Template v2.0 for ACM-CCPC in JLU

zig_zag

September 23, 2016

E	录			4.16 Splay_Segment	
1	Math 1.1 BigInter 1.2 BinarySearch 1.3 CRT 1.4 EulerFunction 1.5 EulerSieve 1.6 ExtendedEuclidean 1.7 FFT 1.8 FFT-2 1.9 GCD 1.10 GaussElimination 1.11 LucasTheory 1.12 MatrixQuickPower	1 1 1 1 2 2 2 2 3 3 3 3	5	5.6 SAP_MaxFlow 5.7 SPFA 5.8 SPFA_CostFlow	17 17 17 18 18 18 19 19
	1.13 MillerRabin_PollardRho	3 4 5	6	Geometry 6.1 Geometry	20 20
	1.16 NTT 1.17 QuadraticField 1.18 QuickPower 1.19 SlowMul 1.20 Trisection 1.21 XorGauss	5 5 6 6 6	7	Others 7.1 Bitwise . . 7.2 MoDui . . 7.3 OpenStack . . 7.4 PK . . 7.5 QuickRead . .	23 23 23
2	DP 2.1 DigitDP	6 6 6 6 7			
3	String 3.1 AC_Automaton 3.2 KMP 3.3 StringHash 3.4 Suffix_Automaton	7 7 8 8 8			
4	4.7 ChainPartitionTree 4.8 ChairManTree 4.9 DSJ 4.10 KD_Tree 4.11 LCT 4.12 LeftistTree 4.13 Merge	8 8 9 9 9 9 10 10 11 13 13 14 14 14			

1 Math

1.1 BigInter

```
//高精度
2
    struct high
3
      int e[100];
      void clean()
 5
 6
         memset(e, 0, sizeof(e));
8
9
    }a;
10
11
    high operator +(high a, high b)
12
      high c; c.clean();
13
      c.e[0]=max(a.e[0],b.e[0]);
14
15
      for (int i=1;i<=c.e[0];i++)</pre>
16
17
         c.e[i]+=a.e[i]+b.e[i];
18
         c.e[i+1]+=c.e[i]/p;
19
         c.e[i]=c.e[i]%p;
20
      if (c.e[c.e[0]+1]) c.e[0]++;
21
22
      return c;
23
24
25
    high operator — (high a, high b)
26
      high c; c.clean();
27
28
      int add=0;
29
      c.e[0]=max(a.e[0],b.e[0]);
30
      for (int i=1;i<=c.e[0];i++)</pre>
31
32
         c.e[i]=a.e[i]-add-b.e[i];
33
         if (c.e[i]<0) c.e[i]+=p, add=1;</pre>
34
         else add=0;
35
36
      while (c.e[c.e[0]]==0) c.e[0]---;
37
      return c;
    }
38
39
    high operator *(high a, high b)
40
41
42
      high c; c.clean();
      c.e[0]=a.e[0]+b.e[0]-1;
43
      for (int i=1;i<=a.e[0];i++)</pre>
44
45
         for (int j=1;j<=b.e[0];j++)</pre>
           c.e[i+j-1]+=a.e[i]*b.e[j];
46
      for (int i=1;i<=c.e[0];i++)</pre>
47
48
         c.e[i+1]+=c.e[i]/p;
49
50
         c.e[i]=c.e[i]%p;
51
52
      if (c.e[c.e[0]+1]) c.e[0]++;;
53
      return c;
54
55
56
    high operator /(high a, int b)
57
58
      int down=0;
      high c; c.clean();
59
60
      c.e[0]=a.e[0];
61
      for (int i=a.e[0];i;i—)
62
         c.e[i]=(a.e[i]+down*p)/b;
63
64
        down=(a.e[i]+down*p)-c.e[i]*b;
65
      while ((c.e[c.e[0]]==0)&&(c.e[0]>0)) c.e[0]--;
67
      return c;
68
69
70
    int operator %(high a,int b)
71
72
73
      for (int i=a.e[0];i;i—)
74
         mod=((mod*p%b)+a.e[i])%b;
75
      return mod;
76
   high max(high a,high b)
78
```

```
if (a.e[0]>b.e[0]) return a;
                                                                 80
  if (a.e[0]<b.e[0]) return b;
                                                                 81
  for (int i=a.e[0];i;i-
                                                                 82
                                                                 83
    if (a.e[i]>b.e[i]) return a;
                                                                 84
    if (a.e[i] < b.e[i]) return b;</pre>
                                                                 85
                                                                 86
                                                                 87
  return a;
                                                                 88
                                                                 89
void read(high &a)
                                                                 90
                                                                 91
  char ch=getchar(); high x; x.clean();
                                                                 92
  for (;ch<'0'||ch>'9';ch=getchar());
                                                                 93
  for (;ch>='0'&&ch<='9';ch=getchar()) x.e[++x.e[0]]=ch-'0
                                                                 94
  for (int i=1; i \le x.e[0]; i++) a.e[x.e[0]-i+1]=x.e[i];
                                                                 95
  a.e[0]=x.e[0];
                                                                 96
                                                                 97
                                                                 98
void write(high a)
                                                                 99
                                                                 100
  printf("%d",a.e[a.e[0]]);
                                                                 101
  for (int i=a.e[0]-1;i>0;i—) printf("%d",a.e[i]);
                                                                 102
  printf("\n");
                                                                 103
                                                                 104
```

1.2 BinarySearch

```
|//整数二分
int l=1, r=n, ans;
while (1<=r)
                                                             3
  int mid=(l+r)>>1;
  if (judge(mid)) l=mid+1, ans=mid;
                                                             6
  else r=mid-1;
                                                             7
                                                             8
//浮点数二分
                                                             9
hile (1-r<=1e-6)
                                                             10
                                                             11
  double mid=(1+r)/2.0;
                                                             12
  if (judge(mid)<=0.0) r=mid-0.000001,ans=mid;//所有>0的解
                                                             13
  均不合法
  else l=mid+0.000001;
                                                             14
                                                             15
```

1.3 CRT

```
//中国剩余定理(互质)
LL crt(int n, LL* a, LL* p)
                                                                2
                                                                3
    LL pp=1, tmp=0;
    for(int i=0;i<n;i++) pp=pp*p[i];</pre>
                                                                5
    for(int i=0;i<n;i++)</pre>
                                                                6
         LL m=pp/p[i], x, y;
                                                                8
        ex_gcd(m,p[i],x,y);
                                                                9
         x=(x%p[i]+p[i])%p[i];
                                                                10
         tmp=(tmp+(a[i]*m%pp*x)%pp)%pp;//注意overflow
                                                                11
                                                                12
    return tmp;
                                                                13
```

1.4 EulerFunction

1.5 EulerSieve

```
//欧拉筛法
2
    void get_prime(int n)
3
4
      memset(np, 0, sizeof(np));
5
      phi[1]=1;
6
      for (int i=2;i<=n;i++)</pre>
7
8
         if (!np[i])
9
10
           prime.push_back(i);
11
           phi[i]=i-1;
12
         for (int j=0;j<prime.size()&&prime[j]*i<=n;j++)</pre>
13
14
15
           np[prime[j]*i]=1;
16
           if (i%prime[j]==0)
17
18
             phi[i*prime[j]]=phi[i]*prime[j];
19
             break;
20
21
           else
22
             phi[i*prime[j]]=phi[i]*(prime[j]-1);
23
24
      }
    }
25
```

1.6 ExtendedEuclidean

```
//扩展欧几里得
2
    LL ex_gcd(LL a, LL b, LL &x, LL &y)
 3
        if (!b)
 4
 5
 6
            x=1, y=0;
7
            return a;
 8
 9
        else
10
            LL d=ex_gcd(b, a \% b, y, x);
12
            v=a/b*x;
13
             return d;
14
15
    //求乘法逆元
16
17
    LL inv(LL a, LL p)
18
19
        LL gcd, x, y;
        gcd=ex\_gcd(a, p, x, y);
20
21
        if (gcd==1) return (x+p)%p;
22
        else return -1;
    }
23
```

1.7 FFT

```
#define PI acos(-1.0)//acosl(-1.0)
    typedef double LD;//long double
 5
    typedef long long LL;
    //typedef complex<LD> cpx;
    struct cpx
8
      LD x, y;
 9
      cpx(){}
10
      cpx(LD x, LD y):x(x),y(y){}
12
13
    cpx operator +(cpx a, cpx b)
14
15
      return cpx(a.x+b.x, a.y+b.y);
16
    cpx operator -(cpx a, cpx b)
17
18
19
      return cpx(a.x-b.x, a.y-b.y);
20
21
    cpx operator *(cpx a, cpx b)
22
      return cpx(a.x*b.x-a.y*b.y, a.x*b.y+a.y*b.x);
23
24
25
   int rev(int x, int n)
26
```

```
27
    int tmp=0:
                                                                    28
    for (int i=n>>1;i;i>>=1,x>>=1)
                                                                    29
         tmp=tmp<<1|x&1;
                                                                    30
    return tmp:
                                                                    31
                                                                    32
                                                                    33
void fft(cpx *a, int n, int flag)
                                                                    34
                                                                    35
  for (int i=0, j=i;i<n;i++, j=rev(i, n))</pre>
                                                                    36
    if (i<j) swap(a[i], a[j]);</pre>
                                                                    37
  for (int k=1;k<n;k<<=1)</pre>
                                                                    38
                                                                    39
    cpx wn(cos(PI/k), flag*sin(PI/k));
                                                                    40
    //cpx wn(cosl(PI/i), flag*sinl(PI/i));
                                                                    41
    cpx w(1, 0);
                                                                    42
    for (int i=0;i<k;i++,w=w*wn)</pre>
                                                                    43
      for (int j=i; j<n; j+=(k<<1))
                                                                    44
                                                                    45
         cpx x=a[j], y=w*a[j|k];
                                                                    46
         a[j]=x+y;
                                                                    47
         a[j|k]=x-y;
                                                                    48
                                                                    49
                                                                    50
  if (flag==-1)
                                                                    51
    for (int i=0;i<n;i++)</pre>
                                                                    52
      a[i].x/=n, a[i].y/=n;
                                                                    53
                                                                    54
                                                                    55
cpx A[maxn], B[maxn];
                                                                    56
int a[maxn];
                                                                    57
int n:
                                                                    58
void roll(int *a, int *b, int *c, int n, int m)
                                                                    59
                                                                    60
  int num=1:
                                                                    61
  while (num<n+m) num<<=1;//move out if slow
                                                                    62
  for (int i=0;i<num;i++) A[i]=(i<n)?cpx(a[i],0):cpx(0,0);</pre>
                                                                    63
  for (int i=0;i<num;i++) B[i]=(i<m)?cpx(b[i],0):cpx(0,0);</pre>
                                                                    64
  fft(A, num, 1);
                                                                    65
  fft(B, num, 1);
                                                                    66
  for (int i=0;i<num;i++) A[i]=A[i]*B[i];</pre>
                                                                    67
  fft(A, num, -1);
                                                                    68
  for (int i=0;i<num;i++) c[i]=(LL)(A[i].x+0.5)%mod;</pre>
                                                                    69
                                                                    70
```

1.8 FFT-2

```
void fft(cpx *a, int n, int flag)
                                                                   2
    if (flag==-1)
                                                                   3
       for (int i=1;i<n/2;i++) swap(a[i], a[n-i]);</pre>
         for (int i=0;i<n;i++) a[i].x/=n, a[i].y/=n;</pre>
                                                                   6
                                                                   7
    cpx wn(cos(2*PI/n), sin(2*PI/n));
    for (int s=n>>1;s;s>>=1, wn=wn*wn)
                                                                   9
                                                                   10
       cpx w(1,0);
                                                                   11
         for (int k=0; k<s; k++, w=w*wn)
                                                                   12
       for (int i=k; i<n; i+=s<<1)</pre>
                                                                   13
                                                                   14
                 cpx x=a[i], y=a[i|s];
                                                                   15
         a[i]=x+y;
                                                                   16
         a[i|s]=(x-y)*w;
                                                                   17
                                                                   18
                                                                   19
    for (int i=1, j=0;i<n;i++,j=0)</pre>
                                                                   20
                                                                   21
    for (int s=i, k=n>>1; k; s>>=1, k>>=1)
                                                                   22
       j=j<<1|s&1;
                                                                   23
    if (i<j) swap(a[i], a[j]);</pre>
                                                                   25
                                                                   26
inline void ntt(LL a[], int n, int flag)
                                                                   27
                                                                   28
  LL wn=exp(G, (mod-1)/n, mod), w=1;
                                                                   29
  if (flag==-1)
                                                                   30
                                                                   31
    for (int i=1;i<n/2;i++) swap(a[i], a[n-i]);</pre>
                                                                   32
    LL inv=exp(n, mod—2, mod);
                                                                   33
    for (int i=0;i<n;i++) a[i]=a[i]*inv\mod;</pre>
                                                                   34
                                                                   35
  for (int s=n>>1; s; s=s>>1, wn=wn*wn%mod, w=1)
                                                                   36
```

```
37
         for (int k=0; k<s; k++, w=w*wn%mod)</pre>
38
           for (int i=k; i<n; i+=s<<1)</pre>
39
40
              LL x=a[i], y=a[i|s]%mod;
41
             a[i]=(x+y) \mod;
42
              a[i|s]=(x-y+mod)%mod*w%mod;
43
      for (int i=1, j=0;i<n;i++,j=0)</pre>
44
45
         for (int s=i, k=n>>1; k; s>>=1, k>>=1)
46
47
           j=j<<1|s&1;
48
         if (i<j) swap(a[i], a[j]);</pre>
49
    }
50
```

1.9 GCD

HIT

```
//GCD
1
   long long gcd(long long a, long long b)
2
3
       return (b==0?a:gcd(b, a%b));
5
   }
```

GaussElimination 1.10

```
//高斯消元
1
2
    void gauss(int n)
3
         int k=1:
4
5
         for (int i=1;i<=n;i++)</pre>
6
7
             int p=0;
8
             for (int j=k;j<=n;j++)</pre>
                  if (fab(a[j][i])>eps) {
                                                p=j; break;
                                                                 }
9
10
             if (!p) continue;
11
             for (int l=1;l<=n+1;l++) swap(a[p][l],a[k][l]);</pre>
             for (int j=k+1; j \le n; j++)
12
             {
14
                  lb\ rate=a[j][i]/a[k][i];
15
                  for (int l=1;l<=n+1;l++)</pre>
                      a[j][1]=a[k][1]*rate;
17
             k++:
18
19
         for (int i=n;i;i—)
20
21
             v[i]=a[i][n+1];
22
23
             for (int j=i+1; j<=n; j++)
24
                  v[i]=v[j]*a[i][j];
25
             v[i]/=a[i][i];
26
         }
27
    }
```

LucasTheory 1.11

```
//Lucas定理(求组合数取模,注意p<=10^5)
 1
2
    LL com(LL n, LL m, LL p)
3
      if (m>n) return 0;
 5
      LL ans=1;
      for (int i=1;i<=m;i++)</pre>
 8
         ans=ans*(n-i+1)%p;
         ans=ans*exp(i, p-2, p)%p;
10
      return ans;
11
    }
12
13
    LL lucas(LL n, LL m, LL p)
15
16
      if (m==0) return 1;
17
      return (com(n%p, m%p, p)*lucas(n/p, m/p, p))%p;
    }
18
19
20
    LL get_ans(LL n, LL m)
21
22
      return lucas(n+m, n, oo);
23
    }
```

1.12MatrixQuickPower

```
//矩阵乘法&快速幂(ms<=500)
const int ms=5:
struct matrix
  LL e[ms][ms];
                                                                   6
  matrix()
    memset(e, 0, sizeof(e));
                                                                   9
  matrix(int x)
                                                                   10
                                                                   11
    memset(e, 0, sizeof(e));
                                                                   12
    for (int i=0;i<ms;i++) e[i][i]=x;</pre>
                                                                   13
                                                                   14
  void clear()
                                                                   15
                                                                   16
    memset(e, 0, sizeof(e));
                                                                   17
                                                                   18
  friend matrix operator *(matrix a, matrix b)
                                                                   19
                                                                   20
    matrix c;
                                                                   21
    for (int k=0;k<ms;k++)</pre>
                                                                   22
      for (int i=0;i<ms;i++)</pre>
                                                                   23
         for (int j=0;j<ms;j++)</pre>
                                                                   24
           (c.e[i][j]+=a.e[i][k]*b.e[k][j]‰o)%=oo;
                                                                   25
    return c;
                                                                   26
  friend matrix operator ^(matrix e, LL k)
                                                                   28
                                                                   29
    matrix tmp=matrix(1);
                                                                   30
    while (k)
                                                                   31
                                                                   32
       if (k&1) tmp=tmp*e;
                                                                   33
      k=k>>1;
                                                                   34
      e=e*e;
                                                                   35
                                                                   36
    return tmp;
                                                                   37
  }
                                                                   38
};
                                                                   39
```

MillerRabin PollardRho

```
//miller_rabin+pollard_rho分解质因数
const int S=20;//随机算法判定次数,S越大,判错概率越小
//计算 (a*b)‰.
                 a,b都是long long的数,直接相乘可能溢出的
                                                          3
// a,b,c <2^63
long long mult_mod(long long a,long long b,long long c)
                                                          5
    a%=c;
    h%=c:
                                                          8
    long long ret=0;
    while(b)
                                                          10
                                                          11
        if(b&1){ret+=a;ret%=c;}
        a <<=1;
                                                          13
        if(a>=c)a%=c;
        b>>=1;
                                                          15
                                                          16
    return ret;
                                                          18
long long pow_mod(long long x,long long n,long long mod)//
x^n%c
    if(n==1)return x%mod;
    x\%=mod;
    long long tmp=x;
    long long ret=1;
                                                          26
    while(n)
        if(n&1) ret=mult_mod(ret,tmp,mod);
                                                          29
        tmp=mult_mod(tmp,tmp,mod);
                                                          30
        n>>=1;
                                                          31
    return ret;
                                                          33
                                                          34
//以a为基,n-1=x*2^t
                        a^(n-1)=1(mod n) 验证n是不是合数
                                                          36
//一定是合数返回true, 不一定返回false
bool check(long long a, long long n, long long x, long long t
```

6

14

17

19

20

21

22

23

24

25

27

28

32

35

37

38

74

```
39
     {
         long long ret=pow_mod(a,x,n);
 40
 41
         long long last=ret;
 42
         for(int i=1;i<=t;i++)</pre>
 43
             ret=mult_mod(ret,ret,n);
 44
             if(ret==1&&last!=1&&last!=n-1) return true;//合数
 45
 46
 47
         if(ret!=1) return true;
 48
         return false;
 49
 50
     }
 51
     // Miller_Rabin()算法素数判定
     //是素数返回true.(可能是伪素数,但概率极小)
 53
     //合数返回false;
 54
 56
     bool Miller_Rabin(long long n)
 57
 58
         if(n<2)return false;</pre>
         if(n==2)return true;
 59
 60
         if((n&1)==0) return false;//偶数
         long long x=n-1;
 61
 62
         long long t=0;
         while((x&1)==0){x>>=1;t++;}
 63
         for(int i=0;i<S;i++)</pre>
 64
 65
 66
             long long a=rand()%(n-1)+1;//rand()需要stdlib.h头
             if(check(a,n,x,t))
 67
                 return false;//合数
 68
 69
         return true;
 70
     }
 71
 72
     //pollard_rho 算法进行质因数分解
     long long factor[100];//质因数分解结果(刚返回时是无序的)
 74
 75
     int tol;//质因数的个数。数组小标从0开始
 76
     long long gcd(long long a, long long b)
 77
 78
         return (b==1?a:gcd(b, a%b));
 79
 80
     }
 81
     long long Pollard_rho(long long x,long long c)
 82
 83
 84
         long long i=1, k=2;
         long long x0=rand()%x;
 85
         long long y=x0;
 86
         while(1)
 87
 88
             x0=(mult_mod(x0,x0,x)+c)%x;
 90
 91
             long long d=gcd(y-x0,x);
             if(d!=1&&d!=x) return d;
 92
             if(y==x0) return x;
 93
             if(i==k){y=x0;k+=k;}
 94
 95
 96
     //对n进行素因子分解
 97
     void findfac(long long n)
 98
 99
100
         if(Miller_Rabin(n))//素数
101
             factor[tol++]=n;
102
103
             return;
104
105
         long long p=n;
         while(p>=n)p=Pollard_rho(p,rand()%(n-1)+1);
106
107
         findfac(p);
108
         findfac(n/p);
109
110
     int main()
111
112
         //srand(time(NULL));//需要time.h头文件//POJ上G++不能加
114
         long long n;
         while(scanf("%I64d", &n)!=E0F)
115
116
117
             tol=0;
118
             for(int i=0;i<tol;i++)printf("%I64d ",factor[i]);</pre>
119
```

```
printf("\n");
                                                             120
    if(Miller_Rabin(n))printf("Yes\n");
                                                             121
    else printf("No\n");
                                                             122
                                                             123
return 0;
                                                             124
                                                             125
```

1.14 MobiusInversion

```
//莫比乌斯反演
void get_prime(int n)
    memset(np, 0, sizeof(np));
    tot=0;
    mu[1]=1;
    for (int i=2;i<=n;i++)</pre>
         if (!np[i])
                                                                 10
             prime[++tot]=i;
             mu[i]=-1;
                                                                 12
                                                                 13
         for (int j=1;j<=tot && i*prime[j]<=n;j++)</pre>
                                                                 14
                                                                 15
             np[i*prime[j]]=1;
                                                                  16
             if (i%prime[j]) mu[i*prime[j]]=-mu[i];
                                                                 17
             else
                                                                 18
                                                                  19
                 mu[i*prime[j]]=0;
                                                                 20
                 break;
                                                                 21
                                                                  22
         }
                                                                 23
    }
                                                                 24
                                                                  25
//求与序列中第i个数互质数的个数
                                                                 26
int main()
                                                                  27
                                                                  28
    for (int i=1;i<=100000;i++)</pre>
                                                                 29
         for (int j=i; j<=100000; j+=i)</pre>
                                                                 30
             sum[i]+=v[j];
                                                                 31
    for (int i=1;i<=100000;i++)</pre>
                                                                 32
                                                                 33
         for (int j=i;j<=100000;j+=i) f[j]+=sum[i]*mu[i];</pre>
                                                                 34
         if (i==1) f[i]—;
                                                                  35
                                                                 36
                                                                 37
                                                                  38
                                                                 39
//己知f[n]=(求和d|n)g(d),求g nlogn
                                                                  40
for (int i = 1; i <= n; ++i)
for (int j = i + i; j <= n; j += i)</pre>
                                                                  41
                                                                  42
    f[j] -= f[i];
                                                                  44
//f[n]=(求和d|n)g(d), 己知g, 求f
                                                                  45
for (int i = n; i >= 1; —i)
                                                                  46
  for (int j = i + i; j \le n; j += i)
                                                                  47
    f[j] += f[i];
                                                                  48
                                                                  49
//己知f[n]=(求和n|d)g(d),求g nlogn
                                                                 50
for (int i = n; i >= 1; --i)
  for (int j = i + i; j \le n; j += i)
                                                                 52
    f[i] = f[j];
                                                                 53
                                                                  54
//f[n]=(求和n|d)g(d), 已知g, 求f
                                                                 55
for (int i = 1; i <= n; ++i)</pre>
                                                                 56
  for (int j = i + i; j \le n; j += i)
                                                                 57
    f[i] += f[j];
                                                                 58
//f[s]存原来的元素,之后f[s]存子集所有元素和
                                                                 60
for (int i = 0; i < n; i++)
                                                                 61
    for (int s = 0; s < (1 << n); s++)
         if (s >> i & 1)
                                                                 63
             f[s] += f[s \land 1 << i];
                                                                  64
                                                                 65
                                                                 66
                                                                  67
f[s]存子集所有元素和,之后f[s]存原来的元素
                                                                 68
                                                                 69
for (int i = 0; i < n; i++)</pre>
                                                                 70
    for (int s = 0; s < (1 << n); s++)
                                                                 71
        if (s >> i & 1)
                                                                 72
             f[s] = f[s \land 1 << i];
                                                                 73
```

```
数论卷积nlogn
                    第1-n的h[x]=(求和d|x)(f[d]*g[x/d]) 已知f,
    g的1—n
76
77
    int f[MAXN],g[MAXN],h[MAXN]={0};
78
    void calc(int n)
79
        for (int i=1;i*i<=n;i++)</pre>
80
81
            for (int j=i;i*j<=n;j++)</pre>
                if(j==i)h[i*j]+=f[i]*g[i];
82
83
                else h[i*j]+=f[i]*g[j]+f[j]*g[i];
```

1.15 MulInverse

1.16 NTT

```
//NTT
    #define G 3
    typedef long long LL;
    const LL mod=998244353;
    inline LL exp(LL a, LL b, LL p)
 6
      a%=p;
      LL tmp=1;
 8
 9
      while (b)
10
         if (b&1) tmp=(tmp*a)%p;
11
         a=a*a%p;
         b=b/2:
13
14
15
      return tmp;
16
17
    inline int rev(int x, int n)
18
19
         int tmp=0:
20
             (int i=n>>1;i;i>>=1,x>>=1)
             tmp=tmp<<1|x&1;
21
22
         return tmp;
23
    inline void ntt(LL *a, int n, int flag)
24
25
26
      for (int i=0, j=i;i<n;i++, j=rev(i, n))</pre>
         if (i<j) swap(a[i], a[j]);</pre>
27
       for (int k=1; k<n; k<<=1)</pre>
28
29
         LL wn=exp(G, (mod-1)/(k<<1), mod), w=1;
30
         if (flag==-1) wn=exp(wn, mod-2, mod);
31
         for (int i=0;i<k;i++,w=w*wn%mod)</pre>
32
33
           for (int j=i;j<n;j+=(k<<1))</pre>
34
35
             LL x=a[j], y=w*a[j|k]%mod;
             a[j]=(x+y)\mod;
36
             a[j|k]=(x-y+mod)%mod;
37
38
39
      if (flag==-1)
40
41
42
         LL inv=exp(n, mod-2, mod);
         for (int i=0;i<n;i++) a[i]=a[i]*inv/mod;</pre>
43
44
      }
45
    }
46
47
    inline void roll(LL *a, LL *b, LL *c, int n, int m)
48
49
       int num=1;
      while (num<n+m) num<<=1;</pre>
50
       for (int i=0;i<num;i++) a[i]=(i<n)?a[i]:0;</pre>
51
       for (int i=0;i<num;i++) b[i]=(i<m)?b[i]:0;</pre>
52
53
      ntt(a, num, 1);
54
      ntt(b, num, 1);
55
       for (int i=0;i<num;i++) c[i]=a[i]*b[i]%mod;</pre>
56
      ntt(c, num, -1);
57
58
    inline void solve(int 1, int k)
```

```
if (k==0)
                                                                    61
                                                                    62
    f[1]=f[1]*exp(1, mod-2, mod)%mod;
                                                                   63
    return ;
                                                                   64
                                                                   65
  int r=l+(1<<k)-1, mid=(l+r)>>1;
                                                                   66
  solve(1, k-1);
                                                                   67
  memset(a, 0, sizeof(*a)<<k);</pre>
                                                                    68
  //cout<<k<<endl;
                                                                   69
  for (int i=1;i<=mid;i++) a[i-l]=f[i];</pre>
                                                                   70
  ntt(a, 1<<k, 1);
                                                                    71
  for (int i=0;i<1<<k;i++) (a[i]*=b[k][i])%=mod;</pre>
                                                                   72
  ntt(a, 1<<k, -1);
                                                                   73
  for (int i=mid+1;i<=r;i++)</pre>
                                                                    74
    (f[i]+=a[i-l-1])\%=mod;
                                                                   75
  solve(mid+1, k-1);
                                                                   76
                                                                    77
//求原根(MOD=P * 2^k + 1)
                                                                   78
int g = 2;
while (\exp(g, (MOD - 1) / 2, MOD) == 1 || \exp(g, (MOD - 1))
                                                                   80
 / 479, MOD) == 1) {
                                                                   81
}
                                                                   82
```

1.17 QuadraticField

```
//二次域运算
typedef long long LL;
const LL w=5;
const LL 00=1e9+7;
struct PP
  PP(){}
                                                               8
  PP(LL x, LL y):x(x\%00), y(y\%00){}
PP operator +(PP a, PP b){
                                                               11
  return PP((a.x+b.x)%00, (a.y+b.y)%00);
                                                               12
                                                               13
PP operator -(PP a, PP b){
                                                               14
  return PP((a.x-b.x+00)%00, (a.y-b.y+00)%00);
                                                               15
                                                               16
                                                               17
PP operator *(PP a, PP b){
  return PP((a.x*b.x%oo+a.y*b.y%oo*w%oo)%oo, (a.x*b.y%oo+a
                                                               18
  .y*b.x%oo)%oo);
                                                               19
void write(PP a){
                                                               20
  cout<<a.x<<' '<<a.y<<endl;
                                                               21
//快速幂
                                                               23
PP exp(PP a, LL b, LL p){
                                                               24
  PP tmp=PP(1, 0);
  while (b){
                                                               26
    if (b&1) tmp=tmp*a;
                                                               27
    b=b>>1;
                                                               29
                                                               30
  return tmp:
                                                               31
                                                               32
//二次域逆元
                                                               33
PP inv(PP a)
                                                               34
                                                               35
  PP r=exp(PP((a.x*a.x%oo-a.y*a.y%oo*w%oo+oo)%oo, 0), oo
                                                               36
  -2, 00)
  return r*PP(a.x, oo-a.y);
                                                               37
                                                               38
```

1.18 QuickPower

```
13 |}
```

1.19 SlowMul

```
//慢速乘
 1
    llg mul(llg a,llg b,llg p)
 2
 3
 4
      a%=p,b%=p;
 5
      11g tmp=0;
 6
      while (b)
         if (b&1) tmp=(tmp+a)%p;
 8
9
         a=(a+a)%p;
10
         b/=2;
11
12
      return tmp;
    }
```

1.20 Trisection

```
//三分求最小值
2
    double l=mi,r=mx,ans:
 3
    while (dcmp(1-r) \le 0)
 5
      double lmid=l+(r-1)/3.0;
      double rmid=r-(r-1)/3.0;
 6
      if (dcmp(get_ans(lmid)-get_ans(rmid))<=0)</pre>
 8
 9
         r=rmid-eps;
        ans=get_ans(lmid);
10
11
12
      else
13
         l=lmid+eps;
15
      }
    }
16
```

1.21 XorGauss

```
//高斯消元XOR
1
2
    int gauss(int n, int m)
3
        int k=1:
5
        for (int i=1;i<=m;i++)</pre>
6
            int p=0;
            for (int j=k;j<=n;j++) if (a[j][i]) {</pre>
8
            break:
            if (!p) continue;
9
10
            for (int l=1;l<=m;l++) swap(a[p][l],a[k][l]);//有
             常数的话m+1
11
            for (int j=k+1;j<=n;j++)</pre>
12
              if (a[j][i]==1)
                 for (int l=1;l<=m;l++)//有常数的话m+1
13
                       a[j][1]=a[j][1]^a[k][1];
14
15
16
        return k-1;//返回线性无关方程个数
17
18
    }
```

2 DP

2.1 DigitDP

```
//数位动归
 2
    int get_ans(int x)
 3
      if (!x) return 1;
      int len=0, tmp=0;
 5
 6
      while (x)
        b[++len]=x%10;
8
 9
        x=x/10;
10
11
      b[1]++;
12
      b[len+1]=0;
13
      b[len+21=0:
14
      for (int i=len; i>=1; i—)//固定第i+1位之前的位
15
        if (b[i+1]==4) break;
16
```

```
      if (b[i+2]==6&&b[i+1]==2) break;
      17

      for (int j=0;j<b[i];j++)//枚举第i位</td>
      18

      {
      19

      if ((b[i+1]==6&&j==2)||(j==4)) continue;
      20

      tmp+=f[i-1][0]+f[i-1][1]*(j!=6);
      21

      }
      22

      return tmp;
      24

      }
      25
```

2.2 QuadrilateralInequality

```
for (int len=2;len<=n;len++)</pre>
                                                                    3
  for (int i=1;i<=n-len+1;i++)</pre>
                                                                    5
    int j=i+len-1;
    f[i][j]=inf;
    for (int k=K[i][j-1];k<=min(j-1,K[i+1][j]);k++)</pre>
                                                                    8
       if (f[i][j]>f[i][k]+f[k+1][j]+x[k+1]–x[i]+y[k]–y[j])
                                                                    10
                                                                    11
         f[i][j]=f[i][k]+f[k+1][j]+x[k+1]-x[i]+y[k]-y[j];
                                                                    12
         K[i][j]=k;
                                                                    13
                                                                    14
    }
                                                                    15
  }
                                                                    16
                                                                    17
```

2.3 StateCompressionDP

```
#define inf 1000000007
#define maxn 11
#define maxm 1050
using namespace std;
                                                                   8
int n, m, mb;
long long f[maxn][maxn][88][maxm];
int bit[maxm][maxn];
                                                                   10
                                                                    11
bool legal(int x, int y,int mask)
                                                                   12
                                                                   13
    if (y==1)
                                                                    14
                                                                   15
         if (bit[mask][n]&bit[mask][n+1]) return 0;
                                                                    16
         if (bit[mask][n-1]&bit[mask][n+1]) return 0;
                                                                    17
    }
                                                                   18
    else
                                                                    19
                                                                    20
    {
         if (bit[mask][y-1]&bit[mask][n+1]) return 0;
                                                                   21
         if (bit[mask][y-2]&bit[mask][n+1]) return 0;
                                                                    22
                                                                    23
    for (int i=1;i<n;i++) if (bit[mask][i]&bit[mask][i+1])</pre>
                                                                    24
     return 0;
    return 1;
                                                                    25
                                                                    26
                                                                    27
int main()
                                                                   28
                                                                    29
{
    scanf("%d%d",&n,&m);
                                                                   30
    mb=(1<<(n+1))-1;
                                                                   31
    memset(bit, 0, sizeof(bit));
                                                                    32
    for (int mask=0;mask<=mb;mask++)</pre>
                                                                   33
    {
                                                                    34
         int tmp=mask;
                                                                    35
        for (int i=1;i<=n+1;i++)</pre>
                                                                   36
                                                                    37
         {
             bit[mask][i]=tmp%2;
                                                                    38
             tmp/=2;
                                                                   39
                                                                    40
                                                                    41
    memset(f, 0, sizeof(f));
                                                                    42
    f[0][n][0][0]=1;
                                                                    43
    for (int i=1;i<=n;i++)</pre>
                                                                    44
         for (int j=1;j<=n;j++)</pre>
                                                                    45
             for (int k=0; k<=m; k++)
                                                                   46
                 for (int mask=0;mask<=mb;mask++)</pre>
                                                                    47
                      if (legal(i,j,mask))
                                                                    48
                                                                    49
```

```
50
                              int tmp=1<<(j-1);</pre>
                              int ost=mask&(~tmp), nst=mask|tmp;
51
52
                              if (((mask>>(j-1))&1)!=((mask>>n)
                              &1)) ost^=(1<<n), nst^=(1<<n);
                              if (j==1)
53
                                   f[i][j][k][ost]+=f[i-1][n][k][
55
                                   mask];
                                   if (k && !bit[mask][j] && !bit
                                   [mask][j+1] && !bit[mask][j
                                       f[i][j][k][nst]+=f[i-1][n
57
                                       [k-1][mask];
58
                              else
59
60
                                   f[i][j][k][ost]+=f[i][j-1][k][
                                   if (k && !bit[mask][j] && !bit
62
                                   [mask][j+1] \&\& !bit[mask][j-1]
                                    && !bit[mask][n+1])
63
                                       f[i][j][k][nst]+=f[i][j
                                       -1][k-1][mask];
64
                              }
         long long ans=0;
66
67
         for (int mask=0;mask<=mb;mask++)</pre>
68
             if (legal(n+1,1,mask))
69
70
                 ans+=f[n][n][m][mask];
71
         printf("%lld\n",ans);
72
73
         return 0;
74
75
    Kings
```

2.4 Tree_Divide_Point

```
//Poj1471 Tree
    //点分治
    //每次分治先选择中心点为根,然后删去根
    //分解为若干子树,递归处理。
    struct edge
 6
      int s, t, val;
      edge(){}
      edge(int s, int t, int val):s(s),t(t),val(val){}
9
    };
11
12
    vector<edge> e[maxn];
    vector<int> q;
    int vis[maxn], size[maxn], mx_size[maxn], dep[maxn];
14
15
    int n, m, rot, ans;
    void chose(int x, int fa, int num)//递归寻找重心
17
18
      size[x]=1; mx_size[x]=0;
19
20
      int tot=e[x].size();
21
      for (int i=0;i<tot;i++)</pre>
22
        int y=e[x][i].t;
23
        if (!vis[y] && y!=fa)
24
25
26
          chose(y, x, num);
          size[x]+=size[y];
27
          mx_size[x]=max(mx_size[x], size[y]);
28
29
30
      mx_size[x]=max(mx_size[x], num-size[x]);
31
      if (mx_size[x]<mx_size[rot]) rot=x;</pre>
33
34
35
    int chose_rot(int x, int num)
36
37
      chose(x, 0, num);
38
39
40
      return rot;
    }
41
42
    void get_deep(int x, int fa)//递归求每个点的深度
43
```

```
q.push_back(dep[x]);
  int tot=e[x].size();
                                                                  46
  for (int i=0;i<tot;i++)</pre>
                                                                  47
                                                                  48
    int y=e[x][i].t;
                                                                  49
    if (!vis[y] && y!=fa)
                                                                  50
                                                                  51
      dep[y]=dep[x]+e[x][i].val;
                                                                  52
      get_deep(y, x);
                                                                  53
                                                                  54
                                                                  55
                                                                  56
                                                                  57
int get_ans(int x, int h)//计算当前结点为根的答案
                                                                  59
  q.clear();
                                                                  60
  dep[x]=h;
  get_deep(x, 0);
                                                                  62
  sort(q.begin(), q.end());
                                                                  63
  int l=0, r=q.size()-1, tmp=0;
                                                                  64
  while (1<r)
                                                                  65
                                                                  66
    if (q[1]+q[r]<=m) tmp+=(r-1), 1++;</pre>
                                                                  67
    else r—;
                                                                  68
                                                                  69
  return tmp;
                                                                  70
                                                                  71
                                                                  72
void solve(int x)//分治处理x为根的子树
                                                                  73
                                                                  74
  vis[x]=1;
                                                                  75
  ans+=get_ans(x, 0);
                                                                  76
  int tot=e[x].size();
                                                                  77
  for (int i=0;i<tot;i++)</pre>
                                                                  78
                                                                  79
    int y=e[x][i].t;
    if (!vis[y])
                                                                  81
                                                                  82
      ans-get_ans(y, dep[y]);
                                                                  83
      solve(chose_rot(y, size[y]));
                                                                  84
                                                                  85
  }
                                                                  86
                                                                  87
void init()
                                                                  89
                                                                  90
  for (int i=1;i<=n;i++) e[i].clear();</pre>
                                                                  91
  memset(vis, 0, sizeof(vis));
                                                                  92
  ans=0;
                                                                  93
  mx_size[0]=inf;
                                                                  94
                                                                  95
int main()
                                                                  97
                                                                  98
  while (scanf("%d%d",&n,&m) && (n+m)!=0)
                                                                  99
                                                                  100
    int x,y,z;
                                                                  101
    init();
                                                                  102
    for (int i=1;i<n;i++)</pre>
                                                                  103
                                                                  104
      scanf("%d%d%d",&x,&y,&z);
                                                                  105
      e[x].push_back(edge(x,y,z));
                                                                  106
      e[y].push_back(edge(y,x,z));
                                                                  107
                                                                  108
    solve(chose_rot(1, n));
                                                                  109
    printf("%d\n",ans);
                                                                  110
                                                                  111
  return 0;
                                                                  112
```

3 String

3.1 AC_Automaton

```
9
         if (ch[now][w]==-1) ch[now][w]=++num;
10
         now=ch[now][w];
11
12
13
    void build()
14
         queue<int> q;
15
         memset(fail,0,sizeof(fail));
16
         for (int w=0;w<26;w++)
17
             if (ch[rot][w]==-1)
18
19
                 ch[0][w]=rot;
20
                 q.push(ch[0][w]);
21
         while (!q.empty())
22
23
24
             int x=q.front(); q.pop();
             for (int w=0;w<26;w++)
25
26
                 if (ch[x][w]==-1)
27
                      ch[x][w]=ch[fail[x]][w];
28
29
                 {
30
                      int y=ch[x][w];
                      fail[y]=ch[fail[x]][w];
31
32
                      q.push(y);
33
34
        }
```

3.2 KMP

HIT

```
//KMP
    int kmp(char *t)
2
3
      int n=strlen(t);
      memset(nxt,-1,sizeof(nxt));
5
6
      int j=-1;
      for (int i=1;i<n;i++)</pre>
8
9
        while (j!=-1&&t[i]!=t[j+1]) j=nxt[j];
        if (t[i]==t[j+1]) j++;
10
11
        nxt[i]=j;
      return n-1-nxt[n-1];//n-next[n-1]是最小循环节
13
14
    void match(char *s, char *t)
15
16
      int ls=strlen(s), lt=strlen(t);
17
      int j=-1;
18
19
      for (int i=0;i<ls;i++)</pre>
20
        while (j!=-1 && s[i]!=t[j+1]) j=nxt[j];
21
22
        if (s[i]==t[j+1]) j++;
23
    }
24
```

3.3 StringHash

3.4 Suffix_Automaton

```
//后缀自动机
2
    struct node
3
         node *pre, *ch[27];
5
         int mx;
6
         node(){
           mx=0;
           pre=NULL;
8
           for (int i=0;i<=26;i++) ch[i]=0;</pre>
10
11
    }*rot, *now, *que[maxn];
12
    inline void ins(int w)
```

```
node *p=now, *np=&rot[++num];
                                                                        15
  np \rightarrow mx = p \rightarrow mx + 1;
                                                                        16
  while (p!=NULL && p->ch[w]==NULL) p->ch[w]=np,p=p->pre;
                                                                        17
  if (p==NULL) np->pre=rot;
                                                                        18
                                                                        19
                                                                        20
    node *q=p->ch[w];
                                                                        21
     if (q->mx==p->mx+1) np->pre=q;
                                                                        22
    else
                                                                        23
                                                                        24
       node *nq=&rot[++num];
                                                                        25
       *nq=*q;
                                                                        26
       nq->mx=p->mx+1;
                                                                        27
                                                                        28
       np->pre=q->pre=nq;
       while (p!=NULL \&\& p\rightarrow ch[w]==q) p\rightarrow ch[w]=nq, p=p\rightarrow pre;
                                                                        29
                                                                        30
                                                                        31
  now=np;
                                                                        32
                                                                        33
                                                                        34
inline void build(char *s)
                                                                        35
                                                                        36
  rot=new node[maxn];
                                                                        37
  now=&rot[num=0];
                                                                        38
  int n=strlen(s):
                                                                        39
  for (int i=0;i<n;i++) ins(s[i]-'a');</pre>
                                                                        40
                                                                        41
                                                                        42
void clear()
                                                                        43
                                                                        44
  delete [] rot;
                                                                        45
                                                                        46
                                                                        47
inline void tong_sort()
                                                                        48
  for (int i=1;i<n;i++) tong[i]=0;</pre>
                                                                        50
  for (int i=0;i<=num;i++) tong[rot[i].mx]++;</pre>
                                                                        51
  for (int i=1;i<=n;i++) tong[i]+=tong[i-1];</pre>
                                                                        52
  for (int i=num;i>=0;i—) que[—tong[rot[i].mx]]=&rot[i];
                                                                        53
                                                                        54
```

f 4 DataStruct

目录

4.1 BIT 2Dimensions

```
struct bit
    int b[maxn][maxn];
    void add(int x,int y,int z)
         for (int i=x;i<=n;i+=(i&-i))</pre>
                                                                 6
             for (int j=y;j<=m;j+=(j&-j))</pre>
                 b[i][j]+=z;
    int ask(int x,int y)
                                                                 10
                                                                 11
         int tmp=0:
                                                                 12
             (int i=x;i>0;i-=(i\&-i))
             for (int j=y;j>0;j-=(j&-j))
                                                                 14
                 tmp+=b[i][j];
                                                                 15
         return tmp;
                                                                 16
                                                                 17
}s,t,1,r;
                                                                 18
                                                                 19
void ins(int x1,int y1,int x2,int y2,int z)
                                                                 20
                                                                 21
    s.add(x1,y1,z); s.add(x2+1,y1,-z);
                                                                 22
    s.add(x1,y2+1,-z); s.add(x2+1,y2+1,z);
                                                                 23
    1.add(x1,y1,z*x1); 1.add(x2+1,y1,-z*(x2+1));
                                                                 25
    1.add(x1,y2+1,-z*x1); 1.add(x2+1,y2+1,z*(x2+1));
                                                                 26
                                                                 27
    r.add(x1,y1,z*y1); r.add(x2+1,y1,-z*y1);
                                                                 28
    r.add(x1, y2+1, -z^*(y2+1)); r.add(x2+1, y2+1, z^*(y2+1));
                                                                 29
                                                                 30
    t.add(x1,y1,z*x1*y1); t.add(x2+1,y1,-z*(x2+1)*y1);
                                                                 31
    t.add(x1,y2+1,-z*x1*(y2+1)); t.add(x2+1,y2+1,z*(x2+1))
                                                                 32
     *(y2+1));
                                                                 33
                                                                 34
int query(int x1,int y1,int x2,int y2)
                                                                 35
```

```
36
    {
        int a1=s.ask(x2,y2)*(x2+1)*(y2+1)-s.ask(x1-1,y2)*x1*(
37
38
              -s.ask(x2,y1-1)*(x2+1)*y1+s.ask(x1-1,y1-1)*x1*y1
39
        int a2=1.ask(x2,y2)*(y2+1)-1.ask(x1-1,y2)*(y2+1)
40
41
              -l.ask(x2,y1-1)*y1+l.ask(x1-1,y1-1)*y1;
42
        int a3=r.ask(x2,y2)*(x2+1)-r.ask(x1-1,y2)*x1
43
44
              -r.ask(x2,y1-1)*(x2+1)+r.ask(x1-1,y1-1)*x1;
45
        int a4=t.ask(x2,y2)-t.ask(x1-1,y2)
46
47
              -t.ask(x2,y1-1)+t.ask(x1-1,y1-1);
        return a1-a2-a3+a4;
48
49
```

4.2 BIT_Segment_Add

HIT

```
//BIT改段求段
    struct bit
 3
      int b[maxn];
      int num;
      void add(int x,int z)
 6
         for (int i=x;i<=num;i+=(i&-i)) b[i]+=z;</pre>
 8
9
      int ask(int x)
11
         int tmp=0;
12
13
         for (int i=x;i;i==(i&-i)) tmp+=b[i];
14
         return tmp;
15
      void init(int n)
16
17
18
        memset(b, 0, sizeof(b));
19
20
    }s,t;
21
22
23
    void ins(int x,int y,int z)
24
25
      s.add(x,z); s.add(y+1,-z);
26
      t.add(x,z*x); t.add(y+1,-z*(y+1));
    }
27
28
29
    int query(int x,int y)
30
      int tmp=s.ask(y)*(y+1)-s.ask(x-1)*x
31
32
            -(t.ask(y)-t.ask(x-1));
33
      return tmp;
    }
```

4.3 BIT_Segment_Max

```
//BIT求最值
2
    struct mx bit
3
      int mx[maxn], key[maxn];
      int num;
5
      void cov(int x,int z)
8
9
         for (int i=x;i<=num;i+=(i&-i))</pre>
10
          mx[i]=max(mx[i],z);
      int ask(int 1,int r)
12
13
14
         int tmp=-inf;
        while (1<=r)
15
16
17
           tmp=max(tmp,key[r]);
           for (r=1;r-(r&-r)>=1;r-=(r&-r))
18
             tmp=max(tmp, mx[r]);
19
20
21
         return tmp;
22
23
      void init(int x)
24
25
26
        memset(mx, 0, sizeof(mx));
```

```
memset(key,0,sizeof(key));
}
tree;
```

27

28

29

4.4 BIT_single

目录

```
struct bit
  int sum[maxn];
                                                                   3
  int num;
  bit(){
    memset(sum, 0, sizeof(sum));}
  bit(int n):num(n){
    memset(sum, 0, sizeof(sum));}
                                                                   8
  void add(int x,int z)
                                                                   10
    for (int i=x;i<=num;i+=(i&-i))</pre>
                                                                   11
       sum[i]+=z;
                                                                   12
                                                                   13
  int ask(int x)
                                                                   14
                                                                   15
    int tmp=0:
                                                                   16
    for (int i=x;i>=1;i-=(i&-i))
                                                                    17
      tmp+=sum[i];
                                                                   18
    return tmp;
                                                                   19
                                                                    20
  void clear()
                                                                   21
                                                                   22
    memset(sum, 0, sizeof(sum));
                                                                   23
                                                                   24
};
```

4.5 BitSet

```
struct score
    friend bool operator <(score a, score b)
                                                                    5
         return a.x<b.x;
}s[6][maxn];
                                                                    8
int n,m,p,num,block;
                                                                    10
int bel[maxn], l[maxm], r[maxm];
                                                                    11
int a[6][maxn];
                                                                    12
bitset<50005> b[6][maxm];
                                                                    13
bitset<50005> now[6];
                                                                    15
int main()
                                                                    16
    int Case:
                                                                    18
    scanf("%d", &Case);
                                                                    19
    for (int i=1;i<=5;i++) b[i][0].reset();</pre>
    for (int o=1;o<=Case;o++)</pre>
                                                                    21
                                                                    22
         scanf("%d%d",&n,&m);
                                                                    23
         for (int j=1;j<=n;j++)</pre>
                                                                    24
             for (int i=1;i<=5;i++)</pre>
                                                                    25
                                                                    26
                  scanf("%d", &a[i][j]);
                                                                    27
                  s[i][j].x=a[i][j];
                                                                    28
                  s[i][j].y=j;
                                                                    29
                                                                    30
         for (int i=1;i<=5;i++)
                                                                    31
                                                                    32
             sort(s[i]+1,s[i]+n+1);
                                                                    33
             sort(a[i]+1,a[i]+n+1);
                                                                    34
                                                                    35
         int block=sqrt(n);
         int num=n/block+(n%block!=0);
                                                                    37
         for (int i=1;i<=n;i++) bel[i]=(i-1)/block+1;</pre>
                                                                    38
             (int i=1;i<=num;i++) l[i]=(i-1)*block+1;
                                                                    39
         for
         for (int i=1;i<=num;i++) r[i]=i*block;</pre>
                                                                    40
         r[num]=n;
                                                                    41
         for (int i=1;i<=5;i++)</pre>
                                                                    42
             for (int j=1;j<=num;j++)</pre>
                                                                    43
                                                                    44
                  b[i][j]=b[i][j-1];
                                                                    45
                  for (int k=1[j];k<=r[j];k++)</pre>
                                                                    46
                      b[i][j][s[i][k].y]=1;
                                                                    47
                                                                    48
```

```
49
              scanf("%d", &p);
              int x[6], ans=0;
50
51
              for (int q=1;q<=p;q++)</pre>
52
                  for (int i=1;i<=5;i++) scanf("%d",&x[i]);</pre>
53
                  for (int i=1;i<=5;i++) x[i]^=ans;</pre>
                  for (int i=1;i<=5;i++) now[i].reset();</pre>
55
56
                  for (int i=1;i<=5;i++)</pre>
57
                  {
                       int tmp=upper_bound(a[i]+1,a[i]+n+1,x[i])-
58
                       a[i]-1;
59
                       if (tmp==0) continue;
                       now[i]=b[i][bel[tmp]-1];
60
                       for (int j=l[bel[tmp]];j<=tmp;j++)</pre>
61
                           now[i][s[i][j].y]=1;
62
63
                  now[0]=now[1]&now[2]&now[3]&now[4]&now[5];
64
65
                  ans=now[0].count();
66
                  printf("%d\n", ans);
67
             }
68
69
         return 0;
    }
70
```

4.6 CDQ_Divide

```
1
    //CDQ分治(带修改区间k大)
    int n, m, tot, cnt;
 3
    int a[maxn], ans[maxm];
    struct query
 6
 7
      int id,x,y,k,cur;
 8
       query(){}
       query(int id,int x,int y,int k,int cur):
 9
         id(id), x(x), y(y), k(k), cur(cur){}
    }q[maxm],q1[maxm],q2[maxm];
11
12
13
    struct bit
14
15
       int num;
16
      int s[maxn];
       void add(int x,int z)
17
18
         for (int i=x;i<=num;i+=(i&-i)) s[i]+=z;</pre>
19
20
      int ask(int x)
21
22
23
         int tmp=0;
24
         for (int i=x;i>=1;i-=(i&-i)) tmp+=s[i];
25
         return tmp;
26
27
      void init(int n)
28
29
         memset(s,0,sizeof(s));
30
31
    }tree;
32
33
    void solve(int st,int ed,int l,int r)
34
35
      if (st>ed) return ;
36
37
          (1==r)
38
39
         for (int i=st;i<=ed;i++) ans[q[i].id]=1;</pre>
40
         return ;
41
      int mid=(l+r)>>1, n1=0, n2=0;;
42
43
      for (int i=st;i<=ed;i++)</pre>
44
45
         if (q[i].id==0)
46
47
           if (q[i].y<=mid)</pre>
48
             tree.add(q[i].x, q[i].k);
49
             q1[n1++]=q[i];
50
51
52
           else
53
             q2[n2++]=q[i];
54
55
         else
56
           int tmp=tree.ask(q[i].y)-tree.ask(q[i].x-1);
57
```

```
if (q[i].cur+tmp>=q[i].k)
         q1[n1++]=q[i];
                                                                     59
       else
                                                                     60
                                                                     61
         q[i].cur+=tmp;
                                                                     62
         q2[n2++]=q[i];
                                                                     63
                                                                     64
    }
                                                                     65
                                                                     66
  for (int i=st;i<=ed;i++)</pre>
                                                                     67
    if (q[i].id==0 \& q[i].y\leq mid) tree.add(q[i].x, -q[i].
                                                                     68
    k);
  for (int i=0;i<n1;i++) q[st+i]=q1[i];</pre>
                                                                     69
  for (int i=0;i<n2;i++) q[st+n1+i]=q2[i];</pre>
                                                                     70
  solve(st, st+n1-1, l, mid);
                                                                     71
  solve(st+n1,ed,mid+1,r);
                                                                     72
                                                                     73
                                                                     74
int main()
                                                                     75
                                                                     76
                                                                     77
  int Case:
  scanf("%d", &Case);
                                                                     78
  while (Case-)
                                                                     79
                                                                     80
    scanf("%d%d",&n,&m);
                                                                     81
    tot=cnt=0:
                                                                     82
    for (int i=1;i<=n;i++)</pre>
                                                                     83
                                                                     84
       scanf("%d", &a[i]);
                                                                     85
       q[tot++]=query(0,i,a[i],1,0);
                                                                     86
                                                                     87
    int x,y,z;
                                                                     88
    char sign;
                                                                     89
    for (int i=1;i<=m;i++)</pre>
                                                                     90
                                                                     91
       scanf(" %c", &sign);
       if (sign=='C')
                                                                     93
                                                                     94
         scanf("%d%d",&x,&y);
                                                                     95
         q[tot++]=query(0,x,a[x],-1,0);
                                                                     96
         a[x]=y;
                                                                     97
         q[tot++]=query(0,x,a[x],1,0);
                                                                     98
                                                                     99
       else
                                                                     100
                                                                     101
         scanf("%d%d%d",&x,&y,&z);
                                                                     102
         q[tot++]=query(++cnt,x,y,z,0);
                                                                     103
                                                                     104
                                                                     105
                                                                     106
    tree.init(n);
                                                                     107
    solve(0, tot-1, 0, inf);
                                                                     108
    for (int i=1;i<=cnt;i++) printf("%d\n",ans[i]);</pre>
                                                                     109
                                                                     110
  return 0;
                                                                     111
                                                                     112
```

4.7 ChainPartitionTree

```
vector<int> t[maxn];
int dep[maxn], pos[maxn], top[maxn], size[maxn], fa[maxn];
                                                                   3
int sum[6*maxn], mx[6*maxn];
int n, m, num;
                                                                   5
                                                                   6
void dfs_one(int x)
                                                                   8
  size[x]=1;
  int ss=t[x].size();
                                                                   10
  for (int j=0;j<ss;j++)</pre>
                                                                   11
    int y=t[x][j];
                                                                   13
    if (y==fa[x]) continue;
                                                                   14
    dep[y]=dep[x]+1;
                                                                   15
    fa[y]=x;
                                                                   16
                                                                   17
    dfs_one(y);
    size[x]+=size[y];
                                                                   18
  }
                                                                   19
                                                                   20
{f void} dfs_two(int x, int anc)
                                                                   21
                                                                   22
  pos[x]=++num;
                                                                   23
  top[x]=anc;
                                                                   24
```

目录

```
25
       int son=0, ss=t[x].size();
       for (int j=0;j<ss;j++)</pre>
 26
 27
 28
          int y=t[x][j];
 29
          if (y!=fa[x] && size[y]>size[son]) son=y;
 30
       if (son) dfs_two(son, anc);
31
 32
       for (int j=0;j<ss;j++)</pre>
 33
 34
          int y=t[x][j];
 35
          if (y!=fa[x] && y!=son) dfs_two(y, y);
 36
37
 38
     void update(int x)
 39
 40
       sum[x]=sum[x<<1]+sum[x<<1|1];
       mx[x]=max(mx[x<<1], mx[x<<1|1]);</pre>
 41
 42
 43
     void build(int x, int 1, int r)
 44
       if (l==r)
 45
 46
          sum[x]=0;
 47
 48
          mx[x]=-inf;
 49
          return ;
 50
 51
       int mid=(l+r)>>1;
 52
       build(x<<1, 1, mid);
       build(x<<1|1, mid+1, r);</pre>
53
 54
55
     void change(int x, int 1, int r, int p, int z)
56
       if (l==r)
 57
 58
 59
          sum[x]=mx[x]=z;
 60
          return ;
 61
 62
       int mid=(l+r)>>1;
       if (p<=mid) change(x<<1, 1, mid, p, z);</pre>
 63
       else change(x << 1|1, mid+1, r, p, z);
 64
 65
       update(x);
66
 67
     int ask_sum(int x, int 1, int r, int 11, int rr)
 68
69
       if (11<=1 && r<=rr) return sum[x];</pre>
70
       int mid=(1+r)>>1, tmp=0;
 71
       if (ll<=mid) tmp+=ask_sum(x<<1, l, mid, ll, rr);</pre>
       if (rr>mid) tmp+=ask_sum(x<<1|1, mid+1, r, ll, rr);</pre>
72
       return tmp;
73
 74
     int ask_max(int x, int 1, int r, int 11, int rr)
75
       if (11<=1 && r<=rr) return mx[x];</pre>
 77
78
       int mid=(l+r)>>1, tmp=-inf;
       if (l1<=mid) tmp=max(tmp, ask_max(x<<1, 1, mid, l1, rr))</pre>
79
 80
       if (rr>mid) tmp=max(tmp, ask_max(x<<1|1, mid+1, r, ll,
       rr));
 81
       return tmp;
 82
     int get_sum(int x, int y)
83
 84
 85
       int tmp=0;
       while (top[x]!=top[y])
86
 87
 88
          if (dep[top[x]]<dep[top[y]]) swap(x, y);</pre>
          tmp+=ask\_sum(1, 1, n, pos[top[x]], pos[x]);
 89
          x=fa[top[x]];
 91
 92
       if (pos[x]>pos[y]) swap(x, y);
 93
       tmp+=ask\_sum(1, 1, n, pos[x], pos[y]);
       return tmp:
 94
 95
     int get_max(int x, int y)
96
97
98
       int tmp=—inf;
99
       while (top[x]!=top[y])
100
101
          if (dep[top[x]]<dep[top[y]]) swap(x, y);</pre>
          tmp=max(tmp, ask_max(1, 1, n, pos[top[x]], pos[x]));
102
103
          x=fa[top[x]];
104
105
       if (pos[x]>pos[y]) swap(x, y);
```

HIT

```
tmp=max(tmp, ask_max(1, 1, n, pos[x], pos[y]));
                                                                     106
  return tmp;
                                                                     107
                                                                     108
int change_value(int x, int z)
                                                                     109
                                                                     110
  change(1, 1, n, pos[x], z);
                                                                     111
                                                                     112
                                                                     113
int main()
                                                                     114
                                                                     115
  scanf("%d", &n);
                                                                     116
                                                                     117
  for (int i=1;i<=n;i++) t[i].clear();</pre>
                                                                     118
  for (int i=1;i<n;i++)</pre>
                                                                     120
    scanf("%d%d",&x,&y);
                                                                     121
    t[x].push_back(y);
                                                                     122
    t[y].push_back(x);
                                                                     123
                                                                     124
  num=0:
                                                                     125
  dfs_one(1);
                                                                     126
  dfs_two(1, 1);
                                                                     127
  build(1,1,n);
                                                                     128
  for (int i=1;i<=n;i++)</pre>
                                                                     129
                                                                     130
    scanf("%d", &x);
                                                                     131
    change_value(i, x);
                                                                     132
                                                                     133
  char sign[10];
                                                                     134
  int z;
                                                                     135
  scanf("%d", &m);
                                                                     136
  for (int i=1;i<=m;i++)
                                                                     137
                                                                     138
    scanf(" %s%d%d",sign,&x,&y);
if (sign[0]=='C')
                                                                     139
                                                                     140
      change_value(x, y)
                                                                     141
    else if (sign[1]=='M')
                                                                     142
      printf("%d\n",get_max(x, y));
                                                                     143
                                                                     144
      printf("%d\n",get_sum(x, y));
                                                                     145
                                                                     146
                                                                     147
  return 0:
                                                                     148
```

4.8 ChairManTree

```
//主席树
struct node
  node *lc, *rc;
  int s
  node () {s=0;}
}*rot[maxn], *sum[maxn], *p;
vector <node*> st[4];
                                                                     8
struct et
                                                                     10
  int s,t,next;
}e[maxn*2];
                                                                     12
int ll[maxn], rr[maxn], fir[maxn], key[maxn];
                                                                     1.3
int w[maxn*2], dep[maxn], fa[maxn][20];
int x[maxn], y[maxn], z[maxn];
                                                                     15
int n, m, tot, low, cnt, num;
                                                                     16
int v[5];
                                                                     17
                                                                     18
node *build(int 1,int r)
                                                                     19
                                                                     20
  node *now=new node();
                                                                     21
  if (l==r) return now;
                                                                     22
  int mid=(l+r)>>1;
                                                                     23
  now—>lc=build(1,mid);
                                                                     24
  now->rc=build(mid+1,r);
  return now;
                                                                     26
                                                                     27
                                                                     28
node *change(int 1, int r, int x, int z)
                                                                     29
                                                                     30
  node *now=new node();
                                                                     31
  now->lc=p->lc; now->rc=p->rc;
                                                                     32
  now->s=p->s+z;
  if (l==r) return now;
                                                                     34
  int mid=(l+r)>>1;
                                                                     35
  if (x \le mid) p = p \longrightarrow lc, now \longrightarrow lc = change(1, mid, x, z);
                                                                     36
  else p=p->rc, now->rc=change(mid+1, r, x, z);
                                                                     37
```

12/24

```
38
       return now;
 39
 40
 41
     void adj(int x,int y,int z)
 42
       if (x==0) return;
 43
       for (int i=x;i<=n;i+=(i&-i)) p=rot[i],rot[i]=change(0,</pre>
       num, y, z);
 46
     void fill(int k)
 47
 48
       st[k].clear();
 49
       st[k].push_back(sum[v[k]]);
 50
       for (int i=11[v[k]];i>0;i-=(i\&-i)) st[k].push_back(rot[i
 51
 52
 53
 54
     int getsum(int k)
 55
       int ans=0, n=st[k].size();
 56
 57
       for (int i=0;i<n;i++) ans+=st[k][i]->s;
       return ans;
 58
 59
 60
     int getsize(int k)
 61
 62
 63
       int ans=0, n=st[k].size();
       for (int i=0;i<n;i++) ans+=st[k][i]->rc->s;
 64
       return ans;
 65
 66
 67
     void goleft(int k)
 68
 69
 70
       int n=st[k].size();
       for (int i=0;i<n;i++) st[k][i]=st[k][i]->lc;
 71
 72
 73
 74
     void goright(int k)
 75
 76
       int n=st[k].size();
       for (int i=0;i<n;i++) st[k][i]=st[k][i]->rc;
 77
 78
     int query(int 1,int r,int k)
 80
 81
       if (k>getsum(1)+getsum(2)-getsum(3)-getsum(4)) return 0;
 82
       if (l==r) return 1;
 83
       int tmp=getsize(1)+getsize(2)-getsize(3)-getsize(4);
 84
       //cout<<l<' '<<r<"tmp:"<<tmp<<endl;
 85
 86
       int mid=(l+r)>>1;
       if (k<=tmp)</pre>
 87
 88
 89
          //cout<<"right\n";
          goright(1), goright(2), goright(3), goright(4);
 90
          return query(mid+1,r,k);
 91
 92
       else
 93
 94
          //cout<<"left\n";
 95
          goleft(1), goleft(2), goleft(3), goleft(4);
 96
 97
          return query(1,mid,k-tmp);
 98
     }
99
100
101
     int lca(int x,int y)
102
103
       if (dep[x]>dep[y]) swap(x,y);
       for (int i=low;i>=0;i---)
104
105
          if (dep[fa[y][i]]>=dep[x]) y=fa[y][i];
106
       if (x==y) return x;
       for (int i=low;i>=0;i---)
107
108
         if (fa[x][i]!=fa[y][i]) x=fa[x][i],y=fa[y][i];
       return fa[x][0];
109
110
111
     void write(node *now,int 1,int r)
112
113
114
       if (l==r)
115
116
          cout<<now->s;
          return ;
117
118
```

HIT

```
int mid=(l+r)>>1;
  write(now->lc, 1, mid);
                                                                   120
  write(now->rc,mid+1,r);
                                                                   121
                                                                   122
                                                                   123
void dfs(int now)
                                                                   124
                                                                   125
  11[now]=rr[now]=++cnt;
                                                                   126
  p=sum[fa[now][0]];
                                                                   127
  sum[now]=change(0, num, key[now], 1);
                                                                   128
                                                                   129
  for (int j=fir[now];j;j=e[j].next)
                                                                   130
                                                                   131
    int k=e[j].t;
                                                                   132
    if (fa[now][0]!=k)
                                                                   133
                                                                   134
      fa[k][0]=now; dep[k]=dep[now]+1;
                                                                   135
      for (int i=1;i<=low;i++) fa[k][i]=fa[fa[k][i-1]][i</pre>
                                                                   136
       -11:
      dfs(k);
                                                                   137
      rr[now]=rr[k];
                                                                   138
                                                                   139
 }
                                                                   140
                                                                   141
                                                                   142
void add(int x,int y)
                                                                   143
                                                                   144
  e[++tot].s=x; e[tot].t=y; e[tot].next=fir[x]; fir[x]=tot
  e[++tot].s=y; e[tot].t=x; e[tot].next=fir[y]; fir[y]=tot
                                                                   146
                                                                   147
                                                                   148
int main()
                                                                   149
                                                                   150
  //freopen("network10.in", "r", stdin);
                                                                   151
  //freopen("network.out","w",stdout);
                                                                   152
  scanf("%d%d",&n,&m);
                                                                   153
  low=log(n)/log(2.0);
                                                                   154
  for (int i=1;i<=n;i++) scanf("%d",&key[i]),w[++num]=key[</pre>
                                                                   155
  int st.ed:
                                                                   156
  for (int i=1;i<n;i++) scanf("%d%d",&st,&ed),add(st,ed);</pre>
                                                                   157
                                                                   158
  for (int i=1;i<=m;i++)</pre>
                                                                   159
                                                                   160
    scanf("%d%d%d",&z[i],&x[i],&y[i]);
                                                                   161
    if (!z[i]) w[++num]=y[i];
                                                                   162
                                                                   163
  sort(w+1, w+num+1);
                                                                   164
  num=unique(w+1, w+num+1)—w;
                                                                   165
  for (int i=1;i<=n;i++) key[i]=lower_bound(w+1, w+num+1,</pre>
                                                                   166
  key[i])—w;
  for (int i=1;i<=m;i++) if (!z[i]) y[i]=lower_bound(w+1,</pre>
                                                                   167
  w+num+1, y[i])—w;
                                                                   168
  sum[0]=rot[0]=build(0,num);
                                                                   169
  for (int i=1;i<=n;i++) rot[i]=rot[0];</pre>
                                                                   170
  int root=rand()%n+1; dep[root]=1;
                                                                   171
  dfs(root);
                                                                   172
                                                                   173
  for (int i=1;i<=m;i++)</pre>
                                                                   174
                                                                   175
    if (z[i])
                                                                   176
                                                                   177
      int tmp;
                                                                   178
      v[1]=x[i], v[2]=y[i], v[3]=tmp=lca(x[i], y[i]), v[4]=fa[
                                                                   179
      tmp][0];
      fill(1); fill(2); fill(3); fill(4);
                                                                   180
      int ans=query(0,num,z[i]);
                                                                   181
      if (ans)
                                                                   182
        printf("%d\n",w[ans]);
                                                                   183
                                                                   184
        printf("invalid request!\n");
                                                                   185
                                                                   186
    else
                                                                   187
                                                                   188
      adj(ll[x[i]], key[x[i]], -1), adj(rr[x[i]]+1, key[x[i]])
                                                                   189
      11,1);
      key[x[i]]=y[i];
                                                                   190
      adj(ll[x[i]], key[x[i]], 1), adj(rr[x[i]]+1, key[x[i
                                                                   191
      ]],-1);
                                                                   192
```

4.9 DSJ

```
//并查集
2
    int f[maxn];
 3
    int find(int x)
 5
      return f[x]==-1?x:f[x]=find(f[x]);
    bool merge(int x, int y)
 8
9
      int fx=find(x), fy=find(y);
10
      if (fx!=fy)
11
12
13
         f[fx]=fy;
14
         return 1;
15
      return 0;
16
    }
17
```

4.10 KD_Tree

```
//KD-Tree
    //维护K维空间上离某点最近的点序
    //HDU4347
    #include <bits/stdc++.h>
    #define maxn 52000
    #define square(x) ((x) * (x))
    using namespace std;
    int idx:
9
10
    struct Node
11
12
      int x[12];
      Node *lc,
13
14
      node(){}
      bool operator < (const Node &u) const</pre>
15
16
         return x[idx]<u.x[idx];</pre>
17
18
19
    }data[maxn];
    typedef pair<double, Node*> pdn;
20
21
    struct kdt
22
      Node *rot;
23
24
      int k:
      priority_queue<pdn>que;
25
26
      kdt(int k):k(k){rot=NULL;}
27
      Node* build(int 1, int r, int dep)
28
29
         if (l>r) return NULL;
30
         Node *now=new Node();
31
         idx=dep%k;
32
         int mid=(l+r)>>1;
33
34
         nth_element(data+l, data+mid, data+r+1);
         memcpy(now->x, data[mid].x, sizeof(now->x));
35
         now->lc=build(1, mid-1, dep+1);
36
37
         now->rc=build(mid+1, r, dep+1);
38
         return now;
39
      void build(int n, int nk)
40
41
42
         k=nk:
43
         rot=build(0, n-1, 0);
44
45
      void query(Node* now, Node *p, int m, int dep)
46
         if (now==NULL) return ;
47
         double sum=0;
48
         for (int i=0;i<k;i++)</pre>
49
50
           sum+=square(now->x[i]-p->x[i]);
         int dim=dep%k;
         Node *ll=now->lc, *rr=now->rc;
52
53
         if (p\rightarrow x[dim] >= now\rightarrow x[dim]) swap(ll, rr);
54
         query(ll, p, m, dep+1);
         if (que.size()<m)</pre>
55
```

```
que.push(pdn(sum, now));
                                                                     57
      query(rr, p, m, dep+1);
                                                                     58
                                                                     59
    else
                                                                     60
                                                                     61
    {
       if (sum<que.top().first)</pre>
                                                                     62
                                                                     63
         que.pop();
                                                                     64
         que.push(pdn(sum, now));
                                                                     65
                                                                     66
       if (square(p->x[dim]-now->x[dim])<que.top().first)</pre>
                                                                     67
         query(rr, p, m, dep+1);
                                                                     68
                                                                     69
                                                                     70
  void query(Node *p, int m)
                                                                     71
                                                                     72
    while (!que.empty()) que.pop();
                                                                     73
    query(rot, p, m, 0);
                                                                     74
                                                                     75
  void print()
                                                                     76
                                                                     77
    vector<Node*> t;
                                                                     78
    t.clear();
                                                                     79
    while (!que.empty())
                                                                     80
                                                                     81
      t.push_back(que.top().second);
                                                                     82
      que.pop();
                                                                     83
                                                                     84
    int num=t.size();
                                                                     85
    for (int i=num-1;i>=0;i-
                                                                     86
      for (int j=0;j<k;j++)</pre>
                                                                     87
         printf(j==k-1?"%d\n":"%d ",t[i]=>x[j]);
                                                                     88
                                                                     89
                                                                     90
int n, k, m, q;
                                                                     92
int main()
                                                                     93
                                                                     94
  while (scanf("%d%d",&n,&k)!=EOF)
                                                                     95
                                                                     96
    for (int i=0;i<n;i++)</pre>
                                                                     97
      for (int j=0; j<k; j++)
                                                                     98
         scanf("%d", &data[i].x[j]);
                                                                     99
    s.build(n, k);
                                                                     100
    scanf("%d", &q);
                                                                     101
    for (int i=1;i<=q;i++)</pre>
                                                                     102
                                                                     103
       Node *p=new Node();
                                                                     104
      for (int j=0;j<k;j++)
  scanf("%d",&p->x[j]);
                                                                     105
                                                                     106
       scanf("%d", &m);
                                                                     107
      s.query(p, m);
                                                                     108
      printf("the closest %d points are:\n",m);
                                                                     109
       s.print();
                                                                     110
      delete p:
                                                                     111
                                                                     112
                                                                     113
  return 0;
                                                                     114
```

4.11 LCT

```
//LCT
void update(int x)
                                                                        3
  if (!x) return ;
  sum[x]=(sum[c[x][0]]+sum[c[x][1]]+key[x])%00;
  siz[x]=(siz[c[x][0]]+siz[c[x][1]]+1)\%00;
                                                                        6
void change(int x, int mu, int ad)
                                                                        10
  if (!x) return ;
                                                                        11
  \text{key}[x]=(\text{key}[x]*\text{mu}\%\text{oo}+\text{ad})\%\text{oo};
                                                                        12
  mul[x]=mul[x]*mu%oo;
                                                                        13
  add[x]=(add[x]*mu\%oo+ad)\%oo;
                                                                        14
  sum[x]=(sum[x]*mu%oo+ad*siz[x]%oo)%oo;
                                                                        15
                                                                        16
                                                                        17
void reverse(int x)
                                                                        18
                                                                        19
  if (!x) return ;
                                                                        20
```

```
21
       swap(c[x][0], c[x][1]);
 22
       rev[x]^=1;
 23
     }
 24
 25
     void down(int x)
 26
 27
       if (rev[x])
 28
 29
          reverse(c[x][0]);
 30
         reverse(c[x][1]);
 31
          rev[x]=0;
 32
       change(c[x][0], mul[x], add[x]);
 33
 34
       change(c[x][1], mul[x], add[x]);
 35
       mul[x]=1;
 36
       add[x]=0;
 37
     }
 38
39
     bool is_root(int x)
 40
       return c[fa[x]][0]!=x && c[fa[x]][1]!=x;
 41
 42
 43
     void relax(int x)
 44
 45
       if (!is_root(x)) relax(fa[x]);
 46
 47
       down(x);
 48
     }
 49
50
     void zigzag(int x)
51
       int y=fa[x], z=fa[y];
52
       int p=(c[y][1]==x), q=p^1;
 53
       if (!is_root(y)) c[z][c[z][0]!=y]=x;
 54
 55
       fa[x]=z; fa[y]=x; fa[c[x][q]]=y;
       c[y][p]=c[x][q]; c[x][q]=y;
57
       update(y);
 58
59
     void splay(int x)
 60
 61
       relax(x);
 62
 63
       while (!is_root(x))
 64
          int y=fa[x], z=fa[y];
 65
 66
          if (!is_root(y))
 67
            ((c[y][0]==x) \land (c[z][0]==y))?zigzag(x):zigzag(y);
 68
          zigzag(x);
 69
 70
       update(x);
71
 73
     int access(int x)
 74
 75
       int y=0;
 76
       for (; x; y=x, x=fa[x])
 77
          splay(x);
 78
79
         c[x][1]=y;
 80
         update(x);
81
 82
       return y;
 83
84
 85
     int root(int x)
 86
 87
       while (fa[x]) x=fa[x];
 88
       return x;
 89
 90
91
     void make_root(int x)
92
 93
       access(x);
 94
       splay(x)
95
       reverse(x);
 96
97
98
     void split(int x, int y)//x is root
99
       make_root(y);
100
101
       access(x);
102
       splay(x);
```

HIT

103

```
104
void link(int x, int y)
                                                                  105
                                                                  106
  make_root(x);
                                                                  107
  fa[x]=y;
                                                                  108
                                                                  109
                                                                  110
void cut(int x, int y)
                                                                  111
                                                                  112
  split(y, x);
                                                                  113
  c[y][0]=fa[x]=0;
                                                                  114
                                                                  115
                                                                  116
void solve(int x, int y, int z)
                                                                  117
                                                                  118
  access(x);
                                                                  119
  int lca=access(y);
                                                                  120
  splay(x);
                                                                  121
  change(c[lca][1], z);
                                                                  122
  kev[lca]=z;
                                                                  123
  if (x!=lca) change(x, z);
                                                                  124
                                                                  125
```

4.12 LeftistTree

```
//左偏树
int v[maxn], dis[maxn], sz[maxn];
int l[maxn], r[maxn];
                                                                3
int rot[maxn], st[maxn], ed[maxn];
int num;
                                                                5
int build(int z){
  v[++num]=z;
  sz[num]=1;
                                                                8
  l[num]=r[num]=dis[num]=0;
  return num;
                                                                10
                                                                11
int merge(int x, int y){
                                                                12
  if (!x || !y) return x+y;
                                                                13
  if (v[x] < v[y]) swap(x, y);//bit rot
  r[x]=merge(r[x], y);
                                                                15
  if (dis[r[x]]>dis[l[x]]) swap(l[x], r[x]);
                                                                16
  sz[x]=sz[l[x]]+sz[r[x]]+1;
                                                                17
  dis[x]=dis[r[x]]+1;
                                                                18
  return x;
                                                                19
                                                                20
int top(int x){
                                                                21
  return v[x];
                                                                22
                                                                23
int size(int x){
                                                                24
  return sz[x];
                                                                25
                                                                26
void pop(int &x){
                                                                27
  x=merge(1[x], r[x]);
                                                                28
                                                                29
```

4.13 Merge

```
//分治求逆序对
void solve(int 1, int r)
  if (l==r) return ;
  int mid=(l+r)>>1;
  solve(l, mid)
                                                                6
  solve(mid+1, r);
  int i=1, j=mid+1, k=0;
                                                                8
  while (i<=mid || j<=r)
                                                                9
                                                                10
    if (j>r || i<=mid && q[i]<=q[j]) nq[k++]=q[i++], sum</pre>
                                                                11
    +=(j-mid-1);
    else
                                                                12
    if (i>mid || j<=r && q[i]>q[j]) nq[k++]=q[j++];
                                                                13
                                                                14
  for (int i=0;i<k;i++) q[l+i]=nq[i];</pre>
                                                                15
                                                                16
```

4.14 SegTree

```
//线段树最大连续1

struct segtree 2

{

   int mx[maxn*8], lmx[maxn*8], rmx[maxn*8], size[maxn*8]; 4

   void update(int x) 5
```

目录

```
6
      {
        mx[x]=max(max(mx[x*2],mx[x*2+1]),rmx[x*2]+lmx[x*2+1]);
8
        if (lmx[x*2]==size[x*2]) lmx[x]=size[x*2]+lmx[x*2+1];
        else lmx[x]=lmx[x*2];
        if (rmx[x*2+1]==size[x*2+1]) rmx[x]=size[x*2+1]+rmx[x
10
         *2];
        else rmx[x]=rmx[x*2+1];
11
12
      void build(int x,int 1,int r)
13
14
15
        if (l==r)
16
          size[x]=1;
17
           return ;
19
20
        int mid=(l+r)>>1;
        build(x*2,1,mid);
21
        build(x*2+1,mid+1,r);
22
23
        size[x]=size[x*2]+size[x*2+1];
24
25
      void add(int x, int 1, int r, int pos, int z)
26
        if (l==r)
27
28
          mx[x]=lmx[x]=rmx[x]=z;
29
30
          return ;
31
32
        int mid=(l+r)>>1;
        if (pos<=mid) add(x*2, 1, mid, pos, z);
33
        else add(x*2+1, mid+1, r, pos, z);
34
35
        update(x);
36
37
      int ask(int x, int 1, int r, int 11, int rr, int &res)
38
        if (ll<=l && r<=rr)
39
40
           int tmp=max(res+lmx[x], mx[x]);
41
42
          if (rmx[x]==size[x]) res+=size[x];
          else res=rmx[x];
43
44
          return tmp;
45
        int mid=(l+r)>>1, tmp=0;
46
47
        if (11 \le mid) tmp=max(tmp,ask(x*2, 1, mid, 11, rr, res)
        if (rr>mid) tmp=max(tmp,ask(x*2+1, mid+1, r, ll, rr,
48
        res));
        return tmp;
49
50
      void init()
51
52
        memset(mx, 0, sizeof(mx));
53
        memset(lmx, 0, sizeof(lmx));
54
55
        memset(rmx, 0, sizeof(rmx));
56
    }tree;
```

4.15 SetUnion

```
//set 启发式合并
2
    int merge(int x,int y)
3
      int tmp=inf;
      SIT prev, succ;
      if (s[x].size() < s[y].size()) swap(s[x],s[y]);
      for (SIT i=s[y].begin();i!=s[y].end();i++)
        prev=succ=s[x].lower_bound(*i);
9
        if (prev!=s[x].begin()) prev—
10
        if (prev!=s[x].end()) tmp=min(tmp,abs(*prev-*i));
11
12
        if (succ!=s[x].end())
                                 tmp=min(tmp,abs(*succ-*i));
        s[x].insert(*i);
14
      s[y].clear();
16
      return tmp;
17
```

4.16 Splay_Segment

```
1 //Splay区间维护
2 //支持区间求和,最大连续子段,区间反转,区间覆盖操作
3 int c[maxn][2],fa[maxn];
4 int a[maxn],key[maxn],sum[maxn],la[maxn],ra[maxn],ma[maxn],cov[maxn],size[maxn];
```

```
int q[maxn];
bool rev[maxn];
                                                                                                                                                                                                                                                         6
int n,m,tot,num,rot,st,ed,tail;
void update(int x)
                                                                                                                                                                                                                                                         9
                                                                                                                                                                                                                                                          10
                if (!x) return;
                                                                                                                                                                                                                                                          11
                la[x]=max(la[c[x][0]], sum[c[x][0]]+key[x]+max(0, la[c[x][0])+key[x]+max(0, la[c[x][0])+key[x]
                                                                                                                                                                                                                                                          12
                 ][1]]));
                ra[x]=max(ra[c[x][1]], sum[c[x][1]]+key[x]+max(0, ra[c[x][1])+key[x]+max(0, ra[c[x][1])+key[x]
                                                                                                                                                                                                                                                          13
                 ][0]]));
                ma[x]=max(max(ma[c[x][0]],ma[c[x][1]]),key[x]+max(0,ra)
                 [c[x][0]]+max(0,la[c[x][1]]));
                 sum[x]=sum[c[x][0]]+sum[c[x][1]]+key[x];
                                                                                                                                                                                                                                                          15
                size[x]=size[c[x][0]]+size[c[x][1]]+1;
                                                                                                                                                                                                                                                          16
                                                                                                                                                                                                                                                         17
                                                                                                                                                                                                                                                         18
void reverse(int x)
                                                                                                                                                                                                                                                         19
                                                                                                                                                                                                                                                         20
                 if (!x) return;
                                                                                                                                                                                                                                                         21
                swap(c[x][0], c[x][1]);
                                                                                                                                                                                                                                                         22
                 swap(la[x],ra[x]);
                                                                                                                                                                                                                                                         23
                rev[x]^=1;
                                                                                                                                                                                                                                                         24
                                                                                                                                                                                                                                                         25
                                                                                                                                                                                                                                                          26
void recover(int x,int z)
                                                                                                                                                                                                                                                         27
                                                                                                                                                                                                                                                         28
                 if (!x) return ;
                                                                                                                                                                                                                                                          29
                key[x]=cov[x]=z;
                                                                                                                                                                                                                                                         30
                sum[x]=size[x]*z;
                                                                                                                                                                                                                                                         31
                la[x]=ra[x]=ma[x]=max(z,sum[x]);
                                                                                                                                                                                                                                                         32
                                                                                                                                                                                                                                                        33
                                                                                                                                                                                                                                                          34
void down(int x)
                                                                                                                                                                                                                                                          35
                                                                                                                                                                                                                                                         36
                 if (!x) return;
                                                                                                                                                                                                                                                         37
                                                                                                                                                                                                                                                         38
                if (rev[x])
                                                                                                                                                                                                                                                         39
                 {
                                 reverse(c[x][0]);
                                                                                                                                                                                                                                                          40
                                 reverse(c[x][1]);
                                                                                                                                                                                                                                                          41
                                 rev[x]=0;
                                                                                                                                                                                                                                                          42
                                                                                                                                                                                                                                                         43
                if (cov[x]!=-inf)
                                                                                                                                                                                                                                                          44
                                                                                                                                                                                                                                                          45
                                 recover(c[x][0],cov[x]);
                                                                                                                                                                                                                                                         46
                                 recover(c[x][1],cov[x]);
                                                                                                                                                                                                                                                          47
                                 cov[x]=-inf;
                                                                                                                                                                                                                                                          48
                                                                                                                                                                                                                                                          49
                                                                                                                                                                                                                                                         50
                                                                                                                                                                                                                                                         51
void relax(int x,int rot)
                                                                                                                                                                                                                                                        52
                                                                                                                                                                                                                                                         53
                if (x!=rot) relax(fa[x],rot);
                                                                                                                                                                                                                                                         54
                down(x);
                                                                                                                                                                                                                                                         55
                                                                                                                                                                                                                                                        56
                                                                                                                                                                                                                                                         57
void rotate(int x,int &rot)
                                                                                                                                                                                                                                                         58
                                                                                                                                                                                                                                                        59
                int y=fa[x],z=fa[y];
                                                                                                                                                                                                                                                          60
                int p=(c[y][1]==x),q=p^1;
                                                                                                                                                                                                                                                          61
                if (y==rot) rot=x;
                                                                                                                                                                                                                                                         62
                c[z][c[z][1]==y]=x;
                                                                                                                                                                                                                                                         63
                fa[x]=z; fa[y]=x; fa[c[x][q]]=y;
                                                                                                                                                                                                                                                          64
                c[y][p]=c[x][q]; c[x][q]=y;
                                                                                                                                                                                                                                                         65
                                                                                                                                                                                                                                                          66
                update(v);
                                                                                                                                                                                                                                                          67
                                                                                                                                                                                                                                                         68
void splay(int x,int &rot)
                                                                                                                                                                                                                                                          69
                                                                                                                                                                                                                                                         70
                relax(x,rot);
                                                                                                                                                                                                                                                        71
                while (x!=rot)
                                                                                                                                                                                                                                                         72
                                                                                                                                                                                                                                                         73
                                 int y=fa[x], z=fa[y];
                                                                                                                                                                                                                                                        74
                                 if (y!=rot)
                                                                                                                                                                                                                                                         75
                                                                                                                                                                                                                                                         76
                                                  if ((c[y][0]==x)xor(c[z][0]==y)) rotate(x,rot)
                                                                                                                                                                                                                                                          77
                                                 else rotate(y,rot);
                                                                                                                                                                                                                                                         78
                                                                                                                                                                                                                                                          79
                                 rotate(x, rot);
                                                                                                                                                                                                                                                        80
                                                                                                                                                                                                                                                         81
                update(x);
                                                                                                                                                                                                                                                         82
                                                                                                                                                                                                                                                         83
```

```
84
      int pick()
 85
 86
 87
          if (tail) return q[tail—];
 88
          else return ++num:
 89
 90
      int setup(int x)
 91
 92
      {
          int t=pick();
 93
 94
          key[t]=a[x];
 95
          cov[t]=-inf;
          rev[t]=0:
 96
 97
          la[t]=ra[t]=ma[t]=-inf;
 98
          return t;
 99
      }
100
      int build(int 1,int r)
101
102
103
          int mid=(l+r)>>1,left=0,right=0;
                            left=build(1,mid-1);
104
          if (l<mid)</pre>
105
          int t=setup(mid);
          if (r>mid)
                            right=build(mid+1,r);
106
                            c[t][0]=left,fa[left]=t;
107
          if (left)
108
          if (right)c[t][1]=right,fa[right]=t;
          update(t);
109
110
          return t;
111
      }
112
      int find(int t,int k)
113
114
115
          down(t);
          if (k==size[c[t][0]]+1) return t;
116
          if (k<size[c[t][0]]+1) return find(c[t][0],k);
if (k>size[c[t][0]]+1) return find(c[t][1],k-size[c[t]
117
118
          ][0]]-1);
119
      }
120
121
      void del(int &x)
122
123
          if (!x) return;
          q[++tail]=x;
124
125
          fa[x]=0;
126
          del(c[x][0]);
127
          del(c[x][1]);
128
          la[x]=ra[x]=ma[x]=-inf;
129
          x=0;
      }
130
131
132
      int main()
133
          //freopen("build.in", "r", stdin);
134
          //freopen("build.out", "w", stdout);
135
136
          scanf("%d %d",&n,&m);
137
          for (int i=2;i<=n+1;i++)</pre>
                   scanf("%d", &a[i]);
138
139
          a[st=1]=0; a[ed=n+2]=0;
          ra[0]=la[0]=ma[0]=-inf;
140
141
          rot=build(1,n+2);
142
          char sign[20];
143
          int x, y, 1, r, z, ans;
144
          for (int i=1;i<=m;i++)</pre>
145
               scanf(" %s", sign);
146
147
               if (sign[0]=='I')
148
               {
                   scanf("%d %d",&x,&y);
149
150
                   //处理[x..y]区间,
                   //l=find(rot, x), r=find(rot, y+2)
151
152
                   l=find(rot,x+1); r=find(rot,x+2);
153
                   splay(r,rot); splay(1,c[rot][0]);
                   for (int j=1;j<=y;j++)</pre>
154
                        scanf("%d",&a[j]);
155
                   int tmp=build(1,y);
156
157
                   fa[tmp]=1; c[1][1]=tmp;
158
                   update(1); update(r);
159
160
              if (sign[0]=='D')
161
                   scanf("%d %d",&x,&y);
162
163
                   l=find(rot,x); r=find(rot,x+y+1);
                   splay(r,rot); splay(1,c[rot][0]);
164
165
                   del(c[l][1]);
```

HIT

```
update(1); update(r);
                                                             166
                                                             167
    if (sign[0]=='M'&&sign[2]=='K')
                                                             168
                                                             169
        scanf("%d %d %d",&x,&y,&z);
                                                             170
        l=find(rot,x); r=find(rot,x+y+1);
                                                             171
        splay(r,rot); splay(1,c[rot][0]);
                                                             172
        recover(c[1][1],z);
                                                             173
                                                             174
    if (sign[0]=='R')
                                                             175
                                                             176
        scanf("%d %d",&x,&y);
                                                             177
        l=find(rot,x); r=find(rot,x+y+1);
                                                             178
        splay(r,rot); splay(l,c[rot][0]);
                                                             179
        reverse(c[1][1]);
                                                             180
                                                             181
    if (sign[0]=='G')
                                                             182
                                                             183
        scanf("%d %d",&x,&y);
                                                             184
        l=find(rot,x); r=find(rot,x+y+1);
                                                             185
        splay(r,rot); splay(l,c[rot][0]);
                                                             186
        ans=sum[c[l][1]];
                                                             187
        printf("%d\n", ans);
                                                             188
                                                             189
    if (sign[0]=='M'&&sign[2]=='X')
                                                             190
                                                             191
    {
        splay(ed,rot); splay(st,c[rot][0]);
                                                             192
        ans=ma[c[st][1]];
                                                             193
        printf("%d\n", ans);
                                                             194
    }
                                                             195
                                                             196
return 0;
                                                             197
                                                             198
```

4.17 TriPartialOrder

目录

```
//三维偏序
                                                                   1
struct point{
                                                                   2
    int x,y,z,s;
}a[maxn],q1[maxn],q2[maxn];
                                                                   5
int f[maxn];
int n,m;
                                                                   6
struct bit
                                                                   7
                                                                   8
    int b[maxn];
                                                                   9
    void clear(int x){
                                                                   10
         for (int i=x;i;i-=(i&-i)) b[i]=0;
                                                                   11
                                                                   12
    void add(int x,int z){
                                                                   13
         for (int i=x;i;i==(i&-i)) b[i]=max(b[i],z);
                                                                   14
                                                                   15
    int ask(int x){
                                                                   17
        int tmp=0:
        for (int i=x;i<=n;i+=(i&-i)) tmp=max(b[i],tmp);</pre>
                                                                   18
         return tmp;
                                                                   19
                                                                   20
}s;
                                                                   21
                                                                   22
bool cmp1(point a, point b)
                                                                   23
                                                                   24
     return a.x>b.x;}
                                                                   25
bool cmp2(point a, point b)
                                                                   26
     return a.y>b.y;}
                                                                   27
                                                                   28
void solve(int l,int r){
                                                                   29
    if (l==r) return ;
                                                                   30
    int mid=(l+r)>>1:
                                                                   31
    solve(1, mid);
                                                                   32
    int t1=0, t2=0;
                                                                   33
    for (int i=1;i<=mid;i++) q1[++t1]=a[i];</pre>
                                                                   34
    for (int i=mid+1;i<=r;i++) q2[++t2]=a[i];</pre>
    sort(q1+1,q1+t1+1,cmp2);
                                                                   36
    sort(q2+1,q2+t2+1,cmp2);
                                                                   37
    int i=1;
                                                                   38
                                                                   39
    for (int j=1;j<=t2;j++){</pre>
         for (;q1[i].y>q2[j].y&&i<=t1;i++)</pre>
                                                                   40
             s.add(q1[i].z,f[q1[i].s]);
                                                                   41
         f[q2[j].s]=max(f[q2[j].s],s.ask(q2[j].z)+1);
                                                                   42
                                                                   43
    for (int i=1;i<=t1;i++)</pre>
                                                                   44
         s.clear(q1[i].z);
                                                                   45
    solve(mid+1,r);
                                                                   46
                                                                   47
```

```
48
49
    int main()
50
         //freopen("3sort.in", "r", stdin);
51
         scanf("%d",&n);
52
         for (int i=1;i<=n;i++)</pre>
53
              scanf("%d%d%d", &a[i].x, &a[i].y, &a[i].z), a[i].s=i;
54
55
         sort(a+1, a+n+1, cmp1);
         for (int i=1;i<=n;i++) f[i]=1;</pre>
56
57
         solve(1,n);
58
         int ans=0:
59
         for (int i=1;i<=n;i++) ans=max(ans,f[i]);</pre>
         printf("%d\n", ans);
60
         return 0;
61
62
    }
```

5 Graph

5.1 Floyd

```
//floyd最小环
    for (int i=1;i<=n;i++)</pre>
       for (int j=1; j<=n; j++)</pre>
         a[i][j]=d[i][j]=inf;
    for (int i=1;i<=m;i++)</pre>
      scanf("%d%d%d",&x,&y,&z);
8
       if (z<a[x][y])
10
         a[x][y]=a[y][x]=z;
11
         d[x][y]=d[y][x]=z;
      }
12
13
14
    ans=inf;
    for (int k=1;k<=n;k++)</pre>
15
17
      for (int i=1;i<k;i++)</pre>
         for (int j=i+1; j<k; j++)
18
           ans=min(ans, d[i][j]+a[j][k]+a[k][i]);
19
       for (int i=1;i<=n;i++)</pre>
20
21
         for (int j=1;j<=n;j++)</pre>
22
           d[i][j]=min(d[i][j],dis[i][k]+dis[k][j]);
23
```

5.2 HeapDij

```
//堆DJ
 1
    struct node
 2
 3
      int id.dis:
      node(){}
 5
 6
      node(int x,int y)
         id=x; dis=y;
 8
 9
       friend bool operator <(node a, node b)</pre>
10
11
12
         return a.dis>b.dis;
13
      }
14
    };
    priority_queue<node> q;
16
17
18
    void dij(int st)
19
20
      for (int i=1;i<=n;i++) dis[i]=inf;</pre>
21
      memset(vis, 0, sizeof(vis));
22
      while (!q.empty()) q.pop();
23
       q.push(node(st,0));
24
      dis[st]=0;
25
      while (!q.empty())
26
         int i=q.top().id;
27
28
         q.pop();
         if (vis[i]) continue;
29
30
         vis[i]=1;
31
         for (int j=fir[i];j;j=e[j].next)
32
33
           int k=e[j].t;
           if (!vis[k]&&dis[i]+e[j].val<dis[k])</pre>
34
35
```

```
dis[k]=dis[i]+e[j].val;
    q.push(node(k,dis[k]));
    }
}

}

40

41
```

5.3 Hungary

```
//匈牙利算法(二分图匹配)
bool match(int i)
  for (int j=fir[i];j;j=e[j].next)
                                                                 5
    int k=e[j].t;
    if (!vis[k])
                                                                 8
      vis[k]=1;
      if ((!mch[k])||(match(mch[k])))
                                                                 10
        mch[k]=i;
                                                                 12
        return 1;
                                                                 13
    }
                                                                 15
                                                                 16
  return 0;
                                                                 17
                                                                 18
                                                                 19
int hungary()
                                                                 20
                                                                 21
  int tmp=0;
                                                                 22
  memset(mch, 0, sizeof(mch));
                                                                 23
  for (int i=1;i<=n;i++)</pre>
                                                                 24
                                                                 25
    memset(vis, 0, sizeof(vis));
                                                                 26
    if (match(i)) tmp++;
                                                                 27
                                                                 28
  return tmp;
                                                                 29
```

5.4 LCA

```
//倍增LCA
void set_father(int n)
  for (int k=1;k<=low;k++)</pre>
    for (int i=1;i<=n;i++)</pre>
                                                                 5
      fa[i][k]=fa[fa[i][k-1]][k-1];
                                                                 8
int get_father(int x,int y)
                                                                 10
  if (dep[x]>dep[y]) swap(x,y);
                                                                 11
  for (int i=low;i>=0;i—)
                                                                 12
    if (dep[fa[y][i]]>=dep[x]) y=fa[y][i];
                                                                 13
  if (x==y) return x;
                                                                 14
  for (int i=low; i>=0; i---)
                                                                 15
    if (fa[x][i]!=fa[y][i]) x=fa[x][i],y=fa[y][i];
                                                                 16
  return fa[x][0];
                                                                 17
                                                                 18
```

5.5 Prim

```
//prim算法
void prim()
                                                                        3
     int ans=0;
     for (int i=1;i<=n;i++) d[i]=e[0][i],v[i]=0;</pre>
                                                                        5
     V[0]=1;
                                                                        6
     for (int i=1;i<=n;i++)</pre>
                                                                        8
         int min=inf,k;
         for (int j=1;j<=n;j++)</pre>
                                                                        10
              if (!v[j]&&d[j]<min)</pre>
                                                                        11
                                                                        12
                   min=d[j];
                                                                        13
                   k=j;
                                                                        14
                                                                        15
         v[k]=1;
                                                                        16
         ans+=min;
                                                                        17
         for (int j=1; j<=n; j++)
                                                                        18
         if (!v[j]&&e[k][j]<d[j])</pre>
                                                                        19
```

```
d[j]=e[k][j];
20
21
         printf("%d\n", ans);
22
23
    }
24
25
    //堆优化prim
26
27
    priority_queue<node> q;
28
29
    int prim(int n)
30
31
         int ans=0;
         memset(vis, 0, sizeof(vis));
32
33
         for (int i=0;i<=n;i++) dis[i]=inf;</pre>
34
         while (!q.empty())q.pop();
35
       q.push(node(0,0));
         dis[0]=0;
36
37
         while (!q.empty())
38
39
           int i=q.top().id;
40
           q.pop();
41
           if (vis[i]) continue;
           vis[i]=1;
42
43
           ans+=dis[i];
             for (int j=fir[i];j;j=e[j].next)
44
45
46
               int k=e[j].t;
47
                if (!vis[k] && e[j].val<dis[k])</pre>
48
49
             dis[k]=e[j].val;
50
             q.push(node(k,dis[k]));
51
52
53
54
         return ans;
    }
```

5.6 SAP _MaxFlow

HIT

```
//最大流sap
 1
 2
    struct edge
3
 4
      int s,t,val,next;
    }e[maxm];
 6
 7
    void add_edge(int x, int y, int z)
 8
      e[++tot].s=x; e[tot].t=y; e[tot].val=z; e[tot].next=fir[
9
       x]; fir[x]=tot;
      e[++tot].s=y; e[tot].t=x; e[tot].val=0; e[tot].next=fir[
10
      y]; fir[y]=tot;
12
    int dfs(int x, int flow)
13
14
      if (x==ed) return flow;
15
16
      int sap=0;
      for (int j=last[x];j;j=e[j].next)
17
18
19
         int y=e[j].t;
         if (e[j].val&&dis[x]==dis[y]+1)
20
21
22
           last[x]=j;
           int tmp=dfs(y, min(e[j].val, flow-sap));
23
24
           e[j].val-=tmp;
25
           e[j^1].val+=tmp;
           sap+=tmp;
26
27
           if (sap==flow) return sap;
28
        }
29
30
      if (dis[st]>=num) return sap;
      if (!(—gap[dis[x]])) dis[st]=num;
++gap[++dis[x]];
31
32
33
      last[x]=fir[x];
34
      return sap;
35
36
37
    int max_flow()
38
      gap[0]=num;
39
40
      memcpy(last, fir, sizeof(fir));
41
      int tmp=0;
      while (dis[st]<num) tmp+=dfs(st, inf);</pre>
42
```

```
return tmp;
                                                                   44
                                                                   45
void init()
                                                                   46
                                                                   47
  st=0; ed=2*n+m+p+1; num=ed+1; tot=1;
                                                                   48
 memset(fir, 0, sizeof(fir));
                                                                   49
 memset(last, 0, sizeof(last));
                                                                   50
  memset(dis, 0, sizeof(dis));
                                                                   51
 memset(gap, 0, sizeof(gap));
                                                                   52
                                                                   53
```

5.7 SPFA

```
//负环spfa
queue<int> q;
                                                                  3
bool neg_loop(int st)
                                                                  5
  while (!q.empty()) q.pop();
                                                                  6
  for (int i=0;i<=n;i++) dis[i]=inf;</pre>
  memset(times, 0, sizeof(times));
                                                                  8
  memset(vis, 0, sizeof(vis));
  a.push(st):
                                                                  10
  vis[st]=1;
                                                                  11
  times[st]=1;
                                                                  12
  dis[st]=0:
                                                                  13
  while (!q.empty())
                                                                  14
                                                                  15
    int i=q.front();
                                                                  16
    q.pop();
                                                                  17
    for (int j=fir[i];j;j=e[j].next)
                                                                  18
                                                                  19
      int k=e[j].t;
      if (dis[k]>dis[i]+e[j].val)
                                                                  21
                                                                  22
        dis[k]=dis[i]+e[j].val;
        if (!vis[k])
                                                                  24
                                                                  25
           q.push(k);
                                                                  26
          vis[k]=1;
                                                                  27
           times[k]++
                                                                  28
          if (times[k]>=n) return 1;
                                                                  29
                                                                  30
                                                                  31
                                                                  32
    vis[i]=0;
                                                                  33
                                                                  34
  return 0;
                                                                  35
```

5.8 SPFA CostFlow

```
//spfa费用流(min)
struct edge
                                                                 3
  int s, t, val, cost, next;
}e[maxm];
                                                                5
                                                                 6
void add_edge(int x, int y, int z, int w)
                                                                 8
  e[++tot].s=x; e[tot].t=y; e[tot].val=z; e[tot].cost=w; e
  [tot].next=fir[x], fir[x]=tot;
  e[++tot].s=y; e[tot].t=x; e[tot].val=0; e[tot].cost=-w;
                                                                 10
  e[tot].next=fir[y]; fir[y]=tot;
                                                                 11
                                                                 12
void init()
                                                                 13
                                                                 14
  st=0, ed=n*m+1; tot=1;
                                                                15
  memset(fir, 0, sizeof(fir));
                                                                16
  memset(dis, 0, sizeof(dis));
                                                                 17
                                                                18
                                                                 19
bool find_path()
                                                                 20
                                                                 21
  for (int i=st;i<=ed;i++) dis[i]=inf;//max</pre>
                                                                 22
  queue<int> q;
                                                                 23
  q.push(st);
                                                                24
  dis[st]=0;
                                                                25
  inque[st]=1;
                                                                 26
  while (!q.empty())
                                                                 27
```

```
28
      {
         int x=q.front();
29
30
         q.pop();
31
         inque[x]=0;
         for (int j=fir[x];j;j=e[j].next)
32
33
           int y=e[j].t;
34
           if (e[j].val && dis[y]>dis[x]+e[j].cost)//max
35
36
             dis[y]=dis[x]+e[j].cost;
37
38
             pre[y]=j;
39
             if (!inque[y]) q.push(y), inque[y]=1;
40
41
42
43
      return dis[ed]<inf;//max</pre>
44
45
46
    int fare_flow()
47
      int fare=0, flow=0;
48
49
      while (find_path())
50
         int tmp=inf;
51
         for (int j=pre[ed];j;j=pre[e[j].s]) tmp=min(tmp, e[j].
52
         val):
53
         for (int j=pre[ed];j;j=pre[e[j].s]) e[j].val=tmp, e[j
         ^1].val+=tmp;
         flow+=tmp;
54
         fare+=tmp*dis[ed];
55
56
      return fare;
57
    }
58
```

5.9 Simplex

```
//单纯形(复杂度不确定,慎用!!!)
1
    int a[maxn][maxm], next[maxm];
    int n,m;
5
    void pivot(int 1,int e)
6
7
         int last=-1;
8
         for (int i=0;i<=m;i++)</pre>
             if (a[l][i])
9
10
11
                  next[i]=last;
12
                 last=i;
13
         for (int i=0;i<=n;i++)</pre>
14
15
             if (a[i][e]==0||i==1) continue;
             for (int j=last;j!=-1;j=next[j])
17
18
                  //cout<<j<<' ';
                 if (j==e) continue;
20
21
                  a[i][j]=a[i][e]*a[l][j];
22
             //cout<<end1:
23
             a[i][e]=-a[i][e];
24
25
         }
26
    int simplex()
28
29
30
         while (1)
31
             int now=0;
32
33
             for (int i=1;i<=m;i++)</pre>
                  if (a[0][i]>0) { now=i; break; }
34
             if (now==0) return -a[0][0];
             int tmp, mi=inf;
36
37
             for (int i=1;i<=n;i++)</pre>
38
                  if (a[i][now]>0&&a[i][0]<mi)</pre>
39
40
                      tmp=i;
41
42
                      mi=a[i][0];
43
44
45
             pivot(tmp, now);
46
    }
```

```
int main()
                                                                     49
                                                                     50
    //freopen("defend.in", "r", stdin);
                                                                    51
    scanf("%d%d",&n,&m);
                                                                    52
    for (int i=1;i<=n;i++) scanf("%d",&a[i][0]);</pre>
                                                                    53
    int x, y;
                                                                    54
    for (int i=1;i<=m;i++)</pre>
                                                                    55
                                                                     56
         scanf("%d%d%d",&x,&y,&a[0][i]);
                                                                    57
         for (int j=x; j<=y; j++)
                                                                    58
             a[j][i]=1;
                                                                     59
                                                                    60
    int ans=simplex();
    printf("%d\n",ans);
                                                                    62
    return 0;
                                                                    63
```

5.10 Tarjan BCC

```
//tarjan割点
void tarjan(int now)
                                                                 2
                                                                 3
    dfn[now]=low[now]=++tim;
    v[now]=1;
                                                                 5
    for (int j=fir[now];j;j=e[j].next)
    int k=e[j].t;
                                                                 8
    if (!v[k])
                                                                 10
      tarjan(k);
      low[now]=min(low[now],low[k]);
                                                                 12
      if (dfn[now]<=low[k]) v[now]++;</pre>
                                                                 13
    else
                                                                 15
      low[now]=min(low[now],dfn[k]);
                                                                 16
  if ((dfn[now]==1&&v[now]>2)||(dfn[now]>1&&v[now]>1))
                                                                 18
    cut[now]=1;
                                                                 19
                                                                 20
//tarjan割边
                                                                 21
void tarjan(int i)
                                                                 22
                                                                 23
  vis[i]=1;
                                                                 24
  dfn[i]=low[i]=++cnt;
                                                                 25
  for (int j=fir[i];j;j=e[j].next)
                                                                 26
                                                                 27
    int k=e[j].t;
                                                                 28
    if (!vis[k])
                                                                 29
      pre[k]=j^1;
                                                                 31
      tarjan(k);
                                                                 32
      low[i]=min(low[i],low[k]);
                                                                 34
    else if (j!=pre[i])
                                                                 35
      low[i]=min(low[i],dfn[k]);
                                                                 36
                                                                 37
  if (pre[i] && dfn[i]==low[i]) bridge[pre[i]]=bridge[pre[
                                                                 38
  i]^1]=1:
                                                                 39
```

5.11 Tarjan SCC

```
//tarjan强连通分量
void tarjan(int now)
    dfn[now]=low[now]=++tim;
  stack[++top]=now;
                                                                5
    v[now]=1;
                                                                6
  for (int j=fir[now];j;j=e[j].next)
                                                                8
    int k=e[j].t;
    if (!v[k]) tarjan(k);
                                                                10
    if (v[k]<2) low[now]=min(low[now],low[k]);</pre>
                                                                11
                                                                 12
  if (dfn[now]==low[now])
                                                                13
                                                                14
                                                                 15
    while (stack[top+1]!=now)
                                                                16
                                                                17
      web[stack[top]]=num;
                                                                18
      v[stack[top--]]=2;
                                                                19
```

```
20
21
22
     }
```

ZKW CostFlow 5.12

HIT

```
//zkw最大费用流
2
    struct et
 3
         int s,t,val,cost,next;
 5
    }e[maxm];
    int fir[maxn], dis[maxn], v[maxn], q[maxm], pre[maxn];
    bool inque[maxn];
    int n,m,st,ed,tot,ans;
10
    void prepare()
11
         for (int i=st;i<=ed;i++) dis[i]=-inf;//min</pre>
12
13
         int head=0, tail=1;
14
         q[1]=ed; dis[ed]=0; inque[ed]=1;
        while (head<tail)</pre>
15
16
17
             int now=q[++head];
             for (int j=fir[now];j;j=e[j].next)
18
19
                 int k=e[i].t;
20
21
                 if (e[j^1].val&&dis[k]<dis[now]+e[j^1].cost)//
                 min
22
23
                     dis[k]=dis[now]+e[j^1].cost;
24
                     if (!inque[k]) q[++tail]=k,inque[k]=1;
25
26
27
             inque[now]=0;
28
         }
30
31
    int dfs(int now,int flow)
32
33
         if (now==ed)
34
35
             ans+=dis[st]*flow;
36
             return flow:
37
         int sap=0:
                        v[now]=1:
38
39
         for (int j=fir[now];j;j=e[j].next)
40
             int k=e[i].t;
41
             if (!v[k]&&e[j].val&&dis[now]==dis[k]+e[j].cost)
42
43
                 int tmp=dfs(k,min(e[j].val,flow-sap));
44
45
                 e[j].val—=tmp;
                 e[j^1].val+=tmp;
46
                 sap+=tmp;
47
                 if (sap==flow) return sap;
49
             }
50
         return sap;
51
52
    }
53
    bool adjust()
54
55
56
         int tmp=-inf;
         for (int i=st;i<=ed;i++) if (v[i])</pre>
57
58
             for (int j=fir[i];j;j=e[j].next)
59
                 int k=e[j].t;
60
                 if (!v[k]&&e[j].val) tmp=max(tmp,dis[k]+e[j].
                 cost-dis[i]);//min
62
63
         if (tmp==—inf) return 0;
         for (int i=st;i<=ed;i++) if (v[i])</pre>
64
65
             dis[i]+=tmp;
         return 1;
66
    }
67
68
    void add_edge(int x,int y,int z,int w)
69
70
         e[++tot].s=x; e[tot].t=y; e[tot].val=z; e[tot].cost=w;
          e[tot].next=fir[x]; fir[x]=tot;
72
         e[++tot].s=y; e[tot].t=x; e[tot].val=0; e[tot].cost=-w
         ; e[tot].next=fir[y]; fir[y]=tot;
73 }
```

```
int zkw_flow()
                                                                    75
                                                                   76
  ans=0:
                                                                   78
  prepare():
                                                                   79
  do{
    do memset(v, 0, sizeof(v));
                                                                   80
    while (dfs(st,inf));
                                                                   81
  }while (adjust());
                                                                   82
                                                                   83
                                                                   84
int init()
                                                                   85
                                                                   86
  memset(fir, 0, sizeof(fir));
                                                                   87
  tot=1; st=0; ed=n+2;
                                                                   88
                                                                   89
```

6 Geometry

6.1Geometry

```
//=====计算几何======
const double EPS=1e-10;
                                                              2
const double PI=acos(-1.0);
                                                              3
int dcmp(double x){ if (fabs(x)<EPS) return 0; else return</pre>
 x>0 ?1 :-1;}
                                                              6
//=====点与向量======
                                                              7
                                                              8
struct Point
                                                              9
                                                              10
  double x, y;
                                                              11
  Point(){}
                                                              12
  Point(double xx, double yy):x(xx), y(yy){}
                                                              13
  friend bool operator <(Point A, Point B)</pre>
                                                              14
                                                              15
    return A.x<B.x || (A.x==B.x && A.y<B.y);
                                                              16
                                                              17
  friend bool operator ==(Point A, Point B)
                                                              18
                                                              19
    return dcmp(A.x-B.x)==0 \&\& dcmp(A.y-B.y)==0;
                                                              20
                                                               21
                                                              22
typedef Point Vec;
                                                              23
typedef vector<Point> Points;
                                                               24
typedef vector<Point> Poly;
                                                              25
double AngleOnEarth(Point A, Point B)
                                                              26
                                                              27
  double x1=PI*A.x/180.0;
                                                              28
  double y1=PI*A.y/180.0;
                                                              29
  double x2=PI*B.x/180.0;
                                                              30
  double y2=PI*B.y/180.0;
                                                              31
  return acos( cos(x1-x2)*cos(y1)*cos(y2)+sin(y1)*sin(y2)
                                                              32
  );
                                                               33
                                                              34
//向量加法
                                                               35
Vec operator +(Vec a, Vec b){ return Vec(a.x+b.x, a.y+b.y)
                                                               36
                                                              37
//向量减法
                                                               38
Vec operator -(Point A, Point B){ return Vec(A.x-B.x, A.y-
                                                              39
B.y);}
                                                               40
//向量粉乘
                                                               41
Vec operator *(Vec a, double k){ return Vec(a.x*k, a.y*k)
                                                               42
                                                               43
//向量数除
Vec operator /(Vec a, double k){ return Vec(a.x/k, a.y/k)
                                                               45
                                                               46
//向量点积
                                                               47
double operator ^(Vec a, Vec b){ return a.x*b.x+a.y*b.y;}
                                                               48
                                                              49
//向量叉积
                                                              50
double operator *(Vec a, Vec b){ return a.x*b.y-a.y*b.x;}
                                                               51
                                                              52
//向量长度
                                                              53
double Len(Vec a) { return sqrt(a^a);}
                                                              54
```

55

```
56
     //向量极角
     double Ang(Vec a) { return atan2(a.y, a.x);}
 57
 58
 59
     //向量与向量夹角
 60
     double Ang(Vec a, Vec b) { return acos((a^b)/(Len(a)*Len(b
 61
     //三角形有向面积的两倍
 62
     double Area2(Point A, Point B, Point C) { return (B-A)*(C-
 63
 64
     //向量逆时针旋转rad度
 65
     Vec Rotate(Vec a, double rad)
 66
 67
       return Vec(a.x*cos(rad)—a.y*sin(rad), a.x*sin(rad)+a.y*
 68
       cos(rad));
 69
 70
     //向量的单位方向向量
 71
 72
     Vec Orient(Vec a)
 73
 74
       double l=Len(a);
 75
       return a/1;
 76
 77
     //向量的单位法向量
 78
 79
     Vec Normal(Vec a)
 80
       double l=Len(a);
 81
 82
       return Vec(-a.y/1, a.x/1);
 83
 84
     //凸包
 85
     Poly ConvexHull(Points P)
 86
 87
 88
       sort(P.begin(),P.end());
       P.erase(unique(P.begin(), P.end()), P.end());
 89
 90
       int n=P.size();
 91
       Poly St(n+1);
       int m=0;
 92
 93
       for (int i=0;i<n;i++)</pre>
 94
 95
         while (m>1 && dcmp((St[m-1]-St[m-2])*(P[i]-St[m-2]))
 96
           m—:
 97
         St[m++]=P[i];
 98
       int k=m:
 99
100
       for (int i=n-2;i>=0;i---)
101
         while (m>k && dcmp((St[m-1]-St[m-2])*(P[i]-St[m-2]))
102
         <=0)
103
           m---;
104
         St[m++]=P[i];
105
       if (n>1) St.resize(--m);
106
107
       return St;
108
109
     //旋转卡壳
110
     //t为卡壳上顶点, 1, r分别为左右顶点
111
112
     //D为最大直径,S为外接矩形的最小面积,C为最小周长
     void RotatingCalipers(Points &P, double &D, double &S,
113
     double &C)
114
115
       P=ConvexHull(P);
       int n=P.size();
116
117
       S=C=1e15, D=0;
       int t=1, l=1, r=1;
118
119
       for (int i=0;i<n;i++)</pre>
120
         while (dcmp( (P[(i+1)\%n]-P[i]) * (P[(t+1)\%n]-P[t]) ) >
121
          0) t=(t+1)%n;
         while (dcmp((P[(i+1)\%n]-P[i]) \land (P[(r+1)\%n]-P[r])) >
122
          0) r=(r+1)%n;
         if (i==0) l=t;
123
         while (dcmp( (P[(i+1)\%n]-P[i]) \land (P[(l+1)\%n]-P[l]) ) <
124
          0) l=(1+1)%n;
         double h=(P[(i+1)\%n]-P[i]) * (P[t]-P[i]) / Len(P[(i+1)
125
         %n]-P[i]);
         double W=(P[(i+1)\%n]-P[i]) \land (P[r]-P[l]) / Len(P[(i+1)
126
         %n]_P[i]);
127
         D=max(D,h);
```

```
S=min(S,w*h);
                                                                128
    C=min(C,(w+h)*2);
                                                                129
                                                                130
                                                                131
                                                                132
//多边形有向面积
                                                                133
double PolygonArea(Poly P)
                                                                134
                                                                135
  double area=0;
                                                                136
  int n=P.size();
                                                                137
  for (int i=1;i<n-1;i++)</pre>
                                                                138
    area+=(P[i]-P[0])*(P[i+1]-P[0]);
                                                                139
  return area/2;
                                                                140
                                                                141
                                                                142
//多边形周长
                                                                143
double PolygonPerimeter(Poly P)
                                                                144
                                                                145
  double len=0;
                                                                146
  int n=P.size();
                                                                147
  for (int i=0;i<n-1;i++)</pre>
                                                                148
    len+=Len(P[i+1]—P[i]);
                                                                149
  return len+Len(P[0]-P[n-1]);
                                                                150
                                                                151
                                                                152
//====直线与线段======
                                                                153
                                                                154
struct Line
                                                                155
                                                                156
 Point P;
                                                                157
  Vec v
                                                                158
  double ang;
                                                                159
  Line(){}
                                                                160
  Line(Point pp, Vec vv):P(pp), v(Orient(vv)), ang(Ang(vv)
                                                                161
  friend bool operator <(Line a, Line b)</pre>
                                                                162
                                                                163
    return a.ang<b.ang;
                                                                164
                                                                165
 Point point(double k)
                                                                166
                                                                167
    return P+v*k:
                                                                168
                                                                169
                                                                170
                                                                171
typedef vector<Line> Lines;
                                                                172
                                                                173
//直线交点(确保唯一)
                                                                174
Point GetIntersection(Line a, Line b)
                                                                175
                                                                176
  Point P=a.P, Q=b.P;
                                                                177
  Vec v=a.v, w=b.v;
                                                                178
  Vec u=P-Q;
                                                                179
  double t=(w^*u)/(v^*w);
                                                                180
  return a.point(t);
                                                                181
                                                                182
                                                                183
//点到直线距离
                                                                184
double DistanceToLine(Point P, Line 1)
                                                                185
                                                                186
  Point A=1.P, B=1.P+1.v;
                                                                187
 Vec u=B-A, v=P-A;
                                                                188
  //不取绝对值得到有向距离
                                                                189
  return fabs((u*v)/Len(u));
                                                                190
                                                                191
                                                                192
//点到线段距离
                                                                193
double DistanceToSeg(Point P, Point A, Point B)
                                                                194
                                                                195
  if (A==B) return Len(P-A);
                                                                196
  Vec u=B-A, v=P-A, w=P-B;
                                                                197
  if (dcmp(u^v)<0) return Len(v);
                                                                198
  else if (dcmp(u^w)>0) return Len(w);
                                                                199
  else return fabs((u*v)/Len(u));
                                                                200
                                                                201
                                                                202
//点在直线上投影
                                                                203
Point GetProjection(Point P, Line 1)
                                                                204
                                                                205
  Point A=1.P, B=1.P+1.v;
                                                                206
 Vec v=B-A;
                                                                207
  return A+v*((v^(P-A))/(v^v));
                                                                208
                                                                209
```

```
210
     //判断是否规范相交
211
     bool ProIntersection(Point A1, Point A2, Point B1, Point
212
     B2)
213
       double c1=(A2-A1)*(B1-A1);
214
       double c2=(A2-A1)*(B2-A1);
215
       double c3=(B2-B1)*(A1-B1);
216
       double c4=(B2-B1)*(A2-B1);
       return dcmp(c1)*dcmp(c2)<0 && dcmp(c3)*dcmp(c4)<0;
218
219
220
     //判断点在线段上
221
222
     bool OnSegment(Point P, Point A1, Point A2)
223
224
       return dcmp((A1-P)*(A2-P))==0 \&\& dcmp((A1-P)^(A2-P))<0;
225
226
     //点与多边形的位置关系
227
228
     int InPolygon(Point P, const Poly &Poly)
229
230
       int cnt=0, n=Poly.size();
       for (int i=0;i<n;i++)</pre>
231
232
         if (OnSegment(P,Poly[i], Poly[(i+1)%n])) return -1;
233
         int k=dcmp((Poly[(i+1)%n]-Poly[i])*(P-Poly[i]));
234
235
         int d1=dcmp(Poly[i].y-P.y);
236
         int d2=dcmp(Poly[(i+1)%n].y-P.y);
         if (k>0 && d1<=0 && d2>0) cnt++;
237
         if (k<0 && d2<=0 && d1>0) cnt--;
238
239
       if (cnt!=0) return 1;
240
       else return 0;
241
242
243
     //半平面交
245
     bool OnLeft(Point P, Line 1)
246
247
       return dcmp(1.v*(P-1.P))>=0;
248
     }
249
     Poly HalfPlaneIntersection(vector<Line> 1)
250
251
       Poly Sol;
252
       Sol.clear();
253
254
       int n=1.size();
255
       sort(1.begin(),1.end());
       int first,last;
256
       Point *p=new Point[n];
257
       Line *q=new Line[n];
258
259
       q[first=last=0]=1[0];
       for (int i=1;i<n;i++)</pre>
260
261
262
         while (first<last && !OnLeft(p[last-1],l[i])) last-</pre>
263
         while (first<last && !OnLeft(p[first],l[i])) first++;</pre>
264
         a[++last]=l[i]:
265
         if (dcmp(q[last].v*q[last-1].v)==0)
266
267
           last—:
           if (OnLeft(l[i].P,q[last])) q[last]=l[i];
268
269
270
         if (first<last) p[last-1]=GetIntersection(q[last-1], q</pre>
         [last]);
271
       while (first<last && !OnLeft(p[last-1],q[first])) last</pre>
272
       if (last-first<=1) return Sol;</pre>
273
274
       p[last]=GetIntersection(q[last], q[first]);
       for (int i=first;i<=last;i++) Sol.push_back(p[i]);</pre>
275
276
       return Sol;
277
     }
278
279
     //======圆相关======
280
     struct Circle
281
282
       Point 0;
283
284
       double r;
285
       Circle(Point 0, double r):0(0), r(r){}
       Point point(double rad)
286
287
288
         return Point(0.x+r*cos(rad), 0.y+r*sin(rad));
289
```

HIT

```
|};
                                                                 290
                                                                 291
  /圆和直线交点(方程法)
                                                                 292
 Points GetIntersection(Line L, Circle C, double &t1,
                                                                 293
 double &t2)
                                                                 294
   Points Sol; Sol.clear();
                                                                 295
   double a=L.v.x, b=L.P.x-C.0.x, c=L.v.y, d=L.P.y-C.0.y;
                                                                 296
   double e=a*a+c*c, f= 2*(a*b+c*d), g=b*b-C.r*C.r;
                                                                 297
   double delta=f*f-4*e*g;
                                                                 298
   if (dcmp(delta)<0) return Sol;</pre>
                                                                 299
   if (dcmp(delta)==0)
                                                                 300
                                                                 301
     t1=t2=-f/(2*e);
                                                                 302
     Sol.push_back(L.point(t1));
                                                                 303
                                                                 304
   else//相交
                                                                 305
                                                                 306
     t1=(-f-sqrt(delta))/(2*e);
                                                                 307
     Sol.push_back(L.point(t1));
                                                                 308
     t2=(-f+sqrt(delta))/(2*e);
                                                                 309
     Sol.push_back(L.point(t2));
                                                                 310
                                                                 311
   return Sol;
                                                                 312
                                                                 313
                                                                 314
 //圆和直线交点(几何法)
                                                                 315
 Points GetIntersection(Line L, Circle C)
                                                                 316
                                                                 317
   Points Sol; Sol.clear();
                                                                 318
   double d=DistanceToLine(C.0, L);
                                                                 319
   if (dcmp(d-C.r)>0) return Sol;//相离
                                                                 320
   Point ans=GetIntersection(L,Line(C.0,Normal(L.v)));
                                                                 321
   if (dcmp(d-C.r)==0)
                                                                 322
     Sol.push_back(ans);
                                                                 323
   else
                                                                 324
                                                                 325
     double len=sqrt(C.r*C.r-d*d);
                                                                 326
     Vec v=Orient(L.v);
                                                                 327
     Sol.push_back(ans+v*len);
                                                                 328
     Sol.push_back(ans-v*len);
                                                                 329
                                                                 330
   return Sol;
                                                                 331
                                                                 332
                                                                 333
 //圆与圆的交点
                                                                 334
 Points GetIntersection(Circle C, Circle D)
                                                                 335
                                                                 336
   Vec v=D.0-C.0;
                                                                 337
   Points Sol; Sol.clear();
                                                                 338
   double d=Len(v);
                                                                 339
   if (dcmp(C.r-D.r)>0) swap(C,D);
                                                                 340
   if (dcmp(d)==0 \mid | dcmp(C.r+D.r-d) < 0 \mid | dcmp(D.r-C.r-d)
                                                                 341
   >0) return Sol;//同心 | 外离 | 内含
   if (dcmp(C.r+D.r-d)==0 || dcmp(D.r-C.r-d)==0)//相切
                                                                 342
     Sol.push_back(C.O+Orient(v)*C.r);
                                                                 343
   else//相交
                                                                 344
                                                                 345
     double rad=Ang(v);
                                                                 346
     double th=acos((C.r*C.r+d*d-D.r*D.r)/(2*C.r*d));
                                                                 347
     Point P=C.point(rad+th);
                                                                 348
     Point Q=C.point(rad-th);
                                                                 349
     Sol.push_back(P);
                                                                 350
     Sol.push_back(Q);
                                                                 351
                                                                 352
   return Sol;
                                                                 353
                                                                 354
                                                                 355
 //过点P到圆C的切线
                                                                 356
 Lines GetTangents(Point P,Circle C)
                                                                 357
                                                                 358
   Vec v=C.O-P, tmp;
                                                                 359
   Lines Sol; Sol.clear();
                                                                 360
   double d=Len(v);
                                                                 361
   if (d<C.r) return Sol;</pre>
                                                                 362
      (dcmp(d-C.r)==0)
                                                                 363
     Sol.push_back(Line(P, Normal(v)));
                                                                 364
   else
                                                                 365
                                                                 366
     double rad=asin(C.r/d);
                                                                 367
     Sol.push_back(Line(P,Rotate(v,rad)));
                                                                 368
     Sol.push_back(Line(P,Rotate(v,-rad)));
                                                                 369
```

370

```
371
                    return Sol;
372
              }
373
              //两圆公切线
374
              Lines GetTangents(Circle C, Circle D)
375
376
377
                    Point A,B;
                    Lines Sol; Sol.clear();
378
                    if (dcmp(C.r-D.r)<0) swap(C, D);
379
                    double dd=(C.0.x-D.0.x)*(C.0.x-D.0.x)+(C.0.y-D.0.y)*(C.0.y-D.0.y)*(C.0.y-D.0.y)*(C.0.y-D.0.y)*(C.0.y-D.0.y)*(C.0.y-D.0.y)*(C.0.y-D.0.y)*(C.0.y-D.0.y)*(C.0.y-D.0.y)*(C.0.y-D.0.y)*(C.0.y-D.0.y)*(C.0.y-D.0.y)*(C.0.y-D.0.y)*(C.0.y-D.0.y)*(C.0.y-D.0.y)*(C.0.y-D.0.y)*(C.0.y-D.0.y)*(C.0.y-D.0.y)*(C.0.y-D.0.y)*(C.0.y-D.0.y)*(C.0.y-D.0.y)*(C.0.y-D.0.y)*(C.0.y-D.0.y)*(C.0.y-D.0.y)*(C.0.y-D.0.y)*(C.0.y-D.0.y)*(C.0.y-D.0.y)*(C.0.y-D.0.y)*(C.0.y-D.0.y)*(C.0.y-D.0.y)*(C.0.y-D.0.y)*(C.0.y-D.0.y)*(C.0.y-D.0.y)*(C.0.y-D.0.y)*(C.0.y-D.0.y)*(C.0.y-D.0.y)*(C.0.y-D.0.y)*(C.0.y-D.0.y)*(C.0.y-D.0.y)*(C.0.y-D.0.y)*(C.0.y-D.0.y)*(C.0.y-D.0.y)*(C.0.y-D.0.y)*(C.0.y-D.0.y)*(C.0.y-D.0.y)*(C.0.y-D.0.y)*(C.0.y-D.0.y)*(C.0.y-D.0.y)*(C.0.y-D.0.y)*(C.0.y-D.0.y)*(C.0.y-D.0.y)*(C.0.y-D.0.y)*(C.0.y-D.0.y)*(C.0.y-D.0.y)*(C.0.y-D.0.y)*(C.0.y-D.0.y)*(C.0.y-D.0.y)*(C.0.y-D.0.y)*(C.0.y-D.0.y)*(C.0.y-D.0.y)*(C.0.y-D.0.y)*(C.0.y-D.0.y)*(C.0.y-D.0.y)*(C.0.y-D.0.y)*(C.0.y-D.0.y)*(C.0.y-D.0.y)*(C.0.y-D.0.y)*(C.0.y-D.0.y)*(C.0.y-D.0.y)*(C.0.y-D.0.y)*(C.0.y-D.0.y)*(C.0.y-D.0.y)*(C.0.y-D.0.y)*(C.0.y-D.0.y)*(C.0.y-D.0.y)*(C.0.y-D.0.y)*(C.0.y-D.0.y)*(C.0.y-D.0.y)*(C.0.y-D.0.y)*(C.0.y-D.0.y)*(C.0.y-D.0.y)*(C.0.y-D.0.y)*(C.0.y-D.0.y)*(C.0.y-D.0.y)*(C.0.y-D.0.y)*(C.0.y-D.0.y)*(C.0.y-D.0.y)*(C.0.y-D.0.y)*(C.0.y-D.0.y)*(C.0.y-D.0.y)*(C.0.y-D.0.y)*(C.0.y-D.0.y)*(C.0.y-D.0.y)*(C.0.y-D.0.y)*(C.0.y-D.0.y)*(C.0.y-D.0.y)*(C.0.y-D.0.y)*(C.0.y-D.0.y)*(C.0.y-D.0.y)*(C.0.y-D.0.y)*(C.0.y-D.0.y)*(C.0.y-D.0.y)*(C.0.y-D.0.y)*(C.0.y-D.0.y)*(C.0.y-D.0.y)*(C.0.y-D.0.y)*(C.0.y-D.0.y)*(C.0.y-D.0.y)*(C.0.y-D.0.y)*(C.0.y-D.0.y)*(C.0.y-D.0.y)*(C.0.y-D.0.y)*(C.0.y-D.0.y)*(C.0.y-D.0.y)*(C.0.y-D.0.y)*(C.0.y-D.0.y)*(C.0.y-D.0.y)*(C.0.y-D.0.y)*(C.0.y-D.0.y)*(C.0.y-D.0.y)*(C.0.y-D.0.y)*(C.0.y-D.0.y)*(C.0.y-D.0.y)*(C.0.y-D.0.y)*(C.0.y-D.0.y)*(C.0.y-D.0.y)*(C.0.y-D.0.y)*(C.0.y-D.0.y)*(C.0.y-D.0.y)*(C.0.y-D.0.y)*(C.0.y-D.0.y)*(C.0.y-D.0.y)*(C.0.y-D.0.y)*(C.0.y-D.0.y)*(C.0.y-D.0.y)*(C.0.y-D.0.y)*(C.0.y-D.0.y)*(C.0.y-D.0.y)*(C.0.y-D.0.y)*(C.0.y-D.0.y)*(C.0.y-D.0.y)*(C.0.y-D.0.y)*(C.0.y-D.0.y)*(C
380
                      .y-D.O.y);
381
                    double rdif=C.r-D.r;
                    double rsum=C.r+D.r:
382
383
                    if (dcmp(dd-rdif*rdif)<0) return Sol;//内含
                    if (dcmp(dd)==0 && dcmp(C.r-D.r)==0) return Sol;//重合
384
                    double rad=atan2(D.0.y-C.0.y,D.0.x-C.0.x);
385
                    if (dcmp(dd-rdif*rdif)==0)//内切
386
387
388
                          A=C.point(rad), B=D.point(rad);
                          Sol.push_back(Line(A,B-A));
389
                          //Sol.push_back(Line(A, Normal(C.0-D.0)));
390
391
                          return Sol;
392
                    double th=acos((C.r-D.r)/sqrt(dd));
393
                    A=C.point(rad+th), B=D.point(rad+th);
394
                    Sol.push_back(Line(A,B-A));
395
396
                    A=C.point(rad-th), B=D.point(rad-th);
397
                    Sol.push_back(Line(A,B-A));
                    if (dcmp(dd-rsum*rsum)==0)//外切
398
399
                          A=C.point(rad), B=D.point(rad+PI);
Sol.push_back(Line(A,B-A));
400
401
                          //Sol.push_back(Line(A,Normal(D.O—C.O)));
402
403
                    else if (dcmp(dd-rsum*rsum)>0)//相离
404
405
                          double ro=acos((C.r+D.r)/sqrt(dd));
406
407
                          A=C.point(rad+ro), B=D.point(PI+rad+ro);
408
                          Sol.push_back(Line(A,B-A));
                          A=C.point(rad_ro), B=D.point(PI+rad_ro);
409
410
                          Sol.push_back(Line(A,B-A));
411
412
                    return Sol;
```

7 Others

7.1 Bitwise

```
//位运算
   while(mask<(1<< n))
   {//枚举大小为i的集合
3
     int x=mask&_mask, y=mask+x;
     int p=mask:
     {//枚举包含最低位的mask的子集p
8
       if ((p & x) >0)//保证该子集不被任何一个子集包含
10
11
       p=(p-1)&(mask);
12
13
     }while (p!=mask);
     mask=((mask \& ~y)/x >>1)|y;
14
   }
15
```

7.2 MoDui

```
//莫队算法
    #define inf 2147483647
    struct query
3
        int 1,r,s,w;
    }a[maxn];
    int c[maxn]:
    long long col[maxn], size[maxn], ans[maxn];
9
    int n,m,cnt,len;
    long long gcd(long long x, long long y)
11
12
13
        return (!x)?y:gcd(y%x,x);
   }
14
```

```
bool cmp(query a, query b)
                                                                    16
                                                                    17
    return (a.w==b.w)?a.r<b.r:a.w<b.w;
                                                                   18
}
                                                                   19
                                                                   20
int main()
                                                                   21
                                                                    22
    //freopen("hose.in","r",stdin);
                                                                    23
    scanf("%d%d",&n,&m);
                                                                    24
    for (int i=1;i<=n;i++) scanf("%d",&c[i]);</pre>
                                                                    25
    len=(int)sqrt(m);
                                                                    26
    cnt=(len*len==m)?len:len+1;
                                                                   27
    for (int i=1;i<=m;i++)</pre>
                                                                    28
                                                                    29
         scanf("%d%d",&a[i].1,&a[i].r);
                                                                    30
         if (a[i].l>a[i].r) swap(a[i].l,a[i].r);
                                                                    31
         size[i]=a[i].r-a[i].l+1;
                                                                    32
        a[i].w=a[i].l/len+1;
                                                                   33
         a[i].s=i;
                                                                    34
                                                                    35
    sort(a+1, a+m+1, cmp);
                                                                    36
    int i=1;
                                                                   37
    while (i<=m)
                                                                    38
                                                                    39
         int now=a[i].w;
                                                                    40
         memset(col, 0, sizeof(col));
                                                                    41
         for (int j=a[i].1;j<=a[i].r;j++) ans[a[i].s]+=2*(</pre>
                                                                    42
         col[c[j]]++);
         i++;
                                                                    43
         for (;a[i].w==now;i++)
                                                                    44
         {
                                                                    45
             ans[a[i].s]=ans[a[i-1].s];
                                                                    46
             for (int j=a[i-1].r+1;j<=a[i].r;j++)</pre>
                                                                    47
                 ans[a[i].s]+=2*(col[c[j]]++);
                                                                    48
             if (a[i-1].l<a[i].l)</pre>
                  for (int j=a[i-1].1; j< a[i].1; j++)
                                                                    50
                      ans[a[i].s]=2*(--col[c[j]]);
                                                                    51
                                                                   52
                 for (int j=a[i].l;j<a[i-1].l;j++)</pre>
                                                                   53
                      ans[a[i].s]+=2*(col[c[j]]++);
                                                                    54
        }
                                                                   55
                                                                    56
    long long all, num;
                                                                    57
    for (int i=1;i<=m;i++)</pre>
                                                                   58
                                                                    59
         if (size[i]==1) all=1; else all=size[i]*(size[i
                                                                    60
         num=gcd(ans[i],all);
                                                                    61
        printf("%lld/%lld\n", ans[i]/num, all/num);
                                                                    62
                                                                    63
    return 0;
                                                                    64
}
                                                                    65
```

7.3 OpenStack

7.4 PK

```
//对拍
:loop
datamaker.exe
a1.exe a2.exe
fc a1.out a2.out
if %errorlevel%==1 pause
goto loop
```

7.5 QuickRead

```
//快速读(慎用!!!)

const int BufferSize=1<<16;
char buffer[BufferSize],*head,*tail;
inline char Getchar() {

4
```

```
if(head==tail) {
 5
                 int l=fread(buffer,1,BufferSize,stdin);
tail=(head=buffer)+1;
 6
 7
 8
           return *head++;
 9
10
     inline int read() {
   int x=0, f=1; char c=Getchar();
   for(;!isdigit(c); c=Getchar()) if(c=='-') f=-1;
11
12
13
           for(;isdigit(c);c=Getchar()) x=x*10+c-'0';
return x*f;
14
15
      }
16
17
18
     inline LL read() {
   LL x=0,f=1;char c=getchar();
   for(;!isdigit(c);c=getchar()) if(c=='-') f=-1;
19
20
21
            for(;isdigit(c);c=getchar()) x=x*10+c-'0';
22
23
            return x*f;
     }
24
```