# Spear of Longinus

**Shanghai Jiao Tong University**

# Contents

## 0.1 数论

### 0.1.1 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;
    }
}
```

### 0.1.2 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;
}
```

### 0.1.3 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;
}
```

### 0.1.4 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;
}
```

### 0.1.5 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;
}
```

### 0.1.6 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;
}
```

### 0.1.7 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);
}
```

### 0.1.8 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：$\mathcal{O}(nlogn)$

```
1  /*
2  {(mod,G)}={(81788929,7),(101711873,3),(167772161,3)
3           ,(377487361,7),(998244353,3),(1224736769,3)
4           ,(1300234241,3),(1484783617,5)}
5  */
6  int mo=998244353,G=3;
7  void NTT(int a[],int n,int f){
8      for(register int i=0;i<n;i++)
9          if(i<rev[i])
10             swap(a[i],a[rev[i]]);
11     for (register int i=2;i<=n;i<<=1){
12         static int exp[maxn];
13         exp[0]=1;exp[1]=pw(G,(mo-1)/i);
14         if(f==-1)exp[1]=pw(exp[1],mo-2);
15         for(register int k=2;k<(i>>1);k++)
16             exp[k]=1LL*exp[k-1]*exp[1]%mo;
17         for(register int j=0;j<n;j+=i){
18             for(register int k=0;k<(i>>1);k++){
19                 register int &pA=a[j+k],&pB=a[j+k+(i>>1)];
20                 register int A=pA,B=1LL*pB*exp[k]%mo;
21                 pA=(A+B)%mo;
22                 pB=(A-B+mo)%mo;
23             }
24         }
25     }
26     if(f==-1){
27         int rv=pw(n,mo-2)%mo;
28         for(int i=0;i<n;i++)
29             a[i]=1LL*a[i]*rv%mo;
30     }
31 }
32 void mul(int m,int a[],int b[],int c[]){
33     int n=1,len=0;
34     while(n<m)n<<=1,len++;
35     for (int i=1;i<n;i++)
36         rev[i]=(rev[i>>1]>>1)|((i&1)<<(len-1));
37     NTT(a,n,1);
38     NTT(b,n,1);
39     for(int i=0;i<n;i++)
40         c[i]=1LL*a[i]*b[i]%mo;
41     NTT(c,n,-1);
42 }
```

### 0.1.9 原根

```
1  vector<LL>fct;
2  bool check(LL x,LL g){
3      for(int i=0;i<fct.size();i++)
4          if(pw(g,(x-1)/fct[i],x)==1)
5              return 0;
6      return 1;
7  }
8  LL findrt(LL x){
9      LL tmp=x-1;
10     for(int i=2;i*i<=tmp;i++){
11         if(tmp%i==0){
12             fct.push_back(i);
13             while(tmp%i==0)tmp/=i;
14         }
15     }if(tmp>1)fct.push_back(tmp);
16     // x is 1,2,4,p^n,2p^n
17     // x has phi(phi(x)) primitive roots
18     for(int i=2;i<int(1e9);i++)if(check(x,i))
19         return i;
20     return -1;
21 }
```

```
22 const int BASE[12] = {2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37};
23 bool check(long long n,int base) {
24     long long n2=n-1,res;
25     int s=0;
26     while(n2%2==0) n2>>=1,s++;
27     res=pw(base,n2,n);
28     if((res==1)||(res==n-1)) return 1;
29     while(s--) {
30         res=mul(res,res,n);
31         if(res==n-1) return 1;
32     }
33     return 0; // n is not a strong pseudo prime
34 }
35 bool isprime(const long long &n) {
36     if(n==2)
37         return true;
38     if(n<2 || n%2==0)
39         return false;
40     for(int i=0;i<12&&BASE[i]<n;i++){
41         if(!check(n,BASE[i]))
42             return false;
43     }
44     return true;
45 }
```

### 0.1.10 线性递推

```
1  //已知 a_0,a_1,...,a_{m-1}\\
2        a_n = c_0 * a_{n-m} + ... + c_{m-1} * a_{n-1}\\
3      求 a_n = v_0 * a_0 + v_1 * a_1 + ... + v_{m-1} * a_{m-1}\\
4
5  void linear_recurrence(long long n, int m, int a[], int c[], int p) {
6      long long v[M] = {1 % p}, u[M << 1], msk = !!n;
7      for(long long i(n); i > 1; i >>= 1) {
8          msk <<= 1;
9      }
10     for(long long x(0); msk; msk >>= 1, x <<= 1) {
11         fill_n(u, m << 1, 0);
12         int b(!!(n & msk));
13         x |= b;
14         if(x < m) {
15             u[x] = 1 % p;
16         }else {
17             for(int i(0); i < m; i++) {
18                 for(int j(0), t(i + b); j < m; j++, t++) {
19                     u[t] = (u[t] + v[i] * v[j]) % p;
20                 }
21             }
22             for(int i((m << 1) - 1); i >= m; i--) {
23                 for(int j(0), t(i - m); j < m; j++, t++) {
24                     u[t] = (u[t] + c[j] * u[i]) % p;
25                 }
26             }
27         }
28         copy(u, u + m, v);
29     }
30     //a[n] = v[0] * a[0] + v[1] * a[1] + ... + v[m - 1] * a[m - 1].
31     for(int i(m); i < 2 * m; i++) {
32         a[i] = 0;
33         for(int j(0); j < m; j++) {
34             a[i] = (a[i] + (long long)c[j] * a[i + j - m]) % p;
35         }
36     }
37     for(int j(0); j < m; j++) {
38         b[j] = 0;
39         for(int i(0); i < m; i++) {
40             b[j] = (b[j] + v[i] * a[i + j]) % p;
```

```
41            }
42        }
43        for(int j(0); j < m; j++) {
44            a[j] = b[j];
45        }
46 }
```

### 0.1.11 直线下整点个数

返回结果：

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

```
1 //calc \sum_{i=0}^{n-1} [(a+bi)/m]
2 // n,a,b,m >0
3 LL solve(LL n,LL a,LL b,LL m){
4     if(b==0)
5         return n*(a/m);
6     if(a>=m || b>=m)
7         return n*(a/m)+(n-1)*n/2*(b/m)+solve(n,a%m,b%m,m);
8     return solve((a+b*n)/m,(a+b*n)%m,m,b);
9 }
```

## 0.2 数值

### 0.2.1 高斯消元

```
1 void Gauss(){
2     int r,k;
3     for(int i=0;i<n;i++){
4         r=i;
5         for(int j=i+1;j<n;j++)
6             if(fabs(A[j][i])>fabs(A[r][i]))r=j;
7         if(r!=i)for(int j=0;j<=n;j++)swap(A[i][j],A[r][j]);
8         for(int k=i+1;k<n;k++){
9             double f=A[k][i]/A[i][i];
10            for(int j=i;j<=n;j++)A[k][j]-=f*A[i][j];
11        }
12    }
13    for(int i=n-1;i>=0;i--){
14        for(int j=i+1;j<n;j++)
15            A[i][n]-=A[j][n]*A[i][j];
16        A[i][n]/=A[i][i];
17    }
18    for(int i=0;i<n-1;i++)
19        cout<<fixed<<setprecision(3)<<A[i][n]<<" ";
20    cout<<fixed<<setprecision(3)<<A[n-1][n];
21 }
22 bool Gauss(){
23    for(int i=1;i<=n;i++){
24        int r=0;
25        for(int j=i;j<=m;j++)
26        if(a[j][i]){r=j;break;}
27        if(!r)return 0;
28        ans=max(ans,r);
29        swap(a[i],a[r]);
30        for(int j=i+1;j<=m;j++)
31        if(a[j][i])a[j]^=a[i];
32    }for(int i=n;i>=1;i--){
33        for(int j=i+1;j<=n;j++)if(a[i][j])
34        a[i][n+1]=a[i][n+1]^a[j][n+1];
35    }return 1;
36 }
37 LL Gauss(){
38    for(int i=0;i<n;i++)for(int j=0;j<n;j++)A[i][j]%=m;
39    for(int i=0;i<n;i++)for(int j=0;j<n;j++)A[i][j]=(A[i][j]+m)%m;
40    LL ans=n%2?-1:1;
```

```
41    for(int i=0;i<n;i++){
42        for(int j=i+1;j<n;j++){
43            while(A[j][i]){
44                LL t=A[i][i]/A[j][i];
45                for(int k=0;k<n;k++)
46                A[i][k]=(A[i][k]-A[j][k]*t%m+m)%m;
47                swap(A[i],A[j]);
48                ans=-ans;
49            }
50        }ans=ans*A[i][i]%m;
51    }return (ans%m+m)%m;
52 }
53 int Gauss(){//求秩
54    int r,now=-1;
55    int ans=0;
56    for(int i = 0; i <n; i++){
57        r = now + 1;
58        for(int j = now + 1; j < m; j++)
59            if(fabs(A[j][i]) > fabs(A[r][i]))
60                r = j;
61        if (!sgn(A[r][i])) continue;
62        ans++;
63        now++;
64        if(r != now)
65            for(int j = 0; j < n; j++)
66                swap(A[r][j], A[now][j]);
67
68        for(int k = now + 1; k < m; k++){
69            double t = A[k][i] / A[now][i];
70            for(int j = 0; j < n; j++){
71                A[k][j] -= t * A[now][j];
72            }
73        }
74    }
75    return ans;
76 }
```

### 0.2.2 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：$\mathcal{O}(nlogn)$

```
1 typedef complex<double> cp;
2 const double pi = acos(-1);
3 void FFT(vector<cp>&num,int len,int ty){
4     for(int i=1,j=0;i<len-1;i++){
5         for(int k=len;j^=k>>=1,~j&k;);
6         if(i<j)
7             swap(num[i],num[j]);
8     }
9     for(int h=0;(1<<h)<len;h++){
10        int step=1<<h,step2=step<<1;
11        cp w0(cos(2.0*pi/step2),ty*sin(2.0*pi/step2));
12        for(int i=0;i<len;i+=step2){
13            cp w(1,0);
14            for(int j=0;j<step;j++){
15                cp &x=num[i+j+step];
16                cp &y=num[i+j];
17                cp d=w*x;
18                x=y-d;
19                y=y+d;
20                w=w*w0;
21            }
22        }
```

```
23        }
24    if(ty==-1)
25        for(int i=0;i<len;i++)
26            num[i]=cp(num[i].real()/(double)len,num[i].imag());
27 }
28 vector<cp> mul(vector<cp>a,vector<cp>b){
29    int len=a.size()+b.size();
30    while((len&-len)!=len)len++;
31    while(a.size()<len)a.push_back(cp(0,0));
32    while(b.size()<len)b.push_back(cp(0,0));
33    FFT(a,len,1);
34    FFT(b,len,1);
35    vector<cp>ans(len);
36    for(int i=0;i<len;i++)
37        ans[i]=a[i]*b[i];
38    FFT(ans,len,-1);
39    return ans;
40 }
```

### 0.2.3 单纯形法求解线性规划

返回结果：

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

```
1 namespace LP{
2    const int maxn=233;
3    double a[maxn][maxn];
4    int Ans[maxn],pt[maxn];
5    int n,m;
6    void pivot(int l,int i){
7        double t;
8        swap(Ans[l+n],Ans[i]);
9        t=-a[l][i];
10       a[l][i]=-1;
11       for(int j=0;j<=n;j++)a[l][j]/=t;
12       for(int j=0;j<=m;j++){
13           if(a[j][i]&&j!=l){
14               t=a[j][i];
15               a[j][i]=0;
16               for(int k=0;k<=n;k++)a[j][k]+=t*a[l][k];
17           }
18       }
19    }
20    vector<double> solve(vector<vector<double>
   ↪ >A,vector<double>B,vector<double>C){
21        n=C.size();
22        m=B.size();
23        for(int i=0;i<C.size();i++)
24            a[0][i+1]=C[i];
25        for(int i=0;i<B.size();i++)
26            a[i+1][0]=B[i];
27
28        for(int i=0;i<m;i++)
29            for(int j=0;j<n;j++)
30                a[i+1][j+1]=-A[i][j];
31
32        for(int i=1;i<=n;i++)Ans[i]=i;
33
34        double t;
35        for(;;){
36            int l=0;t=-eps;
37            for(int j=1;j<=m;j++)if(a[j][0]<t)t=a[l=j][0];
38            if(!l)break;
39            int i=0;
40            for(int j=1;j<=n;j++)if(a[l][j]>eps){i=j;break;}
41            if(!i){
42                puts("Infeasible");
43                return vector<double>();
```

```
44            }
45            pivot(l,i);
46        }
47        for(;;){
48            int i=0;t=eps;
49            for(int j=1;j<=n;j++)if(a[0][j]>t)t=a[0][i=j];
50            if(!i)break;
51            int l=0;
52            t=1e30;
53            for(int j=1;j<=m;j++)if(a[j][i]<-eps){
54                double tmp;
55                tmp=-a[j][0]/a[j][i];
56                if(t>tmp)t=tmp,l=j;
57            }
58            if(!l){
59                puts("Unbounded");
60                return vector<double>();
61            }
62            pivot(l,i);
63        }
64        vector<double>x;
65        for(int i=n+1;i<=n+m;i++)pt[Ans[i]]=i-n;
66        for(int i=1;i<=n;i++)x.push_back(pt[i]?a[pt[i]][0]:0);
67        return x;
68    }
69 }
```

### 0.2.4 自适应辛普森

```
1 double area(const double &left, const double &right) {
2     double mid = (left + right) / 2;
3     return (right - left) * (calc(left) + 4 * calc(mid) + calc(right)) / 6;
4 }
5
6 double simpson(const double &left, const double &right,
7                const double &eps, const double &area_sum) {
8     double mid = (left + right) / 2;
9     double area_left = area(left, mid);
10    double area_right = area(mid, right);
11    double area_total = area_left + area_right;
12    if (std::abs(area_total - area_sum) < 15 * eps) {
13        return area_total + (area_total - area_sum) / 15;
14    }
15    return simpson(left, mid, eps / 2, area_left)
16         + simpson(mid, right, eps / 2, area_right);
17 }
18
19 double simpson(const double &left, const double &right, const double &eps) {
20    return simpson(left, right, eps, area(left, right));
21 }
```

### 0.2.5 多项式求根

```
1 const double eps=1e-12;
2 double a[10][10];
3 typedef vector<double> vd;
4 int sgn(double x) { return x < -eps ? -1 : x > eps; }
5 double mypow(double x,int num){
6     double ans=1.0;
7     for(int i=1;i<=num;++i)ans*=x;
8     return ans;
9 }
10 double f(int n,double x){
11    double ans=0;
12    for(int i=n;i>=0;--i)ans+=a[n][i]*mypow(x,i);
13    return ans;
```

```
14 }
15 double getRoot(int n,double l,double r){
16     if(sgn(f(n,l))==0)return l;
17     if(sgn(f(n,r))==0)return r;
18     double temp;
19     if(sgn(f(n,l))>0)temp=-1;else temp=1;
20     double m;
21     for(int i=1;i<=10000;++i){
22         m=(l+r)/2;
23         double mid=f(n,m);
24         if(sgn(mid)==0){
25             return m;
26         }
27         if(mid*temp<0)l=m;else r=m;
28     }
29     return (l+r)/2;
30 }
31 vd did(int n){
32     vd ret;
33     if(n==1){
34         ret.push_back(-1e10);
35         ret.push_back(-a[n][0]/a[n][1]);
36         ret.push_back(1e10);
37         return ret;
38     }
39     vd mid=did(n-1);
40     ret.push_back(-1e10);
41     for(int i=0;i+1<mid.size();++i){
42         int t1=sgn(f(n,mid[i])),t2=sgn(f(n,mid[i+1]));
43         if(t1*t2>0)continue;
44         ret.push_back(getRoot(n,mid[i],mid[i+1]));
45     }
46     ret.push_back(1e10);
47     return ret;
48 }
49 int main(){
50     int n; scanf("%d",&n);
51     for(int i=n;i>=0;--i){
52         scanf("%lf",&a[n][i]);
53     }
54     for(int i=n-1;i>=0;--i)
55         for(int j=0;j<=i;++j)a[i][j]=a[i+1][j+1]*(j+1);
56     vd ans=did(n);
57     sort(ans.begin(),ans.end());
58     for(int i=1;i+1<ans.size();++i)printf("%.10f\n",ans[i]);
59     return 0;
60 }
```

## 0.3 数据结构
### 0.3.1 平衡的二叉查找树
**Treap**

```
1 #include<bits/stdc++.h>
2 using namespace std;
3 const int maxn=1e5+5;
4 #define sz(x) (x?x->siz:0)
5 struct Treap{
6     struct node{
7         int key,val;
8         int siz,s;
9         node *c[2];
10        node(int v=0){
11            val=v;
12            key=rand();
13            siz=1,s=1;
14            c[0]=c[1]=0;
15        }
16        void rz(){siz=s;if(c[0])siz+=c[0]->siz;if(c[1])siz+=c[1]->siz;}
```

```
17    }pool[maxn],*cur,*root;
18    Treap(){cur=pool;}
19    node* newnode(int val){return *cur=node(val),cur++;}
20    void rot(node *&t,int d){
21        if(!t->c[d])t=t->c[!d];
22        else{
23            node *p=t->c[d];t->c[d]=p->c[!d];
24            p->c[!d]=t;t->rz();p->rz();t=p;
25        }
26    }
27    void insert(node *&t,int x){
28        if(!t){t=newnode(x);return;}
29        if(t->val==x){t->s++;t->siz++;return;}
30        insert(t->c[x>t->val],x);
31        if(t->key<t->c[x>t->val]->key)
32            rot(t,x>t->val);
33        else t->rz();
34    }
35    void del(node *&t,int x){
36        if(!t)return;
37        if(t->val==x){
38            if(t->s>1){t->s--;t->siz--;return;}
39            if(!t->c[0]||!t->c[1]){
40                if(!t->c[0])t=t->c[1];
41                else t=t->c[0];
42                return;
43            }
44            int d=t->c[0]->key<t->c[1]->key;
45            rot(t,d);
46            del(t,x);
47            return;
48        }
49        del(t->c[x>t->val],x);
50        t->rz();
51    }
52    int pre(node *t,int x){
53        if(!t)return INT_MIN;
54        int ans=pre(t->c[x>t->val],x);
55        if(t->val<x)ans=max(ans,t->val);
56        return ans;
57    }
58    int nxt(node *t,int x){
59        if(!t)return INT_MAX;
60        int ans=nxt(t->c[x>=t->val],x);
61        if(t->val>x)ans=min(ans,t->val);
62        return ans;
63    }
64    int rank(node *t,int x){
65        if(!t)return 0;
66        if(t->val==x)return sz(t->c[0]);
67        if(t->val<x)return sz(t->c[0])+t->s+rank(t->c[1],x);
68        if(t->val>x)return rank(t->c[0],x);
69    }
70    int kth(node *t,int x){
71        if(sz(t->c[0])>=x)return kth(t->c[0],x);
72        if(sz(t->c[0])+t->s>=x)return t->val;
73        return kth(t->c[1],x-t->s-sz(t->c[0]));
74    }
75 }T;
```

### 0.3.2 坚固的数据结构
**坚固的平衡树**

```
1 #define sz(x) (x?x->siz:0)
2 struct node{
3     int siz,key;
4     LL val,sum;
```

```
5        LL mu,a,d;
6        node *c[2],*f;
7        void split(int ned,node *&p,node *&q);
8        node* rz(){
9            sum=val;siz=1;
10           if(c[0])sum+=c[0]->sum,siz+=c[0]->siz;
11           if(c[1])sum+=c[1]->sum,siz+=c[1]->siz;
12           return this;
13       }
14       void make(LL _mu,LL _a,LL _d){
15           sum=sum*_mu+_a*siz+_d*siz*(siz-1)/2;
16           val=val*_mu+_a+_d*sz(c[0]);
17           mu*=_mu;a=a*_mu+_a;d=d*_mu+_d;
18       }
19       void pd(){
20           if(mu==1&&a==0&&d==0)return;
21           if(c[0])c[0]->make(mu,a,d);
22           if(c[1])c[1]->make(mu,a+d+d*sz(c[0]),d);
23           mu=1;a=d=0;
24       }
25       node(){mu=1;}
26   }nd[maxn*2],*root;
27   node *merge(node *p,node *q){
28       if(!p||!q)return p?p->rz():(q?q->rz():0);
29       p->pd();q->pd();
30       if(p->key<q->key){
31           p->c[1]=merge(p->c[1],q);
32           return p->rz();
33       }else{
34           q->c[0]=merge(p,q->c[0]);
35           return q->rz();
36       }
37   }
38   void node::split(int ned,node *&p,node *&q){
39       if(!ned){p=0;q=this;return;}
40       if(ned==siz){p=this;q=0;return;}
41       pd();
42       if(sz(c[0])>=ned){
43           c[0]->split(ned,p,q);c[0]=0;rz();
44           q=merge(q,this);
45       }else{
46           c[1]->split(ned-sz(c[0])-1,p,q);c[1]=0;rz();
47           p=merge(this,p);
48       }
49   }
50   int main(){
51       for(int i=1;i<=n;i++){
52           nd[i].val=in();
53           nd[i].key=rand();
54           nd[i].rz();
55           root=merge(root,nd+i);
56       }
57   }
```

**坚固的字符串**
**坚固的左偏树**

```
1   int Merge(int x, int y){
2     if (x == 0 || y == 0) return x + y;
3     if (Heap[x].Key < Heap[y].Key) swap(x, y);
4     Heap[x].Ri = Merge(Heap[x].Ri, y);
5     if (Heap[Heap[x].Le].Dis < Heap[Heap[x].Ri].Dis) swap(Heap[x].Le, Heap[x].Ri);
6     if (Heap[x].Ri == 0) Heap[x].Dis = 0;
7     else Heap[x].Dis = Heap[Heap[x].Ri].Dis + 1;
8     return x;
9   }
10
```

```
11  for (int i = 0; i <= n; i++){
12    Heap[i].Le = Heap[i].Ri = 0;
13    Heap[i].Dis = 0;
14    Heap[i].Key = Cost[i];
15  }
16  Heap[0].Dis = -1;
```

### 0.3.3 树上的魔术师
**Link Cut Tree(zky)**

```
1   struct LCT{
2       struct node{
3           bool rev;
4           int mx,val;
5           node *f,*c[2];
6           bool d(){return this==f->c[1];}
7           bool rt(){return !f||(f->c[0]!=this&&f->c[1]!=this);}
8           void sets(node *x,int d){pd();if(x)x->f=this;c[d]=x;rz();}
9           void makerv(){rev^=1;swap(c[0],c[1]);}
10          void pd(){
11              if(rev){
12                  if(c[0])c[0]->makerv();
13                  if(c[1])c[1]->makerv();
14                  rev=0;
15              }
16          }
17          void rz(){
18              mx=val;
19              if(c[0])mx=max(mx,c[0]->mx);
20              if(c[1])mx=max(mx,c[1]->mx);
21          }
22      }nd[int(1e4)+1];
23      void rot(node *x){
24          node *y=x->f;if(!y->rt())y->f->pd();
25          y->pd();x->pd();bool d=x->d();
26          y->sets(x->c[!d],d);
27          if(y->rt())x->f=y->f;
28          else y->f->sets(x,y->d());
29          x->sets(y,!d);
30      }
31      void splay(node *x){
32          while(!x->rt())
33              if(x->f->rt())rot(x);
34              else if(x->d()==x->f->d())rot(x->f),rot(x);
35              else rot(x),rot(x);
36      }
37      node* access(node *x){
38          node *y=0;
39          for(;x;x=x->f){
40              splay(x);
41              x->sets(y,1);y=x;
42          }return y;
43      }
44      void makert(node *x){
45          access(x)->makerv();
46          splay(x);
47      }
48      void link(node *x,node *y){
49          makert(x);
50          x->f=y;
51          access(x);
52      }
53      void cut(node *x,node *y){
54          makert(x);access(y);splay(y);
55          y->c[0]=x->f=0;
56          y->rz();
```

```
57        }
58        void link(int x,int y){link(nd+x,nd+y);}
59        void cut(int x,int y){cut(nd+x,nd+y);}
60 }T;
```

**Link Cut Tree(Splay)**

```
 1 struct node{
 2     bool Rev;
 3     int c[2], fa;
 4 }T[N];
 5 inline void Rev(int x){
 6     if (!x) return;
 7     swap(T[x].c[0], T[x].c[1]);
 8     T[x].Rev ^= 1;
 9 }
10 inline void Lazy_Down(int x){
11     if (!x) return;
12     if (T[x].Rev) Rev(T[x].c[0]), Rev(T[x].c[1]), T[x].Rev = 0;
13 }
14 void Rotate(int x, int c){
15     int y = T[x].c[c];
16     int z = T[y].c[1 - c];
17     if (T[x].fa){
18         if (T[T[x].fa].c[0] == x) T[T[x].fa].c[0] = y;
19         else T[T[x].fa].c[1] = y;
20     }
21     T[z].fa = x; T[x].c[c] = z;
22     T[y].fa = T[x].fa; T[x].fa = y; T[y].c[1 - c] = x;
23     //Update(x);
24     //Update(y);
25 }
26 int stack[N], fx[N];
27 void Splay(int x){
28     int top = 0;
29     for (int  u = x; u; u = T[u].fa)
30         stack[++top] = u;
31     for (int i = top; i >= 1; i--)
32         Lazy_Down(stack[i]);
33     for (int i = 2; i <= top; i++)
34         if (T[stack[i]].c[0] == stack[i - 1]) fx[i] = 0;
35         else fx[i] = 1;
36     for (int i = 2; i <= top; i += 2){
37         if (i == top) Rotate(stack[i], fx[i]);
38         else {
39             if (fx[i] == fx[i + 1]){
40                 Rotate(stack[i + 1], fx[i + 1]);
41                 Rotate(stack[i], fx[i]);
42             } else {
43                 Rotate(stack[i], fx[i]);
44                 Rotate(stack[i + 1], fx[i + 1]);
45             }
46         }
47     }
48     if (x != stack[top]) Par[x] = Par[stack[top]], Par[stack[top]] = 0;
49     //if (fa == 0) Root = x;
50 }
51 inline int Access(int u){
52     int Nxt = 0;
53     while (u){
54         Splay(u);
55         if (T[u].c[1]){
56             T[T[u].c[1]].fa = 0;
57             Par[T[u].c[1]] = u;
58         }
59         T[u].c[1] = Nxt;
60         if (Nxt){
```

```
61             T[Nxt].fa = u;
62             Par[Nxt] = 0;
63         }
64         //Update(u)
65         Nxt = u;
66         u = Par[u];
67     }
68     return Nxt;
69 }
70 inline void Root(int u){
71     Access(u);
72     Splay(u);
73     Rev(u);
74 }
75 inline void Link(int u, int v){
76     Root(u);
77     Par[u] = v;
78 }
79 inline void Cut(int u, int v){
80     Access(u);
81     Splay(v);
82     if (Par[v] != u){
83         swap(u, v);
84         Access(u);
85         Splay(v);
86     }
87     Par[v] = 0;
88 }
89 inline int Find_Root(int x){
90     Access(x);
91     Splay(x);
92     int y = x;
93     while (T[y].c[0]){
94         Lazy_Down(y);
95         y = T[y].c[0];
96     }
97     return y;
98 }
```

### 0.3.4 可持久化线段树

```
 1 struct node1 {
 2     int L, R, Lson, Rson, Sum;
 3 } tree[N * 40];
 4 int root[N], a[N], b[N];
 5 int tot, n, m;
 6 int Real[N];
 7 int Same(int x) {
 8     ++tot;
 9     tree[tot] = tree[x];
10     return tot;
11 }
12 int build(int L, int R) {
13     ++tot;
14     tree[tot].L = L;
15     tree[tot].R = R;
16     tree[tot].Lson = tree[tot].Rson = tree[tot].Sum = 0;
17     if (L == R) return tot;
18     int s = tot;
19     int mid = (L + R) >> 1;
20     tree[s].Lson = build(L, mid);
21     tree[s].Rson = build(mid + 1, R);
22     return s;
23 }
24 int Ask(int Lst, int Cur, int L, int R, int k) {
25     if (L == R) return L;
26     int Mid = (L + R) >> 1;
27     int Left = tree[tree[Cur].Lson].Sum - tree[tree[Lst].Lson].Sum;
```

```
28        if (Left >= k) return Ask(tree[Lst].Lson, tree[Cur].Lson, L, Mid, k);
29        k -= Left;
30        return Ask(tree[Lst].Rson, tree[Cur].Rson, Mid + 1, R, k);
31 }
32 int Add(int Lst, int pos) {
33        int root = Same(Lst);
34        tree[root].Sum++;
35        if (tree[root].L == tree[root].R) return root;
36        int mid = (tree[root].L + tree[root].R) >> 1;
37        if (pos <= mid) tree[root].Lson = Add(tree[root].Lson, pos);
38        else tree[root].Rson = Add(tree[root].Rson, pos);
39        return root;
40 }
41 int main() {
42        scanf("%d%d", &n, &m);
43        int up = 0;
44        for (int i = 1; i <= n; i++){
45            scanf("%d", &a[i]);
46            b[i] = a[i];
47        }
48        sort(b + 1, b + n + 1);
49        up = unique(b + 1, b + n + 1) - b - 1;
50        for (int i = 1; i <= n; i++){
51            int tmp = lower_bound(b + 1, b + up + 1, a[i]) - b;
52            Real[tmp] = a[i];
53            a[i] = tmp;
54        }
55        tot = 0;
56        root[0] = build(1, up);
57        for (int i = 1; i <= n; i++){
58            root[i] = Add(root[i - 1], a[i]);
59        }
60        for (int i = 1; i <= m; i++){
61            int u, v, w;
62            scanf("%d%d%d", &u, &v, &w);
63            printf("%d\n", Real[Ask(root[u - 1], root[v], 1, up, w)]);
64        }
65        return 0;
66 }
```

### 0.3.5 k-d 树

```
1 long long norm(const long long &x) {
2     //    For manhattan distance
3     return std::abs(x);
4     //    For euclid distance
5     return x * x;
6 }
7
8 struct Point {
9     int x, y, id;
10
11    const int& operator [] (int index) const {
12        if (index == 0) {
13            return x;
14        } else {
15            return y;
16        }
17    }
18
19    friend long long dist(const Point &a, const Point &b) {
20        long long result = 0;
21        for (int i = 0; i < 2; ++i) {
22            result += norm(a[i] - b[i]);
23        }
24        return result;
25    }
26 } point[N];
27
```

```
28 struct Rectangle {
29     int min[2], max[2];
30
31     Rectangle() {
32         min[0] = min[1] = INT_MAX;
33         max[0] = max[1] = INT_MIN;
34     }
35
36     void add(const Point &p) {
37         for (int i = 0; i < 2; ++i) {
38             min[i] = std::min(min[i], p[i]);
39             max[i] = std::max(max[i], p[i]);
40         }
41     }
42
43     long long dist(const Point &p) {
44         long long result = 0;
45         for (int i = 0; i < 2; ++i) {
46             //    For minimum distance
47             result += norm(std::min(std::max(p[i], min[i]), max[i]) - p[i]);
48             //    For maximum distance
49             result += std::max(norm(max[i] - p[i]), norm(min[i] - p[i]));
50         }
51         return result;
52     }
53 };
54
55 struct Node {
56     Point seperator;
57     Rectangle rectangle;
58     int child[2];
59
60     void reset(const Point &p) {
61         seperator = p;
62         rectangle = Rectangle();
63         rectangle.add(p);
64         child[0] = child[1] = 0;
65     }
66 } tree[N << 1];
67
68 int size, pivot;
69
70 bool compare(const Point &a, const Point &b) {
71     if (a[pivot] != b[pivot]) {
72         return a[pivot] < b[pivot];
73     }
74     return a.id < b.id;
75 }
76
77 int build(int l, int r, int type = 1) {
78     pivot = type;
79     if (l >= r) {
80         return 0;
81     }
82     int x = ++size;
83     int mid = l + r >> 1;
84     std::nth_element(point + l, point + mid, point + r, compare);
85     tree[x].reset(point[mid]);
86     for (int i = l; i < r; ++i) {
87         tree[x].rectangle.add(point[i]);
88     }
89     tree[x].child[0] = build(l, mid, type ^ 1);
90     tree[x].child[1] = build(mid + 1, r, type ^ 1);
91     return x;
92 }
93
94 int insert(int x, const Point &p, int type = 1) {
95     pivot = type;
96     if (x == 0) {
```

```cpp
 97            tree[++size].reset(p);
 98            return size;
 99        }
100        tree[x].rectangle.add(p);
101        if (compare(p, tree[x].seperator)) {
102            tree[x].child[0] = insert(tree[x].child[0], p, type ^ 1);
103        } else {
104            tree[x].child[1] = insert(tree[x].child[1], p, type ^ 1);
105        }
106        return x;
107    }
108
109    //    For minimum distance
110    void query(int x, const Point &p, std::pair<long long, int> &answer, int type =
        ↪ 1) {
111        pivot = type;
112        if (x == 0 || tree[x].rectangle.dist(p) > answer.first) {
113            return;
114        }
115        answer = std::min(answer,
116                std::make_pair(dist(tree[x].seperator, p), tree[x].seperator.id));
117        if (compare(p, tree[x].seperator)) {
118            query(tree[x].child[0], p, answer, type ^ 1);
119            query(tree[x].child[1], p, answer, type ^ 1);
120        } else {
121            query(tree[x].child[1], p, answer, type ^ 1);
122            query(tree[x].child[0], p, answer, type ^ 1);
123        }
124    }
125
126    std::priority_queue<std::pair<long long, int> > answer;
127
128    void query(int x, const Point &p, int k, int type = 1) {
129        pivot = type;
130        if (x == 0 ||
131            (int)answer.size() == k && tree[x].rectangle.dist(p) >
                ↪ answer.top().first) {
132            return;
133        }
134        answer.push(std::make_pair(dist(tree[x].seperator, p),
            ↪ tree[x].seperator.id));
135        if ((int)answer.size() > k) {
136            answer.pop();
137        }
138        if (compare(p, tree[x].seperator)) {
139            query(tree[x].child[0], p, k, type ^ 1);
140            query(tree[x].child[1], p, k, type ^ 1);
141        } else {
142            query(tree[x].child[1], p, k, type ^ 1);
143            query(tree[x].child[0], p, k, type ^ 1);
144        }
145    }
```

### 0.3.6 莫队算法

```cpp
 1 struct node{
 2     int l, r, id;
 3     friend bool operator < (const node &a, const node &b){
 4         if (a.l / Block == b.l / Block) return a.r < b.r;
 5         return a.l / Block < b.l / Block;
 6     }
 7 }q[N];
 8 Block = int(sqrt(n));
 9 for (int i = 1; i <= m; i++){
10     scanf("%d%d", &q[i].l, &q[i].r);
11     q[i].id = i;
12 }
```

```cpp
13 sort(q + 1, q + 1 + m);
14 Cur = a[1]; /// Hints: adjust by yourself
15 Le = Ri = 1;
16 for (int i = 1; i <= m; i++){
17     while (q[i].r > Ri) Ri++, ChangeRi(1, Le, Ri);
18     while (q[i].l > Le) ChangeLe(-1, Le, Ri), Le++;
19     while (q[i].l < Le) Le--, ChangeLe(1, Le, Ri);
20     while (q[i].r < Ri) ChangeRi(-1, Le, Ri), Ri--;
21     Ans[q[i].id] = Cur;
22 }
```

### 0.3.7 树状数组 kth

```cpp
1 int find(int k){
2     int cnt=0,ans=0;
3     for(int i=22;i>=0;i--){
4         ans+=(1<<i);
5         if(ans>n || cnt+d[ans]>=k)ans-=(1<<i);
6         else cnt+=d[ans];
7     }
8     return ans+1;
9 }
```

### 0.3.8 虚树

```cpp
 1 int a[maxn*2],sta[maxn*2];
 2 int top=0,k;
 3 void build(){
 4     top=0;
 5     sort(a,a+k,bydfn);
 6     k=unique(a,a+k)-a;
 7     sta[top++]=1;_n=k;
 8     for(int i=0;i<k;i++){
 9         int LCA=lca(a[i],sta[top-1]);
10         while(dep[LCA]<dep[sta[top-1]]){
11             if(dep[LCA]>=dep[sta[top-2]]){
12                 add_edge(LCA,sta[--top]);
13                 if(sta[top-1]!=LCA)sta[top++]=LCA;
14                 break;
15             }add_edge(sta[top-2],sta[top-1]);top--;
16         }if(sta[top-1]!=a[i])sta[top++]=a[i];
17     }
18     while(top>1)
19         add_edge(sta[top-2],sta[top-1]),top--;
20     for(int i=0;i<k;i++)inr[a[i]]=1;
21 }
```

### 0.3.9 点分治 (zky)

```cpp
 1 int siz[maxn],f[maxn],dep[maxn],cant[maxn],root,All,d[maxn];
 2 void makert(int u,int fa){
 3     siz[u]=1;f[u]=0;
 4     for(int i=0;i<G[u].size();i++){
 5         edge e=G[u][i];
 6         if(e.v!=fa&&!cant[e.v]){
 7             dep[e.v]=dep[u]+1;
 8             makert(e.v,u);
 9             siz[u]+=siz[e.v];
10             f[u]=max(f[u],siz[e.v]);
11         }
12     }f[u]=max(f[u],All-f[u]);
13     if(f[root]>f[u])root=u;
14 }
15 void dfs(int u,int fa){
16     //Gain data
17     for(int i=0;i<G[u].size();i++){
```

```
18          edge e=G[u][i];
19          if(e.v==fa||cant[e.v])continue;
20          d[e.v]=d[u]+e.w;
21          dfs(e.v,u);
22      }
23 }
24 void calc(int u){
25      d[u]=0;
26      for(int i=0;i<G[u].size();i++){
27          edge e=G[u][i];
28          if(cant[e.v])continue;
29          d[e.v]=e.w;
30          dfs(e.v,u);
31      }
32  }
33 }
34 void solve(int u){
35      calc(u);cant[u]=1;
36      for(int i=0;i<G[u].size();i++){
37          edge e=G[u][i];
38          if(cant[e.v])continue;
39          All=siz[e.v];
40          f[root=0]=n+1;
41          makert(e.v,0);
42          solve(root);
43      }
44 }
45 All=n
46 f[root=0]=n+1;
47 makert(1,1);
48 solve(root);
```

### 0.3.10  元芳树

```
1 void tarjan(int u){
2      dfn[u]=low[u]=++tot;
3      for(int i=0;i<G[u].size();i++){
4          edge e=G[u][i];
5          if(dfn[e.v])
6              low[u]=min(low[u],dfn[e.v]);
7          else{
8              S.push(e);
9              tarjan(e.v);
10             if(low[e.v]==dfn[u]){
11
12                 if(S.top()==e){
13                     fa[e.v][0]=u;
14                     fw[e.v]=e.w;
15                     S.pop();
16                     continue;
17                 }
18
19                 Rcnt++;
20                 edge ed;
21                 do{
22                     ed=S.top();S.pop();
23                     ring[Rcnt].push_back(ed);
24                 }while(ed!=e);
25                 reverse(ring[Rcnt].begin(),ring[Rcnt].end());
26                 int last=ring[Rcnt].back().v;
27                 ring[Rcnt].push_back((edge){last,u,Mw[pack(last,u)]});
28             }
29             low[u]=min(low[u],low[e.v]);
30         }
31     }
32 }
33 void up(int u){
34     if(dep[u]||u==1)return ;
```

```
35     if(fa[u][0])up(fa[u][0]);
36     dep[u]=dep[fa[u][0]]+1;
37     fw[u]+=fw[fa[u][0]];
38 }
39 void build(){
40     S.push((edge){0,1,0});
41     tarjan(1);
42
43     for(int i=1;i<=Rcnt;i++){
44         rlen[i]=0;
45         sum[i].resize(ring[i].size());
46         dis[i].resize(ring[i].size());
47         for(int j=0;j<ring[i].size();j++){
48             rlen[i]+=ring[i][j].w;
49             ind[i].push_back(make_pair(ring[i][j].u,j));
50         }
51         sum[i][0]=0;
52         fw[i+n]=0;
53         fa[i+n][0]=ring[i][0].u;
54         for(int j=1;j<ring[i].size();j++){
55             sum[i][j]=sum[i][j-1]+ring[i][j-1].w;
56             dis[i][j]=min(sum[i][j],rlen[i]-sum[i][j]);
57             fw[ring[i][j].u]=dis[i][j];
58             fa[ring[i][j].u][0]=i+n;
59         }
60         sort(ind[i].begin(),ind[i].end());
61     }
62
63     for(int i=1;i<=n+Rcnt;i++)
64         up(i);
65
66     for(int j=1;j<BIT;j++)
67         for(int i=1;i<=n+Rcnt;i++)if(fa[i][j-1])
68             fa[i][j]=fa[fa[i][j-1]][j-1];
69
70 }
71 pair<int,int>second_lca;
72 int lca(int u,int v){
73     if(dep[u]<dep[v])swap(u,v);
74     int d=dep[u]-dep[v];
75     for(int i=0;i<BIT;i++)if(d>>i&1)
76         u=fa[u][i];
77     if(u==v)return u;
78     for(int i=BIT-1;i>=0;i--)if(fa[u][i]!=fa[v][i]){
79         u=fa[u][i];
80         v=fa[v][i];
81     }
82     second_lca=make_pair(u,v);
83     return fa[u][0];
84 }
```

## 0.4  图论

### 0.4.1  点双连通分量 (lyx)

```
1 void Dfs(int x, int lst){
2     dfn[x] = ++dfc;
3     low[x] = dfn[x];
4     stack[++cnt] = x;
5     int son = 0;
6
7     for (int i = g[x]; i; i = nxt[i]){
8         if (!dfn[adj[i]]){
9             ++son;
10            Dfs(adj[i], i);
11            low[x] = min(low[x], low[adj[i]]);
12            if (low[adj[i]] >= dfn[x]){
```

```
13              int Tmp;
14              iscut[x] = 1;
15              ++block;
16              E[x].push_back(block + n);
17              do{
18                  Tmp = stack[cnt --];
19                  belong[Tmp] = block + n;
20
21                  E[Tmp].push_back(block + n);
22              }while (Tmp != adj[i]);
23          }
24      }
25      else
26          if ((i ^ lst) != 1) low[x] = min(low[x], dfn[adj[i]]);
27  }
28  if (x == Root && son == 1) iscut[x] = 0, belong[x] = E[x][0];
29  if (x == Root && son == 0){
30      ++block;
31      belong[x] = block + n;
32  }
33 }
34  tot = 1;//!!!!!!!!!!!!!!!!!!!!!!!!!!!1
35  block = 0;
36  cnt = 0;
37  dfc = 0;
38  for (int i = 1; i <= n; i++)
39      if (dfn[i] == 0){
40          Root = i;
41          Dfs(i, 0);
42      }
```

### 0.4.2  2-SAT 问题 (强连通分量)

```
1 int stamp, comps, top;
2 int dfn[N], low[N], comp[N], stack[N];
3
4 void add(int x, int a, int y, int b) {
5     edge[x << 1 | a].push_back(y << 1 | b);
6 }
7
8 void tarjan(int x) {
9     dfn[x] = low[x] = ++stamp;
10    stack[top++] = x;
11    for (int i = 0; i < (int)edge[x].size(); ++i) {
12        int y = edge[x][i];
13        if (!dfn[y]) {
14            tarjan(y);
15            low[x] = std::min(low[x], low[y]);
16        } else if (!comp[y]) {
17            low[x] = std::min(low[x], dfn[y]);
18        }
19    }
20    if (low[x] == dfn[x]) {
21        comps++;
22        do {
23            int y = stack[--top];
24            comp[y] = comps;
25        } while (stack[top] != x);
26    }
27 }
28
29 bool solve() {
30    int counter = n + n + 1;
31    stamp = top = comps = 0;
32    std::fill(dfn, dfn + counter, 0);
33    std::fill(comp, comp + counter, 0);
34    for (int i = 0; i < counter; ++i) {
35        if (!dfn[i]) {
```

```
36            tarjan(i);
37        }
38    }
39    for (int i = 0; i < n; ++i) {
40        if (comp[i << 1] == comp[i << 1 | 1]) {
41            return false;
42        }
43        answer[i] = (comp[i << 1 | 1] < comp[i << 1]);
44    }
45    return true;
46 }
```

### 0.4.3  二分图最大匹配
**Hungary 算法**
> 时间复杂度：$\mathcal{O}(V \cdot E)$

```
1 vector<int>G[maxn];
2 int Link[maxn],vis[maxn],T;
3 bool find(int x){
4     for(int i=0;i<G[x].size();i++){
5         int v=G[x][i];
6         if(vis[v]==T)continue;
7         vis[v]=T;
8         if(!Link[v]||find(Link[v])){
9             Link[v]=x;
10            return 1;
11        }
12    }return 0;
13 }
14 int Hungarian(int n){
15    int ans=0;
16    memset(Link,0,sizeof Link);
17    for(int i=1;i<=n;i++){
18        T++;
19        ans+=find(i);
20    }return ans;
21 }
```

**Hopcroft Karp 算法**
> 时间复杂度：$\mathcal{O}(\sqrt{V} \cdot E)$

```
1 int matchx[N], matchy[N], level[N];
2
3 bool dfs(int x) {
4     for (int i = 0; i < (int)edge[x].size(); ++i) {
5         int y = edge[x][i];
6         int w = matchy[y];
7         if (w == -1 || level[x] + 1 == level[w] && dfs(w)) {
8             matchx[x] = y;
9             matchy[y] = x;
10            return true;
11        }
12    }
13    level[x] = -1;
14    return false;
15 }
16
17 int solve() {
18    std::fill(matchx, matchx + n, -1);
19    std::fill(matchy, matchy + m, -1);
20    for (int answer = 0; ; ) {
21        std::vector<int> queue;
22        for (int i = 0; i < n; ++i) {
23            if (matchx[i] == -1) {
24                level[i] = 0;
```

```
25                 queue.push_back(i);
26             } else {
27                 level[i] = -1;
28             }
29         }
30         for (int head = 0; head < (int)queue.size(); ++head) {
31             int x = queue[head];
32             for (int i = 0; i < (int)edge[x].size(); ++i) {
33                 int y = edge[x][i];
34                 int w = matchy[y];
35                 if (w != -1 && level[w] < 0) {
36                     level[w] = level[x] + 1;
37                     queue.push_back(w);
38                 }
39             }
40         }
41         int delta = 0;
42         for (int i = 0; i < n; ++i) {
43             if (matchx[i] == -1 && dfs(i)) {
44                 delta++;
45             }
46         }
47         if (delta == 0) {
48             return answer;
49         } else {
50             answer += delta;
51         }
52     }
53 }
```

### 0.4.4 二分图最大权匹配
时间复杂度：$\mathcal{O}(V^4)$

```
1 int labelx[N], labely[N], match[N], slack[N];
2 bool visitx[N], visity[N];
3
4 bool dfs(int x) {
5     visitx[x] = true;
6     for (int y = 0; y < n; ++y) {
7         if (visity[y]) {
8             continue;
9         }
10         int delta = labelx[x] + labely[y] - graph[x][y];
11         if (delta == 0) {
12             visity[y] = true;
13             if (match[y] == -1 || dfs(match[y])) {
14                 match[y] = x;
15                 return true;
16             }
17         } else {
18             slack[y] = std::min(slack[y], delta);
19         }
20     }
21     return false;
22 }
23
24 int solve() {
25     for (int i = 0; i < n; ++i) {
26         match[i] = -1;
27         labelx[i] = INT_MIN;
28         labely[i] = 0;
29         for (int j = 0; j < n; ++j) {
30             labelx[i] = std::max(labelx[i], graph[i][j]);
31         }
32     }
33     for (int i = 0; i < n; ++i) {
34         while (true) {
```

```
35             std::fill(visitx, visitx + n, 0);
36             std::fill(visity, visity + n, 0);
37             for (int j = 0; j < n; ++j) {
38                 slack[j] = INT_MAX;
39             }
40             if (dfs(i)) {
41                 break;
42             }
43             int delta = INT_MAX;
44             for (int j = 0; j < n; ++j) {
45                 if (!visity[j]) {
46                     delta = std::min(delta, slack[j]);
47                 }
48             }
49             for (int j = 0; j < n; ++j) {
50                 if (visitx[j]) {
51                     labelx[j] -= delta;
52                 }
53                 if (visity[j]) {
54                     labely[j] += delta;
55                 } else {
56                     slack[j] -= delta;
57                 }
58             }
59         }
60     }
61     int answer = 0;
62     for (int i = 0; i < n; ++i) {
63         answer += graph[match[i]][i];
64     }
65     return answer;
66 }
```

### 0.4.5 最大流 (dinic)
时间复杂度：$\mathcal{O}(V^2 \cdot E)$

```
1 struct edge{int u,v,cap,flow;};
2 vector<edge>edges;
3 vector<int>G[maxn];
4 int s,t;
5 int cur[maxn],d[maxn];
6 void add(int u,int v,int cap){
7     edges.push_back((edge){u,v,cap,0});
8     G[u].push_back(edges.size()-1);
9     edges.push_back((edge){v,u,0,0});
10     G[v].push_back(edges.size()-1);
11 }
12 bool bfs(){
13     static int vis[maxn];
14     memset(vis,0,sizeof vis);vis[s]=1;
15     queue<int>q;q.push(s);d[s]=0;
16     while(!q.empty()){
17         int u=q.front();q.pop();
18         for(int i=0;i<G[u].size();i++){
19             edge e=edges[G[u][i]];if(vis[e.v]||e.cap==e.flow)continue;
20             d[e.v]=d[u]+1;vis[e.v]=1;q.push(e.v);
21         }
22     }return vis[t];
23 }
24 int dfs(int u,int a){
25     if(u==t||!a)return a;
26     int flow=0,f;
27     for(int &i=cur[u];i<G[u].size();i++){
28         edge e=edges[G[u][i]];
```

```
29        if(d[e.v]==d[u]+1&&(f=dfs(e.v,min(a,e.cap-e.flow)))>0){
30            edges[G[u][i]].flow+=f;
31            edges[G[u][i]^1].flow-=f;
32            flow+=f;a-=f;if(!a)break;
33        }
34    }return flow;
35 }
36 int dinic(){
37    int flow=0,x;
38    while(bfs()){
39        memset(cur,0,sizeof cur);
40        while(x=dfs(s,INT_MAX)){
41            flow+=x;
42            memset(cur,0,sizeof cur);
43        }
44    }return flow;
45 }
```

### 0.4.6 最大流 (sap)

时间复杂度：$\mathcal{O}(V^2 \cdot E)$

```
1  int g[T], adj[M], nxt[M], f[M];
2  int cnt[T], dist[T], cur[T], fa[T], dat[T];
3  void Ins(int x, int y, int ff, int rf){
4      adj[++tot] = y; nxt[tot] = g[x]; g[x] = tot; f[tot] = ff;
5      adj[++tot] = x; nxt[tot] = g[y]; g[y] = tot; f[tot] = rf;
6  }
7  int sap(int s, int t){
8      int x, sum;
9      for (int i = 1; i <= t; i++){
10         dist[i] = 1;
11         cur[i] = g[i];
12         fa[i] = 0;
13         dat[i] = 0;
14         cnt[i] = 0;
15     }
16     cnt[0] = 1; cnt[1] = t - 1;
17     dist[t] = 0;
18     dat[s] = INF;
19     x = s;
20     sum = 0;
21     while (1){
22         int p;
23         for (p = cur[x]; p; p = nxt[p]){
24             if (f[p] > 0 && dist[adj[p]] == dist[x] - 1) break;
25         }
26         if (p > 0){
27             cur[x] = p;
28             fa[adj[p]] = p;
29             dat[adj[p]] = min(dat[x], f[p]);
30             x = adj[p];
31             if (x == t){
32                 sum += dat[x];
33                 while (x != s){
34                     f[fa[x]] -= dat[t];
35                     f[fa[x] ^ 1] += dat[t];
36                     x = adj[fa[x] ^ 1];
37                 }
38             }
39         } else {
40             cnt[dist[x]] --;
41             if (cnt[dist[x]] == 0) return sum;
42             dist[x] = t + 1;
43             for (int p = g[x]; p; p = nxt[p]){
44                 if (f[p] > 0 && dist[adj[p]] + 1 < dist[x]){
45                     dist[x] = dist[adj[p]] + 1;
```

```
46                     cur[x] = p;
47                 }
48             }
49             cnt[dist[x]]++;
50             if (dist[s] > t) return sum;
51             if (x != s) x = adj[fa[x] ^ 1];
52         }
53     }
54 }
55 /*
56 tot = 1
57 edges' id start from 2
58 remember to clean g
59 t is the number of points
60 */
```

### 0.4.7 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 $\infty$，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 $\infty$，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 $\infty$，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 $\infty$ 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。

### 0.4.8 最小费用最大流
**稀疏图**

时间复杂度：$\mathcal{O}(V \cdot E^2)$

```
1  struct EdgeList {
2      int size;
3      int last[N];
4      int succ[M], other[M], flow[M], cost[M];
5      void clear(int n) {
6          size = 0;
7          std::fill(last, last + n, -1);
8      }
9      void add(int x, int y, int c, int w) {
10         succ[size] = last[x];
11         last[x] = size;
12         other[size] = y;
13         flow[size] = c;
14         cost[size++] = w;
15     }
16 } e;
```

```
17
18  int n, source, target;
19  int prev[N];
20
21  void add(int x, int y, int c, int w) {
22      e.add(x, y, c, w);
23      e.add(y, x, 0, -w);
24  }
25
26  bool augment() {
27      static int dist[N], occur[N];
28      std::vector<int> queue;
29      std::fill(dist, dist + n, INT_MAX);
30      std::fill(occur, occur + n, 0);
31      dist[source] = 0;
32      occur[source] = true;
33      queue.push_back(source);
34      for (int head = 0; head < (int)queue.size(); ++head) {
35          int x = queue[head];
36          for (int i = e.last[x]; ~i; i = e.succ[i]) {
37              int y = e.other[i];
38              if (e.flow[i] && dist[y] > dist[x] + e.cost[i]) {
39                  dist[y] = dist[x] + e.cost[i];
40                  prev[y] = i;
41                  if (!occur[y]) {
42                      occur[y] = true;
43                      queue.push_back(y);
44                  }
45              }
46          }
47          occur[x] = false;
48      }
49      return dist[target] < INT_MAX;
50  }
51
52  std::pair<int, int> solve() {
53      std::pair<int, int> answer = std::make_pair(0, 0);
54      while (augment()) {
55          int number = INT_MAX;
56          for (int i = target; i != source; i = e.other[prev[i] ^ 1]) {
57              number = std::min(number, e.flow[prev[i]]);
58          }
59          answer.first += number;
60          for (int i = target; i != source; i = e.other[prev[i] ^ 1]) {
61              e.flow[prev[i]] -= number;
62              e.flow[prev[i] ^ 1] += number;
63              answer.second += number * e.cost[prev[i]];
64          }
65      }
66      return answer;
67  }
```

**稠密图**

使用条件：费用非负

时间复杂度：$\mathcal{O}(V \cdot E^2)$

```
1   struct EdgeList {
2       int size;
3       int last[N];
4       int succ[M], other[M], flow[M], cost[M];
5       void clear(int n) {
6           size = 0;
7           std::fill(last, last + n, -1);
8       }
9       void add(int x, int y, int c, int w) {
10          succ[size] = last[x];
11          last[x] = size;
12          other[size] = y;
13          flow[size] = c;
14          cost[size++] = w;
15      }
16  } e;
17
18  int n, source, target, flow, cost;
19  int slack[N], dist[N];
20  bool visit[N];
21
22  void add(int x, int y, int c, int w) {
23      e.add(x, y, c, w);
24      e.add(y, x, 0, -w);
25  }
26
27  bool relabel() {
28      int delta = INT_MAX;
29      for (int i = 0; i < n; ++i) {
30          if (!visit[i]) {
31              delta = std::min(delta, slack[i]);
32          }
33          slack[i] = INT_MAX;
34      }
35      if (delta == INT_MAX) {
36          return true;
37      }
38      for (int i = 0; i < n; ++i) {
39          if (visit[i]) {
40              dist[i] += delta;
41          }
42      }
43      return false;
44  }
45
46  int dfs(int x, int answer) {
47      if (x == target) {
48          flow += answer;
49          cost += answer * (dist[source] - dist[target]);
50          return answer;
51      }
52      visit[x] = true;
53      int delta = answer;
54      for (int i = e.last[x]; ~i; i = e.succ[i]) {
55          int y = e.other[i];
56          if (e.flow[i] > 0 && !visit[y]) {
57              if (dist[y] + e.cost[i] == dist[x]) {
58                  int number = dfs(y, std::min(e.flow[i], delta));
59                  e.flow[i] -= number;
60                  e.flow[i ^ 1] += number;
61                  delta -= number;
62                  if (delta == 0) {
63                      dist[x] = INT_MIN;
64                      return answer;
65                  }
66              } else {
67                  slack[y] = std::min(slack[y], dist[y] + e.cost[i] - dist[x]);
68              }
69          }
70      }
71      return answer - delta;
72  }
73
74  std::pair<int, int> solve() {
75      flow = cost = 0;
76      std::fill(dist, dist + n, 0);
77      do {
78          do {
79              fill(visit, visit + n, 0);
```

```
80          } while (dfs(source, INT_MAX));
81      } while (!relabel());
82      return std::make_pair(flow, cost);
83 }
```

### 0.4.9 一般图最大匹配
时间复杂度：$\mathcal{O}(V^3)$

```
 1 int match[N], belong[N], next[N], mark[N], visit[N];
 2 std::vector<int> queue;
 3
 4 int find(int x) {
 5     if (belong[x] != x) {
 6         belong[x] = find(belong[x]);
 7     }
 8     return belong[x];
 9 }
10
11 void merge(int x, int y) {
12     x = find(x);
13     y = find(y);
14     if (x != y) {
15         belong[x] = y;
16     }
17 }
18
19 int lca(int x, int y) {
20     static int stamp = 0;
21     stamp++;
22     while (true) {
23         if (x != -1) {
24             x = find(x);
25             if (visit[x] == stamp) {
26                 return x;
27             }
28             visit[x] = stamp;
29             if (match[x] != -1) {
30                 x = next[match[x]];
31             } else {
32                 x = -1;
33             }
34         }
35         std::swap(x, y);
36     }
37 }
38
39 void group(int a, int p) {
40     while (a != p) {
41         int b = match[a], c = next[b];
42         if (find(c) != p) {
43             next[c] = b;
44         }
45         if (mark[b] == 2) {
46             mark[b] = 1;
47             queue.push_back(b);
48         }
49         if (mark[c] == 2) {
50             mark[c] = 1;
51             queue.push_back(c);
52         }
53         merge(a, b);
54         merge(b, c);
55         a = c;
56     }
57 }
58
59 void augment(int source) {
```

```
 60     queue.clear();
 61     for (int i = 0; i < n; ++i) {
 62         next[i] = visit[i] = -1;
 63         belong[i] = i;
 64         mark[i] = 0;
 65     }
 66     mark[source] = 1;
 67     queue.push_back(source);
 68     for (int head = 0; head < (int)queue.size() && match[source] == -1; ++head) {
 69         int x = queue[head];
 70         for (int i = 0; i < (int)edge[x].size(); ++i) {
 71             int y = edge[x][i];
 72             if (match[x] == y || find(x) == find(y) || mark[y] == 2) {
 73                 continue;
 74             }
 75             if (mark[y] == 1) {
 76                 int r = lca(x, y);
 77                 if (find(x) != r) {
 78                     next[x] = y;
 79                 }
 80                 if (find(y) != r) {
 81                     next[y] = x;
 82                 }
 83                 group(x, r);
 84                 group(y, r);
 85             } else if (match[y] == -1) {
 86                 next[y] = x;
 87                 for (int u = y; u != -1; ) {
 88                     int v = next[u];
 89                     int mv = match[v];
 90                     match[v] = u;
 91                     match[u] = v;
 92                     u = mv;
 93                 }
 94                 break;
 95             } else {
 96                 next[y] = x;
 97                 mark[y] = 2;
 98                 mark[match[y]] = 1;
 99                 queue.push_back(match[y]);
100             }
101         }
102     }
103 }
104
105 int solve() {
106     std::fill(match, match + n, -1);
107     for (int i = 0; i < n; ++i) {
108         if (match[i] == -1) {
109             augment(i);
110         }
111     }
112     int answer = 0;
113     for (int i = 0; i < n; ++i) {
114         answer += (match[i] != -1);
115     }
116     return answer;
117 }
```

### 0.4.10 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

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

```
3
4  int solve(int n) {
5      int answer = INT_MAX;
6      for (int i = 0; i < n; ++i) {
7          node[i] = i;
8      }
9      while (n > 1) {
10         int max = 1;
11         for (int i = 0; i < n; ++i) {
12             dist[node[i]] = graph[node[0]][node[i]];
13             if (dist[node[i]] > dist[node[max]]) {
14                 max = i;
15             }
16         }
17         int prev = 0;
18         memset(visit, 0, sizeof(visit));
19         visit[node[0]] = true;
20         for (int i = 1; i < n; ++i) {
21             if (i == n - 1) {
22                 answer = std::min(answer, dist[node[max]]);
23                 for (int k = 0; k < n; ++k) {
24                     graph[node[k]][node[prev]] =
25                         (graph[node[prev]][node[k]] += graph[node[k]][node[max]]);
26                 }
27                 node[max] = node[--n];
28             }
29             visit[node[max]] = true;
30             prev = max;
31             max = -1;
32             for (int j = 1; j < n; ++j) {
33                 if (!visit[node[j]]) {
34                     dist[node[j]] += graph[node[prev]][node[j]];
35                     if (max == -1 || dist[node[max]] < dist[node[j]]) {
36                         max = j;
37                     }
38                 }
39             }
40         }
41     }
42     return answer;
43 }
```

### 0.4.11 哈密尔顿回路（ORE 性质的图）

ORE 性质：

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

返回结果：从顶点 1 出发的一个哈密尔顿回路

使用条件：$n \geq 3$

```
1  int left[N], right[N], next[N], last[N];
2
3  void cover(int x) {
4      left[right[x]] = left[x];
5      right[left[x]] = right[x];
6  }
7
8  int adjacent(int x) {
9      for (int i = right[0]; i <= n; i = right[i]) {
10         if (graph[x][i]) {
11             return i;
12         }
13     }
14     return 0;
15 }
16
17 std::vector<int> solve() {
18     for (int i = 1; i <= n; ++i) {
```

```
19         left[i] = i - 1;
20         right[i] = i + 1;
21     }
22     int head, tail;
23     for (int i = 2; i <= n; ++i) {
24         if (graph[1][i]) {
25             head = 1;
26             tail = i;
27             cover(head);
28             cover(tail);
29             next[head] = tail;
30             break;
31         }
32     }
33     while (true) {
34         int x;
35         while (x = adjacent(head)) {
36             next[x] = head;
37             head = x;
38             cover(head);
39         }
40         while (x = adjacent(tail)) {
41             next[tail] = x;
42             tail = x;
43             cover(tail);
44         }
45         if (!graph[head][tail]) {
46             for (int i = head, j; i != tail; i = next[i]) {
47                 if (graph[head][next[i]] && graph[tail][i]) {
48                     for (j = head; j != i; j = next[j]) {
49                         last[next[j]] = j;
50                     }
51                     j = next[head];
52                     next[head] = next[i];
53                     next[tail] = i;
54                     tail = j;
55                     for (j = i; j != head; j = last[j]) {
56                         next[j] = last[j];
57                     }
58                     break;
59                 }
60             }
61         }
62         next[tail] = head;
63         if (right[0] > n) {
64             break;
65         }
66         for (int i = head; i != tail; i = next[i]) {
67             if (adjacent(i)) {
68                 head = next[i];
69                 tail = i;
70                 next[tail] = 0;
71                 break;
72             }
73         }
74     }
75     std::vector<int> answer;
76     for (int i = head; ; i = next[i]) {
77         if (i == 1) {
78             answer.push_back(i);
79             for (int j = next[i]; j != i; j = next[j]) {
80                 answer.push_back(j);
81             }
82             answer.push_back(i);
83             break;
84         }
85         if (i == tail) {
86             break;
```

```
87        }
88    }
89    return answer;
90 }
```

### 0.4.12  必经点树

```
1 vector<int>G[maxn],rG[maxn],dom[maxn];
2 int n,m;
3 int dfn[maxn],rdfn[maxn],dfs_c,semi[maxn],idom[maxn],fa[maxn];
4 struct ufsets{
5     int fa[maxn],best[maxn];
6     int find(int x){
7         if(fa[x]==x)
8             return x;
9         int f=find(fa[x]);
10        if(dfn[semi[best[x]]]>dfn[semi[best[fa[x]]]])
11            best[x]=best[fa[x]];
12        fa[x]=f;
13        return f;
14    }
15    int getbest(int x){
16        find(x);
17        return best[x];
18    }
19    void init(){
20        for(int i=1;i<=n;i++)
21            fa[i]=best[i]=i;
22    }
23 }uf;
24 void init(){
25    uf.init();
26    for(int i=1;i<=n;i++){
27        semi[i]=i;
28        idom[i]=0;
29        fa[i]=0;
30        dfn[i]=rdfn[i]=0;
31    }
32    dfs_c=0;
33 }
34 void dfs(int u){
35    dfn[u]=++dfs_c;
36    rdfn[dfn[u]]=u;
37    for(int i=0;i<G[u].size();i++){
38        int v=G[u][i];
39        if(!dfn[v]){
40            fa[v]=u;
41            dfs(v);
42        }
43    }
44 }
45
46 void tarjan(){
47    for(int i=n;i>1;i--){
48        int tmp=1e9;
49        int y=rdfn[i];
50        for(int i=0;i<rG[y].size();i++){
51            int x=rG[y][i];
52            tmp=min(tmp,dfn[semi[uf.getbest(x)]]);
53        }
54        semi[y]=rdfn[tmp];
55        int x=fa[y];
56        dom[semi[y]].push_back(y);
57        uf.fa[y]=x;
58        for(int i=0;i<dom[x].size();i++){
59            int z=dom[x][i];
60            if(dfn[semi[uf.getbest(z)]]<dfn[x])
61                idom[z]=uf.getbest(z);
```

```
62            else
63                idom[z]=semi[z];
64        }
65        dom[x].clear();
66    }
67    semi[rdfn[1]]=1;
68    for(int i=2;i<=n;i++){
69        int x=rdfn[i];
70        if(idom[x]!=semi[x])
71            idom[x]=idom[idom[x]];
72
73    }
74    idom[rdfn[1]]=0;
75 }
76 init();
77 dfs(1);
78 tarjan();
```

## 0.5  字符串
### 0.5.1  模式匹配
#### KMP 算法

```
1 void build(char *pattern) {
2     int length = (int)strlen(pattern + 1);
3     fail[0] = -1;
4     for (int i = 1, j; i <= length; ++i) {
5         for (j = fail[i - 1]; j != -1 && pattern[i] != pattern[j + 1]; j =
↪ fail[j]);
6         fail[i] = j + 1;
7     }
8 }
9
10 void solve(char *text, char *pattern) {
11     int length = (int)strlen(text + 1);
12     for (int i = 1, j; i <= length; ++i) {
13         for (j = match[i - 1]; j != -1 && text[i] != pattern[j + 1]; j = fail[j]);
14         match[i] = j + 1;
15     }
16 }
17 ///Hint: 1 - Base
```

#### 扩展 KMP 算法
返回结果：

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

```
1 void solve(char *text, int length, int *next) {
2     int j = 0, k = 1;
3     for (; j + 1 < length && text[j] == text[j + 1]; j++);
4     next[0] = length - 1;
5     next[1] = j;
6     for (int i = 2; i < length; ++i) {
7         int far = k + next[k] - 1;
8         if (next[i - k] < far - i + 1) {
9             next[i] = next[i - k];
10        } else {
11            j = std::max(far - i + 1, 0);
12            for (; i + j < length && text[j] == text[i + j]; j++);
13            next[i] = j;
14            k = i;
15        }
16    }
17 }
18 /// 0 - Base
```

### AC 自动机

```
1  struct Node{
2      int Next[30], fail, mark;
3  }Tree[N];
4
5  void Init(){
6      memset(Tree, 0, sizeof Tree);
7      cnt = 1;
8
9      for (int i = 1; i <= n; i++){
10         char c;
11         int now = 1;
12         scanf("%s", s + 1);
13         int Length = strlen(s + 1);
14         for (int j = 1; j <= Length; j++){
15             c = s[j];
16             if (Tree[now].Next[c - 'a']) now = Tree[now].Next[c - 'a']; else
17                 Tree[now].Next[c - 'a'] = ++ cnt, now = cnt;
18         }
19     }
20 }
21
22 void Build_Ac(){
23     int en = 0;
24     Q[0] = 1;
25     for (int fi = 0; fi <= en; fi++){
26         int now = Q[fi];
27         for (int next = 0; next < 26; next++)
28             if (Tree[now].Next[next])
29             {
30                 int k = Tree[now].Next[next];
31                 if (now == 1) Tree[k].fail = 1; else
32                 {
33                     int h = Tree[now].fail;
34                     while (h && !Tree[h].Next[next]) h = Tree[h].fail;
35                     if (!h) Tree[k].fail = 1;
36                     else Tree[k].fail = Tree[h].Next[next];
37                 }
38                 Q[++ en] = k;
39             }
40     }
41 }
42
43 /// Hints : when not match , fail = 1
```

### 0.5.2 后缀三姐妹
#### 后缀数组

```
1  struct Sa{
2      int heap[N],s[N],sa[N],r[N],tr[N],sec[N],m,cnt;
3      int h[19][N];
4
5      void Prep(){
6          for (int i=1; i<=m; i++) heap[i]=0;
7          for (int i=1; i<=n; i++) heap[s[i]]++;
8          for (int i=2; i<=m; i++) heap[i]+=heap[i-1];
9          for (int i=n; i>=1; i--) sa[heap[s[i]]--]=i;
10         r[sa[1]]=1; cnt=1;
11         for (int i=2; i<=n; i++){
12             if (s[sa[i]]!=s[sa[i-1]]) cnt++;
13             r[sa[i]]=cnt;
14         }
15         m=cnt;
16     }
17
18     void Suffix(){
19         int j=1;
20         while (cnt<n){
```

```
21             cnt=0;
22             for (int i=n-j+1; i<=n; i++) sec[++cnt]=i;
23             for (int i=1; i<=n; i++) if (sa[i]>j)
24                 sec[++cnt]=sa[i]-j;
25             for (int i=1; i<=n; i++) tr[i]=r[sec[i]];
26             for (int i=1; i<=m; i++) heap[i]=0;
27             for (int i=1; i<=n; i++) heap[tr[i]]++;
28             for (int i=2; i<=m; i++) heap[i]+=heap[i-1];
29             for (int i=n; i>=1; i--)
30                 sa[heap[tr[i]]--]=sec[i];
31             tr[sa[1]]=1; cnt=1;
32             for (int i=2; i<=n; i++){
33                 if ((r[sa[i]]!=r[sa[i-1]]) || (r[sa[i]+j]!=r[sa[i-1]+j]))
34                     cnt++;
35                 tr[sa[i]]=cnt;
36             }
37             for (int i=1; i<=n; i++) r[i]=tr[i];
38             m=cnt; j=j+j;
39         }
40     }
41
42     void Calc(){
43         int k=0;
44         for (int i=1; i<=n; i++){
45             if (r[i]==1) continue;
46             int j=sa[r[i]-1];
47             while ((i+k<=n) && (j+k<=n) && (s[i+k]==s[j+k])) k++;
48             h[0][r[i]]=k;
49             if (k) k--;
50         }
51         for (int i=1; i<19; i++)
52             for (int j=1; j+(1 << i)-1<=n; j++)
53                 h[i][j]=min(h[i-1][j],h[i-1][j + (1 << (i - 1)) + 1]);
54     }
55
56     int Query(int L,int R){
57         L=r[L], R=r[R];
58         if (L>R) swap(L,R);
59         L++;
60         int l0 = Lg[R-L+1];
61         return min(h[l0][L],h[l0][R-(1 << l0)+1]);
62     }
63
64     void Work(){
65         Prep(); Suffix(); Calc();
66     }
67 }P,S;
68
69 /// Hints : 1 - Base
```

#### 后缀数组 (dc3)

```
1  //`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.`
2  //`约定除 r[n-1] 外所有的 r[i] 都大于 0, r[n-1]=0。`
3  //`函数结束后，结果放在 sa 数组中，从 sa[0] 到 sa[n-1]。`
4  //`r 必须开长度乘 3`
5  #define maxn 10000
6  #define F(x) ((x)/3+((x)%3==1?0:tb))
7  #define G(x) ((x)<tb?(x)*3+1:((x)-tb)*3+2)
8
9  int wa[maxn],wb[maxn],wv[maxn],wss[maxn];
10 int s[maxn*3],sa[maxn*3];
11 int c0(int *r,int a,int b)
12 {
13     return r[a]==r[b]&&r[a+1]==r[b+1]&&r[a+2]==r[b+2];
14 }
15 int c12(int k,int *r,int a,int b)
16 {
```

```
17      if(k==2) return r[a]<r[b]||r[a]==r[b]&&c12(1,r,a+1,b+1);
18      else return r[a]<r[b]||r[a]==r[b]&&wv[a+1]<wv[b+1];
19 }
20 void sort(int *r,int *a,int *b,int n,int m)
21 {
22      int i;
23      for(i=0;i<n;i++) wv[i]=r[a[i]];
24      for(i=0;i<m;i++) wss[i]=0;
25      for(i=0;i<n;i++) wss[wv[i]]++;
26      for(i=1;i<m;i++) wss[i]+=wss[i-1];
27      for(i=n-1;i>=0;i--) b[--wss[wv[i]]]=a[i];
28 }
29 void dc3(int *r,int *sa,int n,int m)
30 {
31      int i,j,*rn=r+n,*san=sa+n,ta=0,tb=(n+1)/3,tbc=0,p;
32      r[n]=r[n+1]=0;
33      for(i=0;i<n;i++)
34          if(i%3!=0) wa[tbc++]=i;
35      sort(r+2,wa,wb,tbc,m);
36      sort(r+1,wb,wa,tbc,m);
37      sort(r,wa,wb,tbc,m);
38      for(p=1,rn[F(wb[0])]=0,i=1;i<tbc;i++)
39          rn[F(wb[i])]=c0(r,wb[i-1],wb[i])?p-1:p++;
40      if (p<tbc) dc3(rn,san,tbc,p);
41      else for (i=0;i<tbc;i++) san[rn[i]]=i;
42      for (i=0;i<tbc;i++)
43          if(san[i]<tb) wb[ta++]=san[i]*3;
44      if(n%3==1) wb[ta++]=n-1;
45      sort(r,wb,wa,ta,m);
46      for(i=0;i<tbc;i++)
47          wv[wb[i]=G(san[i])]=i;
48      for(i=0,j=0,p=0;i<ta && j<tbc;p++)
49          sa[p]=c12(wb[j]%3,r,wa[i],wb[j])?wa[i++]:wb[j++];
50      for(;i<ta;p++) sa[p]=wa[i++];
51      for(;j<tbc;p++) sa[p]=wb[j++];
52 }
53
54 int main(){
55      int n,m=0;
56      scanf("%d",&n);
57      for (int i=0;i<n;i++) scanf("%d",&s[i]),s[i]++,m=max(s[i]+1,m);
58      printf("%d\n",m);
59      s[n++]=0;
60      dc3(s,sa,n,m);
61      for (int i=0;i<n;i++) printf("%d ",sa[i]);printf("\n");
62 }
```

### 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

```
1 struct Node{
2      int len, fail;
3      int To[30];
4 }T[N];
5 int Lst, Root, tot, ans;
6 char s[N];
7 int Len[N], Ans[N], Ord[N];
8 void Add(int x, int l){
9      int Nt = ++tot, p = Lst;
10     T[Nt].len = l;
11     for (;p && !T[p].To[x]; p = T[p].fail) T[p].To[x] = Nt;
12     if (!p) T[Nt].fail = Root; else
```

```
13     if (T[T[p].To[x]].len == T[p].len + 1) T[Nt].fail = T[p].To[x];
14     else{
15         int q = ++tot, qt = T[p].To[x];
16         T[q] = T[qt];
17         T[q].len = T[p].len + 1;
18         T[qt].fail = T[Nt].fail = q;
19         for (;p && T[p].To[x] == qt; p = T[p].fail) T[p].To[x] = q;
20     }
21     Lst = Nt;
22 }
23 bool cmp(int a, int b){
24     return T[a].len < T[b].len;
25 }
26 int main(){
27     scanf("%s", s + 1);
28     int n = strlen(s + 1);
29     ans = n;
30     Root = tot = Lst = 1;
31     for (int i = 1; i <= n; i++)
32         Add(s[i] - 'a' + 1, i);
33     for (int i = 1; i <= tot; i++)
34         Ord[i] = i;
35     sort(Ord + 1, Ord + tot + 1, cmp);
36     for (int i = 1; i <= tot; i++)
37         Ans[i] = T[i].len;
38     bool flag = 0;
39     while (scanf("%s", s + 1) != EOF){
40         flag = 1;
41         int n = strlen(s + 1);
42         int p = Root, len = 0;
43         for (int i = 1; i <= tot; i++) Len[i] = 0;
44         for (int i = 1; i <= n; i++){
45             int x = s[i] - 'a' + 1;
46             if (T[p].To[x]) len++, p = T[p].To[x];
47             else {
48                 while (p && !T[p].To[x]) p = T[p].fail;
49                 if (!p) p = Root, len = 0;
50                 else len = T[p].len + 1, p = T[p].To[x];
51             }
52             Len[p] = max(Len[p], len);
53         }
54         for (int i = tot; i >= 1; i--){
55             int Cur = Ord[i];
56             Ans[Cur] = min(Ans[Cur], Len[Cur]);
57             if (Len[Cur] && T[Cur].fail)
58                 Len[T[Cur].fail] = T[T[Cur].fail].len;
59         }
60     }
61     if (flag){
62         ans = 0;
63         for (int i = 1; i <= tot; i++){
64             ans = max(ans, Ans[i]);
65         }
66     }
67     printf("%d\n", ans);
68     return 0;
69 }
```

#### 0.5.3　回文三兄弟

马拉车

```
1 void Manacher(){
2      R[1] = 1;
3      for (int i = 2, j = 1; i <= length; i++){
4          if (j + R[j] <= i){
5              R[i] = 0;
```

```
 6        } else {
 7            R[i] = min(R[j * 2 - i], j + R[j] - i);
 8        }
 9        while (i - R[i] >= 1 && i + R[i] <= length
10            && text[i - R[i]] == text[i + R[i]]){
11            R[i]++;
12        }
13        if (i + R[i] > j + R[j]){
14            j = i;
15        }
16    }
17 }
18    length = 0;
19    int n = strlen(s + 1);
20    for (int i = 1; i <= n; i++){
21        text[++length] = '*';
22        text[++length] = s[i];
23    }
24    text[++length] = '*';
25 /// Hints: 1 - Base
```

### 回文自动机 (zky)

```
 1 struct PAM{
 2    int tot,last,str[maxn],nxt[maxn][26],n;
 3    int len[maxn],suf[maxn],cnt[maxn];
 4    int newnode(int l){
 5        len[tot]=l;
 6        return tot++;
 7    }
 8    void init(){
 9        tot=0;
10        newnode(0);// tree0 is node 0
11        newnode(-1);// tree-1 is node 1
12        str[0]=-1;
13        suf[0]=1;
14    }
15    int find(int x){
16        while(str[n-len[x]-1]!=str[n])x=suf[x];
17        return x;
18    }
19    void add(int c){
20        str[++n]=c;
21        int u=find(last);
22        if(!nxt[u][c]){
23            int v=newnode(len[u]+2);
24            suf[v]=nxt[find(suf[u])][c];
25            nxt[u][c]=v;
26        }last=nxt[u][c];
27        cnt[last]++;
28    }
29    void count(){
30        for(int i=tot-1;i>=0;i--)cnt[suf[i]]+=cnt[i];
31    }
32 }P;
33 int main(){
34    P.init();
35    for(int i=0;i<n;i++)
36        P.add(s[i]-'a');
37    P.count();
```

### 0.5.4    循环串最小表示

```
 1 string sol(char *s){
 2    int n=strlen(s);
 3    int i=0,j=1,k=0,p;
 4    while(i<n&&j<n&&k<n){
 5        int t=s[(i+k)%n]-s[(j+k)%n];
```

```
 6            if(t==0)k++;
 7            else if(t<0)j+=k+1,k=0;
 8            else i+=k+1,k=0;
 9            if(i==j)j++;
10    }p=min(i,j);
11    string S;
12    for(int i=p;i<p+n;i++)S.push_back(s[i%n]);
13    return S;
14 }
```

## 0.6    计算几何
### 0.6.1    二维基础
点类

```
 1 struct P{
 2    double x,y;
 3    P turn90(){return P(-y,x);}
 4 };
 5 double det(P a,P b,P c){
 6    return (b-a)*(c-a);
 7 }
 8 P intersect(L l1,L l2){
 9    double s1=det(l1.a,l1.b,l2.a);
10    double s2=det(l1.a,l1.b,l2.b);
11    return (l2.a*s2-l2.b*s1)/(s2-s1);
12 }
13 P project(P p,L l){
14    return l.a+l.v()*((p-l.a)^l.v())/l.v().len2();
15 }
16 double dis(P p,L l){
17    return fabs((p-l.a)*l.v())/l.v().len();
18 }
```

凸包

```
 1 vector<P> convex(vector<P>p){
 2    sort(p.begin(),p.end());
 3    vector<P>ans,S;
 4    for(int i=0;i<p.size();i++){
 5        while(S.size()>=2
 6            && sgn(det(S[S.size()-2],S.back(),p[i]))<=0)
 7                S.pop_back();
 8        S.push_back(p[i]);
 9    }//dw
10    ans=S;
11    S.clear();
12    for(int i=(int)p.size()-1;i>=0;i--){
13        while(S.size()>=2
14            && sgn(det(S[S.size()-2],S.back(),p[i]))<=0)
15                S.pop_back();
16        S.push_back(p[i]);
17    }//up
18    for(int i=1;i+1<S.size();i++)
19        ans.push_back(S[i]);
20    return ans;
21 }
```

半平面交

```
 1 struct P{
 2    int quad() const { return sgn(y) == 1 || (sgn(y) == 0 && sgn(x) >= 0);}
 3 };
 4 struct L{
 5    bool onLeft(const P &p) const { return sgn((b - a)*( p - a)) > 0; }
 6    L push() const{ // push out eps
```

```cpp
        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
struct line{
    point p,v;
};
point Rev(point v){return point(-v.y,v.x);}
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);
```

```cpp
    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;
}
int main(){
    scanf("%d",&n);
    for(int i=1;i<=n;i++)scanf("%lf%lf",&p[i].x,&p[i].y);
    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]);
            }
        }
    }printf("%.2lf %.2lf %.2lf\n",O.x,O.y,r);
    return 0;
}s
```

### 0.6.2  多边形
判断点在多边形内部

```cpp
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;
}
```

## 0.7  其他
### 0.7.1  斯坦纳树

```cpp
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++){
        if (id[i]) continue;
        for (int s0 = 0; s0 < s; s0++)
            if ( (s0 & s) == s0 ){
                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++){
        if (id[i]) continue;
        Q.push(mp(-f[s][i], i));
    }
    while (!Q.empty()){
```

```
21          while (!Q.empty() && Q.top().first != -f[s][Q.top().second]) Q.pop();
22              if (Q.empty()) break;
23              int Cur = Q.top().second; Q.pop();
24              for (int p = g[Cur]; p; p = nxt[p]){
25                  int y = adj[p];
26                  if ( f[s][y] > f[s][Cur] + 1){
27                      f[s][y] = f[s][Cur] + 1;
28                      Q.push(mp(-f[s][y], y));
29                  }
30              }
31          }
32 }
```

### 0.7.2 无敌的读入优化

```
1  namespace Reader {
2      const int L = (1 << 20) + 5;
3      char buffer[L], *S, *T;
4      __inline bool getchar(char &ch) {
5          if (S == T) {
6              T = (S = buffer) + fread(buffer, 1, L, stdin);
7              if (S == T) {
8                  ch = EOF;
9                  return false;
10             }
11         }
12         ch = *S ++;
13         return true;
14     }
15     __inline bool getint(int &x) {
16         char ch;
17         for (; getchar(ch) && (ch < '0' || ch > '9'); );
18         if (ch == EOF) return false;
19         x = ch - '0';
20         for (; getchar(ch), ch >= '0' && ch <= '9'; )
21             x = x * 10 + ch - '0';
22         return true;
23     }
24 }
25 Reader::getint(x);
26 Reader::getint(y);
```

### 0.7.3 最小树形图

```
1  const int maxn=1100;
2
3  int n,m , g[maxn][maxn] , used[maxn] , pass[maxn] , eg[maxn] , more ,
     ↪ queue[maxn];
4
5  void combine (int id , int &sum ) {
6      int tot = 0 , from , i , j , k ;
7      for ( ; id!=0 && !pass[ id ] ; id=eg[id] ) {
8          queue[tot++]=id ; pass[id]=1;
9      }
10     for ( from=0; from<tot && queue[from]!=id ; from++);
11     if ( from==tot ) return ;
12     more = 1 ;
13     for ( i=from ; i<tot ; i++) {
14         sum+=g[eg[queue[i]]][queue[i]] ;
15         if ( i!=from ) {
16             used[queue[i]]=1;
17             for ( j = 1 ; j <= n ; j++) if ( !used[j] )
18                 if ( g[queue[i]][j]<g[id][j] ) g[id][j]=g[queue[i]][j] ;
19         }
20     }
21     for ( i=1; i<=n ; i++) if ( !used[i] && i!=id ) {
22         for ( j=from ; j<tot ; j++){
23             k=queue[j];
```

```
24             if ( g[i][id]>g[i][k]-g[eg[k]][k] ) g[i][id]=g[i][k]-g[eg[k]][k];
25         }
26     }
27 }
28
29 int mdst( int root ) { // return the total length of MDST
30     int i , j , k , sum = 0 ;
31     memset ( used , 0 , sizeof ( used ) ) ;
32     for ( more =1; more ; ) {
33         more = 0 ;
34         memset (eg,0,sizeof(eg)) ;
35         for ( i=1 ; i <= n ; i ++) if ( !used[i] && i!=root ) {
36             for ( j=1 , k=0 ; j <= n ; j ++) if ( !used[j] && i!=j )
37                 if ( k==0 || g[j][i] < g[k][i] ) k=j ;
38             eg[i] = k ;
39         }
40         memset(pass,0,sizeof(pass));
41         for ( i=1; i<=n ; i++) if ( !used[i] && !pass[i] && i!= root ) combine (
             ↪ i , sum ) ;
42     }
43     for ( i =1; i<=n ; i ++) if ( !used[i] && i!= root ) sum+=g[eg[i]][i];
44     return sum ;
45 }
```

### 0.7.4 DLX

```
1  int n,m,K;
2  struct DLX{
3      int L[maxn],R[maxn],U[maxn],D[maxn];
4      int sz,col[maxn],row[maxn],s[maxn],H[maxn];
5      bool vis[233];
6      int ans[maxn],cnt;
7      void init(int m){
8          for(int i=0;i<=m;i++){
9              L[i]=i-1;R[i]=i+1;
10             U[i]=D[i]=i;s[i]=0;
11         }
12         memset(H,-1,sizeof H);
13         L[0]=m;R[m]=0;sz=m+1;
14     }
15     void Link(int r,int c){
16         U[sz]=c;D[sz]=D[c];U[D[c]]=sz;D[c]=sz;
17         if(H[r]<0)H[r]=L[sz]=R[sz]=sz;
18         else{
19             L[sz]=H[r];R[sz]=R[H[r]];
20             L[R[H[r]]]=sz;R[H[r]]=sz;
21         }
22         s[c]++;col[sz]=c;row[sz]=r;sz++;
23     }
24     void remove(int c){
25         for(int i=D[c];i!=c;i=D[i])
26             L[R[i]]=L[i],R[L[i]]=R[i];
27     }
28     void resume(int c){
29         for(int i=U[c];i!=c;i=U[i])
30             L[R[i]]=R[L[i]]=i;
31     }
32     int A(){
33         int res=0;
34         memset(vis,0,sizeof vis);
35         for(int i=R[0];i;i=R[i])if(!vis[i]){
36             vis[i]=1;res++;
37             for(int j=D[i];j!=i;j=D[j])
38                 for(int k=R[j];k!=j;k=R[k])
39                     vis[col[k]]=1;
40         }
41         return res;
```

```
42      }
43      void dfs(int d,int &ans){
44          if(R[0]==0){ans=min(ans,d);return;}
45          if(d+A()>=ans)return;
46          int tmp=23333,c;
47          for(int i=R[0];i;i=R[i])
48              if(tmp>s[i])tmp=s[i],c=i;
49          for(int i=D[c];i!=c;i=D[i]){
50              remove(i);
51              for(int j=R[i];j!=i;j=R[j])remove(j);
52              dfs(d+1,ans);
53              for(int j=L[i];j!=i;j=L[j])resume(j);
54              resume(i);
55          }
56      }
57      void del(int c){//exactly cover
58          L[R[c]]=L[c];R[L[c]]=R[c];
59          for(int i=D[c];i!=c;i=D[i])
60              for(int j=R[i];j!=i;j=R[j])
61                  U[D[j]]=U[j],D[U[j]]=D[j],--s[col[j]];
62      }
63      void add(int c){   //exactly cover
64          R[L[c]]=L[R[c]]=c;
65          for(int i=U[c];i!=c;i=U[i])
66              for(int j=L[i];j!=i;j=L[j])
67                  ++s[col[U[D[j]]=D[U[j]]=j]];
68      }
69      bool dfs2(int k){//exactly cover
70          if(!R[0]){
71              cnt=k;return 1;
72          }
73          int c=R[0];
74          for(int i=R[0];i;i=R[i])
75              if(s[c]>s[i])c=i;
76          del(c);
77          for(int i=D[c];i!=c;i=D[i]){
78              for(int j=R[i];j!=i;j=R[j])
79                  del(col[j]);
80              ans[k]=row[i];if(dfs2(k+1))return true;
81              for(int j=L[i];j!=i;j=L[j])
82                  add(col[j]);
83          }
84          add(c);
85          return 0;
86      }
87  }dlx;
88  int main(){
89      dlx.init(n);
90      for(int i=1;i<=m;i++)
91          for(int j=1;j<=n;j++)
92              if(dis(station[i],city[j])<mid-eps)
93                  dlx.Link(i,j);
94      dlx.dfs(0,ans);
95  }
```

### 0.7.5 某年某月某日是星期几

```
1  int solve(int year, int month, int day) {
2      int answer;
3      if (month == 1 || month == 2) {
4          month += 12;
5          year--;
6      }
7      if ((year < 1752) || (year == 1752 && month < 9) ||
8          (year == 1752 && month == 9 && day < 3)) {
9          answer = (day + 2 * month + 3 * (month + 1) / 5 + year + year / 4 + 5) % 7;
10     } else {
11         answer = (day + 2 * month + 3 * (month + 1) / 5 + year + year / 4
12                 - year / 100 + year / 400) % 7;
13     }
14     return answer;
15 }
```

### 0.7.6 枚举大小为 $k$ 的子集

使用条件：$k > 0$

```
1  void solve(int n, int k) {
2      for (int comb = (1 << k) - 1; comb < (1 << n); ) {
3          // ...
4          int x = comb & -comb, y = comb + x;
5          comb = ((((comb & ~y) / x) >> 1) | y;
6      }
7  }
```

### 0.7.7 环状最长公共子串

```
1  int n, a[N << 1], b[N << 1];
2
3  bool has(int i, int j) {
4      return a[(i - 1) % n] == b[(j - 1) % n];
5  }
6
7  const int DELTA[3][2] = {{0, -1}, {-1, -1}, {-1, 0}};
8
9  int from[N][N];
10
11 int solve() {
12     memset(from, 0, sizeof(from));
13     int ret = 0;
14     for (int i = 1; i <= 2 * n; ++i) {
15         from[i][0] = 2;
16         int left = 0, up = 0;
17         for (int j = 1; j <= n; ++j) {
18             int upleft = up + 1 + !!from[i - 1][j];
19             if (!has(i, j)) {
20                 upleft = INT_MIN;
21             }
22             int max = std::max(left, std::max(upleft, up));
23             if (left == max) {
24                 from[i][j] = 0;
25             } else if (upleft == max) {
26                 from[i][j] = 1;
27             } else {
28                 from[i][j] = 2;
29             }
30             left = max;
31         }
32         if (i >= n) {
33             int count = 0;
34             for (int x = i, y = n; y; ) {
35                 int t = from[x][y];
36                 count += t == 1;
37                 x += DELTA[t][0];
38                 y += DELTA[t][1];
39             }
40             ret = std::max(ret, count);
41             int x = i - n + 1;
42             from[x][0] = 0;
43             int y = 0;
44             while (y <= n && from[x][y] == 0) {
45                 y++;
46             }
47             for (; x <= i; ++x) {
```

```
48                    from[x][y] = 0;
49                    if (x == i) {
50                        break;
51                    }
52                    for (; y <= n; ++y) {
53                        if (from[x + 1][y] == 2) {
54                            break;
55                        }
56                        if (y + 1 <= n && from[x + 1][y + 1] == 1) {
57                            y++;
58                            break;
59                        }
60                    }
61                }
62            }
63        }
64        return ret;
65 }
```

### 0.7.8 LLMOD

```
1 LL multiplyMod(LL a, LL b, LL P) { // `需要保证 a 和 b 非负`
2     LL t = (a * b - LL((long double)a / P * b + 1e-3) * P) % P;
3     return t < 0 : t + P : t;
4 }
```

## 0.8 Java
### 0.8.1 基础模板

```
1 public class Main {
2     public static void main(String[] args) {
3         InputReader in = new InputReader(System.in);
4         PrintWriter out = new PrintWriter(System.out);
5     }
6 }
7 public static class cmp implements Comparator<edge>{
8     public int compare(edge a,edge b){
9         if(a.w<b.w)return 1;
10        if(a.w>b.w)return -1;
11        return 0;
12    }
13 }
14 class InputReader {
15     public BufferedReader reader;
16     public StringTokenizer tokenizer;
17     public InputReader(InputStream stream) {
18         reader = new BufferedReader(new InputStreamReader(stream), 32768);
19         tokenizer = null;
20     }
21     public String next() {
22         while (tokenizer == null || !tokenizer.hasMoreTokens()) {
23             try {
24                 tokenizer = new StringTokenizer(reader.readLine());
25             } catch (IOException e) {
26                 throw new RuntimeException(e);
27             }
28         }
29         return tokenizer.nextToken();
30     }
31     public int nextInt() {
32         return Integer.parseInt(next());
33     }
34 }
```