ACM/ICPC TEMPLATE

fz

Last build at July 18, 2015

此页为双面打印之留白页
打印需注意：选择双面打印，并缩放到可打印区域！

Contents

1 设置

1.1 头文件

```
#include<cstdio>
#include<cstdlib>
#include<cstring>
#include<cmath>
#include<ctime>
#include<cassert>
#include<climits>
#include<iostream>
#include<algorithm>
#include<string>
#include<vector>
#include<deque>
#include<list>
#include<set>
#include<map>
#include<stack>
#include<queue>
#include<numeric>
#include<iomanip>
#include<bitset>
#include<sstream>
#include<fstream>
#define debug puts("------")
#define pi (acos(-1.0))
#define eps (1e-8)
#define inf (1<<30)
#define INF (1ll<<62)
using namespace std;
int main()
{

    return 0;
}
```

1.2 Ubuntu 相关设置

CodeBlocks 终端命令行：
gnome-terminal –disable-factory -t $TITLE –x
计算器：
gnome-calculator
xcalc

2 STL

```
///欧几里得求最大公约数
__gcd(a,b);
///对[0,n)建堆
make_heap(a,a+n);
///将n-1插入到[0,n-1)的堆中
push_heap(a,a+n);
///将1从[0,n)的堆中取出到n
pop_heap(a,a+n);
///对[0,n)的堆进行排序
sort_heap(a,a+n);
```

```
11 ///翻转容器内元素
12 reverse(v.begin(),v.end());
13 ///对容器中的key计数
14 count(v.begin(),v.end(),key);
15 ///对容器中符合pred条件的元素计数
16 count_if(v.begin(),v.end(),pred),bool pred(type a);
17 ///删除容器中值为的key元素，返回end()迭代器
18 remove(v.begin(),v.end(),key);
19 v.erase(remove(v.begin(),v.end(),key),v.end());
20 ///删除容器中符合pred条件的元素，返回end()迭代器
21 remove_if(v.begin(),v.end(),pred),bool pred(type a);
22 ///对容器中元素排重，返回end()迭代器。符合pred的视为相同
23 unique(v.begin(),v.end());
24 unique(v.begin(),v.end(),pred),bool pred(type a,type b);
25 v.erase(unique(v.begin(),v.end()),v.end());
26 ///遍历容器找到值为key元素，返回迭代器，找不到则返回end()
27 find(v.begin(),v.end(),key);
28 ///生成下一个全排列，有则返回1，无则返回0
29 next_permutation(v.begin(),v.end());
30 ///生成上一个全排列，有则返回1，无则返回0
31 prev_permutation(v.begin(),v.end());
32 ///可以插入value，而不会破坏容器顺序的第一个位置（插入后其和其后数后移）。
       指向键值>=key的第一个元素
33 lower_bound(v.begin(),v.end(),key);
34 ///可以插入value，而不会破坏容器顺序的最后一个位置（插入后其和其后数后移）。
       指向键值>key的第一个元素
35 upper_bound(v.begin(),v.end(),key);
36 ///将容器内元素按pred条件分开，pred为1的在前，返回分开处end()迭代器
37 partition(v.begin(),v.end(),pred);
38 stable_partition(v.begin(),v.end(),pred);
39 ///求两个容器内积，返回sum+内积
40 inner_product(v1.begin(),v1.end(),v2.begin(),sum);
41 ///对容器内元素求和，返回sum+和
42 accumulate(v.begin(),v.end(),sum);
43 ///返回容器内元素的最大/最小值
44 max_element(v.begin(),v.end());
45 min_element(v.begin(),v.end());
46 ///求第n小的元素，并把它放在第n位置上
47 nth_element(v.begin(),v.end());
48 ///将容器内元素都赋值为key
49 fill(v.begin(),v.end(),key);
50 ///交换区间 [__first,__middle) [__middle,__last)
51 rotate(__first,__middle,__last);
52 ///将字符串以base进制转换成long，并通过ed得到一个char *
       指向后面未被识别的第一个字符
53 long strtol(const char *s, char **ed, int base);
54 ///将字符串str1用str2中的字符分隔开。第一次调用传入str1，以后传入NULL。
       全部分隔完时返回NULL
55 char *strtok(char *str1, const char *str2);
56 /**
57     char str[] = "now # is the $time for all # good men to co$$me to the # aid
          of the$ir country";
58     char delims[] = "#$";
59     char *result = NULL;
60     result = strtok(str, delims);
61     while(result != NULL)
62     {
63         printf("result is \"%s\"\n", result);
64         result = strtok(NULL, delims);
65     }
66     运行结果：
67     result is "now "
68     result is " is the "
```

```
    result is "time for all "
    result is " good men to co"
    result is "me to the "
    result is " aid of the"
    result is "ir country"
**/
```

bitset<n> b;	b 有 n 位，每位都为 0
bitset<n> b(u);	b 是 unsigned long 型 u 的一个副本
bitset<n> b(s);	b 是 string 对象 s 中含有的位串的副本
bitset<n> b(s, pos, n);	b 是 s 中从位置 pos 开始的 n 个位的副本

b.any()	b 中是否存在置为 1 的二进制位？
b.none()	b 中不存在置为 1 的二进制位吗？
b.count()	b 中置为 1 的二进制位的个数
b.size()	b 中二进制位的个数
b[pos]	访问 b 中在 pos 处的二进制位
b.test(pos)	b 中在 pos 处的二进制位是否为 1？
b.set()	把 b 中所有二进制位都置为 1
b.set(pos)	把 b 中在 pos 处的二进制位置为 1
b.reset()	把 b 中所有二进制位都置为 0
b.reset(pos)	把 b 中在 pos 处的二进制位置为 0
b.flip()	把 b 中所有二进制位逐位取反
b.flip(pos)	把 b 中在 pos 处的二进制位取反
b.to_ulong()	用 b 中同样的二进制位返回一个 unsigned long 值
os « b	把 b 中的位集输出到 os 流

3 基本算法

3.1 二分

```
///二分（单调增）
double l=0,r=100,mid;
while(r-l>eps)
{
    mid=(l+r)/2;
    if (f(mid)>0)
        r=mid;
    else
        l=mid;
}

///二分（精确binary_search）
int l=0,r=10000,mid;
while(l<=r)
{
    mid=l+r>>1;
    if (a[mid]>key)
        r=mid-1;
    else if (a[mid]<key)
        l=mid+1;
    else
        break;
}
```

```
if (a[mid]==key);

///二分（lower_bound）
int l=0,r=10000,mid;
while(l<r)
{
    mid=l+r>>1;
    if (a[mid]>=key)
        r=mid;
    else if (a[mid]<key)
        l=mid+1;
}
if (a[l]==key);
```

3.2 三分

```
///（极小值）
double left,right,mid1,mid2;
left=-1e7;
right=1e7;
while(fabs(left-right)>eps)
{
    mid1=(left+right)/2;
    mid2=(mid1+right)/2;
    if (f(mid1)<f(mid2))
        right=mid2;
    else
        left=mid1;
}
```

4 数据结构

4.1 离散化

```
int discrete(int data[],int n,int dis[],int idx[])
{
    int sub[n+1];
    memcpy(sub,data,sizeof(sub));
    sort(sub+1,sub+n+1);
    int m=unique(sub+1,sub+n+1)-sub-1;
    for(int i=1; i<=n; i++)
    {
        dis[i]=lower_bound(sub+1,sub+m+1,data[i])-sub;
        idx[dis[i]]=data[i];
    }
    return m;
}

const int NV=60005;
int m,data[NV];
int inithash(int n)
{
    sort(data+1,data+n+1);
    return unique(data+1,data+n+1)-data-1;
}
int hash(int x)
{
    return lower_bound(data+1,data+m+1,x)-data;
}
```

4.2 并查集

```
int f[LEN],rk[LEN];
int finds(int x) //非递归
{
    int k,j,r;
    r=x;
    while(r!=f[r])
        r=f[r];
    k=x;
    while(k!=r)
    {
        j=f[k];
        f[k]=r;
        k=j;
    }
    return r;
}
int finds(int x) //递归
{
    return f[x]==x?x:f[x]=finds(f[x]);
}
void uni(int a,int b)
{
    a=finds(a);
    b=finds(b);
    if (a==b) return;
    if (rk[a]>rk[b]) f[b]=a;
    else
    {
        if (rk[a]==rk[b]) rk[b]++;
        f[a]=b;
    }
}
void init(int n)
{
    memset(rk,0,sizeof(rk));
    for (int i=1; i<=n; i++)
        f[i]=i;
}
```

4.3 树状数组

4.3.1 一维树状数组

```
int N;
const int NV=500005;
int c[NV];
inline int lowbit(int t)
{
    return t&(-t);
}
void update(int x,int v)
{
    while(x<=N)
    {
        c[x]+=v;
        x+=lowbit(x);
    }
}
int query(int x)
```

```
{
    int ans=0;
    while(x>0)
    {
        ans+=c[x];
        x-=lowbit(x);
    }
    return ans;
}
int findkth(int k)
{
    int idx = 0;
    for(int i=20; i>=0; i--)
    {
        idx |= 1 << i;
        if(idx <= N && c[idx] < k)
            k -= c[idx];
        else idx ^= 1 << i;
    }
    return idx + 1;
}
```

4.3.2 一维树状数组区间更新区间查询

```
int N;
const int NV=500005;
long long c[NV],b[NV];
inline int lowbit(int t)
{
    return t&(-t);
}
void update(long long c[],int flag,int x,long long v)
{
    if (flag) for (int i=x; i<=N; i+=lowbit(i)) c[i]+=x*v;
    else for (int i=x; i>0; i-=lowbit(i)) c[i]+=v;
}
long long query(long long c[],int flag,int x)
{
    long long ans=0;
    if (flag) for (int i=x; i>0; i-=lowbit(i)) ans+=c[i];
    else for (int i=x; i<=N; i+=lowbit(i)) ans+=c[i];
    return ans;
}
void add(int l,int r,long long v)
{
    update(b,0,r,v);
    update(c,1,r,v);
    if (l>1)
    {
        update(b,0,l-1,-v);
        update(c,1,l-1,-v);
    }
}
long long sum(int x)
{
    if (x) return query(b,0,x)*x+query(c,1,x-1);
    else return 0;
}
long long sum(int l,int r)
{
    return sum(r)-sum(l-1);
}
```

4.3.3 多维树状数组的处理

```
int N;
int c[2005][2005];
inline int lowbit(int t)
{
    return t&(-t);
}
void update(int x,int y,int v)
{
    for (int i=x; i<=N; i+=lowbit(i))
        for (int j=y; j<=N; j+=lowbit(j))
            c[i][j]+=v;
}
int query(int x,int y)
{
    int s=0;
    for (int i=x; i>0; i-=lowbit(i))
        for (int j=y; j>0; j-=lowbit(j))
            s+=c[i][j];
    return s;
}
int sum(int x,int y,int xx,int yy)
{
    x--,y--;
    return query(xx,yy)-query(xx,y)-query(x,yy)+query(x,y);
}

int N;
long long c[130][130][130]= {};
inline int lowbit(int t)
{
    return t&(-t);
}
void update(int x,int y,int z,long long v)
{
    for (int i=x; i<=N; i+=lowbit(i))
        for (int j=y; j<=N; j+=lowbit(j))
            for (int k=z; k<=N; k+=lowbit(k))
                c[i][j][k]+=v;
}
long long query(int x,int y,int z)
{
    long long s=0;
    for (int i=x; i>0; i-=lowbit(i))
        for (int j=y; j>0; j-=lowbit(j))
            for (int k=z; k>0; k-=lowbit(k))
                s+=c[i][j][k];
    return s;
}
long long sum(int x,int y,int z,int xx,int yy,int zz)
{
    x--,y--,z--;
    return query(xx,yy,zz)
           -query(x,yy,zz)-query(xx,y,zz)-query(xx,yy,z)
           +query(x,y,zz)+query(xx,y,z)+query(x,yy,z)
           -query(x,y,z);
}
```

4.4 RMQ

4.4.1 一维 RMQ

```
const int NV=50005;
const int NVB=20;
int mx[NVB][NV],mn[NVB][NV],a[NV];
void init(int data[],int n)
{
    int k=log2(n);
    for (int i=1; i<=n; i++)
        mx[0][i]=mn[0][i]=data[i];
    for (int i=1; i<=k; i++)
        for (int j=1; j+(1<<i)-1<=n; j++)
        {
            mx[i][j]=max(mx[i-1][j],mx[i-1][j+(1<<i>>1)]);
            mn[i][j]=min(mn[i-1][j],mn[i-1][j+(1<<i>>1)]);
        }
}
int query(int l,int r,int flag)
{
    int k=log2(r-l+1);
    if (flag) return max(mx[k][l],mx[k][r-(1<<k)+1]);
    else return min(mn[k][l],mn[k][r-(1<<k)+1]);
}
```

4.4.2 二维 RMQ

```
const int NV=255;
const int NVB=20;
int mx[NVB][NV][NV],mn[NVB][NV][NV],a[NV][NV],mxt,mnt;
void init(int data[][NV],int n)
{
    for(int i=1; i<=n; i++)
        for(int j=1; j<=n; j++)
            mx[0][i][j]=mn[0][i][j]=data[i][j];
    for(int k=1; (1<<k)<=n; k++)
        for(int i=1; i<=n; i++)
            for(int j=1; j+(1<<k)-1<=n; j++)
            {
                mx[k][i][j]=max(mx[k-1][i][j],mx[k-1][i][j+(1<<k>>1)]);
                mn[k][i][j]=min(mn[k-1][i][j],mn[k-1][i][j+(1<<k>>1)]);
            }
}
///行，列，矩阵宽度
void query(int row,int col,int b)
{
    int k=log2(b);
    mxt=-1;
    mnt=inf;
    int l=col,r=col+b-1;
    for(int i=row; i<row+b; i++)
    {
        mxt=max(mxt,max(mx[k][i][l],mx[k][i][r-(1<<k)+1]));
        mnt=min(mnt,min(mn[k][i][l],mn[k][i][r-(1<<k)+1]));
    }
}
int main()
{
    int n,b,q;
    while(scanf("%d%d%d",&n,&b,&q)!=EOF)
```

```
34      {
35          for(int i=1; i<=n; i++)
36              for(int j=1; j<=n; j++)
37                  scanf("%d",&a[i][j]);
38          init(a,n);
39          while(q--)
40          {
41              int r,c;
42              scanf("%d%d",&r,&c);
43              query(r,c,b);
44              printf("%d\n",mxt-mnt);
45          }
46      }
47      return 0;
48  }
```

4.5 线段树

4.5.1 单点更新

```
1   #define lson l,m,rt<<1
2   #define rson m+1,r,rt<<1|1
3   const int NV = 100005;
4   int sum[NV<<2];
5   void PushUp(int rt)
6   {
7       sum[rt]=sum[rt<<1]+sum[rt<<1|1];
8   }
9   void build(int l,int r,int rt=1)
10  {
11      if (l == r)
12      {
13          sum[rt]=0;
14          return ;
15      }
16      int m = (l + r) >> 1;
17      build(lson);
18      build(rson);
19      PushUp(rt);
20  }
21  void update(int L,int c,int l,int r,int rt=1)
22  {
23      if (L == l && l == r)
24      {
25          sum[rt] += c;
26          return ;
27      }
28      int m = (l + r) >> 1;
29      if (L <= m) update(L , c , lson);
30      else update(L , c , rson);
31      PushUp(rt);
32  }
33  int query(int L,int R,int l,int r,int rt=1)
34  {
35      if (L <= l && r <= R)
36          return sum[rt];
37      int m = (l + r) >> 1;
38      int ret = 0;
39      if (L <= m) ret += query(L , R , lson);
40      if (m < R) ret += query(L , R , rson);
41      return ret;
```

```
}
```

4.5.2 区间更新

```
#define lson l,m,rt<<1
#define rson m+1,r,rt<<1|1
const int NV = 100005;
int add[NV<<2],sum[NV<<2];
void PushUp(int rt)
{
    sum[rt]=sum[rt<<1]+sum[rt<<1|1];
}
void PushDown(int rt,int m)
{
    if (add[rt])
    {
        add[rt<<1] += add[rt];
        add[rt<<1|1] += add[rt];
        sum[rt<<1] += add[rt] * (m - (m >> 1));
        sum[rt<<1|1] += add[rt] * (m >> 1);
        add[rt] = 0;
    }
}
void build(int l,int r,int rt=1)
{
    add[rt] = 0;
    if (l == r)
    {
        sum[rt]=0;
        return ;
    }
    int m = (l + r) >> 1;
    build(lson);
    build(rson);
    PushUp(rt);
}
void update(int L,int R,int c,int l,int r,int rt=1)
{
    if (L <= l && r <= R)
    {
        add[rt] += c;
        sum[rt] += c * (r - l + 1);
        return ;
    }
    PushDown(rt , r - l + 1);
    int m = (l + r) >> 1;
    if (L <= m) update(L , R , c , lson);
    if (m < R) update(L , R , c , rson);
    PushUp(rt);
}
int query(int L,int R,int l,int r,int rt=1)
{
    if (L <= l && r <= R)
    {
        return sum[rt];
    }
    PushDown(rt , r - l + 1);
    int m = (l + r) >> 1;
    int ret = 0;
    if (L <= m) ret += query(L , R , lson);
    if (m < R) ret += query(L , R , rson);
    return ret;
```

```
}
```

4.5.3 对一棵树进行线段树操作

```
const int NV=10005;
const int NE=NV;
int he[NV],ecnt;
struct edge
{
    int v,next;
} E[NE];
void adde(int u,int v)
{
    ecnt++;
    E[ecnt].v=v;
    E[ecnt].next=he[u];
    he[u]=ecnt;
}
int l[NV],r[NV],p;
void dfs(int u)
{
    p++;
    l[u]=p;
    for (int i=he[u]; i!=-1; i=E[i].next)
        dfs(E[i].v);
    r[u]=p;
}
void init()
{
    ecnt=p=0;
    memset(he,-1,sizeof(he));
}
```

4.5.4 二维线段树区间更新单点求和

```
const int NV=1005;
struct Nodey
{
    int l,r;
    int val;
};
int n;
int locx[NV],locy[NV];
struct Nodex
{
    int l,r;
    Nodey sty[NV*3];
    void build(int _l,int _r,int i=1)
    {
        sty[i].l = _l;
        sty[i].r = _r;
        sty[i].val = 0;
        if(_l == _r)
        {
            locy[_l] = i;
            return;
        }
        int mid = (_l + _r)>>1;
        build(_l,mid,i<<1);
        build(mid+1,_r,(i<<1)|1);
```

```
    }
    void add(int _l,int _r,int val,int i=1)
    {
        if(sty[i].l == _l && sty[i].r == _r)
        {
            sty[i].val += val;
            return;
        }
        int mid = (sty[i].l + sty[i].r)>>1;
        if(_r <= mid)add(_l,_r,val,i<<1);
        else if(_l > mid)add(_l,_r,val,(i<<1)|1);
        else
        {
            add(_l,mid,val,i<<1);
            add(mid+1,_r,val,(i<<1)|1);
        }
    }
} stx[NV*3];
void build(int l,int r,int i=1)
{
    stx[i].l = l;
    stx[i].r = r;
    stx[i].build(1,n);
    if(l == r)
    {
        locx[l] = i;
        return;
    }
    int mid = (l+r)>>1;
    build(l,mid,i<<1);
    build(mid+1,r,(i<<1)|1);
}
void add(int x1,int x2,int y1,int y2,int val,int i=1)
{
    if(stx[i].l == x1 && stx[i].r == x2)
    {
        stx[i].add(y1,y2,val);
        return;
    }
    int mid = (stx[i].l + stx[i].r)/2;
    if(x2 <= mid)add(x1,x2,y1,y2,val,i<<1);
    else if(x1 > mid)add(x1,x2,y1,y2,val,(i<<1)|1);
    else
    {
        add(x1,mid,y1,y2,val,i<<1);
        add(mid+1,x2,y1,y2,val,(i<<1)|1);
    }
}
int sum(int x,int y)
{
    int ret = 0;
    for(int i = locx[x]; i; i >>= 1)
        for(int j = locy[y]; j; j >>= 1)
            ret += stx[i].sty[j].val;
    return ret;
}
```

4.5.5 二维线段树单点更新区间求最大最小值

```
const int NV=1005;
struct Nodey
{
```

```
    int l,r;
    int Max,Min;
};
int locy[NV],locx[NV];
struct Nodex
{
    int l,r;
    Nodey sty[NV*4];
    void build(int _l,int _r,int i=1)
    {
        sty[i].l = _l;
        sty[i].r = _r;
        sty[i].Max = -inf;
        sty[i].Min = inf;
        if(_l == _r)
        {
            locy[_l] = i;
            return;
        }
        int mid = (_l + _r)/2;
        build(_l,mid,i<<1);
        build(mid+1,_r,(i<<1)|1);
    }
    int queryMin(int _l,int _r,int i=1)
    {
        if(sty[i].l == _l && sty[i].r == _r)
            return sty[i].Min;
        int mid = (sty[i].l + sty[i].r)/2;
        if(_r <= mid)return queryMin(_l,_r,i<<1);
        else if(_l > mid)return queryMin(_l,_r,(i<<1)|1);
        else return min(queryMin(_l,mid,i<<1),queryMin(mid+1,_r,(i<<1)|1));
    }
    int queryMax(int _l,int _r,int i=1)
    {
        if(sty[i].l == _l && sty[i].r == _r)
            return sty[i].Max;
        int mid = (sty[i].l + sty[i].r)/2;
        if(_r <= mid)return queryMax(_l,_r,i<<1);
        else if(_l > mid)return queryMax(_l,_r,(i<<1)|1);
        else return max(queryMax(_l,mid,i<<1),queryMax(mid+1,_r,(i<<1)|1));
    }
} stx[NV*4];
int n;
void build(int l,int r,int i=1)
{
    stx[i].l = l;
    stx[i].r = r;
    stx[i].build(1,n);
    if(l == r)
    {
        locx[l] = i;
        return;
    }
    int mid = (l+r)/2;
    build(l,mid,i<<1);
    build(mid+1,r,(i<<1)|1);
}
void modify(int x,int y,int val)
{
    int tx = locx[x];
    int ty = locy[y];
    stx[tx].sty[ty].Min = stx[tx].sty[ty].Max = val;
    for(int i = tx; i; i >>= 1)
```

```
67          for(int j = ty; j; j >>= 1)
68          {
69              if(i == tx && j == ty)continue;
70              if(j == ty)
71              {
72                  stx[i].sty[j].Min = min(stx[i<<1].sty[j].Min,stx[(i<<1)|1].sty[j
                        ].Min);
73                  stx[i].sty[j].Max = max(stx[i<<1].sty[j].Max,stx[(i<<1)|1].sty[j
                        ].Max);
74              }
75              else
76              {
77                  stx[i].sty[j].Min = min(stx[i].sty[j<<1].Min,stx[i].sty[(j<<1)
                        |1].Min);
78                  stx[i].sty[j].Max = max(stx[i].sty[j<<1].Max,stx[i].sty[(j<<1)
                        |1].Max);
79              }
80          }
81  }
82  int queryMin(int x1,int x2,int y1,int y2,int i=1)
83  {
84      if(stx[i].l == x1 && stx[i].r == x2)
85          return stx[i].queryMin(y1,y2);
86      int mid = (stx[i].l + stx[i].r)/2;
87      if(x2 <= mid)return queryMin(x1,x2,y1,y2,i<<1);
88      else if(x1 > mid)return queryMin(x1,x2,y1,y2,(i<<1)|1);
89      else return min(queryMin(x1,mid,y1,y2,i<<1),queryMin(mid+1,x2,y1,y2,(i<<1)
            |1));
90  }
91  int queryMax(int x1,int x2,int y1,int y2,int i=1)
92  {
93      if(stx[i].l == x1 && stx[i].r == x2)
94          return stx[i].queryMax(y1,y2);
95      int mid = (stx[i].l + stx[i].r)/2;
96      if(x2 <= mid)return queryMax(x1,x2,y1,y2,i<<1);
97      else if(x1 > mid)return queryMax(x1,x2,y1,y2,(i<<1)|1);
98      else return max(queryMax(x1,mid,y1,y2,i<<1),queryMax(mid+1,x2,y1,y2,(i<<1)
            |1));
99  }
```

4.5.6 线段树相关例题

单点增加，区间求和

```
1   #define lson l,m,rt<<1
2   #define rson m+1,r,rt<<1|1
3   const int NV = 100005;
4   int sum[NV<<2];
5   void PushUp(int rt)
6   {
7       sum[rt]=sum[rt<<1]+sum[rt<<1|1];
8   }
9   void build(int l,int r,int rt=1)
10  {
11      if (l == r)
12      {
13          sum[rt]=0;
14          return ;
15      }
16      int m = (l + r) >> 1;
17      build(lson);
18      build(rson);
```

```
    PushUp(rt);
}
void update(int L,int c,int l,int r,int rt=1)
{
    if (L == l && l == r)
    {
        sum[rt] += c;
        return ;
    }
    int m = (l + r) >> 1;
    if (L <= m) update(L , c , lson);
    else update(L , c , rson);
    PushUp(rt);
}
int query(int L,int R,int l,int r,int rt=1)
{
    if (L <= l && r <= R)
        return sum[rt];
    int m = (l + r) >> 1;
    int ret = 0;
    if (L <= m) ret += query(L , R , lson);
    if (m < R) ret += query(L , R , rson);
    return ret;
}
int main()
{
    int t,cas=0;
    cin>>t;
    while(t--)
    {
        int n;
        scanf("%d",&n);
        int x,v;
        build(1,n);
        for (int i=1; i<=n; i++)
        {
            scanf("%d",&v);
            update(i,v,1,n);
        }
        printf("Case %d:\n",++cas);
        char s[10];
        while(scanf("%s",s),s[0]!='E')
        {
            scanf("%d%d",&x,&v);
            if (s[0]=='A')
                update(x,v,1,n);
            if (s[0]=='S')
                update(x,-v,1,n);
            if (s[0]=='Q')
                printf("%d\n",query(x,v,1,n));
        }
    }
    return 0;
}
```

单点修改，区间求最值

```
#define lson l,m,rt<<1
#define rson m+1,r,rt<<1|1
const int NV = 200005;
int mx[NV<<2];
void PushUp(int rt)
{
    mx[rt]=max(mx[rt<<1],mx[rt<<1|1]);
```

```
}
void build(int l,int r,int rt=1)
{
    if (l == r)
    {
        scanf("%d",&mx[rt]);
        return ;
    }
    int m = (l + r) >> 1;
    build(lson);
    build(rson);
    PushUp(rt);
}
void update(int L,int c,int l,int r,int rt=1)
{
    if (L == l && l == r)
    {
        mx[rt] = c;
        return ;
    }
    int m = (l + r) >> 1;
    if (L <= m) update(L , c , lson);
    else update(L , c , rson);
    PushUp(rt);
}
int query(int L,int R,int l,int r,int rt=1)
{
    if (L <= l && r <= R)
        return mx[rt];
    int m = (l + r) >> 1;
    int ret = -1;
    if (L <= m) ret = max(ret,query(L , R , lson));
    if (m < R) ret = max(ret,query(L , R , rson));
    return ret;
}
int main()
{
    int n,m;
    while(scanf("%d%d",&n,&m)!=EOF)
    {
        build(1,n);
        char s[10];
        int a,b;
        while(m--)
        {
            scanf("%s%d%d",s,&a,&b);
            if (s[0]=='U')
                update(a,b,1,n);
            else
                printf("%d\n",query(a,b,1,n));
        }
    }
    return 0;
}
```

单点找最大值 $\geq$c 的第一个位置，同时单点更新

```
#define lson l,m,rt<<1
#define rson m+1,r,rt<<1|1
int h,w,n;
const int NV = 200005;
int mx[NV<<2];
void PushUp(int rt)
{
```

```
    mx[rt]=max(mx[rt<<1],mx[rt<<1|1]);
}
void build(int l,int r,int rt=1)
{
    if (l == r)
    {
        mx[rt]=w;
        return ;
    }
    int m = (l + r) >> 1;
    build(lson);
    build(rson);
    PushUp(rt);
}
void update(int c,int l,int r,int rt=1)
{
    if (l == r)
    {
        mx[rt]-=c;
        printf("%d\n",l);
        return;
    }
    int m = (l + r) >> 1;
    if (mx[rt<<1]>=c) update(c , lson);
    else update(c , rson);
    PushUp(rt);
}
int main()
{
    while(~scanf("%d%d%d",&h,&w,&n))
    {
        build(1,min(h,200000));
        for (int i=1; i<=n; i++)
        {
            int x;
            scanf("%d",&x);
            if (x<=mx[1]) update(x,1,min(h,200000));
            else puts("-1");
        }
    }
    return 0;
}
```

找第 K 大数，同时单点更新。（使用 sum 数组）

```
#define lson l,m,rt<<1
#define rson m+1,r,rt<<1|1
const int NV = 200005;
int sum[NV<<2];
void PushUp(int rt)
{
    sum[rt]=sum[rt<<1]+sum[rt<<1|1];
}
void build(int l,int r,int rt=1)
{
    if (l == r)
    {
        sum[rt]=1;
        return ;
    }
    int m = (l + r) >> 1;
    build(lson);
    build(rson);
    PushUp(rt);
```

```
}
int update(int c,int l,int r,int rt=1)
{
    if (l == r)
    {
        sum[rt]--;
        return l;
    }
    int m = (l + r) >> 1;
    int ret;
    if (sum[rt<<1]>=c) ret=update(c, lson);
    else ret=update(c-sum[rt<<1] , rson);
    PushUp(rt);
    return ret;
}
int a[200005];
int q[200005];
int w[200005];
int main()
{
    int n;
    while(scanf("%d",&n)!=EOF)
    {
        build(1,n);
        for (int i=1; i<=n; i++)
            scanf("%d%d",&q[i],&w[i]);
        for (int i=n; i>=1; i--)
            a[update(q[i]+1,1,n)]=w[i];
        for (int i=1; i<=n; i++)
            printf("%d%c",a[i]," \n"[i==n]);
    }
    return 0;
}
```

1、add x 表示往集合里添加数 x。2、del x 表示将集合中数 x 删除。3、sum 求出从小到大排列的集合中下标模 5 为 3 的数的和。集合中的数都是唯一的。
在线段树中维护当前这个集合中数的个数 sum，和所有的数模 5 为 $0\cdots4$ 内的数的和设为 ans$[0\cdots4]$。在进行区间合并的时候，父区间里的 ans$[0\cdots4]$ 首先等于左子区间里的 ans$[0\cdots4]$，设要加入的右子区间的数为 ans[i]，则它应该加到父区间的 ans[sum[rt«1]+i]。

```
#define lson l,m,rt<<1
#define rson m+1,r,rt<<1|1
const int NV = 100005;
int sum[NV<<2];
long long ans[NV<<2][5];
void PushUp(int rt)
{
    sum[rt]=sum[rt<<1]+sum[rt<<1|1];
    for (int i=0; i<5; i++)
        ans[rt][i]=ans[rt<<1][i]+ans[rt<<1|1][((i-sum[rt<<1])%5+5)%5];
}
void build(int l,int r,int rt=1)
{
    if (l == r)
    {
        sum[rt]=0;
        memset(ans[rt],0,sizeof(ans[rt]));
        return ;
    }
    int m = (l + r) >> 1;
    build(lson);
    build(rson);
    PushUp(rt);
```

```
}
void update(int L,int t,int c,int l,int r,int rt=1)
{
    if (L == l && l == r)
    {
        sum[rt]+=t;
        ans[rt][1]+=t*c;
        return ;
    }
    int m = (l + r) >> 1;
    if (L <= m) update(L, t , c , lson);
    else update(L, t , c , rson);
    PushUp(rt);
}
int discrete(int data[],int n,int dis[])
{
    int sub[n+1];
    memcpy(sub,data,sizeof(sub));
    sort(sub+1,sub+n+1);
    int m=unique(sub+1,sub+n+1)-sub-1;
    for(int i=1; i<=n; i++)
        dis[i]=lower_bound(sub+1,sub+m+1,data[i])-sub;
    return m;
}
int op[100005],data[100005],dis[100005];
int main()
{
    int n;
    while(~scanf("%d",&n))
    {
        char s[5];
        int m=0;
        for (int i=1; i<=n; i++)
        {
            scanf("%s",s);
            if (s[0]=='a') op[i]=1,scanf("%d",&data[++m]);
            else if (s[0]=='d') op[i]=2,scanf("%d",&data[++m]);
            else op[i]=3;
        }
        m=discrete(data,m,dis);
        build(1,m);
        int x=0;
        for (int i=1; i<=n; i++)
        {
            if (op[i]==1) x++,update(dis[x],1,data[x],1,m);
            else if (op[i]==2) x++,update(dis[x],-1,data[x],1,m);
            else printf("%I64d\n",ans[1][3]);
        }
    }
    return 0;
}
```

有三种操作 “add x y” 往平面上添加 (x,y) 这个点，“remove x y”，将平面上已经存在的点 (x,y) 删除，“find x y” 找出平面上坐标严格大于 (x,y) 的点，如果有多个点找 x 最小的，再找 y 最小的。
用线段树维护 y 的最大值。

```
#define lson l,m,rt<<1
#define rson m+1,r,rt<<1|1
const int NV = 200005;
set<int> stt[NV];
int mx[NV<<2];
void PushUp(int rt)
```

```
{
    mx[rt]=max(mx[rt<<1],mx[rt<<1|1]);
}
void build(int l,int r,int rt=1)
{
    if (l == r)
    {
        mx[rt]=-1;
        return ;
    }
    int m = (l + r) >> 1;
    build(lson);
    build(rson);
    PushUp(rt);
}
void add(int L,int c,int l,int r,int rt=1)
{
    if (l == r)
    {
        mx[rt] = max(mx[rt],c);
        stt[l].insert(c);
        return ;
    }
    int m = (l + r) >> 1;
    if (L <= m) add(L , c , lson);
    else add(L , c , rson);
    PushUp(rt);
}
void remove(int L,int c,int l,int r,int rt=1)
{
    if (l == r)
    {
        stt[l].erase(c);
        mx[rt]=stt[l].empty()?-1:*--stt[l].end();
        return ;
    }
    int m = (l + r) >> 1;
    if (L <= m) remove(L , c , lson);
    else remove(L , c , rson);
    PushUp(rt);
}
pair<int,int> find(int L,int c,int l,int r,int rt=1)
{
    if (mx[rt]<c||r<L) return make_pair(l,-1);
    if (l == r) return make_pair(l,*stt[l].lower_bound(c));
    int m = (l + r) >> 1;
    pair<int,int> p=find(L , c , lson);
    if (p.second!=-1) return p;
    else return find(L , c , rson);
}
int discrete(int data[],int n,int dis[],int index[])
{
    int sub[n+1];
    memcpy(sub,data,sizeof(sub));
    sort(sub+1,sub+n+1);
    int m=unique(sub+1,sub+n+1)-sub-1;
    for(int i=1; i<=n; i++)
    {
        dis[i]=lower_bound(sub+1,sub+m+1,data[i])-sub;
        index[dis[i]]=data[i];
    }
    return m;
}
```

```
int dis[NV],op[NV],x[NV],y[NV],index[NV];
int main()
{
    int n;
    cin>>n;
    for (int i=1; i<=n; i++)
    {
        char s[10];
        scanf("%s%d%d",s,&x[i],&y[i]);
        if (s[0]=='a') op[i]=1;
        else if (s[0]=='r') op[i]=2;
        else op[i]=3;
    }
    int m=discrete(x,n,dis,index);
    build(1,m);
    for (int i=1; i<=n; i++)
    {
        if (op[i]==1)
            add(dis[i],y[i],1,m);
        else if (op[i]==2)
            remove(dis[i],y[i],1,m);
        else
        {
            pair<int,int> p=find(dis[i]+1,y[i]+1,1,m);
            if (p.second==-1) puts("-1");
            else printf("%d %d\n",index[p.first],p.second);
        }
    }
    return 0;
}
```

区间置值，区间求和

```
#define lson l,m,rt<<1
#define rson m+1,r,rt<<1|1
const int NV = 100005;
int stt[NV<<2],sum[NV<<2];
void PushUp(int rt)
{
    sum[rt]=sum[rt<<1]+sum[rt<<1|1];
}
void PushDown(int rt,int m)
{
    if (stt[rt])
    {
        stt[rt<<1] = stt[rt];
        stt[rt<<1|1] = stt[rt];
        sum[rt<<1] = stt[rt] * (m - (m >> 1));
        sum[rt<<1|1] = stt[rt] * (m >> 1);
        stt[rt] = 0;
    }
}
void build(int l,int r,int rt=1)
{
    stt[rt] = 0;
    if (l == r)
    {
        sum[rt]=1;
        return ;
    }
    int m = (l + r) >> 1;
    build(lson);
    build(rson);
    PushUp(rt);
```

```
}
void update(int L,int R,int c,int l,int r,int rt=1)
{
    if (L <= l && r <= R)
    {
        stt[rt] = c;
        sum[rt] = c * (r - l + 1);
        return ;
    }
    PushDown(rt , r - l + 1);
    int m = (l + r) >> 1;
    if (L <= m) update(L , R , c , lson);
    if (m < R) update(L , R , c , rson);
    PushUp(rt);
}
int query(int L,int R,int l,int r,int rt=1)
{
    if (L <= l && r <= R)
    {
        return sum[rt];
    }
    PushDown(rt , r - l + 1);
    int m = (l + r) >> 1;
    int ret = 0;
    if (L <= m) ret += query(L , R , lson);
    if (m < R) ret += query(L , R , rson);
    return ret;
}
int main()
{
    int t;
    cin>>t;
    int cas=0;
    while(t--)
    {
        int n,q;
        cin>>n>>q;
        build(1,n);
        while(q--)
        {
            int l,r,c;
            scanf("%d%d%d",&l,&r,&c);
            update(l,r,c,1,n);
        }
        printf("Case %d: The total value of the hook is %d.\n",++cas,sum[1]);
    }
    return 0;
}
```

区间置值，单点询问

```
#define lson l,m,rt<<1
#define rson m+1,r,rt<<1|1
const int NV = 20005;
int col[NV<<2];
void PushDown(int rt,int m)
{
    if (col[rt])
    {
        col[rt<<1] = col[rt];
        col[rt<<1|1] = col[rt];
        col[rt] = 0;
    }
}
```

```
14 void build(int l,int r,int rt=1)
15 {
16     col[rt] = 0;
17     if (l == r)
18     {
19         return ;
20     }
21     int m = (l + r) >> 1;
22     build(lson);
23     build(rson);
24 }
25 void update(int L,int R,int c,int l,int r,int rt=1)
26 {
27     if (L <= l && r <= R)
28     {
29         col[rt] = c;
30         return ;
31     }
32     PushDown(rt , r - l + 1);
33     int m = (l + r) >> 1;
34     if (L <= m) update(L , R , c , lson);
35     if (m < R) update(L , R , c , rson);
36 }
37 int query(int L,int l,int r,int rt=1)
38 {
39     if (L == l && l == r)
40     {
41         return col[rt];
42     }
43     PushDown(rt , r - l + 1);
44     int m = (l + r) >> 1;
45     if (L <= m) return query(L , lson);
46     else return query(L , rson);
47 }
48 int discrete(int data[],int n,int dis[])
49 {
50     int sub[n+1];
51     memcpy(sub,data,sizeof(sub));
52     sort(sub+1,sub+n+1);
53     int m=unique(sub+1,sub+n+1)-sub-1;
54     for(int i=1; i<=n; i++)
55         dis[i]=lower_bound(sub+1,sub+m+1,data[i])-sub;
56     return m;
57 }
58 int a[NV],dis[NV],co[NV];
59 int main()
60 {
61     int t;
62     cin>>t;
63     while(t--)
64     {
65         int n;
66         scanf("%d",&n);
67         for (int i=1; i<=n; i++)
68             scanf("%d%d",&a[i],&a[i+n]);
69         discrete(a,2*n,dis);
70         build(1,2*n);
71         for (int i=1; i<=n; i++)
72             update(dis[i],dis[i+n],i,1,2*n);
73         memset(co,0,sizeof(co));
74         for (int i=1; i<=2*n; i++)
75             co[query(i,1,2*n)]=1;
76         printf("%d\n",accumulate(co+1,co+2*n+1,0));
```

```
    }
    return 0;
}
```

区间置值，区间查询有几种值

```
#define lson l,m,rt<<1
#define rson m+1,r,rt<<1|1
const int NV = 8005;
int stt[NV<<3];
bool a[NV][NV];
void PushDown(int rt,int m)
{
    if (stt[rt])
    {
        stt[rt<<1] = stt[rt];
        stt[rt<<1|1] = stt[rt];
        stt[rt] = 0;
    }
}
void build(int l,int r,int rt=1)
{
    stt[rt] = 0;
    if (l == r)
    {
        return ;
    }
    int m = (l + r) >> 1;
    build(lson);
    build(rson);
}
void update(int L,int R,int c,int l,int r,int rt=1)
{
    if (L <= l && r <= R)
    {
        stt[rt] = c;
        return ;
    }
    PushDown(rt , r - l + 1);
    int m = (l + r) >> 1;
    if (L <= m) update(L , R , c , lson);
    if (m < R) update(L , R , c , rson);
}
void query(int L,int R,int c,int l,int r,int rt=1)
{
    if (stt[rt])
    {
        a[stt[rt]][c]=1;
        return;
    }
    if (l == r)
    {
        return;
    }
    PushDown(rt , r - l + 1);
    int m = (l + r) >> 1;
    if (L <= m) query(L , R , c , lson);
    if (m < R) query(L , R , c , rson);
}
struct node
{
    int x,y1,y2;
} q[NV];
bool cmp(node a,node b)
```

```
{
    return a.x<b.x;
}
int main()
{
    int t;
    cin>>t;
    while(t--)
    {
        int n;
        scanf("%d",&n);
        for (int i=1; i<=n; i++)
        {
            scanf("%d%d%d",&q[i].y1,&q[i].y2,&q[i].x);
            q[i].y1*=2;
            q[i].y1++;
            q[i].y2*=2;
            q[i].y2++;
        }
        sort(q+1,q+1+n,cmp);
        build(1,16001);
        memset(a,0,sizeof(a));
        for (int i=1; i<=n; i++)
        {
            query(q[i].y1,q[i].y2,i,1,16001);
            update(q[i].y1,q[i].y2,i,1,16001);
        }
        int ans=0;
        for (int i=1; i<=n; i++)
            for (int j=1; j<=n; j++)
                if (a[i][j])
                    for (int k=1; k<=n; k++)
                        ans+=a[i][k]&&a[j][k];
        printf("%d\n",ans);
    }
    return 0;
}
```

有 N 根杆子，前后两根杆子通过一个节点连接，每个节点可以旋转一定的角度，每次给你一个操作（s,a）表示将第 S 与 S+1 之间的角度修改为 a 度，并且每次操作之后都需要求出第 N 个节点的位置。

首先，最后一个节点的坐标（即位置）可以通过所有杆子末尾的坐标相加得到（这跟向量相加很类似）。因为每次更新需要修改 S 和 S+1 的位置，所以我们需要知道每次更新之后 S 和 S+1 的角度——查询操作，知道之后，再根据这次需要修改的角度 a，得到 S+1 到最后一段杆子需要改变的角度——更新操作。最后输出答案即所有节点的 x 坐标和，y 坐标和。

add 为 ang 增加的 lazy 标记，ang 记录杆子末尾的角度，每次更新前 query 角度，就可以知道此次更新的角度的相对值，从而修改坐标 x,y。

```
#define lson l,m,rt<<1
#define rson m+1,r,rt<<1|1
const int NV = 10005;
double add[NV<<2],x[NV<<2],y[NV<<2],ang[NV<<2];
//ang不需要pushup，因为查询是单点查询
void PushUp(int rt)
{
    x[rt]=x[rt<<1]+x[rt<<1|1];
    y[rt]=y[rt<<1]+y[rt<<1|1];
}
void PushDown(int rt,int m)
{
    if (add[rt])
    {
```

```
15          ang[rt<<1] += add[rt];
16          ang[rt<<1|1] += add[rt];
17          add[rt<<1] += add[rt];
18          add[rt<<1|1] += add[rt];
19          double xx=x[rt<<1],yy=y[rt<<1];
20          x[rt<<1] = cos(add[rt])*xx-sin(add[rt])*yy;
21          y[rt<<1] = sin(add[rt])*xx+cos(add[rt])*yy;
22          xx=x[rt<<1|1],yy=y[rt<<1|1];
23          x[rt<<1|1] = cos(add[rt])*xx-sin(add[rt])*yy;
24          y[rt<<1|1] = sin(add[rt])*xx+cos(add[rt])*yy;
25          add[rt] = 0;
26      }
27  }
28  void build(int l,int r,int rt=1)
29  {
30      add[rt] = ang[rt] = 0;
31      if (l == r)
32      {
33          scanf("%lf",&y[rt]);
34          x[rt] = 0;
35  //        ang[rt] = pi/2; //此处可以不加，因为只需要相对值
36          return ;
37      }
38      int m = (l + r) >> 1;
39      build(lson);
40      build(rson);
41      PushUp(rt);
42  }
43  void update(int L,int R,double c,int l,int r,int rt=1)
44  {
45      if (L <= l && r <= R)
46      {
47          ang[rt] += c;
48          add[rt] += c;
49          double xx=x[rt],yy=y[rt];
50          x[rt] = cos(c)*xx-sin(c)*yy;
51          y[rt] = sin(c)*xx+cos(c)*yy;
52          return ;
53      }
54      PushDown(rt , r - l + 1);
55      int m = (l + r) >> 1;
56      if (L <= m) update(L , R , c , lson);
57      if (m < R) update(L , R , c , rson);
58      PushUp(rt);
59  }
60  double query(int L,int l,int r,int rt=1)
61  {
62      if (L == l && l == r)
63      {
64          return ang[rt];
65      }
66      PushDown(rt , r - l + 1);
67      int m = (l + r) >> 1;
68      if (L <= m) return query(L , lson);
69      else return query(L , rson);
70  }
71  int main()
72  {
73      int n,q;
74      while(scanf("%d%d",&n,&q)!=EOF)
75      {
76          build(1,n);
77          while(q--)
```

```
78          {
79              int s,d;
80              scanf("%d%d",&s,&d);
81              update(s+1,n,d/180.0*pi-(query(s+1,1,n)-query(s,1,n)+pi),1,n);
82              printf("%.2f %.2f\n",x[1],y[1]);
83          }
84          puts("");
85      }
86      return 0;
87  }
```

给定 n 长的序列 m 个操作
序列默认为 1, 2, 3...n
操作 1：D [l,r] 把 [l,r] 区间增长：（1,2,3,4 进行 D [1,3] 变成 1,1,2,2,3,3,4 ）
操作 2：Q [l,r] 问区间 [l,r] 上出现最多次数的数的次数
区间乘积，单点增加，区间求最值，找第 K 大数

```
1   #define lson l,m,rt<<1
2   #define rson m+1,r,rt<<1|1
3   const int NV = 50005;
4   long long mul[NV<<2],sum[NV<<2],mx[NV<<2];
5   void PushUp(int rt)
6   {
7       sum[rt]=sum[rt<<1]+sum[rt<<1|1];
8       mx[rt]=max(mx[rt<<1],mx[rt<<1|1]);
9   }
10  void PushDown(int rt,int m)
11  {
12      if (mul[rt]!=1)
13      {
14          mul[rt<<1] *= mul[rt];
15          mul[rt<<1|1] *= mul[rt];
16          sum[rt<<1] *= mul[rt];
17          sum[rt<<1|1] *= mul[rt];
18          mx[rt<<1] *= mul[rt];
19          mx[rt<<1|1] *= mul[rt];
20          mul[rt] = 1;
21      }
22  }
23  void build(int l,int r,int rt=1)
24  {
25      mul[rt] = 1;
26      if (l == r)
27      {
28          sum[rt]=1;
29          mx[rt]=1;
30          return ;
31      }
32      int m = (l + r) >> 1;
33      build(lson);
34      build(rson);
35      PushUp(rt);
36  }
37  void update(int L,long long c,int l,int r,int rt)
38  {
39      if (L == l && l == r)
40      {
41          sum[rt] += c;
42          mx[rt] += c;
43          return ;
44      }
45      PushDown(rt , r - l + 1);
46      int m = (l + r) >> 1;
```

```
47      if (L <= m) update(L , c , lson);
48      else update(L , c , rson);
49      PushUp(rt);
50  }
51  void update(int L,int R,long long c,int l,int r,int rt)
52  {
53      if (L <= l && r <= R)
54      {
55          mul[rt] *= c;
56          sum[rt] *= c;
57          mx[rt] *= c;
58          return ;
59      }
60      PushDown(rt , r - l + 1);
61      int m = (l + r) >> 1;
62      if (L <= m) update(L , R , c , lson);
63      if (m < R) update(L , R , c , rson);
64      PushUp(rt);
65  }
66  long long num;
67  int query(long long c,int l,int r,int rt)
68  {
69      if (l == r)
70      {
71          num=c;
72          return l;
73      }
74      PushDown(rt , r - l + 1);
75      int m = (l + r) >> 1;
76      if (sum[rt<<1]>=c) return query(c , lson);
77      else return query(c-sum[rt<<1] , rson);
78  }
79  long long query(int L,int R,int l,int r,int rt)
80  {
81      if (L <= l && r <= R)
82      {
83          return mx[rt];
84      }
85      PushDown(rt , r - l + 1);
86      int m = (l + r) >> 1;
87      long long ret = 0;
88      if (L <= m) ret = max(ret,query(L , R , lson));
89      if (m < R) ret = max(ret,query(L , R , rson));
90      return ret;
91  }
92  int main()
93  {
94      int t;
95      cin>>t;
96      int cas=0;
97      while(t--)
98      {
99          printf("Case #%d:\n",++cas);
100         int n,m;
101         scanf("%d%d",&n,&m);
102         build(1,n);
103         while(m--)
104         {
105             char str[5];
106             long long a,b;
107             scanf("%s%I64d%I64d",str,&a,&b);
108             long long mx=0,s1,s2,mx1=0,mx2=0;
109             int l=query(a,1,n,1);
```

```
            s1=num;
            mx1=query(l,l,1,n,1)-s1+1;
            l++;
            int r=query(b,1,n,1);
            s2=num;
            mx2=s2;
            r--;
            if (str[0]=='Q')
            {
                if (l<=r) mx=query(l,r,1,n,1);
                if (l-1!=r+1) mx=max(mx,max(mx1,mx2));
                else mx=b-a+1;
                printf("%I64d\n",mx);
            }
            else
            {
                if (l<=r) update(l,r,2,1,n,1);
                if (l-1==r+1)
                    update(l-1,b-a+1,1,n,1);
                else
                {
                    if (mx1) update(l-1,mx1,1,n,1);
                    if (mx2) update(r+1,mx2,1,n,1);
                }
            }
        }
    }
    return 0;
}
```

区间更新最值，单点查询最值

```
#define lson l,m,rt<<1
#define rson m+1,r,rt<<1|1
#define ll long long
const int NV = 55555;
long long stt[NV << 2], mn[NV << 2];
void PushUp(int rt)
{
    mn[rt] = min(mn[rt << 1], mn[rt << 1 | 1]);
}
void PushDown(int rt, int m)
{
    if (stt[rt] != inf)
    {
        stt[rt << 1] = min(stt[rt], stt[rt << 1]);
        stt[rt << 1 | 1] = min(stt[rt], stt[rt << 1 | 1]);
        mn[rt << 1] = min(mn[rt << 1], stt[rt << 1]);
        mn[rt << 1 | 1] = min(mn[rt << 1 | 1], stt[rt << 1 | 1]);
        stt[rt] = inf;
    }
}
void build(int l, int r, int rt = 1)
{
    stt[rt] = inf;
    if (l == r)
    {
        mn[rt] = inf;
        return ;
    }
    int m = (l + r) >> 1;
    build(lson);
    build(rson);
    PushUp(rt);
```

```
}
void update(int L, int R, long long c, int l, int r, int rt = 1)
{
    if (L <= l && r <= R)
    {
        stt[rt] = min(c, stt[rt]);
        mn[rt] = min(mn[rt], stt[rt]);
        return ;
    }
    PushDown(rt , r - l + 1);
    int m = (l + r) >> 1;
    if (L <= m) update(L , R , c , lson);
    if (m < R) update(L , R , c , rson);
    PushUp(rt);
}
ll query(int L, int l, int r, int rt = 1)
{
    if (L == l && l == r)
    {
        return mn[rt];
    }
    PushDown(rt , r - l + 1);
    int m = (l + r) >> 1;
    if (L <= m) return query(L , lson);
    else return query(L , rson);
}
ll mon[NV];
int day[NV], n;
int main()
{
    while(~scanf("%d", &n))
    {
        for (int i = 1; i <= n; i++) scanf("%lld", mon + i);
        for (int i = 1; i <= n; i++) scanf("%d", day + i);
        build(1, n, 1);
        ll get = 0;
        for (int i = 1; i <= n; i++)
        {
            update(i, min(i + day[i] - 1, n), get + mon[i], 1, n, 1);
            get = query(i, 1, n, 1);
            //cout<<get<<endl;
        }
        printf("%lld\n", get);
    }
    return 0;
}
```

4.6 左偏树

int finds(int x) 找到 x 的堆顶
int merge(int x, int y) 将以 x 和 y 为堆顶的两个堆合并，返回新堆顶编号
int pop(int x) 删除以 x 为堆顶的元素（merge 与 pop 都要用 finds 找到堆顶后才可操作）
void init(int n) 对左偏树初始化

```
struct node
{
    int l,r,dis,val,dad;
} heap[100005];
int finds(int x)
{
    return heap[x].dad == x ? x : heap[x].dad = finds (heap[x].dad);
```

```
}
int merge(int x, int y)
{
    if (!x) return y;
    if (!y) return x;
    if (heap[y].val > heap[x].val) swap (x, y);
    heap[x].r = merge (heap[x].r, y);
    heap[heap[x].r].dad = x;
    if (heap[heap[x].l].dis < heap[heap[x].r].dis)
        swap (heap[x].l, heap[x].r);
    if (heap[x].r == 0) heap[x].dis = 0;
    else heap[x].dis = heap[heap[x].r].dis + 1;
    return x;
}
int pop(int x)
{
    int l = heap[x].l;
    int r = heap[x].r;
    heap[l].dad = l;
    heap[r].dad = r;
    heap[x].l = heap[x].r = heap[x].dis = 0;
    return merge(l, r);
}
void init(int n)
{
    for (int i=1; i<=n; i++)
    {
        heap[i].l = heap[i].r = heap[i].dis = 0;
        heap[i].dad = i;
    }
}
int main()
{
    int n;
    while(cin>>n)
    {
        init(n);
        for (int i=1; i<=n; i++)
            scanf("%d",&heap[i].val);
        int m;
        cin>>m;
        while(m--)
        {
            int x,y;
            scanf("%d%d",&x,&y);
            x=finds(x);
            y=finds(y);
            if (x==y) puts("-1");
            else
            {
                heap[x].val>>=1;
                heap[y].val>>=1;
                x=merge(pop(x),x);//因为堆顶改变所以要重新pop再merge
                y=merge(pop(y),y);
                printf("%d\n",heap[merge(x,y)].val);
            }
        }
    }
    return 0;
}
```

4.7 划分树

在 $n\log n$ 的时间内离线处理一个序列，在 $\log n$ 的时间查询 $[l,r]$ 区间内的第 K 大数。

```
const int NV=100005;
int a[NV],sum[20][NV],s[20][NV];
void build(int l,int r,int rt=0)
{
    int mid=l+r>>1;
    int ln=l,rn=mid+1;
    int x=a[mid];
    sum[rt][l]=0;
    int cnt=mid-l+1;
    for (int i=l; i<=r; i++)
        if (s[rt][i]<x) cnt--;
    for (int i=l; i<=r; i++)
    {
        if (i>l) sum[rt][i]=sum[rt][i-1];
        if (ln<=mid&&(s[rt][i]<x||s[rt][i]==x&&cnt-->0))
            s[rt+1][ln++]=s[rt][i],sum[rt][i]++;
        else s[rt+1][rn++]=s[rt][i];
    }
    if (l<mid) build(l,mid,rt+1);
    if (mid+1<r) build(mid+1,r,rt+1);
}
int query(int k,int L,int R,int l,int r,int rt=0)
{
    if (l==r) return s[rt][l];
    int mid=l+r>>1;
    int lsum=0;
    if (L>l) lsum=sum[rt][L-1];
    int t=sum[rt][R]-lsum;
    if (t>=k) return query(k,l+lsum,l+sum[rt][R]-1,l,mid,rt+1);
    else return query(k-t,mid+1+L-l-lsum,mid+1+R-l-sum[rt][R],mid+1,r,rt+1);
}
void init(int n)
{
    for (int i=0; i<20; i++) sum[i][0]=0;
    for (int i=1; i<=n; i++) s[0][i]=a[i];
    sort(a+1,a+n+1);
    build(1,n);
}
```

4.8 Size Balanced Tree

```
const int NV=200005;
template<typename T>
struct SBT
{
    int sz[NV];
    T K[NV];
    int lc[NV];
    int rc[NV];
    int rt,tsz;
    void clear()
    {
        tsz=0;
        lc[0]=rc[0]=0;
        sz[0]=0;
        rt=0;
    }
    SBT()
```

```
18      {
19          clear();
20      }
21      int Size()
22      {
23          return sz[rt];
24      }
25      bool empty()
26      {
27          return rt==0;
28      }
29      void Build(int s,int e)
30      {
31          Build(rt,s,e);
32      }
33      bool Find(T key)
34      {
35          return Find(rt ,key);
36      }
37      void Insert(T key)
38      {
39          Insert(rt,key);
40      }
41      void Delete(T key)
42      {
43          Delete(rt,key);
44      }
45      T DeleteSelect(int k)
46      {
47          return DeleteSelect(rt,k);
48      }
49      void DeleteLess(T key)
50      {
51          DeleteLess(rt,key);
52      }
53      void DeleteGreater(T key)
54      {
55          DeleteGreater(rt,key);
56      }
57      int Rank(T key)
58      {
59          return Rank(rt,key);
60      }
61      T Select(int k)
62      {
63          return Select(rt,k);
64      }
65      T pred(T key)
66      {
67          return pred(rt,key);
68      }
69      T succ(T key)
70      {
71          return succ(rt,key);
72      }
73      void inorder()
74      {
75          inorder(rt);
76      }
77      T getMin()
78      {
79          int t=rt;
80          while (lc[t]) t=lc[t];
```

```
81          return K[t];
82      }
83      T getMax()
84      {
85          int t=rt;
86          while (rc[t]) t=rc[t];
87          return K[t];
88      }
89      T DeleteMax()
90      {
91          int t=rt;
92          if(rc[rt]==0)
93          {
94              rt=lc[rt];
95              return K[t];
96          }
97          while (rc[rc[t]])
98          {
99              sz[t]--;
100             t=rc[t];
101         }
102         sz[t]--;
103         T ret=K[rc[t]];
104         rc[t]=lc[rc[t]];
105         return ret;
106     }
107     T DeleteMin()
108     {
109         int t=rt;
110         if(lc[rt]==0)
111         {
112             rt=rc[rt];
113             return K[t];
114         }
115         while (lc[lc[t]])
116         {
117             sz[t]--;
118             t=lc[t];
119         }
120         sz[t]--;
121         T ret=K[lc[t]];
122         lc[t]=rc[lc[t]];
123         return ret;
124     }
125 private:
126     void Build(int &rt,int s,int e)
127     {
128         if(s>e) return;
129         int mid=(s+e)/2;
130         rt=++tsz;
131         K[rt]=mid;
132         lc[rt]=0;
133         rc[rt]=0;
134         sz[rt]=e-s+1;
135         if(s==e) return;
136         Build(lc[rt],s,mid-1);
137         Build(rc[rt],mid+1,e);
138     }
139     bool Find(int &rt,T key)
140     {
141         if (rt==0) return false;
142         else if (key<K[rt]) return Find(lc[rt],key);
143         else return K[rt]==key||Find(rc[rt],key);
```

```
144     }
145     void Insert(int &rt,T key)
146     {
147         if (rt==0)
148         {
149             rt=++tsz;
150             lc[rt]=rc[rt]=0;
151             sz[rt]=1;
152             K[rt]=key;
153             return;
154         }
155         sz[rt]++;
156         if (key<K[rt]) Insert(lc[rt],key);
157         else Insert(rc[rt],key);
158         maintain(rt,!(key<K[rt]));
159     }
160     T Delete(int &rt,T key)
161     {
162         sz[rt]--;
163         if ((K[rt]==key)||(key<K[rt]&&lc[rt]==0)||(K[rt]<key&&rc[rt]==0))
164         {
165             T ret=K[rt];
166             if (lc[rt]==0||rc[rt]==0)
167                 rt=lc[rt]+rc[rt];
168             else K[rt]=Delete(lc[rt],K[rt]+1);
169             return ret;
170         }
171         else
172         {
173             if (key<K[rt]) return Delete(lc[rt],key);
174             else return Delete(rc[rt],key);
175         }
176     }
177     void DeleteLess(int &rt,T key)
178     {
179         if (rt==0) return;
180         if (K[rt]<key)
181         {
182             rt=rc[rt];
183             DeleteLess(rt,key);
184         }
185         else
186         {
187             DeleteLess(lc[rt],key);
188             sz[rt]=1+sz[lc[rt]]+sz[rc[rt]];
189         }
190     }
191     void DeleteGreater(int &rt,T key)
192     {
193         if (rt==0) return;
194         if (K[rt]>key)
195         {
196             rt=lc[rt];
197             DeleteGreater(rt,key);
198         }
199         else
200         {
201             DeleteGreater(rc[rt],key);
202             sz[rt]=1+sz[lc[rt]]+sz[rc[rt]];
203         }
204     }
205     int Rank(int &rt,T key)
206     {
```

```
207         if (K[rt]==key) return sz[lc[rt]]+1;
208         else if (key<K[rt]) return Rank(lc[rt], key);
209         else return sz[lc[rt]]+1+Rank(rc[rt],key);
210     }
211     T Select(int &rt,int k)
212     {
213         if (sz[lc[rt]]+1==k) return K[rt];
214         else if (k>sz[lc[rt]]) return Select(rc[rt],k-1-sz[lc[rt]]);
215         else return Select(lc[rt],k);
216     }
217     T DeleteSelect(int &rt,int k)
218     {
219         sz[rt]--;
220         if (sz[lc[rt]]+1==k)
221         {
222             T ret=K[rt];
223             if (lc[rt]==0||rc[rt]==0) rt=lc[rt]+rc[rt];
224             else K[rt]=Delete(lc[rt],K[rt]+1);
225             return ret;
226         }
227         else if (k>sz[lc[rt]]) return DeleteSelect(rc[rt],k-1-sz[lc[rt]]);
228         else return DeleteSelect(lc[rt],k);
229     }
230     T pred(int &rt,T key)
231     {
232         if (rt==0) return key;
233         else if (key>K[rt])
234         {
235             T ret=pred(rc[rt],key);
236             if(ret==key) return K[rt];
237             return ret;
238         }
239         else return pred(lc[rt],key);
240     }
241     T succ(int &rt,T key)
242     {
243         if (rt==0) return key;
244         else if (K[rt]>key)
245         {
246             T ret=succ(lc[rt],key);
247             if (ret==key) return K[rt];
248             return ret;
249         }
250         else return succ(rc[rt],key);
251     }
252     void zag(int &rt)  //LeftRotate
253     {
254         int t=rc[rt];
255         rc[rt]=lc[t];
256         lc[t]=rt;
257         sz[t]=sz[rt];
258         sz[rt]=1+sz[lc[rt]]+sz[rc[rt]];
259         rt=t;
260     }
261     void zig(int &rt)  //RightRotate
262     {
263         int t=lc[rt];
264         lc[rt]=rc[t];
265         rc[t]=rt;
266         sz[t]=sz[rt];
267         sz[rt]=1+sz[lc[rt]]+sz[rc[rt]];
268         rt=t;
269     }
```

```
270     void maintain(int &rt,bool flag)
271     {
272         if (rt==0) return;
273         if (!flag)
274         {
275             if (sz[lc[lc[rt]]]>sz[rc[rt]]) zig(rt);
276             else if (sz[rc[lc[rt]]]>sz[rc[rt]])
277             {
278                 zag(lc[rt]);
279                 zig(rt);
280             }
281             else return;
282         }
283         else
284         {
285             if (sz[rc[rc[rt]]]>sz[lc[rt]]) zag(rt);
286             else if (sz[lc[rc[rt]]]>sz[lc[rt]])
287             {
288                 zig(rc[rt]);
289                 zag(rt);
290             }
291             else return;
292         }
293         maintain(lc[rt],false);
294         maintain(rc[rt],true);
295         maintain(rt,false);
296         maintain(rt,true);
297     }
298     void inorder(int &rt)
299     {
300         if(rt==0) return;
301         else
302         {
303             inorder(lc[rt]);
304             cout<<"rt:"<<rt<<" key:"<<K[rt]<<" size:"<<sz[rt]<<endl;
305             inorder(rc[rt]);
306         }
307     }
308 };
309 struct node
310 {
311     int key,id;
312     node(int key=0,int id=0):key(key),id(id) {}
313     bool operator <(node b)
314     {
315         return key<b.key;
316     }
317     bool operator >(node b)
318     {
319         return key>b.key;
320     }
321     bool operator ==(node b)
322     {
323         return key==b.key;
324     }
325     node operator +(int K)
326     {
327         return node(key+K,id);
328     }
329 };
```

4.9 可持久化线段树

4.9.1 可持久化线段树

```
#define lson l,m
#define rson m+1,r
const int NV = 100005;
const int NN = NV*25;
int sum[NN],ln[NN],rn[NN];
int tot,root[NV];
void PushUp(int k)
{
    sum[k]=sum[ln[k]]+sum[rn[k]];
}
int build(int l,int r)
{
    int k=++tot;
    if (l == r)
    {
        sum[k]=0;
        return k;
    }
    int m = (l + r) >> 1;
    ln[k]=build(lson);
    rn[k]=build(rson);
    PushUp(k);
    return k;
}
int update(int o,int L,int c,int l,int r)
{
    int k=++tot;
    sum[k]=sum[o];
    ln[k]=ln[o];
    rn[k]=rn[o];
    if (L == l && l == r)
    {
        sum[k]+=c;
        return k;
    }
    int m = (l + r) >> 1;
    if (L <= m) ln[k]=update(ln[o], L , c , lson);
    else rn[k]=update(rn[o], L , c , rson);
    PushUp(k);
    return k;
}
int query(int a,int b,int L,int l,int r)
{
    if (l==r) return l;
    int m = (l + r) >> 1;
    int tmp=sum[ln[b]]-sum[ln[a]];
    if (L <= tmp) return query(ln[a], ln[b], L , lson);
    return query(rn[a], rn[b], L-tmp , rson);
}
int main()
{
    root[0]=build(1,n);
    for (int i=1; i<=n; i++)
        root[i]=update(root[i-1],pos,val,1,n);
    query(root[a-1],root[b],pos,1,n);
    return 0;
}
```

4.9.2 树状数组套可持久化线段树

在 $\log n$ 的时间查询 $[l,r]$ 区间内的第 K 大数，同时支持更新操作。

```
const int NV=60005;
const int M=NV*40;
int N,m,tot;
int a[NV],T[NV],S[NV],use[NV];
int lson[M],rson[M],c[M];
int build(int l,int r)
{
    int rt = tot++;
    c[rt] = 0;
    if(l != r)
    {
        int mid = l+r>>1;
        lson[rt] = build(l,mid);
        rson[rt] = build(mid+1,r);
    }
    return rt;
}
int insert(int rt,int pos,int val)
{
    int nrt = tot++, tmp = nrt;
    int l = 1, r = m;
    c[nrt] = c[rt] + val;
    while(l < r)
    {
        int mid = l+r>>1;
        if(pos <= mid)
        {
            lson[nrt] = tot++;
            rson[nrt] = rson[rt];
            nrt = lson[nrt];
            rt = lson[rt];
            r = mid;
        }
        else
        {
            rson[nrt] = tot++;
            lson[nrt] = lson[rt];
            nrt = rson[nrt];
            rt = rson[rt];
            l = mid+1;
        }
        c[nrt] = c[rt] + val;
    }
    return tmp;
}
inline int lowbit(int x)
{
    return x&(-x);
}
int sum(int x)
{
    int ans = 0;
    while(x > 0)
    {
        ans += c[lson[use[x]]];
        x -= lowbit(x);
    }
    return ans;
}
void update(int x,int p,int d)
```

```
{
    while(x <= N)
    {
        S[x] = insert(S[x],p,d);
        x += lowbit(x);
    }
}
int query(int L,int R,int k)
{
    int lrt = T[L-1];
    int rrt = T[R];
    int l = 1, r = m;
    for(int i = L-1; i; i -= lowbit(i)) use[i] = S[i];
    for(int i = R; i ; i -= lowbit(i)) use[i] = S[i];
    while(l < r)
    {
        int mid = l+r>>1;
        int tmp = sum(R) - sum(L-1) + c[lson[rrt]] - c[lson[lrt]];
        if(tmp >= k)
        {
            r = mid;
            for(int i = L-1; i ; i -= lowbit(i))
                use[i] = lson[use[i]];
            for(int i = R; i; i -= lowbit(i))
                use[i] = lson[use[i]];
            lrt = lson[lrt];
            rrt = lson[rrt];
        }
        else
        {
            l = mid+1;
            k -= tmp;
            for(int i = L-1; i; i -= lowbit(i))
                use[i] = rson[use[i]];
            for(int i = R; i; i -= lowbit(i))
                use[i] = rson[use[i]];
            lrt = rson[lrt];
            rrt = rson[rrt];
        }
    }
    return l;
}
int op[NV],ql[NV],qr[NV],qk[NV],data[NV];
void inithash()
{
    sort(data+1,data+m+1);
    m=unique(data+1,data+m+1)-data;
}
int hash(int x)
{
    return lower_bound(data+1,data+m+1,x)-data;
}
void init(int n,int q)
{
    N=n;
    tot=0;
    m=0;
    for(int i = 1; i <= n; i++)
        scanf("%d",&a[i]),data[++m]=a[i];
    char s[10];
    for(int i = 1; i <= q; i++)
    {
        scanf("%s",s);
```

```
124         if(s[0] == 'Q')
125         {
126             op[i]=0;
127             scanf("%d%d%d",&ql[i],&qr[i],&qk[i]);
128         }
129         else
130         {
131             op[i]=1;
132             scanf("%d%d",&ql[i],&data[++m]);
133             qk[i]=data[m];
134         }
135     }
136     inithash();
137     T[0] = build(1,m);
138     for(int i = 1; i <= n; i++)
139         T[i] = insert(T[i-1],hash(a[i]),1);
140     for(int i = 1; i <= n; i++)
141         S[i] = T[0];
142 }
143 int main()
144 {
145     int t;
146     cin>>t;
147     while(t--)
148     {
149         int n,q;
150         scanf("%d%d",&n,&q);
151         init(n,q);
152         for(int i = 1; i <= q; i++)
153         {
154             if(op[i])
155             {
156                 update(ql[i],hash(a[ql[i]]),-1);
157                 update(ql[i],hash(qk[i]),1);
158                 a[ql[i]]=qk[i];
159             }
160             else printf("%d\n",data[query(ql[i],qr[i],qk[i])]);
161         }
162     }
163     return 0;
164 }
```

4.9.3 相关例题

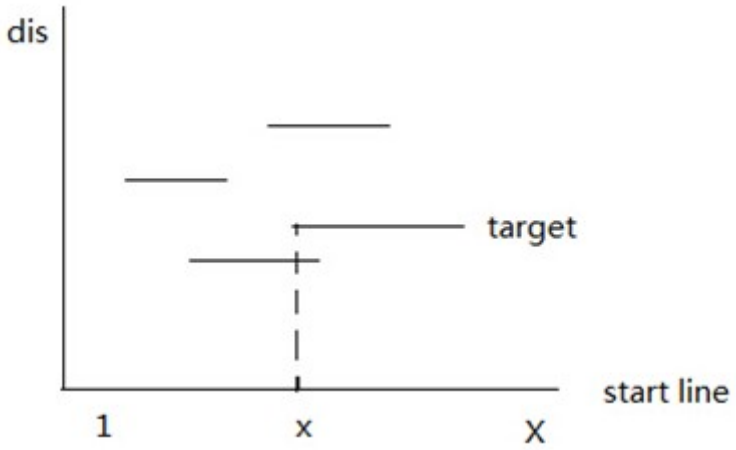
区间第 K 大数

```
1  #define lson l,m
2  #define rson m+1,r
3  const int NV = 100005;
4  const int NN = NV*25;
5  int sum[NN],ln[NN],rn[NN];
6  int tot,root[NV];
7  void PushUp(int k)
8  {
9      sum[k]=sum[ln[k]]+sum[rn[k]];
10 }
11 int build(int l,int r)
12 {
13     int k=++tot;
14     if (l == r)
15     {
16         sum[k]=0;
```

```
17          return k;
18      }
19      int m = (l + r) >> 1;
20      ln[k]=build(lson);
21      rn[k]=build(rson);
22      PushUp(k);
23      return k;
24  }
25  int update(int o,int L,int c,int l,int r)
26  {
27      int k=++tot;
28      sum[k]=sum[o];
29      ln[k]=ln[o];
30      rn[k]=rn[o];
31      if (L == l && l == r)
32      {
33          sum[k]+=c;
34          return k;
35      }
36      int m = (l + r) >> 1;
37      if (L <= m) ln[k]=update(ln[o], L , c , lson);
38      else rn[k]=update(rn[o], L , c , rson);
39      PushUp(k);
40      return k;
41  }
42  int query(int a,int b,int L,int l,int r)
43  {
44      if (l==r) return l;
45      int m = (l + r) >> 1;
46      int tmp=sum[ln[b]]-sum[ln[a]];
47      if (L <= tmp) return query(ln[a], ln[b], L , lson);
48      return query(rn[a], rn[b], L-tmp , rson);
49  }
50  int discrete(int data[],int n,int dis[],int idx[])
51  {
52      int sub[n+1];
53      memcpy(sub,data,sizeof(sub));
54      sort(sub+1,sub+n+1);
55      int m=unique(sub+1,sub+n+1)-sub-1;
56      for(int i=1; i<=n; i++)
57      {
58          dis[i]=lower_bound(sub+1,sub+m+1,data[i])-sub;
59          idx[dis[i]]=data[i];
60      }
61      return m;
62  }
63  int a[NV],dis[NV],idx[NV];
64  int main()
65  {
66      int n,m;
67      scanf("%d%d",&n,&m);
68      for (int i=1; i<=n; i++)
69          scanf("%d",&a[i]);
70      discrete(a,n,dis,idx);
71      root[0]=build(1,n);
72      for (int i=1; i<=n; i++)
73          root[i]=update(root[i-1],dis[i],1,1,n);
74      while(m--)
75      {
76          int aa,bb,k;
77          scanf("%d%d%d",&aa,&bb,&k);
78          printf("%d\n",idx[query(root[aa-1],root[bb],k,1,n)]);
79      }
```

```
80      return 0;
81  }
```

区间更新单点求和，求第 K 大，二分




```
1   #define lson l,m
2   #define rson m+1,r
3   const int NV = 200005;
4   const int NN = NV*25;
5   int cnt[NN],ln[NN],rn[NN];
6   long long sum[NN];
7   int tot,root[NV];
8   void PushUp(int k)
9   {
10      sum[k]=sum[ln[k]]+sum[rn[k]];
11      cnt[k]=cnt[ln[k]]+cnt[rn[k]];
12  }
13  int build(int l,int r)
14  {
15      int k=++tot;
16      if (l == r)
17      {
18          sum[k]=0;
19          cnt[k]=0;
20          return k;
21      }
22      int m = (l + r) >> 1;
23      ln[k]=build(lson);
24      rn[k]=build(rson);
25      PushUp(k);
26      return k;
27  }
28  int M,data[NV];
29  int update(int o,int L,int c,int l,int r)
30  {
31      int k=++tot;
32      sum[k]=sum[o];
33      cnt[k]=cnt[o];
34      ln[k]=ln[o];
35      rn[k]=rn[o];
36      if (L == l && l == r)
37      {
38          sum[k]+=c*data[l];
39          cnt[k]+=c;
40          return k;
```

```
41      }
42      int m = (l + r) >> 1;
43      if (L <= m) ln[k]=update(ln[o], L , c , lson);
44      else rn[k]=update(rn[o], L , c , rson);
45      PushUp(k);
46      return k;
47  }
48  long long query(int a,int L,int l,int r)
49  {
50      if (l==r) return (long long)data[l]*L; //注意不能是sum[a]
51      int m = (l + r) >> 1;
52      if (L <= cnt[ln[a]]) return query(ln[a], L , lson);
53      return sum[ln[a]]+query(rn[a], L-cnt[ln[a]] , rson);
54  }
55  int inithash(int n)
56  {
57      sort(data+1,data+n+1);
58      return unique(data+1,data+n+1)-data-1;
59  }
60  int hash(int x)
61  {
62      return lower_bound(data+1,data+M+1,x)-data;
63  }
64  struct node
65  {
66      int d,x;
67      node(int d=0,int x=0):d(d),x(x) {}
68      int sgn() const
69      {
70          if (d>0) return 1;
71          if (d==0) return 0;
72          return -1;
73      }
74      bool operator <(const node &b) const
75      {
76          if (x!=b.x) return x<b.x;
77          else if (sgn()!=b.sgn()) return sgn()>b.sgn();
78          return abs(d)<abs(b.d);
79      }
80  } a[NV];
81  int main()
82  {
83      int n,m,x,P;
84      while(~scanf("%d%d%d%d",&n,&m,&x,&P))
85      {
86          tot=0;
87          for (int i=1; i<=n; i++)
88          {
89              int l,r;
90              scanf("%d%d%d",&l,&r,&data[i]);
91              a[i].d=data[i];
92              a[i].x=l;
93              //在r处加负的，但在排序时把负的的全排到最后
94              //二分时，如果当前位置没有点，那么会找到x后面的一个位置
95              //如果有很多点，肯定会找到正数后面的一个位置（下个x或当前x但为负的点
                    ）
96              //最后位置减1即可
97              a[i+n].d=-data[i];
98              a[i+n].x=r;
99          }
100         sort(a+1,a+2*n+1);
101         M=inithash(n);
102         root[0]=build(1,M);
```

```
        for (int i=1; i<=2*n; i++)
        {
            int d=a[i].d;
            if (d>0) root[i]=update(root[i-1],hash(d),1,1,M);
            else root[i]=update(root[i-1],hash(-d),-1,1,M);
        }
        long long pre=1;
        while(m--)
        {
            int X,aa,bb,cc;
            scanf("%d%d%d%d",&X,&aa,&bb,&cc);
            int k=(aa%cc*pre%cc+bb)%cc;
            int p=lower_bound(a+1,a+2*n+1,node(0,X))-a-1;
            long long ans;
            if (k) ans=query(root[p],k,1,M);
            else ans=0;
            if (pre>P) ans*=2;
            pre=ans;
            printf("%I64d\n",ans);
        }
    }
    return 0;
}
```

4.10 splay

```
#define keyTree (ch[ch[rt][1]][0])
const int NV=100005;
struct SplayTree
{
    int sz[NV];
    int ch[NV][2];
    int pre[NV];
    int rt,top,tops;
    int stk[NV],que[NV];
    SplayTree():rt(0),top(0) {}
    void Rotate(int x,int c)
    {
        int y=pre[x];
        push_down(y);
        push_down(x);
        ch[y][!c]=ch[x][c];
        pre[ch[x][c]]=y;
        pre[x]=pre[y];
        if(pre[x]) ch[pre[y]][ch[pre[y]][1]==y]=x;
        ch[x][c]=y;
        pre[y]=x;
        push_up(y);
    }
    void Splay(int x,int f)
    {
        if (x==f) return;
        push_down(x);
        while(pre[x]!=f)
        {
            if(pre[pre[x]]==f)
                Rotate(x,ch[pre[x]][0]==x);
            else
            {
                int y=pre[x],z=pre[y];
                int t=(ch[z][0]==y);
                if(ch[y][t]==x) Rotate(x,!t),Rotate(x,t);
```

```
37                  else Rotate(y,t),Rotate(x,t);
38              }
39          }
40          push_up(x);
41          if(f==0) rt=x;
42      }
43      void Select(int k,int f)  //把第k位的数转到f下边
44      {
45          int x=rt;
46          push_down(x);
47          while(sz[ch[x][0]]!=k)
48          {
49              if(k<sz[ch[x][0]]) x=ch[x][0];
50              else
51              {
52                  k-=(sz[ch[x][0]]+1);
53                  x=ch[x][1];
54              }
55              push_down(x);
56          }
57          Splay(x,f);
58      }
59      void Erase(int x)  //把以x为祖先结点删掉放进内存池，回收内存
60      {
61          int father=pre[x];
62          int head=0,tail=0;
63          for (que[tail++]=x; head<tail; head++)
64          {
65              stk[tops++]=que[head];
66              if(ch[que[head]][0]) que[tail++]=ch[que[head]][0];
67              if(ch[que[head]][1]) que[tail++]=ch[que[head]][1];
68          }
69          ch[father][ch[father][1]==x]=0;
70          push_up(father);
71      }
72      int Find(int c)
73      {
74          int x=rt;
75          while(x)
76          {
77              if (c==val[x])
78              {
79                  Splay(x,rt);
80                  return x;
81              }
82              else if (c<val[x]) x=ch[x][0];
83              else x=ch[x][1];
84          }
85          return 0;
86      }
87      void Insert(int c)
88      {
89          int p=rt;
90          int *t=&p;
91          int f=0;
92          while(*t)
93          {
94              f=*t;
95              if (c<val[*t]) t=&ch[*t][0];
96              else t=&ch[*t][1];
97          }
98          int &x=*t;
99          if (tops) x=stk[--tops];
```

```
100         else x=++top;
101         ch[x][0]=ch[x][1]=pre[x]=0;
102         sz[x]=1;
103         val[x]=sum[x]=c;
104         pre[x]=f;
105         while(f)
106         {
107             push_up(f);
108             f=pre[f];
109         }
110         Splay(x,rt);
111     }
112     void Debug()
113     {
114         cout<<rt<<endl;
115         Inorder(rt);
116     }
117     void Inorder(int x)
118     {
119         if(x)
120         {
121             Inorder(ch[x][0]);
122             printf("结点%2d:左儿子 %2d 右儿子 %2d 父结点 %2d size=%2d ,val=%2d\n
                    ",x,ch[x][0],ch[x][1],pre[x],sz[x],val[x]);
123             Inorder(ch[x][1]);
124         }
125     }
126
127     //以下是题目的特定函数
128     void newnode(int &x,int c)
129     {
130         if (tops) x=stk[--tops];
131         else x=++top;
132         ch[x][0]=ch[x][1]=pre[x]=0;
133         sz[x]=1;
134         ///题目特定
135         val[x]=sum[x]=c;
136         add[x]=0;
137     }
138     void push_down(int x) ///题目特定
139     {
140         if(add[x])
141         {
142             val[x]+=add[x];
143             add[ch[x][0]]+=add[x];
144             add[ch[x][1]]+=add[x];
145             sum[ch[x][0]]+=(long long)sz[ch[x][0]]*add[x];
146             sum[ch[x][1]]+=(long long)sz[ch[x][1]]*add[x];
147             add[x]=0;
148         }
149     }
150     void push_up(int x)
151     {
152         sz[x]=1+sz[ch[x][0]]+sz[ch[x][1]];
153         ///题目特定
154         sum[x]=val[x]+sum[ch[x][0]]+sum[ch[x][1]];
155     }
156     void build(int &x,int l,int r,int f)
157     {
158         if(l>r) return;
159         int m=(l+r)>>1;
160         newnode(x,num[m]); ///题目特定
161         build(ch[x][0],l,m-1,x);
```

```
        build(ch[x][1],m+1,r,x);
        pre[x]=f;
        push_up(x);
    }
    void init(int n)
    {
        rt=top=0;
        //为了方便处理边界，加两个边界顶点
        newnode(rt,-1);
        newnode(ch[rt][1],-1);
        pre[top]=rt;
        sz[rt]=2;
        build(keyTree,1,n,ch[rt][1]);
        push_up(ch[rt][1]);
        push_up(rt);
    }
    void update(int l,int r,int c)
    {
        Select(l-1,0);
        Select(r+1,rt);
        ///题目特定
        add[keyTree]+=c;
        sum[keyTree]+=(long long)c*sz[keyTree];
    }
    long long query(int l,int r)
    {
        Select(l-1,0);
        Select(r+1,rt);
        return sum[keyTree];
    }
    ///题目特定变量
    int num[NV];
    int val[NV];
    int add[NV];
    long long sum[NV];
} sp;
int main()
{
    sp.Insert(5);
    sp.Insert(8);
    sp.Insert(2);
    sp.Insert(8);
    sp.Insert(6);
    sp.Insert(1);
    sp.Debug();
    cout<<sp.Find(8)<<endl;
    cout<<sp.Find(2)<<endl;
    cout<<sp.Find(5)<<endl;
    return 0;
}
```

4.11 KD-tree

离线处理一堆点，在线询问一个点到这一堆点最近的是哪个点。

4.11.1 二维 KD-tree

```
const int NV=100005;
struct point
{
    int x, y;
```

```
    void in()
    {
        scanf("%d%d",&x,&y);
    }
} p[NV];
bool Div[NV];
bool cmpX(const point &p1,const point &p2)
{
    return p1.x<p2.x;
}
bool cmpY(const point &p1,const point &p2)
{
    return p1.y<p2.y;
}
long long dis(const point &p1,const point &p2)
{
    return (long long)(p1.x-p2.x)*(p1.x-p2.x)+(long long)(p1.y-p2.y)*(p1.y-p2.y)
        ;
}
void build(int l, int r, point p[])
{
    if (l > r) return;
    int mid = l + r >> 1;
    int minX, minY, maxX, maxY;
    minX = min_element(p + l, p + r + 1, cmpX)->x;
    minY = min_element(p + l, p + r + 1, cmpY)->y;
    maxX = max_element(p + l, p + r + 1, cmpX)->x;
    maxY = max_element(p + l, p + r + 1, cmpY)->y;
    Div[mid] = (maxX - minX >= maxY - minY);
    nth_element(p + l, p + mid, p + r + 1, Div[mid] ? cmpX : cmpY);
    build(l, mid - 1, p);
    build(mid + 1, r, p);
}
void find(long long &res, int l, int r, point &a, point p[])
{
    if (l > r)  return;
    int mid = l + r >> 1;
    long long dist = dis(a, p[mid]);
    if (dist > 0) ///NOTICE
        res = min(res, dist);
    long long d = Div[mid] ? (a.x - p[mid].x) : (a.y - p[mid].y);
    int l1, l2, r1, r2;
    l1 = l, l2 = mid + 1;
    r1 = mid - 1, r2 = r;
    if (d > 0) swap(l1, l2), swap(r1, r2);
    find(res, l1, r1, a, p);
    if (d * d < res) find(res, l2, r2, a, p);
}
long long find(int l,int r, point &a, point p[])
{
    long long res=INF;
    find(res,l,r,a,p);
    return res;
}
point pp[NV];
int main()
{
    int t;
    cin>>t;
    while(t--)
    {
        int n;
        cin>>n;
```

```
67          for (int i=1; i<=n; i++) p[i].in(),pp[i]=p[i];
68          build(1,n,p);
69          for (int i=1; i<=n; i++)
70              printf("%I64d\n",find(1,n,pp[i],p));
71      }
72      return 0;
73  }
```

4.11.2 K 维 KD-tree

```
1   #define sqr(x) (x)*(x)
2   const int NV=50005,K=5;
3   //k为维数,n为点数,idx为当前比较的维度
4   int k,n,idx;
5   struct point
6   {
7       int x[K];
8       bool operator <(const point &u) const
9       {
10          return x[idx]<u.x[idx];
11      }
12  } po[NV]; //输入的点
13  priority_queue<pair<double,point> >nq;
14  //线段树节点
15  point pt[NV<<2];
16  int son[NV<<2];
17  void build(int l,int r,int rt=1,int dep=0)
18  {
19      if(l>r) return;
20      son[rt]=r-l;
21      son[rt<<1]=son[rt<<1|1]=-1;
22      idx=dep%k;
23      int mid=l+r>>1;
24      nth_element(po+l,po+mid,po+r+1);
25      pt[rt]=po[mid];
26      build(l,mid-1,rt<<1,dep+1);
27      build(mid+1,r,rt<<1|1,dep+1);
28  }
29  void query(point &p,int m,int rt=1,int dep=0)
30  {
31      if(son[rt]==-1) return;
32      pair<double,point> nd(0,pt[rt]);
33      for(int i=0; i<k; i++) nd.first+=sqr(nd.second.x[i]-p.x[i]);
34      int dim=dep%k,x=rt<<1,y=rt<<1|1,fg=0;
35      if(p.x[dim]>=pt[rt].x[dim]) swap(x,y);
36      if(~son[x]) query(p,m,x,dep+1);
37      if(nq.size()<m) nq.push(nd),fg=1;
38      else
39      {
40          if(nd.first<nq.top().first) nq.pop(),nq.push(nd);
41          if(sqr(p.x[dim]-pt[rt].x[dim])<nq.top().first) fg=1;
42      }
43      if(~son[y]&&fg) query(p,m,y,dep+1);
44  }
45  int main()
46  {
47      while(scanf("%d%d",&n,&k)!=EOF)
48      {
49          for(int i=1; i<=n; i++)
50              for(int j=0; j<k; j++)
51                  scanf("%d",&po[i].x[j]);
52          build(1,n);
```

```
        int t;
        scanf("%d",&t);
        while(t--)
        {
            point ask;
            for(int j=0; j<k; j++) scanf("%d",&ask.x[j]);
            int m;
            scanf("%d",&m);
            query(ask,m);
            printf("the closest %d points are:\n", m);
            point ans[20];
            int cnt=0;
            while(!nq.empty())
            {
                ans[++cnt]=nq.top().second;
                nq.pop();
            }
            for(int i=cnt; i>=1; i--)
                for(int j=0; j<k; j++)
                    printf("%d%c",ans[i].x[j]," \n"[j==k-1]);
        }
    }
    return 0;
}
```

4.12 树链剖分

'I' 区间增加
'D' 区间减少
'Q' 单点查询
点权，change(update) 区间更新，query 单点查询

```
const int NV=50005;
//num为初始数组，rank只在为初始数组build时使用
int num[NV],siz[NV],top[NV],son[NV];
int dep[NV],w[NV],rank[NV],fa[NV];
int he[NV],to[2*NV],next[2*NV],ecnt,tot;
void adde(int u,int v)
{
    to[ecnt]=v,next[ecnt]=he[u],he[u]=ecnt++;
    to[ecnt]=u,next[ecnt]=he[v],he[v]=ecnt++;
}
void dfs1(int u,int father,int d)
{
    dep[u]=d;
    fa[u]=father;
    siz[u]=1;
    for(int i=he[u]; ~i; i=next[i])
    {
        int v=to[i];
        if(v!=father)
        {
            dfs1(v,u,d+1);
            siz[u]+=siz[v];
            if(son[u]==-1||siz[v]>siz[son[u]])
                son[u]=v;
        }
    }
}
void dfs2(int u,int tp)
{
```

```
30      top[u]=tp;
31      w[u]=++tot;
32      rank[w[u]]=u;
33      if(son[u]==-1) return;
34      dfs2(son[u],tp);
35      for(int i=he[u]; ~i; i=next[i])
36      {
37          int v=to[i];
38          if(v!=son[u]&&v!=fa[u])
39              dfs2(v,v);
40      }
41  }
42  #define lson l,m,rt<<1
43  #define rson m+1,r,rt<<1|1
44  int add[NV<<2],sum[NV<<2];
45  void PushUp(int rt)
46  {
47      sum[rt]=sum[rt<<1]+sum[rt<<1|1];
48  }
49  void PushDown(int rt,int m)
50  {
51      if (add[rt])
52      {
53          add[rt<<1] += add[rt];
54          add[rt<<1|1] += add[rt];
55          sum[rt<<1] += add[rt] * (m - (m >> 1));
56          sum[rt<<1|1] += add[rt] * (m >> 1);
57          add[rt] = 0;
58      }
59  }
60  void build(int l,int r,int rt=1)
61  {
62      add[rt] = 0;
63      if (l == r)
64      {
65          sum[rt]=num[rank[l]];
66          return ;
67      }
68      int m = (l + r) >> 1;
69      build(lson);
70      build(rson);
71      PushUp(rt);
72  }
73  void update(int L,int R,int c,int l,int r,int rt=1)
74  {
75      if (L <= l && r <= R)
76      {
77          add[rt] += c;
78          sum[rt] += c * (r - l + 1);
79          return ;
80      }
81      PushDown(rt , r - l + 1);
82      int m = (l + r) >> 1;
83      if (L <= m) update(L , R , c , lson);
84      if (m < R) update(L , R , c , rson);
85      PushUp(rt);
86  }
87  int query(int L,int l,int r,int rt=1)
88  {
89      if (L == l && l == r)
90      {
91          return sum[rt];
92      }
```

```
93      PushDown(rt , r - l + 1);
94      int m = (l + r) >> 1;
95      if (L <= m) return query(L , lson);
96      return query(L , rson);
97  }
98  void change(int x,int y,int l,int r,int c)
99  {
100     while(top[x]!=top[y])
101     {
102         if(dep[top[x]]<dep[top[y]]) swap(x,y);
103         update(w[top[x]],w[x],c,l,r);
104         x=fa[top[x]];
105     }
106     if(dep[x]>dep[y]) swap(x,y);
107     update(w[x],w[y],c,l,r);
108 }
109 void init(int n)
110 {
111     memset(he,-1,sizeof(he));
112     memset(son,-1,sizeof(son));
113     tot=0;
114     ecnt=0;
115     for(int i=1; i<=n; i++)
116         scanf("%d",&num[i]);
117     for(int i=1; i<n; i++)
118     {
119         int u,v;
120         scanf("%d%d",&u,&v);
121         adde(u,v);
122     }
123     dfs1(1,0,0);
124     dfs2(1,1);
125     build(1,n);
126 }
127 int main()
128 {
129     int n,q;
130     while(~scanf("%d%*d%d",&n,&q))
131     {
132         init(n);
133         while(q--)
134         {
135             int a,b,c;
136             char op[5];
137             scanf("%s",op);
138             if(op[0]=='Q')
139             {
140                 scanf("%d",&a);
141                 printf("%d\n",query(w[a],1,n));
142             }
143             else
144             {
145                 scanf("%d%d%d",&a,&b,&c);
146                 if(op[0]=='D') c=-c;
147                 change(a,b,1,n,c);
148             }
149         }
150     }
151     return 0;
152 }
```

Message A: 0 u
A kid in hut u calls Wind. She should go to hut u from her current position.

Message B: 1 i w
The time required for i-th road is changed to w.
边权，update 单点更新，getsum(query) 区间求和

```
const int NV=100005;
//num为初始数组，rank只在为初始数组build时使用
int num[NV],siz[NV],top[NV],son[NV];
int dep[NV],w[NV],rank[NV],fa[NV];
int he[NV],to[2*NV],next[2*NV],ecnt,tot;
void adde(int u,int v)
{
    to[ecnt]=v,next[ecnt]=he[u],he[u]=ecnt++;
    to[ecnt]=u,next[ecnt]=he[v],he[v]=ecnt++;
}
void dfs1(int u,int father,int d)
{
    dep[u]=d;
    fa[u]=father;
    siz[u]=1;
    for(int i=he[u]; ~i; i=next[i])
    {
        int v=to[i];
        if(v!=father)
        {
            dfs1(v,u,d+1);
            siz[u]+=siz[v];
            if(son[u]==-1||siz[v]>siz[son[u]])
                son[u]=v;
        }
    }
}
void dfs2(int u,int tp)
{
    top[u]=tp;
    w[u]=++tot;
    rank[w[u]]=u;
    if(son[u]==-1) return;
    dfs2(son[u],tp);
    for(int i=he[u]; ~i; i=next[i])
    {
        int v=to[i];
        if(v!=son[u]&&v!=fa[u])
            dfs2(v,v);
    }
}
#define lson l,m,rt<<1
#define rson m+1,r,rt<<1|1
int sum[NV<<2];
void PushUp(int rt)
{
    sum[rt]=sum[rt<<1]+sum[rt<<1|1];
}
void build(int l,int r,int rt=1)
{
    if (l == r)
    {
        sum[rt]=num[l];
        return ;
    }
    int m = (l + r) >> 1;
    build(lson);
    build(rson);
```

```
59      PushUp(rt);
60  }
61  void update(int L,int c,int l,int r,int rt=1)
62  {
63      if (L == l && l == r)
64      {
65          sum[rt] = c;
66          return ;
67      }
68      int m = (l + r) >> 1;
69      if (L <= m) update(L , c , lson);
70      else update(L , c , rson);
71      PushUp(rt);
72  }
73  int query(int L,int R,int l,int r,int rt=1)
74  {
75      if (L <= l && r <= R)
76      {
77          return sum[rt];
78      }
79      int m = (l + r) >> 1;
80      int ret = 0;
81      if (L <= m) ret += query(L , R , lson);
82      if (m < R) ret += query(L , R , rson);
83      return ret;
84  }
85  int getsum(int x,int y,int l,int r)
86  {
87      int ret=0;
88      while(top[x]!=top[y])
89      {
90          if(dep[top[x]]<dep[top[y]]) swap(x,y);
91          ret+=query(w[top[x]],w[x],l,r);
92          x=fa[top[x]];
93      }
94      if (x==y) return ret; ///Notice! 防止产生son[x]为-1的情况
95      if(dep[x]>dep[y]) swap(x,y);
96      ret+=query(w[son[x]],w[y],l,r); ///Notice! 比点权多了个son
97      return ret;
98  }
99  int uu[NV],vv[NV],ll[NV];
100 void init(int n)
101 {
102     memset(he,-1,sizeof(he));
103     memset(son,-1,sizeof(son));
104     tot=0;
105     ecnt=0;
106     for(int i=1; i<n; i++)
107     {
108         scanf("%d%d%d",&uu[i],&vv[i],&ll[i]);
109         adde(uu[i],vv[i]);
110     }
111     dfs1(1,0,0);
112     dfs2(1,1);
113     num[1]=0;
114     for (int i=1; i<n; i++)
115     {
116         if (dep[uu[i]]<dep[vv[i]]) num[w[vv[i]]]=ll[i];
117         else num[w[uu[i]]]=ll[i];
118     }
119     build(1,n);
120 }
121 int main()
```

```
{
    int n,q,s;
    while(~scanf("%d%d%d",&n,&q,&s))
    {
        init(n);
        while(q--)
        {
            int a,b;
            int op;
            scanf("%d",&op);
            if(op==0)
            {
                scanf("%d",&a);
                printf("%d\n",getsum(a,s,1,n));
                s=a;
            }
            else
            {
                scanf("%d%d",&a,&b);
                if (dep[uu[a]]<dep[vv[a]]) a=w[vv[a]];
                else a=w[uu[a]];
                update(a,b,1,n);
            }
        }
    }
    return 0;
}
```

CHANGE iv Change the weight of the ith edge to v
NEGATE ab Negate the weight of every edge on the path from a to b
QUERY ab Find the maximum weight of edges on the path from a to b
边权，change **单点更新，**Negate(update) **区间更新，**getmax(query) **区间求最值**

```
const int NV=10005;
//num为初始数组，rank只在为初始数组build时使用
int num[NV],siz[NV],top[NV],son[NV];
int dep[NV],w[NV],rank[NV],fa[NV];
int he[NV],to[2*NV],next[2*NV],ecnt,tot;
void adde(int u,int v)
{
    to[ecnt]=v,next[ecnt]=he[u],he[u]=ecnt++;
    to[ecnt]=u,next[ecnt]=he[v],he[v]=ecnt++;
}
void dfs1(int u,int father,int d)
{
    dep[u]=d;
    fa[u]=father;
    siz[u]=1;
    for(int i=he[u]; ~i; i=next[i])
    {
        int v=to[i];
        if(v!=father)
        {
            dfs1(v,u,d+1);
            siz[u]+=siz[v];
            if(son[u]==-1||siz[v]>siz[son[u]])
                son[u]=v;
        }
    }
}
void dfs2(int u,int tp)
{
```

```
30      top[u]=tp;
31      w[u]=++tot;
32      rank[w[u]]=u;
33      if(son[u]==-1) return;
34      dfs2(son[u],tp);
35      for(int i=he[u]; ~i; i=next[i])
36      {
37          int v=to[i];
38          if(v!=son[u]&&v!=fa[u])
39              dfs2(v,v);
40      }
41  }
42  #define lson l,m,rt<<1
43  #define rson m+1,r,rt<<1|1
44  int sgn[NV<<2],mx[NV<<2],mn[NV<<2];
45  void PushUp(int rt)
46  {
47      mx[rt]=max(mx[rt<<1],mx[rt<<1|1]);
48      mn[rt]=min(mn[rt<<1],mn[rt<<1|1]);
49  }
50  void PushDown(int rt,int m)
51  {
52      if (sgn[rt]==-1)
53      {
54          sgn[rt<<1] *= -1;
55          swap(mx[rt<<1],mn[rt<<1]);
56          mx[rt<<1] = -mx[rt<<1];
57          mn[rt<<1] = -mn[rt<<1];
58          sgn[rt<<1|1] *= -1;
59          swap(mx[rt<<1|1],mn[rt<<1|1]);
60          mx[rt<<1|1] = -mx[rt<<1|1];
61          mn[rt<<1|1] = -mn[rt<<1|1];
62          sgn[rt] = 1;
63      }
64  }
65  void build(int l,int r,int rt=1)
66  {
67      sgn[rt] = 1;
68      if (l == r)
69      {
70          mx[rt]=num[l];
71          mn[rt]=num[l];
72          return ;
73      }
74      int m = (l + r) >> 1;
75      build(lson);
76      build(rson);
77      PushUp(rt);
78  }
79  void update(int L,int R,int c,int l,int r,int rt=1)
80  {
81      if (L <= l && r <= R)
82      {
83          sgn[rt] *= c;
84          swap(mx[rt],mn[rt]);
85          mx[rt] = -mx[rt];
86          mn[rt] = -mn[rt];
87          return ;
88      }
89      PushDown(rt , r - l + 1);
90      int m = (l + r) >> 1;
91      if (L <= m) update(L , R , c , lson);
92      if (m < R) update(L , R , c , rson);
```

```
93      PushUp(rt);
94  }
95  void change(int L,int c,int l,int r,int rt=1)
96  {
97      if (L == l && l == r)
98      {
99          mx[rt] = c;
100         mn[rt] = c;
101         return ;
102     }
103     PushDown(rt , r - l + 1);
104     int m = (l + r) >> 1;
105     if (L <= m) change(L , c , lson);
106     else change(L , c , rson);
107     PushUp(rt);
108 }
109 int query(int L,int R,int l,int r,int rt=1)
110 {
111     if (L <= l && r <= R)
112     {
113         return mx[rt];
114     }
115     PushDown(rt , r - l + 1);
116     int m = (l + r) >> 1;
117     int ret = -inf;
118     if (L <= m) ret = max(ret,query(L , R , lson));
119     if (m < R) ret = max(ret,query(L , R , rson));
120     return ret;
121 }
122 void Negate(int x,int y,int l,int r)
123 {
124     while(top[x]!=top[y])
125     {
126         if(dep[top[x]]<dep[top[y]]) swap(x,y);
127         update(w[top[x]],w[x],-1,l,r);
128         x=fa[top[x]];
129     }
130     if (x==y) return; ///Notice! 防止产生son[x]为-1的情况
131     if(dep[x]>dep[y]) swap(x,y);
132     update(w[son[x]],w[y],-1,l,r); ///Notice! 比点权多了个son
133 }
134 int getmax(int x,int y,int l,int r)
135 {
136     int ret=-inf;
137     while(top[x]!=top[y])
138     {
139         if(dep[top[x]]<dep[top[y]]) swap(x,y);
140         ret=max(ret,query(w[top[x]],w[x],l,r));
141         x=fa[top[x]];
142     }
143     if (x==y) return ret; ///Notice! 防止产生son[x]为-1的情况
144     if(dep[x]>dep[y]) swap(x,y);
145     ret=max(ret,query(w[son[x]],w[y],l,r)); ///Notice! 比点权多了个son
146     return ret;
147 }
148 int uu[NV],vv[NV],ll[NV];
149 void init(int n)
150 {
151     memset(he,-1,sizeof(he));
152     memset(son,-1,sizeof(son));
153     tot=0;
154     ecnt=0;
155     for(int i=1; i<n; i++)
```

```
156     {
157         scanf("%d%d%d",&uu[i],&vv[i],&ll[i]);
158         adde(uu[i],vv[i]);
159     }
160     dfs1(1,0,0);
161     dfs2(1,1);
162     num[1]=0;
163     for (int i=1; i<n; i++)
164     {
165         if (dep[uu[i]]<dep[vv[i]]) num[w[vv[i]]]=ll[i];
166         else num[w[uu[i]]]=ll[i];
167     }
168     build(1,n);
169 }
170 int main()
171 {
172     int t;
173     cin>>t;
174     while(t--)
175     {
176         int n;
177         scanf("%d",&n);
178         init(n);
179         char op[10];
180         while(scanf("%s",op),op[0]!='D')
181         {
182             int a,b;
183             scanf("%d%d",&a,&b);
184             if (op[0]=='C')
185             {
186                 if (dep[uu[a]]<dep[vv[a]]) a=w[vv[a]];
187                 else a=w[uu[a]];
188                 change(a,b,1,n);
189             }
190             else if (op[0]=='N') Negate(a,b,1,n);
191             else printf("%d\n",getmax(a,b,1,n));
192         }
193     }
194     return 0;
195 }
```

5 字符串

5.1 KMP

```
1  const int LEN=1000005;
2  int next[LEN];
3  char s[LEN];
4  char sub[LEN];
5  void getnext(char sub[],int next[],int len2)
6  {
7      int i=0,j=-1;
8      next[0]=-1;
9      while(i<len2)
10     {
11         if (j==-1||sub[i]==sub[j])
12             i++,j++,next[i]=j;
13         else j=next[j];
14     }
15 }
16 int kmp(char s[],char sub[],int len1,int len2,int next[])
```

```
{
    int i=0,j=0;
    while(i<len1&&j<len2)
    {
        if (j==-1||s[i]==sub[j])
            i++,j++;
        else j=next[j];
    }
    if (j>=len2) return i-len2;
    else return -1;
}
```

5.2 扩展 KMP

```
const int LEN = 100005;
char s[LEN], sub[LEN];
int next[LEN], ret[LEN];
void exkmp(char *a, char *b, int M, int N, int *next, int *ret)
{
    int i, j, k;
    //得到A关于A的后缀的最长公共前缀，存在next数组中
    for (j = 0; 1 + j < M && a[j] == a[1 + j]; j++) ;
    next[1] = j;
    k = 1;
    for (i = 2; i < M; i++)
    {
        int len = k + next[k], L = next[i - k];
        if (L < len - i) next[i] = L;
        else
        {
            for (j = max(0, len - i); i + j < M && a[j] == a[i + j]; j++) ;
            next[i] = j;
            k = i;
        }
    }
    //得到A关于B的后缀的最长公共前缀，存在ret数组中
    for (j = 0; j < N && j < M && a[j] == b[j]; j++) ;
    ret[0] = j;
    k = 0;
    for (i = 1; i < N; i++)
    {
        int len = k + ret[k], L = next[i - k];
        if (L < len - i) ret[i] = L;
        else
        {
            for (j = max(0, len - i); j < M && i + j < N && a[j] == b[i + j]; j
                ++) ;
            ret[i] = j;
            k = i;
        }
    }
}
```

5.3 trie 字典树

```
const int NS=1005*25;
int c[NS][128];
struct trie
{
//    int val[NS];
```

```
    int cnt[NS];
    int sz;
    trie()
    {
        sz=1;
        memset(c[0],0,sizeof(c[0]));
        memset(cnt,0,sizeof(cnt));
    }
    void insert(char s[],int v=0)
    {
        int u=0;
        for (int i=0; s[i]; i++)
        {
            if (!c[u][s[i]])
            {
                memset(c[sz],0,sizeof(c[sz]));
//                val[sz]=0;
                c[u][s[i]]=sz++;
            }
            u=c[u][s[i]];
            cnt[u]++;
        }
//        val[u]=v;
    }
    int query(char s[])
    {
        int u=0,n=strlen(s);
        for (int i=0; i<n; i++)
        {
            if (!c[u][s[i]]||u!=0&&cnt[u]<=1)
                return i;
            u=c[u][s[i]];
        }
        return n;
//        return cnt[u];
    }
};
```

5.4 AC 自动机

```
struct trie
{
    int next[500010][26],fail[500010],ed[500010];
    int root,L;
    int newnode()
    {
        for(int i = 0; i < 26; i++)
            next[L][i] = -1;
        ed[L++] = 0;
        return L-1;
    }
    void init()
    {
        L = 0;
        root = newnode();
    }
    void insert(char buf[])
    {
        int len = strlen(buf);
        int now = root;
        for(int i = 0; i < len; i++)
        {
```

```
23              if(next[now][buf[i]-'a'] == -1)
24                  next[now][buf[i]-'a'] = newnode();
25              now = next[now][buf[i]-'a'];
26          }
27          ed[now]++;
28      }
29      void build()
30      {
31          queue<int> q;
32          fail[root] = root;
33          for(int i = 0; i < 26; i++)
34              if(next[root][i] == -1)
35                  next[root][i] = root;
36              else
37              {
38                  fail[next[root][i]] = root;
39                  q.push(next[root][i]);
40              }
41          while(!q.empty())
42          {
43              int now = q.front();
44              q.pop();
45              for(int i = 0; i < 26; i++)
46                  if(next[now][i] == -1)
47                      next[now][i] = next[fail[now]][i];
48                  else
49                  {
50                      fail[next[now][i]]=next[fail[now]][i];
51                      q.push(next[now][i]);
52                  }
53          }
54      }
55      int query(char buf[])
56      {
57          int len = strlen(buf);
58          int now = root;
59          int res = 0;
60          for(int i = 0; i < len; i++)
61          {
62              now = next[now][buf[i]-'a'];
63              int temp = now;
64              while(temp != root)
65              {
66                  res += ed[temp];
67                  ed[temp] = 0;
68                  temp = fail[temp];
69              }
70          }
71          return res;
72      }
73      void Debug()
74      {
75          for(int i = 0; i < L; i++)
76          {
77              printf("id = %3d,fail = %3d,end = %3d,chi = [",i,fail[i],ed[i]);
78              for(int j = 0; j < 26; j++)
79                  printf("%2d",next[i][j]);
80              printf("]\n");
81          }
82      }
83  };
84  char buf[1000010];
85  trie ac;
```

```
int main()
{
    int t;
    int n;
    scanf("%d",&t);
    while(t--)
    {
        scanf("%d",&n);
        ac.init();
        for(int i = 0; i < n; i++)
        {
            scanf("%s",buf);
            ac.insert(buf);
        }
        ac.build();
        scanf("%s",buf);
        printf("%d\n",ac.query(buf));
    }
    return 0;
}
```

5.5 哈希

5.5.1 字符串哈希

```
const int LEN=100005;
unsigned long long f[LEN],r[LEN],pw[LEN];
int len;
void init(char *s)
{
    pw[0]=1;
    len=1;
    f[0]=s[0];
    for (int i=1; s[i]; i++)
        f[i] = f[i-1] * 131 + s[i],len++,pw[i]=pw[i-1]*131;
    r[len-1]=s[len-1];
    for (int i=len-2; i>=0; i--)
        r[i] = r[i+1] * 131 + s[i];
}
unsigned long long fwd(int i,int j)
{
    if (i<0||j>=len) return 0;
    if (i==0) return f[j];
    return f[j]-f[i-1]*pw[j-i+1];
}
unsigned long long rev(int i,int j)
{
    if (i<0||j>=len) return 0;
    if (j==len-1) return r[i];
    return r[i]-r[j+1]*pw[j-i+1];
}
```

5.5.2 字符串矩阵哈希

```
const int NN=1005;
const int NM=1005;
unsigned long long hash[NN][NM],pw1[NM],pw2[NN],seed1=131,seed2=1313;
void init(char s[][NM],int n,int m)
{
    int mx=max(n,m);
```

```
    pw1[0]=pw2[0]=1;
    for (int i=1; i<=m; i++)
        pw1[i]=pw1[i-1]*seed1;
    for (int i=1; i<=n; i++)
        pw2[i]=pw2[i-1]*seed2;
    hash[0][1]=hash[1][0]=hash[0][0]=0;
    for (int i=1; i<=n; i++)
    {
        for (int j=1; j<=m; j++)
            hash[i][j] = hash[i][j-1] * seed1 + s[i][j];
        for (int j=1; j<=m; j++)
            hash[i][j] = hash[i-1][j] * seed2 + hash[i][j];
    }
}
unsigned long long query(int x,int y,int xx,int yy,int n,int m)
{
    if (x<1||x>n||xx<1||x>n||y<1||y>m||yy<1||yy>m||x>xx||y>yy)
        return 0;
    unsigned long long t1,t2;
    t1=hash[xx][y-1]-hash[x-1][y-1]*pw2[xx-x+1];
    t2=hash[xx][yy]-hash[x-1][yy]*pw2[xx-x+1];
    return t2-t1*pw1[yy-y+1];
}
```

5.5.3 哈希函数

```
// BKDR Hash Function
unsigned long long BKDRHash(char *str)
{
    unsigned long long seed = 131; // 31 131 1313 13131 131313 etc..
    unsigned long long hash = 0;
    while (*str)
        hash = hash * seed + (*str++);
    return hash;
}

// BKDR Hash Function
unsigned int BKDRHash(char *str)
{
    unsigned int seed = 131; // 31 131 1313 13131 131313 etc..
    unsigned int hash = 0;
    while (*str)
        hash = hash * seed + (*str++);
    return (hash & 0x7FFFFFFF);
}

// AP Hash Function
unsigned int APHash(char *str)
{
    unsigned int hash = 0;
    for (int i=0; *str; i++)
    {
        if ((i & 1) == 0)
        {
            hash ^= ((hash << 7) ^ (*str++) ^ (hash >> 3));
        }
        else
        {
            hash ^= (~((hash << 11) ^ (*str++) ^ (hash >> 5)));
        }
    }
    return (hash & 0x7FFFFFFF);
```

```
}

// DJB Hash Function
unsigned int DJBHash(char *str)
{
    unsigned int hash = 5381;
    while (*str)
        hash += (hash << 5) + (*str++);
    return (hash & 0x7FFFFFFF);
}

// JS Hash Function
unsigned int JSHash(char *str)
{
    unsigned int hash = 1315423911;
    while (*str)
    {
        hash ^= ((hash << 5) + (*str++) + (hash >> 2));
    }
    return (hash & 0x7FFFFFFF);
}

// RS Hash Function
unsigned int RSHash(char *str)
{
    unsigned int b = 378551;
    unsigned int a = 63689;
    unsigned int hash = 0;
    while (*str)
    {
        hash = hash * a + (*str++);
        a *= b;
    }
    return (hash & 0x7FFFFFFF);
}

unsigned int SDBMHash(char *str)
{
    unsigned int hash = 0;
    while (*str)
    {
        // equivalent to: hash = 65599*hash + (*str++);
        hash = (*str++) + (hash << 6) + (hash << 16) - hash;
    }
    return (hash & 0x7FFFFFFF);
}

// P. J. Weinberger Hash Function
unsigned int PJWHash(char *str)
{
    unsigned int BitsInUnignedInt = (unsigned int)(sizeof(unsigned int) * 8);
    unsigned int ThreeQuarters    = (unsigned int)((BitsInUnignedInt * 3) / 4);
    unsigned int OneEighth        = (unsigned int)(BitsInUnignedInt / 8);
    unsigned int HighBits         = (unsigned int)(0xFFFFFFFF) << (
       BitsInUnignedInt - OneEighth);
    unsigned int hash             = 0;
    unsigned int test             = 0;
    while (*str)
    {
        hash = (hash << OneEighth) + (*str++);
        if ((test = hash & HighBits) != 0)
        {
            hash = ((hash ^ (test >> ThreeQuarters)) & (~HighBits));
```

```
        }
    }
    return (hash & 0x7FFFFFFF);
}

// ELF Hash Function
unsigned int ELFHash(char *str)
{
    unsigned int hash = 0;
    unsigned int x    = 0;
    while (*str)
    {
        hash = (hash << 4) + (*str++);
        if ((x = hash & 0xF0000000L) != 0)
        {
            hash ^= (x >> 24);
            hash &= ~x;
        }
    }
    return (hash & 0x7FFFFFFF);
}
```

5.6 字符串的最小表示法

```
int getmin(char s[],int len)
{
    int i=0,j=1,k=0;
    while(i<len&&j<len&&k<len)
    {
        int t=s[(i+k)%len]-s[(j+k)%len];
        if (!t) k++;
        else
        {
            if (t>0) i+=k+1;
            else j+=k+1;
            if (i==j) j++;
            k=0;
        }
    }
    return min(i,j);
}
```

6 动态规划

6.1 最大子段和

```
int a[100005];
int dp[100005]= {};
for (int i=1; i<=n; i++)
    dp[i]=max(a[i],dp[i-1]+a[i]);
for (int i=2; i<=n; i++)
    dp[i]=max(dp[i-1],dp[i]);
///输出子段起始点
while(scanf("%d",&n),n)
{
    for (int i=1; i<=n; i++)
        scanf("%d",&a[i]);
    memset(dp,-1,sizeof(dp));
    for (int i=1; i<=n; i++)
```

```
    {
        if (dp[i-1]<0)
        {
            b[i]=i;
            dp[i]=a[i];
        }
        else
        {
            b[i]=b[i-1];
            dp[i]=dp[i-1]+a[i];
        }
    }
    int index,mx=-1;
    for (int i=1; i<=n; i++)
        if (dp[i]>mx)
        {
            mx=dp[i];
            index=i;
        }
    if (mx<0) printf("%d %d %d\n",0,a[1],a[n]);
    else printf("%d %d %d\n",mx,a[b[index]],a[index]);
}
```

6.2 二维最大子段和

```
int main()
{
    int t[500][500];
    int n;
    while(cin>>n)
    {
        for (int i=1; i<=n; i++)
            for (int j=1; j<=n; j++)
                scanf("%d",&t[i][j]);
        int a[500][500]= {};
        int mx=0;
        for (int i=1; i<=n; i++)
        {
            for (int j=1; j<=n; j++)
                a[i][j]+=t[i][j]+a[i-1][j];
            for (int j=1; j<=i; j++)
            {
                int sum=0;
                for (int k=1; k<=n; k++)
                {
                    if (sum<0)
                        sum=a[i][k]-a[j-1][k];
                    else
                        sum+=a[i][k]-a[j-1][k];
                    mx=max(mx,sum);
                }
            }
        }
        printf("%d\n",mx);
    }
    return 0;
}
```

6.3 最长上升子序列 (LIS)

```
///O(n^2)
int dp[5000]= {};
dp[1]=1;
for (int i=2; i<=n; i++)
{
    int mx=0;
    for (int j=1; j<i; j++)
        if (a[j]<a[i])
            mx=max(mx,dp[j]);
    dp[i]=mx+1;
}
int mx=0;
for (int i=1; i<=n; i++)
    mx=max(mx,dp[i]);

///O(nlogn)
int dp[5000]= {};
dp[1]=a[1];
int ans=1;
for (int i=2; i<=n; i++)
{
    if (a[i]>=dp[ans]) dp[++ans]=a[i];
    else *lower_bound(dp+1,dp+ans+1,a[i])=a[i];
}
```

6.4 最长公共子序列 (LCS)

```
const int LEN=1005;
int dp[LEN][LEN],res;
int a[LEN],b[LEN];
int n,m;
int lcs()
{
    memset(dp,0,sizeof(dp));
    for (int i=1; i<=n; i++)
        for (int j=1; j<=m; j++)
            if (a[i]==b[j]) dp[i][j]=dp[i-1][j-1]+1;
            else dp[i][j]=max(dp[i-1][j],dp[i][j-1]);
    return res=dp[n][m];
}
int ans[LEN];
void getans()
{
    int num=res;
    for (int i=n,j=m; i>=1&&j>=1;)
        if (a[i]==b[j])
        {
            ans[num--]=a[i];
            i--;
            j--;
        }
        else
        {
            if (dp[i-1][j]>dp[i][j-1]) i--;
            else j--;
        }
}
```

6.5 最长公共上升子序列 (LCIS)

```
const int LEN=1005;
int dp[LEN][LEN],path[LEN][LEN],res;
int a[LEN],b[LEN];
int n,m,ai,aj;
int lcis()
{
    memset(dp,0,sizeof(dp));
    res=0;
    for (int i=1; i<=n; i++)
    {
        int mx=0,t;
        for (int j=1; j<=m; j++)
        {
            dp[i][j]=dp[i-1][j];
            path[i][j]=-1;
            if (a[i]>b[j]&&mx<dp[i-1][j]) mx=dp[i-1][j],t=j;
            if (a[i]==b[j]) dp[i][j]=mx+1,path[i][j]=t;
            if(res<dp[i][j])
            {
                res=dp[i][j];
                ai=i;
                aj=j;
            }
        }
    }
    return res;
}
int ans[LEN];
void getans()
{
    int num=res;
    while(num)
    {
        if (path[ai][aj]!=-1)
        {
            ans[num--]=b[aj];
            aj=path[ai][aj];
        }
        ai--;
    }
}
```

6.6 数位 DP

```
int dfs(int pos, int s, bool e) //e==1表示满了（当前pos之前的数位都相同）
{
    if (pos==-1) return s==target_s; //返回 当前状态==目标状态
    if (~f[pos][s] && !e ) return f[pos][s]; //
        如果值已经算过并且没满则返回记录的值
    int ans = 0;
    int u = e ? bit[pos] : 9; //9可以相应改成进制
    for (int i = 0; i <= u; ++i)
        ans += dfs(pos-1, new_s(s, i), e&&i==u); //new_s表示新的状态。另外，“
            肯定不符合目标状态”的状态可以直接continue掉
    return e ? ans : f[pos][s]=ans; //如果满了就直接返回，否则记录再返回
}
int solve(int n)
{
    memset(f,-1,sizeof(f));
    int pos=-1;
    while(n)
    {
```

```
17          bit[++pos]=n%10; //10为进制
18          n/=10;
19      }
20      return dfs(pos,0,1); //初始状态
21  }
22
23  int f[20][5];
24  int bit[20];
25  int dfs(int pos, int s, bool e)
26  {
27      if (pos==-1) return s==0||s==1;
28      if (~f[pos][s]&&!e) return f[pos][s];
29      int ans=0;
30      int u=e?bit[pos]:9;
31      for (int i=0; i<=u; ++i)
32      {
33          if(i==4||s==1&&i==2)
34              continue;
35          ans+=dfs(pos-1,i==6,e&&i==u);
36      }
37      return e?ans:f[pos][s]=ans;
38  }
39  int solve(int n)
40  {
41      memset(f,-1,sizeof(f));
42      int pos=-1;
43      while(n)
44      {
45          bit[++pos]=n%10;
46          n/=10;
47      }
48      return dfs(pos,0,1);
49  }
```

7 数论

7.1 筛法打质数表

```
1   const int NP=1000005;
2   int ispri[NP]= {},prime[NP],pcnt=0;
3   void getprime()
4   {
5       ispri[0]=ispri[1]=1;
6       for (long long i=2; i<NP; i++)
7           if (ispri[i]==0)
8           {
9               prime[++pcnt]=i;
10              for (long long j=i*i; j<NP; j+=i)
11                  ispri[j]=1;
12          }
13  }
```

7.2 区间筛质数

```
1   const int NP=100005;
2   int ispri[NP]= {},prime[NP],pcnt=0;
3   void getprime()
4   {
5       ispri[0]=ispri[1]=1;
```

```
    for (long long i=2; i<NP; i++)
        if (ispri[i]==0)
        {
            prime[++pcnt]=i;
            for (long long j=i*i; j<NP; j+=i)
                ispri[j]=1;
        }
}
const int NI=1000005;
int iispri[NI],iprime[NI],icnt;
void igetprime(int l,int r)
{
    memset(iispri,0,sizeof(iispri));
    if (l<2) l=2;
    for (int i=1; i<=pcnt&&(long long)prime[i]*prime[i]<=r; i++)
    {
        int s=l/prime[i]+(l%prime[i]>0);
        if (s==1)s=2;
        for (long long j=s; j*prime[i]<=r; j++)
            if(j*prime[i]>=l)
                iispri[j*prime[i]-l]=1;
    }
    icnt=0;
    for(int i=0; i<=r-l; i++)
        if(!iispri[i])
            iprime[++icnt]=i+l;
}
```

7.3 分解质因数

```
int a[1000000]= {},pcnt=0;
void pdec(int n)
{
    int x=sqrt(n);
    for (int i=1; prime[i]<=x; i++)
        if (n%prime[i]==0)
        {
            a[++pcnt]=prime[i];
            n/=prime[i];
            i--;
        }
    if (n!=1)
        a[++pcnt]=n;
}
```

7.4 因数个数打表

```
int y[500005]= {};
void getfactor(int n)
{
    int x=sqrt(n);
    for (int i=1; i<=x; i++)
    {
        for (int j=i+1; j*i<=n; j++)
            y[i*j]+=2;
        y[i*i]++;
    }
}
```

7.5 快速乘法

```
const long long M=1000000007;
long long quickmul(long long a, long long b)
{
    long long ret = 0;
    for (; b; b >>= 1, a = (a << 1) % M)
        if (b & 1)
            ret = (ret + a) % M;
    return ret;
}

///What the fuck is this??!! It works!!
long long quickmul(long long a,long long b)
{
    return (a*b-(long long)(a/(long double)M*b+1e-3)*M+M)%M;
}
```

7.6 快速幂

```
const long long M=1000000007;
long long quickpow(long long a, long long b)
{
    if(b < 0) return 0;
    long long ret = 1;
    a %= M;
    for (; b; b >>= 1, a = (a * a) % M)
        if (b & 1)
            ret = (ret * a) % M;
    return ret;
}
```

7.7 费马小定理求逆元

M 必须是质数

```
const long long M=1000000007;
long long quickpow(long long a, long long b)
{
    if(b < 0) return 0;
    long long ret = 1;
    a %= M;
    for (; b; b >>= 1, a = (a * a) % M)
        if (b & 1)
            ret = (ret * a) % M;
    return ret;
}
long long inv(long long a)
{
    return quickpow(a,M-2);
}
```

7.8 扩展欧几里得

```
long long exgcd(long long a,long long b,long long &x,long long &y)
{
    if (b==0)
    {
        x=1;
```

```
6         y=0;
7         return a;
8     }
9     long long ans=exgcd(b,a%b,x,y);
10    long long temp=x;
11    x=y;
12    y=temp-(a/b)*y;
13    return ans;
14 }
```

7.9 扩展欧几里得求逆元

需要与 M 互质

```
1  const long long M=1000000007;
2  long long exgcd(long long a,long long b,long long &x,long long &y)
3  {
4      if (b==0)
5      {
6          x=1;
7          y=0;
8          return a;
9      }
10     long long ans=exgcd(b,a%b,x,y);
11     long long temp=x;
12     x=y;
13     y=temp-(a/b)*y;
14     return ans;
15 }
16 long long inv(long long a)
17 {
18     long long x,y;
19     long long t=exgcd(a,M,x,y);
20     if(t!=1)
21         return -1;
22     return (x%M+M)%M;
23 }
```

7.10 欧拉函数

```
1  long long phi(long long n)
2  {
3      long long ans=n;
4      long long x=sqrt(n);
5      for (long long i=2; i<=x; i++)
6      {
7          if (n%i==0)
8          {
9              while(n%i==0)
10                 n/=i;
11             ans=ans/i*(i-1);
12         }
13     }
14     if (n>1)
15         ans=ans/n*(n-1);
16     return ans;
17 }
```

7.11 欧拉函数打表

```
const int MAXN=10005;
long long phi[MAXN];
void getphi()
{
    for (int i=1; i<MAXN; i++)
        phi[i]=i;
    for (int i=2; i<MAXN; i++)
        if (phi[i]==i)
            for (int j=i; j<MAXN; j+=i)
                phi[j]=phi[j]/i*(i-1);
}
```

7.12 中国剩余定理不互质

```
void crt()
{
    int t;
    while(cin>>t)
    {
        int flag=1;
        long long n1,a1;
        if (t) scanf("%lld%lld",&n1,&a1),t--;
        while(t--)
        {
            long long n2,a2,k1,k2;
            scanf("%lld%lld",&n2,&a2);
            if (flag==0)
                continue;
            long long d=exgcd(n1,n2,k1,k2);
            if ((a2-a1)%d!=0)
                flag=0;
            if (flag)
            {
                k1=(k1*(a2-a1)/d%(n2/d)+n2/d)%(n2/d);
                long long a=n1*k1+a1;
                long long n=n1/d*n2;
                n1=n;
                a1=a;
            }
        }
        if (flag) return a1;
        else return -1;
    }
}
```

7.13 组合数打表

```
const int CN=20;
long long c[CN][CN]= {};
void cinit()
{
    for(int i=0; i<CN; i++)
    {
        c[i][0]=c[i][i]=1;
        for(int j=1; j<i; j++)
            c[i][j]=c[i-1][j]+c[i-1][j-1];
    }
}
```

7.14 组合数在线

```
const int fcnt=100005;
const long long M=1000000007;
long long fac[fcnt];
void getfac()
{
    fac[0]=1;
    for (int i=1; i<fcnt; i++)
        fac[i]=fac[i-1]*i%M;
}
long long C(long long n,long long m)
{
    if (n<m)
        return 0;
    return fac[n]*inv(fac[m])%M*inv(fac[n-m])%M;
}
```

7.15 lucas 定理

```
const int fcnt=100005;
const long long M=1000000007;
long long fac[fcnt];
void getfac()
{
    fac[0]=1;
    for (int i=1; i<fcnt; i++)
        fac[i]=fac[i-1]*i%M;
}
long long C(long long n,long long m)
{
    if (n<m)
        return 0;
    return fac[n]*inv(fac[m])%M*inv(fac[n-m])%M;
}
long long lucas(long long n,long long m)
{
    if (m==0)
        return 1;
    return (lucas(n/M,m/M)*C(n%M,m%M))%M;
}
```

7.16 指数循环节

$$a^b \ mod \ c = a^{b \ mod \ phi(c)+phi(c)} \ mod \ c$$

输入 $m, n, a_1, a_2, a_3, \cdots, a_n$，求 $a_{123\cdots n} \ mod \ m$

```
long long phi(long long n)
{
    long long ans=n;
    long long x=sqrt(n);
    for (long long i=2; i<=x; i++)
    {
        if (n%i==0)
        {
            while(n%i==0)
                n/=i;
            ans=ans/i*(i-1);
        }
```

```
    }
    if (n>1)
        ans=ans/n*(n-1);
    return ans;
}
long long quickpow(long long a, long long b, long long c)
{
    if(b < 0) return 0;
    long long ret = 1;
    a %= c;
    for (; b; b >>= 1, a = (a * a) % c)
        if (b & 1)
            ret = (ret * a) % c;
    return ret;
}
long long solve(long long a[],int i,int n,long long c)
{
    if (i==n)
        return a[i]%c;
    long long p=phi(c);
    return quickpow(a[i],solve(a,i+1,n,p)%p+p,c);
}
int main()
{
    int m,n;
    int cas=0;
    while(scanf("%d%d",&m,&n)!=0)
    {
        long long a[50];
        for (int i=1; i<=n; i++)
            scanf("%lld",&a[i]);
        printf("Case #%d: %lld\n",++cas,solve(a,1,n,m));
    }
    return 0;
}
```

8 图论

8.1 邻接表

```
typedef int mytype;
const int NV=105;
const int NE=10005*2;
int he[NV],ecnt;
struct edge
{
    int v,next;
    mytype l;
} E[NE];
void adde(int u,int v,mytype l)
{
    E[++ecnt].v=v;
    E[ecnt].l=l;
    E[ecnt].next=he[u];
    he[u]=ecnt;
}
///初始化：
ecnt=0;
memset(he,-1,sizeof(he));
///调用：
for (int i=he[u]; i!=-1; i=E[i].next) ;
```

8.2 spfa

```
typedef int mytype;
const int NV=105;
const int NE=10005*2;
mytype dis[NV];
int pre[NV],vis[NV],vcnt[NV],he[NV],ecnt,flag;
struct edge
{
    int v,next;
    mytype l;
} E[NE];
void adde(int u,int v,mytype l)
{
    E[++ecnt].v=v;
    E[ecnt].l=l;
    E[ecnt].next=he[u];
    he[u]=ecnt;
}
void init(int n,int m,int s)
{
    ecnt=0;
    memset(pre,0,sizeof(pre));
    memset(vis,0,sizeof(vis));
    memset(vcnt,0,sizeof(vcnt));
    memset(he,-1,sizeof(he));
    for (int i=0; i<=n; i++)
        dis[i]=inf;
    dis[s]=0;
    for (int i=1; i<=m; i++)
    {
        int u,v;
        mytype l;
        scanf("%d%d%d",&u,&v,&l);
        adde(u,v,l);
        adde(v,u,l);
    }
}
void spfa(int n,int m,int s)
{
    queue<int> q;
    vis[s]=1;
    q.push(s);
    flag=0;
    while(!q.empty())
    {
        int u=q.front();
        q.pop();
        vis[u]=0;
        if(vcnt[u]>=n)
        {
            flag=1;
            break;
        }
        for (int i=he[u]; i!=-1; i=E[i].next)
            if (dis[u]+E[i].l<dis[E[i].v])
            {
                dis[E[i].v]=dis[u]+E[i].l;
                pre[E[i].v]=u;
                if (!vis[E[i].v])
                {
                    vis[E[i].v]=1;
                    q.push(E[i].v);
```

```
                    vcnt[E[i].v]++;
                }
            }
    }
}
```

8.3 dijkstra

```
typedef int mytype;
const int NV=1005;
int pre[NV],vis[NV];
mytype dis[NV],g[NV][NV];
void init(int n,int m,int s)
{
    memset(pre,0,sizeof(pre));
    memset(vis,0,sizeof(vis));
    for (int i=0; i<=n; i++)
        dis[i]=inf;
    dis[s]=0;
    for (int i=1; i<=n; i++)
        for (int j=1; j<=n; j++)
            g[i][j]=inf;
    for (int i=1; i<=m; i++)
    {
        int u,v,l;
        scanf("%d%d%d",&u,&v,&l);
        g[u][v]=l;
        g[v][u]=l;
    }
}
void dijkstra(int n)
{
    for (int i=1; i<=n; i++)
    {
        int u=0;
        for (int j=1; j<=n; j++)
            if (!vis[j]&&dis[j]<dis[u])
                u=j;
        vis[u]=1;
        for (int j=1; j<=n; j++)
            if (g[u][j]!=inf&&!vis[j]&&dis[u]+g[u][j]<dis[j])
            {
                dis[j]=dis[u]+g[u][j];
                pre[j]=u;
            }
    }
}
```

8.4 dijkstra+heap

```
typedef int mytype;
const int NV=105;
const int NE=10005*2;
mytype dis[NV];
int pre[NV],vis[NV],he[NV],ecnt;
struct edge
{
    int v,next;
    mytype l;
} E[NE];
```

```
11 void adde(int u,int v,mytype l)
12 {
13     E[++ecnt].v=v;
14     E[ecnt].l=l;
15     E[ecnt].next=he[u];
16     he[u]=ecnt;
17 }
18 void init(int n,int m,int s)
19 {
20     ecnt=0;
21     memset(pre,0,sizeof(pre));
22     memset(vis,0,sizeof(vis));
23     memset(he,-1,sizeof(he));
24     for (int i=0; i<=n; i++)
25         dis[i]=inf;
26     dis[s]=0;
27     for (int i=1; i<=m; i++)
28     {
29         int u,v;
30         mytype l;
31         scanf("%d%d%d",&u,&v,&l);
32         adde(u,v,l);
33         adde(v,u,l);
34     }
35 }
36 struct point
37 {
38     int u;
39     mytype l;
40     point(int u,mytype l):u(u),l(l) {}
41     bool operator<(const point &p) const
42     {
43         return l>p.l;
44     }
45 };
46 void dijkstra_heap(int s)
47 {
48     priority_queue<point> q;
49     q.push(point(s,0));
50     while(!q.empty())
51     {
52         point p=q.top();
53         q.pop();
54         int u=p.u;
55         if (vis[u]) continue;
56         vis[u]=1;
57         for (int i=he[u]; i!=-1; i=E[i].next)
58             if (!vis[E[i].v]&&p.l+E[i].l<dis[E[i].v])
59             {
60                 dis[E[i].v]=dis[u]+E[i].l;
61                 pre[E[i].v]=u;
62                 q.push(point(E[i].v,dis[E[i].v]));
63             }
64     }
65 }
```

8.5 floyd-warshall

```
1 const int NV=1005;
2 int g[NV][NV];
3 int path[NV][NV];
4 void init(int n,int m)
```

```
{
    for (int i=1; i<=n; i++)
        for (int j=1; j<=n; j++)
        {
            g[i][j]=inf;
            if (i==j) g[i][j]=0;
        }
    for (int i=1; i<=m; i++)
    {
        int u,v,l;
        scanf("%d%d%d",&u,&v,&l);
        if (l<g[u][v]) g[u][v]=l;
        if (l<g[v][u]) g[v][u]=l;
    }
}
void floyd_warshall(int n)
{
    for (int k=1; k<=n; k++)
        for (int i=1; i<=n; i++)
            for (int j=1; j<=n; j++)
                if (g[i][k]!=inf&&g[k][j]!=inf&&g[i][k]+g[k][j]<g[i][j])
                    g[i][j]=g[i][k]+g[k][j],path[i][j]=path[i][k];
}
void out(int u,int v)
{
    int tmp=u;
    printf("%d",u);
    while(tmp!=v)
    {
        printf("-->%d",path[tmp][v]);
        tmp=path[tmp][v];
    }
}
```

8.6 kruskal

```
typedef int mytype;
const int NV=105;
const int NE=10005;
struct edge
{
    int u,v;
    mytype l;
    bool operator<(const edge e) const
    {
        return l<e.l;
    }
} E[NE];
int f[NV],rk[NV];
int finds(int x)
{
    return f[x]==x?x:f[x]=finds(f[x]);
}
void uni(int a,int b)
{
    a=finds(a);
    b=finds(b);
    if (a==b) return;
    if (rk[a]>rk[b]) f[b]=a;
    else
    {
        if (rk[a]==rk[b]) rk[b]++;
```

```
27          f[a]=b;
28      }
29  }
30  void init(int n,int m)
31  {
32      memset(rk,0,sizeof(rk));
33      for (int i=1; i<=n; i++)
34          f[i]=i;
35      for (int i=1; i<=m; i++)
36          scanf("%d%d%d",&E[i].u,&E[i].v,&E[i].l);
37  }
38  mytype kruskal(int n,int m)
39  {
40      sort(E+1,E+m+1);
41      mytype ans=0;
42      for (int i=1; i<=m; i++)
43          if (finds(E[i].u)!=finds(E[i].v))
44          {
45              uni(E[i].u,E[i].v);
46              ans+=E[i].l;
47          }
48      return ans;
49  }
50  bool judge(int n)
51  {
52      int flag=0;
53      for (int i=1; i<=n; i++)
54          if (finds(i)==i)
55              flag++;
56      return flag==1;
57  }
```

8.7 prim

```
1   typedef int mytype;
2   const int NV=1005;
3   int pre[NV],vis[NV];
4   mytype dis[NV],g[NV][NV];
5   void init(int n,int m,int s)
6   {
7       memset(pre,0,sizeof(pre));
8       memset(vis,0,sizeof(vis));
9       for (int i=0; i<=n; i++)
10          dis[i]=inf;
11      dis[s]=0;
12      for (int i=1; i<=n; i++)
13          for (int j=1; j<=n; j++)
14              g[i][j]=inf;
15      for (int i=1; i<=m; i++)
16      {
17          int u,v,l;
18          scanf("%d%d%d",&u,&v,&l);
19          g[u][v]=l;
20          g[v][u]=l;
21      }
22  }
23  mytype prim(int n)
24  {
25      mytype ans=0;
26      for (int i=1; i<=n; i++)
27      {
28          int u=0;
```

```
        for (int j=1; j<=n; j++)
            if (!vis[j]&&dis[j]<dis[u])
                u=j;
        vis[u]=1;
        ans+=dis[u];
        for (int j=1; j<=n; j++)
            if (!vis[j]&&g[u][j]<dis[j])
            {
                dis[j]=g[u][j];
                pre[j]=u;
            }
    }
    return ans;
}
bool judge(int n)
{
    int cnt=0;
    for (int i=1; i<=n; i++)
        cnt+=vis[i];
    return cnt==n;
}
```

8.8 prim+heap

```
typedef int mytype;
const int NV=105;
const int NE=10005*2;
mytype dis[NV];
int pre[NV],vis[NV],he[NV],ecnt,pcnt;
struct edge
{
    int v,next;
    mytype l;
} E[NE];
void adde(int u,int v,mytype l)
{
    E[++ecnt].v=v;
    E[ecnt].l=l;
    E[ecnt].next=he[u];
    he[u]=ecnt;
}
void init(int n,int m,int s)
{
    ecnt=0;
    memset(pre,0,sizeof(pre));
    memset(vis,0,sizeof(vis));
    memset(he,-1,sizeof(he));
    for (int i=0; i<=n; i++)
        dis[i]=inf;
    dis[s]=0;
    for (int i=1; i<=m; i++)
    {
        int u,v;
        mytype l;
        scanf("%d%d%d",&u,&v,&l);
        adde(u,v,l);
        adde(v,u,l);
    }
}
struct point
{
    int u;
```

```
    mytype l;
    point(int u,mytype l):u(u),l(l) {}
    bool operator<(const point &p) const
    {
        return l>p.l;
    }
};
mytype prim_heap(int s)
{
    priority_queue<point> q;
    q.push(point(s,0));
    mytype ans=0;
    pcnt=0;
    while(!q.empty())
    {
        point p=q.top();
        q.pop();
        int u=p.u;
        if (vis[u])
            continue;
        vis[u]=1;
        ans+=p.l;//==dis[x]
        pcnt++;
        for (int i=he[u]; i!=-1; i=E[i].next)
            if (!vis[E[i].v]&&E[i].l<dis[E[i].v])
            {
                dis[E[i].v]=E[i].l;
                pre[E[i].v]=u;
                q.push(point(E[i].v,dis[E[i].v]));
            }
    }
    return ans;
}
bool judge(int n)
{
    return pcnt==n;
}
```

8.9 最小树形图朱刘算法

```
typedef int mytype;
const int NV=1005;
const int NE=NV*NV;
struct edge
{
    int u,v;
    mytype l;
} E[NE];
int pre[NV],ID[NV],vis[NV];
mytype In[NV];
void init(int m)
{
    for(int i=1; i<=m; i++)
        scanf("%d%d%d",&E[i].u,&E[i].v,&E[i].l);
}
mytype Directed_MST(int root,int NV,int NE)
{
//    memset(pre,0,sizeof(pre));
    mytype ret = 0;
    while(1)
    {
        //1.找最小入边
```

```
        for(int i=1; i<=NV; i++)
            In[i] = inf;
        for(int i=1; i<=NE; i++)
        {
            int u = E[i].u;
            int v = E[i].v;
            if(E[i].l < In[v] && u != v)
            {
                pre[v] = u;
                In[v] = E[i].l;
            }
        }
        for(int i=1; i<=NV; i++)
        {
            if(i == root)
                continue;
            if(fabs(In[i]-inf)<eps)
                return -1;//除了跟以外有点没有入边,则根无法到达它
        }
        //2.找环
        int cntnode = 0;
        memset(ID,-1,sizeof(ID));
        memset(vis,-1,sizeof(vis));
        In[root] = 0;
        for(int i=1; i<=NV; i++) //标记每个环
        {
            ret += In[i];
            int v = i;
            while(vis[v] != i && ID[v] == -1 && v != root)
            {
                vis[v] = i;
                v = pre[v];
            }
            if(v != root && ID[v] == -1)
            {
                ID[v] = ++cntnode;
                for(int u = pre[v] ; u != v ; u = pre[u])
                    ID[u] = cntnode;
            }
        }
        if(cntnode == 0)
            break;//无环
        for(int i=1; i<=NV; i++)
            if(ID[i] == -1)
                ID[i] = ++cntnode;
        //3.缩点,重新标记
        for(int i=1; i<=NE; i++)
        {
            int v = E[i].v;
            E[i].u = ID[E[i].u];
            E[i].v = ID[E[i].v];
            if(E[i].u != E[i].v)
            {
                E[i].l -= In[v];
            }
        }
        NV = cntnode;
        root = ID[root];
    }
    return ret;
}
bool judge(mytype ans)
{
```

```
86      return fabs(ans+1)>eps;
87  }
```

8.10 树的直径

```
1   typedef int mytype;
2   const int NV=40005;
3   const int NE=2*NV;
4   int vis[NV],he[NV],ecnt;
5   struct edge
6   {
7       int v,next;
8       mytype l;
9   } E[NE];
10  void adde(int u,int v,mytype l)
11  {
12      E[++ecnt].v=v;
13      E[ecnt].l=l;
14      E[ecnt].next=he[u];
15      he[u]=ecnt;
16  }
17  void init(int n,int m)
18  {
19      ecnt=0;
20      memset(he,-1,sizeof(he));
21      for (int i=1; i<=m; i++)
22      {
23          int u,v;
24          mytype l;
25          scanf("%d%d%d",&u,&v,&l);
26          adde(u,v,l);
27          adde(v,u,l);
28      }
29  }
30  int U;
31  mytype L;
32  void dfs(int u,int uu,mytype l)
33  {
34      if (l>L)
35      {
36          U=u;
37          L=l;
38      }
39      for (int i=he[u]; i!=-1; i=E[i].next)
40          if (E[i].v!=uu)
41              dfs(E[i].v,u,l+E[i].l);
42  }
43  mytype solve()
44  {
45      dfs(1,0,0);
46      dfs(U,0,0);
47      return L;
48  }
```

8.11 二分图最大匹配匈牙利算法

1、一个二分图中的最大匹配数等于这个图中的最小点覆盖数
König 定理是一个二分图中很重要的定理，它的意思是，一个二分图中的最大匹配数等于这个图中的最小点覆盖数。
最小点覆盖：假如选了一个点就相当于覆盖了以它为端点的所有边，你需要选择最少的点来

覆盖所有的边。

2、最小路径覆盖 =|G|-最大匹配数
在一个 $N*N$ 的有向图中，路径覆盖就是在图中找一些路经，使之覆盖了图中的所有顶点，且任何一个顶点有且只有一条路径与之关联。（如果把这些路径中的每条路径从它的起始点走到它的终点，那么恰好可以经过图中的每个顶点一次且仅一次）。如果不考虑图中存在回路，那么每条路径就是一个弱连通子集。
由上面可以得出：
1. 一个单独的顶点是一条路径
2. 如果存在一路径 $p_1, p_2, \ldots, p_k$,其中 p_1 为起点,p_k 为终点,那么在覆盖图中,顶点 $p_1, p_2, \ldots, p_k$ 不再与其它的顶点之间存在有向边最小路径覆盖就是找出最小的路径条数，使之成为 G 的一个路径覆盖。

2、二分图最大独立集 = 顶点数 -最大匹配数
独立集：图中任意两个顶点都不相连的顶点集合。

```
const int NV=505;
const int NE=10005;
int he[NV],ecnt,pre[NV],vis[NV];
struct edge
{
    int v,next;
} E[NE];
void adde(int u,int v)
{
    E[++ecnt].v=v;
    E[ecnt].next=he[u];
    he[u]=ecnt;
}
int dfs(int u)
{
    for(int i=he[u]; i!=-1; i=E[i].next)
    {
        int v=E[i].v;
        if(!vis[v])
        {
            vis[v]=1;
            if(pre[v]==0||dfs(pre[v]))
            {
                pre[v]=u;
                return 1;
            }
        }
    }
    return 0;
}
void init(int m)
{
    ecnt=0;
    memset(he,-1,sizeof(he));
    memset(pre,0,sizeof(pre));
    while(m--)
    {
        int u,v;
        scanf("%d%d",&u,&v);
        adde(u,v);
    }
}
int hungary(int n)
```

```
{
    int ans=0;
    for (int i=1; i<=n; i++)
    {
        memset(vis,0,sizeof(vis));
        ans+=dfs(i);
    }
    return ans;
}
```

8.12 网络流 Dinic 算法

```
const int NV=20005;
const int NE=500005;
int he[NV],ecnt;
int src,sink;
struct edge
{
    int v,next,f;
} E[2*NE];
void adde(int u,int v,int c)
{
    E[++ecnt].v=v;
    E[ecnt].f=c;
    E[ecnt].next=he[u];
    he[u]=ecnt;
    E[++ecnt].v=u;
    E[ecnt].f=0;
    E[ecnt].next=he[v];
    he[v]=ecnt;
}
void init()
{
    ecnt=0;
    memset(he,-1,sizeof(he));
}
queue<int> que;
bool vis[NV];
int dis[NV];
void bfs()
{
    memset(dis,0,sizeof(dis));
    while(!que.empty()) que.pop();
    vis[src]=1;
    que.push(src);
    while(!que.empty())
    {
        int u=que.front();
        que.pop();
        for (int i=he[u]; i!=-1; i=E[i].next)
            if (E[i].f&&!vis[E[i].v])
            {
                que.push(E[i].v);
                dis[E[i].v]=dis[u]+1;
                vis[E[i].v]=1;
            }
    }
}
int dfs(int u,int delta)
{
    if (u==sink)
        return delta;
```

```
    else
    {
        int ret=0;
        for (int i=he[u]; delta&&i!=-1; i=E[i].next)
            if (E[i].f&&dis[E[i].v]==dis[u]+1)
            {
                int dd=dfs(E[i].v,min(E[i].f,delta));
                E[i].f-=dd;
                E[(i+1)^1-1].f+=dd;
                delta-=dd;
                ret+=dd;
            }
        return ret;
    }
}
int maxflow()
{
    int ret=0;
    while(1)
    {
        memset(vis,0,sizeof(vis));
        bfs();
        if (!vis[sink]) return ret;
        ret+=dfs(src,inf);
    }
}
```

8.13 LCA 的 tarjan 离线算法

```
typedef int mytype;
const int NV=40005;
const int NE=NV;
const int NQ=10005;
mytype dis[NV],ans[NV];
int vis[NV],he[NV],hq[NV],ecnt,qcnt;
struct edge
{
    int v,next;
    mytype l;
} E[2*NE];
struct quer
{
    int v,next,i;
} q[2*NQ];
void adde(int u,int v,mytype l)
{
    E[++ecnt].v=v;
    E[ecnt].l=l;
    E[ecnt].next=he[u];
    he[u]=ecnt;
}
void addq(int u,int v,int i)
{
    q[++qcnt].v=v;
    q[qcnt].i=i;
    q[qcnt].next=hq[u];
    hq[u]=qcnt;
}
int fa[NV],rk[NV];
void init(int n,int m)
{
    ecnt=0;
```

```
    qcnt=0;
    memset(vis,0,sizeof(vis));
    memset(rk,0,sizeof(rk));
    memset(he,-1,sizeof(he));
    memset(hq,-1,sizeof(hq));
    for (int i=1; i<=m; i++)
    {
        int u,v;
        mytype l;
        scanf("%d%d%d",&u,&v,&l);
        adde(u,v,l);
        adde(v,u,l);
    }
}
int finds(int x)
{
    int k,j,r;
    r=x;
    while(r!=fa[r])
        r=fa[r];
    k=x;
    while(k!=r)
    {
        j=fa[k];
        fa[k]=r;
        k=j;
    }
    return r;
}
void tarjan(int u,mytype d)
{
    dis[u]=d;
    fa[u]=u;
    vis[u]=1;
    for (int i=he[u]; i!=-1; i=E[i].next)
        if (!vis[E[i].v])
            tarjan(E[i].v,d+E[i].l),fa[E[i].v]=u;
    for (int i=hq[u]; i!=-1; i=q[i].next)
        if (vis[q[i].v])
            ans[q[i].i]=dis[u]+dis[q[i].v]-2*dis[finds(q[i].v)];
}
void solve(int n,int m)
{
    init(n,m);
    int k;
    scanf("%d",&k);
    for (int i=1; i<=k; i++)
    {
        int u,v;
        scanf("%d%d",&u,&v);
        addq(u,v,i);
        addq(v,u,i);
    }
    tarjan(1,0);
    for (int i=1; i<=k; i++)
        printf("%d\n",ans[i]);
}
```

8.14 2-SAT

```
/*
x         <x', x >
```

```
~x        <x , x'>
x&y       <x', x > <y', y >
~(x&y)    <x , y'> <y , x'>
x|y       <x', y > <y', x >
~(x|y)    <x , x'> <y , y'>
x^y       <x , y'> <y , x'> <x', y > <y', x >
~(x^y)    <x , y > <y , x > <x', y'> <y', x'>
*/
```

```
const int NV=1005;
const int NE=2000005;
int low[2*NV],dfn[2*NV],stk[2*NV],instk[2*NV],he[2*NV],belong[2*NV],sol[2*NV];
int idcnt,ecnt,ccnt,top;
vector<int> c[2*NV];
struct edge
{
    int v,next;
} E[2*NE];
void adde(int u,int v)
{
    E[++ecnt].v=v;
    E[ecnt].next=he[u];
    he[u]=ecnt;
}
int getvalue(int n,int x)
{
    int r=x>n?x-n:x;
    if (sol[r]==-1)
        return -1;
    return x>n?!sol[r]:sol[r];
}
void dfs(int u)
{
    low[u]=dfn[u]=++idcnt;
    stk[++top]=u;
    instk[u]=1;
    for (int i=he[u],v; i!=-1; i=E[i].next)
        if (!dfn[v=E[i].v])
        {
            dfs(v);
            low[u]=min(low[u],low[v]);
        }
        else if (instk[v])
            low[u]=min(low[u],dfn[v]);
    if (dfn[u]==low[u])
    {
        stk[top+1]=-1;
        for (++ccnt; stk[top+1]!=u; --top)
        {
            c[ccnt].push_back(stk[top]);
            instk[stk[top]]=0;
            belong[stk[top]]=ccnt;
        }
    }
}
bool two_sat(int n,int m)
{
    ecnt=idcnt=ccnt=top=0;
    for (int i=1; i<=2*n; i++)
    {
        he[i]=-1;
        dfn[i]=instk[i]=0;
        c[i].clear();
```

```
    }
    for (int i=1; i<=n; i++)
        sol[i]=-1;
    for (int i=0; i<m; i++)
    {
        ///TODO: adde
    }
    adde(1,n+1);
    for (int i=1; i<=2*n; i++)
        if (!dfn[i])
            dfs(i);
    for (int i=1; i<=n; i++)
        if (belong[i]==belong[i+n])
            return 0;
    for (int i=1; i<=ccnt; i++)
    {
        int val=1;
        for (int j=0; j<c[i].size(); j++)
        {
            int cur=c[i][j];
            if (getvalue(n,cur)==0)
                val=0;
            for (int k=he[cur]; k!=-1; k=E[k].next)
                if (getvalue(n,E[k].v)==0)
                    val=0;
            if (val==0)
                break;
        }
        for (int j=0; j<c[i].size(); j++)
            if (c[i][j]>n)
                sol[c[i][j]-n]=!val;
            else
                sol[c[i][j]]=val;
    }
    return 1;
}
```

9 数学

9.1 高斯消元

```
const int MAXN=105;
double a[MAXN][MAXN],b[MAXN]; ///要注意-0.000的情况 +eps
int gauss_elimination(int n,double a[][MAXN],double b[])
{
    int i,j,k,row;
    double mx,t;
    for(k=1; k<=n; k++)
    {
        for(mx=0,i=k; i<=n; i++)
            if (fabs(a[i][k])>fabs(mx))
                mx=a[row=i][k];
        if(fabs(mx)<eps)
            return 0;
        if(row!=k)
        {
            for(j=k; j<=n; j++)
                swap(a[k][j],a[row][j]);
            swap(b[k],b[row]);
        }
        for(j=k+1; j<=n; j++)
```

```
            for(a[k][j]/=mx,i=k+1; i<=n; i++)
                a[i][j]-=a[i][k]*a[k][j];
        for(b[k]/=mx,i=k+1; i<=n; i++)
            b[i]-=b[k]*a[i][k];
    }
    for(i=n; i>=1; i--)
        for(j=i+1; j<=n; j++)
            b[i]-=a[i][j]*b[j];
    return 1;
}
```

9.2 二进制下的高斯消元

```
const int MAXN=105;
int a[MAXN][MAXN],b[MAXN];
void gauss(int n,int a[][MAXN],int b[])
{
    for (int i=1; i<=n; i++)
    {
        int k;
        for (int j=i; j<=n; j++)
            if (a[j][i])
            {
                k=j;
                break;
            }
        for (int j=1; j<=n; j++)
            swap(a[i][j],a[k][j]);
        swap(b[i],b[k]);
        for (int j=1; j<=n; j++)
            if (i!=j&&a[j][i])
            {
                for (k=1; k<=n; k++)
                    a[j][k]^=a[i][k];
                b[j]^=b[i];
            }
    }
}
```

9.3 高精度整数类

在进行大数的其他操作时要注意输出字符串为空的情况
WARNING! integer 类型与 long long 大小的整数进行乘除操作时，非常有可能溢出，尤其是除法。最好使用 integer 与 integer 的乘除操作。
另外，使用 integer 会产生大量开销，使用适当的数组长度可以很大程度上减少这种开销。

```
const int ra=10;
int ten[4]= {1,ra,ra*ra,ra*ra*ra};
int radix=ra*ra*ra*ra;
const int NV=1000;
struct integer
{
    int d[NV];
    integer()
    {
        *this=integer(0);
    }
    integer(int x)
    {
        for (int i=0; i<NV; i++) d[i]=0;
```

```
15          if (!x) d[0]=1;
16          while(x)
17          {
18              d[++d[0]]=x%radix;
19              x/=radix;
20          }
21      }
22      integer(long long x)
23      {
24          for (int i=0; i<NV; i++) d[i]=0;
25          if (!x) d[0]=1;
26          while(x)
27          {
28              d[++d[0]]=x%radix;
29              x/=radix;
30          }
31      }
32      integer(const char s[])
33      {
34          int len=strlen(s),i,j,k;
35          d[0]=(len-1)/4+1;
36          for (i=1; i<NV; i++) d[i]=0;
37          for (i=len-1; i>=0; i--)
38          {
39              j=(len-i-1)/4+1;
40              k=(len-i-1)%4;
41              d[j]+=ten[k]*(s[i]-'0');
42          }
43          while(d[0]>1&&d[d[0]]==0) d[0]--;
44      }
45      string tostring()
46      {
47          string s;
48          int i,j,temp;
49          for (i=3; i>=1; i--) if (d[d[0]]>=ten[i]) break;
50          temp=d[d[0]];
51          for (j=i; j>=0; j--)
52          {
53              s+=(char) (temp/ten[j]+'0');
54              temp%=ten[j];
55          }
56          for (i=d[0]-1; i>0; i--)
57          {
58              temp=d[i];
59              for (j=3; j>=0; j--)
60              {
61                  s+=(char) (temp/ten[j]+'0');
62                  temp%=ten[j];
63              }
64          }
65          return s;
66      }
67      void output()
68      {
69          int k=d[0];
70          printf("%d",d[k--]);
71          while(k) printf("%04d",d[k--]);
72          putchar('\n');
73      }
74  } d,mid1[15];
75  bool operator <(const integer &a,const integer &b)
76  {
77      if (a.d[0]!=b.d[0]) return a.d[0]<b.d[0];
```

```
78      for (int i=a.d[0]; i>0; i--)
79          if (a.d[i]!=b.d[i])
80              return a.d[i]<b.d[i];
81      return 0;
82  }
83  integer operator +(const integer &a,const integer &b)
84  {
85      integer c;
86      c.d[0]=max(a.d[0],b.d[0]);
87      int i,x=0;
88      for (i=1; i<=c.d[0]; i++)
89      {
90          x+=a.d[i]+b.d[i];
91          c.d[i]=x%radix;
92          x/=radix;
93      }
94      while(x)
95      {
96          c.d[++c.d[0]]=x%radix;
97          x/=radix;
98      }
99      return c;
100 }
101 integer operator -(const integer &a,const integer &b)
102 {
103     integer c;
104     c.d[0]=a.d[0];
105     int i,x=0;
106     for (i=1; i<=c.d[0]; i++)
107     {
108         x+=radix+a.d[i]-b.d[i];
109         c.d[i]=x%radix;
110         x=x/radix-1;
111     }
112     while(c.d[0]>1&&c.d[c.d[0]]==0) c.d[0]--;
113     return c;
114 }
115 integer operator *(const integer &a,const integer &b)
116 {
117     integer c;
118     c.d[0]=a.d[0]+b.d[0];
119     int i,j,x=0;
120     for (i=1; i<=a.d[0]; i++)
121     {
122         x=0;
123         for (j=1; j<=b.d[0]; j++)
124         {
125             x=a.d[i]*b.d[j]+x+c.d[i+j-1];
126             c.d[i+j-1]=x%radix;
127             x/=radix;
128         }
129         c.d[i+b.d[0]]=x;
130     }
131     while(c.d[0]>1&&c.d[c.d[0]]==0) c.d[0]--;
132     return c;
133 }
134 integer operator *(const integer &a,const long long &k)
135 {
136     integer c;
137     c.d[0]=a.d[0];
138     int i;
139     long long x=0;
140     for (i=1; i<=a.d[0]; i++)
```

```
141     {
142         x+=a.d[i]*k;
143         c.d[i]=x%radix;
144         x/=radix;
145     }
146     while(x>0)
147     {
148         c.d[++c.d[0]]=x%radix;
149         x/=radix;
150     }
151     while(c.d[0]>1&&c.d[c.d[0]]==0) c.d[0]--;
152     return c;
153 }
154 long long rem;
155 integer operator /(const integer &a,const long long &k)
156 {
157     integer c;
158     c.d[0]=a.d[0];
159     long long x=0;
160     for (int i=a.d[0]; i>=1; i--)
161     {
162         x+=a.d[i];
163         c.d[i]=x/k;
164         x%=k;
165         rem=x;
166         x*=radix;
167     }
168     while(c.d[0]>1&&c.d[c.d[0]]==0) c.d[0]--;
169     return c;
170 }
171 bool smaller(const integer &a,const integer &b,int delta)
172 {
173     if (a.d[0]+delta!=b.d[0]) return a.d[0]+delta<b.d[0];
174     for (int i=a.d[0]; i>0; i--)
175         if (a.d[i]!=b.d[i+delta])
176             return a.d[i]<b.d[i+delta];
177     return 1;
178 }
179 void Minus(integer &a,const integer &b,int delta)
180 {
181     int i,x=0;
182     for (i=1; i<=a.d[0]-delta; i++)
183     {
184         x+=radix+a.d[i+delta]-b.d[i];
185         a.d[i+delta]=x%radix;
186         x=x/radix-1;
187     }
188     while(a.d[0]>1&&a.d[a.d[0]]==0) a.d[0]--;
189 }
190 integer operator /(const integer &a,const integer &b)
191 {
192     integer c;
193     d=a;
194     int i,j,temp;
195     mid1[0]=b;
196     for (i=1; i<=13; i++) mid1[i]=mid1[i-1]*2;
197     for (i=a.d[0]-b.d[0]; i>=0; i--)
198     {
199         temp=8192;
200         for (j=13; j>=0; j--)
201         {
202             if (smaller(mid1[j],d,i))
203             {
```

```
                Minus(d,mid1[j],i);
                c.d[i+1]+=temp;
            }
            temp/=2;
        }
    }
    c.d[0]=max(1,a.d[0]-b.d[0]+1);
    while(c.d[0]>1&&c.d[c.d[0]]==0) c.d[0]--;
    return c;
}
bool operator ==(const integer &a,const integer &b)
{
    if (a.d[0]!=b.d[0]) return 0;
    for (int i=1; i<=a.d[0]; i++)
        if (a.d[i]!=b.d[i])
            return 0;
    return 1;
}
void init(int b) ///将大数切换至任意 <=10进制
{
    for (int i=1; i<=3; i++)
        ten[i]=ten[i-1]*b;
    radix=b*b*b*b;
}
```

9.4 分数类

num 为 0 则为 0，den 为 0 则不存在，都为 0 无法构造

```
struct frac
{
    long long num,den;
    frac(long long num=0,long long den=1)
    {
        if (den<0) num=-num,den=-den;
        long long g=__gcd(abs(num),den);
        this->num=num/g;
        this->den=den/g;
    }
    frac operator +(const frac &o) const
    {
        return frac(num*o.den+den*o.num,den*o.den);
    }
    frac operator -(const frac &o) const
    {
        return frac(num*o.den-den*o.num,den*o.den);
    }
    frac operator *(const frac &o) const
    {
        return frac(num*o.num,den*o.den);
    }
    frac operator /(const frac &o) const
    {
        return frac(num*o.den,den*o.num);
    }
    bool operator <(const frac &o) const
    {
        return num*o.den<den*o.num;
    }
    bool operator ==(const frac &o) const
    {
        return num*o.den==den*o.num;
```

```
    }
    void out()
    {
        printf("%lld/%lld",num,den);
    }
};
```

9.5 矩阵类

```
typedef long long mytype;
const int SZ=105;
const long long M=1000000007;
long long quickpow(long long a, long long b)
{
    if(b < 0) return 0;
    long long ret = 1;
    a %= M;
    for (; b; b >>= 1, a = (a * a) % M)
        if (b & 1)
            ret = (ret * a) % M;
    return ret;
}
long long inv(long long a)
{
    return quickpow(a,M-2);
}
struct mat
{
    int n,m;
    mytype a[SZ][SZ];
    void init()
    {
        memset(a,0,sizeof(a));
    }
    mat(int n=SZ,int m=SZ):n(n),m(m) {}
    mat unit()
    {
        mat t(n,n);
        t.init();
        for (int i=0; i<n; i++)
            t.a[i][i]=1;
        return t;
    }
    mytype *operator [](int n)
    {
        return *(a+n);
    }
    mat operator +(const mat &b)
    {
        mat t(n,m);
        for (int i=0; i<n; i++)
            for (int j=0; j<m; j++)
                t.a[i][j]=(a[i][j]+b.a[i][j]+M)%M;
        return t;
    }
    mat operator -(const mat &b)
    {
        mat t(n,m);
        for (int i=0; i<n; i++)
            for (int j=0; j<m; j++)
                t.a[i][j]=(a[i][j]-b.a[i][j]+M)%M;
        return t;
```

```
54      }
55      mat operator *(const mat &b)
56      {
57          mat t(n,m);
58          for(int i=0; i<n; i++)
59              for(int j=0; j<m; j++)
60              {
61                  t.a[i][j]=0;
62                  for(int k=0; k<m; k++)
63                      t.a[i][j]=(t.a[i][j]+(a[i][k]*b.a[k][j])%M)%M;
64              }
65          return t;
66      }
67      mat operator *(const mytype &b)
68      {
69          mat t(n,m);
70          for(int i=0; i<n; i++)
71              for(int j=0; j<m; j++)
72                  t.a[i][j]=a[i][j]*b%M;
73          return t;
74      }
75      mat operator /(const mytype &b)
76      {
77          mat t(n,m);
78          for(int i=0; i<n; i++)
79              for(int j=0; j<m; j++)
80                  t.a[i][j]=a[i][j]*inv(b)%M;
81          return t;
82      }
83      mat operator !()
84      {
85          mat t(n,m);
86          for(int i=0; i<n; i++)
87              for(int j=0; j<m; j++)
88                  t.a[i][j]=a[j][i];
89          return t;
90      }
91      mytype det()
92      {
93      }
94      mat invm(mat &a)
95      {
96      }
97      friend mat quickpow(mat a, mytype b)
98      {
99          if(b<0) return a.unit();
100         mat ret=a.unit();
101         for (; b; b>>=1,a=a*a)
102             if (b&1)
103                 ret=ret*a;
104         return ret;
105     }
106     void in()
107     {
108         for (int i=0; i<n; i++)
109             for (int j=0; j<m; j++)
110                 scanf("%lld",&a[i][j]);
111     }
112     void out()
113     {
114         for (int i=0; i<n; i++)
115             for (int j=0; j<m; j++)
116                 printf("%lld%c",a[i][j]," \n"[j==m-1]);
```

```
117     }
118 };
```

9.6 分治法等比数列求和

$1 + q^1 + \cdots + q^n$

```
1  long long qsum(long long q,long long n)
2  {
3      n++;
4      long long ans=0,temp=1;
5      while(n>0)
6      {
7          if (n&1)
8              ans=((ans*q)%M+temp%M)%M;
9          temp=(temp*(1+q))%M;
10         q=(q*q)%M;
11         n>>=1;
12     }
13     return ans;
14 }
15
16 mat qsum(mat q,long long n)
17 {
18     n++;
19     mat unit=q.unit();
20     mat ans(q.n,q.m),temp=unit;
21     ans.init();
22     while(n>0)
23     {
24         if (n&1) ans=ans*q+temp;
25         temp=temp*(unit+q);
26         q=q*q;
27         n>>=1;
28     }
29     return ans;
30 }
```

9.7 自适应 Simpson 积分法

```
1  double simpson(double a,double b)
2  {
3      double c = a + (b-a)/2;
4      return (f(a)+4*f(c)+f(b))*(b-a)/6;
5  }
6  double asr(double a,double b,double epss,double A)
7  {
8      double c = a+(b-a)/2;
9      double L = simpson(a,c) , R = simpson(c,b);
10     if (fabs(L+R-A) <= 15*epss) return L+R+(L+R-A)/15;
11     return asr(a,c,epss/2,L) + asr(c,b,epss/2,R);
12 }
13 double solve(double l,double r)
14 {
15     return asr(l,r,eps,simpson(l,r));
16 }
```

9.8 Romberg 积分法

```
double romberg(double l,double r,double epss=eps)
{
    vector<double> t;
    double h=r-l,last,curr;
    int k=1,i=1;
    t.push_back(h*(f(l)+f(r))/2);
    do
    {
        last=t.back();
        curr=0;
        double x=l+h/2;
        for (int j=0; j<k; j++)
        {
            curr+=f(x);
            x+=h;
        }
        curr=(t[0]+h*curr)/2;
        double k1=4.0/3,k2=1.0/3;
        for (int j=0; j<i; j++)
        {
            double temp=k1*curr-k2*t[j];
            t[j]=curr;
            curr=temp;
            k2/=4*k1-k2;
            k1=k2+1;
        }
        t.push_back(curr);
        k*=2;
        h/=2;
        i++;
    }
    while(fabs(last-curr)>epss);
    return t.back();
}
```

9.9 De Bruijn 序列

k 为元素值范围，n 为串长度。

```
void db(int n,int k,vector<int> &v,int a[],int t=1,int p=1)
{
    if (t>n)
    {
        if (n%p==0)
            for (int i=1; i<=p; i++)
                v.push_back(a[i]);
    }
    else
    {
        a[t]=a[t-p];
        db(n,k,v,a,t+1,p);
        for (int i=a[t-p]+1; i<k; i++)
        {
            a[t]=i;
            db(n,k,v,a,t+1,t);
        }
    }
}
vector<int> de_bruijn(int n,int k)
{
    vector<int> v;
    int a[n*k];
```

```
    memset(a,0,sizeof(a));
    db(n,k,v,a);
    return v;
}
```

9.10 格雷码

```
vector<unsigned long> gray(int n)
{
    vector<unsigned long> res;
    for (unsigned long i=0; i<(1<<n); i++)
        res.push_back(i^(i>>1));
    return res;
}
int main()
{
    vector<unsigned long> v=gray(8);
    for (int i=0; i<v.size(); i++)
        cout<<bitset<8>(v[i])<<endl;
    return 0;
}
```

9.11 表达式求值

```
char str1[1005], str2[1005];
int s1[1005], s2[1005];
int nei[300], wai[300]; //运算符优先级
int cust[1005]; //自定义变量
stack<int> optr;
stack<long long> opnd;
stack<int> neg;
bool flag;
long long calculate(long long x, long long y, char c)
{
    switch(c)
    {
    case '+':
        return x + y;
    case '-':
        return x - y;
    case '*':
        return x * y;
    case '/':
        return x / y;
    case '^':
        long long tmp = 1;
        for(int i = 0; i < y; i++) tmp *= x;
        return tmp;
    }
}
char cmp(char a, char b)
{
    if(nei[a] > wai[b]) return '>';
    else if(nei[a] == wai[b]) return '=';
    return '<';
}
long long calc(int *s)
{
    flag=1;
    int i = 0;
```

```
37      long long num;
38      int sgn=0;
39      while(s[i] != '#' || optr.top() != '#')
40      {
41          num = 0;
42          if(!flag) return -1;
43          if(s[i] >= '0' && s[i] <= '9')
44          {
45              while(s[i] >= '0' && s[i] <= '9')
46              {
47                  num *= 10;
48                  num += s[i] - '0';
49                  i++;
50              }
51              opnd.push(num);
52              sgn++;
53          }
54          else if(s[i] >= 300)
55          {
56              opnd.push(cust[s[i] - 300]);
57              i++;
58              sgn++;
59          }
60          else
61          {
62              if (s[i]=='(') neg.push(sgn),sgn=0;
63              switch(cmp(optr.top(), s[i]))
64              {
65              case '<' :
66                  optr.push(s[i]);
67                  i++;
68                  break;
69              case '=' :
70                  optr.pop();
71                  if (s[i]==')') sgn=neg.top()+1,neg.pop();
72                  i++;
73                  break;
74              case '>' :
75                  if(opnd.empty())
76                  {
77                      flag = false;
78                      return -1;
79                  }
80                  long long ta = opnd.top();
81                  opnd.pop();
82                  if ((opnd.empty()||sgn==1)&&(optr.top()=='+'||optr.top()=='-'))
83                  {
84                      if (optr.top()=='-') opnd.push(-ta);
85                      else opnd.push(ta);
86                      optr.pop();
87                      break;
88                  }
89                  if(opnd.empty())
90                  {
91                      flag = false;
92                      return -1;
93                  }
94                  long long tb = opnd.top();
95                  opnd.pop();
96                  opnd.push(calculate(tb, ta, optr.top()));
97                  optr.pop();
98                  break;
99              }
```

```
100         }
101     }
102     return opnd.top();
103 }
104 void init()
105 {
106     nei['+'] = 2,wai['+'] = 1;
107     nei['-'] = 2,wai['-'] = 1;
108     nei['*'] = 4,wai['*'] = 3;
109     nei['/'] = 4,wai['/'] = 3;
110     ///乘方自右向左结合，内外交换后可变成自左向右结合
111     nei['^'] = 5,wai['^'] = 6;
112     nei[')'] = 8,wai[')'] = 0;
113     nei['('] = 0,wai['('] = 8;
114     nei['#'] =-1,wai['#'] =-1;
115 }
116 void initstr(int *s,char *str)
117 {
118     int len = 0;
119     for(int i = 0; str[i]; i++)
120     {
121         if(!isspace(str[i]))
122         {
123             s[len++] = str[i];
124             //在栈中计算时，超过300的数视为自定义数
125             //cust数组从0开始
126             ///'a'-'z'之间的数用cust数组的数计算
127             if (isalpha(s[len-1])) s[len-1]+=-'a'+300;
128         }
129     }
130     s[len++] = '#';
131     s[len] = 0;
132 }
133 void initstack()
134 {
135     while(!opnd.empty()) opnd.pop();
136     while(!optr.empty()) optr.pop();
137     optr.push('#');
138 }
139 int main()
140 {
141     init();
142     int t;
143     scanf("%d\n", &t);
144     srand(time(0));
145     while(t--)
146     {
147         gets(str1);
148         initstr(s1,str1);
149         gets(str2);
150         initstr(s2,str2);
151         flag = 1;
152         for(int i = 0; i < 10; i++)
153         {
154             for(int j = 0; j < 26; j++) cust[j] = abs(rand()) % 100;
155             initstack();
156             long long t1 = calc(s1);
157             initstack();
158             long long t2 = calc(s2);
159             if(t1 != t2)
160             {
161                 flag = 0;
162                 break;
```

```
163             }
164         }
165         if(flag) puts("YES");
166         else puts("NO");
167     }
168     return 0;
169 }
```

9.12 卡特兰数

$\frac{1}{n+1}C_{2n}^{n}$
1, 2, 5, 14, 42, 132, 429, 1430, 4862, 16796, 58786, 208012, 742900, 2674440, 9694845, 35357670, 129644790, 477638700, 1767263190, 6564120420, 24466267020, 91482563640, 343059613650, 1289904147324, 4861946401452
递推式： h(n)=h(n-1)*(4*n-2)/(n+1);

9.13 求和公式

$\sum k = \frac{n\times(n+1)}{2}$
$\sum 2k-1 = n^2$
$\sum k^2 = \frac{n\times(n+1)\times(2n+1)}{6}$
$\sum (2k-1)^2 = \frac{n\times(4n^2-1)}{3}$
$\sum k^3 = (\frac{n\times(n+1)}{2})^2$
$\sum (2k-1)^3 = n^2\times(2n^2-1)$
$\sum k^4 = \frac{n\times(n+1)\times(2n+1)\times(3n^2+3n-1)}{30}$
$\sum k^5 = \frac{n^2\times(n+1)^2\times(2n^2+2n-1)}{12}$
$\sum k\times(k+1) = \frac{n\times(n+1)\times(n+2)}{3}$
$\sum k\times(k+1)\times(k+2) = \frac{n\times(n+1)\times(n+2)\times(n+3)}{4}$
$\sum k\times(k+1)\times(k+2)\times(k+3) = \frac{n\times(n+1)\times(n+2)\times(n+3)\times(n+4)}{5}$

9.14 小公式

球扇形：
全面积： $T=\pi r(2h+r_0)$，h 为球冠高，r_0 为球冠底面半径
体积： $V=\frac{2\pi r^2 h}{3}$
Pick **公式：** $A=E\times 0.5+I-1$（A 是多边形面积，E 是边界上的整点，I 是多边形内部的整点）

海伦公式： $S=\sqrt{p(p-a)(p-b)(p-c)}$，其中 $p=\frac{(a+b+c)}{2}$，abc 为三角形的三条边长

求 $\binom{n}{k}$ 中素因子 P 的个数：

1. 把 n 转化为 P 进制，并记它每个位上的和为 $S1$
2. 把 $n-k$，k 做同样的处理，得到 $S2$，$S3$

则 $\binom{n}{k}$ 中素因子 P 的个数：$\frac{S2+S3-S1}{P-1}$

部分错排公式：
$n+m$ 个数中 m 个数必须错排求排列数

```
dp[i] = n*dp[i-1]+(i-1)*(dp[i-1]+dp[i-2]);
dp[0] = n!;
dp[1] = n*n!;
```

$dp[m]$ 为所求解

10 博弈

10.1 巴什博弈

只有一堆 n 个物品，两个人轮流从这堆物品中取物，规定每次至少取一个，最多取 m 个。最后取光者得胜。

```
int bash(int n, int m)
{
    if(n%(m+1) != 0) return 1;
    else return 0;
}
int main()
{
    int n,m;
    while(~scanf("%d%d",&n,&m))
    {
        if (bash(n,m))
        {
            if (n>m)
                printf("%d",n%(m+1));
            else
            {
                printf("%d",n);
                for (int i=n+1; i<=m; i++)
                    printf(" %d",i);
            }
            puts("");
        }
        else puts("none");
    }
    return 0;
}
```

10.2 尼姆博弈

有 m 堆各若干个物品，两个人轮流从某一堆取任意多的物品，规定每次至少取一个，多者不限，最后取光者得胜。

```
int sg(int n,int numsg[])
{
    int ans = 0;
    for(int i=1; i <= n; i++)
        ans ^= numsg[i];
    if(ans == 0)
        printf("No\n");
    else
    {
        printf("Yes\n");
        for(int i = 1; i <= n; i++)
        {
            int x = ans ^ numsg[i];
            if(x < numsg[i])
                printf("%d %d\n", numsg[i], x);
```

```
        }
    }
}
```

10.3 威佐夫博弈

有两堆各若干个物品，两个人轮流从某一堆或同时从两堆中取同样多的物品，规定每次至少取一个，多者不限，最后取光者得胜。

```
int wzf(int n, int m)
{
    if(n > m)
        swap(n, m);
    int k = m-n;
    int a = (k * (1.0 + sqrt(5.0))/2.0);
    if(a == n)
        return 0;
    else
        return 1;
}
int main()
{
    int a,b;
    while(scanf("%d%d",&a,&b),a|b)
    {
        if (wzf(a,b))
        {
            puts("1");
            if(a>b)
                swap(a,b);
            //第一种
            int k=b-a;
            double q=(1+sqrt(5.0))/2;
            int ak=k*q;
            //特殊情况
            if(a==0)
                puts("0 0");
            //同时减
            if(a>=ak)
                printf("%d %d\n",ak,ak+k);
            for (int i=1; i<=b; i++)
            {
                //b减一点
                if(a==(int)(i*q)&&b>a+i)
                    printf("%d %d\n",a,a+i);
                //b减很多
                if(a==(int)(i*q)+i)
                    printf("%d %d\n",a-i,a);
                //a减一点
                if(b==(int)(i*q)+i&&b-i<a)
                    printf("%d %d\n",b-i,b);
            }
            //第二种
            //同时减
            for(int i=a-1,j=b-1; i>=0,j>=0; i--,j--)
                if(wzf(i,j)==0)
                {
                    cout<<i<<' '<<j<<endl;
                    break;
                }
            //a减一点
            for(int i=a-1; i>=0; i--)
```

```
                if(wzf(i,b)==0)
                {
                    cout<<i<<' '<<b<<endl;
                    break;
                }
            for(int i=b-1; i>=0; i--)
                if(wzf(i,a)==0)
                {
                    //b减很多
                    if (i<a)
                        cout<<i<<' '<<a<<endl;
                    //b减一点
                    else
                        cout<<a<<' '<<i<<endl;
                }

        }
        else
            puts("0");
    }
    return 0;
}
```

11 计算几何

```
#include<cstdio>
#include<cstdlib>
#include<cstring>
#include<cmath>
#include<ctime>
#include<cassert>
#include<climits>
#include<iostream>
#include<algorithm>
#include<string>
#include<vector>
#include<deque>
#include<list>
#include<set>
#include<map>
#include<stack>
#include<queue>
#include<numeric>
#include<iomanip>
#include<bitset>
#include<sstream>
#include<fstream>
#define debug puts("------")
#define pi (acos(-1.0))
#define eps (1e-8)
#define inf (1<<30)
#define INF (1ll<<62)
using namespace std;
inline double sqr(const double &x)
{
    return x * x;
}
inline int sgn(const double &x)
{
    return x < -eps ? -1 : x > eps;
}
```

```
37 struct point
38 {
39     double x, y;
40     point(const double &x = 0, const double &y = 0): x(x), y(y) {}
41     friend point operator + (const point &a, const point &b)
42     {
43         return point(a.x + b.x, a.y + b.y);
44     }
45     friend point operator - (const point &a, const point &b)
46     {
47         return point(a.x - b.x, a.y - b.y);
48     }
49     friend point operator * (const point &a, const double &b)
50     {
51         return point(a.x * b, a.y * b);
52     }
53     friend point operator * (const double &a, const point &b)
54     {
55         return point(a * b.x, a * b.y);
56     }
57     friend point operator / (const point &a, const double &b)
58     {
59         return point(a.x / b, a.y / b);
60     }
61     friend bool operator == (const point &a, const point &b)
62     {
63         return !sgn(a.x - b.x) && !sgn(a.y - b.y);
64     }
65     friend bool operator < (const point &a, const point &b)
66     {
67         return sgn(a.x - b.x) < 0 || (sgn(a.x - b.x) == 0 && sgn(a.y - b.y) < 0)
            ;
68     }
69     double norm()
70     {
71         return sqrt(sqr(x) + sqr(y));
72     }
73     friend double det(const point &a, const point &b)
74     {
75         return a.x * b.y - a.y * b.x;
76     }
77     friend double dot(const point &a, const point &b)
78     {
79         return a.x * b.x + a.y * b.y;
80     }
81     friend double dist(const point &a, const point &b)
82     {
83         return (a - b).norm();
84     }
85     double arg()
86     {
87         return atan2(y, x);
88 //        double res = atan2(y, x);   //(-pi, pi]
89 //        if(res < -pi / 2 + eps) res += 2 * pi; //eps修正精度
90 //        return res;
91     }
92     //逆时针旋转angle弧度
93     point rotate(const double &angle)
94     {
95         return rotate(cos(angle), sin(angle));
96     }
97     point rotate(const point &p, const double &angle)
98     {
```

```
99          return (*this-p).rotate(angle)+p;
100     }
101     point rotate(const double &cosa, const double &sina)
102     {
103         return point(x * cosa - y * sina, x * sina + y * cosa);
104     }
105     int in()
106     {
107         return scanf("%lf %lf", &x, &y);
108     }
109     void out()
110     {
111         printf("%.2f %.2f\n", x, y);
112     }
113 };
114 
115 struct line
116 {
117     point s, t;
118     line(const point &s = point(), const point &t = point()): s(s), t(t) {}
119     point vec() const
120     {
121         return t - s;
122     }
123     double norm() const
124     {
125         return vec().norm();
126     }
127     //点在直线上
128     bool ispointonline(const point &p) const
129     {
130         return sgn(det(p - s, t - s)) == 0;
131     }
132     //点在线段上
133     bool ispointonseg(const point &p) const
134     {
135         return ispointonline(p) && sgn(dot(p - s, p - t)) <= 0;
136     }
137     //点在线段上不含端点
138     bool ispointonsegex(const point &p)
139     {
140         return ispointonline(p) && sgn(dot(p - s, p - t)) < 0;
141     }
142     //点到直线的垂足
143     point pointprojline(const point &p)
144     {
145         return s + vec() * ((dot(p - s, vec()) / norm()) / (norm()));
146     }
147     //点到直线的距离
148     double pointdistline(const point &p)
149     {
150         return fabs(det(p - s, vec()) / norm());
151     }
152     //点到线段的距离
153     double pointdistseg(const point &p)
154     {
155         if (sgn(dot(p - s, t - s)) < 0) return (p - s).norm();
156         if (sgn(dot(p - t, s - t)) < 0) return (p - t).norm();
157         return pointdistline(p);
158     }
159     //判断两直线是否平行
160     friend bool parallel(const line &l1, const line &l2)
161     {
```

```
162         return !sgn(det(l1.vec(), l2.vec()));
163     }
164     //判断两个点是否在直线的同一侧
165     friend bool sameside(const line &l, const point &a, const point &b)
166     {
167         return sgn(det(b - l.s, l.vec())) * sgn(det(a - l.s, l.vec())) > 0;
168     }
169     //两直接的交点
170     friend point linexline(const line l1, const line l2) //
            利用相似三角形对应边成比例
171     {
172         double s1 = det(l1.s - l2.s, l2.vec());
173         double s2 = det(l1.t - l2.s, l2.vec());
174         return (l1.t * s1 - l1.s * s2) / (s1 - s2);
175     }
176     //判断线段交
177     friend bool issegxseg(const line &l1, const line &l2)
178     {
179         if(!sgn(det(l2.s - l1.s, l1.vec())) && !sgn(det(l2.t - l1.s, l1.vec())))
180         {
181             return l1.ispointonseg(l2.s) ||
182                    l1.ispointonseg(l2.t) ||
183                    l2.ispointonseg(l1.s) ||
184                    l2.ispointonseg(l1.t);
185         }
186         return !sameside(l1, l2.s, l2.t) && !sameside(l2, l1.s, l1.t);
187     }
188     //直线沿法线方向移动d距离
189     friend line move(const line &l, const double &d)
190     {
191         point t = l.vec();
192         t = t / t.norm();
193         t = t.rotate(pi / 2);
194         return line(l.s + t * d, l.t + t * d);
195     }
196     int in()
197     {
198         s.in();
199         return t.in();
200     }
201     void out()
202     {
203         s.out(), t.out();
204     }
205 };
206
207 struct polygon
208 {
209 #define next(i) ((i+1)%n)
210     int n;
211     vector<point> p;
212     polygon(int n = 0): n(n)
213     {
214         p.resize(n);
215     }
216 //    polygon(vector<point> &v):p(v){}
217     //多边形周长
218     double perimeter()
219     {
220         double sum = 0;
221         for (int i = 0; i < n; i++)
222             sum += (p[next(i)] - p[i]).norm();
223         return sum;
```

```
224     }
225     //多边形面积
226     double area()
227     {
228         double sum = 0;
229         for (int i = 0; i < n; i++)
230             sum += det(p[i], p[next(i)]);
231         return sum / 2 + eps; //要加eps吗？
232     }
233     //判断点与多边形的位置关系，0外，1内,2边上
234     int pointin(const point &t)
235     {
236         int num = 0;
237         for (int i = 0; i < n; i++)
238         {
239             if (line(p[i], p[next(i)]).ispointonseg(t))
240                 return 2;
241             int k = sgn(det(p[next(i)] - p[i], t - p[i]));
242             int d1 = sgn(p[i].y - t.y);
243             int d2 = sgn(p[next(i)].y - t.y);
244             if (k > 0 && d1 <= 0 && d2 > 0) num++;
245             if (k < 0 && d2 <= 0 && d1 > 0) num--;
246         }
247         return num != 0;
248     }
249     //多边形重心
250     point masscenter()
251     {
252         point ans;
253         if (sgn(area()) == 0) return ans;
254         for (int i = 0; i < n; i++)
255             ans = ans + (p[i] + p[next(i)]) * det(p[i], p[next(i)]);
256         return ans / area() / 6 + eps; //要加eps吗？
257     }
258     //多边形边界上格点的数量
259     int borderpointnum()
260     {
261         int num = 0;
262         for (int i = 0; i < n; i++)
263         {
264             int a = fabs(p[next(i)].x - p[i].x);
265             int b = fabs(p[next(i)].y - p[i].y);
266             num += __gcd(a, b);
267         }
268         return num;
269     }
270     //多边形内格点数量
271     int insidepointnum()
272     {
273         return int(area()) + 1 - borderpointnum() / 2;
274     }
275     void in()
276     {
277         for (int i = 0; i < n; i++)
278             p[i].in();
279     }
280     void out()
281     {
282         for (int i = 0; i < n; i++)
283             p[i].out();
284     }
285 };
286
```

```
287 struct convex : public polygon
288 {
289     convex(int n = 0): polygon(n) {}
290 //    convex(vector<point> &v):polygon(v){}
291     //需要先求凸包，若凸包每条边除端点外都有点，则可唯一确定凸包
292     bool isunique(vector<point> &v)
293     {
294         if (sgn(area()) == 0)
295             return 0;
296         for (int i = 0; i < n; i++)
297         {
298             line l(p[i], p[next(i)]);
299             bool flag = 0;
300             for (int j = 0; j < v.size(); j++)
301                 if (l.ispointonsegex(v[j]))
302                 {
303                     flag = 1;
304                     break;
305                 }
306             if (!flag)
307                 return 0;
308         }
309         return 1;
310     }
311     //O(n)时间内判断点是否在凸包内
312     bool containon(const point &a)
313     {
314         int sign = 0;
315         for (int i = 0; i < n; i++)
316         {
317             int x = sgn(det(p[i] - a, p[next(i)] - a));
318             if (x)
319             {
320                 if (sign)
321                 {
322                     if (sign != x)
323                         return 0;
324                 }
325                 else
326                     sign = x;
327             }
328         }
329         return 1;
330     }
331     //O(logn)时间内判断点是否在凸包内
332     bool containologn(const point &a)
333     {
334         point g = (p[0] + p[n / 3] + p[2 * n / 3]) / 3.0;
335         int l = 0, r = n;
336         while(l + 1 < r)
337         {
338             int m = (l + r) / 2;
339             if (sgn(det(p[l] - g, p[m] - g)) > 0)
340             {
341                 if (sgn(det(p[l] - g, a - g)) >= 0 && sgn(det(p[m] - g, a - g))
                        < 0)
342                     r = m;
343                 else
344                     l = m;
345             }
346             else
347             {
```

```
348                 if (sgn(det(p[l] - g, a - g)) < 0 && sgn(det(p[m] - g, a - g))
                        >= 0)
349                     l = m;
350                 else
351                     r = m;
352             }
353         }
354         return sgn(det(p[r % n] - a, p[l] - a)) - 1;
355     }
356     //最远点对（直径）
357     int first, second; //最远的两个点对应标号
358     double diameter()
359     {
360         double mx = 0;
361         if (n == 1)
362         {
363             first = second = 0;
364             return mx;
365         }
366         for (int i = 0, j = 1; i < n; i++)
367         {
368             while(sgn(det(p[next(i)] - p[i], p[j] - p[i]) -
369                       det(p[next(i)] - p[i], p[next(j)] - p[i])) < 0)
370                 j = next(j);
371             double d = dist(p[i], p[j]);
372             if (d > mx)
373             {
374                 mx = d;
375                 first = i;
376                 second = j;
377             }
378             d = dist(p[next(i)], p[next(j)]);
379             if (d > mx)
380             {
381                 mx = d;
382                 first = next(i);
383                 second = next(j);
384             }
385         }
386         return mx;
387     }
388     //凸包是否与直线有交点O(log(n)), 需要On的预处理, 适合判断与直线集是否有交点
389     vector<double> ang; //角度
390     bool isinitangle;
391     int finda(const double &x)
392     {
393         return upper_bound(ang.begin(), ang.end(), x) - ang.begin();
394     }
395     double getangle(const point &p) //获取向量角度[0, 2pi]
396     {
397         double res = atan2(p.y, p.x);   //(-pi, pi]
398 //        if (res < 0) res += 2 * pi; //为何不可以
399         if(res < -pi / 2 + eps) res += 2 * pi; //eps修正精度
400         return res;
401     }
402     void initangle()
403     {
404         for(int i = 0; i < n; i++)
405             ang.push_back(getangle(p[next(i)] - p[i]));
406         isinitangle = 1;
407     }
408     bool isxline(const line &l)
409     {
```

```
410         if(!isinitangle) initangle();
411         int i = finda(getangle(l.t - l.s));
412         int j = finda(getangle(l.s - l.t));
413         if(sgn(det(l.t - l.s, p[i] - l.s) * det(l.t - l.s, p[j] - l.s) >= 0))
414             return 0;
415         return 1;
416     }
417 };
418 convex convexhull(vector<point> &a)
419 {
420     //从一个vector获取凸包
421     convex res(2 * a.size() + 5); //为何？经测试好像只需要a.size()？
422     sort(a.begin(), a.end());
423     a.erase(unique(a.begin(), a.end()), a.end());
424     int m = 0;
425     for (int i = 0; i < a.size(); i++)
426     {
427         //<=0则不含边界，<0则含边界
428         while(m > 1 && sgn(det(res.p[m - 1] - res.p[m - 2], a[i] - res.p[m - 2])
                ) <= 0)
429             m--;
430         res.p[m++] = a[i];
431     }
432     int k = m;
433     for (int i = a.size() - 2; i >= 0; i--)
434     {
435         while(m > k && sgn(det(res.p[m - 1] - res.p[m - 2], a[i] - res.p[m - 2])
                ) <= 0)
436             m--;
437         res.p[m++] = a[i];
438     }
439     if (m > 1) m--;
440     res.p.resize(m);
441     res.n = m;
442     return res;
443 }
444
445 struct halfplane: public line
446 {
447     //ax+by+c<=0
448     double a, b, c;
449     //s->t的左侧表示半平面
450     halfplane(const point &s = point(), const point &t = point()): line(s, t)
451     {
452         a = t.y - s.y;
453         b = s.x - t.x;
454         c = det(t, s);
455     }
456     halfplane(double a, double b, double c): a(a), b(b), c(c) {}
457     //求点p带入直线方程的值
458     double calc(const point &p) const
459     {
460         return p.x * a + p.y * b + c;
461     }
462     //好像跟linexline一样，那个是4个点计算。这个是用abc与两点进行计算
463     friend point halfxline(const halfplane &h, const line &l)
464     {
465         point res;
466         double t1 = h.calc(l.s), t2 = h.calc(l.t);
467         res.x = (t2 * l.s.x - t1 * l.t.x) / (t2 - t1);
468         res.y = (t2 * l.s.y - t1 * l.t.y) / (t2 - t1);
469         return res;
470     }
```

```
471     //用abc进行计算 尚未测试
472     friend point halfxhalf(const halfplane &h1, const halfplane&h2)
473     {
474         return point((h1.b * h2.c - h1.c * h2.b) / (h1.a * h2.b - h2.a * h1.b) +
                eps,
475                      (h1.a * h2.c - h2.a * h1.c) / (h1.b * h2.a - h1.a * h2.b) +
                         eps);
476     }
477     //凸多边形与半平面交(cut)
478     friend convex halfxconvex(const halfplane &h, const convex &c)
479     {
480         convex res;
481         for (int i = 0; i < c.n; i++)
482         {
483             if (h.calc(c.p[i]) < -eps)
484                 res.p.push_back(c.p[i]);
485             else
486             {
487                 int j = i - 1;
488                 if (j < 0) j = c.n - 1;
489                 if (h.calc(c.p[j]) < -eps)
490                     res.p.push_back(halfxline(h, line(c.p[j], c.p[i])));
491                 j = i + 1;
492                 if (j == c.n) j = 0;
493                 if (h.calc(c.p[j]) < -eps)
494                     res.p.push_back(halfxline(h, line(c.p[i], c.p[j])));
495             }
496         }
497         res.n = res.p.size();
498         return res;
499     }
500     //点在半平面内
501     friend int satisfy(const point &p, const halfplane &h)
502     {
503         return sgn(det(p - h.s, h.t - h.s)) <= 0;
504     }
505     friend bool operator <(const halfplane &h1, const halfplane &h2)
506     {
507         int res = sgn(h1.vec().arg() - h2.vec().arg());
508         return res == 0 ? satisfy(h1.s, h2) : res < 0;
509     }
510     //半平面交出的凸多边形
511     friend convex halfx(vector<halfplane> &v)
512     {
513         sort(v.begin(), v.end());
514         deque<halfplane> q;
515         deque<point> ans;
516         q.push_back(v[0]);
517         for (int i = 1; i < v.size(); i++)
518         {
519             if (sgn(v[i].vec().arg() - v[i - 1].vec().arg()) == 0)
520                 continue;
521             while(ans.size() > 0 && !satisfy(ans.back(), v[i]))
522             {
523                 ans.pop_back();
524                 q.pop_back();
525             }
526             while(ans.size() > 0 && !satisfy(ans.front(), v[i]))
527             {
528                 ans.pop_front();
529                 q.pop_front();
530             }
531             ans.push_back(linexline(q.back(), v[i]));
```

```
532             q.push_back(v[i]);
533         }
534         while(ans.size() > 0 && !satisfy(ans.back(), q.front()))
535         {
536             ans.pop_back();
537             q.pop_back();
538         }
539         while(ans.size() > 0 && !satisfy(ans.front(), q.back()))
540         {
541             ans.pop_front();
542             q.pop_front();
543         }
544         ans.push_back(linexline(q.back(), q.front()));
545         convex c(ans.size());
546         int i = 0;
547         for (deque<point>::iterator it = ans.begin(); it != ans.end(); it++, i
                ++)
548             c.p[i] = *it;
549         return c;
550     }
551 };
552 //多边形的核，逆时针
553 convex core(const polygon &a)
554 {
555     convex res;
556     res.p.push_back(point(-inf, -inf));
557     res.p.push_back(point(inf, -inf));
558     res.p.push_back(point(inf, inf));
559     res.p.push_back(point(-inf, inf));
560     res.n = 4;
561     for (int i = 0; i < a.n; i++)
562         res = halfxconvex(halfplane(a.p[i], a.p[(i + 1) % a.n]), res);
563     return res;
564 }
565 //凸多边形交出的凸多边形
566 convex convexxconvex(convex &c1, convex &c2)
567 {
568     vector<halfplane> h;
569     for (int i = 0; i < c1.p.size(); i++)
570         h.push_back(halfplane(c1.p[i], c1.p[(i + 1) % c1.p.size()]));
571     for (int i = 0; i < c2.p.size(); i++)
572         h.push_back(halfplane(c2.p[i], c2.p[(i + 1) % c2.p.size()]));
573     return halfx(h);
574 }
575 double mysqrt(double n) //防止出现sqrt(-eps)的情况
576 {
577     return sqrt(max(0.0, n));
578 }
579 struct circle
580 {
581     point o;
582     double r;
583     circle(point o = point(), double r = 0): o(o), r(r) {}
584     bool operator ==(const circle &c)
585     {
586         return o == c.o && !sgn(r - c.r);
587     }
588     double area()
589     {
590         return pi * r * r;
591     }
592     //圆与线段交 用参数方程表示直线：P=A+t*(B-A)，带入圆的方程求解t
593     friend vector<point> cirxseg(const circle &c, const line &l)
```

```
    {
        double dx = l.t.x - l.s.x, dy = l.t.y - l.s.y;
        double A = dx * dx + dy * dy;
        double B = 2 * dx * (l.s.x - c.o.x) + 2 * dy * (l.s.y - c.o.y);
        double C = sqr(l.s.x - c.o.x) + sqr(l.s.y - c.o.y) - sqr(c.r);
        double delta = B * B - 4 * A * C;
        vector<point> res;
        if(sgn(delta) >= 0) //or delta > -eps ?
        {
            //可能需要注意delta接近-eps的情况，所以使用mysqrt
            double w1 = (-B - mysqrt(delta)) / (2 * A);
            double w2 = (-B + mysqrt(delta)) / (2 * A);
            if(sgn(w1 - 1) <= 0 && sgn(w1) >= 0)
                res.push_back(l.s + w1 * (l.t - l.s));
            if(sgn(w2 - 1) <= 0 && sgn(w2) >= 0)
                res.push_back(l.s + w2 * (l.t - l.s));
        }
        return res;
    }
    //圆与直线交
    friend vector<point> cirxline(const circle &c, const line &l)
    {
        double dx = l.t.x - l.s.x, dy = l.t.y - l.s.y;
        double A = dx * dx + dy * dy;
        double B = 2 * dx * (l.s.x - c.o.x) + 2 * dy * (l.s.y - c.o.y);
        double C = sqr(l.s.x - c.o.x) + sqr(l.s.y - c.o.y) - sqr(c.r);
        double delta = B * B - 4 * A * C;
        vector<point> res;
        if(sgn(delta) >= 0) //or delta > -eps ?
        {
            double w1 = (-B - mysqrt(delta)) / (2 * A);
            double w2 = (-B + mysqrt(delta)) / (2 * A);
            res.push_back(l.s + w1 * (l.t - l.s));
            res.push_back(l.s + w2 * (l.t - l.s));
        }
        return res;
    }
    //扇形面积 a->b
    double sectorarea(const point &a, const point &b) const
    {
        double theta = atan2(a.y, a.x) - atan2(b.y, b.x);
        while (theta < 0) theta += 2 * pi;
        while (theta > 2 * pi) theta -= 2 * pi;
        theta = min(theta, 2 * pi - theta);
        return sgn(det(a, b)) * theta * r * r / 2.0; //此处交大板子有误
    }
    //与线段AB的交点计算面积 a->b
    double calcarea(const point &a, const point &b) const
    {
        vector<point> p = cirxseg(*this, line(a, b));
        bool ina = sgn((a - o).norm() - r) < 0;
        bool inb = sgn((b - o).norm() - r) < 0;
        if (ina)
        {
            if (inb) return det(a - o, b - o) / 2;
            else return det(a - o, p[0] - o) / 2 + sectorarea(p[0] - o, b - o);
        }
        else
        {
            if (inb) return det(p[0] - o, b - o) / 2 + sectorarea(a - o, p[0] -
                o);
            else
            {
```

```
656                 if (p.size() == 2)
657                     return sectorarea(a - o, p[0] - o) + sectorarea(p[1] - o, b
                           - o)
658                             + det(p[0] - o, p[1] - o) / 2;
659                 else
660                     return sectorarea(a - o, b - o);
661             }
662         }
663     }
664     //圆与多边形交，结果可以尝试+eps
665     friend double cirxpolygon(const circle &c, const polygon &a)
666     {
667         int n = a.p.size();
668         double ans = 0;
669         for(int i = 0; i < n; i++)
670         {
671             if(sgn(det(a.p[i] - c.o, a.p[next(i)] - c.o)) == 0)
672                 continue;
673             ans += c.calcarea(a.p[i], a.p[next(i)]);
674         }
675         return ans;
676     }
677     //点在圆内，不包含边界
678     bool pointin(const point &p)
679     {
680         return sgn((p - o).norm() - r) < 0;
681     }
682 };
683 //三点求圆
684 circle getcircle3(const point &p0, const point &p1, const point &p2)
685 {
686     double a1 = p1.x - p0.x, b1 = p1.y - p0.y, c1 = (a1 * a1 + b1 * b1) / 2;
687     double a2 = p2.x - p0.x, b2 = p2.y - p0.y, c2 = (a2 * a2 + b2 * b2) / 2;
688     double d = a1 * b2 - a2 * b1;
689     point o(p0.x + (c1 * b2 - c2 * b1) / d, p0.y + (a1 * c2 - a2 * c1) / d);
690     return circle(o, (o - p0).norm());
691 }
692 //直径上两点求圆
693 circle getcircle2(const point &p0, const point &p1)
694 {
695     point o((p0.x + p1.x) / 2, (p0.y + p1.y) / 2);
696     return circle(o, (o - p0).norm());
697 }
698 //最小圆覆盖 用之前可以随机化random_shuffle
699 circle mincircover(vector<point> &a)
700 {
701     int n = a.size();
702     circle c(a[0], 0);
703     for (int i = 1; i < n; i++)
704         if (!c.pointin(a[i]))
705         {
706             c.o = a[i];
707             c.r = 0;
708             for (int j = 0; j < i; j++)
709                 if (!c.pointin(a[j]))
710                 {
711                     c = getcircle2(a[i], a[j]);
712                     for (int k = 0; k < j; k++)
713                         if (!c.pointin(a[k]))
714                             c = getcircle3(a[i], a[j], a[k]);
715                 }
716         }
717     return c;
```

```
}
int main()
{
    return 0;
}
```

```
const int MAXN=100005;
// 分治算法求最近点对
struct point
{
    double x,y;
};
point p[MAXN];
int index[MAXN];  //保存筛选的坐标点的索引
int cmpx(const point &a,const point &b)
{
    return a.x<b.x;
}
int cmpy(int a,int b)  //这里用的是下标索引
{
    return p[a].y<p[b].y;
}
inline double dis(point &a , point &b)
{
    return sqrt( (a.x-b.x)*(a.x-b.x) + (a.y-b.y)*(a.y-b.y) );
}
inline double min(double a , double b)
{
    return a<b?a:b;
}
double closest(int low , int high)
{
    if(low+1==high)
        return dis(p[low],p[high]);
    if(low+2==high)
        return min( dis(p[low],p[high]) , min(dis(p[low],p[low+1]),dis(p[low+1],
            p[high])) );
    int mid = (low + high)>>1;//求中点
    double ans = min( closest(low,mid),closest(mid+1,high) );  //
        分治法进行递归求解
    int cnt = 0;
    for(int i=low; i<=high; i++) //把x坐标在p[mid].x-ans~p[mid].x+ans
        范围内的点取出来
    {
        if(p[i].x >= p[mid].x - ans && p[i].x <= p[mid].x + ans)
            index[cnt++]=i;     //保存的是下标索引
    }
    sort(index,index+cnt,cmpy);   //按y坐标进行升序排序
    for(int i=0; i<cnt; i++)
        for(int j=i+1; j<cnt; j++)
        {
            if(p[index[j]].y - p[index[i]].y >= ans)   //注意下标索引
                break;
            ans = min( ans,dis(p[index[i]],p[index[j]]) );
        }
    return ans;
}
void init(int n)
{
    for (int i=0; i<n; i++)
        scanf("%lf%lf",&p[i].x,&p[i].y);
    sort(p,p+n,cmpx);
    return;
```

```
}
int main()
{
    int n;
    while(cin>>n,n!=0)
    {
        init(n);
        printf("%.2lf\n",closest(0,n-1)/2);
    }
    return 0;
}
```

12 其他

12.1 编译器相关

12.1.1 强制 O2 优化

```
#pragma GCC optimize(2)
```

12.1.2 G++(MinGW32) 扩栈

```
int SIZE_OF_STACK = 256 << 20; // 256MB
char *p = (char*)malloc(SIZE_OF_STACK) + SIZE_OF_STACK;
__asm__("movl %0, %%esp\n" :: "r"(p));
```

12.1.3 G++(64 位 linux) 扩栈

```
///只在会爆栈的函数前后使用汇编语句会更安全
///但要注意避免使用局部变量保存函数返回值
long rsp;
int main()
{
    int SIZE_OF_STACK = 256 << 20; // 256MB
    char *p = (char*)malloc(SIZE_OF_STACK) + SIZE_OF_STACK;
    __asm__("movq %%rsp, %0" :: "m"(rsp));
    __asm__("movq %0, %%rsp" :: "r"(p));

    //程序体

    __asm__("movq %0, %%rsp\n" :: "m"(rsp));
    return 0;
}
```

12.1.4 C++ 扩栈

```
#pragma comment(linker, "/STACK:102400000,102400000")
```

12.2 输入输出优化

```
inline void RD(int &ret)
{
    char c;
    do c=getchar();
    while(c<'0'||c>'9');
```

```
    ret=c-'0';
    while((c=getchar())>='0'&&c<='9')
        ret=ret*10+(c-'0');
}
inline void OT(int a)
{
    if(a>=10) OT(a/10);
    putchar(a%10+'0');
}
//负数
inline void RD(int &ret)
{
    char c;
    int sgn;
    while(c!='-'&&(c<'0'||c>'9')) c=getchar();
    sgn=(c=='-')?-1:1;
    ret=(c=='-')?0:(c-'0');
    while(c=getchar(),c>='0'&&c<='9') ret=ret*10+(c-'0');
    ret*=sgn;
}
```

12.3 位反转

```
inline int reversebits(int x)
{
    x = ((x >> 1) & 0x55555555) | ((x << 1) & 0xaaaaaaaa);
    x = ((x >> 2) & 0x33333333) | ((x << 2) & 0xcccccccc);
    x = ((x >> 4) & 0x0f0f0f0f) | ((x << 4) & 0xf0f0f0f0);
    x = ((x >> 8) & 0x00ff00ff) | ((x << 8) & 0xff00ff00);
    x = ((x >>16) & 0x0000ffff) | ((x <<16) & 0xffff0000);
    return x;
}

inline long long reversebits(long long x)
{
    x = ((x >> 1) & 0x5555555555555555LL) | ((x << 1) & 0xaaaaaaaaaaaaaaaaLL);
    x = ((x >> 2) & 0x3333333333333333LL) | ((x << 2) & 0xccccccccccccccccLL);
    x = ((x >> 4) & 0x0f0f0f0f0f0f0f0fLL) | ((x << 4) & 0xf0f0f0f0f0f0f0f0LL);
    x = ((x >> 8) & 0x00ff00ff00ff00ffLL) | ((x << 8) & 0xff00ff00ff00ff00LL);
    x = ((x >>16) & 0x0000ffff0000ffffLL) | ((x <<16) & 0xffff0000ffff0000LL);
    x = ((x >>32) & 0x00000000ffffffffLL) | ((x <<32) & 0xffffffff00000000LL);
    return x;
}
```

12.4 蔡勒公式

$w = \left(y + \left[\frac{y}{4}\right] + \left[\frac{c}{4}\right] - 2c + \left[\frac{26(m+1)}{10}\right] + d - 1\right) \mod 7$

若要计算的日期是在 1582 年 10 月 4 日或之前，公式则为：

$w = \left(y + \left[\frac{y}{4}\right] + \left[\frac{c}{4}\right] - 2c + \left[\frac{26(m+1)}{10}\right] + d + 2\right) \mod 7$

公式中的符号含义如下：

w：星期（计算所得的数值对应的星期：0-星期日；1-星期一；2-星期二；3-星期三；4-星期四；5-星期五；6-星期六）

c：年份前两位数

y：年份后两位数

m：月（m 的取值范围为 3 至 14，即在蔡勒公式中，某年的 1、2 月要看作上一年的 13、14 月来计算，比如 2003 年 1 月 1 日要看作 2002 年的 13 月 1 日来计算）

d：日

（因罗马教宗额我略十三世颁布新历法 (公历)，把 1582 年 10 月 4 日的后一天改为 1582 年 10 月 15 日）

```
int zeller(int y,int m,int d,int flag)
{
    if (m==1||m==2)
        y--,m+=12;
    int c=y/100;
    y%=100;
    int w=((c/4-2*c+y+y/4+13*(m+1)/5+d-1)%7+7)%7;
    if (flag) return (w+3)%7;
    return w;
}
```

12.5 坐标旋转变换

要注意在处理多个点的时候，n 和 m 要存副本。

```
void rotate(int &x,int &y,int &n,int &m,int t)
{
    t%=4;
    while(t--)
    {
        swap(x,y);
        y=n-y+1;
        swap(n,m);
    }
}

void rotate(int &x,int &y,int &n,int &m,int t)
{
    int a,b;
    if (t==1)
        a=y,b=n-x+1;
    if (t==2)
        a=n-x+1,b=m-y+1;
    if (t==3)
        a=m-y+1,b=x;
    if (t) x=a,y=b;
    if (t&&t!=2) swap(n,m);
}
```

12.6 归并排序求逆序数

```
int a[500005],b[500005];
long long mergesort(int left,int right)
{
    if (left==right)
        return 0;
    long long cnt=0;
    int mid=(left+right)/2;
    cnt=mergesort(left,mid)+mergesort(mid+1,right);
    int i=left,j=mid+1,k=left;
    while(i<=mid&&j<=right)
        if(a[i]<=a[j])
            b[k++]=a[i++];
        else
        {
            b[k++]=a[j++];
            cnt+=(mid-i+1);
```

```
        }
    while(i<=mid)
        b[k++]=a[i++];
    while(j<=right)
        b[k++]=a[j++];
    for(i=left; i<=right; i++)
        a[i]=b[i];
    return cnt;
}
```

12.7 奇怪的东西

格式化 html 标签

```
char str[500000];
inline void jump()
{
    char c;
    while(isspace(c=getchar())) ;
    ungetc(c,stdin);
}
inline string gtag()
{
    jump();
    str[0]=0;
    scanf("%[^>]",str);
    string s=str;
    s+=getchar();
    return s;
}
inline void ptag(string &s)
{
    puts(s.c_str());
}
inline string gstr()
{
    jump();
    str[0]=0;
    scanf("%[^ <\n]",str);
    return str;
}
inline void pstr(string &s)
{
    printf("%s",s.c_str());
}
inline int judge(string &s)
{
    if (s[1]=='/') return -1;
    int flag=0;
    for (int i=0; i<s.length(); i++)
    {
        if (s[i]=='\"') flag=!flag;
        if (flag==0&&s[i]=='/') return 0;
    }
    return 1;
}
inline void pspace(int x)
{
    for (int i=0; i<x; i++)
        putchar(' ');
}
inline bool ishtml(string &s)
```

```
{
    return s=="</html>";
}
void dostr(int k)
{
    string s=gstr();
    if (s!="")
    {
        pspace(k);
        pstr(s);
        while(1)
        {
            s=gstr();
            if (s=="") break;
            putchar(' ');
            pstr(s);
        }
        puts("");
    }
}
void solve()
{
    int k=0,x;
    while(1)
    {
        string s=gtag();
        x=judge(s);
        if (x==-1) pspace(k-1);
        else pspace(k);
        k+=x;
        if (ishtml(s))
        {
            ptag(s);
            break;
        }
        ptag(s);
        dostr(k);
    }
}
```

输出数字的英文表示

```
char a[50][15]= {"","one","two","three","four","five","six","seven","eight","
    nine","ten","eleven","twelve","thirteen","fourteen","fifteen","sixteen","
    seventeen","eighteen","nineteen","twenty"};
char b[50][15]= {"","ten","twenty","thirty","forty","fifty","sixty","seventy","
    eighty","ninety"};
char c[50][15]= {"","thousand","million","billion"};
void gao(int n)
{
    int t=n/100,t1=n/10%10,t2=n%10,t3=n%100;
    if (t) printf("%s hundred",a[t]);
    if (t3)
    {
        if (t) printf(" and ");
        if (t3<=20) printf("%s",a[t3]);
        else
        {
            printf("%s",b[t3/10]);
            if (t2) printf("-%s",a[t2]);
        }
    }
}
int main()
```

```
{
    int n;
    while(scanf("%d",&n),n!=-1)
    {
        if (n==0)
        {
            puts("zero");
            continue;
        }
        int q[5];
        int x=-1;
        while(n) q[++x]=n%1000,n/=1000;
        for (int i=x; i>=0; i--)
            if (q[i])
            {
                gao(q[i]);
                if (i) printf(" %s",c[i]);
                int flag=0;
                for (int j=i-1; j>=0; j--)
                    if (q[j]) flag=1;
                if (flag) printf(", ");
            }
        puts("");
    }
    return 0;
}
```

13 一些题目

Description

M 公司是一个非常庞大的跨国公司，在许多国家都设有它的下属分支机构或部门。为了让分布在世界各地的 N 个部门之间协同工作，公司搭建了一个连接整个公司的通信网络。该网络的结构由 N 个路由器和 N-1 条高速光缆组成。每个部门都有一个专属的路由器，部门局域网内的所有机器都联向这个路由器，然后再通过这个通信子网与其他部门进行通信联络。该网络结构保证网络中的任意两个路由器之间都存在一条直接或间接路径以进行通信。高速光缆的数据传输速度非常快，以至于利用光缆传输的延迟时间可以忽略。但是由于路由器老化，在这些路由器上进行数据交换会带来很大的延迟。而两个路由器之间的通信延迟时间则与这两个路由器通信路径上所有路由器中最大的交换延迟时间有关。作为 M 公司网络部门的一名实习员工，现在要求你编写一个简单的程序来监视公司的网络状况。该程序能够随时更新网络状况的变化信息（路由器数据交换延迟时间的变化），并且根据询问给出两个路由器通信路径上延迟第 k 大的路由器的延迟时间。【任务】你的程序从输入文件中读入 N 个路由器和 N-1 条光缆的连接信息，每个路由器初始的数据交换延迟时间 Ti，以及 Q 条询问（或状态改变）的信息。并依次处理这 Q 条询问信息，它们可能是：1. 由于更新了设备，或者设备出现新的故障，使得某个路由器的数据交换延迟时间发生了变化。2. 查询某两个路由器 a 和 b 之间的路径上延迟第 k 大的路由器的延迟时间。

Input

第一行为两个整数 N 和 Q，分别表示路由器总数和询问的总数。第二行有 N 个整数，第 i 个数表示编号为 i 的路由器初始的数据延迟时间 Ti。紧接着 N-1 行，每行包含两个整数 x 和 y。表示有一条光缆连接路由器 x 和路由器 y。紧接着是 Q 行，每行三个整数 k、a、b。如果 k=0，则表示路由器 a 的状态发生了变化，它的数据交换延迟时间由 Ta 变为 b。如果 k>0，则表示询问 a 到 b 的路径上所经过的所有路由器（包括 a 和 b）中延迟第 k 大的路由器的延迟时间。注意 a 可以等于 b，此时路径上只有一个路由器。

Output

对于每一个第二种询问（k>0），输出一行。包含一个整数为相应的延迟时间。如果路径上的

路由器不足 k 个，则输出信息“invalid request!”（全部小写不包含引号，两个单词之间有一个空格）。

Sample Input
5 5
5 1 2 3 4
3 1
2 1
4 3
5 3
2 4 5
0 1 2
2 2 3
2 1 4
3 3 5
Sample Output
3
2
2
invalid request!
Hint
10
40
100

一样对树进行划分，对于一条链我们需要支持的操作就是修改一个数，以及查询区间第 k 大。这是一个经典问题，可以用树套树。具体的说就是链用一个线段树维护，对于线段树中的每个节点，维护当前区间节点的一个平衡树，这样首先内存是 NlogN 的，然后对于修改，最多涉及到 O(logN) 个区间，所以复杂度是 O(log2N)。对于查询，和普通的区间第 k 大问题一样，首先二分答案，然后将问题转化成一个判定性问题，从而在区间中查找比这个二分答案小的有几个，复杂度就是 O(log3N)。因为一个点到根最多跨越 logN 条，所以总复杂度就是 O(N+Qlog4N)。

```
const int maxn=100010;
const int maxnode=8000010;

int lc[maxnode],rc[maxnode],num[maxnode],cnt;

int ins(int old,int pos,int l,int r,int k)
{
    int mid;
    int n=++cnt,t=n;
    num[n]=num[old]+k;
    lc[n]=lc[old],rc[n]=rc[old];

    while(l<r)
    {
        mid=(l+r)/2;
        if(pos<=mid)
            lc[n]=++cnt,n=lc[n],old=lc[old],r=mid;
        else
            rc[n]=++cnt,n=rc[n],old=rc[old],l=mid+1;
        num[n]=num[old]+k,lc[n]=lc[old],rc[n]=rc[old];
    }
    return t;
}
int build(int l,int r)
{
```

```
    int mid=(l+r)/2,n=++cnt;
    num[n]=0;

    if(l==r)
        return n;
    lc[n]=build(l,mid);
    rc[n]=build(mid+1,r);
    return n;
}

int get(int l,int r,int *a,int numa,int *b,int numb,int k)
{
    int t=0,i,tt=0;
    for(i=0; i<numa; i++)
        t+=num[rc[a[i]]],tt+=num[a[i]];
    for(i=0; i<numb; i++)
        t-=num[rc[b[i]]],tt-=num[b[i]];
    if(tt<k)
        return -1;
    if(l==r)
        return l;
    if(t>=k)
    {
        for(i=0; i<numa; i++)
            a[i]=rc[a[i]];
        for(i=0; i<numb; i++)
            b[i]=rc[b[i]];
        return get((l+r)/2+1,r,a,numa,b,numb,k);
    }
    for(i=0; i<numa; i++)
        a[i]=lc[a[i]];
    for(i=0; i<numb; i++)
        b[i]=lc[b[i]];
    return get(l,(l+r)/2,a,numa,b,numb,k-t);
}
int root[maxn],c[maxn];

int a[1100],numa,b[1100],numb;
int lft[maxn],rht[maxn];
void gao(int n,int is)
{
    if(is)
    {
        a[numa++]=root[n];
        n=lft[n];
        while(n)
            a[numa++]=c[n],n-=n&(-n);
    }
    else
    {
        b[numb++]=root[n];
        n=lft[n];
        while(n)
            b[numb++]=c[n],n-=n&(-n);
    }
}

vector<pair<int,int> >V[maxn];
int lca[maxn],fa[maxn],w[maxn],first[maxn],nxt[maxn<<1],vv[maxn<<1],M,tim;
int f[maxn],vis[maxn];

int fin(int n)
{
```

```
89      if(n==fa[n])
90          return n;
91      return fa[n]=fin(fa[n]);
92  }
93  void dfs(int n)
94  {
95      root[n]=ins(root[f[n]],w[n],1,M,1);
96      lft[n]=++tim;
97      fa[n]=n;
98
99      for(int e=first[n]; e; e=nxt[e])if(vv[e]-f[n])
100             f[vv[e]]=n,dfs(vv[e]),fa[vv[e]]=n;
101     vis[n]=1;
102     rht[n]=tim;
103
104     for(int i=0; i<V[n].size(); i++)if(vis[V[n][i].first])
105         {
106             int v=V[n][i].first;
107             lca[V[n][i].second]=fin(v);
108         }
109 }
110 int A[maxn],B[maxn],C[maxn],wa[maxn<<1];
111 int N;
112
113 void upd(int n,int pos,int k)
114 {
115     while(n<=N)
116         c[n]=ins(c[n],pos,1,M,k),n+=n&(-n);
117 }
118 int main()
119 {
120     int n,q,i,j;
121
122     cin>>n>>q;
123     for(i=1; i<=n; i++)
124         scanf("%d",&w[i]),wa[i]=w[i];
125     N=n;
126     M=n;
127     int e=2;
128
129     for(i=1; i<n; i++)
130     {
131         int u,v;
132         scanf("%d%d",&u,&v);
133         nxt[e]=first[u],vv[e]=v,first[u]=e++;
134         nxt[e]=first[v],vv[e]=u,first[v]=e++;
135     }
136     for(i=1; i<=q; i++)
137     {
138         scanf("%d%d%d",&A[i],&B[i],&C[i]);
139         if(A[i])
140             V[B[i]].push_back(make_pair(C[i],i)),V[C[i]].push_back(make_pair(B[i
                    ],i));
141         else
142             wa[++M]=C[i];
143     }
144     sort(wa+1,wa+1+M);
145     M=unique(wa+1,wa+1+M)-wa-1;
146     for(i=1; i<=n; i++)
147         w[i]=lower_bound(wa+1,wa+1+M,w[i])-wa;
148     root[0]=build(1,M);
149     dfs((n+1)/2);
150     for(i=1; i<=N; i++)
```

```
        c[i]=root[0];

    for(i=1; i<=q; i++)
    {
        if(A[i])
        {
            int u=B[i],v=C[i],l=lca[i];
            numa=numb=0;
            gao(u,1),gao(v,1),gao(l,0),gao(f[l],0);
            u=get(1,M,a,numa,b,numb,A[i]);
            if(u<0)
                puts("invalid request!");
            else
                printf("%d\n",wa[u]);
        }
        else
        {
            int u=B[i],v=C[i];
            v=lower_bound(wa+1,wa+1+M,v)-wa;
            upd(lft[u],w[u],-1);
            upd(rht[u]+1,w[u],1);
            upd(lft[u],v,1);
            upd(rht[u]+1,v,-1);
            w[u]=v;
        }
    }
}
```

```
inline int readint()
{
    char c = getchar();
    while (!isdigit(c)) c = getchar();
    int x = 0;
    while (isdigit(c))
    {
        x = x * 10 + c - '0';
        c = getchar();
    }
    return x;
}

inline long long readlong()
{
    char c = getchar();
    while (!isdigit(c)) c = getchar();
    long long x = 0;
    while (isdigit(c))
    {
        x = x * 10 + c - '0';
        c = getchar();
    }
    return x;
}

#define FOR(i, n) for (int i = 0; i < (int)(n); i++)
#define REP(i, a, b) for (int i = (int)(a); i <= (int)(b); i++)
#define CIR(i, a, b) for (int i = (int)(b); i >= (int)(a); i--)
#define PII pair<int, int>
#define FI first
#define SE second
#define MP make_pair
#define PB push_back
#define SZ(v) v.size();
```

```
#define ALL(v) v.begin(), v.end()
#define CLR(v, a) memset(v, a, sizeof(v));
#define IT iterator
#define LL long long
#define DB double
#define PI 3.1415926
#define INF 1000000000

#define N 80005
#define M N * 60

int dep[N], W[N], top[N], idx[N], size[N], fa[N], id;
int son[N], head[N], to[N << 1], next[N << 1];
int n, E, it;
int test;

void init()
{
    for (int i = 1; i <= n; i++)
    {
        head[i] = -1, son[i] = 0, dep[i] = 0;
    }
    id = 0;
    E = 0, it = 1;
}

void addEdge(int u, int v)
{
    to[E] = v, next[E] = head[u], head[u] = E++;
    to[E] = u, next[E] = head[v], head[v] = E++;
}

void dfs(int u, int p)
{
    fa[u] = p;
    size[u] = 1, son[u] = 0;

    for (int i = head[u]; i != -1; i = next[i])
    {
        int v = to[i];

        if (v != p)
        {
            dep[v] = dep[u] + 1;
            dfs(v, u);
            if (size[v] > size[son[u]]) son[u] = v;
            size[u] += size[v];
        }
    }
}

void dfs(int u, int p, int tu)
{

    idx[u] = ++id, top[u] = tu;
    if (son[u]) dfs(son[u], u, tu);
    for (int i = head[u]; i != -1; i = next[i])
    {
        int v = to[i];
        if (v == p || son[u] == v) continue;
        dfs(v, u, v);
    }
}
```

```
 99
100 int lca(int u, int v)
101 {
102     int tu = top[u], tv = top[v];
103     while (tu != tv)
104     {
105         if (dep[tu] < dep[tv])
106         {
107             swap(tu, tv);
108             swap(u, v);
109         }
110         u = fa[tu], tu = top[u];
111     }
112
113     if (dep[u] > dep[v]) swap(u, v);
114     return u;
115 }
116
117 int root[N], r[M], v[M], ch[M][2], sz[M];
118
119 #define lch ch[rt][0]
120 #define rch ch[rt][1]
121
122 inline void push_up(int rt)
123 {
124     sz[rt] = sz[lch] + sz[rch] + 1;
125 }
126
127 inline int cmp(int rt, int x)
128 {
129     if (v[rt] == x) return -1;
130     else if (x < v[rt])
131         return 0;
132     else
133         return 1;
134 }
135
136 void rotate(int& rt, int d)
137 {
138     int t = ch[rt][d ^ 1];
139     ch[rt][d ^ 1] = ch[t][d];
140     ch[t][d] = rt;
141     push_up(rt);
142     push_up(t);
143     rt = t;
144 }
145
146 void insert(int&rt, int x)
147 {
148     if (!rt)
149     {
150         rt = it++;
151         sz[rt] = 1;
152         lch = rch = 0;
153         v[rt] = x;
154         r[rt] = rand();
155     }
156     else
157     {
158         int d = x < v[rt] ? 0 : 1;
159         insert(ch[rt][d], x);
160         if (r[ch[rt][d]] > r[rt])
161             rotate(rt, d ^ 1);
```

```
162     }
163     push_up(rt);
164 }
165 
166 int kth(int rt, int k)
167 {
168     int  t = sz[rch] + 1;
169     if (k == t)
170         return v[rt];
171     else if (k < t)
172         return kth(rch, k);
173     else
174         return kth(lch, k - t);
175 }
176 
177 int upper(int rt, int val)
178 {
179     if (!rt) return 0;
180     int res = 0;
181     if (v[rt] == val)
182     {
183         res++;
184         res += upper(lch, val);
185         res += upper(rch, val);
186     }
187     else if (val < v[rt])
188     {
189         res += sz[rch] + 1;
190         res += upper(lch, val);
191     }
192     else
193     {
194         res += upper(rch, val);
195     }
196     return res;
197 }
198 
199 void remove(int& rt, int x)
200 {
201     int d = cmp(rt, x);
202     if (d == -1)
203     {
204         if (lch && rch)
205         {
206             int d2 = r[lch] > r[rch] ? 0 : 1;
207             rotate(rt, d2 ^ 1);
208             remove(ch[rt][d2 ^ 1], x);
209             push_up(rt);
210         }
211         else
212         {
213             rt = lch ? lch : rch;
214         }
215     }
216     else
217     {
218         remove(ch[rt][d], x);
219         push_up(rt);
220     }
221 }
222 
223 int gao(int L, int R, int val)
224 {
```

```
225     int res = 0;
226     int p = R;
227     while (p)
228     {
229         res += upper(root[p], val);
230         p -= p & (-p);
231     }
232 
233     p = L - 1;
234 
235     while (p)
236     {
237         res -= upper(root[p], val);
238         p -= p & (-p);
239     }
240 
241     return res;
242 }
243 
244 int solve(int u, int v, int val)
245 {
246     int res = 0;
247     int tu = top[u], tv = top[v];
248     while (tu != tv)
249     {
250         if (dep[tu] < dep[tv])
251         {
252             swap(tu, tv);
253             swap(u, v);
254         }
255         res += gao(idx[tu], idx[u], val);
256         u = fa[tu], tu = top[u];
257     }
258 
259     if (dep[u] > dep[v]) swap(u, v);
260     res += gao(idx[u], idx[v], val);
261 
262     return res;
263 }
264 
265 void modify(int u, int val)
266 {
267     if (W[u] == val) return;
268 
269     int p = idx[u];
270     while (p <= n)
271     {
272         remove(root[p], W[u]);
273         insert(root[p], val);
274         p += p & (-p);
275     }
276 
277     remove(root[N - 1], W[u]);
278     insert(root[N - 1], val);
279     W[u] = val;
280 }
281 
282 int query(int u, int v, int k)
283 {
284     int pa = lca(u, v);
285     int temp = dep[u] + dep[v] - 2 * dep[pa] + 1;
286     if (k > temp) return -1;
287     int L = 1, R = n;
```

```
288
289     while (L <= R)
290     {
291         int mid = L + R >> 1;
292         int tmp = kth(root[N - 1], mid);
293
294         if (solve(u, v, tmp) >= k)
295             R = mid - 1;
296         else
297             L = mid + 1;
298     }
299
300     return kth(root[N - 1], L);
301 }
302
303 int main()
304 {
305     int k, a, b, m;
306     while (~scanf("%d%d", &n, &m))
307     {
308         REP(i, 1, n) scanf("%d", &W[i]);
309
310         init();
311
312         FOR (i, N) root[i] = 0;
313
314         FOR (i, n - 1)
315         {
316             scanf("%d%d", &a, &b);
317             addEdge(a, b);
318         }
319
320         dfs(1, -1);
321         dfs(1, -1, 1);
322
323         REP(i, 1, n)
324         {
325             int p = idx[i];
326             insert(root[N - 1], W[i]);
327             while (p <= n)
328             {
329                 insert(root[p], W[i]);
330                 p += p & (-p);
331             }
332         }
333
334         FOR (i, m)
335         {
336             scanf("%d%d%d", &k, &a, &b);
337             if (k == 0)
338                 modify(a, b);
339             else
340             {
341                 int res = query(a, b, k);
342                 if (res == -1)
343                     puts("invalid request!");
344                 else
345                     printf("%d\n", res);
346             }
347
348         }
349     }
350     return 0;
```

```
351 }
```

SAT 问题就是说我现在有一个 n 个布尔值组成的序列 a，每个元素可取 0 或 1，现有一些限制条件，例如 a[1] or a[5]，a[5] xor not a[8]，a[3] and a[4] or a[5]，求 a 的一个取值序列可以使这些式子都为真。即求使这些式子的合取式为真的一组解。

而 2-SAT 问题就是所有限制条件中最多有两个元素，在上面的那个例子中 a[3] and a[4] or a[5] 有 3 个元素，所以这个例子是 3-SAT 问题不是 2-SAT。3-SAT 及以上是 NPC 问题，目前还没有多项式级别的算法。

2-SAT 问题关键在于建图。若图中存在边 <i, j>，则表示若选了 i 必须选 j。

如果我们必须选 x，则只需连边 <x', x > 即可。因为这样的话如果我们选了 x' 即不选 x，那么我们就必须选 x，这就导致了矛盾。所以我们只能选 x。

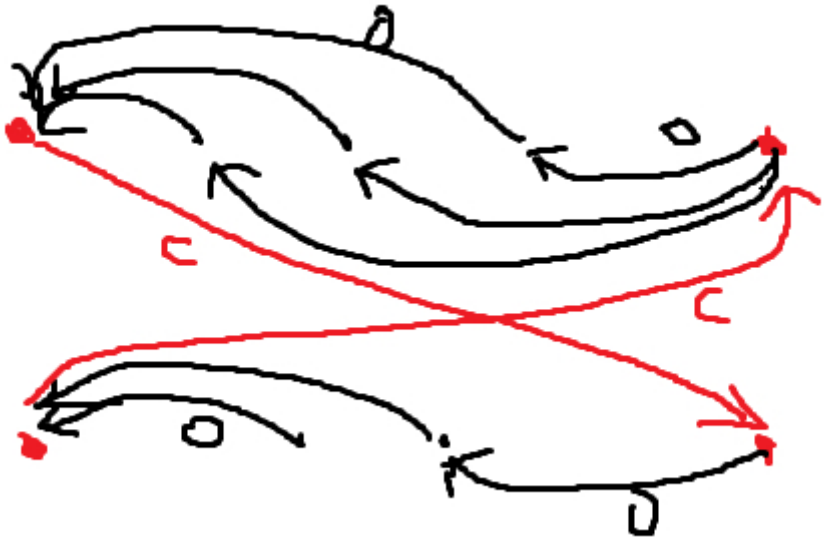
有一对新人结婚，邀请 n 对夫妇去参加婚礼。有一张很长的桌子，人只能坐在桌子的两边，还要满足下面的要求：1. 每对夫妇不能坐在同一侧 2.n 对夫妇之中可能有通奸关系（包括男男，男女，女女），有通奸关系的不能同时坐在新娘的对面，可以分开坐，也可以同时坐在新娘这一侧。如果存在一种可行的方案，输出与新娘同侧的人。

所以分成 4 种情况来建图。1-n 的点为 h(usband)，n+1 到 2n 的点为 w(ife)。

```
for (int i=0; i<m; i++)
{
    ///TODO: adde
    int x,y;
    char c1,c2;
    scanf("%d%c%d%c",&x,&c1,&y,&c2);
    x++,y++;
    if (c1=='h'&&c2=='h')
    {
        adde(x+n,y);
        adde(y+n,x);
    }
    else if (c1=='h'&&c2=='w')
    {
        adde(x+n,y+n);
        adde(y,x);
    }
    else if (c1=='w'&&c2=='h')
    {
        adde(x,y);
        adde(y+n,x+n);
    }
    else
    {
        adde(x,y+n);
        adde(y,x+n);
    }
}
```

题意：有 n 层，有 n 个点分布在这 n 层上（一层可以有多个点，也可以有 0 个）。层与层之间的点可以互相走，权为 c。另外还有 m 条无向边，从某个点到某个点，权为 w。求 1 到 n 的最短路。
此题的关键在于建图。如果采用朴素建图法，相邻两层的点两两加边，复杂度高达 N2。如果用下面的这个方法，可以将复杂度降到 N。
每层虚拟出来两个点，入点和出点。像图中所示连接（省略了 m 条给定的边）。黑线的权都为 0，红线权为 c。这样借助两个点，可以实现层与层连接。
一开始想只用一个点，但是这样的话同一层的点就可以通过权为 0 的路连接起来了。

（后来发现图中最下面少画了一条边……）



```
typedef int mytype;
const int NV=300005;
const int NE=300005*2;
mytype dis[NV];
int pre[NV],vis[NV],he[NV],ecnt;
struct edge
{
    int v,next;
    mytype l;
} E[NE];
void adde(int u,int v,mytype l)
{
    ecnt++;
    E[ecnt].v=v;
    E[ecnt].l=l;
    E[ecnt].next=he[u];
    he[u]=ecnt;
}
void init(int n,int m,int s)
{
    for (int i=0; i<=n; i++)
        dis[i]=inf;
    dis[s]=0;
    for (int i=1; i<=m; i++)
    {
        int u,v;
        mytype l;
        scanf("%d%d%d",&u,&v,&l);
        adde(u,v,l);
        adde(v,u,l);
    }
}
struct point
{
    int u;
    mytype l;
    point(int a,mytype b):u(a),l(b) {}
    bool operator<(const point p) const
    {
        return l>p.l;
```

```
    }
};
void dijkstra_heap(int s)
{
    priority_queue<point> q;
    q.push(point(s,0));
    while(!q.empty())
    {
        point p=q.top();
        q.pop();
        int u=p.u;
        if (vis[u])
            continue;
        vis[u]=1;
        for (int i=he[u]; i!=-1; i=E[i].next)
            if (!vis[E[i].v]&&p.l+E[i].l<dis[E[i].v])
            {
                dis[E[i].v]=dis[u]+E[i].l;
                pre[E[i].v]=u;
                q.push(point(E[i].v,dis[E[i].v]));
            }
    }
}
int main()
{
    int t;
    cin>>t;
    int cas=0;
    while(t--)
    {
        int n,m,c;
        scanf("%d%d%d",&n,&m,&c);
        ecnt=0;
        memset(pre,0,sizeof(pre));
        memset(vis,0,sizeof(vis));
        memset(he,-1,sizeof(he));
        for (int i=1; i<=n; i++)
        {
            int x;
            scanf("%d",&x);
            adde(i,n+x,0);
            adde(2*n+x,i,0);
        }
        for (int i=1; i<n; i++)
        {
            adde(n+i,2*n+i+1,c);
            adde(n+i+1,2*n+i,c);
        }
        init(n*3,m,1);
        dijkstra_heap(1);
        int ans=dis[n];
        if (ans==inf)
            ans=-1;
        printf("Case #%d: %d\n",++cas,ans);
    }
    return 0;
}
```

一个次短路问题。题目是要求最短路和比最短路距离长 1 的次短路的个数。

```
typedef int mytype;
const int NV=1005;
const int NE=10005;
mytype dis[NV][2];
```

```
5  int vis[NV][2],he[NV],ecnt,cnt[NV][2];
6  struct edge
7  {
8      int v,next;
9      mytype l;
10 } e[NE];
11 void adde(int u,int v,mytype l)
12 {
13     ecnt++;
14     e[ecnt].v=v;
15     e[ecnt].l=l;
16     e[ecnt].next=he[u];
17     he[u]=ecnt;
18 }
19 struct point
20 {
21     int u,flag;
22     mytype l;
23     point(int a,mytype b,int c):u(a),l(b),flag(c) {}
24     bool operator<(const point p) const
25     {
26         return l>p.l;
27     }
28 };
29 int dijkstra_heap(int n,int m,int s,int E)
30 {
31     priority_queue<point> q;
32     q.push(point(s,0,0));
33     while(!q.empty())
34     {
35         point p=q.top();
36         q.pop();
37         int u=p.u;
38         if (vis[u][p.flag])
39             continue;
40         vis[u][p.flag]=1;
41         for (int i=he[u]; i!=-1; i=e[i].next)
42         {
43             int v=e[i].v;
44             int dist=p.l+e[i].l;
45             if (dist<dis[v][0])
46             {
47                 if (dis[v][0]!=inf)
48                 {
49                     dis[v][1]=dis[v][0];
50                     cnt[v][1]=cnt[v][0];
51                     q.push(point(v,dis[v][1],1));
52                 }
53                 dis[v][0]=dist;
54                 cnt[v][0]=cnt[u][p.flag];
55                 q.push(point(v,dis[v][0],0));
56             }
57             else if (dist==dis[v][0])
58             {
59                 cnt[v][0]+=cnt[u][p.flag];
60             }
61             else if (dist<dis[v][1])
62             {
63                 dis[v][1]=dist;
64                 cnt[v][1]=cnt[u][p.flag];
65                 q.push(point(v,dis[v][1],1));
66             }
67             else if (dist==dis[v][1])
```

```
68              {
69                  cnt[v][1]+=cnt[u][p.flag];
70              }
71          }
72      }
73      if (dis[E][0]+1==dis[E][1])
74          return cnt[E][0]+cnt[E][1];
75      return cnt[E][0];
76  }
77  int main()
78  {
79      int t;
80      cin>>t;
81      while(t--)
82      {
83          int n,m;
84          scanf("%d%d",&n,&m);
85          ecnt=0;
86          memset(he,-1,sizeof(he));
87          memset(vis,0,sizeof(vis));
88          memset(cnt,0,sizeof(cnt));
89          for (int i=1; i<=m; i++)
90          {
91              int u,v;
92              mytype l;
93              scanf("%d%d%d",&u,&v,&l);
94              adde(u,v,l);
95          }
96          int s,E;
97          scanf("%d%d",&s,&E);
98          for (int i=1; i<=n; i++)
99              dis[i][0]=dis[i][1]=inf;
100         dis[s][0]=0;
101         cnt[s][0]=1;
102         printf("%d\n",dijkstra_heap(n,m,s,E));
103     }
104     return 0;
105 }
```

请写一个程序，要求维护一个数列，支持以下 6 种操作：**（请注意，格式栏中的下划线'_'表示实际输入文件中的空格）**

操作编号	输入文件中的格式	说明
1. 插入	INSERT_*posi*_*tot*_c_1_c_2_…_c_{tot}	在当前数列的第 *posi* 个数字后插入 *tot* 个数字：$c_1, c_2, \ldots, c_{tot}$；若在数列首插入，则 *posi* 为 0
2. 删除	DELETE_*posi*_*tot*	从当前数列的第 *posi* 个数字开始连续删除 *tot* 个数字
3. 修改	MAKE-SAME_*posi*_*tot*_*c*	将当前数列的第 *posi* 个数字开始的连续 *tot* 个数字统一修改为 *c*
4. 翻转	REVERSE_*posi*_*tot*	取出从当前数列的第 *posi* 个数字开始的 *tot* 个数字，翻转后放入原来的位置
5. 求和	GET-SUM_*posi*_*tot*	计算从当前数列开始的第 *posi* 个数字开始的 *tot* 个数字的和并输出
6. 求和最大的子列	MAX-SUM	求出当前数列中和最大的一段子列，并输出最大和

```
#define Key_value ch[ch[root][1]][0]
const int MAXN = 500010;
const int INF = 0x3f3f3f3f;
int pre[MAXN],ch[MAXN][2],key[MAXN],size[MAXN];
int root,tot1;
int sum[MAXN],rev[MAXN],same[MAXN];
int lx[MAXN],rx[MAXN],mx[MAXN];
int s[MAXN],tot2;//内存池和容量
int a[MAXN];
int n,q;
//debug部分**********************************
void Treavel(int x)
{
    if(x)
    {
        Treavel(ch[x][0]);
        printf("结点: %2d: 左儿子 %2d 右儿子 %2d 父结点 %2d size = %2d\n",x,ch[x
            ][0],ch[x][1],pre[x],size[x]);
        Treavel(ch[x][1]);
    }
}
void debug()
{
    printf("root:%d\n",root);
    Treavel(root);
}
//以上是debug部分***********************************
void NewNode(int &r,int father,int k)
{
    if(tot2) r = s[tot2--];//取的时候是tot2--,存的时候就是++tot2
    else r = ++tot1;
    pre[r] = father;
    ch[r][0] = ch[r][1] = 0;
    key[r] = k;
    sum[r] = k;
    rev[r] = same[r] = 0;
    lx[r] = rx[r] = mx[r] = k;
    size[r] = 1;
}
void Update_Rev(int r)
{
    if(!r)return;
    swap(ch[r][0],ch[r][1]);
    swap(lx[r],rx[r]);
    rev[r] ^= 1;
}
void Update_Same(int r,int v)
{
    if(!r)return;
    key[r] = v;
    sum[r] = v*size[r];
    lx[r] = rx[r] = mx[r] = max(v,v*size[r]);
    same[r] = 1;
}
void push_up(int r)
{
    int lson = ch[r][0], rson = ch[r][1];
    size[r] = size[lson] + size[rson] + 1;
    sum[r] = sum[lson] + sum[rson] + key[r];
    lx[r] = max(lx[lson],sum[lson] + key[r] + max(0,lx[rson]));
    rx[r] = max(rx[rson],sum[rson] + key[r] + max(0,rx[lson]));
    mx[r] = max(0,rx[lson]) + key[r] + max(0,lx[rson]);
    mx[r] = max(mx[r],max(mx[lson],mx[rson]));
```

```
}
void push_down(int r)
{
    if(same[r])
    {
        Update_Same(ch[r][0],key[r]);
        Update_Same(ch[r][1],key[r]);
        same[r] = 0;
    }
    if(rev[r])
    {
        Update_Rev(ch[r][0]);
        Update_Rev(ch[r][1]);
        rev[r] = 0;
    }
}
void Build(int &x,int l,int r,int father)
{
    if(l > r)return;
    int mid = (l+r)/2;
    NewNode(x,father,a[mid]);
    Build(ch[x][0],l,mid-1,x);
    Build(ch[x][1],mid+1,r,x);
    push_up(x);
}
void Init()
{
    root = tot1 = tot2 = 0;
    ch[root][0] = ch[root][1] = size[root] = pre[root] = 0;
    same[root] = rev[root] = sum[root] = key[root] = 0;
    lx[root] = rx[root] = mx[root] = -INF;
    NewNode(root,0,-1);
    NewNode(ch[root][1],root,-1);
    for(int i = 0; i < n; i++)
        scanf("%d",&a[i]);
    Build(Key_value,0,n-1,ch[root][1]);
    push_up(ch[root][1]);
    push_up(root);
}
//旋转,0为左旋，1为右旋
void Rotate(int x,int kind)
{
    int y = pre[x];
    push_down(y);
    push_down(x);
    ch[y][!kind] = ch[x][kind];
    pre[ch[x][kind]] = y;
    if(pre[y])
        ch[pre[y]][ch[pre[y]][1]==y] = x;
    pre[x] = pre[y];
    ch[x][kind] = y;
    pre[y] = x;
    push_up(y);
}
//Splay调整，将r结点调整到goal下面
void Splay(int r,int goal)
{
    push_down(r);
    while(pre[r] != goal)
    {
        if(pre[pre[r]] == goal)
        {
            push_down(pre[r]);
```

```
126             push_down(r);
127             Rotate(r,ch[pre[r]][0] == r);
128         }
129         else
130         {
131             push_down(pre[pre[r]]);
132             push_down(pre[r]);
133             push_down(r);
134             int y = pre[r];
135             int kind = ch[pre[y]][0]==y;
136             if(ch[y][kind] == r)
137             {
138                 Rotate(r,!kind);
139                 Rotate(r,kind);
140             }
141             else
142             {
143                 Rotate(r,kind);
144                 Rotate(r,kind);
145             }
146         }
147     }
148     push_up(r);
149     if(goal == 0) root = r;
150 }
151 int Get_kth(int r,int k)
152 {
153     push_down(r);
154     int t = size[ch[r][0]] + 1;
155     if(t == k)return r;
156     if(t > k)return Get_kth(ch[r][0],k);
157     else return Get_kth(ch[r][1],k-t);
158 }
159 //在第pos个数后面插入tot个数
160 void Insert(int pos,int tot)
161 {
162     for(int i = 0; i < tot; i++)scanf("%d",&a[i]);
163     Splay(Get_kth(root,pos+1),0);
164     Splay(Get_kth(root,pos+2),root);
165     Build(Key_value,0,tot-1,ch[root][1]);
166     push_up(ch[root][1]);
167     push_up(root);
168 }
169 //删除子树
170 void erase(int r)
171 {
172     if(!r)return;
173     s[++tot2] = r;
174     erase(ch[r][0]);
175     erase(ch[r][1]);
176 }
177 //从第pos个数开始连续删除tot个数
178 void Delete(int pos,int tot)
179 {
180     Splay(Get_kth(root,pos),0);
181     Splay(Get_kth(root,pos+tot+1),root);
182     erase(Key_value);
183     pre[Key_value] = 0;
184     Key_value = 0;
185     push_up(ch[root][1]);
186     push_up(root);
187 }
188 //将从第pos个数开始的连续的tot个数修改为c
```

```
189 void Make_Same(int pos,int tot,int c)
190 {
191     Splay(Get_kth(root,pos),0);
192     Splay(Get_kth(root,pos+tot+1),root);
193     Update_Same(Key_value,c);
194     push_up(ch[root][1]);
195     push_up(root);
196 }
197 //将第pos个数开始的连续tot个数进行反转
198 void Reverse(int pos,int tot)
199 {
200     Splay(Get_kth(root,pos),0);
201     Splay(Get_kth(root,pos+tot+1),root);
202     Update_Rev(Key_value);
203     push_up(ch[root][1]);
204     push_up(root);
205 }
206 //得到第pos个数开始的tot个数的和
207 int Get_Sum(int pos,int tot)
208 {
209     Splay(Get_kth(root,pos),0);
210     Splay(Get_kth(root,pos+tot+1),root);
211     return sum[Key_value];
212 }
213 //得到第pos个数开始的tot个数中最大的子段和
214 int Get_MaxSum(int pos,int tot)
215 {
216     Splay(Get_kth(root,pos),0);
217     Splay(Get_kth(root,pos+tot+1),root);
218     return mx[Key_value];
219 }
220 void InOrder(int r)
221 {
222     if(!r)return;
223     push_down(r);
224     InOrder(ch[r][0]);
225     printf("%d ",key[r]);
226     InOrder(ch[r][1]);
227 }
228 int main()
229 {
230     //freopen("in.txt","r",stdin);
231     //freopen("out.txt","w",stdout);
232     while(scanf("%d%d",&n,&q) == 2)
233     {
234         Init();
235         char op[20];
236         int x,y,z;
237         while(q--)
238         {
239             scanf("%s",op);
240             if(strcmp(op,"INSERT") == 0)
241             {
242                 scanf("%d%d",&x,&y);
243                 Insert(x,y);
244             }
245             else if(strcmp(op,"DELETE") == 0)
246             {
247                 scanf("%d%d",&x,&y);
248                 Delete(x,y);
249             }
250             else if(strcmp(op,"MAKE-SAME") == 0)
251             {
```

```
252                 scanf("%d%d%d",&x,&y,&z);
253                 Make_Same(x,y,z);
254             }
255             else if(strcmp(op,"REVERSE") == 0)
256             {
257                 scanf("%d%d",&x,&y);
258                 Reverse(x,y);
259             }
260             else if(strcmp(op,"GET-SUM") == 0)
261             {
262                 scanf("%d%d",&x,&y);
263                 printf("%d\n",Get_Sum(x,y));
264             }
265             else if(strcmp(op,"MAX-SUM") == 0)
266                 printf("%d\n",Get_MaxSum(1,size[root]-2));
267         }
268     }
269     return 0;
270 }
```

14 现场赛宝典

1、比赛前晚一定要睡眠充足，至少保证 8-10 个小时睡眠时间。
2、热身赛不要纠结 AK 什么的，保证 A 掉一题就可以了。其他时间测试下编译器打表（多长能编译通过）、内存空间、快捷键等等。
3、热身赛每个人都轮流适应一下比赛机器。同时测试一下打印代码的流程，和提问、提交流程。要求每个队员都熟悉流程，保证提交不会出现错误。
4、热身赛后及时调节心情，如果一定要想出这道题怎么做，要果断问身边的大牛，没有疑惑和纠结度过周六。
5、赛前的心情非常非常重要！队友之间一定要互相开玩笑，保持愉快的心情开场。实在所有人都没有心情娱乐，那就脸部保持微笑状态三分钟，保证能改变心情。（强颜欢笑也是乐）
6、开场后分配好题目（一般前中后三部分），英语好要迅速读完题目，并且指定一个人每两三分钟看一次 rank，保证 FB 后迅速换题）
7、如果读题过程中能确定某题可出，并且为队友的擅长题型，迅速通知队友，并说清题意、题目条件，讨论是否可敲。如果敲题，这时候读完自己题目后，还继续把队友没读的题读完。
8、看到有人出题后一定要第一时间阅读该题，有条件最好两人同时读，并且保证题目描述、条件和输入输出都没有异议。讨论该题的可行性，如果完全没有思路，并且是神校出题，再观望一段时间，别盲目敲题。
9、一定一定要记住！比赛有五个小时，开场一定要淡定，保证好首次提交，很影响士气。两人仔细检查输入输出范围，多跑几遍样例，保证多 case 考虑了，保证输出格式、没多余空格、空行。可能的 trick 还有提交流程。
10、提交后有一段等待时间，这时候不要全部盯着屏幕等，不管对错，直接打印代码，然后再检查一遍输入输出和数据，检查代码有没有问题。空闲的人继续读题或看榜。
11、如果提交没过，有其他题在开着，换人敲。之前的人拿着打印的代码检查 bug。如不能保证 bug 很快找到，至少两个人同时 debug，不需要双开。
12、如果某题 2 次 WA，并且找不到原因，这时候一边 debug 一边让别人重新读一遍题目，逐字分析条件和输入输出，不受其他人影响，然后交流题意等等是否出现问题。题意和算法没问题的情况下，想 trick 出数据。
13、YES 了一道题后，主敲队员最好的放松方法就是去趟洗手间！不管想不想方便，去洗个脸，呼吸一下新鲜空气也是很有必要的。清醒一下，把 YES 的那道题抛到脑后，关注其他题。还有就是在脑子非常乱的时候，非常憋屈的时候，也去趟洗手间，说不定还能偷瞄到大神的思路……
14、比赛士气和心情很重要，一定要有一个队员心理素质够硬，负责调动全队状态。小口喝水、身体坐正，双手交叉，互相打气能快速帮你调整状态。
15、比赛期间以观察 rank 为出题标准，但是如果认定该题不适合你们，要果断先选择另外一道过的人较多的题！
16、在没有两个人都足够主敲的情况下，切忌双开敲题，尤其到了最后一两个小时，一般三人全力攻题！出题的同时，空闲队员想好 trick 数据等等。
17、封榜后一定不能乱，不要盲目交题，不要重复交同样代码!! 参考 11、12
18、记得比赛比的是心态和状态，如果浮躁，自己就先输了，一定要淡定！如果开始自己或队友发现出现浮躁慌乱，一定要提醒大家，参考 13、14
19、相信自己，相信队友，和谐相处，共同拿牌!!!
20、欢迎补充