ICPC World Finals 2019 Templates

SYSU_Balloon

2019 年 3 月 17 日

目录

1 数学

1.1 结论和定理

五边形定理 五边形数 $n(3n\pm1)/2$ 。$(1-x)(1-x^2)(1-x^3)\cdots=\Sigma(-1)^k x^{n(3n\pm1)/2}$，即 $f(n)=f(n-1)+f(n-2)-f(n-5)-f(n-7)+f(n-12)+f(n-15)-\cdots$ 。

斐波那契数性质 ① $f[n]=f[n-1]+f[n-2]$; ② $f[n+m+1]=f[n]f[m]+f[n+1]f[m+1]$; ③ $\gcd(f[n],f[n+1])=1$; ④ $\gcd(f[n],f[n+2])=1$; ⑤ $\gcd(f[n],f[m])=f[\gcd(n,m)]$; ⑥ $f[n+1]^2-f[n]f[n+2]=(-1)^n$; ⑦ $\Sigma_{i=1}^n f[i]^2=f[n]f[n+1]$; ⑧ $\Sigma_{i=0}^n f[i]=f[n+2]-1$; ⑨ $\Sigma_{i=1}^n f[2i-1]=f[2n]$; ⑩ $\Sigma_{i=1}^n f[2i]=f[2n+1]-1$; ❶ $\Sigma_{i=0}^n(-1)^i f[i]=(-1)^n(f[n+1]-f[n])+1$; ❷ $f[2n-1]=f[n]^2-f[n-2]^2$; ❸ $f[2n+1]=f[n]^2+f[n+1]^2$; ❹ $3f[n]=f[n+2]+f[n-2]$; ❺ $f[n]=\Sigma_{i=0}^m\binom{n-1-i}{i}(m\le n-1-m)$; ❻ $\Sigma_{i=1}^n if[i]=nf[n+2]-f[n+3]+2$.

卡特兰数性质 ① 凸多边形三角剖分数; ② 简单有序根树的计数; ③ $(0,0)$ 走到 (n,n) 经过的点 (a,b) 满足 $a\le b$ 的方案数; ④ $h_1=1,h_n=\frac{h_{n-1}(4n-2)}{n+1}=\frac{C(2n,n)}{n+1}=C(2n,n)-C(2n,n-1)$; ⑤ 在一个格点阵列中，从 $(0,0)$ 点走到 (n,m) 点且不经过/穿过对角线 $x=y$ 的方案数: $\binom{n+m-1}{m}-\binom{n+m-1}{m-1}(x>y)$; $\binom{n+m}{m}-\binom{n+m}{m-1}(x\ge y)$

第一类斯特林数性质 ① 有正有负，其绝对值是 n 个元素的项目分作 k 个非空循环排列的数量 $s[n][k]$; ② $s[n][0]=0(n\ge1),s[n][n]=1(n\ge0)$; ③ $s[n][k]=(n-1)s[n-1][k]+s[n-1][k-1](1\le k\le n-1)$; ④ $|s[n][1]|=(n-1)!$; ⑤ $s[n][k]=(-1)^{n+k}|s[n][k]|$; ⑥ $s[n][n-1]=-\binom{n}{2}$; ⑦ $x(x-1)(x-2)\cdots(x-n+1)=\Sigma s[n][k]x^k$.

第二类斯特林数性质 ① 将 n 个物体划分为 k 个非空的不可辨别（可理解为盒子没有编号）集合的方法数; ② $s[n][0]=0(n\ge1),s[n][n]=1(n\le0)$; ③ $s[n][k]=ks[n-1][k]+s[n-1][k-1]$; ④ $s[n][n-1]=\binom{n}{2}$ ⑤ $s[n][2]=2^{n-1}-1$ ⑥ $s[n][k]=\frac{1}{k!}\Sigma_{i=0}^k(-1)^i\binom{k}{i}(k-i)^n$.

Bell 数性质 ① n 个元素的集合划分数目; ② $B[n]=\Sigma_{k=1}^n s[n][k]$; ③ $B[n+1]=\Sigma_{k=0}^n\binom{n}{k}B[k]$; ④ $B[n+p]=B[n]+B[n+1](\bmod\ p)$.

多项式性质 ① $f(x)$ 不存在重根 $\Leftrightarrow\gcd(f(x),f'(x))$ 的次数小于 1 次; ② 多项式 gcd 可以用来判断两多项式是否有公共根.

多项式取模 $f(x)=0(\bmod\ m_0),m_0=\Pi_{i=1}^k m_i$. 用 T_i 表示 $f(x)=0(\bmod\ m_i)$ 的解数，则 $T_0=\Pi_{i=1}^k T_i$.

数论 ① $a^n\bmod b=a^{n\bmod\ \phi(b)+\phi(b)}\bmod b(n\ge\phi(b))$; ② lucas 定理 $\binom{n}{m}=\binom{n\%p}{m\%p}\binom{n/p}{m/p}(\bmod\ p)$; ③ lucas 函数 $f(n,m)=f(n\%p,m\%p)f(n/p,m/p)(\bmod\ p)$，可以猜测满足.

蔡勒公式 $w=(\lfloor\frac{c}{4}\rfloor-2c+y+\lfloor\frac{y}{4}\rfloor+\lfloor\frac{13(m+1)}{5}\rfloor+d-1)\bmod 7$ 。① w：0 星期日，1 星期一，$\cdots$，6 星期六; ② c：年份前两位数; ③ y：年份后两位数; ④ m：月（$3\le m\le14$，即在蔡勒公式中，1、2 月要看作上一年的 13、14 月来计算）; ⑤ d：日。

皮克定理 给定顶点坐标均是整点（或正方形格点）的简单多边形（凸多边形），皮克定理说明了其面积 S 和内部格点数目 n 、边上格点数目 s 的关系: $S=n+\frac{s}{2}+1$ 。

1.2 Miller Rabin

```
 1   // 2,7,61 : < 4759123141
 2   // 2,3,5,7,11,13,17 : < 341550071728320
 3   // 2,3,7,61,24251 : < 10^16 only 46856248255981
 4
 5   //BZOJ-3667
 6   #include<cstdio>
 7   #include<cstdlib>
 8   typedef long long ll;
 9   ll _,n,x,ans,st;
10   ll gcd(ll x,ll y){return y==0?x:gcd(y,x%y);}
11   #define abs(x) (x>0?x:-(x))
12   #define cmax(a,b) (a<b?a=b:1)
13   ll mul(ll a,ll b,ll p){
14       ll tmp=(a*b-(ll)((long double)a/p*b+1e-7)*p);
15       return tmp<0?tmp+p:tmp;
16   }
17   ll power(ll t,ll k,ll p){
18       ll f=1;
19       for(;k;k>>=1,t=mul(t,t,p))if(k&1)f=mul(f,t,p);
20       return f;
21   }
22   bool check(ll a,int k,ll p,ll q){
23       ll t=power(a,q,p);
24       if(t==1||t==p-1)return 1;
25       for(;k--;){
26           t=mul(t,t,p);
```

```
27          if(t==p-1)return 1;
28      }
29      return 0;
30 }
31 bool mr(ll p){
32      if(p<=1)return 0;
33      if(p==2)return 1;
34      if(~p&1)return 0;
35      ll q=p-1;int i,k=0;
36      while(~q&1)q>>=1,k++;
37      for(i=0;i<5;i++)
38      if(!check(rand()%(p-1)+1,k,p,q))return 0;
39      return 1;
40 }
41 ll rho(ll n,ll c){
42      ll x=rand()%n,y=x,p=1;
43      while(p==1)
44          x=(mul(x,x,n)+c)%n,
45          y=(mul(y,y,n)+c)%n,
46          y=(mul(y,y,n)+c)%n,
47          p=gcd(n,abs(x-y));
48      return p;
49 }
50 void solve(ll n){
51      if(n==1)return;
52      if(mr(n)){cmax(ans,n);return;}
53      if(~n&1)cmax(ans,2),solve(n>>1);
54      else{
55          ll t=n;
56          while(t==n)t=rho(n,rand()%(n-1)+1);
57          solve(t),solve(n/t);
58      }
59 }
60 int main(){
61      for(srand(1626),scanf("%lld",&_);_--;){
62          scanf("%lld",&x),ans=0;solve(x);
63          if(ans==x)puts("Prime");
64          else printf("%lld\n",ans);
65      }
66 }
```

1.3 同余方程

```
1 void gcd(LL a, LL b, LL &d, LL &x, LL &y){
2      if (!b){ d=a; x=1; y=0; return; }
```

```
3      gcd(b,a%b,d,y,x); y-=x*(a/b);
4 }
5 IL void sim(LL &a, LL n) { a%=n; if (a<0) a+=n; }
6 IL LL solve(LL a, LL b, LL n){   // a*x==b (mod n)
7      sim(a,n); sim(b,n);   // optional
8      static LL d,x,y;
9      gcd(a,n,d,x,y);
10     if (b%d) return -1;
11     b/=d; n/=d;
12     if (x<0) x+=n;
13     return b*x%n;
14 }
15 // x==a1 (mod n1); x==a2 (mod n2);
16 // passing gcd in solve can reduce time
17 void merge(LL a1, LL n1, LL a2, LL n2, LL &x, LL &n){
18     n=lcm(n1,n2);
19     LL k=solve(n1,a2-a1,n2);
20     if (k==-1) { x=-1; return; }
21     sim(x=n1*k+a1,n);
22 }
23 // getinv , gcd(a,n) must be 1
24 IL LL getinv(LL a, LL n){
25     static LL d,x,y;
26     gcd(a,n,d,x,y);
27     // if (d!=1) return -1;
28     return x<0?x+n:x;
29 }
```

1.4 线性筛法

```
1 const int N=100050;
2 int b[N],a[N],cnt,mx[N],phi[N],mu[N];
3
4 void getprime(int n=100000){
5      memset(b+2,1,sizeof(b[0])*(n-1));
6      mu[1]=1;
7      ft(i,2,n){
8          if (b[i]){
9              a[mx[i]=++cnt]=i;
10             phi[i]=i-1; mu[i]=-1;
11         }
12         ft(j,1,mx[i]){
13             int k=i*a[j];
14             if (k>n) break;
15             b[k]=0; mx[k]=j;
```

```
16              phi[k]= phi[i]*(a[j]-(j!=mx[i]));
17              mu[k]= j==mx[i] ? 0 : -mu[i];
18          }
19      }
20  }
```

1.5 离散对数

```
1   // BSGS , a^x==b (mod n) , n is a prime
2   LL bsgs(LL a, LL b, LL n){
3       int m=sqrt(n+0.5);
4       LL p=power(a,m,n);
5       LL v=getinv(p,n);
6       static hash_map x;
7       x.clear();
8       LL e=1; x[e]=0;
9       ft(i,1,m){
10          e=e*a%n;
11          if (!x.count(e)) x[e]=i;
12      }
13      for(LL i=0;i<n;i+=m){
14          if (x.count(b)) return i+x[b];
15          b=b*v%n;
16      }
17      return -1;
18  }
19
20  //BSGS
21  //y^x==z (mod p) ->x=?
22  scanf("%d%d%d",&y,&z,&p),y%=p,z%=p;j=z;
23  if(y==0){puts("Cannot_find_x");continue;}
24  for(k=s=1;k*k<=p;k++);
25  std::map<int,int>hash;flag=0;
26  for(int i=0;i<k;i++,s=1LL*s*y%p,j=1LL*j*y%p)hash[j]=i;
27  for(int i=1,j=s;i<=k&&!flag;i++,j=1LL*j*s%p)
28  if(hash.count(j))ans=i*k-hash[j],flag=1;
29  if(flag==0)puts("Cannot_find_x");
30  else printf("%d\n",ans);
31
32  //exBSGS
33  int bsgs(int a,ll b,int p){
34      if(a%=p,b%=p,b==1)return 0;
35      ll t=1;int f,g,delta=0,m=sqrt(p)+1,i;
36      for(g=gcd(a,p);g!=1;g=gcd(a,p)){
37          if(b%g)return -1;
```

```
38          b/=g,p/=g,t=t*(a/g)%p,delta++;
39          if(b==t)return delta;
40      }
41      std::map<int,int>hash;
42      for(i=0;i<m;i++,b=b*a%p)hash[b]=i;
43      for(i=1,f=power(a,m);i<=m+1;i++)
44      if(t=t*f%p,hash.count(t))return i*m-hash[t]+delta;
45      return -1;
46  }
```

1.6 Lucas

```
1   void init_Lucas(){
2       fac[0]=1; ft(i,1,P-1) fac[i]=fac[i-1]*i%P;
3       inv[1]=1; ft(i,2,P-1) inv[i]=(P-P/i)*inv[P%i]%P;
4       inv[0]=1; ft(i,1,P-1) inv[i]=inv[i-1]*inv[i]%P;
5   }
6   IL LL C(int n, int m){
7       LL ans=1;
8       while (n||m){
9           int a=n%P, b=m%P;
10          if (a<b) return 0;
11          n/=P; m/=P;
12          ans= ans *fac[a]%P *inv[b]%P *inv[a-b]%P;
13      }
14      return ans;
15  }
```

1.7 高斯消元法实数方程

```
1   void Gauss(int n,int m) {
2       int i,j,k,t;
3       double mul;
4       for (i=j=1;i<=n&&j<=m;i++,j++) {
5           for (k=i+1;k<=n;k++)
6               if (abs(mat[k][j])>abs(mat[i][j]))
7                   for (t=1;t<=m+1;t++) swap(mat[i][t],mat[k][t]);
8           if (abs(mat[i][j])<eps) { i--; continue; }
9           for (k=i+1;k<=n;k++) {
10              mul=mat[k][j]/mat[i][j];
11              for (t=1;t<=m+1;t++) mat[k][t]-=mat[i][t]*mul;
12          }
13      }
14      for (i=n;i>=1;i--) { //solved表示那个变量是否确定
```

```
15        for (j=1;j<=m;j++) if (abs(mat[i][j])>eps) break;
16        if  (j>m) continue; solved[j]=true; ans[j]=mat[i][m+1];
17        for (k=j+1;k<=m;k++)
18            if (abs(mat[i][k])>eps&&!solved[k]) solved[j]=false;
19        for (k=j+1;k<=m;k++) ans[j]-=ans[k]*mat[i][k];
20        ans[j]/=mat[i][j];
21    }
22 }
```

1.8   高斯消元解异或方程

```
1  int n,m;
2  bitset<N> a[N];
3  bool solve(){
4      int i=1, j=1;
5      while (i<=n && j<=m){
6          int k=i;
7          while (k<=n && !a[k][j]) k++;
8          if (k>n) { j++; continue; }
9          if (j==m) return false;  // no solution
10         if (k!=i) swap(a[i],a[k]);
11         ft(t,1,n) if (t!=i && a[t][j]) a[t]^=a[i];
12         i++; j++;
13     }
14     return true; // have solution (but may have 0==0)
15 }
```

1.9   高斯消元法模方程

```
1  void Gauss(LL n,LL m) {
2      LL i,j,k,t,lcm,muli,mulk;
3      for (i=j=1;i<=n&&j<=m;i++,j++) {
4          for (k=i;k<=n;k++) if (mat[k][j]) {
5              for (t=1;t<=m+1;t++) swap(mat[k][t],mat[i][t]);
6              break;
7          }
8          if (mat[i][j]==0) { i--; continue; }
9          for (k=i+1;k<=n;k++) if (mat[k][j]) {
10             lcm=mat[k][j]*mat[i][j]/__gcd(mat[k][j],mat[i][j]);
11             muli=lcm/mat[i][j]; mulk=lcm/mat[k][j];
12             for (t=1;t<=m+1;t++) {
13                 mat[k][t]=mat[k][t]*mulk-mat[i][t]*muli;
14                 mat[k][t]=(mat[k][t]%mod+mod)%mod;
15             }
```

```
16         }
17     }
18     for (i=n;i>=1;i--) {
19         for (j=1;j<=m;j++) if (mat[i][j]) break;
20         if  (j>m) continue; ans[j]=mat[i][m+1];
21         for (k=j+1;k<=m;k++) ans[j]-=ans[k]*mat[i][k];
22         ans[j]=(ans[j]*power(mat[i][j],mod-2)%mod+mod)%mod;
23     }
24 }
```

1.10   瀚之的莫比乌斯

```
1  void getprime(int n){
2      miu[1]=pre[1]=b[1]=1;
3      for(int i=2;i<=n;i++){
4          if(!b[i]) p[mx[i]=++cnt]=i, miu[i]=-1;
5          for(int j=1;j<=mx[i];j++){
6              int k=i*p[j]; if(k>n) break;
7              b[k]=1; mx[k]=j;
8              if(j==mx[i]) miu[k]=0;
9                  else miu[k]=miu[i]*miu[p[j]];
10         }
11         pre[i]=pre[i-1]+i*miu[i];
12     }
13 }
14 ll f(int n,int m){return 1ll*n*m*(n+m+2)/2;}
15 ll calc(int n,int m){ // sigma{ i+j | i<=n,j<=m,gcd(i,j)==1 }
16     if(n>m) swap(n,m); ll ans=0;
17     for(int i=1,j=0,k=0;i<=n;i=min(j,k)+1)
18         ans+=(pre[min(j=n/(n/i),k=m/(m/i))]-pre[i-1])*f(n/i,m/i);
19     return ans;
20 }
```

1.11   FFT|NTT

```
1  typedef complex<double> comp;
2  comp A[N], B[N];
3  int rev[N], m, len;
4  inline void init(int n) {
5      for (m = 1, len = 0; m < n + n; m <<= 1 , len ++);
6      for (int i = 0; i < m; ++i) rev[i]=(rev[i>>1]>>1) | ((i&1)<<(len-1));
7      for (int i = 0; i < m; ++i) A[i] = B[i] = comp(0, 0);
8  }
9  inline void dft(comp *a, int v) {
```

```
10        for (int i = 0; i < m; ++i) if (i < rev[i]) swap(a[i] , a[rev[i]]);
11        for (int s = 2; s <= m; s <<= 1) {
12            comp g(cos(2 * pi / s) , v * sin(2 * pi / s));
13            // NTT: int g = power(gg, (mod - 1) / s);
14            // NTT: if (v == -1) g = power(g, mod - 2);
15            for (int k = 0; k < m; k += s) {
16                comp w(1, 0);
17                // NTT: int w = 1;
18                for (int j = 0; j < s / 2; ++j) {
19                    comp &u = a[k + j + s / 2], &v = a[k + j];
20                    comp t = w * u; u = v - t; v = v + t; w = w * g;
21                    // NTT: be aware of "+-*"
22                }
23            }
24        }
25        if (v == -1) for (int i = 0; i < m; ++i) a[i] /= m;
26        // NTT: be aware of "/"
27    }
```

1.12 求原根

```
1  vector <LL> a;
2  bool g_test(LL g, LL p) { for (LL i = 0; i < a.size(); ++i) if (pow_mod(g, (p-1)
       /a[i], p) == 1) return 0; return 1; }
3  LL p_root(LL p) {
4      LL tmp = p - 1;
5      for (LL i = 2; i <= tmp / i; ++i)
6          if (tmp % i == 0) { a.push_back(i); while (tmp % i == 0) tmp /= i; }
7      if (tmp != 1) a.push_back(tmp);
8      LL g = 1; while (1) { if (g_test(g, p)) return g; ++g; }
9  }
```

1.13 FWT

给定长度为 2^n 的序列 $A[0 \cdots 2^n - 1], B[0 \cdots 2^n - 1]$ ，求这两序列的 ① or 卷积: $C_k = \sum_{i \ or \ j=k} A_i B_j$; ② and 卷积: $C_k = \sum_{i \ and \ j=k} A_i B_j$; ③ xor 卷积: $C_k = \sum_{i \ xor \ j=k} A_i B_j$。

```
1  void FWT(int *a, int n) {
2      for (int d = 1; d < n; d <<= 1)
3          for (int m = d << 1, i = 0; i < n; i += m)
4              for (int j = 0; j < d; ++j) {
5                  int x = a[i + j], y = a[i + j + d];
6                  //or: a[i + j + d] = x + y;
7                  //and: a[i + j] = x + y;
```

```
8                  //xor: a[i + j] = x + y, a[i + j + d] = x - y;
9                  // 如答案要求取模，此处记得取模
10             }
11     }
12 void UFWT(int *a, int n) {
13     for (int d = 1; d < n; d <<= 1)
14         for (int m = d << 1, i = 0; i < n; i += m)
15             for (int j = 0; j < d; ++j) {
16                 int x = a[i + j], y = a[i + j + d];
17                 //or: a[i + j + d] = y - x;
18                 //and: a[i + j] = x - y;
19                 //xor: a[i + j] = (x + y) * 2^(-1), a[i + j + d] = (x - y) *
                       2^(-1);
20                 // 如答案要求取模，此处记得取模；2^(-1)表示2的逆元。
21             }
22 }
```

1.14 线性基

```
1  #define B 30
2  const int allset=(1<<B)-1;
3  struct LB {
4      int mat[B],cnt;
5      multiset<int> st;
6      LB(){}
7      void clear() { st.clear(); cnt=0; memset(mat,0,sizeof(mat)); }
8      void add(int x) {
9          for (int i=B-1;i>=0;i--) if ((x>>i)&1) {
10             if (mat[i]) x^=mat[i];
11             else { cnt++; mat[i]=x; break; }
12         }
13     }
14     void fix() {
15         for (int i=0;i<B;i++) if (mat[i])
16             for (int j=i+1;j<B;j++) if ((mat[j]>>i)&1) mat[j]^=mat[i];
17     }
18     void preset() { //正确性待定
19         fix(); for (int i=0;i<B;i++) if (mat[i]) st.insert(mat[i]);
20     }
21     int kth(int k) { //正确性待定
22         int i=0,ans=0; if (k<=0||k>(1<<cnt)-1) return 0;//无解
23         for (multiset<int>::iterator it=st.begin();it!=st.end();it++,i++)
24             if ((k>>i)&1) ans^=(*it);
25         return ans;
26     }
```

```
27      int getmax() {
28          fix(); int ans=0;
29          for (int i=B-1;i>=0;i--) if (ans^mat[i]>ans) ans^=mat[i];
30          return ans;
31      }
32  } tree[N*10];
```

2 计算几何

2.1 凸包

```
1   bool cmp(const Point &a,const Point &b) {
2       return F(a.x-b.x)<0||F(a.x-b.x)==0&&a.y<b.y; }
3   void Gram(int id[], int n) {
4       int i,mid; sort(id,id+n,cmp); tp=0; //凸包从x最小的点出发, 逆时针方向
5       for (i=0;i<n;i++) {
6           for (;tp>=2&&Cross(p[sk[tp-1]]-p[sk[tp-2]],p[id[i]]-p[sk[tp-1]])<=0;tp
                --);
7           //有重点必须用<=不留共线点, 无重点用<=不留共线点, 无重点用<留共线点
8           sk[tp++]=id[i];
9       }
10      mid=tp;
11      for (i=n-2;i>=0;i--) {
12          for (;tp>mid&&Cross(p[sk[tp-1]]-p[sk[tp-2]],p[id[i]]-p[sk[tp-1]])<=0;tp
                --);
13          //有重点必须用<=不留共线点, 无重点用<=不留共线点, 无重点用<留共线点
14          sk[tp++]=id[i];
15      }
16      if  (n>1) tp--;
17  }
```

2.2 定义

```
1   struct Point { double x,y; Point(){} Point(double _x,double _y):x(_x),y(_y){} };
2   struct Seg { Point a,b; Seg(){} Seg(Point _a,Point _b):a(_a),b(_b){} };
3   struct Circle { double x,y,r;
4       Point pt() { return Point(x,y); }
5       double Area() { return pi*r*r; }
6   };
7   Point operator +(const Point &a,const Point &b);
8   Point operator -(const Point &a,const Point &b);
9   Point operator *(const Point &a,double b);
10  Point operator /(const Point &a,double b);
11  int F(double x) { if (x>eps) return 1; if (x<-eps) return -1; return 0; }
```

```
12  bool operator ==(const Point &a,const Point &b) {
13      return F(a.x-b.x)==0&&F(a.y-b.y)==0; }
14  double Dist(const Point &a) { return sqrt(a.x*a.x+a.y*a.y); }
15  double Dot(const Point &a,const Point &b) { return a.x*b.x+a.y*b.y; }
16  double Cross(const Point &a,const Point &b) { return a.x*b.y-a.y*b.x; }
17  Point Rotate(const Point &p,double a) { // 逆时针旋转
18      return Point(p.x*cos(a)-p.y*sin(a),p.x*sin(a)+p.y*cos(a)); }
19  Point Inter(Seg a,Seg b) { // 两线段相交 (前提有交点)
20      double s=Cross(a.b-a.a,b.a-a.a),t=Cross(a.b-a.a,b.b-a.a);
21      return b.a+(b.b-b.a)*s/(s-t); }
22  vector<Point> SegCir(Seg seg,Point pt,double r) { //线圆
23      vector<Point> ans; double mul; Point vec,mid;
24      ans.clear(); vec=Rotate(seg.b-seg.a,pi/2);
25      mid=Inter(seg,Seg(pt,pt+vec));
26      if (F(Dist(pt-mid)-r)>0) return ans;
27      if (F(Dist(pt-mid)-r)==0) {
28          ans.push_back(mid); ans.push_back(mid); return ans;
29      }
30      vec=seg.b-seg.a; mul=sqrt(r*r-Dist2(mid-pt))/Dist(vec);
31      ans.push_back(mid+vec*mul); ans.push_back(mid-vec*mul);
32      return ans;
33  }
34  vector<Point> Circir(Circle a,Circle b) { //圆圆相交
35      vector<Point> ans; double dis,dis2,alpha; Point pa,pb,vec;
36      ans.clear(); if (a.r<b.r) swap(a,b);
37      pa=a.pt(); pb=b.pt(); vec=pb-pa;
38      dis=Dist(vec); dis2=Dist2(vec);
39      if (F(dis-(a.r+b.r))>0||F(dis-(a.r-b.r))<0) return ans;
40      if (F(dis-(a.r+b.r))==0) {
41          ans.push_back(pa+vec*a.r/(a.r+b.r)); return ans;
42      }
43      if (F(dis-(a.r-b.r))==0) {
44          ans.push_back(pa+vec*a.r/(a.r-b.r)); return ans;
45      }
46      alpha=acos((a.r*a.r+dis2-b.r*b.r)/2/a.r/dis);
47      ans.push_back(pa+Rotate(vec,alpha)*a.r/dis);
48      ans.push_back(pa+Rotate(vec,-alpha)*a.r/dis);
49      return ans;
50  }
51  double Bing(double ra,double rb,double dis) {
52      double alpha,beta; if (ra<rb) swap(ra,rb);
53      if (F(dis-(ra-rb))<=0) return pi*ra*ra;
54      if (F(dis-(ra+rb))>=0) return pi*ra*ra+pi*rb*rb;
55      alpha=acos((ra*ra+dis*dis-rb*rb)/2/dis/ra);
56      beta=acos((rb*rb+dis*dis-ra*ra)/2/dis/rb);
57      return (pi-alpha)*ra*ra+(pi-beta)*rb*rb+ra*dis*sin(alpha);
```

```
58  }
59  double Jiao(double ra,double rb,double dis) {
60      return pi*ra*ra+pi*rb*rb-Bing(ra,rb,dis); }
61  Point Gongmid(Circle a,Circle b) { //正确性待定
62      Point pa=a.pt(),pb=b.pt();
63      return pa+(pb-pa)*a.r/(a.r+b.r); }
64  Point Gongright(Circle a,Circle b) {
65      Point pa=a.pt(),pb=b.pt();
66      return pa+(pb-pa)*a.r/(a.r-b.r); }
67  int Ptinpol(Point pt) {
68      int wn=0;
69      for(int i=0;i<n;i++) {
70          if(Ins(pt,Seg(p[i],p[(i+1)%n]))) return 2;
71          int k=F(Cross(p[(i+1)%n]-p[i],pt-p[i]));
72          int d1=F(p[i].y-pt.y), d2=F(p[(i+1)%n].y-pt.y);
73          if(k>0&&d1<=0&&d2>0)wn++;
74          if(k<0&&d2<=0&&d1>0)wn--;
75      }
76      return wn!=0;
77  }
78  bool Cirinpol(Point pt) { //需要点在多边形内的前提
79      double nearest=inf;
80      for (int i=0;i<n;i++) {
81          nearest=min(nearest,Dist(p[i]-pt));
82          if   (F(Dot(pt-p[i],p[(i+1)%n]-p[i]))>0&&
83              F(Dot(pt-p[(i+1)%n],p[i]-p[(i+1)%n]))>0)
84          nearest=min(nearest,abs(Cross(p[i]-pt,p[(i+1)%n]-pt))/dis[i]);
85      }
86      return F(nearest-r)>=0;
87  }
88  bool Ins(const Point &p,const Seg &s) {
89      return F(Cross(s.a-p,s.b-p))==0&&F(p.x-min(s.a.x,s.b.x))>=0&&
90          F(p.x-max(s.a.x,s.b.x))<=0&&F(p.y-min(s.a.y,s.b.y))>=0&&
91          F(p.y-max(s.a.y,s.b.y))<=0; }
92  double PS(const Point &p,const Seg &s) { // 点到线段最短距离
93      if (F(Dot(p-s.a,s.b-s.a))<0||F(Dot(p-s.b,s.a-s.b))<0)
94          return min(Dist(p-s.a),Dist(p-s.b));
95      return abs(Cross(s.a-p,s.b-p))/Dist(s.a-s.b); }
96  double SS(const Seg &a,const Seg &b) { // 线段到线段最短距离
97      return min(min(PS(a.a,b),PS(a.b,b)),min(PS(b.a,a),PS(b.b,a))); }
98  double Alpha(Point a,Point b) {
99      double ans=atan2(b.y,b.x)-atan2(a.y,a.x);
100     if (ans<0) ans=-ans; if (ans>pi) ans=2*pi-ans; return ans; }
101 double Shan(Circle c,double a) { return c.r*c.r*a/2; }
```

2.3 半平面交

```
1   bool Cmphp(Seg a,Seg b) {
2       Point va=a.b-a.a, vb=b.b-b.a;
3       double dega=atan2(va.y,va.x), degb=atan2(vb.y,vb.x);
4       return F(dega-degb)<0||F(dega-degb)==0&&Cross(a.b-a.a,b.a-a.a)<0;
5   }
6   void HalfPlane(Seg hp[], int n, Point pol[], int &pols) {
7       Point mid;
8       hp[n++]=Seg(Point(-oo,-oo),Point(oo,-oo));
9       hp[n++]=Seg(Point(oo,-oo),Point(oo,oo));
10      hp[n++]=Seg(Point(oo,oo),Point(-oo,oo));
11      hp[n++]=Seg(Point(-oo,oo),Point(-oo,-oo));
12      sort(hp,hp+n,Cmphp);
13      int tp=0, low=0, high=-1; //sk 0~tp-1
14      for (int i=0;i<n;i++)
15      if (high-low+1==0||F(Cross(sk[high].b-sk[high].a,hp[i].b-hp[i].a))) {
16          for (;low<high;high--) {
17              mid=Inter(sk[high],sk[high-1]);
18              if (F(Cross(hp[i].b-hp[i].a,mid-hp[i].a))>0) break;
19          }
20          for (;low<high;low++) {
21              mid=Inter(sk[low],sk[low+1]);
22              if (F(Cross(hp[i].b-hp[i].a,mid-hp[i].a))>0) break;
23          }
24          sk[++high]=hp[i];
25      }
26      for (;low<high;high--) {
27          mid=Inter(sk[high],sk[high-1]);
28          if (Cross(sk[low].b-sk[low].a,mid-sk[low].a)>0) break;
29      }
30      tp=high-low+1; for (int i=0;i<tp;i++) sk[i]=sk[low+i];
31      pols=0; if  (tp<=2) return;
32      for (int i=0;i<tp;i++) pol[pols++]=Inter(sk[i],sk[(i+1)%tp]);
33  }
```

2.4 圆与多边形交集

```
1   double CT(Circle c,Point a,Point b) { // 圆与三角形交（多边形）
2       double da=Dist(a-c.pt()), db=Dist(b-c.pt());
3       if (da>db) { swap(a,b); swap(da,db); }
4       Seg s=Seg(a,b);
5       vector<Point> temp=CS(c,s);
6       if (F(db-c.r)<=0) return 0.5*abs(Cross(a-c.pt(),b-c.pt()));
```

```
7      if (F(da-c.r)<0) {
8          if (F(Dot(a-temp[1],b-temp[1]))<0) swap(temp[0],temp[1]);
9          return Shan(c,Alpha(temp[0]-c.pt(),b-c.pt()))+
10             0.5*abs(Cross(a-c.pt(),temp[0]-c.pt()));
11     }
12     if (!temp.size()) return Shan(c,Alpha(a-c.pt(),b-c.pt()));
13     if (Ins(temp[1],s)&&Dist2(a-temp[1])<Dist2(a-temp[0])) swap(temp[0],temp[1])
           ;
14     if (Ins(temp[0],s)&&Ins(temp[1],s)) {
15         return Shan(c,Alpha(a-c.pt(),temp[0]-c.pt()))+
16             Shan(c,Alpha(b-c.pt(),temp[1]-c.pt()))+
17             0.5*abs(Cross(temp[0]-c.pt(),temp[1]-c.pt()));
18     }
19     return Shan(c,Alpha(a-c.pt(),b-c.pt()));
20 }
```

2.5 三角形面积并

```
1  #define pr pair<ld,ld>
2  typedef long double ld;
3  const ld EPS=1e-8, INF=1e100;
4  struct Point {
5      ld x,y; Point(){} Point(ld _,ld __):x(_),y(__){}
6      void read() { double _x,_y; scanf("%lf%lf",&_x,&_y); x=_x,y=_y; }
7      friend bool operator<(Point a,Point b) {
8          if(fabs(a.x-b.x)<EPS) return a.y<b.y;
9          return a.x<b.x;
10     }
11     friend Point operator +(Point a,Point b) { return Point(a.x+b.x,a.y+b.y); }
12     friend Point operator -(Point a,Point b) { return Point(a.x-b.x,a.y-b.y); }
13     friend Point operator *(ld a,Point b) { return Point(a*b.x,a*b.y); }
14     friend ld operator *(Point a,Point b) { return a.x*b.x+a.y*b.y; }
15     friend ld operator ^(Point a,Point b) { return a.x*b.y-a.y*b.x; }
16 } a[N][3],Poi[N*N];
17 struct Line {
18     Point p,v; Line(){} Line(Point x,Point y){p=x,v=y-x;}
19     Point operator [](int k) { if(k) return p+v; else return p; }
20     friend bool Cross(Line a,Line b) {
21         return (a.v^b[0]-a.p)*(a.v^b[1]-a.p)<-EPS &&
22             (b.v^a[0]-b.p)*(b.v^a[1]-b.p)<-EPS;
23     }
24     friend Point getP(Line a,Line b) {
25         Point u=a.p-b.p; ld temp=(b.v^u)/(a.v^b.v);
26         return a.p+temp*a.v;
27     }
```

```
28 }l[N][3],T;
29 pr p[N];
30 int main() {
31     int n,m,i,j,k,x,y,cnt,tot;
32     ld ans,last,A,B,sum;
33     scanf("%d",&n);
34     for(i=1,tot=0;i<=n;i++) {
35         a[i][0].read(),a[i][1].read(),a[i][2].read();
36         Poi[++tot]=a[i][0],Poi[++tot]=a[i][1],Poi[++tot]=a[i][2];
37         sort(a[i],a[i]+3);
38         if((a[i][2]-a[i][0]^a[i][1]-a[i][0])>EPS)
39             l[i][0]=Line(a[i][0],a[i][2]),l[i][1]=Line(a[i][2],a[i][1]),l[i][2]=
                   Line(a[i][1],a[i][0]);
40         else
41             l[i][0]=Line(a[i][2],a[i][0]),l[i][1]=Line(a[i][1],a[i][2]),l[i][2]=
                   Line(a[i][0],a[i][1]);
42     }
43     for(i=1;i<=n;i++) for(j=1;j<i;j++) for(x=0;x<3;x++) for(y=0;y<3;y++)
44         if(Cross(l[i][x],l[j][y])) Poi[++tot]=getP(l[i][x],l[j][y]);
45     sort(Poi+1,Poi+tot+1);
46     ans=0,last=Poi[1].x; T=Line(Point(0,-INF),Point(0,INF));
47     for(i=2;i<=tot;i++) {
48         T.p.x=(last+Poi[i].x)/2;
49         for(j=1,cnt=0;j<=n;j++)
50             if(Cross(l[j][0],T)) {
51                 if(Cross(l[j][1],T)) B=getP(l[j][1],T).y;
52                 else B=getP(l[j][2],T).y;
53                 A=getP(l[j][0],T).y; if (A>B) swap(A,B);
54                 p[++cnt]=pr(A,B);
55             }
56         sort(p+1,p+cnt+1);
57         for(j=1,sum=0,A=-INF;j<=cnt;j++) {
58             if(p[j].first>A) sum+=p[j].second-p[j].first, A=p[j].second;
59             else if(p[j].second>A) sum+=p[j].second-A, A=p[j].second;
60         }
61         ans+=(Poi[i].x-last)*sum; last=Poi[i].x;
62     }
63     printf("%.2lf\n",(double)ans);
64 }
```

2.6 K 圆并

```
1  #define sqr(x) ((x)*(x))
2  const double eps = 1e-8;
3  double area[N]; int n;
```

```
4   int dcmp(double x) { if (x < -eps) return -1; else return x > eps; }
5   struct cp { double x, y, r, angle; int d;
6       cp(){} cp(double xx, double yy, double ang = 0, int t = 0) {
7           x = xx;  y = yy;  angle = ang;  d = t; }
8       void get() { scanf("%lf%lf%lf", &x, &y, &r); d = 1; }
9   }cir[N], tp[N * 2];
10  double dis(cp a, cp b) { return sqrt(sqr(a.x - b.x) + sqr(a.y - b.y)); }
11  double cross(cp p0, cp p1, cp p2) {
12      return (p1.x - p0.x) * (p2.y - p0.y) - (p1.y - p0.y) * (p2.x - p0.x);
13  }
14  int CirCrossCir(cp p1, double r1, cp p2, double r2, cp &cp1, cp &cp2) {
15      double mx = p2.x - p1.x, sx = p2.x + p1.x, mx2 = mx * mx;
16      double my = p2.y - p1.y, sy = p2.y + p1.y, my2 = my * my;
17      double sq = mx2 + my2, d = -(sq - sqr(r1 - r2)) * (sq - sqr(r1 + r2));
18      if (d + eps < 0) return 0; if (d < eps) d = 0; else d = sqrt(d);
19      double x = mx * ((r1 + r2) * (r1 - r2) + mx * sx) + sx * my2;
20      double y = my * ((r1 + r2) * (r1 - r2) + my * sy) + sy * mx2;
21      double dx = mx * d, dy = my * d; sq *= 2;
22      cp1.x = (x - dy) / sq; cp1.y = (y + dx) / sq;
23      cp2.x = (x + dy) / sq; cp2.y = (y - dx) / sq;
24      if (d > eps) return 2; else return 1;
25  }
26  bool circmp(const cp& u, const cp& v) { return dcmp(u.r - v.r) < 0; }
27  bool cmp(const cp& u, const cp& v) {
28      if (dcmp(u.angle - v.angle)) return u.angle < v.angle;
29      return u.d > v.d;
30  }
31  double calc(cp cir, cp cp1, cp cp2) {
32      double ans = (cp2.angle - cp1.angle) * sqr(cir.r)
33          - cross(cir, cp1, cp2) + cross(cp(0, 0), cp1, cp2);
34      return ans / 2;
35  }
36  void CirUnion(cp cir[], int n) {
37      cp cp1, cp2; sort(cir, cir + n, circmp);
38      for (int i = 0; i < n; ++i) for (int j = i + 1; j < n; ++j)
39          if (dcmp(dis(cir[i], cir[j]) + cir[i].r - cir[j].r) <= 0) cir[i].d++;
40      for (int i = 0; i < n; ++i) {
41          int tn = 0, cnt = 0;
42          for (int j = 0; j < n; ++j) {
43              if (i == j) continue;
44              if (CirCrossCir(cir[i],cir[i].r,cir[j],cir[j].r,cp2,cp1)<2) continue
                    ;
45              cp1.angle = atan2(cp1.y - cir[i].y, cp1.x - cir[i].x);
46              cp2.angle = atan2(cp2.y - cir[i].y, cp2.x - cir[i].x);
47              cp1.d = 1; tp[tn++] = cp1; cp2.d = -1; tp[tn++] = cp2;
48              if (dcmp(cp1.angle - cp2.angle) > 0) cnt++;
```

```
49          }
50          tp[tn++] = cp(cir[i].x - cir[i].r, cir[i].y, pi, -cnt);
51          tp[tn++] = cp(cir[i].x - cir[i].r, cir[i].y, -pi, cnt);
52          sort(tp, tp + tn, cmp);
53          int p, s = cir[i].d + tp[0].d;
54          for (int j = 1; j < tn; ++j) {
55              p = s;   s += tp[j].d;
56              area[p] += calc(cir[i], tp[j - 1], tp[j]);
57          }
58      }
59  }
60  void solve() {
61      for (int i = 0; i < n; ++i) cir[i].get();
62      memset(area, 0, sizeof(area));
63      CirUnion(cir, n);
64      for (int i = 1; i <= n; ++i) {
65          area[i] -= area[i + 1];
66          printf("[%d] = %.3lf\n", i, area[i]);
67      }
68  }
```

2.7 三维计算几何

```
1   Point Cross(Point a,Point b) {
2       return Point(a.y*b.z-a.z*b.y,a.z*b.x-a.x*b.z,a.x*b.y-a.y*b.x); }
3   double Crossxy(Point a,Point b) { return a.x*b.y-a.y*b.x; }
4   vector<Point> SegPlane(Seg seg,Plane p) {
5       vector<Point> ans; ans.clear();
6       Point fa=Cross(p.b-p.a,p.c-p.a);
7       if (F(Dot(fa,seg.b-seg.a))==0) return ans;
8       double s=Dot(p.a-seg.a,fa)/Dist(fa), t=Dot(p.a-seg.b,fa)/Dist(fa);
9       ans.push_back(seg.a+(seg.b-seg.a)*s/(s-t));
10      return ans;
11  }
12  // mixed product
13  double Mix(Point3 a,Point3 b,Point3 c) { return Dot(Cross(a,b),c); }
14  double PP(Point3 pt,Plane pl) { // distance from point to plane
15      Point3 fa=Cross(pl.b-pl.a,pl.c-pl.a);
16      return abs(Dot(fa,pt-pl.a))/Dist(fa);
17  }
18  // get the center point from 3D(need plane well prepared)
19  Point3 Getcenter(Point3 p[],int n,Plane pp[],int nn) {
20      double sumv=0;
21      Point3 sum=Point3(0,0,0);
22      for (int i=0;i<nn;i++)
```

```
23          {
24              double tempv=Mix(pp[i].b-pp[i].a,pp[i].c-pp[i].a,Point3(0,0,0)-pp[i].a);
25              sum=sum+(pp[i].a+pp[i].b+pp[i].c)*tempv/4.0;
26              sumv+=tempv;
27          }
28          return sum/sumv;
29  }
```

3 字符串

3.1 哈希

```
1  const int P=31,D=1000173169;
2  int hash(int l, int r) { return (LL)(f[r]-(LL)f[l-1]*pow[r-l+1]%D+D)%D; }
3  pow[0] = 1; for (int i=1;i<=n;i++) pow[i] = (LL)pow[i-1]*P%D;
4  for (int i=1;i<=n;i++) f[i] = (LL)((LL)f[i-1]*P+a[i])%D;
```

3.2 KMP

输入：模式串长度 n，模式串 a，匹配串长度 m，匹配串 b；输出：依次输出每个匹配成功的起始位置；下标从 0 开始。

```
1  void kmp(int n, char* a, int m, char *b) {
2      int i, j;
3      for (nxt[0] = j = -1, i = 1; i < n; nxt[i++] = j) {
4          while (~j && a[j + 1] != a[i]) j = nxt[j];
5          if (a[j + 1] == a[i]) ++j;
6      }
7      for (j = -1, i = 0; i < m; ++i) {
8          while (~j && a[j + 1] != b[i]) j = nxt[j];
9          if (a[j + 1] == b[i]) ++j;
10          if (j == n - 1) {
11              printf("%d\n", i - n + 1);
12              j = nxt[j];
13          }
14      }
15  }
```

3.3 扩展 KMP

next: a 关于自己每个后缀的最长公共前缀；ret: a 关于 b 的每个后缀的最长公共前缀；EXKMP 的 next[i] 表示：从 i 到 n-1 的字符串 st 前缀和原串前缀的最长重叠长度。

```
1  void get_next(char *a, int *next) {
```

```
2      int i, j, k, n = strlen(a);
3      for (j = 0; j+1<n && a[j]==a[j+1];j++);
4      next[1] = j; k = 1;
5      for (i=2;i<n;i++) {
6          int len = k+next[k], L = next[i-k];
7          if (L < len-i) {
8              next[i] = L;
9          } else {
10              for (j = max(0, len-i);i+j<n && a[j]==a[i+j];j++);
11              next[i] = j;
12              k = i;
13          }
14      }
15  }
16  void ExtendedKMP(char *a, char *b, int *next, int *ret) {
17      get_next(a, next);
18      int n = strlen(a), m = strlen(b);
19      int i, j, k;
20      for (j=0;j<n && j<m && a[j]==b[j];j++);
21      ret[0] = j;
22      k = 0;
23      for (i=1;i<m;i++) {
24          int len = k+ret[k], L = next[i-k];
25          if (L < len-i) {
26              ret[i] = L;
27          } else {
28              for (j = max(0, len-i);j<n && i+j<m && a[j]==b[i+j];j++);
29              ret[i] = j;
30              k = i;
31          }
32      }
33  }
```

3.4 Manacher

p[i] 表示以 i 为对称轴的最长回文串长度

```
1  char st[N*2], s[N];
2  int len, p[N*2];
3  while (scanf("%s", s) != EOF) {
4      len = strlen(s);
5      st[0] = '$', st[1] = '#';
6      for (int i=1;i<=len;i++)
7          st[i*2] = s[i-1], st[i*2+1] = '#';
8      len = len * 2 + 2;
9      int mx = 0, id = 0, ans = 0;
```

```
10      for (int i=1;i<=len;i++) {
11          p[i] = (mx > i) ? min(p[id*2-i]+1, mx-i) : 1;
12          for (; st[i+p[i]] == st[i-p[i]]; ++p[i]) ;
13          if (p[i]+i > mx) mx = p[i]+i, id = i;
14          p[i] --;
15          if (p[i] > ans) ans = p[i];
16      }
17      printf("%d\n", ans);
18  }
```

3.5 AC 自动机

```
1   struct Node { int next[26]; int terminal, fail; };
2   void build() {
3       head = 0, tail = 1; q[1] = 1;
4       while (head != tail) {
5           int x = q[++head];
6           /*(when necessary) node[x].terminal |= node[node[x].fail].terminal; */
7           for (int i=0;i<26;i++)
8               if (node[x].next[i]) {
9                   int y = node[x].fail;
10                  while (y) {
11                      if (node[y].next[i]) {
12                          node[node[x].next[i]].fail = node[y].next[i];
13                          break;
14                      }
15                      y = node[y].fail;
16                  }
17                  if (!node[node[x].next[i]].fail) node[node[x].next[i]].fail = 1;
18                  q[++tail] = node[x].next[i];
19              }
20      }
21  }
```

3.6 后缀数组

参数 m 表示字符集的大小，即 $0 \le r_i < m$

```
1   int n, r[N], wa[N], wb[N], ws[N], sa[N], rank[N], height[N];
2   int cmp(int *r, int a, int b, int l, int n) { return r[a]==r[b] && a+l<n && b+l<
        n && r[a+l]==r[b+l]; }
3   void suffix_array(int m) {
4       int i, j, p, *x=wa, *y=wb, *t;
5       for (i=0;i<m;i++) ws[i]=0; for (i=0;i<n;i++) ws[x[i]=r[i]]++;
6       for (i=1;i<m;i++) ws[i]+=ws[i-1]; for (i=n-1;i>=0;i--) sa[--ws[x[i]]]=i;
```

```
7       for (j=1,p=1;p<n;m=p,j<<=1) {
8           for (p=0,i=n-j;i<n;i++) y[p++]=i;
9           for (i=0;i<n;i++) if (sa[i]>=j) y[p++]=sa[i]-j;
10          for (i=0;i<m;i++) ws[i]=0; for (i=0;i<n;i++) ws[x[y[i]]]++;
11          for (i=1;i<m;i++) ws[i]+=ws[i-1];
12          for (i=n-1;i>=0;i--) sa[--ws[x[y[i]]]]=y[i];
13          for (t=x,x=y,y=t,x[sa[0]]=0,i=1,p=1;i<n;i++)
14              x[sa[i]]=cmp(y,sa[i-1],sa[i],j,n)?p-1:p++;
15      }
16      for (i=0;i<n;i++) rank[sa[i]]=i; rank[n] = -1;
17      for (i=j=0;i<n;i++) if (rank[i]) {
18          while (r[i+j]==r[sa[rank[i]-1]+j]) j++;
19          height[rank[i]]=j;
20          if (j) j--;
21      }
22  }
```

3.7 后缀自动机

下面的代码是求两个串的 LCS（最长公共子串）。

```
1   #define M (N << 1)
2   char st[N];
3   int pre[M], son[26][M], step[M], refer[M], size[M], tmp[M], topo[M], last, total
        ;
4   int apply(int x, int now) {
5       step[++total] = x;
6       refer[total] = now;
7       return total;
8   }
9   void extend(char x, int now) {
10      int p = last, np = apply(step[last]+1, now);
11      size[np] = 1;
12      for (; p && !son[x][p]; p=pre[p]) son[x][p] = np;
13      if (!p) pre[np] = 1;
14      else {
15          int q = son[x][p];
16          if (step[p]+1 == step[q]) pre[np] = q;
17          else {
18              int nq = apply(step[p]+1, now);
19              for (int i=0;i<26;i++) son[i][nq] = son[i][q];
20              pre[nq] = pre[q]; pre[q] = pre[np] = nq;
21              for (; p && son[x][p]==q; p=pre[p]) son[x][p] = nq;
22          }
23      }
24      last = np;
```

```
25 | }
26 | void init() {
27 |     last = total = 0;
28 |     last = apply(0, 0);
29 |     scanf("%s",st);
30 |     int n = strlen(st);
31 |     for (int i = 0; i <= n * 2; ++i) {
32 |         pre[i] = step[i] = refer[i] = size[i] = tmp[i] = topo[i] = 0;
33 |         for (int j = 0; j < 26; ++j) son[j][i] = 0;
34 |     }
35 |     for (int i = 0; i < n; ++i) extend(st[i] - 'a', i);
36 |     for (int i = 1; i <= total; ++i) tmp[step[i]] ++;
37 |     for (int i = 1; i <= n; ++i) tmp[i] += tmp[i - 1];
38 |     for (int i = 1; i <= total; ++i) topo[tmp[step[i]]--] = i;
39 |     for (int i = total; i; --i) size[pre[topo[i]]] += size[topo[i]];
40 | }
41 | int main() {
42 |     init();
43 |     int p = 1, now = 0, ans = 0;
44 |     scanf("%s", st);
45 |     for (int i=0; st[i]; i++) {
46 |         int index = st[i]-'a';
47 |         for (; p && !son[index][p]; p = pre[p], now = step[p]) ;
48 |         if (!p) p = 1;
49 |         if (son[index][p]) {
50 |             p = son[index][p]; now++;
51 |             if (now > ans) ans = now;
52 |         }
53 |     }
54 |     printf("%d\n",ans);
55 |     return 0;
56 | }
```

一些定义和性质：① Right(str) 表示 str 在母串 S 中所有出现的结束位置集合；② 一个状态 s 表示的所有子串 Right 集合相同，为 Right(s)；③ Parent(s) 满足 Right(s) 是 Right(Parent(s)) 的真子集，并且 Right(Parent(s)) 的大小最小；④ Parent 函数可以表示一个树形结构。不妨叫它 Parent 树；⑤ 一个 Right 集合和一个长度定义了一个子串；⑥ 对于状态 s，使得 Right(s) 合法的子串长度是一个区间 $[min(s), max(s)]$；⑦ $max(Parent(s)) = min(s) - 1$；⑧ 令 refer(s) 表示产生 s 状态的字符所在位置。则 Right(s) 的合法子串的起始位置为 $[refer(s) - max(s) + 1, refer(s) - min(s) + 1]$，即 $[refer(s) - max(s) + 1, refer(s) - max(Parent(s))]$。

代码中变量名含义：① pre[s] 为上述定义中的 Parent(s)；② step[s] 为从初始状态走到 s 状态最多需要多少步；③ refer[s] 为上述定义中的 refer(s)；④ size[s] 为 Right(s) 集合的大小；⑤ topo[s] 为 Parent 树的拓扑序，根（初始状态）在前。

3.8 回文树

① len[i] 表示编号为 i 的节点表示的回文串的长度（一个节点表示一个回文串）② next[i][c] 表示编号为 i 的节点表示的回文串在两边添加字符 c 以后变成的回文串的编号（和字典树类似）。③ fail[i] 表示节点 i 失配以后跳转不等于自身的节点 i 表示的回文串的最长后缀回文串（和 AC 自动机类似）。④ cnt[i] 表示节点 i 表示的本质不同的串的个数（建树时求出的不是完全的，最后 count() 函数跑一遍以后才是正确的）⑤ num[i] 表示以节点 i 表示的最长回文串的最右端点为回文串结尾的回文串个数。⑥ last 指向新添加一个字母后所形成的最长回文串表示的节点。⑦ st[i] 表示第 i 次添加的字符（一开始设 st[0] = -1（可以是任意一个在串 S 中不会出现的字符））。⑧ tot 表示添加的节点个数。⑨ n 表示添加的字符个数。

【URAL2040】Palindromes and Super Abilities 2

逐个添加字符串 S 里的字符 $S_1, S_2, ..., S_n$。每次添加字符后，他想知道添加字符后将出现多少个新的本质不同的回文子串。字符集为 $\{a, b\}$

```
1  | struct PAM {
2  |     int n, tot, last, len[N], fail[N], next[N][2], num[N], cnt[N];
3  |     void init() { n=0; tot=1; len[1]=-1; fail[1]=0; len[0]=+0; fail[0]=1; last
   |         =1; }
4  |     int get_fail(int x) { for (; st[n-len[x]-1]!=st[n]; x=fail[x]); return x; }
5  |     void insert(char c) {
6  |         ++n; int cur=get_fail(last); // 判断上一个串的前一个位置和新添加的位置是
   |             否相同，相同则说明构成回文。否则找 fail 指针。
7  |         if (!next[cur][c]) {
8  |             ++tot; len[tot]=len[cur]+2; fail[tot]=next[get_fail(fail[cur])][c];
9  |             next[cur][c]=tot; num[tot] = num[fail[tot]] + 1; answer[n]='1';
10 |         } else answer[n]='0';
11 |         last=next[cur][c]; cnt[last] ++;
12 |     }
13 |     void count () { for (int i=tot-1; i>=0; --i) cnt[fail[i]] += cnt[i]; }
14 |     //父亲累加儿子的cnt，因为如果fail[v]=u，则u一定是v的子回文串。
15 | } pam;
16 | n=strlen(st+1); pam.init();
17 | for (int i=1;i<=n;i++) pam.insert(st[i]-'a');
```

4 数据结构

4.1 ST 表

```
1 | int Log[N],f[17][N];
2 | int ask(int x,int y){ int k=Log[y-x+1]; return max(f[k][x],f[k][y-(1<<k)+1]); }
3 | for (int i=2;i<=n;i++)Log[i]=Log[i>>1]+1; for (int j=1;j<K;j++) for (int i=1;i
  |     +(1<<j-1)<=n;i++) f[j][i]=max(f[j-1][i],f[j-1][i+(1<<j-1)]);
```

4.2   K-D Tree

① change 将编号为 x 的点的权值增加 p；② euclid_lower_bound 欧几里得距离的平方，下界；③ euclid_upper_bound 欧几里得距离的平方，上界；④ manhattan_lower_bound 曼哈顿距离，下界；⑤ manhattan_upper_bound 曼哈顿距离，上界；⑥ add 添加一个点（注意此处的添加可能导致这棵树不平衡，慎用!）；⑦ ask(p, X, Y, ans) 询问距离点 (X, Y) 最远的一个点的距离，ans 需传入无穷小；⑧ ask(p, x1, y1, x2, y2) 查询矩形范围内所有点的权值和。

```
1  int n, cmp_d, root, id[N];
2  struct node { int d[2], l, r, Max[2], Min[2], val, sum, f; } t[N];
3  inline bool cmp(const node &a, const node &b) {
4      if (a.d[cmp_d] != b.d[cmp_d]) return a.d[cmp_d] < b.d[cmp_d];
5      return a.d[cmp_d ^ 1] < b.d[cmp_d ^ 1];
6  }
7  inline void umax(int &a, int b) { if (b > a) a = b; }
8  inline void umin(int &a, int b) { if (b < a) a = b; }
9  inline void up(int x, int y) { umax(t[x].Max[0], t[y].Max[0]); umin(t[x].Min[0],
        t[y].Min[0]); umax(t[x].Max[1], t[y].Max[1]); umin(t[x].Min[1], t[y].Min
        [1]); }
10 int build(int l, int r, int D, int f) {
11     int mid = (l + r) / 2; cmp_d = D;
12     nth_element(t + l + 1, t + mid + 1, t + r + 1, cmp);
13     id[t[mid].f] = mid; t[mid].f = f;
14     t[mid].Max[0] = t[mid].Min[0] = t[mid].d[0];
15     t[mid].Max[1] = t[mid].Min[1] = t[mid].d[1];
16     t[mid].val = t[mid].sum = 0;
17     if (l != mid) t[mid].l = build(l, mid - 1, !D, mid);
18     else t[mid].l = 0;
19     if (r != mid) t[mid].r = build(mid + 1, r, !D, mid);
20     else t[mid].r = 0;
21     if (t[mid].l) up(mid, t[mid].l);
22     if (t[mid].r) up(mid, t[mid].r);
23     return mid;
24 }
25 void change(int x, int p) {
26     x = id[x]; // 将点的编号映成排序后的编号
27     for (t[x].val += p; x; x = t[x].f) t[x].sum += p;
28 }
29 inline long long sqr(long long x) { return x * x; }
30 inline long long euclid_lower_bound(const node &a, int X, int Y) {
31     return sqr(max(max(X - a.Max[0], a.Min[0] - X), 0)) +
32         sqr(max(max(Y - a.Max[1], a.Min[1] - Y), 0)); }
33 inline long long euclid_upper_bound(const node &a, int X, int Y) {
34     return max(sqr(X - a.Min[0]), sqr(X - a.Max[0])) +
35         max(sqr(Y - a.Min[1]), sqr(Y - a.Max[1])); }
36 inline long long manhattan_lower_bound(const node &a, int X, int Y) {
37     return max(a.Min[0] - X, 0) + max(X - a.Max[0], 0) +
38         max(a.Min[1] - Y, 0) + max(Y - a.Max[1], 0);
39 }
40 inline long long manhattan_upper_bound(const node &a, int X, int Y) {
41     return max(abs(X - a.Max[0]), abs(a.Min[0] - X)) +
42         max(abs(Y - a.Max[1]), abs(a.Min[1] - Y));
43 }
44 void add(int k) {
45     t[k].Max[0] = t[k].Min[0] = t[k].d[0]; t[k].Max[1] = t[k].Min[1] = t[k].d
        [1];
46     t[k].val = t[k].sum = 0; t[k].l = t[k].r = t[k].f = 0;
47     if (!root) root = k, return;
48     int p = root, D = 0;
49     while (1) { up(p, k);
50         if (t[k].d[D] <= t[p].d[D]) { if (t[p].l) p = t[p].l; else t[p].l = k, t
            [k].f = p, return; }
51         else { if (t[p].r) p = t[p].r; else t[p].r = k, t[k].f = p, return; }
52         D ^= 1;
53     }
54 }
55 inline long long getdis(const node &a, int X, int Y) { return sqr(a.d[0] - X) +
        sqr(a.d[1] - Y); }
56 void ask(int p, int X, int Y, long long &ans) {
57     if (!p) return; ans = max(ans, getdis(t[p], X, Y));
58     long long dl = t[p].l ? euclid_upper_bound(t[t[p].l], X, Y) : 0;
59     long long dr = t[p].r ? euclid_upper_bound(t[t[p].r], X, Y) : 0;
60     if (dl > dr) { if (dl > ans) ask(t[p].l, X, Y, ans); if (dr > ans) ask(t[p].
        r, X, Y, ans); }
61     else { if (dr > ans) ask(t[p].r, X, Y, ans); if (dl > ans) ask(t[p].l, X, Y,
         ans); }
62 }
63 int ask(int p, int x1, int y1, int x2, int y2) {
64     if (t[p].Min[0] > x2 || t[p].Max[0] < x1 || t[p].Min[1] > y2 || t[p].Max[1]
        < y1) return 0;
65     if (t[p].Min[0] >= x1 && t[p].Max[0] <= x2 && t[p].Min[1] >= y1 && t[p].Max
        [1] <= y2) return t[p].sum;
66     int s = 0;
67     if (t[p].d[0] >= x1 && t[p].d[0] <= x2 && t[p].d[1] >= y1 && t[p].d[1] <= y2
        ) s += t[p].val;
68     if (t[p].l) s += ask(t[p].l, x1, y1, x2, y2);
69     if (t[p].r) s += ask(t[p].r, x1, y1, x2, y2);
70     return s;
71 }
72 for (int i = 1; i <= n; ++i) t[i].d[0] = x, t[i].d[1] = y;
73 root = build(1, n, 0, 0);
```

4.3 左偏树

左偏树是一个可并堆。下面的程序写的是一个小根堆，如果需要改成大根堆请在注释了 here 那行修改。接口：① push 插入一个元素；② merge 合并两个堆，注意，合并后原来那个堆将不可访问；③ top 返回堆顶元素；④ pop 删除堆顶元素；⑤ size 返回堆的大小。

```
template <class T> class leftist { public:
    struct node { T key; int dist; node *l, *r; };
    leftist() : root(NULL), s(0) {}
    void push(const T &x) { leftist y; y.s = 1; y.root = new node; y.root -> key
        = x; y.root -> dist = 0; y.root -> l = y.root -> r = NULL; merge(y); }
    node* merge(node *x, node *y) {
        if (x == NULL) return y; if (y == NULL) return x;
        if (y -> key < x -> key) swap(x, y); //here
        x -> r = merge(x -> r, y);
        int ld = x -> l ? x -> l -> dist : -1;
        int rd = x -> r ? x -> r -> dist : -1;
        if (ld < rd) swap(x -> l, x -> r);
        if (x -> r == NULL) x -> dist = 0;
        else x -> dist = x -> r -> dist + 1; return x;
    }
    void merge(leftist &x) { root = merge(root, x.root); s += x.s; }
    T top() const { if (root == NULL) return T(); return root -> key; }
    void pop() { if (root == NULL) return; node *p = root; root = merge(root ->
        l, root -> r); --s; delete p; }
    int size() const { return s; }
private: node* root; int s;
};
```

4.4 线段树小技巧

给定一个序列 a，寻找一个最大的 i 使得 $i \le y$ 且满足一些条件（如 $a[i] \ge w$，那么需要在线段树维护 a 的区间最大值）

```
int queryl(int p, int left, int right, int y, int w) {
    if (right <= y) {
        if (! __condition__ ) return -1;
        else if (left == right) return left;
    }
    int mid = (left + right) / 2;
    if (y <= mid) return queryl(p<<1|0, left, mid, y, w);
    int ret = queryl(p<<1|1, mid+1, right, y, w);
    if (ret != -1) return ret;
    return queryl(p<<1|0, left, mid, y, w);
}
```

给定一个序列 a，寻找一个最小的 i 使得 $i \ge x$ 且满足一些条件（如 $a[i] \ge w$，那么需要在线段树维护 a 的区间最大值）

```
int queryr(int p, int left, int right, int x, int w) {
    if (left >= x) {
        if (! __condition__ ) return -1;
        else if (left == right) return left;
    }
    int mid = (left + right) / 2;
    if (x > mid) return queryr(p<<1|1, mid+1, right, x, w);
    int ret = queryr(p<<1|0, left, mid, x, w);
    if (ret != -1) return ret;
    return queryr(p<<1|1, mid+1, right, x, w);
}
```

4.5 Splay

接口：① ADD x y d 将 $[x,y]$ 的所有数加上 d；② REVERSE x y 将 $[x,y]$ 翻转；③ INSERT x p 将 p 插入到第 x 个数的后面；④ DEL x 将第 x 个数删除。

```
int w[N], Min[N], son[N][2], size[N], father[N], rev[N], lazy[N];
int top, rt, q[N];
void pushdown(int x) {
    if (!x) return;
    if (rev[x]) rev[son[x][0]] ^= 1, rev[son[x][1]] ^= 1, swap(son[x][0], son[x
        ][1]), rev[x] = 0;
    if (lazy[x]) lazy[son[x][0]] += lazy[x], lazy[son[x][1]] += lazy[x], w[x] +=
        lazy[x], Min[x] += lazy[x], lazy[x] = 0;
}
void pushup(int x) {
    if (!x) return; pushdown(son[x][0]); pushdown(son[x][1]);
    size[x] = size[son[x][0]] + size[son[x][1]] + 1; Min[x] = w[x];
    if (son[x][0]) Min[x] = min(Min[x], Min[son[x][0]]);
    if (son[x][1]) Min[x] = min(Min[x], Min[son[x][1]]);
}
void sc(int x, int y, int w) { son[x][w] = y; father[y] = x; pushup(x); }
void _ins(int w) {
    top++; w[top] = Min[top] = w; son[top][0] = son[top][1] = 0;
    size[top] = 1; father[top] = 0; rev[top] = 0;
}
void init() { top = 0; _ins(0); _ins(0); rt=1; sc(1, 2, 1); }
void rotate(int x) {
    if (!x) return; int y = father[x], w = son[y][1]==x;
    sc(y, son[x][w^1], w); sc(father[y], x, son[father[y]][1]==y); sc(x, y, w^1)
        ;
```

```
23    }
24    void flushdown(int x) {
25        int t=0; for (; x; x=father[x]) q[++t]=x;
26        for (; t; t--) pushdown(q[t]);
27    }
28    void Splay(int x, int root=0) {
29        flushdown(x);
30        while (father[x] != root) { int y=father[x], w=son[y][1]==x;
31            if (father[y] != root && son[father[y]][w]==y) rotate(y);
32            rotate(x); }
33    }
34    int find(int k) {
35        Splay(rt);
36        while (1) { pushdown(rt);
37            if (size[son[rt][0]]+1==k) Splay(rt), return rt;
38            else if (size[son[rt][0]]+1<k) k-=size[son[rt][0]]+1, rt=son[rt][1];
39            else rt=son[rt][0]; }
40    }
41    int split(int x, int y) {
42        int fx = find(x), fy = find(y+2); Splay(fx); Splay(fy, fx); return son[fy
            ][0]; }
43    void add(int x, int y, int d) { //add d to each number in a[x]...a[y]
44        int t = split(x, y); lazy[t] += d; Splay(t); rt=t; }
45    void reverse(int x, int y) { // reverse the x-th to y-th elements
46        int t = split(x, y); rev[t] ^= 1; Splay(t); rt=t; }
47    void insert(int x, int p) { // insert p after the x-th element
48        int fx = find(x+1), fy = find(x+2);
49        Splay(fx); Splay(fy, fx); _ins(p); sc(fy, top, 0); Splay(top); rt=top; }
50    void del(int x) { // delete the x-th element in Splay
51        int fx = find(x), fy = find(x+2);
52        Splay(fx); Splay(fy, fx); son[fy][0] = 0; Splay(fy); rt=fy; }
```

4.6 可持久化 Treap

接口：① insert 在当前第 x 个字符后插入 c；② del 删除第 x 个字符到第 y 个字符；③ copy 复制第 l 个字符到第 r 个字符，然后粘贴到第 x 个字符后；④ reverse 翻转第 x 个到第 y 个字符；⑤ query 表示询问当前第 x 个字符是什么。

```
1    char key[N];
2    bool rev[N];
3    int lc[N], rc[N], size[N]; // if size is long long, remember here
4    int n, root;
5    LL Rand() { return rd = (rd * 20372052ll + 25022087ll) % mod; }
6    void init() { n = root = 0; }
7    inline int copy(int x) { ++ n; key[n] = key[x]; (copy rev, lc, rc, size); return
         n; }
```

```
8    inline void pushdown(int x) {
9        if (!rev[x]) return;
10       if (lc[x]) lc[x] = copy(lc[x]); if (rc[x]) rc[x] = copy(rc[x]);
11       swap(lc[x], rc[x]); rev[lc[x]] ^= 1; rev[rc[x]] ^= 1; rev[x] = 0;
12   }
13   inline void pushup(int x) { size[x] = size[lc[x]] + size[rc[x]] + 1; }
14   int merge(int u, int v) {
15       if (!u || !v) return u+v; pushdown(u); pushdown(v);
16       int t = Rand() % (size[u] + size[v]), r;  // if size is long long, remember
            here
17       if (t < size[u]) r = copy(u), rc[r] = merge(rc[u], v);
18       else r = copy(v), lc[r] = merge(u, lc[v]);
19       pushup(r); return r;
20   }
21   int split(int u, int x, int y) { // if size is long long, remember here
22       if (x > y) return 0; pushdown(u);
23       if (x == 1 && y == size[u]) return copy(u);
24       if (y <= size[lc[u]]) return split(lc[u], x, y);
25       int t = size[lc[u]] + 1; // if size is long long, remember here
26       if (x > t) return split(rc[u], x-t, y-t);
27       int num = copy(u); lc[num]=split(lc[u], x, t-1); rc[num]=split(rc[u], 1, y-t
            );
28       pushup(num); return num;
29   }
30   void insert(int x, char c) {
31       int t1 = split(root, 1, x), t2 = split(root, x+1, size[root]);
32       key[++n] = c; lc[n] = rc[n] = rev[n] = 0; pushup(n); root = merge(merge(t1,
            n), t2); }
33   void del(int x, int y) {
34       int t1 = split(root, 1, x-1), t2 = split(root, y+1, size[root]); root =
            merge(t1, t2); }
35   void copy(int l, int r, int x) {
36       int t1 = split(root, 1, x), t2 = split(root, l, r), t3 = split(root, x+1,
            size[root]);
37       root = merge(merge(t1, t2), t3); }
38   void reverse(int x, int y) {
39       int t1 = split(root, 1, x-1), t2 = split(root, x, y), t3 = split(root, y+1,
            size[root]);
40       rev[t2] ^= 1; root = merge(merge(t1, t2), t3); }
41   char query(int k) {
42       int x = root;
43       while (1) { pushdown(x);
44           if (k <= size[lc[x]]) x = lc[x];
45           else if (k == size[lc[x]] + 1) return key[x];
46           else k -= size[lc[x]] + 1, x = rc[x]; }
47   }
```

4.7 可持久化并查集

接口：① merge 在 time 时刻将 x 和 y 连一条边，注意加边顺序必须按 time 从小到大加边 ② GetFather 询问 time 时刻及以前的连边状态中，x 所属的集合

```
1  const int inf = 0x3f3f3f3f;
2  int father[N], Father[N], Time[N];
3  vector<int> e[N];
4  void init() { for (int i=1;i<=n;i++) father[i]=Father[i]=i,Time[i]=inf,e[i].
       clear(),e[i].push_back(i);}
5  int getfather(int x) { return (father[x]==x) ? x : father[x]=getfather(father[x
       ]); }
6  int GetFather(int x, int time) {return (Time[x]<=time)?GetFather(Father[x],time)
       :x;}
7  void merge(int x, int y, int time) {
8      int fx = getfather(x), fy = getfather(y); if (fx == fy) return;
9      if (e[fx].size() > e[fy].size()) swap(fx, fy);
10     father[fx] = fy; Father[fx] = fy; Time[fx] = time;
11     for (int i=0;i<e[fx].size();i++) e[fy].push_back(e[fx][i]);
12 }
```

4.8 普通莫队

分块块数为 $\sqrt{n}$ 是最优的。记每次进行 add() 操作的复杂度为 $O(A)$，del() 操作的复杂度为 $O(D)$，查询答案 answer() 的复杂度为 $O(S)$。则总复杂度为 $O(n\sqrt{n}(A+D)+qS)$。S 可以大一点，但必须保证 A, D 尽可能小。

```
1  struct Q { int l, r, sqrtl, id; } q[N];
2  int n, m, v[N], ans[N], nowans;
3  bool cmp(const Q &a, const Q &b) { if (a.sqrtl != b.sqrtl) return a.sqrtl < b.
       sqrtl; return a.r < b.r; }
4  void change(int x) { if (!v[x]) add(x); else del(x); v[x] ^= 1; }
5
6  for (int i=1;i<=m;i++) q[i].sqrtl = q[i].l / sqrt(n), q[i].id = i;
7  sort(q+1, q+m+1, cmp);
8  int L=1, R=0;
9  memset(v, 0, sizeof(v));
10 for (int i=1;i<=m;i++) {
11     while (L<q[i].l) change(L++);
12     while (L>q[i].l) change(--L);
13     while (R<q[i].r) change(++R);
14     while (R>q[i].r) change(R--);
15     ans[q[i].id] = answer();
16 }
```

4.9 树上莫队

```
1  struct Query { int l, r, id, l_group; } query[N];
2  int v[N], ans[N];
3  bool cmp(const Query &a, const Query &b) { if (a.l_group != b.l_group) return a.
       l_group < b.l_group; return dfn[a.r] < dfn[b.r]; }
4  void upd(int x) { if (!v[x]) add(x); else del(x); v[x] ^= 1; }
5  void go(int &u, int taru, int v) {
6      int lca0 = lca(u, taru);
7      int lca1 = lca(u, v); upd(lca1);
8      int lca2 = lca(taru, v); upd(lca2);
9      for (int x=u; x!=lca0; x=father[x]) upd(x);
10     for (int x=taru; x!=lca0; x=father[x]) upd(x);
11     u = taru;
12 }
13
14 for (int i=1;i<=m;i++) {
15     if (dfn[query[i].l] > dfn[query[i].r]) swap(query[i].l, query[i].r);
16     query[i].id = i; query[i].l_group = dfn[query[i].l] / sqrt(n);
17 }
18 sort(query+1, query+m+1, cmp);
19 int L=1,R=1; upd(1);
20 for (int i=1;i<=m;i++) {
21     go(L,query[i].l,R);
22     go(R,query[i].r,L);
23     ans[query[i].id] = answer();
24 }
```

5 树

5.1 点分治

```
1  void getsize(int x, int root = 0) {
2      size[x] = 1; son[x] = 0; int dd = 0;
3      for (int p = gh[x]; p; p = edge[p].next) {
4          int y = edge[p].adj;
5          if (y == root || vis[y]) continue;
6          getsize(y, x);
7          size[x] += size[y];
8          if (size[y] > dd) dd = size[y], son[x] = y;
9      }
10 }
11 int getroot(int x) {
12     int sz = size[x];
```

```
13        while (size[son[x]] > sz/2) x = son[x]; return x;
14  }
15  void dc(int x) {
16        getsize(x); x = getroot(x);
17        vis[x] = 1;
18        for (int p = gh[x]; p; p = edge[p].next) {
19            int y = edge[p].adj;
20            if (vis[y]) continue;
21            dc(y);
22        }
23        vis[x] = 0;
24  }
```

5.2   Link Cut Tree

① 注意，一开始必须调用 lct.init(0) ，否则求出的最小值一定会是 0 。② minval 维护的是链上 val 最小值。③ sumval2 维护的是子树 val2 的和。

```
1   int f[N], son[N][2], sz[N], rev[N], tot;
2   int val[N], minid[N], minval[N];
3   int val2[N], sumval2[N]; // 记得开 long long 。注意两个都要开 long long ，因为
        val2 还包含了虚儿子的子树和。
4   stack<int> s;
5   void init(int i) {
6        tot = max(tot, i); son[i][0] = son[i][1] = 0; f[i] = rev[i] = 0;
7        if (i == 0) sz[i] = 0, val[i] = minval[i] = inf, minid[i] = i, val2[i] =
            sumval2[i] = 0;
8        else sz[i] = 1, val[i] = minval[i] = VAL, minid[i] = i, val2[i] = sumval2[i]
            = VAL2;
9   }
10  bool isroot(int x) { return !f[x] || (son[f[x]][0] != x && son[f[x]][1] != x); }
11  void rev1(int x) { if (!x) return; swap(son[x][0], son[x][1]); rev[x] ^= 1; }
12  void down(int x) { if (!x) return; if (rev[x]) rev1(son[x][0]), rev1(son[x][1]),
        rev[x] = 0; }
13  void up(int x) { if (!x) return; down(son[x][0]); down(son[x][1]);
14        sz[x] = sz[son[x][0]] + sz[son[x][1]] + 1; minval[x] = val[x]; minid[x] = x;
15        if (minval[son[x][0]] < minval[x]) minval[x] = minval[son[x][0]], minid[x] =
            minid[son[x][0]];
16        if (minval[son[x][1]] < minval[x]) minval[x] = minval[son[x][1]], minid[x] =
            minid[son[x][1]];
17        sumval2[x] = sumval2[son[x][0]] + sumval2[son[x][1]] + val2[x];
18  }
19  void rotate(int x) {
20        int y = f[x], w = son[y][1] == x; son[y][w] = son[x][w ^ 1];
21        if (son[x][w ^ 1]) f[son[x][w ^ 1]] = y;
22        if (f[y]) {
23            int z = f[y];
24            if (son[z][0] == y) son[z][0] = x;
25            else if (son[z][1] == y) son[z][1] = x;
26        }
27        f[x] = f[y]; f[y] = x; son[x][w ^ 1] = y; up(y);
28  }
29  void splay(int x) {
30        while (!s.empty()) s.pop(); s.push(x);
31        for (int i = x; !isroot(i); i = f[i]) s.push(f[i]);
32        while (!s.empty()) down(s.top()), s.pop();
33        while (!isroot(x)) {
34            int y = f[x];
35            if (!isroot(y)) {
36                if ((son[f[y]][0] == y) ^ (son[y][0] == x)) rotate(x);
37                else rotate(y);
38            }
39            rotate(x);
40        } up(x);
41  }
42  void access(int x) {for (int y = 0; x; y = x, x = f[x]) splay(x), val2[x] +=
        sumval2[son[x][1]], son[x][1] = y, val2[x] -= sumval2[son[x][1]], up(x); }
43  int root(int x) { access(x); splay(x); while (son[x][0]) x = son[x][0]; return x
        ; }
44  void makeroot(int x) { access(x); splay(x); rev1(x); }
45  void link(int x, int y) {
46        makeroot(x); f[x] = y; access(x);
47        // 如果需要维护子树和 val2, sumval2 ，这样是不够的。因为增加了虚边，所以需要
            修改 val2 值。将上面的代码替换为以下代码：
48        // makeroot(x); makeroot(y); f[x] = y; val2[y] += sumval2[x]; up(y); access(
            x);
49  }
50  void cutf(int x) { access(x); splay(x); f[son[x][0]] = 0; son[x][0] = 0; up(x);
        } // 它和父亲的边
51  void cut(int x, int y) { makeroot(x); cutf(y); } // 切断 x 与 y 之间的边（须保证
        x 与 y 相邻）
52  int ask(int x, int y) { makeroot(x); access(y); splay(y); return minid[y]; } //
        询问 x 到 y 之间取得最小值的点
53  int querymin_cut(int x, int y) { int m = ask(x, y); makeroot(x); cutf(m);
        makeroot(y); cutf(m); return val[m]; } // 询问 x 到 y 之间取得最小值的点，并
        把它删去（须保证该点在 x 和 y 之间，且度数恰好为 2）
54  void link(int x, int y, int w) { init(++tot); val[tot] = minval[tot] = w; link(x
        , tot); link(y, tot); } // 在 x 和 y 之间添加一条权值为 w 的边（将边视为点插
        入）
55  int getsumval2(int x, int y) { makeroot(x); access(y); return val2[y]; } // 令 x
        为根，求 y 子树的 val2 的和
```

5.3 虚树

设 $a[0 \cdots k-1]$ 为需要构建虚树的点。

构建出虚树的节点保存在 a 数组中，k 为节点个数。加边调用函数 addedge(int x, int y, int w) 。

```
1  bool cmp(int x, int y) { return dfn[x] < dfn[y]; }
2  stack<int> stk;
3  sort(a, a + k, cmp);
4  int m = k;
5  for (int j = 1; j < m; ++j)
6      a[k++] = lca(a[j - 1], a[j]);
7  sort(a, a + k, cmp);
8  k = unique(a, a + k) - a;
9  stk.push(a[0]);
10 for (int j = 1; j < k; ++j) {
11     int u = lca(stk.top(), a[j]);
12     while (dep[stk.top()] > dep[u]) stk.pop();
13     assert(stk.top() == u);
14     stk.push(a[j]);
15     addedge(u, a[j], dis[a[j]] - dis[u]);
16 }
```

6 图

6.1 Tarjan 有向图强联通分量

① 割点的判断：一个顶点 u 是割点，当且仅当满足 (1) 或 (2)：(1) u 为树根，且 u 有多于一个子树（即：存在一个儿子 v 使得 $dfn[u]+1 \neq dfn[v]$）；(2) u 不为树根，且满足存在 (u,v) 为树枝边（u 为 v 的父亲），使得 $dfn[u] \leq low[v]$ 。② 桥的判断：一条无向边 (u,v) 是桥，当且仅当 (u,v) 为树枝边，满足 $dfn[u] < low[v]$ 。

```
1  struct EDGE { int adj, next; } edge[M];
2  int n, m, top, gh[N];
3  int dfn[N], low[N], cnt, ind, stop, instack[N], stack[N], belong[N];
4  void addedge(int x, int y) { edge[++top].adj = y; edge[top].next = gh[x]; gh[x]
       = top; }
5  void tarjan(int x) {
6      dfn[x] = low[x] = ++ind;
7      instack[x] = 1; stack[++stop] = x;
8      for (int p=gh[x]; p; p=edge[p].next)
9          if (!dfn[edge[p].adj]) tarjan(edge[p].adj), low[x] = min(low[x], low[
               edge[p].adj]);
10         else if (instack[edge[p].adj]) low[x] = min(low[x], dfn[edge[p].adj]);
11     if (dfn[x] == low[x]) {
12         ++cnt; int tmp=0;
```

```
13         while (tmp!=x) tmp = stack[stop--], belong[tmp] = cnt, instack[tmp] = 0;
14     }
15 }
```

6.2 Tarjan 双联通分量

以下代码为点双联通分量。若要更改为边双联通，在第 8 行将 $low[next] \geq dfn[x]$ 改为 $low[next] > dfn[x]$ ，并将 14 行 $vec[tot].push\_back(x)$ 删除。

```
1  void DFS(int x,int fa) {
2      vis[x]=true; dfn[x]=low[x]=++times; sk[++tp]=x;
3      for (int pt=first[x];pt;pt=e[pt].next) {
4          int next=e[pt].to; if (e[pt].id==fa) continue;
5          if (!vis[next]) {
6              DFS(next,e[pt].id);
7              low[x]=min(low[x],low[next]);
8              if (low[next]>=dfn[x]) { // ***
9                  vec[++tot].clear();
10                 while (tp) {
11                     vec[tot].push_back(sk[tp--]);
12                     if (sk[tp+1]==next) break;
13                 }
14                 vec[tot].push_back(x); // ***
15             }
16         } else if (dfn[next]>last) low[x]=min(low[x],dfn[next]);
17     }
18 }
19 for (i=1;i<=n;i++) if (!vis[i]) {
20     DFS(i,0); last=times;
21     if (tp) {
22         tot++; vec[tot].clear();
23         for (i=1;i<=tp;i++) vec[tot].push_back(sk[i]);
24         tp=0;
25     }
26 }
```

6.3 欧拉回路

```
1  struct E { int to,ne; } e[M<<1];
2  int t,n,m,la[N],e_top;
3  int in[N],out[N];
4  void add(int x, int y){
5      out[x]++; in[y]++;
6      e[++e_top]=(E){y,la[x]}; la[x]=e_top;
7  }
```

```
8    int sta[M],top;
9    bool vis[M<<1];
10   void dfs(int x){
11       for(int i=la[x]; i; i=la[x]){
12           la[x]=e[i].ne;
13           if (vis[i]) continue;
14           vis[i]=true; if (t==1) vis[i^1]=true;
15           dfs(e[i].to);
16           if (t==2) sta[++top]=i;
17               else sta[++top]=(i&1)?(-(i>>1)):(i>>1);
18       }
19   }
20   int main(){
21       scanf("%d%d%d",&t,&n,&m);
22       if (m==0) YES(); if (t==1) e_top=1;
23       ft(i,1,m){ scanf("%d%d",&x,&y); add(x,y); if (t==1) add(y,x); }
24       if (t==1) ft(i,1,n) if (in[i]&1) NO();
25       if (t==2) ft(i,1,n) if (in[i]!=out[i]) NO();
26       dfs(e[3-t].to); if (top!=m) NO();
27       YES(); fd(i,top,1) printf("%d_",sta[i]);
28   }
```

```
22           forE(i,x){
23               int y=e[i].to;
24               if (cond[y]==-1){
25                   if (mat[y]==0){
26                       while (x){
27                           int t=mat[x]; mat[x]=y; mat[y]=x; y=t; x=pre[y];
28                       }
29                       return true;
30                   }
31                   cond[y]=1; pre[y]=x; cond[q[++r]=mat[y]]=0;
32               } else if (find(x)!=find(y) && cond[y]==0){
33                   int g=lca(x,y); blossom(x,y,g); blossom(y,x,g);
34               }
35           }
36       }
37       return false;
38   }
39   int main(){
40       scanf("%d%d",&n,&m); int ans=0;
41       while (m--){ scanf("%d%d",&x,&y); add(x,y); add(y,x); }
42       forto(i,1,n) if (!mat[i] && match(i)) ans++;
43       printf("%d\n",ans); forto(i,1,n) printf("%d_",mat[i]);
44   }
```

6.4 带花树

```
1    const int N=550;
2    struct E { int to,ne; } e[N*N];
3    int n,m,la[N],e_top,f[N];
4    int find(int x) { return f[x]=f[x]==x?x:find(f[x]); }
5    int mat[N],pre[N],cond[N],q[N],l,r,vis[N],vt;
6    int lca(int x, int y){
7        vt++; x=find(x); y=find(y);
8        while (vis[x]!=vt){ if(x){vis[x]=vt;x=find(pre[mat[x]]);} swap(x,y); }
9        return x;
10   }
11   void blossom(int x, int y, int g){
12       while (find(x)!=g){
13           pre[x]=y; if (cond[mat[x]]==1) cond[q[++r]=mat[x]]=0;
14           if (f[x]==x) f[x]=g; if (f[mat[x]]==mat[x]) f[mat[x]]=g;
15           y=mat[x]; x=pre[y];
16       }
17   }
18   int match(int s){
19       forto(i,1,n){ cond[i]=-1; pre[i]=0; f[i]=i; }
20       cond[q[l=r=1]=s]=0;
21       while (l<=r){ int x=q[l++];
```

6.5 KM 算法

```
1    const int N=500, inf=0x7fffffff;
2    int n,fx[N],fy[N],pre[N];
3    LL w[N][N],lx[N],ly[N],sla[N];
4    bool vx[N],vy[N],a[N][N];
5    int q[N],l,r;
6    bool check(int x, int y){
7        if (!fy[y]){
8            while (x){ int t=fx[x]; fx[x]=y; fy[y]=x; y=t; x=pre[y]; }
9            return true;
10       }
11       vy[y]=true; pre[y]=x; vx[q[++r]=fy[y]]=true; return false;
12   }
13   void bfs(int s){
14       ft(i,1,n) { vx[i]=vy[i]=false; sla[i]=inf; }
15       vx[q[l=r=1]=s]=true;
16       while (true){
17           while (l<=r){
18               int x=q[l++];
19               ft(y,1,n) if (!vy[y]){
```

```
20              LL t=lx[x]+ly[y]-w[x][y];
21              if (t==0 && check(x,y)) return;
22              if (t && t<sla[y]) { sla[y]=t; pre[y]=x; }
23          }
24      }
25      int d=inf;
26      ft(y,1,n) if (!vy[y]) cmin(d,sla[y]);
27      ft(x,1,n) if (vx[x]) lx[x]-=d;
28      ft(y,1,n) if (vy[y]) ly[y]+=d; else sla[y]-=d;
29      ft(y,1,n) if (!vy[y] && !sla[y] && check(pre[y],y)) return;
30      }
31  }
32  void KM(){
33      ft(x,1,n) { lx[x]=w[x][1]; ft(y,2,n) cmax(lx[x],w[x][y]); }
34      ft(s,1,n) bfs(s);
35  }
36  int main(){
37      int nl,nr,m; scanf("%d%d%d",&nl,&nr,&m);
38      while (m--){ scanf("%d%d%d",&x,&y,&z); w[x][y]=z; a[x][y]=true; }
39      n=MAX(nl,nr); KM();
40      LL ans=0; ft(i,1,n) ans+=lx[i]; ft(j,1,n) ans+=ly[j];
41      printf("%lld\n",ans);
42      ft(i,1,nl) printf("%d ",a[i][fx[i]]?fx[i]:0);
43  }
```

6.6 2-SAT

记 $x \to y$ 的有向边表示选了 x 就要选 y。

```
1   struct MergePoint {
2       struct EDGE { int adj, next; } edge[M];
3       int ex[M], ey[M]; bool instack[N];
4       int gh[N], top, dfn[N], low[N], cnt, ind, stop, stack[N], belong[N];
5       void init() { cnt = ind = stop = top = 0; memset(dfn, 0, sizeof(dfn));
            memset(instack, 0, sizeof(instack)); memset(gh, 0, sizeof(gh)); }
6       void addedge(int x, int y) { swap(x, y); edge[++top].adj = y; edge[top].next
            = gh[x]; gh[x] = top; ex[top] = x; ey[top] = y; }
7       void tarjan(int x) {}
8       void work() { for (i) if (!dfn[i]) tarjan(i); }
9   } merge;
10  struct Topsort {
11      struct EDGE { int adj, next; } edge[M];
12      int n, top, gh[N], ops[N], deg[N], ans[N]; std::queue<int> q;
13      void init() { n = merge.cnt; top = 0; memset(gh, 0, sizeof(gh)); memset(deg,
            0, sizeof(deg)); }
```

```
14      void addedge(int x, int y) { if (x == y) return; edge[++top].adj = y; edge[
            top].next = gh[x]; gh[x] = top; ++deg[y]; }
15      void work() {
16          for (int i = 1; i <= n; ++i) if (!deg[i]) q.push(i);
17          while (!q.empty()) {
18              int x = q.front(); q.pop();
19              for (int p = gh[x]; p; p = edge[p].next) if (!--deg[edge[p].adj]) q.
                    push(edge[p].adj);
20              if (ans[x]) continue; ans[x] = -1; ans[ops[x]] = 1; //-1 NO, 1 YES
21          }
22      }
23  } ts;
24  merge.init(); merge.addedge(); merge.work();
25  for (int i = 1; i <= n; ++i) {
26      int x = merge.belong[U(i, 0)], y = merge.belong[U(i, 1)];
27      if (x==y) NO(); ts.ops[x]=y; ts.ops[y]=x;
28  }
29  ts.init(); ts.work();
30  puts("YES"); for (int i = 1; i <= n; ++i) select(ts.ans[merge.belong[U(i,1)] ==
        1);
```

6.7 网络流

6.7.1 最大流

注意: *top 要初始化为 1*

```
1   struct EDGE { int adj, w, next; } edge[M];
2   int n, top, gh[N], nrl[N], dist[N], q[N];
3   void addedge(int x, int y, int w) { edge[++top].adj = y; edge[top].w = w; edge[
        top].next = gh[x]; gh[x] = top; edge[++top].adj = x; edge[top].w = 0; edge[
        top].next = gh[y]; gh[y] = top; }
4   int bfs() {
5       memset(dist, 0, sizeof(dist));
6       q[1] = S; int head = 0, tail = 1; dist[S] = 1;
7       while (head != tail) {
8           int x = q[++head];
9           for (int p=gh[x]; p; p=edge[p].next)
10              if (edge[p].w && !dist[edge[p].adj]) {
11                  dist[edge[p].adj] = dist[x] + 1;
12                  q[++tail] = edge[p].adj;
13              }
14      }
15      return dist[T];
16  }
17  int dinic(int x, int delta) {
```

```
18      if (x==T) return delta;
19      for (int& p=nrl[x]; p && delta; p=edge[p].next)
20          if (edge[p].w && dist[x]+1 == dist[edge[p].adj]) {
21              int dd = dinic(edge[p].adj, min(delta, edge[p].w));
22              if (!dd) continue;
23              edge[p].w -= dd;
24              edge[p^1].w += dd;
25              return dd;
26          }
27      return 0;
28  }
29  int ans = 0; while (bfs()) { memcpy(nrl, gh, sizeof(gh)); int t; while (t =
        dinic(S, inf)) ans += t; } return ans;
```

6.7.2 上下界有源汇网络流

①T 向 S 连容量为正无穷的边，将有源汇转化为无源汇。②每条边容量减去下界，设 $in[i]$ 表示流入 i 的下界之和减去流出 i 的下界之和。③新建超级源汇 SS, TT ，对于 $in[i] > 0$ 的点，SS 向 i 连容量为 $in[i]$ 的边。对于 $in[i] < 0$ 的点，i 向 TT 连容量为 $-in[i]$ 的边。④求出以 SS, TT 为源汇的最大流，如果等于 $\Sigma in[i](in[i] > 0)$ ，则存在可行流。再求出 S, T 为源汇的最大流即为最大流。⑤费用流：建完图后等价于求以 SS, TT 为源汇的费用流。

6.7.3 费用流

注意：top 要初始化为 1

```
1   struct EDGE { int adj, w, cost, next; } edge[M*2];
2   int gh[N], q[N], dist[N], v[N], pre[N], prev[N], top, S, T;
3   void addedge(int x, int y, int w, int cost) {x->y(w,cost); y->x(0,-cost);}
4   void clear() { top = 1; memset(gh, 0, sizeof(gh)); }
5   bool spfa() {} // 从S出发，返回dist[T] != inf
6   int ans = 0;
7   while (spfa()) {
8       int mx = inf;
9       for (int x=T;x!=S;x=pre[x]) mx = min(edge[prev[x]].w, mx);
10      ans += dist[T] * mx;
11      for (int x=T;x!=S;x=pre[x]) edge[prev[x]].w -= mx, edge[prev[x]^1].w += mx;
12  }
13  return ans;
```

7 杂项

7.1 读入优化

int rd(int &x); 读入一个整数，保存在变量 x 中。如正常读入，返回值为 1 ，否则返回 EOF (-1)

```
1   #define rd RD<int>
2   #define rdll RD<long long>
3   const int S = 2000000; // 2MB
4   char s[S], *h = s+S, *t = h;
5   inline char getchr(void) {
6       if(h == t) { if (t != s + S) return EOF; t = s + fread(s, 1, S, stdin); h =
            s; }
7       return *h++;
8   }
9   template <class T>
10  inline int RD(T &x) {
11      char c = 0; int sign = 0;
12      for (; !isdigit(c); c = getchr()) {
13          if (c == EOF) return -1; if (c == '-') sign ^= 1;
14      }
15      x = 0; for (; isdigit(c); c = getchr()) x = x * 10 + c - '0';
16      if (sign) x = -x; return 1;
17  }
```

7.2 Vim

```
1   syntax on
2   set cindent
3   set nu
4   set tabstop=4
5   set shiftwidth=4
6   set background=dark
7
8   inoremap <C-j> <down>
9   inoremap <C-k> <up>
10  inoremap <C-h> <left>
11  inoremap <C-l> <right>
```

7.3 Java

```
1   头文件
2   import java.math.*;
```

```java
3  import java.util.*;
4  public class Main {
5      public static void main(String []args) {
6      }
7  }
```

8 输入输出
```java
9  Scanner cin = new Scanner(System.in);
10 int a = cin.nextInt();
11 BigDecimal a = cin.nextBigDecimal();
12 while (cin.hasNext()) {} // 输入到 EOF 结束
13 System.out.println(str); // 有换行
14 System.out.print(str); // 无换行
15 System.out.println("Hello,_%s._Next_year,_you'll_be_%d", name, age); // C风格输
       出
```

16 大数类
```java
17 BigInteger a = BigInteger.valueOf(12);
18 BigInteger b = new BigInteger(String.valueOf(12));
19 BigDecimal c = BigDecimal.valueOf(12.0);
20 BigDecimal d = new BigDecimal("12.0"); // 字符串防止double精度误差
```

21 大数比较
```java
22 c.compareTo(BigDecimal.ZERO)==0 //判断相等, c==0
23 c.compareTo(BigDecimal.ZERO)>0 //判断大于, c>0
24 c.compareTo(BigDecimal.ZERO)<0 //判断小于, c<0
```

25 大数基本运算
```java
26 Big*** add(Big*** b) // 加上b
27 Big*** subtract(Big*** b) // 减去b
28 Big*** multiply(Big*** b) // 乘b
29 Big*** divide(Big*** b) // 除以b
30 BigDecimal divide(BigDecimal b, int 精确位数, BigDecimal.ROUND_HALF_UP); // 除以
       b, 保留小数
31 Big*** pow(int b) // this^b
32 Big*** remainder(Big*** b) // mod b
33 Big*** abs() // 绝对值
34 Big*** negate() // 取负号
35 Big*** max(Big*** b) // 返回this和b中的最大值
36 Big*** min(Big*** b) // 返回this和b中的最小值
37 BigInteger gcd(BigInteger val) // 返回abs(this)和abs(val)的最大公约数
38 BigInteger mod(BigInteger val) // 求 this mod val
39 BigInteger modInverse(BigInteger val) // 求逆元, 返回 this^(-1) mod val
```

40 大数格式控制
```java
41 toString()将BigDecimal转成字符串, 然后配合一些字符串函数进行处理:
42 str.startWith("0"); // 以0开始
43 str.endWith("0"); // 以0结束
44 str.subString(int x, int y); // 从x到y的str的子串
45 str.subString(int x); // 从x到结尾的子串
46 c.stripTrailingZeros().toPlainString(); // c去除末尾0, 转成普通字符串
47 setScale(int newScale, RoundingMode roundingMode) 返回BigDecimal。newScale表示保
       留位数。CEILING/DOWN/FLOOR/HALF_DOWN/HALF_UP。
```

48 大数进制转换
49 支持2～36进制 (0-9 + 小写a-z)
```java
50 BigInteger a=cin.nextBigInteger(2); // 读入一个二进制数
51 System.out.println(a.toString(2)); // 输出二进制
```