P4247 [清华集训2012]序列操作

首先这道题从各个方面都知道是一眼线段树。

0x00 记号于约定

所有的运算符号如 $+$ 、 $=+$ 、 $==$ 等，与c++意义相同。 $\oplus$ 表示异或运算。

所有的下表从1开始。

$tree_p$ 表示第 p 个线段树节点。

$tree_p.l$ 表示第 p 个线段树节点的左边界，简记作 l ； $tree_p.r$ 表示第 p 个线段树节点的右边界，简记作 r 。这个节点的区间为 $[l,r]$ 。

定义$tree_p.size=r-l+1$为区间大小

$tree_{p_i}\ (i\in[l,r])$ 表示第 p 个线段树节点区间第 i 元素的值，简记为 ai 。

从一个数组 s 中任意选出 j 个数的序列中的第 i 数记作 s_j^i 。所有的 s_j^i 表示每一种组合。

定义 $mid=\left\lfloor\frac{(l+r)}{2}\right\rfloor$ 。

$tree_p.lChild$ 为其左子区间 $[l,mid]$ 。

$tree_p.rChild$ 为其右子区间 $[l,mid]$ 。

0x01 区间数据

区间数据于查询是息息相关的。

> 3. `Q a b c`表示询问$[a,b]$这一段区间中选择c个数相乘的所有方案的和 $\bmod 19940417$ 的值。

并且注意到

> $1\le c\le\min(b-a+1,20)$

完全可以把每个区间的每个选 i 个的答案用数组存下来记作 $tree_p.c_i$ 。

0x02 区间标记

区间标记于修改是息息相关的。

> 1. `I a b c`表示将$[a,b]$这一段区间的元素集体增加c;
> 2. `R a b`表示将$[a,b]$区间内所有元素变成相反数;

所以有两个标记：区间加与区间取反标记。记作 $tree_p.tagadd$ 和 $tree_p.tagrev$ ，简记为 $tagadd$ 和 $tagrev$ 。

若 $c==tagadd$ 表示区间中每个元素还要加上 c 。

若 $tagrev==0$ 没有修改；若 $tagrev==1$ 则表示区间需要全部取反。

并且规定 $tagrev$ 优先级高于 $tagadd$ 。

0x03 标记叠加

如果区间加 c ，则 $tagadd+=c$ 。

如果区间取反，则 $tagrev\oplus=1$ 。

> 时间复杂度 $O(1)$

0x04 区间修改

区间取反

考虑 $tree_p.c_i$ 的变化，可知 $tree_p.c_i*=(-1)^i$ 。也就是 i 为奇数时直接取反。

> 时间复杂度 $O(n)$

区间加

若有区间元素为: a_1 、 a_2 、 a_3 、 a_i ... a_{size} 。加上 x 则为 a_1+x 、 a_2+x 、 a_3+x 、 a_i+x ... $a_{size}+x$ 。

若计算 c_k 则原来为所有的 a^j 序列的 $\prod_{i=1}^{j} a_i^j$ 的和。

之后则为所有的 a^j+x 序列的 $\prod_{i=1}^{j} a_i^j+x$ 的和。

我们拿一组出来看：

$\prod_{i=1}^{j} a_i^j$ 变成了 $\prod_{i=1}^{j} a_i^j+x$

可见，这玩意展开将是一个很恐怖的多项式。但是将 x 看做元，就会好一些。

常数项也就是：$\prod_{i=1}^{j} a_i^j$ 。可以忽略，将其他的项看做对答案的贡献。

一次项就是：所有的 $\left(a^j\right)^{j-1}$ 的 $\prod_{i=1}^{j-1}\left(a^j\right)_i^{j-1}\times x$ 。

二次项就是：所有的 $\left(a^j\right)^{j-2}$ 的 $\prod_{i=1}^{j-2}\left(a^j\right)_i^{j-2}\times x^2$ 。

三次项就是：所有的 $\left(a^j\right)^{j-3}$ 的 $\prod_{i=1}^{j-3}\left(a^j\right)_i^{j-3}\times x^3$ 。

等等。

贡献就是：

所有的 $\left(a^j\right)^{j-k}$ 的 $\sum_{k=0}^{j}\prod_{i=1}^{j-k} \left(a^{j}\right)_{i}^{j-k}\times x^{k}$$

> 注意：此处的 $\left(a^j\right)^{j-k}$ 表示，从 a 数组中选 j 个数的一个组合中再选 $j-k$ 个数。

回到整体：

整体贡献就是：

所有的 $$\sum_{k=0}^{j}\prod_{i=1}^{j-k} \left(a^{j}\right)_{i}^{j-k}\times x^{k}$$

也就是 $$\sum_{k=0}^{j}\prod_{i=1}^{j-k} a_{i}^{j-k}\times x^{k}$$

其实 $\prod_{i=1}^{j-k} a_i^{j-k}$ 这个东西他就是原来的 c_{i-k} 啊！我们这样就可以倒着算出所有的 c_i 了！

> 预处理出 x^k 时间复杂度 $O(20^2)$

0x05 区间合并

到了最后，已经很好做了。考虑 $tree_p.c_i$ 可以从哪里来，其实就是两个子区，一个选 k 个，一个选 $i-k$ 个来的。

形式化一下：

$$tree_{p}.c_{i}=\sum_{k=0}^{tree_{p}.lChild.size}tree_{p}.lChild.c_{k}\times tree_{p}.rChild.c_{i-k}$$

> 时间复杂度 $O(20^2)$

0x06 代码

```
#include <cstdio>
#include <cstring>
const int mod=19940417;
#define _mid int mid=((l+r)>>1)
#define _lChild (p<<1)
#define _rChild ((p<<1)|1)
#define _lThings (_lChild),(l),(mid)
#define _rThings (_rChild),((mid)+1),(r)
#define _maxC min(20,tree[p].size)
#define INPUT_DATA_TYPE int
#define OUTPUT_DATA_TYPE long long

struct node{
    long long c[21],tagadd;
    int size;
    char tagrev;
    node(){
        tagadd=tagrev=size=0;

c[0]=c[1]=c[2]=c[3]=c[4]=c[5]=c[6]=c[7]=c[8]=c[9]=c[10]=c[11]=c[12]=c[13]=c[14]=c[15]=
c[16]=c[17]=c[18]=c[19]=c[20]=0;
    }
}tree[200001],temp__;

int Cnm[50010][21];

inline int max(int a,int b){
    return a>b?a:b;
}
inline int min(int a,int b){
    return a<b?a:b;
}

struct SEQ{
    long long temp[21];

    inline void updata(int p){
        register int i,j;

tree[p].c[0]=tree[p].c[1]=tree[p].c[2]=tree[p].c[3]=tree[p].c[4]=tree[p].c[5]=tree[p].
c[6]=tree[p].c[7]=tree[p].c[8]=tree[p].c[9]=tree[p].c[10]=tree[p].c[11]=tree[p].c[12]=
tree[p].c[13]=tree[p].c[14]=tree[p].c[15]=tree[p].c[16]=tree[p].c[17]=tree[p].c[18]=tr
ee[p].c[19]=tree[p].c[20]=0;
        for(i=0;i<=min(20,tree[_lChild].size);++i)
            for(j=0;i+j<=20&&j<=tree[_rChild].size;++j)
                tree[p].c[i+j]=
(tree[p].c[i+j]+tree[_lChild].c[i]*tree[_rChild].c[j])%mod;
        return;
    }
```

```
    void add(int p,int x){
        if(!p||!x) return;
        register int i,j;
        temp[0]=1;
        for(i=1;i<=_maxC;++i) temp[i]=temp[i-1]*x%mod;
        for(i=_maxC;i;--i)
            for(j=0;j<i;++j)
                tree[p].c[i]=(tree[p].c[i]+tree[p].c[j]*temp[i-
j]%mod*Cnm[tree[p].size-j][i-j])%mod;
        tree[p].tagadd=(tree[p].tagadd+x)%mod;
        return;
    }

    void rev(int p){
        if(!p) return;
        register int i;
        for(i=1;i<=_maxC;++i) tree[p].c[i]=((i&1)?mod-tree[p].c[i]:tree[p].c[i]);
        tree[p].tagadd=mod-tree[p].tagadd;
        tree[p].tagrev^=1;
        return;
    }

    node merge(node p,node w){
        node e;
        register int i,j;
        e.size=p.size+w.size;
        for(i=0;i<=min(20,p.size);++i)
            for(j=0;i+j<=20&&j<=w.size;++j)
                e.c[i+j]=(e.c[i+j]+p.c[i]*w.c[j])%mod;
        return e;
    }

    void push_down(int p){
        if(tree[p].tagrev){
            rev(_lChild);
            rev(_rChild);
            tree[p].tagrev=0;
        }
        if(tree[p].tagadd){
            add(_lChild,tree[p].tagadd);
            add(_rChild,tree[p].tagadd);
            tree[p].tagadd=0;
        }
        return;
    }

    void build(int p,int l,int r,int *a){
        tree[p].size=r-l+1;
        if(l==r){
            tree[p].c[0]=1;
            tree[p].c[1]=(a[l]%mod+mod)%mod;
```

```
            return;
        }
        _mid;
        build(_lThings,a);
        build(_rThings,a);
        updata(p);
        return;
    }

    void r_add(int p,int l,int r,int s,int t,int x){
        if(s<=l&&r<=t){
            add(p,x);
            return;
        }
        push_down(p);
        _mid;
        if(s<=mid) r_add(_lThings,s,t,x);
        if(mid<t) r_add(_rThings,s,t,x);
        updata(p);
        return;
    }

    void r_rev(int p,int l,int r,int s,int t){
        if(s<=l&&r<=t){
            rev(p);
            return;
        }
        push_down(p);
        _mid;
        if(s<=mid) r_rev(_lThings,s,t);
        if(mid<t) r_rev(_rThings,s,t);
        updata(p);
        return;
    }

    node query(int p,int l,int r,int s,int t){
        if(s<=l&&r<=t) return tree[p];
        push_down(p);
        _mid;
        if(t<=mid) return query(_lThings,s,t);
        if(mid<s) return query(_rThings,s,t);
        else return merge(query(_lThings,s,t),query(_rThings,s,t));
    }
}seq;

int a[50010];

int n,m;

INPUT_DATA_TYPE read();
void print(OUTPUT_DATA_TYPE x);
```

```
int main(){
    register int i,j,k,l,r,x;
    register char op;
    n=read();
    m=read();
    tree[0].c[0]=1;
    Cnm[0][0]=1;
    for(i=1;i<=n;++i){
        a[i]=read();
        Cnm[i][0]=1;
        for(j=1;j<=20&&j<=i;++j) Cnm[i][j]=(Cnm[i-1][j]+Cnm[i-1][j-1])%mod;
    }
    seq.build(1,1,n,a);
    for(i=1;i<=m;++i){
        loop:op=getchar();
        while(op!='I'&&op!='Q'&&op!='R')goto loop;
        l=read();
        r=read();
        if(op=='I'){
            x=read();
            x=(x%mod+mod)%mod;
            seq.r_add(1,1,n,l,r,x);
        }else if(op=='R'){
            seq.r_rev(1,1,n,l,r);
        }else{
            x=read();
            print(((seq.query(1,1,n,l,r)).c[x]%mod+mod)%mod);
            putchar('\n');
        }
    }

    return 0;
}

INPUT_DATA_TYPE read(){
    register INPUT_DATA_TYPE x=0;register char f=0,c=getchar();
    while(c<'0'||'9'<c)f=(c=='-'),c=getchar();//?=if,:=else
    while('0'<=c&&c<='9')x=(x<<3)+(x<<1)+(c&15),c=getchar();
    return f?-x:x;
}

void print(OUTPUT_DATA_TYPE x){
    register char s[20];
    register int i=0;
    if(x<0){
        x=-x;
        putchar('-');
    }
    if(x==0){
        putchar('0');
```

```
        return;
    }
    while(x){
        s[i++]=x%10;
        x/=10;
    }
    while(i){
        putchar(s[--i]+'0');
    }
    return;
}
```

终于写完了，完结撒花！！！！本文可能有点抽象，有什么不懂可以问我，我会很乐意的！