神秘模板库

Toy ASM Truck

Huazhong University of Science and Technology

November 10, 2021

Contents

一切的开始

随机数生成

```cpp
mt19937 rng(chrono::steady_clock::now().time_since_epoch().count());//init
//神秘的高级版本
seed_seq seq{
    (uint64_t) chrono::duration_cast<chrono::nanoseconds>(chrono::high_resolution_clock::now().time_since_epoch()).count(),
    (uint64_t) __builtin_ia32_rdtsc(),
    (uint64_t) (uintptr_t) make_unique<char>().get()
};
mt19937 rng(seq);
//64bit
mt19937_64 rng(seq);
//usage
shuffle(permutation.begin(), permutation.end(), rng);//using shuffle
uniform_int_distribution<int>(0, i)(rng);//get random number [0,i]
```

哈希相关

- 哈希表

```cpp
#include <ext/pb_ds/assoc_container.hpp>
using namespace __gnu_pbds;
gp_hash_table<int, int> table;
```

- 哈希函数

```cpp
struct custom_hash {
    static uint64_t splitmix64(uint64_t x) {
        // http://xorshift.di.unimi.it/splitmix64.c
        x += 0x9e3779b97f4a7c15;
        x = (x ^ (x >> 30)) * 0xbf58476d1ce4e5b9;
        x = (x ^ (x >> 27)) * 0x94d049bb133111eb;
        return x ^ (x >> 31);
    }
    size_t operator()(uint64_t x) const {
        static const uint64_t FIXED_RANDOM = chrono::steady_clock::now().time_since_epoch().count();
        return splitmix64(x + FIXED_RANDOM);
    }
};
//inverse of splitmix
uint64_t splittable64_r(uint64_t x){
    x ^= x >> 31 ^ x >> 62;
    x *= 0x319642b2d24d8ec3U;
    x ^= x >> 27 ^ x >> 54;
    x *= 0x96de1b173f119089U;
    x ^= x >> 30 ^ x >> 60;
    return x;
}
//better hash map
unordered_map<long long, int, custom_hash> safe_map;
gp_hash_table<long long, int, custom_hash> safe_hash_table;
```

Ordered set

```cpp
#include <ext/pb_ds/assoc_container.hpp> // Common file
#include <ext/pb_ds/tree_policy.hpp> // Including tree_order_statistics_node_update
typedef __gnu_pbds::tree<int,null_type,less<int>,rb_tree_tag,tree_order_statistics_node_update> ordered_set;
cout<<*X.find_by_order(1)<<endl;
cout<<X.order_of_key(-5)<<endl;
```

没什么用的卡常

```cpp
#pragma GCC optimize("Ofast,no-stack-protector,unroll-loops,fast-math")
#pragma GCC target("sse,sse2,sse3,ssse3,sse4.1,sse4.2,avx,avx2,popcnt,tune=native")
#include <immintrin.h>//封装好的指令集函数
#include <emmintrin.h>
__m256i _mm256_set_epi32 (int e7, int e6, int e5, int e4, int e3, int e2, int e1, int e0);//将 8 个 int 扔进 256 里面
```

```
__m256i _mm256_set_epi64x (__int64 e3, __int64 e2, __int64 e1, __int64 e0);//int64 的版本
__m256i _mm256_set1_epi32 (int a);//放 8 个 a
__m256i _mm256_add_epi32 (__m256i a, __m256i b);//加法
__m256i _mm256_cmpeq_epi32 (__m256i a, __m256i b);//相等，对应位置为 0xffffffff
__m256i _mm256_cmpgt_epi32 (__m256i a, __m256i b);//大于
__m256i _mm256_and_si256 (__m256i a, __m256i b);//按位与
```

数据结构

ST 表

- 二维

```
int f[maxn][maxn][10][10];
inline int highbit(int x) { return 31 - __builtin_clz(x); }
inline int calc(int x, int y, int xx, int yy, int p, int q) {
    return max(
        max(f[x][y][p][q], f[xx - (1 << p) + 1][yy - (1 << q) + 1][p][q]),
        max(f[xx - (1 << p) + 1][y][p][q], f[x][yy - (1 << q) + 1][p][q])
    );
}
void init() {
    FOR (x, 0, highbit(n) + 1)
    FOR (y, 0, highbit(m) + 1)
        FOR (i, 0, n - (1 << x) + 1)
        FOR (j, 0, m - (1 << y) + 1) {
            if (!x && !y) { f[i][j][x][y] = a[i][j]; continue; }
            f[i][j][x][y] = calc(
                i, j,
                i + (1 << x) - 1, j + (1 << y) - 1,
                max(x - 1, 0), max(y - 1, 0)
            );
        }
}
inline int get_max(int x, int y, int xx, int yy) {
    return calc(x, y, xx, yy, highbit(xx - x + 1), highbit(yy - y + 1));
}
```

树状数组区间修改区间查询

```
struct seg{
    int n;
    vector<ll> c1,c2;
    seg(int n):n(n),c1(n+1),c2(n+1){}
    void add(int x,ll y){for(int i=x;i<=n;i+=i&(-i)) c1[i]+=y,c2[i]+=x*y;}
    void add(int l,int r,ll x){add(l,x);add(r+1,-x);}
    ll sum(int x){
        long long ans(0);
        for(int i=x;i;i-=i&(-i)) ans+=(x+1)*c1[i]-c2[i];
        return ans;
    }
    ll query(int l,int r){return sum(r)-sum(l-1);}
};
```

树剖 &LCA

```
vector<int> fa(n+1),dfn(n+1),top(n+1),sz(n+1),son(n+1),ans(n+1),dep(n+1);
function<void(int,int)> dfs1=[&](int x,int f){
    fa[x]=f;sz[x]=1;
    for(auto v:G[x]){
        if(v==f) continue;
        dep[v]=dep[x]+1;
        dfs1(v,x);
        sz[x]+=sz[v];
        if(sz[v]>sz[son[x]]) son[x]=v;
    }
};
function<void(int,int)> dfs2=[&](int x,int tp){
```

```
    top[x]=tp;
    if(son[x]) dfs2(son[x],tp);
    for(auto v:G[x]){
        if(v==son[x]||v==fa[x]) continue;
        dfs2(v,v);
    }
};
function<int(int x,int y)> lca=[&](int u,int v){
    for(;top[u]^top[v];u=fa[top[u]])
        if(dep[top[u]]<dep[top[v]]) swap(u,v);
    return dep[u]<dep[v] ? u:v;
};
```

无旋 Treap

```
struct Treap{
    Treap *l,*r;
    int fix,key,size;
    Treap(int _key){fix=rand();key=_key;size=1;l=r=NULL;}
    inline void update(){
        size=(l? l->size:0)+(r? r->size:0)+1;
    }
}*root;
typedef pair<Treap*,Treap*> Droot;
inline int size(Treap *x){return x?x->size:0;}
inline int key(Treap *x){return x?x->key:inf;}
Treap* merge(Treap *a,Treap *b){
    if(!a) return b;
    if(!b) return a;
    if(a->fix<b->fix){
        a->r=merge(a->r,b);
        a->update();
        return a;
    }else{
        b->l=merge(a,b->l);
        b->update();
        return b;
    }
}
Droot split(Treap *x,int k){
    if(!x) return Droot(NULL,NULL);
    Droot y;
    if(size(x->l)>=k){
        y=split(x->l,k);
        x->l=y.sc;
        x->update();
        y.sc=x;
    }else{
        y=split(x->r,k-size(x->l)-1);
        x->r=y.fi;
        x->update();
        y.fi=x;
    }
    return y;
}
Treap* build(int *a,int n){
    static Treap *stack[N],*x,*last;
    int p=0;
    rep(i,1,n){
        x=new Treap(a[i]);
        while(p>0&&x->fix<stack[p]->fix){
            stack[p]->update();
            last=stack[p];
            stack[p--]=NULL;
        }
        if(p) stack[p]->r=x;
        x->l=last;
        stack[++p]=x;
    }
    while(p) stack[p--]->update();
    return stack[1];
```

```
}
int findkth(int k){
    Droot x=split(root,k-1);
    Droot y=split(x.sc,1);
    Treap *ans=y.fi;
    root=merge(merge(x.fi,ans),y.sc);
    return key(ans);
}
int getkth(Treap *x,int v){
    if(!x) return 0;
    return v<=x->key ? getkth(x->l,v):getkth(x->r,v)+size(x->l)+1;
}
void insert(int v){
    int k=getkth(root,v);
    Droot x=split(root,k);
    Treap *tmp=new Treap(v);
    root=merge(merge(x.fi,tmp),x.sc);
}
void del(int k){
    Droot x=split(root,k);
    Droot y=split(x.sc,1);
    root=merge(x.fi,y.sc);
}
int n;
int main(){
    n=read();
    while(n --> 0){
        int opt=read();
        if(opt==1){
            insert(read());//insert x
        }else if(opt==2){
            del(getkth(root,read()));//delete x
        }else if(opt==3){
            printf("%d\n",getkth(root,read())+1);//query rank
        }else if(opt==4){
            printf("%d\n",findkth(read()));//query kth
        }else if(opt==5){
            printf("%d\n",findkth(getkth(root,read())));//pre
        }else if(opt==6){
            printf("%d\n",findkth(getkth(root,read()+1)+1));//suc
        }
    }
    return 0;
}
```

替罪羊树

```
const double al=0.73;
int fa[N],c[N][2],size[N],cnt[N],val[N],rt,sz;
int tmp[N],ind;
inline int get(int x){return c[fa[x]][1]==x;}
inline void rec(int x){
    if(c[x][0]) rec(c[x][0]);
    tmp[++ind]=x;
    if(c[x][1]) rec(c[x][1]);
}
inline int build(int l,int r){
    if(l>r) return 0;
    int mid=(l+r)>>1,cur=tmp[mid];
    c[cur][0]=build(l,mid-1);fa[c[cur][0]]=cur;
    c[cur][1]=build(mid+1,r);fa[c[cur][1]]=cur;
    size[cur]=size[c[cur][0]]+size[c[cur][1]]+cnt[cur];
    return cur;
}
inline void rebuild(int x){
    ind=0;int f=fa[x],d=get(x);
    rec(x);int cur=build(1,ind);
    fa[cur]=f;c[f][d]=cur;
    if(x==rt) rt=cur;
}
inline int check(int x){
```

```
    double now=size[x]*al;
    return now>=size[c[x][0]]&&now>=size[c[x][1]];
}
inline void insert(int x){
    int cur=rt;
    if(!cur){cur=rt=++sz;cnt[cur]=size[cur]=1;val[cur]=x;return;}
    while(cur){
        ++size[cur];
        if(val[cur]==x){cnt[cur]++;break;}
        int &v=c[cur][val[cur]<x];
        if(!v){v=++sz;fa[v]=cur;val[v]=x;cnt[v]=size[v]=1;cur=v;break;}
        cur=v;
    }
    int id=0;
    for(int i=cur;i;i=fa[i]) if(!check(i)) id=i;
    if(id) rebuild(id);
}
inline int find(int x){
    int cur=rt;
    while(cur){
        if(val[cur]==x) return cur;
        cur=c[cur][x>val[cur]];
    }
    return -1;
}
inline void del(int x){
    int cur=find(x);int k;assert(cur!=-1);
    if(cnt[cur]>1){--size[cur];--cnt[cur];for(int i=fa[cur];i;i=fa[i]) --size[i];return;}
    int pp=0;
    if(c[cur][0]&&c[cur][1]){
        k=c[cur][0];
        while(c[k][1]) k=c[k][1];//pre
        val[cur]=val[k];cnt[cur]=cnt[k];pp=cur;cur=k;
    }
    k=c[cur][0] ? c[cur][0]:c[cur][1];
    c[fa[cur]][get(cur)]=k;fa[k]=fa[cur];
    if(pp){for(int i=fa[cur];i!=pp;i=fa[i]) size[i]-=cnt[cur];for(int i=pp;i;i=fa[i]) --size[i];}
    else for(int i=fa[cur];i;i=fa[i]) --size[i];
    if(cur==rt) rt=k;
}
inline int findrk(int x){
    int cur=rt,ret=0;
    while(cur){
        int lsize=c[cur][0] ? size[c[cur][0]]:0;
        if(val[cur]==x){ret+=lsize+1;break;}
        if(val[cur]<x){ret+=lsize+cnt[cur];cur=c[cur][1];}
        else cur=c[cur][0];
    }
    return ret;
}
inline int findkth(int k){
    int cur=rt,ret=0;
    while(cur){
        int lsize=c[cur][0] ? size[c[cur][0]]:0;
        if(k<=lsize) cur=c[cur][0];
        else if(k<=lsize+cnt[cur]){ret=cur;break;}
        else k-=lsize+cnt[cur],cur=c[cur][1];
    }
    return ret;
}
inline int getpre(int x){
    int cur=rt,ret=0;
    while(cur){
        if(val[cur]>=x) cur=c[cur][0];
        else{ret=cur;cur=c[cur][1];}
    }
    return ret;
}
inline int getsuc(int x){
    int cur=rt,ret=0;
    while(cur){
```

```
        if(val[cur]<=x) cur=c[cur][1];
        else{ret=cur;cur=c[cur][0];}
    }
    return ret;
}
int main(){
    int n=read();
    rep(i,1,n){
        int op=read(),x=read();
        if(op==1) insert(x);
        else if(op==2) del(x);
        else if(op==3) printf("%d\n",findrk(x));
        else if(op==4) printf("%d\n",val[findkth(x)]);
        else if(op==5) printf("%d\n",val[getpre(x)]);
        else printf("%d\n",val[getsuc(x)]);
    }
    return 0;
}
```

Splay

```
struct Splay{
    int c[N][2],fa[N],size[N],cnt[N],val[N],rt,sz;
    inline int get(int x){return x==c[fa[x]][1];}
    inline void upd(int x){
        size[x]=cnt[x];
        size[x]+=size[c[x][0]];size[x]+=size[c[x][1]];
    }
    inline void rotate(int x,int &k){
        int f=fa[x],p=fa[f],d=get(x);
        if(f==k) k=x;else c[p][get(f)]=x;
        fa[x]=p;c[f][d]=c[x][d^1];if(c[x][d^1]) fa[c[x][d^1]]=f;
        fa[f]=x;c[x][d^1]=f;upd(f);upd(x);
    }
    inline void splay(int x,int &k){
        while(x^k){
            int f=fa[x];
            if(f!=k) rotate(get(x)==get(f) ? f:x,k);
            rotate(x,k);
        }
    }
    inline void insert(int cur,int x){
        if(!cur){cur=rt=++sz;val[cur]=x;size[cur]=cnt[cur]=1;return;}
        while(1){
            ++size[cur];
            if(val[cur]==x){++cnt[cur];break;}
            int &v=c[cur][x>val[cur]];
            if(!v){v=++sz;val[v]=x;size[v]=cnt[v]=1;fa[v]=cur;cur=v;break;}
            cur=v;
        }
        splay(cur,rt);
    }
    inline void find(int cur,int x){
        while(cur){
            if(val[cur]==x){splay(cur,rt);return;}
            cur=c[cur][x>val[cur]];
        }
    }
    inline int findrk(int cur,int x){
        int ret=0;
        while(x){
            if(val[cur]==x){ret+=size[c[cur][0]]+1;splay(cur,rt);break;}
            if(val[cur]>x) cur=c[cur][0];
            else ret+=size[c[cur][0]]+cnt[cur],cur=c[cur][1];
        }
        return ret;
    }
    inline int findkth(int cur,int k){
        int ret=0;
        while(cur){
            int lsize=size[c[cur][0]];
```

```cpp
                if(k<=lsize) cur=c[cur][0];
                else if(k<=lsize+cnt[cur]){ret=val[cur];splay(cur,rt);break;}
                else k-=lsize+cnt[cur],cur=c[cur][1];
            }
            return ret;
        }
        inline int getpre(int cur,int x){
            int ret=0;
            while(cur){
                if(val[cur]>=x) cur=c[cur][0];
                else cur=c[ret=cur][1];
            }
            if(ret) return splay(ret,rt),val[ret];
            else return -1;
        }
        inline int getsuc(int cur,int x){
            int ret=0;
            while(cur){
                if(val[cur]<=x) cur=c[cur][1];
                else cur=c[ret=cur][0];
            }
            if(ret) return splay(ret,rt),val[ret];
            else return -1;
        }
        inline int pre(){
            int now=c[rt][0];
            while(c[now][1]) now=c[now][1];splay(now,rt);
            return now;
        }
        inline int suc(){
            int now=c[rt][1];
            while(c[now][0]) now=c[now][0];splay(now,rt);
            return now;
        }
        inline void del(int x){
            find(rt,x);int cur=rt,k=c[cur][0]*c[cur][1];
            if(cnt[cur]>1){--cnt[cur];--size[cur];return;}
            if(!k){rt=c[cur][0]+c[cur][1];return;}
            else{
                int p=pre();splay(p,rt);
                c[p][1]=c[cur][1];
                fa[c[cur][1]]=c[cur][1] ? p:0;
                upd(p);
            }
        }
}T;
int main(){
    int n=read();
    rep(i,1,n){
        int op=read(),x=read();
        if(op==1){
            T.insert(T.rt,x);
        }else if(op==2){
            T.del(x);
        }else if(op==3){
            printf("%d\n",T.findrk(T.rt,x));
        }else if(op==4){
            printf("%d\n",T.findkth(T.rt,x));
        }else if(op==5){
            printf("%d\n",T.getpre(T.rt,x));
        }else{
            printf("%d\n",T.getsuc(T.rt,x));
        }
    }
    return 0;
}
```

数学

模整数类

除法为整除，请乘逆元。

~~~cpp
```cpp
template<int P>
struct moint {
    int x;
    moint():x(0){}
    moint(int n){x=n<0?n%P+P:n%P;}
    moint(ll n) {x=n<0?n%P+P:n%P;}
    int get()const{return (int)x;}
    moint &operator+=(moint b){x+=b.x;if(x>=P)x-=P;return *this;}
    moint &operator-=(moint b){x-=b.x;if(x<0)x+=P;return *this;}
    moint &operator*=(moint b){x=1ll*x*b.x%P;return *this;}
    moint &operator/=(moint b){x=x/b.x;return *this;}
    moint &operator%=(moint b){x=x%b.x;return *this;}
    moint operator+(moint b)const{return moint(*this)+=b;}
    moint operator-(moint b)const{return moint(*this)-=b;}
    moint operator*(moint b)const{return moint(*this)*=b;}
    moint operator/(moint b)const{return moint(*this)/=b;}
    moint operator%(moint b)const{return moint(*this)%=b;}
    moint operator+(int b)const{return moint(*this)+=moint(b);}
    moint operator-(int b)const{return moint(*this)-=moint(b);}
    moint operator*(int b)const{return moint(*this)*=moint(b);}
    moint operator/(int b)const{return moint(*this)/=moint(b);}
    moint operator%(int b)const{return moint(*this)%=moint(b);}
    bool operator==(moint b)const{return x==b.x;}
    bool operator>=(moint b)const{return x>=b.x;}
    bool operator!=(moint b)const{return x!=b.x;}
};
typedef moint<998244353> mint;
~~~

类欧几里得

- $m=\lfloor\frac{an+b}{c}\rfloor$.
- $f(a,b,c,n)=\sum_{i=0}^{n}\lfloor\frac{ai+b}{c}\rfloor$: 当 $a\ge c$ or $b\ge c$ 时，$f(a,b,c,n)=(\frac{a}{c})n(n+1)/2+(\frac{b}{c})(n+1)+f(a \bmod c,b \bmod c,c,n)$；否则 $f(a,b,c,n)=nm-f(c,c-b-1,a,m-1)$。
- $g(a,b,c,n)=\sum_{i=0}^{n}i\lfloor\frac{ai+b}{c}\rfloor$: 当 $a\ge c$ or $b\ge c$ 时，$g(a,b,c,n)=(\frac{a}{c})n(n+1)(2n+1)/6+(\frac{b}{c})n(n+1)/2+g(a \bmod c,b \bmod c,c,n)$；否则 $g(a,b,c,n)=\frac{1}{2}(n(n+1)m-f(c,c-b-1,a,m-1)-h(c,c-b-1,a,m-1))$。
- $h(a,b,c,n)=\sum_{i=0}^{n}\lfloor\frac{ai+b}{c}\rfloor^2$: 当 $a\ge c$ or $b\ge c$ 时，$h(a,b,c,n)=(\frac{a}{c})^2n(n+1)(2n+1)/6+(\frac{b}{c})^2(n+1)+(\frac{a}{c})(\frac{b}{c})n(n+1)+h(a \bmod c,b \bmod c,c,n)+2(\frac{a}{c})g(a \bmod c,b \bmod c,c,n)+2(\frac{b}{c})f(a \bmod c,b \bmod c,c,n)$；否则 $h(a,b,c,n)=nm(m+1)-2g(c,c-b-1,a,m-1)-2f(c,c-b-1,a,m-1)-f(a,b,c,n)$。

```cpp
struct ans{mint f,g,h;};
static ans calc(int a,int b,int c,int n){
    ans ret;
    if(!a){
        ret.f=mint(b/c)*(n+1);
        ret.g=mint(b/c)*n*(n+1)*iv2;
        ret.h=mint(b/c)*(b/c)*(n+1);
        return ret;
    }
    if(a>=c||b>=c){
        ans to=calc(a%c,b%c,c,n);
        ret.f=mint(a/c)*n*(n+1)*iv2+mint(b/c)*(n+1)+to.f;
        ret.g=mint(a/c)*n*(n+1)*(n*2+1)*iv6+mint(b/c)*n*(n+1)*iv2+to.g;
        ret.h=mint(a/c)*(a/c)*n*(n+1)*(n*2+1)*iv6+mint(b/c)*(b/c)*(n+1)+\
            mint(a/c)*(b/c)*n*(n+1)+to.h+mint(a/c)*2*to.g+mint(b/c)*2*to.f;
        return ret;
    }else{
        ll m=(1ll*a*n+b)/c;
        ans to=calc(c,c-b-1,a,m-1);
        ret.f=mint(n*m%P)-to.f;
        ret.g=(mint(m*n%P*(n+1)%P)-to.f-to.h)*iv2;
        ret.h=mint(m*n%P*(m+1)%P)-to.g*2-to.f*2-ret.f;
        return ret;
    }
}
```

Pollard-Rho

```
template<const int test_case>// set 8 usually
struct Pollard_Rho {
    vector<long long> fac;
    long long quick_pow(long long a, long long b, long long mod) {
        long long ans = 1;
        while (b) {
            if (b&1) ans = (__int128)ans*(__int128)a%mod;
            b >>= 1, a = (__int128)a*(__int128)a%mod;
        }
        return ans;
    }
    bool Miller_Rabin(long long n) {// return if n is a prime
        if (n < 3) return n == 2;
        long long a = n-1, b = 0;
        while (a%2 == 0) a /= 2, ++b;
        for (int i = 0, j; i < test_case; i++) {
            long long x = rand()%(n-2)+2, v = quick_pow(x, a, n);
            if (v == 1 || v == n-1) continue;
            for (j = 0; j < b; j++) {
                v = (__int128)v*(__int128)v%n;
                if (v == n-1) break;
            }
            if (j >= b) return false;
        }
        return true;
    }
    long long f(long long x, long long c, long long n) { return ((__int128)x * x + c) % n; }
    long long rho(long long x) {
        long long s = 0, t = 0;
        long long c = (__int128)rand() % (x - 1) + 1;
        int step = 0, goal = 1;
        long long val = 1;
        for (goal = 1;; goal <<= 1, s = t, val = 1) {
            for (step = 1; step <= goal; ++step) {
                t = f(t, c, x);
                val = (__int128)val * abs(t - s) % x;
                if ((step % 127) == 0) {
                    long long d = __gcd(val, x);
                    if (d > 1) return d;
                }
            }
            long long d = __gcd(val, x);
            if (d > 1) return d;
        }
    }
    void find(long long x) {
        if (x == 1) return;
        if (Miller_Rabin(x)) {
            fac.push_back(x);
            return;
        }
        long long p = x;
        while (p >= x) p = rho(x);
        //while ((x % p) == 0) x /= p;
        find(x/p), find(p);
    }
    vector<long long> factor(long long n) {// return the factors of n
        srand((unsigned)time(NULL));
        fac.clear();
        find(n);
        sort(fac.begin(), fac.end());
        return fac;
    }
};
```

ex-gcd

```
template<typename T>
struct ex_gcd {
```

```
    T gcd(const T a, const T b, T &x, T &y) {// x'=x_0+b/gcd, y'=y_0-a/gcd
        if (b == 0) {x = 1, y = 0; return a; }
        T d = gcd(b, a%b, x, y);
        T t = x;
        x = y;
        y = t - a/b*y;
        return d;
    }
    T inv(const T a, const T m) {// return -1 if inv is not exist
        if (a == 0 || m <= 1) return -1;
        T x, y, d = gcd(a, m, x, y);
        if (d != 1) return -1;
        return (x%m+m)%m;
    }
} ;
```

crt

```
template<typename T>
struct crt {
    ex_gcd<T> *exgcd = new ex_gcd<T>();
    T cal(const T *a, const T *m, const int n) {// a[1..n], m[1..n], gcd(m_i) = 1
        T M = 1, ans = 0;
        for (int i = 1; i <= n; i++) M *= m[i];
        for (int i = 1; i <= n; i++)
            (ans += (__int128)a[i]*(M/m[i])%M*exgcd->inv(M/m[i], m[i])%M) %= M;
        return ans;
    }
} ;
```

ex-crt

```
template<typename T>
struct ex_crt {
    ex_gcd<T> *exgcd = new ex_gcd<T>();
    T cal(T *a, T *m, const int n) {// a[1..n], m[1..n], return -1 if no ans
        T x, y, gcd, lcm;
        for (int i = 2; i <= n; i++) {
            gcd = exgcd->gcd(m[1], m[i], x, y);
            if ((a[i]-a[1])%gcd) return -1;
            lcm = (__int128)m[1]*m[i]/gcd;
            x = (__int128)x*(a[i]-a[1])/gcd%lcm;
            gcd = m[i]/gcd;
            x = (x%gcd+gcd)%gcd;
            a[1] = ((__int128)m[1]*x%lcm+a[1])%lcm, m[1] = lcm;
        }
        return a[1];
    }
} ;
```

Meissel-Lehmer

求解 1e11 内的质数个数，约为 $O(n^{2/3})$。

```
namespace pcf{
#define chkbit(ar, i) (((ar[(i) >> 6]) & (1 << (((i) >> 1) & 31))))
#define setbit(ar, i) (((ar[(i) >> 6]) |= (1 << (((i) >> 1) & 31))))
#define isprime(x) (( (x) && ((x)&1) && (!chkbit(ar, (x)))) || ((x) == 2))
    const int MAXN=100;
    const int MAXM=10001;
    const int MAXP=40000;
    const int MAX=400000;
    long long dp[MAXN][MAXM];
    unsigned int ar[(MAX >> 6) + 5] = {0};
    int len = 0, primes[MAXP], counter[MAX];
    void Sieve(){
        setbit(ar, 0), setbit(ar, 1);
        for (int i = 3; (i * i) < MAX; i++, i++){
            if (!chkbit(ar, i)){
```

```
                int k = i << 1;
                for (int j = (i * i); j < MAX; j += k) setbit(ar, j);
            }
        }
        for (int i = 1; i < MAX; i++){
            counter[i] = counter[i - 1];
            if (isprime(i)) primes[len++] = i, counter[i]++;
        }
    }
    void init(){
        Sieve();
        for (int n = 0; n < MAXN; n++){
            for (int m = 0; m < MAXM; m++){
                if (!n) dp[n][m] = m;
                else dp[n][m] = dp[n - 1][m] - dp[n - 1][m / primes[n - 1]];
            }
        }
    }
    long long phi(long long m, int n){
        if (n == 0) return m;
        if (primes[n - 1] >= m) return 1;
        if (m < MAXM && n < MAXN) return dp[n][m];
        return phi(m, n - 1) - phi(m / primes[n - 1], n - 1);
    }
    long long Lehmer(long long m){
        if (m < MAX) return counter[m];
        long long w, res = 0;
        int i, a, s, c, x, y;
        s = sqrt(0.9 + m), y = c = cbrt(0.9 + m);
        a = counter[y], res = phi(m, a) + a - 1;
        for (i = a; primes[i] <= s; i++) res = res - Lehmer(m / primes[i]) + Lehmer(primes[i]) - 1;
        return res;
    }
}
int main(){
    pcf::init();
    long long n;
    while (scanf("%lld", &n) != EOF){
        printf("%lld\n",pcf::Lehmer(n));
    }
    return 0;
}
```

Cipolla 二次剩余

- $x^2 \equiv n (\bmod P)$
- 仅有两个解，返回小的那个，另一个是相反数。
- 大概 1s 能跑 1e5 个数

```
inline int qpow(int a,int b){
    int q=1;while(b){if(b&1)q=1ll*q*a%P;a=1ll*a*a%P;b>>=1;}return q;
}
namespace Cipolla{
    ll w,a;
    struct node{
        ll x,y;
        node friend operator *(node x,node y){
            node z;
            z.x=(x.x*y.x%P+x.y*y.y%P*w%P)%P;
            z.y=(x.x*y.y%P+x.y*y.x%P)%P;
            return z;
        }
    }u,v;
    inline node Cqpow(node a,ll b){
        node q;q.x=1;q.y=0;
        while(b){if(b&1) q=q*a;a=a*a;b>>=1;}
        return q;
    }
    inline ll cipolla(int n){
        n%=P;srand(0x20010412);
```

```
        if(P==2) return n;
        if(!n) return n;
        if(qpow(n,(P-1)/2)==P-1) return -1;
        while(1){
            a=rand()%P;
            w=(a*a-n+P)%P;
            if(qpow(w%P,(P-1)/2)==P-1) break;
        }
        u.x=a,u.y=1;
        u=Cqpow(u,(P+1)/2);
        ll fir=u.x,sec=P-u.x;
        if(fir>sec)swap(fir,sec);
        return fir;
    }
}
```

BSGS

```
template<typename T>
struct BSGS {
    T cal(T a, T b, T c) { // return a^x = b (mod c), gcd(a, c) = 1
        mp.clear();
        T tim = ceil(sqrt(c)), tmp = b%c;
        for (int i = 0; i <= tim; i++) {
            mp[tmp] = i; tmp = (__int128)tmp*a%c;
        }
        T t = tmp = quick_pow(a, tim, c);
        for (int i = 1; i <= tim; i++) {
            if (mp.count(tmp)) return tim*i-mp[tmp];
            tmp = (__int128)tmp*t%c;
        }
        return -1;
    }
} ;
```

exBSGS

```
template<typename T>
struct exBSGS {
    T cal(T a, T b, T c) { // return a^x = b (mod c)
        if (b == 1) return 0;
        T cnt = 0, d = 1, t;
        while ((t = __gcd(a, c)) != 1) {
            if (b%t) return -1;
            ++cnt, b /= t, c /= t, d = (__int128)d*(a/t)%c;
            if (d == b) return cnt;
        }
        mp.clear();
        T tim = ceil(sqrt(c)), tmp = b%c;
        for (int i = 0; i <= tim; i++) {
            mp[tmp] = i; tmp = (__int128)tmp*a%c;
        }
        t = tmp = quick_pow(a, tim, c); tmp = (__int128)tmp*d%c;
        for (int i = 1; i <= tim; i++) {
            if (mp.count(tmp)) return tim*i-mp[tmp]+cnt;
            tmp = (__int128)tmp*t%c;
        }
        return -1;
    }
} ;
```

exLucas

```
template<typename T>
struct exLucas {
    T quick_pow(T a, T b, T p) {
        T ans = 1;
        while (b) {
            if (b&1) ans = (__int128)ans*a%p;
```

```
            b >>= 1, a = (__int128)a*a%p;
        }
        return ans;
    }
    void ex_gcd(T a, T b, T &x, T &y) {
        if (b == 0) {x = 1, y = 0; return; }
        ex_gcd(b, a%b, x, y);
        T t = x; x = y, y = t-a/b*y;
    }
    T inv(T a, T p) {
        T x, y; ex_gcd(a, p, x, y);
        return (x%p+p)%p;
    }
    T mul(T n, T pi, T pk) {
        if (!n) return 1;
        T ans = 1;
        for (int i = 2; i <= pk; i++) if (i%pi != 0) ans = (__int128)ans*i%pk;
        ans = quick_pow(ans, n/pk, pk);
        for (int i = 2; i <= n%pk; i++) if (i%pi != 0) ans = (__int128)ans*i%pk;
        return (__int128)ans*mul(n/pi, pi, pk)%pk;
    }
    T C(T n, T m, T pi, T pk, T p) {
        T a = mul(n, pi, pk), b = mul(m, pi, pk), c = mul(n-m, pi, pk);
        T k = 0;
        for (T i = n; i; i /= pi) k += i/pi;
        for (T i = m; i; i /= pi) k -= i/pi;
        for (T i = n-m; i; i /= pi) k -= i/pi;
        return (__int128)a*inv(b, pk)%pk*inv(c, pk)%pk*quick_pow(pi, k, pk)%pk;
    }
    T ex_lucas(T n, T m, T p) {
        T ans = 0;
        for (T i = 2, x = p; i <= x; i++)
            if (x%i == 0) {
                T k = 1; while (x%i == 0) k *= i, x /= i;
                (ans += (__int128)C(n, m, i, k, p)*(p/k)%p*inv(p/k, k)%p) %= p;
            }
        return ans;
    }
} ;
```

min_25 筛

```
/* 「LOJ #6053」简单的函数 */
#include <algorithm>
#include <cmath>
#include <cstdio>

using i64 = long long;

constexpr int maxs = 200000;  // 2sqrt(n)
constexpr int mod = 1000000007;

template <typename x_t, typename y_t>
inline void inc(x_t &x, const y_t &y) {
  x += y;
  (mod <= x) && (x -= mod);
}
template <typename x_t, typename y_t>
inline void dec(x_t &x, const y_t &y) {
  x -= y;
  (x < 0) && (x += mod);
}
template <typename x_t, typename y_t>
inline int sum(const x_t &x, const y_t &y) {
  return x + y < mod ? x + y : (x + y - mod);
}
template <typename x_t, typename y_t>
inline int sub(const x_t &x, const y_t &y) {
  return x < y ? x - y + mod : (x - y);
}
template <typename _Tp>
```

```
inline int div2(const _Tp &x) {
  return ((x & 1) ? x + mod : x) >> 1;
}
template <typename _Tp>
inline i64 sqrll(const _Tp &x) {
  return (i64)x * x;
}

int pri[maxs / 7], lpf[maxs + 1], spri[maxs + 1], pcnt;

inline void sieve(const int &n) {
  for (int i = 2; i <= n; ++i) {
    if (lpf[i] == 0)
      pri[lpf[i] = ++pcnt] = i, spri[pcnt] = sum(spri[pcnt - 1], i);
    for (int j = 1, v; j <= lpf[i] && (v = i * pri[j]) <= n; ++j) lpf[v] = j;
  }
}

i64 global_n;
int lim;
int le[maxs + 1],  // x \le \sqrt{n}
    ge[maxs + 1];  // x > \sqrt{n}
#define idx(v) (v <= lim ? le[v] : ge[global_n / v])

int G[maxs + 1][2], Fprime[maxs + 1];
i64 lis[maxs + 1];
int cnt;

inline void init(const i64 &n) {
  for (i64 i = 1, j, v; i <= n; i = n / j + 1) {
    j = n / i;
    v = j % mod;
    lis[++cnt] = j;
    idx(j) = cnt;
    G[cnt][0] = sub(v, 1ll);
    G[cnt][1] = div2((i64)(v + 2ll) * (v - 1ll) % mod);
  }
}

inline void calcFprime() {
  for (int k = 1; k <= pcnt; ++k) {
    const int p = pri[k];
    const i64 sqrp = sqrll(p);
    for (int i = 1; lis[i] >= sqrp; ++i) {
      const i64 v = lis[i] / p;
      const int id = idx(v);
      dec(G[i][0], sub(G[id][0], k - 1));
      dec(G[i][1], (i64)p * sub(G[id][1], spri[k - 1]) % mod);
    }
  }
  /* F_prime = G_1 - G_0 */
  for (int i = 1; i <= cnt; ++i) Fprime[i] = sub(G[i][1], G[i][0]);
}

inline int f_p(const int &p, const int &c) {
  /* f(p^{c}) = p xor c */
  return p xor c;
}

int F(const int &k, const i64 &n) {
  if (n < pri[k] || n <= 1) return 0;
  const int id = idx(n);
  i64 ans = Fprime[id] - (spri[k - 1] - (k - 1));
  if (k == 1) ans += 2;
  for (int i = k; i <= pcnt && sqrll(pri[i]) <= n; ++i) {
    i64 pw = pri[i], pw2 = sqrll(pw);
    for (int c = 1; pw2 <= n; ++c, pw = pw2, pw2 *= pri[i])
      ans +=
          ((i64)f_p(pri[i], c) * F(i + 1, n / pw) + f_p(pri[i], c + 1)) % mod;
  }
  return ans % mod;
```

```
}

int main() {
  scanf("%lld", &global_n);
  lim = sqrt(global_n);

  sieve(lim + 1000);
  init(global_n);
  calcFprime();
  printf("%lld\n", (F(1, global_n) + 1ll + mod) % mod);

  return 0;
}
```

超现实数

```
#include<bits/stdc++.h>
const int inf=1000;
const int maxn=3e5;
using namespace std;
struct fs{                                         //分数结构体
    int fz,fm;
    fs(int _fz=0,int _fm=1):fz(_fz),fm(_fm){}
    bool operator==(const fs &oth)const{
        return fz*oth.fm==fm*oth.fz;
    }
    friend int ceil(const fs &x){
        return (int)ceil(1.0*x.fz/x.fm);
    }
    friend int floor(const fs &x){
        return (int)floor(1.0*x.fz/x.fm);
    }
    friend fs abs(const fs &x){
        return fs(abs(x.fz),abs(x.fm));
    }
    bool operator<(const fs &oth)const{
        return (double)fz/fm<(double)oth.fz/oth.fm;
        //return fz*oth.fm<fm*oth.fz;      //不能这么写啊   负数的时候不成立，直接用 double 就完事了
    }
    fs operator+(const fs &oth)const{
        int n_fm=fm*oth.fm;
        int n_fz=fz*oth.fm+fm*oth.fz;
        int yf=__gcd(n_fz,n_fm);
        return fs(n_fz/yf,n_fm/yf);
    }
    friend void print(const fs &x){
        printf("%d/%d\n",x.fz,x.fm);
    }
};
int pd(int dex,int n){                           //判断二进制状压后的第 n 位上的棋子类型
    if(dex<(1<<(2*n-2)))return 2;              //二进制的右二位、最右位表示第一枚棋子，以此类推
    int flag1,flag2;
    dex>>=(2*n)-2;
    flag1=dex&1;
    if(flag1)return 0;                         //11 空格
    dex>>=1;
    flag2=dex&1;
    if(flag2)return 1;                        // 10 黑色
    else return 2;                            // 00 白色
}
void solve(int x){                            //逆向解出棋盘状态
    char s[7][7];
    for(int i=1;i<=3;i++){
        if(x&1)s[i][0]='#';
        else if((x>>1)&1)s[i][0]='X';
        else s[i][0]='O';
        s[i][1]='|';
        x>>=2;
        if(x&1)s[i][2]='#';
        else if((x>>1)&1)s[i][2]='X';
        else s[i][2]='O';
```

```
        s[i][3]='|';
        x>>=2;
        if(x&1)s[i][4]='#';
        else if((x>>1)&1)s[i][4]='X';
        else s[i][4]='O';
        s[i][5]=0;
        x>>=2;
    }
    for(int i=1;i<=3;i++)
        printf("%s\n",s[i]);
}
int sw(int dex,int n,int typ){        // 换第 i 个石头变为空格
    int tmp=dex;
    tmp|=3<<(2*n-2);
    if(typ==1 || typ==3){                          //变换左右
        if(n%3!=0){
            tmp|=3<<(2*(n+1)-2);
        }
        if(n%3!=1){
            tmp|=3<<(2*(n-1)-2);
        }
    }
    if(typ==2 || typ==3){                     //变换上下
        if((n+2)/3!=1){
            tmp|=3<<(2*(n-3)-2);
        }
        if((n+2)/3!=3){
            tmp|=3<<(2*(n+3)-2);
        }
    }
    return tmp;
}
fs a[maxn];
void init(){                                 //初始化，将数组中每一个值都赋 inf+2
    for(int i=0;i<maxn;i++){
        a[i].fm=1;
        a[i].fz=inf+2;
    }
}
fs getl(int dex);                          //获取左集合的 S
fs getr(int dex);                          //获取右集合的 S
fs getsr(int dex);                         //获取 { | } 整体的 S
fs calc(fs l,fs r){                        //get 辅助函数  Surreal Number 计算规则
    if (r<l) swap(l,r);
    int x=ceil(l);
    if (l==x) ++x;
    if (fs(x)<r){
        if (fs(0)<l||l==0||l<0&&fs(0)<r) return x;
        int y=floor(r);
        if (fs(y)==r) --y;
        if (r<0||r==0) return y;
        return y;
    }
    for (int y=1;;y*=2){
        for (x=1;;++x){
            fs tmp1=fs(x,y);
            fs tmp2=fs(-x,y);
            if (l<tmp1&&tmp1<r) return tmp1;
            if (l<tmp2&&tmp2<r) return tmp2;
            if (fs(0)<r&&r<tmp1) break;
            if (l<0&&tmp2<l) break;
        }
    }
    return fs(0);
}
fs get(fs l,fs r){                         //计算 S
    fs res=fs(0);
    if(l.fz==-inf-1 && r.fz==inf+1)return 0;   //当左集合和右集合都是空，输出 0
    else if(l.fz==-inf-1){                      //当左集合为空，输出 S（右集合）-1
        res=res+r+fs(-1,1);
    }
```

```
    else if(r.fz==inf+1){                     //当右集合为空，输出 S（左集合）+1
        res=res+l+fs(1,1);
    }
    else{
        res=calc(l,r);                        //调用 calc 计算
    }
    return res;
}
fs getl(int dex){                             //左集合最大值
    fs res=-inf-1;                            //若左集合为空，输出-inf-1
    for(int i=1;i<=9;i++){
        if(pd(dex,i)==2){
            res=max(res,getsr(sw(dex,i,1)));   //Alice 三种选择，取最大值
            res=max(res,getsr(sw(dex,i,2)));
            res=max(res,getsr(sw(dex,i,3)));
        }
    }
    return res;
}
fs getr(int dex){                   //右集合最小值
    fs res=inf+1;                   //若左集合为空，输出 inf+1
    for(int i=1;i<=9;i++){
        if(pd(dex,i)==1){
            res=min(res,getsr(sw(dex,i,0)));
        }
    }
    return res;
}
fs getsr(int dex){                  //获得某个状态的 sunr
    if(a[dex].fz!=inf+2){           //如果已经保存过，则直接取用，如果没有则计算。
        return a[dex];
    }
    else{
        fs m=getl(dex),n=getr(dex);
        a[dex]=get(m,n);
        //printf("getl,getr=\n");
        //print(m);
        //print(n);
        //print(a[dex]);
        return a[dex];
    }
}
int make(){                            //构造映射，将棋盘状态转化为二进制数
    int res=0;
    char s[10];
    int dex=1;
    for(int i=1;i<=3;i++){
        scanf("%s",s);
        for(int j=1;j<=3;j++){
            int tmp=0;
            if(s[2*j-2]=='X')tmp=2;
            else if(s[2*j-2]=='.'||s[2*j-2]=='#')tmp=3;
            res+=(tmp<<(2*dex-2));
            dex++;
        }
    }
    return res;
}
int main(){
    init();
    int ca;
    scanf("%d",&ca);
    while(ca--){
        int n;
        fs res=fs(0);
        scanf("%d",&n);
        for(int i=0;i<n;i++){
            int dex=make();
            res=res+getsr(dex);
        }
        if(fs(0)<res)printf("Alice\n");
```

```
        else if(res<fs(0))printf("Bob\n");
        else printf("Second\n");
    }
    return 0;
}
```

图论

LCA

- 倍增

```
void dfs(int u, int fa) {
    pa[u][0] = fa; dep[u] = dep[fa] + 1;
    FOR (i, 1, SP) pa[u][i] = pa[pa[u][i - 1]][i - 1];
    for (int& v: G[u]) {
        if (v == fa) continue;
        dfs(v, u);
    }
}


int lca(int u, int v) {
    if (dep[u] < dep[v]) swap(u, v);
    int t = dep[u] - dep[v];
    FOR (i, 0, SP) if (t & (1 << i)) u = pa[u][i];
    FORD (i, SP - 1, -1) {
        int uu = pa[u][i], vv = pa[v][i];
        if (uu != vv) { u = uu; v = vv; }
    }
    return u == v ? u : pa[u][0];
}
```

同余最短路

```
//d[i] = k[i]*a[1]+i
//smallest d[i] % a[1] ==i
//O(n*a)
sort(a+1,a+n+1,greater<int>());
rep(i,0,a[1]-1) k[i]=inf;
k[0]=0;
deque<int> q;
q.emplace_front(0);
while(!q.empty()){
    int x=q.front();q.pop_front();
    if(vis[x]) continue;
    vis[x]=1;
    rep(i,2,n){
        if(a[i]+x<a[1]){
            chmin(k[x+a[i]],k[x]);
            q.emplace_front(x+a[i]);
        }else{
            int to=(x+a[i])%a[1];
            chmin(k[to],k[x]+1);
            q.emplace_back(to);
        }
    }
}
```

计算几何

二维几何：点与向量

```
#define y1 yy1
#define nxt(i) ((i + 1) % s.size())
typedef double LD;
const LD PI = 3.14159265358979323846;
```

```cpp
const LD eps = 1E-10;
int sgn(LD x) { return fabs(x) < eps ? 0 : (x > 0 ? 1 : -1); }
struct L;
struct P;
typedef P V;
struct P {
    LD x, y;
    explicit P(LD x = 0, LD y = 0): x(x), y(y) {}
    explicit P(const L& l);
};
struct L {
    P s, t;
    L() {}
    L(P s, P t): s(s), t(t) {}
};

P operator + (const P& a, const P& b) { return P(a.x + b.x, a.y + b.y); }
P operator - (const P& a, const P& b) { return P(a.x - b.x, a.y - b.y); }
P operator * (const P& a, LD k) { return P(a.x * k, a.y * k); }
P operator / (const P& a, LD k) { return P(a.x / k, a.y / k); }
inline bool operator < (const P& a, const P& b) {
    return sgn(a.x - b.x) < 0 || (sgn(a.x - b.x) == 0 && sgn(a.y - b.y) < 0);
}
bool operator == (const P& a, const P& b) { return !sgn(a.x - b.x) && !sgn(a.y - b.y); }
P::P(const L& l) { *this = l.t - l.s; }
ostream &operator << (ostream &os, const P &p) {
    return (os << "(" << p.x << "," << p.y << ")");
}
istream &operator >> (istream &is, P &p) {
    return (is >> p.x >> p.y);
}

LD dist(const P& p) { return sqrt(p.x * p.x + p.y * p.y); }
LD dot(const V& a, const V& b) { return a.x * b.x + a.y * b.y; }
LD det(const V& a, const V& b) { return a.x * b.y - a.y * b.x; }
LD cross(const P& s, const P& t, const P& o = P()) { return det(s - o, t - o); }
// --------------------------------------------
```

完整的板板 by zcs

```cpp
#include <bits/stdc++.h>
using namespace std;
const double pi=acos(-1.0);   //高精度圆周率
const double eps=1e-8;        //偏差值
const int maxp=200005;          //点的数量
int sgn(double x)             //判断 x 是否为 0
{
    if(fabs(x)<eps) return 0;
    else return x<0?-1:1;     //小于 0 返回-1，大于 0 返回 1
}
int Dcmp(double x,double y)   //比较浮点数大小
{
    if(fabs(x-y)<eps) return 0;
    else return x<y?-1:1;
}
//---------------------------------平面几何：点和线-----------------------------------
struct Point
{
    double x,y;
    Point(){}
    Point(double x,double y):x(x),y(y){}
    Point operator + (Point B){return Point(x+B.x,y+B.y);}
    Point operator - (Point B){return Point(x-B.x,y-B.y);}
    Point operator * (double k){return Point(x*k,y*k);}                 //长度扩大 k 倍
    Point operator / (double k){return Point(x/k,y/k);}                 //长度缩小 k 倍
    bool operator == (Point B){return sgn(x-B.x)==0 && sgn(y-B.y)==0;}
};
typedef Point Vector;
double Dot(Vector A,Vector B){return A.x*B.x+A.y*B.y;}                  //向量点乘
double Len(Vector A){return sqrt(Dot(A,A));}                            //向量取模
double Angle(Vector A,Vector B){return acos(Dot(A,B)/Len(A)/Len(B));}   //向量 A 和 B 夹角
```

```
double Cross(Vector A,Vector B){return A.x*B.y-A.y*B.x;}                    //向量叉乘
//double Area(Point A,Point B,Point C){return Cross(B-A,C-A);}             //三角形面积 2 倍
double Distance(Point A,Point B){return hypot(A.x-B.x,A.y-B.y);}            //两点距离
Vector Normal(Vector A){return Vector(-A.y/Len(A),A.x/Len(A));}             //向量 A 的单位 * 法 * 向量
bool Parallel(Vector A,Vector B){return sgn(Cross(A,B))==0;}                //平行
Vector Rotate(Vector A,double rad)                                          //向量旋转
{return Vector(A.x*cos(rad)-A.y*sin(rad),A.x*sin(rad)+A.y*cos(rad));}
struct Line
{
    Point p1,p2;
    Line(){}
    Line(Point p1,Point p2):p1(p1),p2(p2){}                                 //两点确定直线
    Line(Point p,double angle)                                              //点 + 倾斜角
    {
        p1=p;
        if(sgn(angle-pi/2)==0){p2=(p1+Point(0,1));}
        else {p2=(p1+Point(1,tan(angle)));}
    }
    Line(double a,double b,double c)                                        //ax+by+c=0;
    {
        if(sgn(a)==0) {p1=Point(0,-c/b);p2=Point(1,-c/b);}
        else if(sgn(b)==0) {p1=Point(-c/a,0);p2=Point(-c/a,1);}
        else {p1=Point(0,-c/b);p2=Point(1,(-c-a)/b);}
    }
};
typedef Line Segment;
//直线倾斜角, 返回值 [0,pi);
double Line_angle(Line v)
{
    double k=atan2(v.p2.y-v.p1.y,v.p2.x-v.p1.x);
    if(sgn(k)<0) k+=pi;
    if(sgn(k-pi)==0) k-=pi;
        return k;
}
//点和直线关系
int Point_line_relation(Point p,Line v)
{
    int c=sgn(Cross(p-v.p1,v.p2-v.p1));
    if(c<0) return 1;                       //1:p 在 v 左侧
    if(c>0) return 2;                       //2:p 在 v 右侧
    return 0;                               //0:p 在 v 上
}
//点和线段关系: 0 为 p 不在线段 v 上; 1 为 p 在线段 v 上
bool Point_on_seg(Point p,Segment v)
{
    return sgn(Cross(p-v.p1,v.p2-v.p1))==0 && sgn(Dot(p-v.p1,p-v.p2))<=0;
}
//两直线的关系: 0 为平行, 1 为重合, 2 为相交
int Line_relation(Line v1,Line v2)
{
    if(sgn(Cross(v1.p2-v1.p1,v2.p2-v2.p1))==0)
    {
        if(Point_line_relation(v1.p1,v2)==0) return 1;
        else return 0;
    }
    return 2;
}
//点到直线距离
double Dis_point_line(Point p,Line v)
{
    return fabs(Cross(p-v.p1,v.p2-v.p1))/Distance(v.p1,v.p2);
}
//点在直线上的投影
Point Point_line_proj(Point p,Line v)
{
    double k=Dot(v.p2-v.p1,p-v.p1)/Dot(v.p2-v.p1,v.p2-v.p1);
    return v.p1+(v.p2-v.p1)*k;
}
//点 p 对直线 v 的对称点
Point Point_line_symmetry(Point p,Line v)
{
```

```
    Point q=Point_line_proj(p,v);
    return Point(2*q.x-p.x,2*q.y-p.y);
}
//点到线段的距离
double Dis_point_seg(Point p,Segment v)
{
    if(sgn(Dot(p-v.p1,v.p2-v.p1))<0 || sgn(Dot(p-v.p2,v.p1-v.p2))<0)
        return min(Distance(p,v.p1),Distance(p,v.p2));                  //点的投影不在线段上
    return Dis_point_line(p,v);                                         //点的投影在线段上
}
//求两直线 ab 和 cd 的交点，在调用前要保证两直线不平行或重合
Point Cross_point(Point a,Point b,Point c,Point d)
{
    double s1=Cross(b-a,c-a);
    double s2=Cross(b-a,d-a);
    return Point(c.x*s2-d.x*s1,c.y*s2-d.y*s1)/(s2-s1);
}
//线段 ab 和 cd 是否相交
bool Cross_segment(Point a,Point b,Point c,Point d)
{
    double c1=Cross(b-a,c-a),c2=Cross(b-a,d-a);
    double d1=Cross(d-c,a-c),d2=Cross(d-c,b-c);
    return sgn(c1)*sgn(c2)<=0 && sgn(d1)*sgn(d2)<=0;
}
//------------------------------------平面几何：多边形------------------------------------
struct Polygon
{
    int n;
    Point p[maxp];                          //从 0 开始
    Line v[maxp];
};
//极角排序
bool Polar_angle_cmp(Point a,Point b)
{
    if(Cross(a,b)==0) return a.x<b.x;
    else return Cross(a,b)>0;
}
//按照 x 大小排序（计算凸包使用）
bool Hull_cmp(Point A,Point B)
{
    return sgn(A.x-B.x)<0 || (sgn(A.x-B.x)==0 && sgn(A.y-B.y)<0);
}
//判断点和任意多边形的关系：3 为点上；2 为边上；1 为内部；0 为外部
int Point_in_polygon(Point pt,Point *p,int n)                          //点 pt，多边形 *p
{
    for(int i=0;i<n;i++)
        if(p[i]==pt) return 3;
    for(int i=0;i<n;i++)
    {
        Line v=Line(p[i],p[(i+1)%n]);
        if(Point_on_seg(pt,v)) return 2;
    }
    int num=0;
    for(int i=0;i<n;i++)
    {
        int j=(i+1)%n;
        int c=sgn(Cross(pt-p[j],p[i]-p[j]));
        int u=sgn(p[i].y-pt.y);
        int v=sgn(p[j].y-pt.y);
        if(c>0 && u<0 && v>=0) num++;
        if(c<0 && u>=0 && v<0) num--;
    }
    return num!=0;
}
//多边形面积
double Polygon_area(Point *p,int n)
{
    double area=0;
    for(int i=0;i<n;i++)
        area+=Cross(p[i],p[(i+1)%n]);
    return area/2;
```

```
}
//求多边形重心
Point Polygon_center(Point *p,int n)
{
    Point ans(0,0);
    if(Polygon_area(p,n)==0) return ans;
    for(int i=0;i<n;i++)
        ans=ans+(p[i]+p[(i+1)%n])*Cross(p[i],p[(i+1)%n]);
    return ans/Polygon_area(p,n)/6;
}
//Convex_hull() 求凸包，凸包顶点放在 ch 中，返回值是凸包的顶点数
int Convex_hull(Point *p,int n,Point *ch)
{
    sort(p,p+n,Hull_cmp);
    n=unique(p,p+n)-p;
    int v=0;
    //求下凸包，如果 p[i] 是右拐的，则不在凸包上，往回退
    for(int i=0;i<n;i++)
    {
        while(v>1 && sgn(Cross(ch[v-1]-ch[v-2],p[i]-ch[v-2]))<=0)
            v--;
        ch[v++]=p[i];
    }
    int j=v;
    //求上凸包
    for(int i=n-2;i>=0;i--)
    {
        while(v>j && sgn(Cross(ch[v-1]-ch[v-2],p[i]-ch[v-2]))<=0)
            v--;
        ch[v++]=p[i];
    }
    if(n>1) v--;
    return v;
}
//---------------------------------平面几何：圆-----------------------------------
struct Circle
{
    Point c;                                //圆心
    double r;
    Circle(){}
    Circle(Point c,double r):c(c),r(r){}
    Circle(double x,double y,double _r){c=Point(x,y);r=_r;}
};
//点和圆的关系：0 为圆内，1 为圆上，2 为圆外
int Point_circle_relation(Point p,Circle C)
{
    double dst=Distance(p,C.c);
    if(sgn(dst-C.r)<0) return 0;
    if(sgn(dst-C.r)==0) return 1;
    return 2;
}
//直线和圆的关系：0 为直线和圆相交，1 为直线和圆相切，2 为直线和圆相离
int Line_circle_relation(Line v,Circle C)
{
    double dst=Dis_point_line(C.c,v);
    if(sgn(dst-C.r)<0) return 0;
    if(sgn(dst-C.r)==0) return 1;
    return 2;
}
//线段和圆的关系：0 为线段和圆相交，1 为线段和圆相切，2 为线段和圆相离
int Seg_circle_relation(Segment v,Circle C)
{
    double dst=Dis_point_seg(C.c,v);
    if(sgn(dst-C.r)<0) return 0;
    if(sgn(dst-C.r)==0) return 1;
    return 2;
}
//直线和圆的交点，pa,pb 是交点，返回值是交点个数
int Line_cross_circle(Line v,Circle C,Point &pa,Point &pb)
{
    if(Line_circle_relation(v,C)==2) return 0;          //无交点
```

```
    Point q=Point_line_proj(C.c,v);
    double d=Dis_point_line(C.c,v);
    double k=sqrt(C.r*C.r-d*d);
    if(sgn(k)==0)
    {
        pa=q;pb=q;return 1;
    }
    Point n=(v.p2-v.p1)/Len(v.p2-v.p1);                 //直线的单位向量
    pa=q+n*k;
    pb=q-n*k;
    return 2;
}
```

字符串

kmp

```
inline void kmp(char *s,int *f){
    //enum from 1
    //every i : s[i-f[i]+1...i]=s[1...f[i]]
    int j=0,n=strlen(s+1);
    f[1]=0;
    for(int i=2;i<=n;++i){
        while(j&&s[j+1]!=s[i]) j=f[j];
        j+=(s[j+1]==s[i]);
        f[i]=j;
    }
}
```

exkmp

```
inline void ex_kmp(char *s,int *nxt,int n){
    //ENUM FROM 1
    //s[1..next[i]]=s[i...i+next[i]-1]
    int a=0,l=0,p=0;
    nxt[1]=n;
    for(int i=2;i<=n;++i){
        l=max(min(nxt[i-a+1],p-i+1),0);
        while(i+l<=n&&s[1+l]==s[i+l]) ++l;
        nxt[i]=l;
        if(i+l-1>p) a=i,p=i+l-1;
    }
}
```

manacher

```
namespace manacher{
    //ENUM FROM 0
    const int N=1.1e7+1000;// CHANGE IT!!! DONT CHANGE +1000
    char ch[N<<1],s[N];
    int f[N<<1],id,mx,n,len;
    // center i == f[i*2] center(i,i+1) == f[i*2+1]
    // len of palindrome = f[i]-1
    void init(){
        n=strlen(s);ch[0]='$';ch[1]='#';
        for(int i=1;i<=n;++i){
            ch[i*2]=s[i-1];
            ch[i*2+1]='#';
        }
        id=0;mx=0;ch[n*2+2]='#';
        for(int i=0;i<=2*n+10;++i) f[i]=0;
        for(int i=1;i<=2*n+2;++i){
            if(i>mx) f[i]=1;else f[i]=min(f[id*2-i],mx-i);
            while(ch[i-f[i]]==ch[i+f[i]]) ++f[i];
            if(i+f[i]>mx){mx=i+f[i];id=i;}
        }
    }
}
```

AC 自动机

```
const int N=1e6+10;
int c[N][26],val[N],f[N],sz;
inline void ins(char *s){
    int n=(int)strlen(s);int now=0;
    rep(i,0,n-1){
    int v=s[i]-'a';
    if(!c[now][v]) c[now][v]=++sz;
    now=c[now][v];
    }
    ++val[now];
}
inline void build(){
    queue<int> q;rep(i,0,25) if(c[0][i]){f[c[0][i]]=0;q.push(c[0][i]);}
    while(!q.empty()){
    int x=q.front();q.pop();
    rep(i,0,25){
        int &v=c[x][i];
        if(!v){v=c[f[x]][i];continue;}
        f[v]=c[f[x]][i];
        q.push(v);
    }
    }
}
```

后缀自动机

```
const int N=1e6+10;
struct SAM{
    int c[N<<1][26],fa[N<<1],len[N<<1],val[N<<1],last,sz;
    SAM(){last=sz=1;}
    void append(int x){
    int cur=++sz,p=last;last=cur;len[cur]=len[p]+1;
    for(;p&&!c[p][x];p=fa[p]) c[p][x]=cur;
    if(!p) fa[cur]=1;
    else{
        int q=c[p][x];
        if(len[q]==len[p]+1) fa[cur]=q;
        else{
        int nq=++sz;len[nq]=len[p]+1;
        memcpy(c[nq],c[q],sizeof(c[q]));
        fa[nq]=fa[q];fa[q]=fa[cur]=nq;
        for(;p&&c[p][x]==q;p=fa[p]) c[p][x]=nq;
        }
    }
    val[cur]=1;
    }
}sam;
```

后缀自动机求第 k 大子串

```
const int N=5e5+10;
struct SAM{
    int c[N<<1][26],fa[N<<1],len[N<<1],val[N<<1],last,sz;
    //dp : routes start from i
    ll dp[N<<1];
    SAM(){last=sz=1;}
    void append(int x){
    int cur=++sz,p=last;last=cur;len[cur]=len[p]+1;
    for(;p&&!c[p][x];p=fa[p]) c[p][x]=cur;
    if(!p) fa[cur]=1;
    else{
        int q=c[p][x];
        if(len[q]==len[p]+1) fa[cur]=q;
        else{
        int nq=++sz;len[nq]=len[p]+1;
        memcpy(c[nq],c[q],sizeof(c[q]));
        fa[nq]=fa[q];fa[q]=fa[cur]=nq;
        for(;p&&c[p][x]==q;p=fa[p]) c[p][x]=nq;
```

```
            }
        }
        val[cur]=1;
        }
        int bkt[N<<1],id[N<<1];
        void init(int t){
            rep(i,1,sz) ++bkt[len[i]];
            rep(i,1,sz) bkt[i]+=bkt[i-1];
            rpe(i,sz,1) id[bkt[len[i]]--]=i;
            if(t){
                rpe(i,sz,1){
                    int cur=id[i];
                    val[fa[cur]]+=val[cur];//count of occurence
                }
            }else{
                rep(i,1,sz) val[i]=1;
            }
            val[1]=0;
            rpe(i,sz,1){
                int cur=id[i];
                dp[cur]=val[cur];
                rep(j,0,25) dp[cur]+=dp[c[cur][j]];
            }
        }
        void dfs(int cur,int k){
            if(k<=val[cur]) return;
            k-=val[cur];
            rep(i,0,25){
                int t=c[cur][i];
                if(t){
                    if(k<=dp[t]){
                        putchar(i+'a');
                        dfs(t,k);
                        return;
                    }
                    k-=dp[t];
                }
            }
        }
}sam;
char s[N];
int main(){
    scanf("%s",s+1);
    int t=read(),k=read();
    int n=(int)strlen(s+1);
    rep(i,1,n) sam.append(s[i]-'a');
    sam.init(t);
    if(sam.dp[1]>=k){
        sam.dfs(1,k);
    }else{
        puts("-1");
    }
    return 0;
}
```

最小表示法

```
//寻找字典序最小的循环表示，下标从 0 开始，字符串扩展两倍
//rev true 返回最后一个循环节的起始点，否则返回第一个
int lex_find(int s[],int n,bool rev){
    int a=0,b=1,l;
    while(a<n&&b<n){
        for(l=0;l<n;++l)
            if(s[a+l]!=s[b+l]) break;
        if(l<n){
            if(s[a+l]<s[b+l]) b=b+l+1;//更改大于号变为最大表示
            else a=a+l+1;
            if(a==b) ++b;
        }else{
            if(a>b) swap(a,b);
            if(rev) return n-(b-a)+a;
```

```
            else return a;
        }
    }
    return min(a,b);
}
```

后缀数组

```
const int N=1e5+10;
char s[N];int sa[N],hei[N],rk[N],t1[N],t2[N],c[N];
inline void buildsa(char *a,int n,int m){
    int *x=t1,*y=t2,p=0;
    rep(i,1,n) c[x[i]=a[i]]++;
    rep(i,1,m) c[i]+=c[i-1];
    rpe(i,n,1) sa[c[x[i]]--]=i;
    for(int k=1;k<=n&&p<=n;k<<=1){
    p=0;
        rep(i,n-k+1,n) y[++p]=i;
        rep(i,1,n) if(sa[i]>k) y[++p]=sa[i]-k;
        rep(i,1,m) c[i]=0;
        rep(i,1,n) c[x[y[i]]]++;
        rep(i,1,m) c[i]+=c[i-1];
        rpe(i,n,1) sa[c[x[y[i]]]--]=y[i];
        swap(x,y);x[sa[1]]=1;p=2;
        rep(i,2,n) x[sa[i]]=y[sa[i]]==y[sa[i-1]]&&y[sa[i]+k]==y[sa[i-1]+k]? p-1:p++;
    m=p;
    }
    rep(i,1,n) rk[sa[i]]=i;
    int k=0;
    rep(i,1,n){
        if(rk[i]==1) continue;if(k) --k;
        while(a[i+k]==a[sa[rk[i]-1]+k]) ++k;
        hei[rk[i]]=k;
    }
}
int main(){
    scanf("%s",s+1);int n=(int)strlen(s+1);
    buildsa(s,n,233);
    rep(i,1,n) printf("%d ",sa[i]);pts;
    rep(i,2,n) printf("%d ",hei[i]);pts;
    return 0;
}
```

Lyndon 分解

将字符串分解成若干个 LyndonWord，他们的字典序单调减，每个 Word 都是它所有后缀中最小的。

```
namespace lyndon{
    vector<int> work(char *s,int n){
        int i=1;vector<int> res;res.clear();
        while(i<=n){
            int j=i;
            int k=i+1;
            while(k<=n&&s[j]<=s[k]){
                if(s[j]<s[k]) j=i;
                else ++j;
                ++k;
            }
            while(i<=j){
                res.emplace_back(i);
                i+=k-j;
            }
        }
        return res;
    }
}
```

Lyndon 分解求所有前缀最小字典序的后缀

```
const int N=1e6+10;
namespace lyndon{
    vector<int> work(char *s,int n){
        int i=1;vector<int> res;res.clear();
        while(i<=n){
            int j=i;
            int k=i+1;
            while(k<=n&&s[j]<=s[k]){
                if(s[j]<s[k]) j=i;
                else ++j;
                ++k;
            }
            while(i<=j){
                res.emplace_back(i);
                i+=k-j;
            }
        }
        return res;
    }
    vector<int> work_min_index(char *s,int n){
        int i=1;vector<int> ans;
        ans.resize(n+1);
        while(i<=n){
            ans[i]=i;
            int j=i;
            int k=i+1;
            while(k<=n&&s[j]<=s[k]){
                if(s[j]<s[k]){
                    ans[k]=i;
                    j=i;
                }
                else{
                    ans[k]=ans[j]+k-j;
                    ++j;
                }
                ++k;
            }
            while(i<=j){
                //res.emplace_back(i);//get Lyndon
                i+=k-j;
            }
        }
        return ans;
    }
}
const int P=1e9+7;
const int base=1112;
char s[N];
inline void wk(){
    scanf("%s",s+1);
    int n=(int)strlen(s+1);
    vector<int> ans=lyndon::work_min_index(s,n);
    int ret=0;
    //rep(i,1,n) cerr<<ans[i]<<" ";cerr<<endl;
    rpe(i,n,1)
        ret=(1ll*ret*base%P+ans[i])%P;
    printf("%d\n",ret);
}
```

多项式

NTT 模数

```
NTTPrimes = {1053818881, 1051721729, 1045430273, 1012924417, 1007681537, 1004535809, 998244353, 985661441,
    976224257, 975175681};
NTTPrimitiveRoots = {7, 6, 3, 5, 3, 3, 3, 3, 3, 17};
```

FFT

```
namespace FFT{
    const db pi=acos(-1);
    struct cp{
        db re,im;
        cp(db _re=0,db _im=0){re=_re;im=_im;}
        cp operator +(cp b){return cp(re+b.re,im+b.im);}
        cp operator -(cp b){return cp(re-b.re,im-b.im);}
        cp operator *(cp b){return cp(re*b.re-im*b.im,re*b.im+im*b.re);}
    };
    int r[N];cp c[N<<1];
    inline void fft(cp *a,int f,int n){
        rep(i,0,n-1) if(r[i]>i) swap(a[r[i]],a[i]);
        for(int i=1;i<n;i<<=1){
            cp wn(cos(pi/i),f*sin(pi/i));
            for(int j=0,p=(i<<1);j<n;j+=p){
                cp w(1,0);
                for(int k=0;k<i;++k,w=w*wn){
                    cp x=a[j+k],y=w*a[j+k+i];
                    a[j+k]=x+y;a[j+k+i]=x-y;
                }
            }
        }
        if(f==-1){rep(i,0,n-1) a[i].re/=n,a[i].im/=n;}
    }
    inline int mul(db *a,db *b,int n,int m){
        n+=m;rep(i,0,n) c[i]=cp(a[i],b[i]);
        int l=0;m=n;for(n=1;n<=m;n<<=1) ++l;
        rep(i,0,n-1) r[i]=(r[i>>1]>>1)|((i&1)<<(l-1));
        rep(i,m+1,n) c[i]=cp(0,0);
        fft(c,1,n);rep(i,0,n-1) c[i]=c[i]*c[i];
        fft(c,-1,n);
        rep(i,0,m) a[i]=c[i].im/2;
        return n;
    }
}
```

NTT

```
namespace NTT{
    const int P=998244353,g=3,ig=332748118;
    inline int qpow(int a,int b){int q=1;while(b){if(b&1)q=1LL*q*a%P;a=1LL*a*a%P;b>>=1;}return q;}
    int r[N],ow[N],inv[N];
    inline void ntt(int *a,int f,int n){
        rep(i,0,n-1) if(r[i]>i) swap(a[i],a[r[i]]);
        for(int i=1;i<n;i<<=1){
            int wn=qpow(f,(P-1)/(i<<1));
            ow[0]=1;rep(k,1,i-1) ow[k]=1LL*ow[k-1]*wn%P;
            for(int j=0,p=(i<<1);j<n;j+=p){
                for(int k=0;k<i;++k){
                    int x=a[j+k],y=1LL*ow[k]*a[j+k+i]%P;
                    a[j+k]=(x+y)%P;a[j+k+i]=(x+P-y)%P;
                }
            }
        }
        if(f==ig){
            int iv=qpow(n,P-2);
            rep(i,0,n-1) a[i]=1LL*a[i]*iv%P;
        }
    }
    int tma[N],tmb[N];
    inline int mul(int *a,int *b,int n,int m,int ci){
        int _n=n,_m=m,l=0;m+=n;for(n=1;n<=m;n<<=1) ++l;
        rep(i,0,n-1) r[i]=(r[i>>1]>>1)|((i&1)<<(l-1));
        rep(i,0,n-1) tma[i]=a[i];rep(i,0,n-1) tmb[i]=b[i];
        rep(i,_n+1,n) tma[i]=0;rep(i,_m+1,n) tmb[i]=0;
        ntt(tma,g,n);ntt(tmb,g,n);
        while(ci){
            if(ci&1) rep(i,0,n-1) tma[i]=1LL*tma[i]*tmb[i]%P;
            rep(i,0,n-1) tmb[i]=1LL*tmb[i]*tmb[i]%P;
```

```
            ci>>=1;
        }
        ntt(tma,ig,n);
        rep(i,0,n-1) a[i]=tma[i];
        return n;
    }
}
inline void prepare(){
    //NTT inv
    using NTT::inv;using NTT::P;
    inv[1]=1;rep(i,2,N-1) inv[i]=1LL*(P-P/i)*inv[P%i]%P;
}
```

完整的板板 by hls

```
/*
    NTTPrimes = {1053818881, 1051721729, 1045430273, 1012924417, 1007681537, 1004535809, 998244353, 985661441,
    976224257, 975175681};
    NTTPrimitiveRoots = {7, 6, 3, 5, 3, 3, 3, 3, 3, 17};
*/
//poly start
namespace Poly{
    const int N=6e5+10;
    namespace NTT{
        const int P=998244353,g=3,ig=332748118;
        inline int qpow(int a,int b){int q=1;while(b){if(b&1)q=1LL*q*a%P;a=1LL*a*a%P;b>>=1;}return q;}
        int r[N],ow[N],inv[N];
        inline void ntt(int *a,int f,int n){
            rep(i,0,n-1) if(r[i]>i) swap(a[i],a[r[i]]);
            for(int i=1;i<n;i<<=1){
                int wn=qpow(f,(P-1)/(i<<1));
                ow[0]=1;rep(k,1,i-1) ow[k]=1LL*ow[k-1]*wn%P;
                for(int j=0,p=(i<<1);j<n;j+=p){
                    for(int k=0;k<i;++k){
                        int x=a[j+k],y=1LL*ow[k]*a[j+k+i]%P;
                        a[j+k]=(x+y)%P;a[j+k+i]=(x+P-y)%P;
                    }
                }
            }
            if(f==ig){
                int iv=qpow(n,P-2);
                rep(i,0,n-1) a[i]=1LL*a[i]*iv%P;
            }
        }
        int tma[N],tmb[N];
        inline int mul(int *a,int *b,int n,int m,int ci){
            int _n=n,_m=m,l=0;m+=n;for(n=1;n<=m;n<<=1) ++l;
            rep(i,0,n-1) r[i]=(r[i>>1]>>1)|((i&1)<<(l-1));
            rep(i,0,n-1) tma[i]=a[i];rep(i,0,n-1) tmb[i]=b[i];
            rep(i,_n+1,n) tma[i]=0;rep(i,_m+1,n) tmb[i]=0;
            ntt(tma,g,n);ntt(tmb,g,n);
            while(ci){
                if(ci&1) rep(i,0,n-1) tma[i]=1LL*tma[i]*tmb[i]%P;
                rep(i,0,n-1) tmb[i]=1LL*tmb[i]*tmb[i]%P;
                ci>>=1;
            }
            ntt(tma,ig,n);
            rep(i,0,n-1) a[i]=tma[i];
            return n;
        }
    }
    namespace FFT{
        const db pi=acos(-1);
        struct cp{
            db re,im;
            cp(db _re=0,db _im=0){re=_re;im=_im;}
            cp operator +(cp b){return cp(re+b.re,im+b.im);}
            cp operator -(cp b){return cp(re-b.re,im-b.im);}
            cp operator *(cp b){return cp(re*b.re-im*b.im,re*b.im+im*b.re);}
        };
        int r[N];cp c[N<<1];
```

```
    inline void fft(cp *a,int f,int n){
        rep(i,0,n-1) if(r[i]>i) swap(a[r[i]],a[i]);
        for(int i=1;i<n;i<<=1){
            cp wn(cos(pi/i),f*sin(pi/i));
            for(int j=0,p=(i<<1);j<n;j+=p){
                cp w(1,0);
                for(int k=0;k<i;++k,w=w*wn){
                    cp x=a[j+k],y=w*a[j+k+i];
                    a[j+k]=x+y;a[j+k+i]=x-y;
                }
            }
        }
        if(f==-1){rep(i,0,n-1) a[i].re/=n,a[i].im/=n;}
    }
    inline int mul(db *a,db *b,int n,int m){
        n+=m;rep(i,0,n) c[i]=cp(a[i],b[i]);
        int l=0;m=n;for(n=1;n<=m;n<<=1) ++l;
        rep(i,0,n-1) r[i]=(r[i>>1]>>1)|((i&1)<<(l-1));
        rep(i,m+1,n) c[i]=cp(0,0);
        fft(c,1,n);rep(i,0,n-1) c[i]=c[i]*c[i];
        fft(c,-1,n);
        rep(i,0,m) a[i]=c[i].im/2;
        return n;
    }
}
using namespace NTT;
ll w,a;
struct node{
    ll x,y;
    node friend operator *(node x,node y){
        node z;
        z.x=(x.x*y.x%P+x.y*y.y%P*w%P)%P;
        z.y=(x.x*y.y%P+x.y*y.x%P)%P;
        return z;
    }
}u,v;
inline node Cqpow(node a,ll b){
    node q;q.x=1;q.y=0;
    while(b){if(b&1) q=q*a;a=a*a;b>>=1;}
    return q;
}
inline ll cipolla(int n,int P){
    n%=P;srand(0x20010412);
    if(P==2) return 1;
    if(qpow(n,(P-1)/2)==P-1) return -1;
    while(1){
        a=rand()%P;
        w=(a*a-n+P)%P;
        if(qpow(w%P,(P-1)/2)==P-1) break;
    }
    u.x=a,u.y=1;
    u=Cqpow(u,(P+1)/2);
    ll fir=u.x,sec=P-u.x;
    if(fir>sec)swap(fir,sec);
    return fir;
}
inline void derivative(int *a,int *b,int n){
    rpe(i,n-2,0) b[i]=1LL*a[i+1]*(i+1)%P;
    b[n-1]=0;
}
inline void integral(int *a,int *b,int n){
    rpe(i,n-1,1) b[i]=1LL*a[i-1]*inv[i]%P;
    b[0]=0;
}
inline void differential(int *a,int *b,int n){
    rep(i,1,n-1) b[i]=(a[i]+P-a[i-1])%P;
    b[0]=0;
}
int tf[N],tg[N];
inline void inverse(int *f,int *g,int n){
    if(n==1){
```

```
            g[0]=qpow(f[0],P-2);return;
        }
        inverse(f,g,n>>1);
        rep(i,0,n-1) tf[i]=f[i],tg[i]=g[i];
        int tmp=mul(tf,tg,n-1,n-1,2);
        rep(i,0,n-1) g[i]=((-tf[i]+2LL*g[i])%P+P)%P;
        rep(i,0,tmp) tf[i]=tg[i]=0;
    }
    int ta[N],tb[N];
    inline void sqrt(int *a,int *b,int n){
        if(n==1){
            b[0]=cipolla(a[0],P);//debug(b[0]);debug(1LL*b[0]*b[0]%P);
            return;
        }
        sqrt(a,b,n>>1);rep(i,n,(n<<1)) b[i]=0;
        inverse(b,tb,n);
        rep(i,0,n-1) ta[i]=a[i];
        mul(ta,tb,n-1,n-1,1);//debug(ta[0]);debug(b[0]);
        rep(i,0,n-1) b[i]=1LL*(b[i]+ta[i])%P*inv[2]%P;
        rep(i,0,n<<1) ta[i]=tb[i]=0;
    }
    inline void ln(int *a,int *b,int n){
        inverse(a,ta,n);
        derivative(a,b,n);
        mul(b,ta,n,n,1);
        integral(b,b,n);
    }
    inline void exp(int *a,int *b,int n){
        if(n==1){
            b[0]=1;return;
        }
        exp(a,b,n>>1);
        ln(b,tb,n);rep(i,0,n-1) ta[i]=a[i];++ta[0];
        rep(i,0,n-1) ta[i]-=tb[i];
        mul(ta,b,n,n,1);rep(i,0,n-1) b[i]=ta[i];
        rep(i,0,n<<1) ta[i]=tb[i]=0;
    }
}
//
using namespace Poly;
inline void prepare(){
    //NTT
    using NTT::inv;using NTT::P;
    inv[1]=1;rep(i,2,N-1) inv[i]=1LL*(P-P/i)*inv[P%i]%P;
}
int n,A[N],B[N];
int main(){
    prepare();
    n=read();rep(i,0,n) A[i]=read();
    int len=1;for(;len<=n;len<<=1);
/*
    sqrt(A,B,len);
    rep(i,0,n) printf("%d ",B[i]);pts;
//    debug(1LL*B[0]*B[0]%P);
    NTT::mul(B,B,n,n,1);
    rep(i,0,n) printf("%d ",B[i]);pts;
*/ //sqrt 1e5 uoj 333ms
/*
    ln(A,B,len);
    rep(i,0,n) printf("%d ",B[i]);pts;
    exp(B,A,len);
    rep(i,0,n) printf("%d ",A[i]);pts;
*/ //ln&exp 1e5 uoj 766ms
   //ln 222ms
   //exp 568ms
    return 0;
}
```

容斥与反演

莫比乌斯反演

2.2.1 莫比乌斯反演定理若 $F(n)$ 和 $f(n)$ 是定义在非负整数集合上的两个函数，并且满足条件 $F(n)=\sum_{d|n}f(d)$ 。那么我们得到结论：

$$f(n)=\sum_{d|n}\mu(d)F(\frac{n}{d})$$

证明：

$$\sum_{d|n}\mu(d)F(\frac{n}{d})=\sum_{d|n}\mu(d)\sum_{k|\frac{n}{d}}f(k)=\sum_{k|n}f(k)\sum_{d|\frac{n}{k}}\mu(d)=f(n)$$

由这个定理我们可以得到莫比乌斯函数的另外一个性质：对于 $\forall n\in\mathbb{N}_+$

$$\sum_{d|n}\frac{\mu(d)}{d}=\frac{\varphi(n)}{n}$$

证明：

引理：$n=\sum_{d|n}\varphi(d)$

证明：

对于 n 的每个因数 d ，我们取出 $[1,d]$ 内的 $\varphi(d)$ 个与 n 互素的数记做集合 $A=\{a_1,a_2,\cdots,a_{\varphi(d)}\}$ ，将集合 A 内的元素对应到集合 $B=\{a_1\frac{n}{d},a_2\frac{n}{d},\cdots,a_{\varphi(d)}\frac{n}{d}\}$ 。显然 $gcd(a_i\frac{n}{d},n)=\frac{n}{d}$ 。由于 d 枚举了所有 n 的因数，所以 $\frac{n}{d}$ 也是。则集合 B 内是 $[1,n]$ 内所有的数。故原命题成立。

有了这个引理，我们将莫比乌斯反演定理中的 $F(n)=n,f(n)=\varphi(n)$ 。

$$\varphi(n)=\sum_{d|n}\frac{n}{d}\mu(d)$$
$$\frac{\varphi(n)}{n}=\sum_{d|n}\frac{\mu(d)}{d}$$

第二形式：

$$F(n)=\sum_{n|d}f(d)\Rightarrow f(n)=\sum_{n|d}\mu(\frac{d}{n})F(d)$$

证明：令 $k=\frac{d}{n}$ ，那么

$$\sum_{k=1}^{+\infty}\mu(k)F(nk)=\sum_{k=1}^{+\infty}\mu(k)\sum_{nk|t}f(t)=\sum_{n|t}f(t)\sum_{k|\frac{t}{n}}\mu(k)=f(n)$$

2.2.2 杜教筛（普适）若 $f(n)$ 是一个积性函数，求 $f(n)$ 的前缀 $S(n)$ 。即 $S(n)=\sum_{i=1}^{n}f(n)$ 。

狄利克雷卷积

对于数论函数 $g(n),f(n)$ ，其狄利克雷卷积 $h(n)$ 也是一个数论函数

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

我们找到另一个积性函数 $g(n)$ ，让 $f(n)$ 和 $g(n)$ 做一个卷积

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

求卷积的前缀

$$\sum_{i=1}^{n}(g*f)(i)=\sum_{i=1}^{n}\sum_{d|i}g(d)f\left(\frac{i}{d}\right)$$

提出右式的 d

$$\begin{aligned}\Rightarrow\sum_{i=1}^{n}(g*f)(i)&=\sum_{d=1}^{n}g(d)\sum_{d|i}f\left(\frac{i}{d}\right)\\&=\sum_{d=1}^{n}g(d)\sum_{i=1}^{\lfloor\frac{n}{d}\rfloor}f(i)\\&=\sum_{d=1}^{n}g(d)S\left(\left\lfloor\frac{n}{d}\right\rfloor\right)\end{aligned}$$

容易得到这个式子

$$g(1)S(n)=\sum_{i=1}^{n}g(i)S\left(\left\lfloor\frac{n}{i}\right\rfloor\right)-\sum_{i=2}^{n}g(i)S\left(\left\lfloor\frac{n}{i}\right\rfloor\right)$$

其实就是

$$g(1)S(n)=\sum_{i=1}^{n}(g*f)(i)-\sum_{i=2}^{n}g(i)S\left(\left\lfloor\frac{n}{i}\right\rfloor\right)$$

我们发现如果狄利克雷卷积前缀很好算的话，积性函数的前缀也可以分块递归来算了。

举几个例子：

1. 求 $S(n)=\sum\limits_{i=1}^{n}\mu(n)$

上述式子

$$g(1)S(n)=\sum_{i=1}^{n}(g*\mu)(i)-\sum_{i=2}^{n}g(i)S\left(\left\lfloor\frac{n}{i}\right\rfloor\right)$$

考虑到 $\sum\limits_{d|n}\mu(d)=[n=1]$, 又由于 $(g*\mu)(n)=\sum\limits_{d|n}g(d)\mu\left(\frac{n}{d}\right)$ 。我们考虑让 $g(n)=1(n)$, 那么 $(1*\mu)(n)=\sum\limits_{d|n}1\cdot\mu(d)=[n=1]$ 。显然这个卷积的前缀为 $\sum\limits_{i=1}^{n}(g*\mu)(i)=1(n)$ 。

故对于 μ

$$S(n)=1-\sum_{i=2}^{n}S\left(\left\lfloor\frac{n}{i}\right\rfloor\right)$$

2. 求 $S(n)=\sum\limits_{i=1}^{n}\varphi(n)$

上述式子

$$g(1)S(n)=\sum_{i=1}^{n}(g*\varphi)(i)-\sum_{i=2}^{n}g(i)S\left(\left\lfloor\frac{n}{i}\right\rfloor\right)$$

考虑到 $\sum\limits_{d|n}\varphi(d)=n$ ，又由于 $(g*\varphi)(n)=\sum\limits_{d|n}g(d)\varphi\left(\frac{n}{d}\right)$ 。我们考虑让 $g(n)=1(n)$ ，那么 $(1*\varphi)(n)=\sum\limits_{d|n}1\cdot\varphi(d)=n$ 。显然这个卷积的前缀为 $\sum\limits_{i=1}^{n}(g*\varphi)(i)=\frac{n(n+1)}{2}$ 。

故对于 φ

$$S(n)=\frac{n(n+1)}{2}-\sum_{i=2}^{n}S\left(\left\lfloor\frac{n}{i}\right\rfloor\right)$$

```cpp
LL mu[N+5], phi[N+5], n;
struct num {
    LL ans1, ans2;
    num() {}
    num(LL _ans1, LL _ans2) {ans1 = _ans1, ans2 = _ans2; }
}ans;
map<LL, num>mp;

int prime[N+5], isprime[N+5], tot;
void get_pre() {
    memset(isprime, 1, sizeof(isprime)); isprime[1] = 0, mu[1] = phi[1] = 1;
    for (int i = 2; i <= N; i++) {
    if (isprime[i]) prime[++tot] = i, mu[i] = -1, phi[i] = i-1;
    for (int j = 1; j <= tot && i*prime[j] <= N; j++) {
        isprime[i*prime[j]] = 0;
        if (i%prime[j]) mu[i*prime[j]] = -mu[i], phi[i*prime[j]] = phi[i]*(prime[j]-1);
        else {mu[i*prime[j]] = 0, phi[i*prime[j]] = phi[i]*prime[j]; break; }
    }
    mu[i] += mu[i-1], phi[i] += phi[i-1];
    }
}
num Less (const num &a, const num &b) {num ans; ans.ans1 = a.ans1 - b.ans1, ans.ans2 = a.ans2-b.ans2; return ans; }
num Times (const num &a, const LL &x) {num ans; ans.ans1 = a.ans1*x , ans.ans2 = a.ans2*x; return ans; }
num cal(LL x) {
    if (x <= N) return num(phi[x], mu[x]);
    if (mp.count(x)) return mp[x];
    num ans = num(x*(x+1)/2, 1);
    for (LL i = 2, last; i <= x; i = last+1) {
    last = x/(x/i); ans = Less(ans, Times(cal(x/i), (last-i+1)));
    }
    return mp[x] = ans;
}
void work() {
    read(n); ans = cal(n);
    write(ans.ans1), putchar(' ');
    if (ans.ans2 < 0) putchar('-'), writeln(-ans.ans2);
    else writeln(ans.ans2);
}
```

快速莫比乌斯变换（反演）与子集卷积

莫比乌斯变换（反演）

问题提出

若 h, f, g 为下标为集合的函数，我们定义

$$h = f * g$$

表示

$$h(S) = \sum_{L \subseteq S} \sum_{R \subseteq S} [L \cup R = S] f(L) \times g(R)$$

容易发现，对于这个问题，我们可以用 $O\left((2^n)^2\right)$ 的枚举 L, R 来计算。

然而这样复杂度较高，我们考虑类比多项式卷积的过程，可以求出 f, g 的点值，直接相乘得到 h 的点值然后再插回去。

值得注意的是为了便于表述以及规范表达，快速莫比乌斯变换就相当于点值，快速莫比乌斯反演就相当于插值。

算法原理

- 我们定义 f 的莫比乌斯变换为 F，其中 $F(S) = \sum_{X \subseteq S} f(X)$；由这个定义，我们可以推出 F 莫比乌斯反演 f 为 $f(S) = \sum_{X \subseteq S} (-1)^{|S|-|X|} F(X)$。对于莫比乌斯反演的证明，可以带入莫比乌斯变换的式子或容斥来证。

- 我们对于一个函数 f,g,h ，记它的点值式为 F,G,H 。我们将**问题提出**中的卷积式两边同时做莫比乌斯变换，得到

$$\begin{aligned}H(S) &= \sum_{L\subseteq S}\sum_{R\subseteq S}[L\cup R\subseteq S]f(L)\times g(R)\\ &= \sum_{L\subseteq S}\sum_{R\subseteq S}f(L)\times g(R)\\ &= \left(\sum_{L\subseteq S}f(L)\right)\times\left(\sum_{R\subseteq S}g(R)\right)\\ &= F(S)\times G(S)\end{aligned}$$

至此算法原理及过程已经完全结束。似乎我们可以用 $O(3^n)$ 枚举子集来变换和反演，实际上我们可以让复杂度更优。

算法实现

- 设 $\hat{f}_S^{(i)}$ 表示 $\sum_{T\subseteq S}[(S-T)\subseteq\{1,2,...,i\}]f_T$
- 易得初始状态：$\hat{f}_S^{(0)}=f_S$
- 对于每一个不包含 $\{i\}$ 的集合 S，可知 $\hat{f}_S^{(i)}=\hat{f}_S^{(i-1)}$（因为 S 并没有 i 这位），$\hat{f}_{S\cup\{i\}}^{(i)}=\hat{f}_S^{(i-1)}+\hat{f}_{S\cup\{i\}}^{(i-1)}$（前者的 T 没有包含 $\{i\}$，而后者的 T 必须包含了 $\{i\}$）。
- 显然，递推了 n 轮之后，$\hat{f}_S^n$ 就是所求的变换了。

用高维前缀和可以做到 $O(n\times 2^n)$ 的递推，求出点值和插值。

```
void FMT(int *A, int o) {// o 为识别因子
    for (int i = 1; i < ST; i <<= 1)//ST-1 表示全集
        for (int j = 0; j < ST; j++)
            if (i&j) (A[j] += A[j^i]*o) %= mod;
}
```

例题 - [HAOI 2015] 按位或

子集卷积

FWT：“你刚才说的那个玩意我也能做啊，要你何用？”

FMT：“......”

问题提出

若 h,f,g 为下标为集合的函数，我们定义

$$h=f*g$$

表示

$$h(S)=\sum_{X\subseteq S}f(X)\times g(S-X)$$

算法实现

回顾刚刚的集合并卷积，子集卷积的条件比集合并卷积更苛刻，即 L 和 R 的集合应该不相交。

我们可以在卷积时多加一维，维护集合的大小，如 $f_{i,S}$ 表示集合中有 i 个元素，集合表示为 S 。显然，当 i 和 S 的真实元素个数符合时才是对的。记数组 `cnt[S]` 表示集合 S 的模。初始时，我们只把 $f_{cnt[S],S}$ 的值赋成原来的 $f(S)$（g 同理），然后每一维做一遍 FMT，点值相乘时这么写：$h_{i,S}=\sum_{j=0}^{i}f_{j,S}\times g_{i-j,S}$ 。最后扫一遍把不符合实际情况的状态赋成 0 即可。

```
for (int i = 0; i <= n; i++) FMT(g[i], 1);
for (int i = 0; i <= n; i++) FMT(f[i], 1);
for (int i = 1; i <= n; i++) {
    for (int j = 0; j <= i; j++)
```

```
        for (int k = 0; k < ST; k++)
            (h[i][k] += 1ll*f[j][k]*g[i-j][k]%mod) %= mod;
    FMT(h[i], -1);
    for (int k = 0; k < ST; k++) if (cnt[k] != i) h[i][k] = 0;
    if (i != n) FMT(h[i], 1);
}
```

例题 - [WC 2018] 州区划分

二项式反演

内容

对于函数 f,g，$\forall p \in \mathbb{N}$ 若 $\forall n \geq p$，满足

$$f(n)=\sum_{k=p}^{n}\binom{n}{k}g(k)$$

那么 $\forall n \geq p$

$$g(n)=\sum_{k=p}^{n}(-1)^{n-k}\binom{n}{k}f(k)$$

证明

为了方便表达，我们取 $p=0$，实质和取 $p \in \mathbb{N}$ 的证明方法是一样的。

$$\begin{aligned}
g(n) &= \sum_{k=0}^{n}(-1)^{n-k}\binom{n}{k}f(k) \\
&= \sum_{k=0}^{n}(-1)^{n-k}\binom{n}{k}\sum_{i=0}^{k}\tbinom{k}{i}g(i) \\
&= \sum_{k=0}^{n}\sum_{i=0}^{k}(-1)^{n-k}\tbinom{n}{k}\tbinom{k}{i}g(i) \\
&= \sum_{i=0}^{n}\left(\sum_{k=i}^{n}(-1)^{n-k}\tbinom{n}{k}\tbinom{k}{i}\right)g(i) \\
&= \sum_{i=0}^{n}\left(\sum_{k=i}^{n}(-1)^{n-k}\tbinom{n}{i}\tbinom{n-i}{k-i}\right)g(i) \\
&= \sum_{i=0}^{n}\left(\tbinom{n}{i}\sum_{k=i}^{n}(-1)^{n-k}\tbinom{n-i}{n-k}\right)g(i) \\
&= \sum_{i=0}^{n}\left(\tbinom{n}{i}(1-1)^{n-i}\right)g(i) \\
&= g(n)
\end{aligned}$$

故成立。

应用举例

推导错排公式

我们记 $f(n)$ 为 n 个数字任意放的方案数，$g(n)$ 为 n 个数没有一个放在自己位置上的方案数。

枚举不在自己位置上的个数，容易得到

$$f(n)=\sum_{i=0}^{n}\binom{n}{i}g(i)$$

那么

$$\begin{aligned}g(n)&=\sum_{i=0}^{n}(-1)^{n-i}\binom{n}{i}f(i)\\&=\sum_{i=0}^{n}(-1)^{i}\binom{n}{i}f(n-i)\end{aligned}$$

注意到 $f(x)=x!$，那么

$$\begin{aligned}g(n)&=\sum_{i=0}^{n}(-1)^{i}\frac{n!}{i!(n-i)!}(n-i)!\\&=n!\sum_{i=0}^{n}(-1)^{i}\frac{1}{i!}\end{aligned}$$

棋盘染色

有个 $1\times n$ 的格子，m 种颜色（$m\geq 2$），要求相邻格子的颜色不相同且每种颜色都要用到，求染色方案数。

我们记 $f(n)$ 为至多用到 n 种颜色的方案数，$g(n)$ 为 n 为恰用到 n 种颜色的方案数。

那么

$$\begin{aligned}f(m)&=\sum_{i=2}^{m}\binom{m}{i}g(i)\\\Rightarrow g(m)&=\sum_{i=2}^{m}(-1)^{i}\binom{m}{i}f(n-i)\end{aligned}$$

注意到 $f(x)=x\times(x-1)^{n-1}$ 。那么就可以带入直接算了。

另一形式

$$a_k=\sum_{i=k}^{n}\binom{i}{k}b_i\Rightarrow b_k=\sum_{i=k}^{n}(-1)^{i-k}\binom{i}{k}a_i$$

证明:

$$
\begin{aligned}
&\sum_{i=k}^{n}(-1)^{i-k}\binom{i}{k}a_i\\
=&\sum_{i=k}^{n}(-1)^{i-k}\binom{i}{k}\sum_{j=k}^{n}\binom{j}{i}b_i\\
=&\sum_{i=k}^{n}\sum_{j=k}^{n}(-1)^{i-k}\binom{i}{k}\binom{j}{i}b_i\\
=&\sum_{i=k}^{n}\sum_{j=k}^{n}(-1)^{i-k}\binom{j}{i}\binom{i}{k}b_i\\
=&\sum_{j=k}^{n}\sum_{i=k}^{j}(-1)^{i-k}\binom{j}{k}\binom{j-k}{i-k}b_i\\
=&\sum_{j=k}^{n}\binom{j}{k}\sum_{i=k}^{j}(-1)^{i-k}\binom{j-k}{i-k}b_i\\
=&\sum_{j=k}^{n}\binom{j}{k}(1-1)^{j-k}b_i\\
=&b_k
\end{aligned}
$$

例题 - [BZOJ 2839] 集合计数 - [BZOJ 3622] 已经没有什么好害怕的了

斯特林反演

第一类斯特林数

$\begin{bmatrix}n\\m\end{bmatrix}$ 表示将 n 个元素排成 m 个轮换的方法数。

递推公式：$\begin{bmatrix}n\\m\end{bmatrix}=\begin{bmatrix}n-1\\m-1\end{bmatrix}+(n-1)\begin{bmatrix}n-1\\m\end{bmatrix}$

含义是考虑第 n 个元素的放法：要么新开一个轮换，要么就放在前 $n-1$ 个元素的左边。

第二类斯特林数

$\begin{Bmatrix}n\\m\end{Bmatrix}$ 表示将 n 个元素划分成 m 个非空子集的方法数。

递推公式：$\begin{Bmatrix}n\\m\end{Bmatrix}=\begin{Bmatrix}n-1\\m-1\end{Bmatrix}+m\begin{Bmatrix}n-1\\m\end{Bmatrix}$

含义是考虑第 n 个元素的放法：要么新开一个组，要么就放在前 m 组内。

通项公式（容斥式）：$\begin{Bmatrix}n\\m\end{Bmatrix}=\frac{1}{m!}\sum\limits_{k=0}^{m}(-1)^k\binom{m}{k}(m-k)^n$

有关通项公式的证明及运用可以参考多项式类数学相关这篇文章。

例题 - [Codeforces 932E]Team Work - [Codeforces 961G]Partitions - [TJOI 2016&HEOI 2016] 求和

反演公式

$$f(x)=\sum_{i=0}^{x}\begin{Bmatrix}x\\i\end{Bmatrix}g(i)\Leftrightarrow g(x)=\sum_{i=0}^{x}(-1)^{x-i}\begin{bmatrix}x\\i\end{bmatrix}f(i)$$

$$f(x)=\sum_{i=0}^{x}\begin{bmatrix}x\\i\end{bmatrix}g(i)\Leftrightarrow g(x)=\sum_{i=0}^{x}(-1)^{x-i}\begin{Bmatrix}x\\i\end{Bmatrix}f(i)$$

例题 - 给出 n 个点的一张简单图，问有多少个边的子集，满足保留子集中的边后，该图连通。（蒯自Sdchr）- 大概就是枚举连通块的个数，然后块内随便连，然后容斥就好。- 考虑如何求容斥系数 $f(i)$ 。设实际上是 x 个连通块的方案，它应该被计算 $[x=1]$ 次，实际上在所有更仔细的分块中被统计，所以 -

$$[x=1]=\sum_{i=1}^{x}\begin{Bmatrix}x\\i\end{Bmatrix}f(i)$$

- 由斯特林反演 -

$$\begin{aligned}f(x)&=\sum_{i=1}^{x}(-1)^{x-i}\begin{bmatrix}x\\i\end{bmatrix}[i=1]\\&=(-1)^{x-1}\begin{bmatrix}x\\1\end{bmatrix}\\&=(-1)^{x-1}(x-1)!\end{aligned}$$

- [BZOJ 4671] 异或图

最值反演（min-max 容斥）

公式

记 $\max(S)$ 为集合 S 中的最大值，$\min(S)$ 为集合 S 中的最小值，$|S|$ 为集合 S 的元素数量，那么以下两个等式成立

$$\max(S)=\sum_{T\subseteq S}(-1)^{|T|+1}\min(T)$$

$$\min(S)=\sum_{T\subseteq S}(-1)^{|T|+1}\max(T)$$

证明

这里只证明第一个等式好了，后边的可以自行推出。

其实只需要证明一件事，就是除了 $\min(T)=\max(S)$ 的那个值，其他的 min 值都被消掉了就可以了（这里说明一下，我们假定集合中的元素两两相异）

先来说明 $\max(S)$ 的系数为什么是 1 ，假设中 S 最大的元素是 a ，那么我们会发现只有 $\min(\{a\})=\max(S)$ 所以 $\max(S)$ 的系数必须是 1 。

然后再说明为什么别的 min 都被消掉了，假设某个元素 b 的排名是 k ，那么 $\min(T)=b$ 当且仅当我们选出的集合是后 $n-k$ 个的元素构成的集合的子集然后并上 $\{b\}$ 得到的，我们会发现显然这样的集合有 2^{n-k} 种，而显然这其中恰有 2^{n-k-1} 中是有奇数个元素的，恰有 2^{n-k-1} 种是有偶数个元素的，两两相消自然就成 0 了，当然上述等式在 $k=n$ 的时候不成立，但是此时剩下的刚好是最大值，所以证明完毕。

拉格朗日插值法

简介

给定 $n+1$ 个**横坐标不相同**的点，可以唯一确定一个 n 次的多项式。最直观的求多项式的做法就是列方程求解。但是这样需要 $O(n^3)$ 的时间来计算。而拉格朗日插值法则通过构造的方法，得到了一个经过 $n+1$ 个点的 n 次多项式。具体的过程是这样的，假设现在我们得到了 $n+1$ 个点:

$$(x_0,y_0),\ (x_1,y_1),\ \ldots,\ (x_n,y_n)$$

设**拉格朗日基本多项式**为

$$\ell_j(x)=\prod_{i=0,i\neq j}^{n}\frac{x-x_i}{x_j-x_i}$$

这个基本多项式构造十分巧妙，因为注意到 $\ell_j(x_j)=1$ ，并且 $\ell_j(x_i)=0,\ \forall i\neq j$ 。那么，接着构造出这个 n 次多项式

$$P(x)=\sum_{i=0}^{n}y_i\ell_i(x)$$

根据基本多项式的性质，我们可以知道 $P(x_i)=y_i$ ，也就是经过了这 $n+1$ 个点。通过简单的多项式乘法和多项式除法就可以在 $O(n^2)$ 的时间求出这个多项式的系数表达。

求解

已知 n 次多项式 $f(n)$ 上的 $n+1$ 个点 $(x_i,y_i),i\in[0,n]$ ，求 $f(xi)$

```
int lagrange(int n, int *x, int *y, int xi) {
    int ans = 0;
    for (int i = 0; i <= n; i++) {
        int s1 = 1, s2 = 1;
        for (int j = 0; j <= n; j++)
            if (i != j) {
                s1 = 1ll*s1*(xi-x[j])%mod;
                s2 = 1ll*s2*(x[i]-x[j])%mod;
            }
        ans = (1ll*ans+1ll*y[i]*s1%mod*quick_pow(s2, mod-2)%mod)%mod;
    }
    return (ans+mod)%mod;
}
```

如果 x 的取值是连续一段的话，我们可以做到 $O(n)$ 求解。假设 $\forall i<j,x_i<x_j$ （具体公式推导的话，如果你有兴趣可以参看之后的内容。因为比较显然，这里不再讲解。）

```
int lagrange(int n, int *x, int *y, int xi) {
    int ans = 0;
    s1[0] = (xi-x[0])%mod, s2[n+1] = 1;
    for (int i = 1; i <= n; i++) s1[i] = 1ll*s1[i-1]*(xi-x[i])%mod;
    for (int i = n; i >= 0; i--) s2[i] = 1ll*s2[i+1]*(xi-x[i])%mod;
    ifac[0] = ifac[1] = 1;
    for (int i = 2; i <= n; i++) ifac[i] = -1ll*mod/i*ifac[mod%i]%mod;
    for (int i = 2; i <= n; i++) ifac[i] = 1ll*ifac[i]*ifac[i-1]%mod;
    for (int i = 0; i <= n; i++)
        (ans += 1ll*y[i]*(i == 0 ? 1 : s1[i-1])%mod*s2[i+1]%mod
            *ifac[i]%mod*(((n-i)&1) ? -1 : 1)*ifac[n-i]%mod) %= mod;
    return (ans+mod)%mod;
}
```

例题 - [BZOJ 2655]calc

自然数的幂的前缀和

问题提出

给定的 n 和 k ，求

$$\sum_{i=1}^{n}i^k$$

通常 n 比较大，而 k 只有几千或者几万。

问题解决

我们可以知道，对于上述式子，推导公式一定是是 $k+1$ 次多项式。对于证明的话，我们可以参考riteme 的介绍。

考虑使用拉格朗日插值法来获得答案多项式。

首先如果我们得知了 $n=0,1,\dots,k+1$ 处的答案 $f(n)$ ，那么给定的 n 处的答案可以写成

$$\begin{aligned}
f(n) &= \sum_{i=0}^{k+1} f(i) \frac{(n-0)(n-1)\cdots[n-(i-1)][(n-(i+1)]\cdots[n-(k+1)]}{(i-0)(i-1)\cdots[i-(i-1)][(i-(i+1)]\cdots[i-(k+1)]} \\
&= \sum_{i=0}^{k+1} f(i) \frac{\prod_{j=0}^{i-1}(n-j)\prod_{j=i+1}^{k+1}(n-j)}{i!(-1)^{k-i+1}(k+1-i)!} \\
&= \sum_{i=0}^{k+1} (-1)^{k-i+1} f(i) \frac{\prod_{j=0}^{i-1}(n-j)\prod_{j=i+1}^{k+1}(n-j)}{i!(k+1-i)!}
\end{aligned}$$

注意到后面的分式中，分子是一个前缀积乘以一个后缀积，而分母是两个阶乘。这些都可以在 $O(k)$ 的时间内求出。现在剩下的问题就是如何求出 $f(0), f(1), \ldots, f(k+1)$ 了。由于 $g(x) = x^k$ 是个完全积性函数，所以我们可以通过欧拉筛法求出 g 函数前面的一些值。具体的就是对于质数采取直接快速幂，合数则拆出任意一个因子来算，通常是欧拉筛法中可以顺便求得的最小质因子。根据素数定理，素数大约有 $O\left(\frac{k}{\ln k}\right)$ 个。每次快速幂需要花费 $O(\log k)$ 的时间，因此总的时间复杂度可以估计为 $O(k)$，是一个非常优秀的算法。上面的方法具有通用性，只要我们可以快速的求出某个 k 次多项式的前 $k+1$ 个值，那么剩下的部分可以使用拉格朗日插值法在 $O(k)$ 的时间内完成计算。

代码实现

```
int lagrange(int k, int *f, int xi) {//k+2 个点对 (i, f[i]), 0 <= i <= k+1
    int ans = 0; ++k;
    s1[0] = xi, s2[k+1] = 1;
    for (int i = 1; i <= k; i++) s1[i] = 1ll*s1[i-1]*(xi-i)%mod;
    for (int i = k; i >= 0; i--) s2[i] = 1ll*s2[i+1]*(xi-i)%mod;
    for (int i = 0; i <= k; i++)
        (ans += 1ll*f[i]*(i == 0 ? 1 : s1[i-1])%mod*s2[i+1]%mod
            *ifac[i]%mod*(((k-i)&1) ? -1 : 1)*ifac[k-i]%mod) %= mod;
    return (ans+mod)%mod;
}
void pre() {//预处理出阶乘逆元、插值的 k+2 个点
    f[1] = ifac[0] = ifac[1] = 1;
    for (int i = 2; i <= k+1; i++) ifac[i] = -1ll*mod/i*ifac[mod%i]%mod;
    for (int i = 2; i <= k+1; i++) ifac[i] = 1ll*ifac[i-1]*ifac[i]%mod;
    memset(isprime, 1, sizeof(isprime));
    for (int i = 2; i <= k+1; i++) {
        if (isprime[i]) prime[++tot] = i, f[i] = quick_pow(i, k);
        for (int j = 1; j <= tot && prime[j]*i <= k+1; j++) {
            isprime[i*prime[j]] = 0;
            f[i*prime[j]] = 1ll*f[i]*f[prime[j]]%mod;
            if (i%prime[j] == 0) break;
        }
    }
    for (int i = 1; i <= k+1; i++) f[i] = (f[i]+f[i-1])%mod;
}
void work() {
    scanf("%d", &k); pre();
    while (~scanf("%d", &n)) {
        if (n <= k+1) printf("%d\n", f[n]);
        else printf("%d\n", lagrange(k, f, n));
    }
}
```

例题 - [JLOI 2016] 成绩比较