LargeDumpling's Template for Programing Contest

I. Data Structure

1. Segment Tree

(1) Classical Segment Tree

```
const int MAXN=200050;
int n,m,L[MAXN<<2],R[MAXN<<2];
long long num[MAXN],d[MAXN<<2],tag[MAXN<<2];
#define lch ((root)<<1)
#define rch ((root)<<1|1)
void maintain(int root)
{
    if(L[root]==R[root]) return;
    d[root]=d[lch]+d[rch];
    return;
}
void down(int root)
{
    if(!tag[root]) return;
    if(L[root]==R[root])
    {
        tag[root]=0;
        return;
    }
    d[lch]+=tag[root]*(R[lch]-L[lch]+1LL);
    d[rch]+=tag[root]*(R[rch]-L[rch]+1LL);
    tag[lch]+=tag[root];
    tag[rch]+=tag[root];
    tag[root]=0;
    return;
}
void build(int root,int l,int r)
{
    L[root]=l; R[root]=r;
    tag[root]=0;
    if(l==r)
    {
        d[root]=num[l];
        return;
    }
```

```
    int mid=(l+r)>>1;
    build(lch,l,mid);
    build(rch,mid+1,r);
    maintain(root);
    return;
}
void change(int root,int l,int r,long long x)
{
    if(l<=L[root]&&R[root]<=r)
    {
        tag[root]+=x;
        d[root]+=(R[root]-L[root]+1LL)*x;
        return;
    }
    down(root);
    int mid=(L[root]+R[root])>>1;
    if(l<=mid) change(lch,l,r,x);
    if(mid<r) change(rch,l,r,x); // *** NO "else" before "if"
    maintain(root);
    return;
}
long long query(int root,int l,int r)
{
    if(l<=L[root]&&R[root]<=r) return d[root];
    down(root);
    int mid=(L[root]+R[root])>>1;
    long long sum=0;
    if(l<=mid) sum+=query(lch,l,r);
    if(mid<r) sum+=query(rch,l,r); // *** NO "else" before "if" AGAIN
    return sum;
}
#undef lch
#undef rch
```

(2) ZKW Segment Tree

```
/*
 * 1, M should be a power to two that is bigger than n;
 * 2, Operations are valid only in the interval [1,M-2], otherwise unexcepted errors may happen.
 */
const int M=131072;
int data[M<<1];
void change(int pos,int x)
{
    data[pos=pos+M]+=x;
    for(pos>>=1;pos;pos>>=1)
        data[pos]=data[pos<<1]+data[pos<<1|1];
```

```
    return;
}
int query(int L,int R)
{
    int sum=0;
    for(L=L+M-1,R=R+M+1;L<R-1;L>>=1,R>>=1)
    {
        if(!(L&1)) sum+=data[L^1];
        if(R&1) sum+=data[R^1];
    }
    return sum;
}
```

(3) Persistent Segment Tree

```
const int MAXN=100050;
int d[MAXN*30],ch[MAXN*30][2],root[MAXN],sz=0;
void add(int &now,int pre,int L,int R,int p,int x)
{
    now=++sz;
    memcpy(ch[now],ch[pre],sizeof(ch[pre]));
    d[now]=d[pre]+x;
    if(L==R) return;
    int mid=(L+R)>>1;
    if(p<=mid) add(ch[now][0],ch[pre][0],L,mid,p,x);
    else if(mid<p) add(ch[now][1],ch[pre][1],mid+1,R,p,x);
    return;
}
int query(int pre,int now,int L,int R,int k)
{
    if(L==R) return L;
    int mid=(L+R)>>1;
    if(k<=d[ch[now][0]]-d[ch[pre][0]]) return query(ch[pre][0],ch[now][0],L,mid,k);
    k-=(d[ch[now][0]]-d[ch[pre][0]]);
    return query(ch[pre][1],ch[now][1],mid+1,R,k);
}
```

2. Binary Indexed Tree

```
#define low(x) ((x)&(-x))
const int MAXN=100050;
int d[MAXN];
void add(int p,int x)
{
    for(;p<MAXN;p+=low(p))
        d[p]+=x;
    return;
```

```
}
int query(int x)
{
    int sum=0;
    for(;x;x-=low(x))
        sum+=d[x];
    return sum;
}
```

3. Binary Search Tree

(1) Treap

```
struct jp *null;
struct jp
{
    int key,num,size,v;
    jp *son[2];
    jp(const int &X=0):key(X) { num=size=1; v=rand(); son[0]=son[1]=null; }
    int cmp(const int &x) { return x==key?-1:(x<key?0:1); }
    void updata() { size=son[0]->size+son[1]->size+num; }
}*root;
void adjust(jp* &r,int d)
{
    jp *u=r->son[d]; r->son[d]=u->son[d^1]; u->son[d^1]=r;
    r->updata(); (r=u)->updata(); return;
}
void insert(jp* &r,int x)
{
    if(r==null)
    {
        r=new jp(x);
        return;
    }
    int d=r->cmp(x);
    if(d==-1) r->num++;
    else
    {
        insert(r->son[d],x);
        if(r->son[d]->v > r->v) adjust(r,d);
    }
    r->updata();
    return;
}
bool del(jp* &r,int x)
{
    if(r==null) return false;
```

```
    int d=r->cmp(x);
    if(d==-1)
    {
        if(r->num>1) r->num--;
        else
        {
            if(r->son[0]==null||r->son[1]==null)
            {
                jp *u=r;
                r=r->son[0]==null?r->son[1]:r->son[0];
                delete u;
            }
            else
            {
                d=r->son[0]->v < r->son[1]->v;
                adjust(r,d);
                del(r->son[d^1],x);
            }
        }
        r->updata();
        return true;
    }
    bool flag=del(r->son[d],x);
    if(flag) r->updata();
    return flag;
}
int rank(jp* &r,int x)
{
    if(r==null) return 0;
    int d=r->cmp(x);
    if(!d) return rank(r->son[0],x);
    return r->son[0]->size+(d==-1?1:rank(r->son[1],x)+r->num);
}
int kth(jp* &r,int rk)
{
    if(r==null) return 0;
    if(rk<=r->son[0]->size) return kth(r->son[0],rk);
    rk-=(r->son[0]->size+r->num);
    return rk>0?kth(r->son[1],rk):r->key;
}
int query(int x,int D)
{
    jp *u=root; int ans=0,d;
    while(u!=null)
    {
        d=u->cmp(x);
        if(d==(D^1)) ans=u->key;
        u=u->son[d==-1?D:d];
```

```
    }
    return ans;
}
root=null=new jp();
null->son[0]=null->son[1]=null;
null->num=null->size=0;
```

(2) Splay

```
struct jp *null;
struct jp
{
    int key,size;
    bool flag;
    jp *son[2];
    jp(const int &x=0):key(x) { size=1; flag=false; son[0]=son[1]=null; }
    int cmp(int k) { return k==(son[0]->size+1)?-1:(k<=son[0]->size?0:1); }
    void updata() { size=son[0]->size+son[1]->size+1; return; }
    void down()
    {
        if(!flag) return;
        flag=false;
        swap(son[0],son[1]);
        son[0]->flag^=true;
        son[1]->flag^=true;
        return;
    }
}*root;
int cnt=0,n,m;
void adjust(jp* &r,int d)
{
    jp* u=r->son[d]; r->son[d]=u->son[d^1]; u->son[d^1]=r;
    r->updata(); (r=u)->updata(); return;
}
void build(jp* &u,int l,int r)
{
    if(l>r) { u=null; return; }
    int mid=(l+r)>>1;
    u=new jp();
    build(u->son[0],l,mid-1);
    u->key=cnt++;
    build(u->son[1],mid+1,r);
    u->updata();//勿忘
    return;
}
void m41441(jp* &r,int k)
{
```

```
    r->down();
    int d=r->cmp(k);
    if(d==1) k=k-r->son[0]->size-1;
    if(d!=-1)
    {
        jp* u=r->son[d];
        u->down();
        int d2=u->cmp(k);
        int k2=d2==1?k-u->son[0]->size-1:k;
        if(d2!=-1)
        {
            m41441(u->son[d2],k2);
            if(d2==d) adjust(r,d);
            else adjust(r->son[d],d2);
        }
        adjust(r,d);
    }
    return;
}
void split(jp* all,int k,jp* &left,jp* &right)
{
    m41441(all,k);
    left=all;
    right=all->son[1];
    left->son[1]=null;
    left->updata();
    return;
}
jp *merge(jp* left,jp* right)
{
    m41441(left,left->size);
    left->son[1]=right;
    left->updata();
    return left;
}
void rEverse(int l,int r)
{
    jp *temp,*left,*mid,*right;
    split(root,l,left,temp);
    split(temp,r-l+1,mid,right);
    mid->flag^=true;
    root=merge(merge(left,mid),right);
    return;
}
int tot=0;
void print(jp* &r)
{
    if(r==null) return;
```

```
    r->down();
    print(r->son[0]);
    if(r->key)
    {
        printf("%d",r->key);
        if(++tot!=n) putchar(' ');
    }
    print(r->son[1]);
    return;
}
root=null=new jp();
null->son[0]=null->son[1]=null;
null->size=0;
build(root,0,n);
```

4. Mergeable Heap

(1) Skew Heap

```
template<typename Ty>
struct heap
{
    static const int MAXN=1000050;
    int tOp;
    int L[MAXN],R[MAXN],dist[MAXN],p[MAXN];
    Ty data[MAXN];
    heap()
    {
        tOp=0;
        memset(L,0,sizeof(L));
        memset(R,0,sizeof(R));
        memset(dist,0,sizeof(dist));
        for(int i=1;i<MAXN;i++) p[i]=i;
        p[0]=MAXN-1;
    }
    void clear()
    {
        tOp=0;
        memset(L,0,sizeof(L));
        memset(R,0,sizeof(R));
        memset(dist,0,sizeof(dist));
        for(int i=1;i<MAXN;i++) p[i]=i;
        p[0]=MAXN-1;
        return;
    }
    bool empty() { return (bool)tOp; }
    int merge(int A,int B)
```

```
    {
        if(!A) return B;
        if(!B) return A;
        if(data[A]>data[B]) swap(A,B);
        R[A]=merge(R[A],B);
        if(dist[L[A]]<dist[R[A]])
            swap(L[A],R[A]);
        dist[A]=R[A]?dist[R[A]]+1:0;
        return A;
    }
    void push(Ty X)
    {
        data[p[p[0]]]=X;
        tOp=merge(tOp,p[p[0]--]);
        return;
    }
    void pop()
    {
        p[++p[0]]=tOp;
        tOp=merge(L[tOp],R[tOp]);
        L[p[p[0]]]=R[p[p[0]]]=0;
        return;
    }
    Ty top() { return data[tOp]; }
};
```

II. String Algorithm

1. The Knuth-Morris-Pratt Algorithm

(1) Classical KMP

```
const int MAXL=100050;
char tex[MAXL],T[MAXL];
int pre[MAXL];
int KMP()
{
    int n=strlen(tex),m=strlen(T),i,k;
    pre[0]=-1;
    for(i=1,k=-1;i<n;pre[i++]=k)
    {
        while(k>=0&&tex[i]!=tex[k+1])
            k=pre[k];
        if(tex[i]==tex[k+1])
            k++;
```

```
    }
    for(i=0,k=-1;i<m;i++)
    {
        while(k>=0&&T[i]!=tex[k+1])
            k=pre[k];
        if(T[i]==tex[k+1])
        {
            k++;
            if(k==n-1) return i-k;
        }
    }
    return -1;
}
```

(2) Extended KMP

```
const int maxn=100010;   //字符串长度最大值
int next[maxn],ex[maxn]; //ex 数组即为 extend 数组
//预处理计算 next 数组
void GETNEXT(char *str)
{
    int i=0,j,po,len=strlen(str);
    next[0]=len;//初始化 next[0]
    while(str[i]==str[i+1]&&i+1<len)//计算 next[1]
    i++;
    next[1]=i;
    po=1;//初始化 po 的位置
    for(i=2;i<len;i++)
    {
        if(next[i-po]+i<next[po]+po)//第一种情况，可以直接得到 next[i]的值
        next[i]=next[i-po];
        else//第二种情况，要继续匹配才能得到 next[i]的值
        {
            j=next[po]+po-i;
            if(j<0)j=0;//如果 i>po+next[po],则要从头开始匹配
            while(i+j<len&&str[j]==str[j+i])//计算 next[i]
            j++;
            next[i]=j;
            po=i;//更新 po 的位置
        }
    }
}
//计算 extend 数组
void EXKMP(char *s1,char *s2)
{
    int i=0,j,po,len=strlen(s1),l2=strlen(s2);
    GETNEXT(s2);//计算子串的 next 数组
```

```
    while(s1[i]==s2[i]&&i<l2&&i<len)//计算 ex[0]
    i++;
    ex[0]=i;
    po=0;//初始化 po 的位置
    for(i=1;i<len;i++)
    {
        if(next[i-po]+i<ex[po]+po)//第一种情况，直接可以得到 ex[i]的值
        ex[i]=next[i-po];
        else//第二种情况，要继续匹配才能得到 ex[i]的值
        {
            j=ex[po]+po-i;
            if(j<0)j=0;//如果 i>ex[po]+po 则要从头开始匹配
            while(i+j<len&&j<l2&&s1[j+i]==s2[j])//计算 ex[i]
            j++;
            ex[i]=j;
            po=i;//更新 po 的位置
        }
    }
}
```

2. Manacher

```
const int MAXL=11000050;
char str[MAXL<<1];
int r[MAXL<<1],len;
/*
 * 1, str should have double lenth
 */
void Manacher()
{
    int rig=0,id=0;
    len=strlen(str);
    str[len<<1]='#';
    for(int i=len-1;0<=i;i--)
    {
        str[i<<1|1]=str[i];
        str[i<<1]='#';
    }
    r[0]=0;
    for(int i=1;i<=(len<<1);i++)
    {
        if(i<=rig) r[i]=min(rig-i,r[2*id-i]);
        else r[i]=0;
        while(0<=i-r[i]-1
                &&i+r[i]+1<=(len<<1)
                &&str[i-r[i]-1]==str[i+r[i]+1])
            r[i]++;
```

```
        if(rig<i+r[i])
        {
            rig=i+r[i];
            id=i;
        }
    }
    return;
}
for(int i=0;i<=(len<<1);i++)
    ans=max(ans,r[i]);
```

3. Aho-Corasick automaton

```
#include<queue>
const int MAXN=100050;
const int MAXC=26;
int ch[MAXN][MAXC],val[MAXN],last[MAXN],pre[MAXN],sz=0;
void insert(char T[],int x)
{
    int lenth=strlen(T),u=0;
    for(int i=0;i<lenth;u=ch[u][T[i++]-97])
        if(!ch[u][T[i]-97])
            ch[u][T[i]-97]=++sz;
    val[u]=x;
    return;
}
void getfail()
{
    int u,v,f;
    queue<int>q;
    for(int i=0;i<MAXC;i++)
    {
        if(!ch[0][i])
            continue;
        pre[ch[0][i]]=last[ch[0][i]]=0;
        q.push(ch[0][i]);
    }
    while(q.size())
    {
        u=q.front();
        q.pop();
        for(int i=0;i<MAXC;i++)
        {
            if(!ch[u][i])
            {
                ch[u][i]=ch[pre[u]][i];
                continue;
```

```
            }
            v=ch[u][i];
            q.push(v);
            f=pre[u];
            while(f&&!ch[f][i])
                f=pre[f];
            pre[v]=ch[f][i];
            last[v]=val[pre[v]]?pre[v]:last[pre[v]];
        }
    }
    return;
}
void count(int u)
{
    if(u)
    {
        // do some operators
        if(last[u]) count(last[u]);
    }
    return;
}
void find(char T[])
{
    int lenth=strlen(T),u=0;
    for(int i=0;i<lenth;i++)
    {
        /*while(u&&!ch[u][T[i]-97])
            u=pre[u];*/
        u=ch[u][T[i]-97];
        if(val[u]) count(u);
        else if(last[u]) count(last[u]);
    }
    return;
}
```

4. Palindrome Automaton

```
const int MAXN=100050;
const int MAXC=26;
int ch[MAXN][MAXC],len[MAXN],fail[MAXN],cnt[MAXN],last,sz;
int getfail(char T[],int x,int i)
{
    while(T[i-len[x]-1]!=T[i]) x=fail[x];
    return x;
}
void init()
{
```

```
    memset(ch[0],last=0,sizeof(ch[0]));//last 为当前的最长后缀回文
    memset(ch[1],0,sizeof(ch[1]));
    len[0]=0; len[1]=-1; fail[0]=1;
    sz=1;
    return;
}
void insert(char T[])
{
    int lenth=strlen(T),cur;
    for(int i=0;i<lenth;i++)
    {
        cur=getfail(T,last,i);
        if(!ch[cur][T[i]-97])
        {
            fail[++sz]=ch[getfail(T,fail[cur],i)][T[i]-97];
            ch[cur][T[i]-97]=sz;
            memset(ch[sz],0,sizeof(ch[sz]));
            len[sz]=len[cur]+2;
        }
        last=ch[cur][T[i]-97];
        cnt[last]++;
    }
}
```

5. Suffix Array

```
const int MAXN=100050;
char str[MAXN];
int len,sa[MAXN],rank[MAXN],height[MAXN];
int num[MAXN],bsa[MAXN],c[MAXN];
void build_sa()
{
    int *x=num,*y=bsa,m=26,p,i,k;
    for(i=0;i<m;i++) c[i]=0;
    for(i=0;i<len;i++) c[x[i]=str[i]-'a']++;
    for(i=1;i<m;i++) c[i]+=c[i-1];
    for(i=len-1;0<=i;i--) sa[--c[x[i]]]=i;
    for(k=1;k<=len;k<<=1)
    {
        p=0;
        for(i=len-1;len-k<=i;i--) y[p++]=i;
        for(i=0;i<len;i++) if(sa[i]>=k) y[p++]=sa[i]-k;
        for(i=0;i<m;i++) c[i]=0;
        for(i=0;i<len;i++) c[x[y[i]]]++;
        for(i=1;i<m;i++) c[i]+=c[i-1];
        for(i=len-1;0<=i;i--) sa[--c[x[y[i]]]]=y[i];
        swap(x,y);
```

```
        p=1;
        x[sa[0]]=0;
        for(i=1;i<len;i++)
            x[sa[i]]=(y[sa[i-1]]==y[sa[i]]&&y[sa[i-1]+k]==y[sa[i]+k])?p-1:p++;
        if(len<=p) break;
        m=p;
    }
    return;
}
void build_height()
{
    int k=0,j,i;
    for(i=0;i<len;i++) rank[sa[i]]=i;
    for(i=0;i<len;i++)
    {
        if(k) k--;
        if(!rank[i]) continue;
        j=sa[rank[i]-1];
        while(str[i+k]==str[j+k]) k++;
        height[rank[i]]=k;
    }
    return;
}
```

6. Minimum representation

```
int minP(char s[])
{
    int l=strlen(s);
    int i=0,j=1,k=0;
    while(true)
    {
        if(i+k>=l||j+k>=l) break;
        if(s[i+k]==s[j+k]) k++;
        else
        {
            if(s[j+k]>s[i+k]) j+=k+1;
            else i+=k+1;
            k=0;
            if(i==j) j++;
        }
    }
    return min(i,j);
}
```

III. Mathematics

1. Berlekamp-Massey Algorithm

```cpp
#include <cstdio>
#include <vector>
using namespace std;
namespace BerlekampMassey {
    const int mod = 1e9 + 7;
    int L, m, b, n;
    vector<int> s, C, B;
    void init() {
        s.clear();
        C.clear();
        B.clear();
        C.push_back(1);
        B.push_back(1);
        L = n = 0;
        m = b = 1;
    }
    int pow_mod(int a, int k) {
        int s = 1;
        while (k) {
            if (k & 1)
                s = 1ll * s * a % mod;
            a = 1ll * a * a % mod;
            k >>= 1;
        }
        return s;
    }
    void update(int d) {
        s.push_back(d);
        for (int i = 1; i <= L; ++i)
            d = (d + 1ll * C[i] * s[n - i] % mod) % mod;
        if (d == 0)
            ++m;
        else if (2 * L <= n) {
            vector<int> T = C;
            C.resize(n + 1 - L + 1);
            for (int i = L + 1; i <= n + 1 - L; ++i)
                C[i] = 0;
            for (int i = 0; i < B.size(); ++i)
                C[i + m] = (C[i + m] + mod - 1ll * d * pow_mod(b, mod - 2) % mod * B[i] % mod) % mod;

            L = n + 1 - L;
            B = T;
```

```
            b = d;
            m = 1;
        } else {
            for (int i = 0; i < B.size(); ++i)
                C[i + m] = (C[i + m] + mod - 1ll * d * pow_mod(b, mod - 2) % mod * B[i] % mod) % mod;
            ++m;
        }
        ++n;
    }
    void output() {
        printf("F(n)=");
        for (int i = 1; i < C.size(); ++i) {
            int output = (mod - C[i]) % mod;
            if (output > mod / 2)
                output -= mod;
            printf("%s%d*F(n-%d)", (output < 0 || i == 1) ? "" : "+", output, i);
        }
        puts("");
    }
    void output_code_for() {
        static const char *name = "dp";
        static const char *index = "i";
        static const char *upperbound = "maxn";
        puts("// Generated by Berlekamp-Massey algorithm");
        for (int i = 1; i < C.size(); ++i) {
            printf("%s[%d]=%d;\n", name, i - 1, s[i - 1]);
        }
        printf("for(int i=%d;i<%s;++i)\n", (int)C.size() - 1, upperbound);
        printf("  %s[%s]=((", name, index);
        for (int i = 1; i < C.size(); ++i) {
            int output = (mod - C[i]) % mod;
            if (output > mod / 2)
                output -= mod;
            printf("%s%d*%s[%s-%d]%%mod", (output < 0 || i == 1) ? "" : "+", output, name, index, i);
        }
        puts(")%mod+mod)%mod;");
    }
    void output_code_matrix() {
        // TODO
    }
};

#include "BerlekampMassey.cpp"
int main() {
    int arr[12] = {2, 24, 96, 416, 1536, 5504, 18944, 64000, 212992, 702464, 2301952, 7512064};
    BerlekampMassey::init();
```

```
    for (int i = 0; i < 12; ++i) {
        BerlekampMassey::update(arr[i]);
    }
    printf("Formule: ");
    BerlekampMassey::output();
    printf("Code: \n");
    BerlekampMassey::output_code_for();
    return 0;
}
```

2. Euclid's algorithm

(1) Classical

```
/*
 * ax+by=gcd(a,b)=d
 */
void gcd(int a,int b,int &d,int &x,int &y)
{
    if(!b)
    {
        d=a;
        x=1;
        y=0;
    }
    else
    {
        gcd(b,a%b,d,y,x);
        y-=x*(a/b);
    }
    return;
}
```

(2) Quasi Euclid's algorithm

3. Sieve

(1) Sieve of Eratosthenes

```
const int MAXN=100050;
bool vis[MAXN];
void Eratosthenes()
{
    int m=sqrt(n+0.5);
```

```
    memset(vis,false,sizeof(vis));
    for(int i=2;i<=m;i++) if(!vis[i])
        for(int j=i*i;j<=n;j+=i) vis[j]=true;
    return;
}
```

(2) Sieve of Euler

```
const int MAXN=100050;
int pri[MAXN],phi[MAXN],u[MAXN];
bool vis[MAXN];
void Euler()
{
    memset(vis,true,sizeof(vis));
    pri[0]=0;
    u[1]=1; phi[1]=1;
    for(int i=2;i<MAXN;i++)
    {
        if(vis[i])
        {
            pri[++pri[0]]=i;
            phi[i]=i-1;
            u[i]=-1;
        }
        for(int j=1;j<=pri[0]&&i*pri[j]<MAXN;j++)
        {
            vis[i*pri[j]]=false;
            if(i%pri[j])
            {
                phi[i*pri[j]]=phi[i]*(pri[j]-1);
                u[i*pri[j]]=-u[i];
            }
            else
            {
                phi[i*pri[j]]=phi[i]*pri[j];
                break;
            }
        }
    }
}
```

4. Möbius inversion

$$\sum_{d|n} \mu(d) = [n = 1]$$

(1) Divisor

$$f(n)=\sum_{d|n} g(d)$$
$$g(n)=\sum_{d|n} \mu(d) f\left(\frac{n}{d}\right)$$

(2) Multiple

$$f(n)=\sum_{n|d} g(d)$$
$$g(n)=\sum_{n|d} \mu\left(\frac{d}{n}\right) f(d)$$

5. Gauss-Jordan Elimination

(1) System of Linear Equations

```
const int MAXN=1050;
const double eps=1e-8;
typedef double Matrix[MAXN][MAXN];
void Gauss_Jordan_Elimination(Matrix A,int n)
{
    int i,j,k,r;
    for(i=0;i<n;i++)
    {
        r=i;
        for(j=i+1;j<n;j++)
        {
            if(abs(A[j][i]>fabs(A[r][i]))) r=j;
            //if(fabs(A[r][i])<eps) continue;
            if(r!=i) for(int j=0;j<=n;j++)
                swap(A[r][j],A[i][j]);
        }
        for(k=0;k<n;k++) if(k!=i)
            for(int j=n;j>=i;j--)
                A[k][j]-=A[k][i]/A[i][i]*A[i][j];
    }
    return;
}
```

(2) System of XOR Equations

```
const int MAXM=512;
vector<bitset<MAXM> >equ;
int Free=0;
void gauss()
{
    int limit=equ.size(),tot=0;
    int j;
    for(int i=0;i<limit;i++)
    {
        for(j=tot;j<limit&&!equ[j][i];j++);
        if(j==limit) Free++;
        else
        {
            swap(equ[tot++],equ[j]);
            for(j=tot;j<limit;j++) if(equ[j][i])
                equ[j]^=equ[tot-1];
        }
    }
    return;
}
```

6. Fast Fourier Transform

```
struct cp
{
    double r,i;
    cp(double a=0,double b=0):r(a),i(b){}
    cp operator+(cp X)const{return cp(r+X.r,i+X.i);}
    cp operator-(cp X)const{return cp(r-X.r,i-X.i);}
    cp operator*(cp X)const{return cp(r*X.r-i*X.i,r*X.i+i*X.r);}
}A[400050],B[400050];
int rev[400050];
void FFT_pre(int &n)
{
    int tem; for(tem=0;(1<<tem)<n;tem++); n=1<<tem;
    rev[0]=0;
    for(int i=1;i<n;i++) rev[i]=(rev[i>>1]>>1)|((i&1)<<(tem-1));
    return;
}
void DFT(cp X[],int n,bool inv)
{
    int temk;
    double wn1=inv?-2.*acos(-1.):2.*acos(-1.);
    cp w,cur,tem;
```

```
    for(int i=1;i<n;i++) if(rev[i]>i)swap(X[i],X[rev[i]]);
    for(int k=2;k<=n;k<<=1)
    {
        temk=k>>1;
        cur=cp(1.,0.);
        w=cp(cos(wn1/k),sin(wn1/k));
        for(int j=0;j<temk;j++)
        {
            for(int i=0;i<n;i+=k)
            {
                tem=cur*X[i+j+temk];
                X[i+j+temk]=X[i+j]-tem;
                X[i+j]=X[i+j]+tem;
            }
            cur=cur*w;
        }
    }
    if(inv) for(int i=0;i<n;i++) X[i].r=X[i].r/n;
    return;
}
int main()
{
    int n,m,lenth;
    scanf("%d%d",&n,&m);
    n++;m++;
    lenth=n+m-1;
    for(int i=0;i<n;i++) scanf("%lf",&A[i].r);
    for(int i=0;i<m;i++) scanf("%lf",&B[i].r);
    n=n<m?m:n;
    n<<=1;
    FFT_pre(n);
    //for(int i=0;i<n;i++) printf("%d ",rev[i]);
    DFT(A,n,false);
    DFT(B,n,false);
    for(int i=0;i<n;i++) A[i]=A[i]*B[i];
    DFT(A,n,true);
    for(int i=0;i<lenth;i++) printf("%d ",(int)(A[i].r+0.5));
    return 0;
}
```

7. Number Theoretic Transform

```
const int P=(479<<21)+1,G=3,D=10000000;
int A[405000],B[405000],rev[400050],n,m,lenth,inver;
int pOw(int a,int n)
{
    int ans;
```

```
    for(ans=1;n;n>>=1,a=(int)((long long)a*a%P))
        if(n&1)ans=(int)((long long)ans*a%P);
    return ans;
}
void FFT_pre(int &n)
{
    int tem; for(tem=0;(1<<tem)<n;tem++); n=1<<tem;
    for(int i=0;i<n;i++) rev[i]=rev[i>>1]>>1|((i&1)<<(tem-1));
    inver=pOw(n,P-2);
    return;
}
void FFT(int X[],int n,bool inv)
{
    for(int i=0;i<n;i++) if(rev[i]>i) swap(X[i],X[rev[i]]);
    int temk,w,cur,temp;
    for(int k=2;k<=n;k<<=1)
    {
        temk=k>>1;
        cur=1; w=pOw(G,(P-1)/k);
        if(inv) w=pOw(w,k-1);
        for(int i=0;i<temk;i++)
        {
            for(int j=0;j<n;j+=k)
            {
                temp=(int)((long long)cur*X[i+j+temk]%P);
                X[i+j+temk]=(int)((long long)(X[i+j]-temp+P)%P);
                X[i+j]=(int)((long long)(X[i+j]+temp)%P);
            }
            cur=(int)((long long)cur*w%P);
        }
    }
    if(inv) for(int i=0;i<n;i++) X[i]=(int)((long long)X[i]*inver%P);
    return;
}
int main()
{
    read1n(n); read1n(m);
    n++; m++;
    lenth=n+m-1;
    n=max(n,m);
    n<<=1;
    FFT_pre(n);
    FFT(A,n,false);
    FFT(B,n,false);
    for(int i=0;i<n;i++) A[i]=(int)((long long)A[i]*B[i]%P);
    FFT(A,n,true);
    return 0;
}
```

8. Fast Walsh-Hadamard Transform

(1) OR

```
void FWT(int *P,int opt)
{
    for(int i=2;i<=N;i<<=1)
        for(int p=i>>1,j=0;j<N;j+=i)
            for(int k=j;k<j+p;++k)
                P[k+p]+=P[k]*opt;
}
```

(2) AND

```
void FWT(int *P,int opt)
{
    for(int i=2;i<=N;i<<=1)
        for(int p=i>>1,j=0;j<N;j+=i)
            for(int k=j;k<j+p;++k)
                P[k]+=P[k+p]*opt;
}
```

(3) XOR

```
void FWT(int *P,int opt)
{
    for(int i=2;i<=N;i<<=1)
        for(int p=i>>1,j=0;j<N;j+=i)
            for(int k=j;k<j+p;++k)
            {
                int x=P[k],y=P[k+p];
                P[k]=(x+y)%MOD;P[k+p]=(x-y+MOD)%MOD;
                if(opt==-1)P[k]=1ll*P[k]*inv2%MOD,P[k+p]=1ll*P[k+p]*inv2%MOD;
            }
}
```

9. Matrix-Tree Theorem

```
#include <iostream>
#include <cstdio>
#include <cstring>
#include <algorithm>
#include <cmath>
using namespace std;
```

```
7.
8.  const int maxn =13;
9.  typedef long long LL;
10.
11. int degree[maxn];
12. LL C[maxn][maxn];
13.
14. LL det(LL a[][maxn],int n){
15.     LL ret=1;
16.     for(int i=0;i<n;i++){
17.         for(int j=i+1;j<n;j++){
18.             while(a[j][i]){
19.                 LL t=a[i][i]/a[j][i];
20.                 for(int k=i;k<n;k++){
21.                     a[i][k]=a[i][k]-a[j][k]*t;
22.                 }
23.
24.                 for(int k=i;k<n;k++){
25.                     swap(a[i][k],a[j][k]);
26.                 }
27.                 ret=-ret;
28.             }
29.         }
30.
31.         if(a[i][i]==0){
32.             return 0;
33.         }
34.         ret=ret*a[i][i];
35.     }
36.
37.     if(ret<0){
38.         ret=-ret;
39.     }
40.     return ret;
41. }
42.
43. int main(){
44.     int t,n,m;
45.     scanf("%d",&t);
46.     while(t--){
47.         memset(degree,0,sizeof(degree));
48.         memset(C,0,sizeof(C));
49.         scanf("%d%d",&n,&m);
50.         while(m--){
51.             int u,v;
52.             scanf("%d%d",&u,&v);
53.             u--;v--;
54.             C[u][v]=C[v][u]=-1;
```

```
            degree[v]++;
            degree[u]++;
        }

        for(int i=0;i<n;i++){
            C[i][i]=degree[i];
        }
        printf("%lld\n",det(C,n-1));
    }
    return 0;
}
```

10.Chinese Remainder Theorem

```
//n 个方程:x=a[i](mod m[i]) 0<=i<n
typedef long long LL;
LL CRT(int n,int *a,int *m)
{
    LL M=1,d,y,x=0;
    for(int i=0;i<n;i++) M*=m[i];
    for(int i=0;i<n;i++)
    {
        LL w=M/m[i];
        gcd(m[i],w,d,d,y);
        x=(x+y*w*a[i])%M;
    }
    return (x+M)%M;
}
```

11.Miller-Rabin Test

```
typedef unsigned long long ull;
ull pri[13]={12,2,3,5,7,11,13,17,19,23,29,31,37};
bool Miller_Rabin(ull X)
{
    ull a,x,y,u,v;
    for(ull i=1;i<=pri[0];i++)
    {
        if(X==pri[i]) return true;
        if(!(X&pri[i]))
            return false;
    }
    for(v=X-1,u=0;!(v%2);v>>=1,u++);
    for(int T=1;T<=20;T++)
    {
        a=rand()%(x-2)+2;
        x=pow(a,v,X);
```

```
        for(ull i=1;i<=u;i++,x=y)
        {
            y=mUl(x,x,X);
            if(y==1&&x!=1&&x!=X-1)
                return false;
        }
        if(x!=1) return false;
    }
    return true;
}
```

IV. Gragh Theory

1. Network-flow

(1) Dinic

```
const int MAXN=3050;
const int MAXM=4000050;
const int INF=1000000050;
int fir[MAXN],cur[MAXN],lev[MAXN],end[MAXM],next[MAXM],f[MAXM],ed,S,T;
void addedge(int u,int v,int cap)
{
    end[++ed]=v;
    next[ed]=fir[u];
    fir[u]=ed;
    f[ed]=cap;
    end[++ed]=u;
    next[ed]=fir[v];
    fir[v]=ed;
    f[ed]=0;
    return;
}
bool BFS()
{
    int u;
    memset(lev,-1,sizeof(lev));
    queue<int>q;
    lev[S]=0;
    q.push(S);
    while(q.size())
    {
        u=q.front(); q.pop();
        for(int i=fir[u];i;i=next[i]) if(f[i]&&lev[end[i]]==-1)
        {
```

```
            lev[end[i]]=lev[u]+1;
            q.push(end[i]);
        }
    }
    memcpy(cur,fir,sizeof fir);
    return lev[T]!=-1;
}
int DFS(int u,int maxf)
{
    if(u==T||!maxf) return maxf;
    int cnt=0;
    for(int &i=cur[u],tem;i;i=next[i]) if(f[i]&&lev[end[i]]==lev[u]+1)
    {
        tem=DFS(end[i],min(maxf,f[i]));
        maxf-=tem;
        f[i]-=tem;
        f[i^1]+=tem;
        cnt+=tem;
        if(!maxf) break;
    }
    if(!cnt) lev[u]=-1;
    return cnt;
}
int Dinic()
{
    int ans=0;
    while(BFS())
        ans+=DFS(S,2147483647);
    return ans;
}
void init(int SS,int TT)
{
    memset(fir,0,sizeof(fir));
    ed=1;
    S=SS; T=TT;
    return;
}
```

(2) Minimum Cost Maximum Flow

```
const int INF=0x7f7f7f7f;
int fir[MAXN],cur[MAXN],lev[MAXN],end[MAXM],next[MAXM],f[MAXM],mono[MAXM],ed,S,T;
int pre[MAXN];
bool exist[MAXN];
void init()
{
    memset(fir,0,sizeof(fir));
```

```
    ed=1; S=MAXN-2; T=MAXN-1;
    return;
}
void addedge(int u,int v,int cap,int val)
{
    end[++ed]=v;
    next[ed]=fir[u];
    fir[u]=ed;
    f[ed]=cap;
    mono[ed]=val;
    end[++ed]=u;
    next[ed]=fir[v];
    fir[v]=ed;
    f[ed]=0;
    mono[ed]=-1*val;
    return;
}
bool BFS()
{
    int u;
    queue<int>q;
    memset(exist,false,sizeof(exist));
    memset(lev,127,sizeof(lev));
    lev[S]=pre[S]=0;
    q.push(S);
    while(q.size())
    {
        u=q.front(); q.pop();
        exist[u]=false;
        for(int i=fir[u];i;i=next[i]) if(f[i]&&lev[u]+mono[i]<lev[end[i]])
        {
            lev[end[i]]=lev[u]+mono[i];
            pre[end[i]]=i;
            if(!exist[end[i]])
            {
                exist[end[i]]=true;
                q.push(end[i]);
            }
        }
    }
    memcpy(cur,fir,sizeof(fir));
    return lev[T]!=INF;
}
int DFS(int u,int maxf)
{
    if(u==T||!maxf) return maxf;
    exist[u]=true;
    int cnt=0;
```

```
    for(int &i=cur[u],tem;i;i=next[i]) if(f[i]&&lev[u]+mono[i]==lev[end[i]])
    {
        if(exist[end[i]]) continue;
        tem=DFS(end[i],min(f[i],maxf));
        maxf-=tem;
        f[i]-=tem;
        f[i^1]+=tem;
        cnt+=tem;
        if(!maxf) break;
    }
    if(!cnt) lev[u]=-1*INF;
    exist[u]=false;
    return cnt;
}
int Augment()
{
    int delta=INF;
    for(int i=pre[T];i;i=pre[end[i^1]])
        if(f[i]<delta)
            delta=f[i];
    for(int i=pre[T];i;i=pre[end[i^1]])
    {
        f[i]-=delta;
        f[i^1]+=delta;
    }
    return delta*lev[T];
}
int MCMF()
{
    int ans=0;
    memset(exist,false,sizeof(exist));
    while(BFS())
        //ans+=DFS(S,INF)*lev[T];
        ans+=Augment();
    return ans;
}
```

(3) With Upper&Lower Bounds

无源汇有上下界最大流

以前写的最大流默认的下界为 0，而这里的下界却不为 0，所以我们要进行再构造让每条边的下界为 0，这样做是为了方便处理。对于每根管子有一个上界容量 up 和一个下界容量 low，我们让这根管子的容量下界变为 0，上界为 up-low。可是这样做了的话流量就不守恒了，为了再次满足流量守恒，即每个节点"入流=出流"，我们增设一个超级源点 st 和一个超级终点 sd。我们开设一个数组 du[]来记录每个节点的流量情况。
du[i]=in[i]（i 节点所有入流下界之和）-out[i]（i 节点所有出流下界之和）。
当 du[i]大于 0 的时候，st 到 i 连一条流量为 du[i]的边。

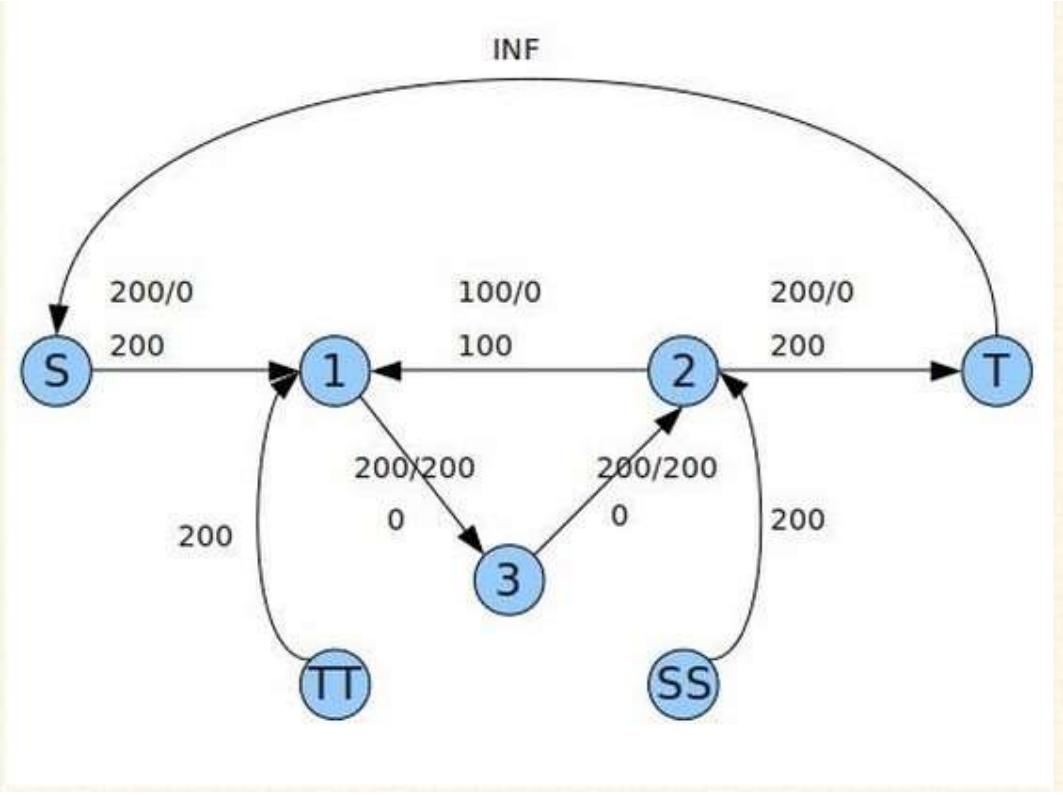
当 du[i]小于 0 的时候，i 到 sd 连一条流量为-du[i]的边。
最后对（st，sd）求一次最大流即可，当所有附加边全部满流时（即 maxflow==所有 du[]>0 之和），有可行解。

有源汇有上下界的最大流

源点 s，终点 d。超级源点 ss，超级终点 dd。首先判断是否存在满足所有边上下界的可行流，方法可以转化成无源汇有上下界的可行流问题。怎么转换呢?
增设一条从 d 到 s 没有下界容量为无穷的边，那么原图就变成了一个无源汇的循环流图。接下来的事情一样，超级源点 ss 连 i（du[i]>0），i 连超级汇点（du[i]<0），
对（ss，dd）进行一次最大流，当 maxflow 等于所有(du[]>0)之和时，有可行流，否则没有。
当有可行流时，删除超级源点 ss 和超级终点 dd，再对（s，d）进行一次最大流，此时得到的 maxflow 则为题目的解。
为什么呢？因为第一次 maxflow（）只是求得所有满足下界的流量，而残留网络（s，d）路上还有许多自由流（没有和超级源点和超级汇点连接的边）没有流满，所有最终得到的 maxflow=（第一次流满下界的流+第二次能流通的自由流）。

有源汇有上下界的最小流

建图模型：同样转换成先求无源汇有上下界的可行流，先添加一条 d 到 s 容量为无穷的边，这里求最小流很容易让人产生歧路，为什么呢？当所有边满足下界条件并且能量守恒时，这时候求得的最大流不就是最小流么。这样是错误了，我开始了在这揣测了良久。
下面来看个例子:



这样求得的最小流为 200，而实际的可行最小流解只需 100。
问题出在原图中存在环（循环流），而我们没有利用，导致流增大了。
解决方法：先不增加 d->s 容量为无穷的边，进行一次 maxflow（），如果还没有满流，则加一条（d，s）容量为无穷的边，再进行一次 maxflow（），当且仅当所有附加弧满载时，有可行解，解为 flow[（d->s）^1]（即 d 到 s 的后悔边权值）。

2. Link-Cut Tree

```
const int MAXN=10050;
int fa[MAXN],son[MAXN][2],st[MAXN];
bool rev[MAXN];
bool isroot(int x)
{// Check whether x is the root of the Auxiliary tree it belongs to
    return x!=son[fa[x]][0]&&x!=son[fa[x]][1];
}
void Down(int x)
{
    if(!rev[x]) return;
    swap(son[son[x][0]][0],son[son[x][0]][1]);
    swap(son[son[x][1]][0],son[son[x][1]][1]);
    rev[son[x][0]]^=true;
    rev[son[x][1]]^=true;
    rev[x]=false;
    return;
}
void adjust(int x)
{
    int fa1=fa[x],fa2=fa[fa[x]],d;
    d=son[fa1][1]==x;
    if(!isroot(fa1))
        son[fa2][son[fa2][1]==fa1]=x;
    fa[x]=fa2;
    fa[fa1]=x;
    fa[son[x][d^1]]=fa1;
    son[fa1][d]=son[x][d^1];
    son[x][d^1]=fa1;
    return;
}
void Splay(int x)
{
    st[0]=0;
    st[++st[0]]=x;
    for(int i=x;!isroot(i);i=fa[i])
        st[++st[0]]=fa[i];
    while(st[0]) Down(st[st[0]--]);
    int fa1,fa2;
    while(!isroot(x))
    {
        fa1=fa[x];
        fa2=fa[fa[x]];
        if(!isroot(fa1))
        {
            if((son[fa1][1]==x)==(son[fa2][1]==fa1))
```

```
46.                 adjust(fa1);
47.             else adjust(x);
48.         }
49.         adjust(x);
50.     }
51.     return;
52. }
53. void Access(int x)
54. {
55.     int t=0;
56.     while(x)
57.     {
58.         Splay(x);
59.         son[x][1]=t;
60.         t=x;
61.         x=fa[x];
62.     }
63.     return;
64. }
65. void Change_to_root(int x)
66. {
67.     Access(x); Splay(x);
68.     swap(son[x][0],son[x][1]);
69.     rev[x]^=true;
70.     return;
71. }
72. void Link(int x,int y)
73. {
74.     Change_to_root(y);
75.     fa[y]=x;
76.     //Splay(y);
77.     return;
78. }
79. void Cut(int x,int y)
80. {
81.     Change_to_root(x);
82.     Access(y);//
83.     Splay(y);
84.     son[y][0]=fa[x]=0;
85.     return;
86. }
87. int Find_root(int x)
88. {
89.     Access(x); Splay(x);
90.     while(son[x][0]) x=son[x][0];
91.     return x;
92. }
93. void read1n(int &x)
```

```
{
    char ch=getchar();
    while(ch<'0'||'9'<ch) ch=getchar();
    for(x=0;'0'<=ch&&ch<='9';ch=getchar())
        x=(x<<1)+(x<<3)+ch-'0';
    return;
}
```

V. Computational Geometry

```
const double eps=1e-10;
const double Pi=acos(-1.);
int dcmp(const double &x) { if(fabs(x)<eps) return 0; return x<0?-1:1; }
typedef struct Poi Vec;
struct Poi
{
    double x,y;
    Poi(const double &a=0,const double &b=0):x(a),y(b) { }
    Poi operator+(const Poi &P)const { return Poi(x+P.x,y+P.y); }
    Poi operator-(const Poi &P)const { return Poi(x-P.x,y-P.y); }
    Poi operator*(const double &P)const { return Poi(x*P,y*P); }
    Poi operator/(const double &P)const { return Poi(x/P,y/P); }
    bool operator<(const Poi &P)const { return !dcmp(x-P.x)?dcmp(y-P.y)<=0:dcmp(x-P.x)<=0; }
};
struct Line
{
    Poi Ps; Vec Dir;
    double ang;
    Line(const Poi &ps=Poi(),const Vec &dir=Vec()):Ps(ps),Dir(dir)
    { ang=atan2(Dir.y,Dir.x); }
    bool operator<(const Line &L1)const { return ang<L1.ang; }
};
struct Circle
{
    Poi Pc;
    double r;
    Circle(const Poi &P=Poi(),const double &R=0):Pc(P),r(R) { }
    Poi loc(double ang,double offset) { return Poi(Pc.x+cos(ang)*(r+offset),Pc.y+sin(ang)*(r+offset))
; }
};
double dOt(const Vec &V1,const Vec &V2) { return V1.x*V2.x+V1.y*V2.y; }
double cRoss(const Vec &V1,const Vec &V2) { return V1.x*V2.y-V1.y*V2.x; }
double lEnth(const Vec V) { return sqrt(dOt(V,V)); }
Vec rOtate(const Vec &V,const double &P)
{ double cOs=cos(P),sIn=sin(P); return Vec(V.x*cOs-V.y*sIn,V.x*sIn+V.y*cOs); }
double aNgle(const Vec &V1,const Vec &V2) { return acos(dOt(V1,V2)/lEnth(V1)/lEnth(V2)); }
Poi iNtersect(const Poi &P1,const Vec &V1,const Poi &P2,const Vec &V2)
```

```
37. { return P1+V1*(cRoss(V2,P1-P2)/cRoss(V1,V2)); }
38. Poi iNtersect(const Line &L1,const Line &L2)
39. { return iNtersect(L1.Ps,L1.Dir,L2.Ps,L2.Dir); }
40. int HPI(Line *L,int N,Poi *Pol)
41. {
42.     int l,r,m;
43.     Poi IP[N+50]; Line q[N+50];
44.     sort(L,L+N);//按极角排序
45.     q[l=r=0]=L[0];
46.     for(int i=1;i<N;i++)
47.     {
48.         while(l<r&&dcmp(cRoss(L[i].Dir,IP[r-1]-L[i].Ps))<=0) r--; //新加入的直线可能是尾部的一些交点失
    效
49.         while(l<r&&dcmp(cRoss(L[i].Dir,IP[l]-L[i].Ps))<=0) l++; //首部
50.         q[++r]=L[i]; //加入
51.         if(!dcmp(cRoss(q[r].Dir,q[r-1].Dir)))
52.         { //对于平行直线要取靠左的
53.             r--;
54.             if(dcmp(cRoss(q[r+1].Dir,q[r].Ps-q[r+1].Ps))<0) q[r]=q[r+1];
55.         }
56.         if(l<r) IP[r-1]=iNtersect(q[r-1],q[r]); //如果队列中有至少两条线，则取交点
57.     }
58.     while(l<r&&dcmp(cRoss(q[l].Dir,IP[r-1]-q[l].Ps))<=0) r--; //后面一些交点可能实际上是无用的
59.     if(r-l<2) return 0; //如果只有不到两条线，则失败了
60.     IP[r]=iNtersect(q[l],q[r]); //将最后一条线和第一条线交起来
61.     for(int i=l;i<=r;i++) Pol[m++]=IP[i];
62.     return m;
63. }
64. int Andrew(Poi *A,int N,Poi *B)
65. {
66.     int m=0;
67.     sort(A,A+N);
68.     for(int i=0;i<N;B[m++]=A[i++])
69.         while(m>1&&cRoss(B[m-1]-B[m-2],A[i]-B[m-2])<=0) m--;
70.     for(int i=N-2,k=m;i>=0;B[m++]=A[i--])
71.         while(m>k&&cRoss(B[m-1]-B[m-2],A[i]-B[m-2])<=0) m--;
72.     if(N>1) m--;
73.     B[m]=B[0];
74.     return m;
75. }
76. int ShamoS(int N,Poi *P)
77. {
78.     int r=1,ans=0,d;
79.     if(N==1) return 0;
80.     else if(N==2) return sQr(P[0]-P[1]);
81.     for(int l=0;l<N;r=(r+1)%N)
82.     {
83.         d=dcmp(cRoss(P[l+1]-P[l],P[r+1]-P[r]));
```

```
        if(d>0) continue;
        ans=max(ans,(int)sQr(P[r--]-P[l++]));
        if(!d) ans=max(ans,(int)sQr(P[r+1]-P[l]));
    }
    return ans;
}
bool onseg(const Poi &A,const Poi &B,const Poi &P)
{
    Vec PA=A-P,PB=B-P;
    if(!dcmp(cRoss(PA,PB))&&dcmp(dOt(PA,PB))<=0) return true;
    return false;
}
bool segI(const Poi &A,const Poi &B,const Poi &C,const Poi &D)
{
    if(onseg(A,B,C) ||onseg(A,B,D) ||onseg(C,D,A) ||onseg(C,D,B)) return true;
    if((dcmp(cRoss(B-C,D-C))^dcmp(cRoss(A-C,D-C)))==-2
            &&(dcmp(cRoss(D-B,A-B))^dcmp(cRoss(C-B,A-B)))==-2) return true;
     return false;
}
```

VI. Divide&Conquer Algorithm

1. Danqi Chen's Algorithm

```
#include<iostream>
#include<cstdio>
#include<cstdlib>
#include<algorithm>
#include<cstring>
#define inf 0x7fffffff
#define ll long long
using namespace std;
inline int read()
{
    int x=0,f=1;char ch=getchar();
    while(ch>'9'||ch<'0'){if(ch=='-')f=-1;ch=getchar();}
    while(ch>='0'&&ch<='9'){x=x*10+ch-'0';ch=getchar();}
    return x*f;
}
int n,m,t[200005],ans[100005];
struct data{int a,b,c,s,ans;}a[100005],p[100005];
inline bool cmp(data a,data b)
{
    if(a.a==b.a&&a.b==b.b)return a.c<b.c;
```

```
    if(a.a==b.a)return a.b<b.b;
    return a.a<b.a;
}
inline bool operator<(data a,data b)
{
    if(a.b==b.b)return a.c<b.c;
    return a.b<b.b;
}
inline int lowbit(int x){return x&(-x);}
inline void update(int x,int num)
{
    for(int i=x;i<=m;i+=lowbit(i))
        t[i]+=num;
}
inline int ask(int x)
{
    int tmp=0;
    for(int i=x;i;i-=lowbit(i))
        tmp+=t[i];
    return tmp;
}
void solve(int l,int r)
{
    if(l==r)return;
    int mid=(l+r)>>1;
    solve(l,mid);solve(mid+1,r);
    sort(p+l,p+mid+1);
    sort(p+mid+1,p+r+1);
    int i=l,j=mid+1;
    while(j<=r)
    {
        while(i<=mid&&p[i].b<=p[j].b)
        {
            update(p[i].c,p[i].s);
            i++;
        }
        p[j].ans+=ask(p[j].c);
        j++;
    }
    for(int j=l;j<i;j++)update(p[j].c,-p[j].s);
}
int main()
{
    int N=read();m=read();
    for(int i=1;i<=N;i++)
        a[i].a=read(),a[i].b=read(),a[i].c=read();
    sort(a+1,a+N+1,cmp);
    int cnt=0;
```

```
    for(int i=1;i<=N;i++)
    {
        cnt++;
        if(a[i].a!=a[i+1].a||a[i].b!=a[i+1].b||a[i].c!=a[i+1].c)
        {
            p[++n]=a[i];
            p[n].s=cnt;
            cnt=0;
        }
    }
    solve(1,n);
    for(int i=1;i<=n;i++)
        ans[p[i].ans+p[i].s-1]+=p[i].s;
    for(int i=0;i<N;i++)
        printf("%d\n",ans[i]);
    return 0;
}
```

2. D&C Algorithm on Tree

(1) Divide by Chains

(2) Divide by Points

VII. Partition

1. Block Array

2. Baby-Step-Giant-Step

VIII.Other Algorithm

1. 奇技淫巧

```
inline int mul_mod0(int a,int b)
{
    int ret;
    __asm__ __volatile__("\tmull %%ebx\n\tdivl %%ecx\n":"=d"(ret):"a"(a),"b"(b),"c"(mod));
    return ret;
}
```

```
inline long long mul_mod1(long long a,long long b)
{
    long long d=(long long)floor(a*(double)b/mod+0.5);
    long long ans=a*b-d*mod;
    if(ans<0) ans+=mod;
    return ans;
}
inline long long mul_mod2(long long a,long long b)
{
    //https://blog.csdn.net/xuxiayang/article/details/81623511
    a%=mod; b%=mod;
    long double c=a*b/mod;
    long long ans=a*b-c*mod;
    if(ans<0) ans+=mod;
    if(mod<=ans) ans-=mod;
    return ans;
}
void calc_inv()
{
    inv[1]=1;
    for(int i=2;i<MAXN;i++)
    {
        inv[i]=mul_mod(mod-mod/i,inv[mod%i],mod);
        cerr<<mul_mod(inv[i],i,mod)<<endl;
    }
    return;
}
for(int i=S;i;i=(i-1)&S)
{//枚举 S 的子集
    ...
}
for(int S=(1<<k)-1;s<1<<n;)
{//枚举一个大小为 n 的集合大小为 k 的子集
    ...
    int x=S&-S;
    int y=S+x;
    S=((S&~y)/x>>1)|y;
}
```

2. Min25 筛

```
using namespace std;
#define pb push_back
#define mp make_pair
typedef pair<int,int> pii;
typedef long long ll;
typedef double ld;
```

```
typedef vector<int> vi;
#define fi first
#define se second
#define fe first
#define FO(x) {freopen(#x".in","r",stdin);freopen(#x".out","w",stdout);}
#define Edg int M=0,fst[SZ],vb[SZ],nxt[SZ];void ad_de(int a,int b){++M;nxt[M]=fst[a];fst[a]=M;vb[M]=b;}void adde(int a,int b){ad_de(a,b);ad_de(b,a);}
#define Edgc int M=0,fst[SZ],vb[SZ],nxt[SZ],vc[SZ];void ad_de(int a,int b,int c){++M;nxt[M]=fst[a];fst[a]=M;vb[M]=b;vc[M]=c;}void adde(int a,int b,int c){ad_de(a,b,c);ad_de(b,a,c);}
#define es(x,e) (int e=fst[x];e;e=nxt[e])
#define esb(x,e,b) (int e=fst[x],b=vb[e];e;e=nxt[e],b=vb[e])
typedef unsigned us;
typedef unsigned long long ull;
const us MOD=1e9+7;
#define SS 2333333
ll n,c0[SS],c1[SS],b0[SS],b1[SS]; int S,ps[SS/10],pn=0;
inline ull F(ull x,us g)
{
    if(x<=1||ps[g]>x) return 0;
    ull ans=((x>S)?b1[n/x]:c1[x])-c1[ps[g-1]]+MOD; //注意这里原来有 bug
    for(us j=g;j<=pn&&ps[j]*(ll)ps[j]<=x;++j)
    {
        ull cn=x/ps[j],ce=ps[j]*(ll)ps[j];
        for(us e=1;cn>=ps[j];++e,cn/=ps[j],ce*=ps[j])
            ans+=F(cn,j+1)*(ps[j]^e)+(ps[j]^(e+1)),ans%=MOD;
    }
    return ans%MOD;
}
int main()
{
    cin>>n; S=sqrtl(n);
    for(int i=1;i<=S;++i)
    {
        ll t=(n/i)%MOD; b0[i]=t;
        b1[i]=t*(t+1)/2%MOD; c0[i]=i;
        c1[i]=i*(ll)(i+1)/2%MOD;
    }
    for(int i=2;i<=S;++i)
    {
        if(c0[i]==c0[i-1]) continue; //not a prime
        ll x0=c0[i-1],x1=c1[i-1],r=(ll)i*i; ps[++pn]=i;
        int u=min((ll)S,n/(i*(ll)i)),uu=min(u,S/i);
        for(int j=1;j<=uu;++j)
            b1[j]-=(b1[j*i]-x1)*i,
            b0[j]-=b0[j*i]-x0;
        ll t=n/i;
        if(t<=2147483647)
        {
```

```
        int tt=t;
        for(int j=uu+1;j<=u;++j)
            b1[j]-=(c1[tt/j]-x1)*i,
            b0[j]-=c0[tt/j]-x0;
        }
        else
        {
        for(int j=uu+1;j<=u;++j)
            b1[j]-=(c1[t/j]-x1)*i,
            b0[j]-=c0[t/j]-x0;
        }
        for(int j=S;j>=r;--j)
            c1[j]-=(c1[j/i]-x1)*i,
            c0[j]-=c0[j/i]-x0;
    }
    for(int i=1;i<=S;++i)
    {
        c1[i]-=c0[i];
        b1[i]-=b0[i];
        if(i>=2) c1[i]+=2;
        if(n>=2LL*i) b1[i]+=2;
        c1[i]=(c1[i]%MOD+MOD)%MOD;
        b1[i]=(b1[i]%MOD+MOD)%MOD;
    }
    ps[pn+1]=S+1;
    ll ans=1+F(n,1);
    ans=(ans%MOD+MOD)%MOD;
    printf("%d\n",int(ans));
}
```