
    S.clear();
    for(int i=(int)p.size()-1;i>=0;i--){
        while(S.size()>=2
                && sgn(det(S[S.size()-2],S.back(),p[i]))<=0)
                    S.pop_back();
        S.push_back(p[i]);
    }//up
    for(int i=1;i+1<S.size();i++)
        ans.push_back(S[i]);
    return ans;
}
```

半平面交

```cpp
struct P{
    int quad() const { return sgn(y) == 1 || (sgn(y) == 0 && sgn(x) >= 0);}
};
struct L{
    bool onLeft(const P &p) const { return sgn((b - a)*( p - a)) > 0; }
    L push() const{ // push out eps
        const double eps = 1e-10;
        P delta = (b - a).turn90().norm() * eps;
        return L(a - delta, b - delta);
    }
};
bool sameDir(const L &l0, const L &l1) {
    return parallel(l0, l1) && sgn((l0.b - l0.a)^(l1.b - l1.a)) == 1;
}
bool operator < (const P &a, const P &b) {
    if (a.quad() != b.quad())
        return a.quad() < b.quad();
    else
        return sgn((a*b)) > 0;
}
bool operator < (const L &l0, const L &l1) {
    if (sameDir(l0, l1))
        return l1.onLeft(l0.a);
    else
        return (l0.b - l0.a) < (l1.b - l1.a);
}
bool check(const L &u, const L &v, const L &w) {
    return w.onLeft(intersect(u, v));
}
vector<P> intersection(vector<L> &l) {
    sort(l.begin(), l.end());
    deque<L> q;
    for (int i = 0; i < (int)l.size(); ++i) {
        if (i && sameDir(l[i], l[i - 1])) {
            continue;
        }
        while (q.size() > 1
            && !check(q[q.size() - 2], q[q.size() - 1], l[i]))
                q.pop_back();
        while (q.size() > 1
            && !check(q[1], q[0], l[i]))
                q.pop_front();
        q.push_back(l[i]);
    }
    while (q.size() > 2
        && !check(q[q.size() - 2], q[q.size() - 1], q[0]))
            q.pop_back();
    while (q.size() > 2
        && !check(q[1], q[0], q[q.size() - 1]))
            q.pop_front();
    vector<P> ret;
    for (int i = 0; i < (int)q.size(); ++i)
    ret.push_back(intersect(q[i], q[(i + 1) % q.size()]));
    return ret;
}
```

最小圆覆盖

```cpp
point operator*(line A,line B){
    point u=B.p-A.p;
    double t=(B.v*u)/(B.v*A.v);
    return A.p+A.v*t;
}
point get(point a,point b){
    return (a+b)/2;
}
point get(point a,point b,point c){
    if(a==b)return get(a,c);
    if(a==c)return get(a,b);
    if(b==c)return get(a,b);
    line ABO=(line){(a+b)/2,Rev(a-b)};
    line BCO=(line){(c+b)/2,Rev(b-c)};
    return ABO*BCO;
}
    random_shuffle(p+1,p+1+n);
    O=p[1];r=0;
    for(int i=2;i<=n;i++){
        if(dis(p[i],O)<r+1e-6)continue;
        O=get(p[1],p[i]);r=dis(O,p[i]);
        for(int j=1;j<i;j++){
            if(dis(p[j],O)<r+1e-6)continue;
            O=get(p[i],p[j]);r=dis(O,p[i]);
            for(int k=1;k<j;k++){
                if(dis(p[k],O)<r+1e-6)continue;
```

```
                O=get(p[i],p[j],p[k]);r=dis(O,p[i]);
            }
        }
    }
```

多边形

判断点在多边形内部

```
bool InPoly(P p,vector<P>poly){
    int cnt=0;
    for(int i=0;i<poly.size();i++){
        P a=poly[i],b=poly[(i+1)%poly.size()];
        if(OnLine(p,L(a,b)))
            return false;
        int x=sgn(det(a,p,b));
        int y=sgn(a.y-p.y);
        int z=sgn(b.y-p.y);
        cnt+=(x>0&&y<=0&&z>0);
        cnt-=(x<0&&z<=0&&y>0);
    }
    return cnt;
}
```

其他

斯坦纳树

```
priority_queue<pair<int, int> > Q;
// m is key point
// n is all point
for (int s = 0; s < (1 << m); s++){
    for (int i = 1; i <= n; i++)
        for (int s0 = (s&(s-1)); s0 ; s0=(s&(s0-1)))
                f[s][i] = min(f[s][i], f[s0][i] + f[s - s0][i]);
    for (int i = 1; i <= n; i++) vis[i] = 0;
    while (!Q.empty()) Q.pop();
    for (int i = 1; i <= n; i++)
        Q.push(mp(-f[s][i], i));
    while (!Q.empty()){
        while (!Q.empty() && Q.top().first != -f[s][Q.top().second]) Q.pop();
            if (Q.empty()) break;
            int Cur = Q.top().second; Q.pop();
            for (int p = g[Cur]; p; p = nxt[p]){
                int y = adj[p];
                if ( f[s][y] > f[s][Cur] + 1){
                    f[s][y] = f[s][Cur] + 1;
                    Q.push(mp(-f[s][y], y));
                }
            }
    }
}
```

无敌的读入优化

```
namespace Reader {
    const int L = (1 << 20) + 5;
    char buffer[L], *S, *T;
    __inline bool getchar(char &ch) {
        if (S == T) {
            T = (S = buffer) + fread(buffer, 1, L, stdin);
            if (S == T) {
                ch = EOF;
                return false;
            }
        }
        ch = *S ++;
        return true;
    }
    __inline bool getint(int &x) {
        char ch;
        for (; getchar(ch) && (ch < '0' || ch > '9'); );
        if (ch == EOF) return false;
        x = ch - '0';
        for (; getchar(ch), ch >= '0' && ch <= '9'; )
            x = x * 10 + ch - '0';
        return true;
    }
}
Reader::getint(x);
Reader::getint(y);
```

最小树形图

```
const int maxn=1100;

int n,m , g[maxn][maxn] , used[maxn] , pass[maxn] , eg[maxn] , more ,
    queue[maxn];

void combine (int id , int &sum ) {
    int tot = 0 , from , i , j , k ;
    for ( ; id!=0 && !pass[ id ] ; id=eg[id] ) {
        queue[tot++]=id ; pass[id]=1;
    }
    for ( from=0; from<tot && queue[from]!=id ; from++);
    if  ( from==tot ) return ;
    more = 1 ;
    for ( i=from ; i<tot ; i++) {
        sum+=g[eg[queue[i]]][queue[i]] ;
        if ( i!=from ) {
            used[queue[i]]=1;
            for ( j = 1 ; j <= n ; j++) if ( !used[j] )
                if ( g[queue[i]][j]<g[id][j] ) g[id][j]=g[queue[i]][j] ;
        }
    }
    for ( i=1; i<=n ; i++) if ( !used[i] && i!=id ) {
        for ( j=from ; j<tot ; j++){
            k=queue[j];
            if ( g[i][id]>g[i][k]-g[eg[k]][k] ) g[i][id]=g[i][k]-g[eg[k]][k];
        }
    }
}

int mdst( int root ) { // return the total length of MDST
    int i , j , k , sum = 0 ;
    memset ( used , 0 , sizeof ( used ) ) ;
    for ( more =1; more ; ) {
        more = 0 ;
        memset (eg,0,sizeof(eg)) ;
        for ( i=1 ; i <= n ; i ++) if ( !used[i] && i!=root ) {
            for ( j=1 , k=0 ; j <= n ; j ++) if ( !used[j] && i!=j )
                if ( k==0 || g[j][i] < g[k][i] ) k=j ;
            eg[i] = k ;
        }
        memset(pass,0,sizeof(pass));
        for ( i=1; i<=n ; i++) if ( !used[i] && !pass[i] && i!= root ) combine (
            i , sum ) ;
    }
    for ( i =1; i<=n ; i ++) if ( !used[i] && i!= root ) sum+=g[eg[i]][i];
    return sum ;
}
```

DLX

```
int n,m,K;
struct DLX{
    int L[maxn],R[maxn],U[maxn],D[maxn];
    int sz,col[maxn],row[maxn],s[maxn],H[maxn];
    bool vis[233];
    int ans[maxn],cnt;
    void init(int m){
        for(int i=0;i<=m;i++){
            L[i]=i-1;R[i]=i+1;
            U[i]=D[i]=i;s[i]=0;
        }
        memset(H,-1,sizeof H);
        L[0]=m;R[m]=0;sz=m+1;
    }
    void Link(int r,int c){
        U[sz]=c;D[sz]=D[c];U[D[c]]=sz;D[c]=sz;
        if(H[r]<0)H[r]=L[sz]=R[sz]=sz;
        else{
            L[sz]=H[r];R[sz]=R[H[r]];
            L[R[H[r]]]=sz;R[H[r]]=sz;
        }
        s[c]++;col[sz]=c;row[sz]=r;sz++;
    }
    void remove(int c){
        for(int i=D[c];i!=c;i=D[i])
            L[R[i]]=L[i],R[L[i]]=R[i];
    }
    void resume(int c){
        for(int i=U[c];i!=c;i=U[i])
            L[R[i]]=R[L[i]]=i;
    }
    int A(){
        int res=0;
        memset(vis,0,sizeof vis);
        for(int i=R[0];i;i=R[i])if(!vis[i]){
            vis[i]=1;res++;
            for(int j=D[i];j!=i;j=D[j])
                for(int k=R[j];k!=j;k=R[k])
                    vis[col[k]]=1;
        }
        return res;
    }
    void dfs(int d,int &ans){
        if(R[0]==0){ans=min(ans,d);return;}
        if(d+A()>=ans)return;
        int tmp=23333,c;
        for(int i=R[0];i;i=R[i])
            if(tmp>s[i])tmp=s[i],c=i;
        for(int i=D[c];i!=c;i=D[i]){
            remove(i);
            for(int j=R[i];j!=i;j=R[j])remove(j);
            dfs(d+1,ans);
            for(int j=L[i];j!=i;j=L[j])resume(j);
            resume(i);
        }
    }
    void del(int c){//exactly cover
        L[R[c]]=L[c];R[L[c]]=R[c];
        for(int i=D[c];i!=c;i=D[i])
            for(int j=R[i];j!=i;j=R[j])
                U[D[j]]=U[j],D[U[j]]=D[j],--s[col[j]];
    }
    void add(int c){  //exactly cover
        R[L[c]]=L[R[c]]=c;
        for(int i=U[c];i!=c;i=U[i])
            for(int j=L[i];j!=i;j=L[j])
                ++s[col[U[D[j]]=D[U[j]]=j]];
    }
    bool dfs2(int k){//exactly cover
        if(!R[0]){
            cnt=k;return 1;
        }
        int c=R[0];
        for(int i=R[0];i;i=R[i])
            if(s[c]>s[i])c=i;
        del(c);
        for(int i=D[c];i!=c;i=D[i]){
            for(int j=R[i];j!=i;j=R[j])
                del(col[j]);
            ans[k]=row[i];if(dfs2(k+1))return true;
            for(int j=L[i];j!=i;j=L[j])
                add(col[j]);
        }
        add(c);
        return 0;
    }
}dlx;
int main(){
    dlx.init(n);
    for(int i=1;i<=m;i++)
        for(int j=1;j<=n;j++)
            if(dis(station[i],city[j])<mid-eps)
                dlx.Link(i,j);
            dlx.dfs(0,ans);
}
```

某年某月某日是星期几

```
int solve(int year, int month, int day) {
    int answer;
    if (month == 1 || month == 2) {
        month += 12;
        year--;
    }
    if ((year < 1752) || (year == 1752 && month < 9) ||
        (year == 1752 && month == 9 && day < 3)) {
       answer = (day + 2 * month + 3 * (month + 1) / 5 + year + year / 4 + 5) % 7;
    } else {
        answer = (day + 2 * month + 3 * (month + 1) / 5 + year + year / 4
               - year / 100 + year / 400) % 7;
    }
    return answer;
}
```

枚举大小为 k 的子集

使用条件：$k > 0$

```
void solve(int n, int k) {
    for (int comb = (1 << k) - 1; comb < (1 << n); ) {
        // ...
        int x = comb & -comb, y = comb + x;
        comb = (((comb & ~y) / x) >> 1) | y;
    }
}
```

环状最长公共子串

```
int n, a[N << 1], b[N << 1];

bool has(int i, int j) {
    return a[(i - 1) % n] == b[(j - 1) % n];
}
```

```cpp

const int DELTA[3][2] = {{0, -1}, {-1, -1}, {-1, 0}};

int from[N][N];

int solve() {
    memset(from, 0, sizeof(from));
    int ret = 0;
    for (int i = 1; i <= 2 * n; ++i) {
        from[i][0] = 2;
        int left = 0, up = 0;
        for (int j = 1; j <= n; ++j) {
            int upleft = up + 1 + !!from[i - 1][j];
            if (!has(i, j)) {
                upleft = INT_MIN;
            }
            int max = std::max(left, std::max(upleft, up));
            if (left == max) {
                from[i][j] = 0;
            } else if (upleft == max) {
                from[i][j] = 1;
            } else {
                from[i][j] = 2;
            }
            left = max;
        }
        if (i >= n) {
            int count = 0;
            for (int x = i, y = n; y; ) {
                int t = from[x][y];
                count += t == 1;
                x += DELTA[t][0];
                y += DELTA[t][1];
            }
            ret = std::max(ret, count);
            int x = i - n + 1;
            from[x][0] = 0;
            int y = 0;
            while (y <= n && from[x][y] == 0) {
                y++;
            }
            for (; x <= i; ++x) {
                from[x][y] = 0;
                if (x == i) {
                    break;
                }
                for (; y <= n; ++y) {
                    if (from[x + 1][y] == 2) {
                        break;
                    }
                    if (y + 1 <= n && from[x + 1][y + 1] == 1) {
                        y++;
                        break;
                    }
                }
            }
        }
    }
    return ret;
}
```

LLMOD

```cpp
LL multiplyMod(LL a, LL b, LL P) { // `需要保证 a 和 b 非负`
    LL t = (a * b - LL((long double)a / P * b + 1e-3) * P) % P;
    return t < 0 : t + P : t;
}
```

Java

基础模板

```java
public class Main {
    public static void main(String[] args) {
        InputReader in = new InputReader(System.in);
        PrintWriter out = new PrintWriter(System.out);
    }
}
public static class cmp implements Comparator<edge>{
    public int compare(edge a,edge b){
        if(a.w<b.w)return 1;
        if(a.w>b.w)return -1;
        return 0;
    }
}
class InputReader {
    public BufferedReader reader;
    public StringTokenizer tokenizer;
    public InputReader(InputStream stream) {
        reader = new BufferedReader(new InputStreamReader(stream), 32768);
        tokenizer = null;
    }
    public String next() {
        while (tokenizer == null || !tokenizer.hasMoreTokens()) {
            try {
                tokenizer = new StringTokenizer(reader.readLine());
            } catch (IOException e) {
                throw new RuntimeException(e);
            }
        }
        return tokenizer.nextToken();
    }
    public int nextInt() {
        return Integer.parseInt(next());
    }
}
```