Harbour.Space Scholarship Contest 2021-2022 (Div. 1 + Div. 2)

比赛链接

F. Pairwise Modulo

题意

给定长度为 n 且互不相同序列 A，对每个 $k=1\sim n$ 计算

$$ f(k)=\sum_{i=1}^k\sum_{j=1}^k a_i\bmod a_j $$

题解

不难发现

$$ f(k)=f(k-1)+\sum_{i=1}^k a_i\bmod a_k+\sum_{i=1}^k a_k\bmod a_i\\ \sum_{i=1}^k a_i\bmod a_k=\sum_{i=1}^{k-1}a_i-\sum_{i=1}^{k-1}\lfloor \frac {a_i}{a_k}\rfloor a_k\\ \sum_{i=1}^k a_k\bmod a_i=(k-1)a_k-\sum_{i=1}^{k-1}\lfloor \frac {a_k}{a_i}\rfloor a_i\\ $$

对于 $\sum_{i=1}^{k-1}\lfloor \frac {a_i}{a_k}\rfloor a_k$，显然可以枚举 $d=\lfloor \frac {a_i}{a_k}\rfloor$ 的取值，然后根据区间 $[d\times a_k,(d+1)\times a_k)$ 中数值的个数计算贡献。

对于 $\sum_{i=1}^{k-1}\lfloor \frac {a_k}{a_i}\rfloor a_i$，一个暴力的想法是利用整数分块枚举 $d=\lfloor\frac {a_k}{a_i}\rfloor$，然后根据区间 $[d\times a_k,(d+1)\times a_k)$ 中数值的和计算贡献。

询问次数 $O(n\log n+n\sqrt v)$，于是总时间复杂度 $O\left(n\log^2 n+n\sqrt v\log2 v\right)$，非常卡常，据说能过，~~但我没过。~~

```
const int MAXN=2e5+5,MAXM=3e5+5;
struct BIT{
    #define lowbit(x) ((x)&(-x))
    LL c[MAXM];
    void add(int pos,int v){
        while(pos<MAXM){
            c[pos]+=v;
            pos+=lowbit(pos);
        }
    }
    LL query(int pos){
        LL ans=0;
        while(pos){
            ans+=c[pos];
            pos-=lowbit(pos);
```

```
        }
        return ans;
    }
    LL query(int L,int R){
        return query(R)-query(L-1);
    }
}tree1,tree2;
int a[MAXN];
int main()
{
    int n=read_int();
    _for(i,0,n)a[i]=read_int();
    LL ans=0,pre=0;
    _for(i,0,n){
        ans+=pre;
        for(int j=1;a[i]*j<MAXM;j++)
        ans-=1LL*j*a[i]*tree1.query(a[i]*j,min(a[i]*(j+1),MAXM)-1);
        pre+=a[i];
        tree1.add(a[i],1);

        ans+=1LL*i*a[i];
        int lef=1,rig=0;
        while(lef<=a[i]){
            rig=a[i]/(a[i]/lef);
            ans-=(a[i]/lef)*tree2.query(lef,rig);
            lef=rig+1;
        }
        tree2.add(a[i],a[i]);
        space(ans);
    }
    return 0;
}
```

优化一

不难发现查询次数是 $O(n\sqrt v)$，更新次数是 $O(n)$，考虑利用分块构造 $O(1)$ 查询 $O(\sqrt v)$ 更新的数据结构。

具体的，$O(\sqrt v)$ 维护每个块内部的前缀和以及所有块的前缀和，即可支持 $O(1)$ 查询。总时间复杂度 $O(n\sqrt v)$。

```
const int MAXN=2e5+5,MAXM=3e5+5,S=600;
struct BIT{
    #define lowbit(x) ((x)&(-x))
    LL c[MAXM];
    void add(int pos,int v){
        while(pos<MAXM){
            c[pos]+=v;
```

```
            pos+=lowbit(pos);
        }
    }
    LL query(int pos){
        LL ans=0;
        while(pos){
            ans+=c[pos];
            pos-=lowbit(pos);
        }
        return ans;
    }
    LL query(int L,int R){
        return query(R)-query(L-1);
    }
}tree;
int a[MAXN];
LL s1[S][S],s2[S];
LL query(int pos){
    LL ans=s1[pos/S][pos%S];
    if(pos>=S)
    ans+=s2[pos/S-1];
    return ans;
}
LL query(int L,int R){
    return query(R)-query(L-1);
}
void update(int pos,int v){
    int idx=pos/S;
    _for(i,pos%S,S)
    s1[idx][i]+=v;
    _for(i,idx,S)
    s2[i]+=v;
}
int main()
{
    int n=read_int();
    _for(i,0,n)a[i]=read_int();
    LL ans=0,pre=0;
    _for(i,0,n){
        ans+=pre;
        for(int j=1;a[i]*j<MAXM;j++)
        ans-=1LL*j*a[i]*tree.query(a[i]*j,min(a[i]*(j+1),MAXM)-1);
        pre+=a[i];
        tree.add(a[i],1);

        ans+=1LL*i*a[i];
        int lef=1,rig=0;
        while(lef<=a[i]){
            rig=a[i]/(a[i]/lef);
            ans-=(a[i]/lef)*query(lef,rig);
            lef=rig+1;
```

```
        }
        update(a[i],a[i]);
        space(ans);
    }
    return 0;
}
```

优化二

对于 $\sum_{i=1}^{k-1}\lfloor \frac {a_k}{a_i}\rfloor a_i$ 提前处理好 a_i 对值域的贡献。

具体的，对 $[d\times a_i,(d+1)\times a_i)$ a_i 的贡献为 $d\times a_i$ 于是更新次数 $O(\log v)$ 查询次数 $O(1)$ 时间复杂度 $O(n\log^2 v)$

```
const int MAXN=2e5+5,MAXM=3e5+5;
struct BIT{
    #define lowbit(x) ((x)&(-x))
    LL c[MAXM];
    void add(int pos,int v){
        while(pos<MAXM){
            c[pos]+=v;
            pos+=lowbit(pos);
        }
    }
    void update(int L,int R,int v){
        add(L,v);
        add(R+1,-v);
    }
    LL query(int pos){
        LL ans=0;
        while(pos){
            ans+=c[pos];
            pos-=lowbit(pos);
        }
        return ans;
    }
    LL query(int L,int R){
        return query(R)-query(L-1);
    }
}tree1,tree2;
int a[MAXN];
int main()
{
    int n=read_int();
    _for(i,0,n)a[i]=read_int();
    LL ans=0,pre=0;
    _for(i,0,n){
        ans+=pre;
```

```
        for(int j=1;a[i]*j<MAXM;j++)
        ans-=1LL*j*a[i]*tree1.query(a[i]*j,min(a[i]*(j+1),MAXM)-1);
        pre+=a[i];
        tree1.add(a[i],1);

        ans+=1LL*i*a[i];
        ans-=tree2.query(a[i]);
        for(int j=a[i];j<MAXM;j+=a[i])
        tree2.update(j,min(j+a[i],MAXM)-1,j);
        space(ans);
    }
    return 0;
}
```