



ACM/ICPC Template Manual

QUST

hxx

October 11, 2018

Contents

| | | |
|----------|---|-----------|
| 0 | Include | 1 |
| 1 | Math | 2 |
| 1.1 | Fast Power | 2 |
| 1.2 | Basic Number Theory | 2 |
| 1.2.1 | Extended Euclidean | 2 |
| 1.2.2 | Multiplicative Inverse Modulo | 2 |
| 1.3 | Eular phi | 3 |
| 1.3.1 | Eular | 3 |
| 1.4 | Prime | 4 |
| 1.4.1 | Miller Rabin | 4 |
| 1.4.2 | Eratosthenes Sieve | 4 |
| 1.4.3 | Segment Sieve | 4 |
| 1.4.4 | primesON | 5 |
| 1.4.5 | divide | 5 |
| 1.4.6 | fact | 6 |
| 1.5 | Matrix | 6 |
| 1.5.1 | pointchanging | 7 |
| 1.6 | Combinatorics | 9 |
| 1.6.1 | Combination | 9 |
| 1.7 | SumRamainder | 9 |
| 2 | String Processing | 11 |
| 2.1 | KMP | 11 |
| 2.2 | Trie | 11 |
| 2.3 | Manacher | 12 |
| 2.4 | SaHash | 13 |
| 2.5 | SA | 14 |
| 2.6 | HashString | 15 |
| 3 | Data Structure | 17 |
| 3.1 | other | 17 |
| 3.1.1 | QuickSelect | 17 |
| 3.1.2 | mergingsort | 17 |
| 3.1.3 | pbds | 18 |
| 3.1.4 | stack | 18 |
| 3.1.5 | queue | 19 |
| 3.2 | Binary Indexed Tree | 19 |
| 3.2.1 | poj3468 | 20 |
| 3.3 | Segment Tree | 21 |
| 3.3.1 | Single-point Update | 21 |
| 3.3.2 | Interval Update | 21 |
| 3.4 | Splay Tree | 22 |
| 3.5 | Functional Segment Tree | 24 |
| 3.6 | Sparse Table | 24 |
| 3.7 | block | 26 |
| 3.8 | Treap | 27 |
| 4 | Graph Theory | 32 |
| 4.1 | Union-Find Set | 32 |
| 4.2 | Minimal Spanning Tree | 32 |
| 4.2.1 | Kruskal | 32 |
| 4.3 | Shortest Path | 33 |
| 4.3.1 | Dijkstra | 33 |
| 4.3.2 | Spfa | 34 |
| 4.3.3 | kth-p | 35 |
| 4.4 | Topo Sort | 37 |
| 4.5 | LCA | 39 |

| | | |
|----------|------------------------------|-----------|
| 4.5.1 | LCA | 39 |
| 4.6 | Depth-First Traversal | 40 |
| 4.6.1 | Biconnected-Component | 40 |
| 4.6.2 | Strongly Connected Component | 42 |
| 4.6.3 | Kosaraju | 43 |
| 4.6.4 | TwoSAT | 44 |
| 4.6.5 | cut-vertex | 46 |
| 4.7 | Bipartite Graph Matching | 47 |
| 4.7.1 | Hungry | 47 |
| 4.8 | Network Flow | 48 |
| 4.8.1 | Dinic | 48 |
| 4.8.2 | MinCost MaxFlow | 50 |
| 5 | Others | 52 |
| 5.1 | Matrix | 52 |
| 5.1.1 | Matrix FastPow | 52 |
| 5.2 | Tricks | 52 |
| 5.2.1 | Stack-Overflow | 52 |
| 5.2.2 | Fast-Scanner | 52 |
| 5.2.3 | Strok-Sscanf | 53 |
| 5.3 | Mo Algorithm | 53 |
| 5.4 | BigNum | 54 |
| 5.4.1 | High-precision | 54 |
| 5.5 | VIM | 56 |
| 5.6 | BASH | 56 |
| 6 | Geometry | 57 |

0 Include

```
1  // #include <bits/stdc++.h>
2  #include <algorithm>
3  #include <iostream>
4  #include <cstring>
5  #include <string>
6  #include <cstdio>
7  #include <vector>
8  #include <stack>
9  #include <queue>
10 #include <cmath>
11 #include <set>
12 #include <map>
13 using namespace std;
14 #define rep(i,a,b) for(int i=a;i<=b;i++)
15 #define per(i,a,b) for(int i=a;i>=b;i--)
16 #define clr(a,x) memset(a,x,sizeof(a))
17 #define pb push_back
18 #define mp make_pair
19 #define all(x) (x).begin(),(x).end()
20 #define fi first
21 #define se second
22 #define SZ(x) ((int)(x).size())
23 typedef unsigned long long ull;
24 typedef long long ll;
25 typedef vector<int> vi;
26 typedef pair<int,int> pii;
27 /*****head*****/
28 int work(){
29
30     return 0;
31 }
32 int main(){
33 #ifdef superkunn
34     freopen("input.txt","rt",stdin);
35 #endif
36     work();
37     return 0;
38 }
```

1 Math

1.1 Fast Power

```

1 typedef long long ll;
2 void add(ll &a,ll b,ll mod){
3     a+=b;
4     a%=mod;
5 }
6 ll mul_mod(ll a,ll b,ll mod){
7     ll res=0;
8     while(b){
9         if(b&1)add(res,a,mod);
10        add(a,a,mod);
11        b>>=1;
12    }
13    return res;
14 }
15 /*
16 ll mul_mod(ll a,ll b,ll mod){
17     a%=mod;
18     b%=mod;
19     ll c=(long double)a*b/mod;
20     ll ans=a*b-c*mod;
21     if(ans<0)ans+=mod;
22     else if(ans>mod)ans-=mod;
23     return ans;
24 }
25 */
26 ll pow_mod(ll a,ll b,ll mod){//a^b
27     ll res=1%mod;
28     while(b){
29         if(b&1)res=mul_mod(res,a,mod);
30         a=mul_mod(a,a,mod);
31         b>>=1;
32     }
33     return res;
34 }

```

1.2 Basic Number Theory

1.2.1 Extended Euclidean

```

1 typedef long long ll;
2 //__gcd(a,b);
3 ll gcd(ll a,ll b){return b==0?a:gcd(b,a%b);}
4 ll exgcd(ll a,ll b,ll &x,ll &y){
5     ll d=a;
6     if(b)d=exgcd(b,a%b,y,x),y-=x*(a/b);
7     else x=1,y=0;
8     return d;
9 }

```

1.2.2 Multiplicative Inverse Modulo

```

1 ll inv(ll a,ll m){
2     ll x,y;
3     ll d=exgcd(a,m,x,y);

```

```

4     return d==1?(x+m)%m:-1;
5 }
6 ll inv(ll a,ll m){
7     return pow_mod(a,m-2,m);
8 }
9 int p=37;
10 inv[1]=1;
11 for(int i=2;i<=40;i++){
12     inv[i]=(p-(p/i))*inv[p%i]%p;
13 }
14 //fact invfact
15 int fact[MAXN];
16 int invfact[MAXN];
17 ll pow_mod(ll a,ll b){
18     ll res=1;
19     while(b){
20         if(b&1)res=res*a%MOD;
21         a=a*a%MOD;
22         b>>=1;
23     }
24     return res;
25 }
26 ll fun(ll n,ll m){
27     return (1LL*fact[n]*invfact[m])%MOD*invfact[n-m]%MOD;
28 }
29 int n=100000;
30 fact[0]=1;
31 for(int i=1;i<=n;i++){
32     fact[i]=1LL*fact[i-1]*i%MOD;
33 }
34 invfact[n]=pow_mod(fact[n],MOD-2);
35 for(int i=n;i>=1;i--){
36     invfact[i-1]=1LL*invfact[i]*i%MOD;
37 }

```

1.3 Euler phi

1.3.1 Euler

```

1 #include<bits/stdc++.h>
2 using namespace std;
3 typedef long long ll;
4 const int MAXN=10000;
5 int phi[MAXN];
6 int phi1(int n){
7     int res=n;
8     for(int i=2;i*i<=n;i++){
9         if(n%i==0){
10             res=res/i*(i-1);
11             for(;n%i==0;n/=i);
12         }
13     }
14     if(n!=1) res=res/n*(n-1);
15     return res;
16 }
17 void phi2(int n){
18     for(int i=0;i<=n;i++) phi[i]=i;
19     for(int i=2;i<=n;i++)
20         if(phi[i]==i)

```

```

21         for(int j=i;j<=n;j+=i) phi[j]=phi[j]/i*(i-1);
22     }
23     int main(){
24         phi2(100);
25         for(int i=1;i<=100;i++)cout<<phi1(i)<<" "<<phi[i]<<endl;
26         return 0;
27     }

```

1.4 Prime

1.4.1 Miller Rabin

```

1 //using Fast Power
2 bool Miller_Rabin(ll n, int s){//s is testing frequency . true -> n is prime
3     if (n == 2) return 1;
4     if (n < 2 || !(n & 1)) return 0;
5     int t = 0;
6     ll x, y, u = n - 1;
7     while ((u & 1) == 0) t++, u >>= 1;
8     for (int i = 0; i < s; i++){
9         ll a = rand() % (n - 1) + 1;
10        ll x = pow_mod(a, u, n);
11        for (int j = 0; j < t; j++){
12            ll y = mul_mod(x, x, n);
13            if (y == 1 && x != 1 && x != n - 1) return 0;
14            x = y;
15        }
16        if (x != 1) return 0;
17    }
18    return 1;
19 }

```

1.4.2 Eratosthenes Sieve

```

1 #define rep(i,a,b) for(int i=a;i<=b;i++)
2 const int MAXN=1e5+5;
3 int prime[MAXN];//1 base
4 bool is_prime[MAXN];
5 int sieve(int n){
6     int cnt=0;
7     rep(i,0,n)is_prime[i]=true;
8     is_prime[0]=is_prime[1]=false;
9     rep(i,2,n){
10        if(is_prime[i]){
11            prime[++cnt]=i;
12            for(int j=i;j<=n/i;j++)is_prime[i*j]=false;
13        }
14    }
15    return cnt;
16 }

```

1.4.3 Segment Sieve

```

1 const int MAXN=1e6+5;
2 //[a,b]
3 bool is_prime[MAXN];
4 bool is_prime_small[MAXN];

```

```

5  ll prime[MAXN]; //1 base
6  int segment_sieve(ll a, ll b){
7      int cnt=0;
8      for(int i=0; 1LL*i*i<b; i++) is_prime_small[i]=true;
9      is_prime_small[0]=is_prime_small[1]=false;
10     for(int i=0; i<b-a; i++) is_prime[i]=true;
11     if(a==1) is_prime[0]=false;
12     for(int i=2; 1LL*i*i<b; i++){
13         if(is_prime_small[i]){
14             for(int j=2*i; 1LL*j*j<b; j+=i) is_prime_small[j]=false; // [2, sqrt(b))
15             for(ll j=max(2LL, (a+i-1)/i)*i; j<b; j+=i) is_prime[j-a]=false;
16         }
17     }
18     // [a, b) [0, b-a)
19     for(ll i=0; i<b-a; i++){
20         if(is_prime[i]) prime[++cnt]=i+a;
21     }
22     return cnt;
23 }

```

1.4.4 primesON

```

1  const int MAXN=2e5+10;
2  int v[MAXN], prime[MAXN];
3  int cnt;
4  void primes(int n){
5      memset(v, 0, sizeof(v));
6      cnt=0;
7      for(int i=2; i<=n; i++){
8          if(v[i]==0){
9              v[i]=i;
10             prime[++cnt]=i;
11         }
12         for(int j=1; j<=cnt; j++){
13             if(prime[j]>v[i] || prime[j]>n/i) break;
14             v[i*prime[j]]=prime[j];
15         }
16     }
17 }

```

1.4.5 divide

```

1  // Vijos 1786
2  const int MAXN=1e5+10;
3  int cnt;
4  int num[MAXN];
5  int p[MAXN];
6  void divide(int n){
7      cnt=0;
8      for(int i=2; 1LL*i*i<=n; i++){
9          if(n%i==0){
10             p[++cnt]=i, num[cnt]=0;
11         }
12         while(n%i==0) n/=i, num[cnt]++;
13     }
14     if(n>1){
15         p[++cnt]=n, num[cnt]=1;
16     }

```



```

17 }
18 int main(){
19     int n;
20     scanf("%d",&n);
21     divide(n);
22     printf("%d",p[2]);
23     return 0;
24 }

```

1.4.6 fact

```

1 int main(){
2     int n;
3     scanf("%d",&n);
4     primes(n);
5     for(int i=1;i<=cnt;i++){
6         int p=prime[i],c=0;
7         for(int j=n;j;j/=p)c+=j/p;
8         printf("%d %d\n",p,c);
9     }
10    return 0;
11 }

```

1.5 Matrix

```

1 //hdu 1005
2 #include <cstdio>
3 #include <algorithm>
4 #include <iostream>
5 using namespace std;
6 const int MOD = 7;
7 struct Matrix {
8     long long a[2][2];
9 };
10 Matrix operator*(const Matrix& lhs, const Matrix& rhs) {
11     Matrix ret;
12     for (int i = 0; i < 2; ++i) {
13         for (int j = 0; j < 2; ++j) {
14             ret.a[i][j] = 0;
15             for (int k = 0; k < 2; ++k) {
16                 ret.a[i][j] += lhs.a[i][k] * rhs.a[k][j];
17             }
18             ret.a[i][j] %= MOD;
19         }
20     }
21     return ret;
22 }
23 int main(){
24     int a,b,n;
25     while(~scanf("%d%d%d",&a,&b,&n)){
26         if(a==0&&b==0&&n==0)break;
27         Matrix x,y;
28         x.a[0][0]=0;
29         x.a[0][1]=1;
30         x.a[1][0]=b;
31         x.a[1][1]=a;
32         y.a[0][1]=y.a[1][1]=0;

```

```

33     y.a[0][0]=y.a[1][0]=1;
34     if(n<=2){
35         puts("1");
36         continue;
37     }
38     n-=2;
39     while(n>0){
40         if(n&1)y=x*y;
41         x=x*x;
42         n>>=1;
43     }
44     printf("%lld\n",y.a[1][0]%MOD);
45 }
46
47 return 0;
48 }

```

1.5.1 pointchanging

```

1  #include<bits/stdc++.h>
2  using namespace std;
3  const double PI=acos(-1.0);
4  struct Matrix{
5      double a[3][3];
6      void init(){
7          for(int i=0;i<3;i++){
8              for(int j=0;j<3;j++){
9                  a[i][j]=0;
10             }
11         }
12     }
13     void print(){
14         for(int i=0;i<3;i++){
15             for(int j=0;j<3;j++){
16                 cout<<a[i][j]<<" ";
17             }
18             cout<<endl;
19         }
20         cout<<"-----"<<endl;
21     }
22 };
23 Matrix operator*(const Matrix& lhs,const Matrix& rhs){
24     Matrix res;
25     res.init();
26     for(int i=0;i<3;i++){
27         for(int j=0;j<3;j++){
28             for(int k=0;k<3;k++){
29                 res.a[i][j]+=lhs.a[i][k]*rhs.a[k][j];
30             }
31         }
32     }
33     return res;
34 }
35 const int MAXN=1e4+10;
36 double x[MAXN],y[MAXN];
37 int main(){
38
39     int n,m;

```

```

40     scanf("%d%d",&n,&m);
41     for(int i=1;i<=n;i++){
42         scanf("%lf%lf",&x[i],&y[i]);
43     }
44     Matrix base;
45     base.init();
46     base.a[0][0]=base.a[1][1]=base.a[2][2]=1;
47     char op[3];
48     Matrix now;
49     while(m--){
50         scanf("%s",op);
51         now.init();
52         if(op[0]=='X'){
53             now.a[0][0]=1;
54             now.a[1][1]=-1;
55             now.a[2][2]=1;
56         }else if(op[0]=='Y'){
57             now.a[0][0]=-1;
58             now.a[1][1]=1;
59             now.a[2][2]=1;
60         }else if(op[0]=='M'){
61             double p,q;
62             scanf("%lf%lf",&p,&q);
63             now.a[0][0]=1;
64             now.a[1][1]=1;
65             now.a[2][2]=1;
66             now.a[0][2]=p;
67             now.a[1][2]=q;
68         }else if(op[0]=='S'){
69             double L;
70             scanf("%lf",&L);
71             now.a[0][0]=L;
72             now.a[1][1]=L;
73             now.a[2][2]=1;
74         }else if(op[0]=='R'){
75             double r;
76             scanf("%lf",&r);
77             r=r/180*PI;
78             now.a[0][0]=cos(r);
79             now.a[0][1]=-sin(r);
80             now.a[1][0]=sin(r);
81             now.a[1][1]=cos(r);
82             now.a[2][2]=1;
83         }
84         base=now*base;
85     }
86
87     for(int i=1;i<=n;i++){
88         Matrix ans;
89         ans.init();
90         ans.a[0][0]=x[i];
91         ans.a[1][0]=y[i];
92         ans.a[2][0]=1;
93         ans=base*ans;
94         printf("%.1f %.1f\n",ans.a[0][0],ans.a[1][0]);
95     }
96     return 0;
97 }

```

1.6 Combinatorics

1.6.1 Combination

```

1 //2^n-C(0,n)...C(k-1,n)=C(k,n)+...+C(n,n)
2 //2017 EC A
3 #include<bits/stdc++.h>
4 using namespace std;
5 typedef long long ll;
6 const int MOD=1000000007;
7 const int MAXN=1e5+10;
8 ll cnk[MAXN],inv[MAXN];
9 ll pow_mod(ll a,ll b){
10     ll res=1;
11     while(b){
12         if(b&1)res=res*a%MOD;
13         a=a*a%MOD;
14         b>>=1;
15     }
16     return res;
17 }
18 int main(){
19     int T;
20     scanf("%d",&T);
21     int kase=0;
22     while(T--){
23         int n,k;
24         scanf("%d%d",&n,&k);
25         ll a=pow_mod(2,n);
26         int p=MOD;
27         inv[1]=1;
28         for(int i=2;i<=k;i++){
29             inv[i]=1LL*(p-p/i)*inv[p/i]%p;
30         }
31         cnk[0]=1;
32         ll ans=cnk[0];
33         for(int i=1;i<k;i++){
34             cnk[i]=cnk[i-1]*(n-i+1)%MOD*inv[i]%MOD;
35             ans+=cnk[i];
36             if(ans>MOD)ans-=MOD;
37         }
38         ans=(a-ans+MOD)%MOD;
39         printf("Case #%d: %I64d\n",++kase,ans);
40     }
41     return 0;
42 }

```

1.7 SumRamaider

```

1 //cf 616 E
2 const int MOD=1e9+7;
3 int main(){
4     ll n,k,ans;
5     scanf("%lld%lld",&k,&n);
6     ans=n%MOD*(k%MOD);
7     ans%=MOD;
8     ll inv2=MOD-MOD/2;
9     for(ll x=1,gx;x<=n;gx=gx+1){
10         gx=k/x?min(k/(k/x),n):n;

```

```
11         ans-=((k/x)%MOD*((x+gx)%MOD)%MOD*((gx-x+1)%MOD)%MOD*inv2)%MOD;
12         if(ans<0)ans+=MOD;
13     }
14     printf("%lld",ans);
15     return 0;
16 }
```

2 String Processing

2.1 KMP

```

1 //MAXN
2 int nxt[MAXN];
3 void initkmp(char x[],int m){
4     int i=0,j=nxt[0]=-1;
5     while(i<m){
6         while(j!=-1&&x[i]!=x[j])j=nxt[j];
7         nxt[++i]=++j;
8     }
9 }
10 //x:pa y:tx
11 int kmp(char x[],int m,char y[],int n){
12     int i,j,ans;
13     i=j=ans=0;
14     initkmp(x,m);
15     while(i<n){
16         while(j!=-1&&y[i]!=x[j])j=nxt[j];
17         i++,j++;
18         if(j>=m){
19             ans++;
20             j=nxt[j];
21             //pos:i-m
22         }
23     }
24     return ans;
25 }

```

2.2 Trie

```

1 //hihocoder 1014
2 const int maxnode=2600000+10;
3 const int sigma_size=26;
4 struct Trie{
5     int ch[maxnode][sigma_size];
6     int val[maxnode];
7     int sz;
8     void init(){sz=0;clr(ch[0],0);}
9     int idx(char c){return c-'a';}
10    void insert(char *s){
11        int u=0,n=strlen(s);
12        rep(i,0,n-1){
13            int x=idx(s[i]);
14            if(!ch[u][x]){
15                ++sz;
16                clr(ch[sz],0);
17                val[sz]=0;
18                ch[u][x]=sz;
19            }
20            u=ch[u][x];
21            val[u]++;
22        }
23    }
24    int query(char *s){
25        int u=0,n=strlen(s),res=0;
26        rep(i,0,n-1){

```

```

27         int x=idx(s[i]);
28         if(!ch[u][x])break;
29         u=ch[u][x];
30         if(i==n-1)res=val[u];
31     }
32     return res;
33 }
34 }trie;
35 char s[30];
36 int work(){
37     trie.init();
38     int n,m;
39     scanf("%d",&n);
40     while(n--){
41         scanf("%s",s);
42         trie.insert(s);
43     }
44     scanf("%d",&m);
45     while(m--){
46         scanf("%s",s);
47         printf("%d\n",trie.query(s));
48     }
49     return 0;
50 }

```

2.3 Manacher

```

1  //hihocoder 1032
2  const int MAXN=2e6+10;//more than 2 times !
3  char s[MAXN],str[MAXN];
4  int len1,len2,p[MAXN];
5  void init(){
6      str[0]='$';
7      str[1]='#';
8      rep(i,0,len1){
9          str[i*2+2]=s[i];
10         str[i*2+3]='#';
11     }
12     len2=len1*2+2;
13     str[len2]='*';
14 }
15 int manacher(){
16     int id=0,mx=0,ans=0;
17     rep(i,1,len2-1){
18         if(mx>i)p[i]=min(p[2*i-id],mx-i);
19         else p[i]=1;
20         while(str[i+p[i]]==str[i-p[i]])p[i]++;
21         if(i+p[i]>mx){
22             mx=i+p[i];
23             id=i;
24         }
25         ans=max(ans,p[i]);
26     }
27     return ans-1;
28 }
29 int work(){
30     int T;
31     scanf("%d",&T);

```

```

32     while(T--){
33         scanf("%s",s);
34         len1=strlen(s);
35         init();
36         printf("%d\n",manacher());
37     }
38     return 0;
39 }

```

2.4 SaHash

```

1  #include<bits/stdc++.h>
2  using namespace std;
3  typedef unsigned long long ull;
4  const int MAXN=3e5+10;
5  const int P=131;
6  char s[MAXN];
7  int len;
8  ull base[MAXN];
9  ull f[MAXN];
10 int sa[MAXN],height[MAXN];
11 ull H(int l,int r){
12     return f[r]-f[l-1]*base[r-l+1];
13 }
14 int lcp(int x,int y){
15     int l=0,r=min(len-x+1,len-y+1),ans=0;
16     while(l<=r){
17         int mid=(l+r)>>1;
18         if(H(x,x+mid-1)==H(y,y+mid-1)){
19             ans=mid;
20             l=mid+1;
21         }else{
22             r=mid-1;
23         }
24     }
25     return ans;
26 }
27 bool cmp(int x,int y){
28     int d=lcp(x,y);
29     return s[x+d]<s[y+d];
30 }
31 void calc_height(){
32     for(int i=2;i<=len;i++){
33         height[i]=lcp(sa[i-1],sa[i]);
34     }
35 }
36 int main(){
37     scanf("%s",s+1);
38     len=strlen(s+1);
39     base[0]=1;
40     for(int i=1;i<=len;i++){
41         sa[i]=i;
42         base[i]=base[i-1]*P;
43         f[i]=f[i-1]*P+(s[i]-'a'+1);
44     }
45     sort(sa+1,sa+1+len,cmp);
46     calc_height();
47     for(int i=1;i<=len;i++){

```



```

48     printf("%d%c",sa[i]-1," \n"[i==len]);
49 }
50 for(int i=1;i<=len;i++){
51     printf("%d%c",height[i]," \n"[i==len]);
52 }
53 return 0;
54 }

```

2.5 SA

```

1  #include<iostream>
2  #include<cstdio>
3  #include<cstring>
4  #include<algorithm>
5  #include<vector>
6  using namespace std;
7  const int SIZE = 300010, INF = 1 << 30;
8  int a[SIZE], sa[SIZE], rk[SIZE], fir[SIZE], sec[SIZE], c[SIZE], h[SIZE];
9  char str[SIZE];
10 int len;
11 bool comp(int i, int j, int k){
12     return sec[i] == sec[j] && sec[i + k] == sec[j + k];
13 }
14 void sufarr(int n){
15     int i, p, l, m = 200;
16     for (i = 0; i < m; i++) c[i] = 0;
17     for (i = 0; i < n; i++) c[rk[i]] = a[i]++;
18     for (i = 1; i < m; i++) c[i] += c[i - 1];
19     for (i = n - 1; i >= 0; i--) sa[--c[a[i]]] = i;
20     for (l = p = 1; p < n; l *= 2, m = p)
21     {
22         for (p = 0, i = n - l; i < n; i++) sec[p++] = i;
23         for (i = 0; i < n; i++)
24             if (sa[i] >= l) sec[p++] = sa[i] - l;
25         for (i = 0; i < n; i++) fir[i] = rk[sec[i]];
26         for (i = 0; i < m; i++) c[i] = 0;
27         for (i = 0; i < n; i++) c[fir[i]]++;
28         for (i = 1; i < m; i++) c[i] += c[i - 1];
29         for (i = n - 1; i >= 0; i--) sa[--c[fir[i]]] = sec[i];
30         memcpy(sec, rk, sizeof(rk));
31         rk[sa[0]] = 0;
32         for (i = p = 1; i < n; i++)
33             rk[sa[i]] = comp(sa[i], sa[i - 1], l) ? p - 1 : p++;
34     }
35 }
36 void calh(){
37     int i, j, k = 0;
38     for (i = 1; i <= len; i++) rk[sa[i]] = i;
39     for (i = 0; i < len; h[rk[i+1]] = k)
40         for (k ? k-- : 0, j = sa[rk[i] - 1]; a[i + k] == a[j + k]; k++);
41 }
42 int main(){
43     scanf("%s", str);
44     len = strlen(str);
45     for (int i = 0; i < len; i++) a[i] = str[i];
46     a[len] = 0;
47     sufarr(len + 1);
48     calh();

```

```

49     for(int i = 1; i <= len; i++) printf("%d ", sa[i]); puts("");
50     for(int i = 1; i <= len; i++) printf("%d ", h[i]); puts("");
51 }

```

2.6 HashString

```

1  //poj 3974
2  #include<cstdio>
3  #include<algorithm>
4  #include<cstring>
5  using namespace std;
6  typedef unsigned long long ull;
7  const int MAXN=1e6+10;
8  char s[MAXN];
9  ull a[MAXN];
10 ull b[MAXN];
11 ull base[MAXN];
12 inline ull H(int i, int j) {
13     return (a[j] - a[i - 1] * base[j - i + 1]);
14 }
15 inline ull H2(int i, int j) {
16     return (b[j + 1] - b[i] * base[j - i + 1]);
17 }
18 int main(){
19     base[0]=1;
20     for(int i=1;i<MAXN;i++){
21         base[i]=base[i-1]*131;
22     }
23     int kase=0;
24     for(;;){
25         scanf("%s",s+1);
26         if(s[1]=='E')break;
27         int len=strlen(s+1);
28         a[0]=b[len+1]=0;
29         for(int i=1;i<=len;i++){
30             a[i]=a[i-1]*131+s[i]-'a';
31         }
32         for(int i=len;i>=1;i--){
33             b[i]=b[i+1]*131+s[i]-'a';
34         }
35         int ans=1;
36         for(int pos=1;pos<=len;pos++){
37             int l=1,r=min(pos-1,len-pos);
38             while(l<=r){
39                 int mid=(l+r)>>1;
40                 if(H(pos-mid,pos-1)==H2(pos+1,pos+mid)){
41                     ans=max(2*mid+1,ans);
42                     l=mid+1;
43                 }else{
44                     r=mid-1;
45                 }
46             }
47             l=1,r=min(pos-1,len-pos+1);
48             while(l<=r){
49                 int mid=(l+r)>>1;
50                 if(H(pos-mid,pos-1)==H2(pos,pos+mid-1)){
51                     ans=max(2*mid,ans);
52                     l=mid+1;

```

```
53         }else{
54             r=mid-1;
55         }
56     }
57 }
58 printf("Case %d: ",++kase);
59 printf("%d\n",ans);
60 }
61 return 0;
62 }
```

3 Data Structure

3.1 other

3.1.1 QuickSelect

```

1 anytype QuickSelect(anytype arr[],int l,int r,int k){
2     int i=l,j=r,mid=arr[(i+j)>>1];
3     while(i<=j){
4         while(arr[i]<mid)i++;
5         while(arr[j]>mid)j--;
6         if(i<=j){
7             swap(arr[i],arr[j]);
8             i++;
9             j--;
10        }
11    }
12    if(l<j&&k<=j)return QuickSelect(arr,l,j,k);
13    if(i<r&&k>=i)return QuickSelect(arr,i,r,k);
14    return arr[k];
15 }
```

3.1.2 mergingsort

```

1 //hdu 1394
2 const int MAXN=5005;
3 int n;
4 vi A;
5 int x[MAXN];
6 int merging(vi &a){
7     int n=SZ(a);
8     if(n<=1)return 0;
9     int cnt=0;
10    vi b(a.begin(),a.begin()+n/2);
11    vi c(a.begin()+n/2,a.end());
12    cnt+=merging(b);
13    cnt+=merging(c);
14    int ai=0,bi=0,ci=0;
15    while(ai<n){
16        if(bi<SZ(b)&&(ci==SZ(c)||b[bi]<=c[ci])){
17            a[ai++]=b[bi++];
18        }else{
19            cnt+=n/2-bi;
20            a[ai++]=c[ci++];
21        }
22    }
23    return cnt;
24 }
25 int work(){
26     while(~scanf("%d",&n)){
27         A.clear();
28         rep(i,1,n)scanf("%d",&x[i]),A.pb(x[i]);
29         int sum=merging(A);
30         int res=sum;
31         rep(i,1,n){
32             sum=sum-x[i]+(n-1-x[i]);
33             res=min(res,sum);
34         }
35         printf("%d\n",res);
36     }
```

```

36     }
37     return 0;
38 }

```

3.1.3 pbds

```

1  //cf 1042d
2  #include<bits/stdc++.h>
3  #include<ext/pb_ds/assoc_container.hpp>
4  using namespace std;
5  using namespace __gnu_pbds;
6  typedef long long ll;
7  tree<pair<ll,int>,null_type,less<pair<ll,int> >,rb_tree_tag,
   tree_order_statistics_node_update > rbt;
8  int main(){
9      int n;
10     ll t;
11     scanf("%d%I64d",&n,&t);
12     rbt.insert({0,0});
13     ll now=0,ans=0;
14     for(int i=1;i<=n;i++){
15         ll x;
16         scanf("%I64d",&x);
17         now+=x;
18         ans+=i-rbt.order_of_key({now-t,n+1});
19         rbt.insert({now,i});
20     }
21     printf("%I64d",ans);
22     return 0;
23 }

```

3.1.4 stack

```

1  //poj 2559
2  #include<cstdio>
3  #include<algorithm>
4  using namespace std;
5  typedef long long ll;
6  const int MAXN=1e5+10;
7  int a[MAXN];
8  int w[MAXN];
9  int stk[MAXN];
10 int top;
11 int main(){
12     int n;
13     while(scanf("%d",&n),n){
14         ll ans=0;
15         top=0;
16         stk[top]=0;
17         for(int i=1;i<=n+1;i++){
18             if(i<=n)scanf("%d",&a[i]);
19             else a[i]=0;
20             if(a[i]>a[stk[top]]){
21                 stk[++top]=i;
22                 w[top]=1;
23             }else{
24                 int width=0;
25                 while(a[i]<a[stk[top]]){

```

```

26         width+=w[top];
27         ans=max(ans,1LL*a[stk[top]]*width);
28         top--;
29     }
30     stk[++top]=i;
31     w[top]=width+1;
32 }
33 }
34 printf("%lld\n",ans);
35 }
36 return 0;
37 }

```

3.1.5 queue

```

1 //ch 1201
2 #include<bits/stdc++.h>
3 using namespace std;
4 typedef long long ll;
5 const int MAXN=3e5+10;
6 ll sum[MAXN];
7 int que[MAXN];
8 int st,ed;
9 int main(){
10     int n,m;
11     scanf("%d%d",&n,&m);
12     sum[0]=0;
13     st=ed=0;
14     que[ed++]=0;
15     ll ans=0;
16     for(int i=1;i<=n;i++){
17         scanf("%lld",&sum[i]);
18         sum[i]+=sum[i-1];
19         while(i-que[st]>m){
20             st++;
21         }
22         ans=max(ans,sum[i]-sum[que[st]]);
23         while(st!=ed&&sum[que[ed-1]]>=sum[i]){
24             ed--;
25         }
26         que[ed++]=i;
27     }
28     printf("%lld",ans);
29     return 0;
30 }

```

3.2 Binary Indexed Tree

```

1 //add(pos,a) sum(r)-sum(l-1)
2 //add(l,a) add(r+1,-a) sum(pos)
3 const int MAXN=100000;
4 struct BIT{
5     int n,c[MAXN<<1];
6     void init(int _n){
7         n=_n;
8         rep(i,0,n)c[i]=0;
9     }

```

```

10 void update(int i,int v){
11     for(;i<=n;i+=i&-i)c[i]+=v;
12 }
13 int query(int i){
14     int s=0;
15     for(;i;i-=i&-i)s+=c[i];
16     return s;
17 }
18 int findpos(int v){
19     int sum=0;
20     int pos=0;
21     int i=1;
22     for(;i<n;i<=1);
23     for(;i;i>=1){
24         if(pos+i<=n&&sum+c[pos+i]<v){
25             sum+=c[pos+i];
26             pos+=i;
27         }
28     }
29     return pos+1;
30 }
31 }bit;

```

3.2.1 poj3468

$$a_i = \sum_{i=1}^x d_i$$

$$\sum_{i=1}^x a_i = \sum_{i=1}^x \sum_{j=1}^i d_j = \sum_{i=1}^x (x-i+1)d_i$$

$$\sum_{i=1}^x a_i = (x+1) \sum_{i=1}^x d_i - \sum_{i=1}^x d_i \times i$$

```

1  const int MAXN=1e5+5;
2  int n,q,x,y,z;
3  long long c1[MAXN],c2[MAXN];
4  void add(int x,int y){
5      for(int i=x;i<=n;i+=i&(-i))c1[i]+=y,c2[i]+=1LL*x*y;
6  }
7  ll sum(int x){
8      ll ans(0);
9      for(int i=x;i;i-=i&(-i))ans+=1LL*(x+1)*c1[i]-c2[i];
10     return ans;
11 }
12 char op[5];
13 int work(){
14     scanf("%d%d",&n,&q);
15     int a1,a2;
16     a1=0;
17     rep(i,1,n){
18         scanf("%d",&a2);
19         add(i,a2-a1);
20         a1=a2;
21     }
22     while(q--){
23         scanf("%s",op);
24         if(op[0]=='Q'){
25             scanf("%d%d%d",&x,&y,&z);
26             printf("%lld\n",sum(y)-sum(x-1));
27         }else{
28             scanf("%d%d%d",&x,&y,&z);

```

```

29         add(x,z);
30         add(y+1,-z);
31     }
32 }
33 return 0;
34 }

```

3.3 Segment Tree

```

1 #define lson rt<<1
2 #define rson rt<<1|1
3 #define le l,m,lson
4 #define ri m+1,r,rson
5 #define mid m=(l+r)>>1

```

3.3.1 Single-point Update

```

1 const int MAXN=5e4+5;
2 int sum[MAXN<<2];
3 void push_up(int rt){
4     sum[rt]=sum[lson]+sum[rson];
5 }
6 void build(int l,int r,int rt){
7     if(l==r){
8         scanf("%d",&sum[rt]);
9         return;
10    }
11    int mid;
12    build(le);
13    build(ri);
14    push_up(rt);
15 }
16 void update(int p,int v,int l,int r,int rt){
17     if(l==r){
18         sum[rt]+=v;
19         return;
20     }
21     int mid;
22     if(p<=m)update(p,v,le);
23     else update(p,v,ri);
24     push_up(rt);
25 }
26 int query(int L,int R,int l,int r,int rt){
27     if(L<=l&&r<=R){
28         return sum[rt];
29     }
30     int mid;
31     int ret=0;
32     if(L<=m)ret+=query(L,R,le);
33     if(R>m)ret+=query(L,R,ri);
34     return ret;
35 }

```

3.3.2 Interval Update

```

1 const int MAXN=1e5+5;
2 ll lazy[MAXN<<2];

```



```

3 ll tree[MAXN<<2];
4 void push_up(int rt){
5     tree[rt]=tree[lson]+tree[rson];
6 }
7 void push_down(int rt,int m){
8     ll w=lazy[rt];
9     if(w){
10         lazy[lson]+=w;
11         lazy[rson]+=w;
12         tree[lson]+=w*(m-(m>>1));
13         tree[rson]+=w*(m>>1);
14         lazy[rt]=0;
15     }
16 }
17 void build(int l,int r,int rt){
18     lazy[rt]=0;
19     if(l==r){
20         scanf("%lld",&tree[rt]);
21         return;
22     }
23     int mid;
24     build(le);
25     build(ri);
26     push_up(rt);
27 }
28 void update(int L,int R,int v,int l,int r,int rt){
29     if(L<=l&&r<=R){
30         lazy[rt]+=v;
31         tree[rt]+=1ll*v*(r-l+1);
32         return;
33     }
34     push_down(rt,r-l+1);
35     int mid;
36     if(L<=m)update(L,R,v,le);
37     if(R>m)update(L,R,v,ri);
38     push_up(rt);
39 }
40 ll query(int L,int R,int l,int r,int rt){
41     if(L<=l&&r<=R){
42         return tree[rt];
43     }
44     push_down(rt,r-l+1);
45     int mid;
46     ll ret=0;
47     if(L<=m)ret+=query(L,R,le);
48     if(R>m)ret+=query(L,R,ri);
49     return ret;
50 }

```

3.4 Splay Tree

```

1 #define key_value ch[ch[rt][1]][0]
2 const int MAXN=1e5;
3 struct Splay{
4     int a[MAXN]; //0 base
5     int sz[MAXN],ch[MAXN][2],fa[MAXN];
6     int key[MAXN],rev[MAXN];
7     int rt,tot;

```

```

8   int stk[MAXN],top;
9   void push_up(int x){
10      sz[x]=sz[ch[x][0]]+sz[ch[x][1]]+1;
11  }
12  void push_down(int x){
13      if(rev[x]){
14          swap(ch[x][0],ch[x][1]);
15          if(ch[x][0])rev[ch[x][0]]^=1;
16          if(ch[x][1])rev[ch[x][1]]^=1;
17          rev[x]=0;
18      }
19  }
20  int newnode(int p=0,int k=0){
21      int x=top?stk[top--]:++tot;
22      fa[x]=p;
23      sz[x]=1;
24      ch[x][0]=ch[x][1]=0;
25      key[x]=k;
26      rev[x]=0;
27      return x;
28  }
29  int build(int l,int r,int p){
30      if(l>r)return 0;
31      int mid=(l+r)>>1;
32      int x=newnode(p,a[mid]);
33      ch[x][0]=build(l,mid-1,x);
34      ch[x][1]=build(mid+1,r,x);
35      push_up(x);
36      return x;
37  }
38  void init(int n){
39      tot=0,top=0;
40      rt=newnode(0,-1);
41      ch[rt][1]=newnode(rt,-1);
42      rep(i,0,n-1)a[i]=i+1;
43      key_value=build(0,n-1,ch[rt][1]);
44      push_up(ch[rt][1]);
45      push_up(rt);
46  }
47  void rotate(int x,int d){
48      int y=fa[x];
49      push_down(y);
50      push_down(x);
51      ch[y][d^1]=ch[x][d];
52      fa[ch[x][d]]=y;
53      if(fa[y])ch[fa[y]][ch[fa[y]][1]==y]=x;
54      fa[x]=fa[y];
55      ch[x][d]=y;
56      fa[y]=x;
57      push_up(y);
58  }
59  void splay(int x,int goal=0){
60      push_down(x);
61      while(fa[x]!=goal){
62          if(fa[fa[x]]==goal){
63              rotate(x,ch[fa[x]][0]==x);
64          }else{
65              int y=fa[x];
66              int d=ch[fa[y]][0]==y;

```

```

67         ch[y][d]==x?rotate(x,d^1):rotate(y,d);
68         rotate(x,d);
69     }
70 }
71 push_up(x);
72 if(goal==0)rt=x;
73 }
74 int kth(int r,int k){
75     push_down(r);
76     int t=sz[ch[r][0]]+1;
77     if(t==k)return r;
78     return t>k?kth(ch[r][0],k):kth(ch[r][1],k-t);
79 }
80 void select(int l,int r){
81     splay(kth(rt,1),0);
82     splay(kth(ch[rt][1],r-l+2),rt);
83 }
84 };

```

3.5 Functional Segment Tree

```

1 //poj 2104
2 const int MAXN=1e5+6;
3 int n,m,cnt,x,y,k,root[MAXN],a[MAXN];
4 struct node{int l,r,sum;}T[MAXN*40];
5 vi v;
6 int getid(int x){return lower_bound(all(v),x)-v.begin()+1;}
7 void update(int l,int r,int &x,int y,int pos){
8     x=++cnt;
9     T[x]=T[y];
10    T[x].sum++;
11    if(l==r)return;
12    int mid=(l+r)>>1;
13    if(mid>=pos)update(l,mid,T[x].l,T[y].l,pos);
14    else update(mid+1,r,T[x].r,T[y].r,pos);
15 }
16 int query(int l,int r,int x,int y,int k){
17     if(l==r)return l;
18     int sum=T[T[y].l].sum-T[T[x].l].sum;
19     int mid=(l+r)>>1;
20     if(sum>=k)return query(l,mid,T[x].l,T[y].l,k);
21     else return query(mid+1,r,T[x].r,T[y].r,k-sum);
22 }
23 int work(){
24     scanf("%d%d",&n,&m);
25     v.clear();
26     rep(i,1,n)scanf("%d",&a[i]),v.pb(a[i]);
27     sort(all(v)),v.erase(unique(all(v)),v.end());
28     cnt=0;
29     rep(i,1,n)update(1,n,root[i],root[i-1],getid(a[i]));
30     rep(i,1,m)scanf("%d%d%d",&x,&y,&k),printf("%d\n",v[query(1,n,root[x-1],root[y],k)-1]);
31     return 0;
32 }

```

3.6 Sparse Table

```

1 //Frequent values UVA - 11235
2 #include<bits/stdc++.h>
3 using namespace std;
4 const int MAXN=1e5+10;
5 int dp[MAXN][33];
6 int a[MAXN],b[MAXN],Belong[MAXN];
7 int rmq(int l,int r){
8     int k=31-__builtin_clz(r-l+1);
9     return max(dp[l][k],dp[r-(1<<k)+1][k]);
10 }
11 int main(){
12     int n;
13     while(scanf("%d",&n),n){
14         int q;
15         scanf("%d",&q);
16         int index=0;
17         int now=-111111;
18         for(int i=1;i<=n;i++){
19             int x;
20             scanf("%d",&x);
21             if(now!=x){
22                 index++;
23                 now=x;
24                 a[index]=i;
25             }
26             Belong[i]=index;
27             b[index]=i;
28         }
29         for(int i=1;i<=index;i++){
30             dp[i][0]=b[i]-a[i]+1;
31         }
32         for (int j = 1; (1 << j) <= index; j++){
33             for (int i = 1; i + (1 << j) - 1 <= index; i++){
34                 dp[i][j] = max(dp[i][j - 1], dp[i + (1 << (j - 1))][j - 1]);
35             }
36         }
37         while(q--){
38             int l,r;
39             scanf("%d%d",&l,&r);
40             if(Belong[l]==Belong[r]){
41                 printf("%d\n",r-l+1);
42             }else{
43                 int pos1=Belong[l];
44                 int ans=b[pos1]-l+1;
45                 int pos2=Belong[r];
46                 ans=max(ans,r-a[pos2]+1);
47                 pos1++;
48                 pos2--;
49                 if(pos1<=pos2){
50                     ans=max(ans,rmq(pos1,pos2));
51                 }
52                 printf("%d\n",ans);
53             }
54         }
55     }
56     return 0;
57 }
58 }

```

3.7 block

```

1 //poj 3468
2 #include <algorithm>
3 #include <iostream>
4 #include <cstring>
5 #include <string>
6 #include <cstdio>
7 #include <vector>
8 #include <stack>
9 #include <queue>
10 #include <cmath>
11 #include <set>
12 #include <map>
13 using namespace std;
14 #define rep(i,a,b) for(int i=a;i<=b;i++)
15 #define per(i,a,b) for(int i=a;i>=b;i--)
16 #define clr(a,x) memset(a,x,sizeof(a))
17 #define pb push_back
18 #define all(x) (x).begin(),(x).end()
19 #define fi first
20 #define se second
21 #define SZ(x) ((int)(x).size())
22 typedef unsigned long long ull;
23 typedef long long ll;
24 typedef vector<int> vi;
25 typedef pair<int,int> pii;
26 /*****head*****/
27 const int MAXN=1e5+10;
28 int L[MAXN],R[MAXN],pos[MAXN];
29 ll a[MAXN],b[MAXN],c[MAXN];
30 int t;
31 void update(int x,int y,int z){
32     int l=pos[x];
33     int r=pos[y];
34     if(l==r){
35         for(int i=x;i<=y;i++){
36             a[i]+=z;
37         }
38         b[l]+=1LL*z*(y-x+1);
39     }else{
40         for(int i=l+1;i<r;i++){
41             c[i]+=z;
42         }
43         for(int i=x;i<=R[l];i++){
44             a[i]+=z;
45         }
46         b[l]+=1LL*z*(R[l]-x+1);
47         for(int i=L[r];i<=y;i++){
48             a[i]+=z;
49         }
50         b[r]+=1LL*z*(y-L[r]+1);
51     }
52 }
53 ll query(int x,int y){
54     ll res=0;
55     int l=pos[x];
56     int r=pos[y];
57     if(l==r){

```

```

58     for(int i=x;i<=y;i++){
59         res+=a[i];
60     }
61     res+=c[l]*(y-x+1);
62 }else{
63     for(int i=l+1;i<r;i++){
64         res+=c[i]*(R[i]-L[i]+1)+b[i];
65     }
66     for(int i=x;i<=R[l];i++){
67         res+=a[i];
68     }
69     res+=c[l]*(R[l]-x+1);
70     for(int i=L[r];i<=y;i++){
71         res+=a[i];
72     }
73     res+=c[r]*(y-L[r]+1);
74 }
75 return res;
76 }
77 int main(){
78     int n,q;
79     scanf("%d%d",&n,&q);
80     t=sqrt(n);
81     for(int i=1;i<=t;i++){
82         L[i]=(i-1)*t+1;
83         R[i]=i*t;
84     }
85     if(R[t]<n){
86         t++;
87         L[t]=R[t-1]+1;
88         R[t]=n;
89     }
90     for(int i=1;i<=n;i++){
91         scanf("%lld",&a[i]);
92     }
93     for(int i=1;i<=t;i++){
94         for(int j=L[i];j<=R[i];j++){
95             pos[j]=i;
96             b[i]+=a[j];
97         }
98     }
99     char op[5];
100    while(q--){
101        int x,y;
102        scanf("%s%d%d",op,&x,&y);
103        if(op[0]=='Q'){
104            printf("%lld\n",query(x,y));
105        }else{
106            int z;
107            scanf("%d",&z);
108            update(x,y,z);
109        }
110    }
111    return 0;
112 }

```

3.8 Treap

```

1  #include <algorithm>
2  #include <iostream>
3  #include <cstring>
4  #include <string>
5  #include <cstdio>
6  #include <vector>
7  #include <stack>
8  #include <queue>
9  #include <cmath>
10 #include <set>
11 #include <map>
12 using namespace std;
13 #define rep(i,a,b) for(int i=a;i<=b;i++)
14 #define per(i,a,b) for(int i=a;i>=b;i--)
15 #define clr(a,x) memset(a,x,sizeof(a))
16 #define pb push_back
17 #define all(x) (x).begin(),(x).end()
18 #define fi first
19 #define se second
20 #define SZ(x) ((int)(x).size())
21 typedef unsigned long long ull;
22 typedef long long ll;
23 typedef vector<int> vi;
24 typedef pair<int,int> pii;
25 /*****head*****/
26 const int SIZE=1e5+10;
27 struct Treap{
28     int l,r;
29     int val,dat;
30     int cnt,sz;
31 }a[SIZE];
32 int tot,root,n,INF=0x7fffffff;
33 int New(int val){
34     a[++tot].val=val;
35     a[tot].dat=rand();
36     a[tot].cnt=a[tot].sz=1;
37     return tot;
38 }
39 void Update(int p){
40     a[p].sz=a[a[p].l].sz+a[a[p].r].sz+a[p].cnt;
41 }
42 void Build(){
43     New(-INF);
44     New(INF);
45     root=1;
46     a[1].r=2;
47     Update(root);
48 }
49 int GetRankByVal(int p,int val){
50     if(p==0)return 0;
51     if(val==a[p].val)return a[a[p].l].sz+1;
52     if(val<a[p].val)return GetRankByVal(a[p].l,val);
53     return GetRankByVal(a[p].r,val)+a[a[p].l].sz+a[p].cnt;
54 }
55 int GetValByRank(int p,int rk){
56     if(p==0)return INF;
57     if(a[a[p].l].sz>=rk)return GetValByRank(a[p].l,rk);
58     if(a[a[p].l].sz+a[p].cnt>=rk)return a[p].val;
59     return GetValByRank(a[p].r,rk-a[a[p].l].sz-a[p].cnt);

```

```

60 }
61 void zig(int &p){
62     int q=a[p].l;
63     a[p].l=a[q].r;
64     a[q].r=p;
65     p=q;
66     Update(a[p].r);
67     Update(p);
68 }
69 void zag(int &p){
70     int q=a[p].r;
71     a[p].r=a[q].l;
72     a[q].l=p;
73     p=q;
74     Update(a[p].l);
75     Update(p);
76 }
77 void Insert(int &p,int val){
78     if(p==0){
79         p=New(val);
80         return;
81     }
82     if(val==a[p].val){
83         a[p].cnt++;
84         Update(p);
85         return;
86     }
87     if(val<a[p].val){
88         Insert(a[p].l,val);
89         if(a[p].dat<a[a[p].l].dat)zig(p);
90     }else{
91         Insert(a[p].r,val);
92         if(a[p].dat<a[a[p].r].dat)zag(p);
93     }
94     Update(p);
95 }
96 int GetPre(int val){
97     int ans=1;
98     int p=root;
99     while(p){
100         if(val==a[p].val){
101             if(a[p].l>0){
102                 p=a[p].l;
103                 while(a[p].r>0)p=a[p].r;
104                 ans=p;
105             }
106             break;
107         }
108         if(a[p].val<val&&a[p].val>a[ans].val)ans=p;
109         p=val<a[p].val?a[p].l:a[p].r;
110     }
111     return a[ans].val;
112 }
113 int GetNext(int val){
114     int ans=2;
115     int p=root;
116     while(p){
117         if(val==a[p].val){
118             if(a[p].r>0){

```



```

119         p=a[p].r;
120         while(a[p].l>0)p=a[p].l;
121         ans=p;
122     }
123     break;
124 }
125 if(a[p].val>val&& a[p].val<a[ans].val)ans=p;
126 p=val<a[p].val?a[p].l:a[p].r;
127 }
128 return a[ans].val;
129 }
130 void Remove(int &p,int val){
131     if(p==0)return;
132     if(val==a[p].val){
133         if(a[p].cnt>1){
134             a[p].cnt--;
135             Update(p);
136             return;
137         }
138         if(a[p].l||a[p].r){
139             if(a[p].r==0||a[a[p].l].dat>a[a[p].r].dat){
140                 zig(p);
141                 Remove(a[p].r,val);
142             }else{
143                 zag(p);
144                 Remove(a[p].l,val);
145             }
146             Update(p);
147         }else{
148             p=0;
149         }
150         return;
151     }
152     val<a[p].val?Remove(a[p].l,val):Remove(a[p].r,val);
153     Update(p);
154 }
155 int main(){
156     Build();
157     int n;
158     scanf("%d",&n);
159     while(n--){
160         int op,x;
161         scanf("%d%d",&op,&x);
162         switch(op){
163             case 1:
164                 Insert(root,x);
165                 break;
166             case 2:
167                 Remove(root,x);
168                 break;
169             case 3:
170                 printf("%d\n",GetRankByVal(root,x)-1);
171                 break;
172             case 4:
173                 printf("%d\n",GetValByRank(root,x+1));
174                 break;
175             case 5:
176                 printf("%d\n",GetPre(x));
177                 break;

```

```
178         case 6:
179             printf("%d\n",GetNext(x));
180             break;
181         }
182     }
183     return 0;
184 }
```

4 Graph Theory

4.1 Union-Find Set

```

1  const int MAXN=1e6+5;
2  struct DSU{
3      int p[MAXN];
4      void init(int n){rep(i,0,n)p[i]=i;}
5      int findp(int x){return x==p[x]?x:p[x]=findp(p[x]);}
6      void unite(int x,int y){x=findp(x);y=findp(y);if(x==y)return;p[y]=x;}
7      bool same(int x,int y){return findp(x)==findp(y);}
8  }dsu;

```

4.2 Minimal Spanning Tree

4.2.1 Kruskal

```

1  //poj 1258
2  #include<cstdio>
3  #include<algorithm>
4  using namespace std;
5  const int MAXE=1e5+5;
6  const int MAXN=1e5+5;
7  struct DSU{
8      int p[MAXN];
9      void init(int n){for(int i=0;i<=n;i++)p[i]=i;}
10     int findp(int x){return x==p[x]?x:p[x]=findp(p[x]);}
11     void unite(int x,int y){x=findp(x);y=findp(y);if(x==y)return;p[y]=x;}
12     bool same(int x,int y){return findp(x)==findp(y);}
13 }dsu;
14 struct edge{int u,v,cost;}es[MAXE];
15 bool cmp(const edge &x,const edge &y){return x.cost<y.cost;}
16 int V,E;
17 int kruskal(){
18     sort(es,es+E,cmp);
19     dsu.init(V);
20     int res=0;
21     for(int i=0;i<E;i++){
22         if(!dsu.same(es[i].u,es[i].v)){
23             dsu.unite(es[i].u,es[i].v);
24             res+=es[i].cost;
25         }
26     }
27     return res;
28 }
29 int main(){
30     while(~scanf("%d",&V)){
31         E=0;
32         for(int i=1;i<=V;i++){
33             for(int j=1;j<=V;j++){
34                 int w;
35                 scanf("%d",&w);
36                 if(i==j)continue;
37                 es[E].u=i;
38                 es[E].v=j;
39                 es[E].cost=w;
40                 E++;
41             }
42         }
43     }

```

```

42     }
43     printf("%d\n",kruskal());
44 }
45 return 0;
46 }

```

4.3 Shortest Path

4.3.1 Dijkstra

```

1  #include<bits/stdc++.h>
2  using namespace std;
3  #define rep(i,a,b) for(int i=a;i<=b;i++)
4  #define clr(a,x) memset(a,x,sizeof(a))
5  #define mp make_pair
6  const int MAXV=2e6;
7  const int MAXE=5e6+10;
8  typedef long long anytype;
9  typedef pair<anytype,int> P;
10 int tot=0;
11 int head[MAXV];
12 struct Edge{
13     int v,c,nxt;
14     Edge(){}
15     Edge(int v,int c,int nxt):v(v),c(c),nxt(nxt){}
16 }edge[MAXE];
17 void init(){
18     tot=0;
19     clr(head,-1);
20 }
21 void add_edge(int u,int v,int c){
22     edge[tot]=Edge(v,c,head[u]);
23     head[u]=tot++;
24 }
25 anytype d[MAXV];
26 void dij(int s){
27     priority_queue<P,vector<P>,greater<P> > que;
28     clr(d,-1);
29     d[s]=0;
30     que.push(P(0,s));
31     while(!que.empty()){
32         P t=que.top();
33         que.pop();
34         int v=t.second;
35         if(d[v]!=-1&&d[v]<t.first)continue;
36         for(int i=head[v];~i;i=edge[i].nxt){
37             Edge e=edge[i];
38             if(d[e.v]==-1||d[e.v]>d[v]+e.c){
39                 d[e.v]=d[v]+e.c;
40                 que.push(mp(d[e.v],e.v));
41             }
42         }
43     }
44 }
45 int main(){
46     int T;
47     scanf("%d",&T);
48     while(T--){
49         int n,m,k;

```

```

50     scanf("%d%d%d",&n,&m,&k);
51     init();
52     rep(i,1,m){
53         int u,v,c;
54         scanf("%d%d%d",&u,&v,&c);
55         rep(j,0,k){
56             add_edge(u+j*n,v+j*n,c);
57             if(j!=k)add_edge(u+j*n,v+(j+1)*n,0);
58         }
59     }
60     dij(1);
61     printf("%lld\n",d[n+k*n]);
62 }
63 return 0;
64 }

```

4.3.2 Spfa

```

1 //hdu3592
2 const int MAXN=1e3+5;
3 const int MAXE=3e4+5;
4 const int INF=0x3f3f3f3f;
5 int N,X,Y;
6 int tot;
7 int head[MAXN];
8 struct Edge{
9     int v,w,nxt;
10     Edge(){}
11     Edge(int v,int w,int nxt):v(v),w(w),nxt(nxt){}
12 }edge[MAXE];
13 void init(){
14     tot=0;
15     clr(head,-1);
16 }
17 void add_edge(int u,int v,int w){
18     edge[tot]=Edge(v,w,head[u]);
19     head[u]=tot++;
20 }
21 queue<int> que;
22 bool inq[MAXN];
23 int qtime[MAXN];
24 int d[MAXN];
25 int spfa(){
26     while(!que.empty())que.pop();
27     clr(qtime,0);
28     clr(inq,0);
29     rep(i,1,N)d[i]=INF;
30     d[1]=0;
31     que.push(1);
32     inq[1]=1;
33     qtime[1]++;
34     while(!que.empty()){
35         int u=que.front();
36         que.pop();
37         inq[u]=0;
38         for(int i=head[u];i!=-1;i=edge[i].nxt){
39             int v=edge[i].v;
40             int w=edge[i].w;

```

```

41         if(d[v]>d[u]+w){
42             d[v]=d[u]+w;
43             if(!inq[v]){
44                 que.push(v);
45                 inq[v]=1;
46                 qtime[v]++;
47                 if(qtime[v]>N)return -1;
48             }
49         }
50     }
51 }
52 if(d[N]==INF)return -2;
53 else return d[N];
54 }
55 int work(){
56     int T;
57     scanf("%d",&T);
58     while(T--){
59         scanf("%d%d%d",&N,&X,&Y);
60         init();
61         rep(i,1,N-1){
62             add_edge(i+1,i,0);
63         }
64         while(X--){
65             int x,y,z;
66             scanf("%d%d%d",&x,&y,&z);
67             add_edge(x,y,z);
68         }
69         while(Y--){
70             int x,y,z;
71             scanf("%d%d%d",&x,&y,&z);
72             add_edge(y,x,-z);
73         }
74         printf("%d\n",spfa());
75     }
76     return 0;
77 }

```

4.3.3 kth-p

```

1  #include<bits/stdc++.h>
2  using namespace std;
3  #define INF 0xffffffff
4  #define MAXN 100010
5  struct node{
6      int to;
7      int val;
8      int next;
9  };
10 struct node2{
11     int to;
12     int g,f;
13     bool operator<(const node2 &r ) const {
14         if(r.f==f)
15             return r.g<g;
16         return r.f<f;
17     }
18 };

```

```

19 node edge[MAXN], edge2[MAXN];
20 int n, m, s, t, k, cnt, cnt2, ans;
21 int dis[1010], visit[1010], head[1010], head2[1010];
22 void init(){
23     memset(head, -1, sizeof(head));
24     memset(head2, -1, sizeof(head2));
25     cnt = cnt2 = 1;
26 }
27 void addedge(int from, int to, int val){
28     edge[cnt].to = to;
29     edge[cnt].val = val;
30     edge[cnt].next = head[from];
31     head[from] = cnt++;
32 }
33 void addedge2(int from, int to, int val){
34     edge2[cnt2].to = to;
35     edge2[cnt2].val = val;
36     edge2[cnt2].next = head2[from];
37     head2[from] = cnt2++;
38 }
39 bool spfa(int s, int n, int head[], node edge[], int dist[]) {
40     queue<int> Q1;
41     int inq[1010];
42     for(int i = 0; i <= n; i++) {
43         dis[i] = INF;
44         inq[i] = 0;
45     }
46     dis[s] = 0;
47     Q1.push(s);
48     inq[s]++;
49     while(!Q1.empty()) {
50         int q = Q1.front();
51         Q1.pop();
52         inq[q]--;
53         if(inq[q] > n)
54             return false;
55         int k = head[q];
56         while(k >= 0) {
57             if(dist[edge[k].to] > dist[q] + edge[k].val) {
58                 dist[edge[k].to] = edge[k].val + dist[q];
59                 if(!inq[edge[k].to]) {
60                     inq[edge[k].to]++;
61                     Q1.push(edge[k].to);
62                 }
63             }
64             k = edge[k].next;
65         }
66     }
67     return true;
68 }
69 int A_star(int s, int t, int n, int k, int head[], node edge[], int dist[]) {
70     node2 e, ne;
71     int cnt = 0;
72     priority_queue<node2> Q;
73     if(s == t)
74         k++;
75     if(dis[s] == INF)
76         return -1;
77     e.to = s;

```

```

78     e.g=0;
79     e.f=e.g+dis[e.to];
80     Q.push(e);
81
82     while(!Q.empty()) {
83         e=Q.top();
84         Q.pop();
85         if(e.to==t)//00000000
86         {
87             cnt++;
88         }
89         if(cnt==k)//00k00
90         {
91             return e.g;
92         }
93         for(int i=head[e.to]; i!=-1; i=edge[i].next) {
94             ne.to=edge[i].to;
95             ne.g=e.g+edge[i].val;
96             ne.f=ne.g+dis[ne.to];
97             Q.push(ne);
98         }
99     }
100     return -1;
101 }
102 int main(){
103     while(~scanf("%d%d",&n,&m)){
104         init();
105         for(int i=1;i<=m;i++){
106             int a,b,c;
107             scanf("%d%d%d",&a,&b,&c);
108             addedge(a,b,c);
109             addedge2(b,a,c);
110         }
111         scanf("%d%d%d",&s,&t,&k);
112         spfa(t,n,head2,edge2,dis);
113         ans=A_star(s,t,n,k,head,edge,dis);
114         printf("%d\n",ans);
115     }
116     return 0;
117 }

```

4.4 Topo Sort

```

1  //cf 915D
2  const int MAXN=505;
3  const int MAXM=1e5+5;
4  int n,m;
5  int tot;
6  int head[MAXN],cur[MAXN],idec[MAXN];
7  struct Edge{
8      int v,nxt;
9      Edge(){}
10     Edge(int v,int nxt):v(v),nxt(nxt){}
11 }edge[MAXM];
12 void init(){
13     tot=0;
14     clr(head,-1);
15 }

```



```

16 void add_edge(int u,int v){
17     edge[tot]=Edge(v,head[u]);
18     head[u]=tot++;
19 }
20 int que[MAXN];
21 int st,ed;
22 bool topsort(int x){
23     int nst=1,ned=0;
24     rep(i,1,n)cur[i]=idec[i];
25     cur[x]--;
26     que[++ned]=x;
27     while(nst<=ned){
28         int u=que[nst++];
29         for(int i=head[u];i!=-1;i=edge[i].nxt){
30             int v=edge[i].v;
31             if(--cur[v]==0)que[++ned]=v;
32         }
33     }
34     if(ned+ed==n)return true;
35     else return false;
36 }
37 int work(){
38     scanf("%d%d",&n,&m);
39     init();
40     while(m--){
41         int u,v;
42         scanf("%d%d",&u,&v);
43         add_edge(u,v);
44         idec[v]++;
45     }
46     st=1,ed=0;
47     rep(i,1,n){
48         if(idec[i]==0)que[++ed]=i;
49     }
50     while(st<=ed){
51         int u=que[st++];
52         for(int i=head[u];i!=-1;i=edge[i].nxt){
53             int v=edge[i].v;
54             if(--idec[v]==0)que[++ed]=v;
55         }
56     }
57     if(ed==n){
58         puts("YES");
59         return 0;
60     }
61     rep(i,1,n){
62         if(idec[i]==1){
63             if(topsort(i)){
64                 puts("YES");
65                 return 0;
66             }
67         }
68     }
69     puts("NO");
70     return 0;
71 }

```

4.5 LCA

4.5.1 LCA

```

1 //hdu 2586
2 const int MAXV=1e5+100;
3 int tot;
4 int head[MAXV];
5 struct Edge{
6     int v,w,nxt;
7     Edge(){}
8     Edge(int v,int w,int nxt):v(v),w(w),nxt(nxt){}
9 }edge[MAXV<<1];
10 void init(){
11     tot=0;
12     memset(head,-1,sizeof(head));
13 }
14 void add_edge(int u,int v,int w){
15     edge[tot]=Edge(v,w,head[u]);
16     head[u]=tot++;
17 }
18 int t,f[MAXV][22],d[MAXV];
19 ll dist[MAXV];
20 void bfs(){
21     queue<int> que;
22     que.push(1);
23     d[1]=1;
24     while(!que.empty()){
25         int u=que.front();
26         que.pop();
27         for(int i=head[u];~i;i=edge[i].nxt){
28             int v=edge[i].v;
29             if(d[v])continue;
30             d[v]=d[u]+1;
31             dist[v]=dist[u]+edge[i].w;
32             f[v][0]=u;
33             for(int j=1;j<=t;j++){
34                 f[v][j]=f[f[v][j-1]][j-1];
35             }
36             que.push(v);
37         }
38     }
39 }
40 int lca(int x,int y){
41     if(d[x]>d[y])swap(x,y);
42     for(int i=t;i>=0;i--){
43         if(d[f[y][i]]>=d[x])y=f[y][i];
44     }
45     if(x==y)return x;
46     for(int i=t;i>=0;i--){
47         if(f[x][i]!=f[y][i]){
48             x=f[x][i];
49             y=f[y][i];
50         }
51     }
52     return f[x][0];
53 }
54 int main() {
55     int T;

```

```

56     cin>>T;
57     while (T--) {
58         int n,m;
59         cin >> n >> m;
60         t = (int)(log(n) / log(2)) + 1;
61         init();
62         memset(d,0,sizeof(d));
63         for (int i = 1; i < n; i++) {
64             int x, y, z;
65             scanf("%d%d%d", &x, &y, &z);
66             add_edge(x, y, z), add_edge(y, x, z);
67         }
68         bfs();
69         for (int i = 1; i <= m; i++) {
70             int x, y;
71             scanf("%d%d", &x, &y);
72             printf("%lld\n", dist[x] + dist[y] - 2 * dist[lca(x, y)]);
73         }
74     }
75     return 0;
76 }

```

4.6 Depth-First Traversal

```

1  vector<int> G[MAXN];
2  int vis[MAXN];
3  void dfs(int u){
4      vis[u]=1;
5      PREVISIT(u);
6      for(auto v:G[u]){
7          if(!vis[v])dfs(v);
8      }
9      POSTVISIT(u);
10 }

```

4.6.1 Biconnected-Component

```

1  //UVALive - 3523
2  #include<bits/stdc++.h>
3  using namespace std;
4  #define clr(a,x) memset(a,x,sizeof(a))
5  #define rep(i,a,b) for(int i=a;i<=b;i++)
6  #define mp make_pair
7  #define fi first
8  #define se second
9  #define pb push_back
10 typedef pair<int,int> pii;
11 typedef vector<int> vi;
12 const int MAXV=1e3+10;
13 const int MAXE=1e6+10;
14 int tot;
15 int head[MAXV];
16 struct Edge{
17     int v,nxt;
18     Edge(){}
19     Edge(int v,int nxt):v(v),nxt(nxt){}
20 }edge[MAXE<<1];

```

```

21 void init(){
22     tot=0;
23     clr(head,-1);
24 }
25 void add_edge(int u,int v){
26     edge[tot]=Edge(v,head[u]);
27     head[u]=tot++;
28 }
29 int pre[MAXV],is_cut[MAXV],bccno[MAXV],dfs_clock,bcc_cnt;
30 vi bcc[MAXV];
31 stack<pii> st;
32 int dfs(int u,int fa){
33     int lowu=pre[u]=++dfs_clock;
34     int child=0;
35     for(int i=head[u];~i;i=edge[i].nxt){
36         int v=edge[i].v;
37         pii e=mp(u,v);
38         if(!pre[v]){
39             st.push(e);
40             child++;
41             int lowv=dfs(v,u);
42             lowu=min(lowu,lowv);
43             if(lowv>=pre[u]){
44                 is_cut[u]=1;
45                 bcc_cnt++;
46                 bcc[bcc_cnt].clear();
47                 for(;;){
48                     pii x=st.top();
49                     st.pop();
50                     if(bccno[x.fi]!=bcc_cnt){
51                         bcc[bcc_cnt].pb(x.fi);
52                         bccno[x.fi]=bcc_cnt;
53                     }
54                     if(bccno[x.se]!=bcc_cnt){
55                         bcc[bcc_cnt].pb(x.se);
56                         bccno[x.se]=bcc_cnt;
57                     }
58                     if(x.fi==u&&v==x.se)break;
59                 }
60             }
61         }else if(pre[v]<pre[u]&&v!=fa){
62             st.push(e);
63             lowu=min(lowu,pre[v]);
64         }
65     }
66     if(fa<0&&child==1)is_cut[u]=0;
67     return lowu;
68 }
69 void find_bcc(int n){
70     clr(pre,0);
71     clr(is_cut,0);
72     clr(bccno,0);
73     dfs_clock=bcc_cnt=0;
74     rep(i,1,n){
75         if(!pre[i])dfs(i,-1);
76     }
77 }
78 int odd[MAXV],color[MAXV];
79 bool bipartite(int u,int b){

```

```

80     for(int i=head[u];~i;i=edge[i].nxt){
81         int v=edge[i].v;
82         if(bccno[v]!=b)continue;
83         if(color[v]==color[u])return false;
84         if(!color[v]){
85             color[v]=3-color[u];
86             if(!bipartite(v,b))return false;
87         }
88     }
89     return true;
90 }
91 bool mmp[MAXV][MAXV];
92 int main(){
93     int n,m;
94     while(scanf("%d%d",&n,&m),n+m){
95         clr(mmp,0);
96         rep(i,1,m){
97             int x,y;
98             scanf("%d%d",&x,&y);
99             mmp[x][y]=1;
100            mmp[y][x]=1;
101        }
102        init();
103        rep(i,1,n){
104            rep(j,i+1,n){
105                if(!mmp[i][j]){
106                    add_edge(i,j);
107                    add_edge(j,i);
108                }
109            }
110        }
111        find_bcc(n);
112        clr(odd,0);
113        for(int i=1;i<=bcc_cnt;i++){
114            clr(color,0);
115            for(int j=0;j<bcc[i].size();j++){
116                bccno[bcc[i][j]]=i;
117            }
118            int u=bcc[i][0];
119            color[u]=1;
120            if(!bipartite(u,i)){
121                for(int j=0;j<bcc[i].size();j++){
122                    odd[bcc[i][j]]=1;
123                }
124            }
125        }
126        int ans=n;
127        rep(i,1,n)if(odd[i])ans--;
128        printf("%d\n",ans);
129    }
130    return 0;
131 }

```

4.6.2 Strongly Connected Component

```

1  const int MAXV=1e4+10;
2  const int MAXE=1e5+10;
3  int tot,head[MAXV];

```

```

4  int low[MAXV],dfn[MAXV],stk[MAXV],Belong[MAXV];
5  int idx,top,scc;
6  bool instk[MAXV];
7  struct Edge{
8      int v,nxt;
9      Edge(){}
10     Edge(int v,int nxt):v(v),nxt(nxt){}
11 }edge[MAXE];
12 void init(){
13     tot=0;
14     clr(head,-1);
15 }
16 void add_edge(int u,int v){
17     edge[tot]=Edge(v,head[u]);
18     head[u]=tot++;
19 }
20 void Tarjan(int u){
21     int v;
22     low[u]=dfn[u]=++idx;
23     stk[top++]=u;
24     instk[u]=true;
25     for(int i=head[u];~i;i=edge[i].nxt){
26         v=edge[i].v;
27         if(!dfn[v]){
28             Tarjan(v);
29             if(low[u]>low[v])low[u]=low[v];
30         }else if(instk[v]&&low[u]>dfn[v])low[u]=dfn[v];
31     }
32     if(low[u]==dfn[u]){
33         scc++;
34         do{
35             v=stk[--top];
36             instk[v]=false;
37             Belong[v]=scc;
38         }while(v!=u);
39     }
40 }
41 void tscc(int N){
42     clr(dfn,0);
43     clr(instk,0);
44     idx=scc=top=0;
45     rep(i,1,N)if(!dfn[i])Tarjan(i);
46 }

```

4.6.3 Kosaraju

```

1  const int MAXV=2e4+10;
2  const int MAXE=5e4+10;
3  int tot,scc,head[MAXV],rhead[MAXV],Belong[MAXV];
4  bool vis[MAXV];
5  int stk[MAXV],top;
6  struct Edge{
7      int v,nxt;
8      Edge(){}
9      Edge(int v,int nxt):v(v),nxt(nxt){}
10 }edge[MAXE],redge[MAXE];
11 void init(){
12     tot=0;

```

```

13     clr(head,-1);
14     clr(rhead,-1);
15 }
16 void add_edge(int u,int v){
17     edge[tot]=Edge(v,head[u]);
18     redge[tot]=Edge(u,rhead[v]);
19     head[u]=rhead[v]=tot++;
20 }
21 void dfs(int u){
22     vis[u]=true;
23     for(int i=head[u];~i;i=edge[i].nxt){
24         int v=edge[i].v;
25         if(!vis[v])dfs(v);
26     }
27     stk[++top]=u;
28 }
29 void rdfs(int u,int k){
30     vis[u]=true;
31     Belong[u]=k;
32     for(int i=rhead[u];~i;i=redge[i].nxt){
33         int v=redge[i].v;
34         if(!vis[v])rdfs(v,k);
35     }
36 }
37 void kscv(int V){
38     scc=top=0;
39     clr(vis,0);
40     rep(i,1,V)if(!vis[i])dfs(i);
41     clr(vis,0);
42     per(i,top,1){
43         int v=stk[i];
44         if(!vis[v])rdfs(v,++scc);
45     }
46 }

```

4.6.4 TwoSAT

```

1  //poj3683
2  //0 base !
3  //if (x V (!y))then add_clause(1,x,0,y)
4  //if x then add_var(1,x)
5  const int MAXV=1e5;
6  const int MAXE=3e6+5;
7  int tot,scc,head[MAXV],rhead[MAXV],Belong[MAXV];
8  bool vis[MAXV];
9  int stk[MAXV],top;
10 struct Edge{
11     int v,nxt;
12     Edge(){}
13     Edge(int v,int nxt):v(v),nxt(nxt){}
14 }edge[MAXE],redge[MAXE];
15 void init(){
16     tot=0;
17     clr(head,-1);
18     clr(rhead,-1);
19 }
20 void add_edge(int u,int v){
21     edge[tot]=Edge(v,head[u]);

```

```

22     redge[tot]=Edge(u,rhead[v]);
23     head[u]=rhead[v]=tot++;
24 }
25 void dfs(int u){
26     vis[u]=true;
27     for(int i=head[u];~i;i=edge[i].nxt){
28         int v=edge[i].v;
29         if(!vis[v])dfs(v);
30     }
31     stk[++top]=u;
32 }
33 void rdfs(int u,int k){
34     vis[u]=true;
35     Belong[u]=k;
36     for(int i=rhead[u];~i;i=redge[i].nxt){
37         int v=redge[i].v;
38         if(!vis[v])rdfs(v,k);
39     }
40 }
41 void kscv(int V){
42     scc=top=0;
43     clr(vis,0);
44     rep(i,0,V-1)if(!vis[i])dfs(i);
45     clr(vis,0);
46     per(i,top,1){
47         int v=stk[i];
48         if(!vis[v])rdfs(v,++scc);
49     }
50 }
51 void add_clause(int xv,int x,int yv,int y){
52     x=x<<1|xv;
53     y=y<<1|yv;
54     add_edge(x^1,y);
55     add_edge(y^1,x);
56 }
57 void add_var(int xv,int x){
58     x=x<<1|xv;
59     add_edge(x^1,x);
60 }
61 int st[MAXV],ed[MAXV],d[MAXV];
62 char tm[10];
63 int fun(){
64     int res=0;
65     int h=(tm[0]-'0')*10+tm[1]-'0';
66     res=h*60;
67     res+=(tm[3]-'0')*10+tm[4]-'0';
68     return res;
69 }
70 int work(){
71     int n;
72     scanf("%d",&n);
73     rep(i,0,n-1){
74         scanf("%s",tm);
75         st[i]=fun();
76         scanf("%s",tm);
77         ed[i]=fun();
78         scanf("%d",&d[i]);
79     }
80     init();

```



```

81     rep(i,0,n-1){
82         rep(j,0,i-1){
83             if(min(st[i]+d[i],st[j]+d[j])>max(st[i],st[j])){
84                 add_clause(0,i,0,j);
85             }
86             if(min(st[i]+d[i],ed[j])>max(st[i],ed[j]-d[j])){
87                 add_clause(0,i,1,j);
88             }
89             if(min(ed[i],st[j]+d[j])>max(ed[i]-d[i],st[j])){
90                 add_clause(1,i,0,j);
91             }
92             if(min(ed[i],ed[j])>max(ed[i]-d[i],ed[j]-d[j])){
93                 add_clause(1,i,1,j);
94             }
95         }
96     }
97     ksc(2*n);
98     rep(i,0,n-1){
99         if(Belong[i<<1]==Belong[i<<1|1]){
100             puts("NO");
101             return 0;
102         }
103     }
104     puts("YES");
105     rep(i,0,n-1){
106         if(Belong[i<<1|1]>Belong[i<<1]){
107             printf("%02d:%02d %02d:%02d\n",st[i]/60,st[i]%60,(st[i]+d[i])/60,(st[i]+d[i]
108 ])%60);
109         }else{
110             printf("%02d:%02d %02d:%02d\n", (ed[i]-d[i])/60,(ed[i]-d[i])%60,ed[i]/60,ed[
111 i]%60);
112         }
113     }
114     return 0;
115 }

```

4.6.5 cut-vertex

```

1  //poj 1144
2  #include<cstdio>
3  #include<cstring>
4  #include<algorithm>
5  using namespace std;
6  #define rep(i,a,b) for(int i=a;i<=b;i++)
7  #define clr(a,x) memset(a,x,sizeof(a))
8  const int MAXV=105;
9  const int MAXE=1e5;
10 int tot;
11 int head[MAXV];
12 struct Edge{
13     int v,nxt;
14     Edge(){ }
15     Edge(int v,int nxt):v(v),nxt(nxt){ }
16 }edge[MAXE<<1];
17 void init(){
18     tot=0;
19     clr(head,-1);
20 }

```

```

21 void add_edge(int u,int v){
22     edge[tot]=Edge(v,head[u]);
23     head[u]=tot++;
24 }
25 int n;
26 bool is_cut[MAXV];
27 int low[MAXV],pre[MAXV];
28 int dfs_clock;
29 int dfs(int u,int fa){
30     int lowu=pre[u]=++dfs_clock;
31     int child=0;
32     for(int i=head[u];~i;i=edge[i].nxt){
33         int v=edge[i].v;
34         if(!pre[v]){
35             child++;
36             int lowv=dfs(v,u);
37             lowu=min(lowu,lowv);
38             if(lowv>=pre[u]){
39                 is_cut[u]=true;
40             }
41         }else if(pre[v]<pre[u]&&v!=fa){
42             lowu=min(lowu,pre[v]);
43         }
44     }
45     if(fa<0&&child==1)is_cut[u]=false;
46     low[u]=lowu;
47     return lowu;
48 }
49 int main(){
50     while(scanf("%d",&n),n){
51         init();
52         int x;
53         while(scanf("%d",&x),x){
54             int y;
55             while(getchar()!='\n'){
56                 scanf("%d",&y);
57                 add_edge(x,y);
58                 add_edge(y,x);
59             }
60         }
61         clr(is_cut,0);
62         clr(low,0);
63         clr(pre,0);
64         dfs_clock=0;
65         int cnt=0;
66         dfs(1,-1);
67         for(int i=1;i<=n;i++){
68             if(is_cut[i])cnt++;
69         }
70         printf("%d\n",cnt);
71     }
72     return 0;
73 }

```

4.7 Bipartite Graph Matching

4.7.1 Hungry

```
1 //poj3041
```

```

2  const int MAXV=1e3+5;
3  struct BM{
4      int V;
5      vi G[MAXV];
6      int match[MAXV];
7      bool vis[MAXV];
8      void init(int x){
9          V=x;
10         rep(i,1,V)G[i].clear();
11     }
12     void add_edge(int u,int v){
13         G[u].pb(v);
14         G[v].pb(u);
15     }
16     bool dfs(int u){
17         vis[u]=true;
18         for(int i=0;i<(int)G[u].size();i++){
19             int v=G[u][i];
20             int w=match[v];
21             if(w==-1||(!vis[w]&&dfs(w))){
22                 match[u]=v;
23                 match[v]=u;
24                 return true;
25             }
26         }
27         return false;
28     }
29     int matching(){
30         int ret=0;
31         clr(match,-1);
32         rep(i,1,V){
33             if(match[i]==-1){
34                 clr(vis,0);
35                 if(dfs(i))ret++;
36             }
37         }
38         return ret;
39     }
40 }bm;
41 int work(){
42     int n,k;
43     scanf("%d%d",&n,&k);
44     bm.init(2*n);
45     while(k--){
46         int u,v;
47         scanf("%d%d",&u,&v);
48         bm.add_edge(u,n+v);
49     }
50     printf("%d",bm.matching());
51     return 0;
52 }

```

4.8 Network Flow

4.8.1 Dinic

```

1  //poj 3281
2  #include<cstdio>
3  #include<iostream>

```

```

4  #include<algorithm>
5  #include<cstring>
6  #include<queue>
7  using namespace std;
8  #define clr(a,x) memset(a,x,sizeof(a))
9  const int MAXV=400+5;
10 const int MAXE=1e5+5;
11 const int INF=0x3f3f3f3f;
12 int tot;
13 int head[MAXV],level[MAXV],iter[MAXV];
14 struct Edge{
15     int v,cap,nxt;
16     Edge(){}
17     Edge(int v,int cap,int nxt):v(v),cap(cap),nxt(nxt){}
18 }edge[MAXE<<1];
19 void init(){
20     tot=0;
21     clr(head,-1);
22 }
23 void add_edge(int u,int v,int c){
24     edge[tot]=Edge(v,c,head[u]);
25     head[u]=tot++;
26     edge[tot]=Edge(u,0,head[v]);
27     head[v]=tot++;
28 }
29 void bfs(int s){
30     clr(level,-1);
31     level[s]=0;
32     queue<int> que;
33     que.push(s);
34     while(!que.empty()){
35         int u=que.front();
36         que.pop();
37         for(int i=head[u];~i;i=edge[i].nxt){
38             int v=edge[i].v;
39             int c=edge[i].cap;
40             if(c>0&&level[v]<0){
41                 level[v]=level[u]+1;
42                 que.push(v);
43             }
44         }
45     }
46 }
47 int dfs(int u,int t,int f){
48     if(u==t)return f;
49     for(int &i=iter[u];~i;i=edge[i].nxt){
50         int v=edge[i].v;
51         int c=edge[i].cap;
52         if(c>0&&level[u]<level[v]){
53             int d=dfs(v,t,min(f,c));
54             if(d>0){
55                 edge[i].cap-=d;
56                 edge[i^1].cap+=d;
57                 return d;
58             }
59         }
60     }
61     return 0;
62 }

```

```

63 int max_flow(int s,int t){
64     int flow=0;
65     while(1){
66         bfs(s);
67         if(level[t]<0)return flow;
68         int f;
69         memcpy(iter,head,sizeof(head));
70         while(f=dfs(s,t,INF))flow+=f;
71     }
72 }
73 int main(){
74     int n,f,d;
75     scanf("%d%d%d",&n,&f,&d);
76     int s=0,t=2*n+f+d;
77     init();
78     for(int i=1;i<=f;i++){
79         add_edge(s,2*n+i,1);
80     }
81     for(int i=1;i<=d;i++){
82         add_edge(2*n+f+i,t,1);
83     }
84     for(int i=1;i<=n;i++){
85         add_edge(i,n+i,1);
86         int ff,dd;
87         scanf("%d%d",&ff,&dd);
88         while(ff--){
89             int x;
90             scanf("%d",&x);
91             add_edge(2*n+x,i,1);
92         }
93         while(dd--){
94             int x;
95             scanf("%d",&x);
96             add_edge(n+i,2*n+f+x,1);
97         }
98     }
99     printf("%d",max_flow(s,t));
100     return 0;
101 }

```

4.8.2 MinCost MaxFlow

```

1 // poj2135
2 #include<cstdio>
3 #include<vector>
4 #include<algorithm>
5 #include<queue>
6 using namespace std;
7 const int MAXV=1005;
8 const int MAXE=50000;
9 const int INF=100000000;
10 typedef pair<int,int> P;
11 struct edge{int to,cap,cost,rev;};
12 int dist[MAXV],h[MAXV],prevv[MAXV],preve[MAXV];
13 int V;
14 vector<edge> G[MAXV];
15 void add_edge(int from,int to,int cap,int cost){
16     G[from].push_back((edge){to,cap,cost,G[to].size()});

```

```

17     G[to].push_back((edge){from,0,-cost,G[from].size()-1});
18 }
19 int min_cost_flow(int s,int t,int f){
20     int res=0;
21     fill(h,h+V,0);
22     while(f>0){
23         priority_queue<P,vector<P>,greater<P> >que;
24         fill(dist,dist+V,INF);
25         dist[s]=0;
26         que.push(P(0,s));
27         while(!que.empty()){
28             P p=que.top(); que.pop();
29             int v=p.second;
30             if(dist[v]<p.first) continue;
31             for(int i=0;i<G[v].size();i++){
32                 edge &e=G[v][i];
33                 if(e.cap>0&&dist[e.to]>dist[v]+e.cost+h[v]-h[e.to]){
34                     dist[e.to]=dist[v]+e.cost+h[v]-h[e.to];
35                     prevv[e.to]=v;
36                     preve[e.to]=i;
37                     que.push(P(dist[e.to],e.to));
38                 }
39             }
40         }
41         if(dist[t]==INF){
42             return -1;
43         }
44         for(int v=0;v<V;v++) h[v]+=dist[v];
45         int d=f;
46         for(int v=t;v!=s;v=prevv[v]){
47             d=min(d,G[prevv[v]][preve[v]].cap);
48         }
49         f-=d;
50         res+=d*h[t];
51         for(int v=t;v!=s;v=prevv[v]){
52             edge &e=G[prevv[v]][preve[v]];
53             e.cap-=d;
54             G[v][e.rev].cap+=d;
55         }
56     }
57     return res;
58 }
59 int main(){
60     int N,M;
61     scanf("%d%d",&N,&M);
62     V=N;
63     for(int i=1;i<=M;i++){
64         int x,y,z;
65         scanf("%d%d%d",&x,&y,&z);
66         add_edge(x-1,y-1,1,z);
67         add_edge(y-1,x-1,1,z);
68     }
69     printf("%d",min_cost_flow(0,N-1,2));
70     return 0;
71 }

```

5 Others

5.1 Matrix

5.1.1 Matrix FastPow

```

1  typedef vector<ll> vec;
2  typedef vector<vec> mat;
3  mat mul(mat& A, mat& B)
4  {
5      mat C(A.size(), vec(B[0].size()));
6      for (int i = 0; i < A.size(); i++)
7          for (int k = 0; k < B.size(); k++)
8              if (A[i][k]) // 00000000
9                  for (int j = 0; j < B[0].size(); j++)
10                     C[i][j] = (C[i][j] + A[i][k] * B[k][j]) % mod;
11     return C;
12 }
13 mat Pow(mat A, ll n)
14 {
15     mat B(A.size(), vec(A.size()));
16     for (int i = 0; i < A.size(); i++) B[i][i] = 1;
17     for (; n >= 1; A = mul(A, A))
18         if (n & 1) B = mul(B, A);
19     return B;
20 }
```

5.2 Tricks

5.2.1 Stack-Overflow

```

1  #pragma comment(linker, "/STACK:1024000000,1024000000")
```

5.2.2 Fast-Scanner

```

1  template <class T>
2  inline bool scan_d(T &ret){
3      char c;
4      int sgn;
5      if (c = getchar(), c == EOF) return 0; //EOF
6      while (c != '-' && (c < '0' || c > '9')) c = getchar();
7      sgn = (c == '-') ? -1 : 1;
8      ret = (c == '-') ? 0 : (c - '0');
9      while (c = getchar(), c >= '0' && c <= '9') ret = ret * 10 + (c - '0');
10     ret *= sgn;
11     return 1;
12 }
13 inline void out(int x){
14     if(x<0){
15         putchar('-');
16         x=-x;
17     }
18     if (x > 9) out(x / 10);
19     putchar(x % 10 + '0');
20 }
```

5.2.3 Strtok-Scanf

```

1 // get some integers in a line
2 gets(buf);
3 int v;
4 char *p = strtok(buf, " ");
5 while (p){
6     sscanf(p, "%d", &v);
7     p = strtok(NULL, " ");
8 }

```

5.3 Mo Algorithm

```

1 //hdu 6333
2 #include<bits/stdc++.h>
3 using namespace std;
4 typedef long long ll;
5 const int MAXN=1e5+10;
6 const int MOD=1e9+7;
7 int block;
8 struct node{
9     int l,r,id;
10 }no[MAXN];
11 bool cmp(node x,node y){
12     if(x.l/block==y.l/block)return x.r<y.r;
13     else return x.l/block<y.l/block;
14 }
15 int ans[MAXN];
16 int fact[MAXN];
17 int invfact[MAXN];
18 ll pow_mod(ll a,ll b){
19     ll res=1;
20     while(b){
21         if(b&1)res=res*a%MOD;
22         a=a*a%MOD;
23         b>>=1;
24     }
25     return res;
26 }
27 ll fun(ll n,ll m){
28     return (1LL*fact[n]*invfact[m])%MOD*invfact[n-m]%MOD;
29 }
30 int main(){
31     int n=100000;
32     fact[0]=1;
33     for(int i=1;i<=n;i++){
34         fact[i]=1LL*fact[i-1]*i%MOD;
35     }
36     invfact[n]=pow_mod(fact[n],MOD-2);
37     for(int i=n;i>=1;i--){
38         invfact[i-1]=1LL*invfact[i]*i%MOD;
39     }
40     int q;
41     scanf("%d",&q);
42     block=(int)sqrt(100000);
43     for(int i=1;i<=q;i++){
44         scanf("%d%d",&no[i].r,&no[i].l);
45         no[i].id=i;

```



```

46     }
47     sort(no+1,no+1+q,cmp);
48     int L=1,R=1;
49     ll now=2;
50     int inv2=pow_mod(2,MOD-2);
51     for(int i=1;i<=q;i++){
52         while(R<no[i].r){
53             now=(now*2-fun(R,L)+MOD)%MOD;
54             R++;
55         }
56         while(L>no[i].l){
57             now=(now-fun(R,L)+MOD)%MOD;
58             L--;
59         }
60         while(R>no[i].r){
61             R--;
62             now+=fun(R,L);
63             now%=MOD;
64             now=now*inv2%MOD;
65         }
66         while(L<no[i].l){
67             L++;
68             now=(now+fun(R,L))%MOD;
69         }
70         ans[no[i].id]=now;
71     }
72     for(int i=1;i<=q;i++){
73         printf("%d\n",ans[i]);
74     }
75     return 0;
76 }

```

5.4 BigNum

5.4.1 High-precision

```

1  import java.io.*;
2  import java.math.*;
3  import java.util.StringTokenizer;
4
5  public class Main{
6      public static void main(String[] args){
7          InputStream inputStream = System.in;//new FileInputStream("C:\\Users\\xxx\\
Downloads\\test.in");
8          OutputStream outputStream = System.out;
9          InputReader in = new InputReader(inputStream);
10         PrintWriter out = new PrintWriter(outputStream);
11         Task solver = new Task();
12         solver.solve(in, out);
13         out.close();
14     }
15     static class Task {
16
17         public void solve(InputReader in, PrintWriter out) {
18             //do sth
19
20         }
21
22     }

```

```
23 static class InputReader {
24     public BufferedReader reader;
25     public StringTokenizer tokenizer;
26
27     public InputReader(InputStream stream) {
28         reader = new BufferedReader(new InputStreamReader(stream), 32768);
29         tokenizer = null;
30     }
31
32     public String next() {
33         while (tokenizer == null || !tokenizer.hasMoreTokens()) {
34             try {
35                 tokenizer = new StringTokenizer(reader.readLine());
36             } catch (IOException e) {
37                 throw new RuntimeException(e);
38             }
39         }
40         return tokenizer.nextToken();
41     }
42
43     public int nextInt() {
44         return Integer.parseInt(next());
45     }
46
47     public long nextLong() {
48         return Long.parseLong(next());
49     }
50
51     public double nextDouble() {
52         return Double.parseDouble(next());
53     }
54
55     public char[] nextCharArray() {
56         return next().toCharArray();
57     }
58
59     public boolean hasNext() {
60         try {
61             String string = reader.readLine();
62             if (string == null) {
63                 return false;
64             }
65             tokenizer = new StringTokenizer(string);
66             return tokenizer.hasMoreTokens();
67         } catch (IOException e) {
68             return false;
69         }
70     }
71     public BigInteger nextBigInteger() {
72         return new BigInteger(next());
73     }
74
75     public BigDecimal nextBigDecimal() {
76         return new BigDecimal(next());
77     }
78 }
79 }
```

5.5 VIM

```

1 syntax on
2 set nu
3 set tabstop=4
4 set expandtab
5 set autoindent
6 set cin
7 set mouse=a
8
9 map<F2> :call SetTitle(<CR>
10 func SetTitle()
11 let l = 0
12 let l = l + 1 | call setline(l, '#include <algorithm>')
13 let l = l + 1 | call setline(l, '#include <iostream>')
14 let l = l + 1 | call setline(l, '#include <cstring>')
15 let l = l + 1 | call setline(l, '#include <string>')
16 let l = l + 1 | call setline(l, '#include <cstdio>')
17 let l = l + 1 | call setline(l, '#include <vector>')
18 let l = l + 1 | call setline(l, '#include <stack>')
19 let l = l + 1 | call setline(l, '#include <queue>')
20 let l = l + 1 | call setline(l, '#include <cmath>')
21 let l = l + 1 | call setline(l, '#include <set>')
22 let l = l + 1 | call setline(l, '#include <map>')
23 let l = l + 1 | call setline(l, 'using namespace std;')
24 let l = l + 1 | call setline(l, '#define rep(i,a,b) for(int i=a;i<=b;i++)')
25 let l = l + 1 | call setline(l, '#define per(i,a,b) for(int i=a;i>=b;i--)')
26 let l = l + 1 | call setline(l, '#define clr(a,x) memset(a,x,sizeof(a))')
27 let l = l + 1 | call setline(l, '#define pb push_back')
28 let l = l + 1 | call setline(l, '#define mp make_pair')
29 let l = l + 1 | call setline(l, '#define all(x) (x).begin(),(x).end()')
30 let l = l + 1 | call setline(l, '#define fi first')
31 let l = l + 1 | call setline(l, '#define se second')
32 let l = l + 1 | call setline(l, '#define SZ(x) ((int)(x).size())')
33 let l = l + 1 | call setline(l, 'typedef unsigned long long ull;')
34 let l = l + 1 | call setline(l, 'typedef long long ll;')
35 let l = l + 1 | call setline(l, 'typedef vector<int> vi;')
36 let l = l + 1 | call setline(l, 'typedef pair<int,int> pii;')
37 let l = l + 1 | call setline(l, '/******head*****')
38 let l = l + 1 | call setline(l, 'int work(){')
39 let l = l + 1 | call setline(l, '    ')
40 let l = l + 1 | call setline(l, '        return 0;')
41 let l = l + 1 | call setline(l, '    }')
42 let l = l + 1 | call setline(l, 'int main(){')
43 let l = l + 1 | call setline(l, '#ifdef superkunn')
44 let l = l + 1 | call setline(l, '    freopen("input.txt","rt",stdin);')
45 let l = l + 1 | call setline(l, '#endif')
46 let l = l + 1 | call setline(l, '    work();')
47 let l = l + 1 | call setline(l, '    return 0;')
48 let l = l + 1 | call setline(l, '    }')
49 endfunc

```

5.6 BASH

```

1 g++ -g -Wall -std=c++11 -Dsuperkunn main.cpp
2 ./a.out

```

6 Geometry

```

1 struct Point{
2     double x,y;
3     Point(double x=0,double y=0):x(x),y(y){}
4 };
5 typedef Point Vector;
6 Vector operator + (Vector A,Vector B){return Vector(A.x+B.x,A.y+B.y);}
7 Vector operator - (Point A,Point B){return Vector(A.x-B.x,A.y-B.y);}
8 Vector operator * (Vector A,double p){return Vector(A.x*p,A.y*p);}
9 Vector operator / (Vector A,double p){return Vector(A.x/p,A.y/p);}
10 bool operator < (const Point& a,const Point &b){
11     return a.x<b.x||(a.x==b.x&&a.y<b.y);
12 }
13 const double eps = 1e-10;
14 int dcmp(double x){
15     if(fabs(x)<eps)return 0;else return x<0?-1:1;
16 }
17 bool operator == (const Point& a,const Point &b){
18     return dcmp(a.x-b.x)==0&&dcmp(a.y-b.y)==0;
19 }
20 //(x,y)-> atan2(y,x)
21 double Dot(Vector A,Vector B){return A.x*B.x+A.y*B.y;}
22 double Length(Vector A){return sqrt(Dot(A,A));}
23 double Angle(Vector A,Vector B){return acos(Dot(A,B)/Length(A)/Length(B));}
24 double Cross(Vector A,Vector B){return A.x*B.y-A.y*B.x;}
25 double Area2(Point A,Point B,Point C){return Cross(B-A,C-A);}
26 Vector Rotate(Vector A,double rad){
27     return Vector(A.x*cos(rad)-A.y*sin(rad),A.x*sin(rad)+A.y*cos(rad));
28 }
29 Vector Normal(Vector A){
30     double L=Length(A);
31     return Vector(-A.y/L,A.x/L);
32 }

```