数论 3

杜教筛

算法简介

一种 $O\left(n^{\frac 23}\right)$ 计算积性函数前缀和的算法。

算法思路

设 f、g 为积性函数，$S(n)=\sum_{i=1}^n f(i)$，考虑 f、g 的狄利克雷卷积的前缀和

\begin{equation}\sum_{i=1}^n (f\ast g)(i)=\sum_{i=1}^n\sum_{d\mid i}f(\frac id)g(d)=\sum_{d=1}^n \left(g(d)\sum_{k=1}^{\lfloor\frac nd\rfloor}f(k)\right)=\sum_{d=1}^n g(d)S(\lfloor\frac nd\rfloor)\end{equation}

所以有

\begin{equation}\sum_{i=1}^n (f\ast g)(i)=g(1)S(n)+\sum_{d=2}^n g(d)S(\lfloor\frac nd\rfloor)\end{equation}

移项得

\begin{equation}g(1)S(n)=\sum_{i=1}^n (f\ast g)(i)-\sum_{d=2}^n g(d)S(\lfloor\frac nd\rfloor)\tag{1}\end{equation}

观察式子，发现如果能快速求出 $(f\ast g)(n)$ 和 $g(n)$ 的前缀和，就可以通过整数分块和记忆化搜索快速求出 $S(n)$。

复杂度证明

下面假设 $(f\ast g)(n)$ 和 $g(n)$ 的前缀和可以 $O(1)$ 求出。

若要求出 $S(n)$，需要先求出 $S(\lfloor\frac nd\rfloor)(d=2\sim n)$。

事实上，有 $\{x|\exists d\left((2\le d\le n)\land \left(\lfloor\frac nd\rfloor=x\right)\right)\}\subseteq \{1,2,3\cdots \lfloor\sqrt n\rfloor\}\cup\{\lfloor\frac n2\rfloor,\lfloor\frac n3\rfloor,\lfloor\frac n4\rfloor\cdots \lfloor\frac n{\lfloor\sqrt n\rfloor}\rfloor\}$。

对 $m\in \{x|\exists d\left((2\le d\le n)\land \left(\lfloor\frac nd\rfloor=x\right)\right)\}$，有 \begin{equation}\{1,2,3\cdots \lfloor\sqrt m\rfloor\}\cup\{\lfloor\frac m2\rfloor,\lfloor\frac m3\rfloor,\lfloor\frac m4\rfloor\cdots \lfloor\frac m{\lfloor\sqrt m\rfloor}\rfloor\} \subset\{1,2,3\cdots \lfloor\sqrt n\rfloor\}\cup\{\lfloor\frac n2\rfloor,\lfloor\frac n3\rfloor,\lfloor\frac n4\rfloor\cdots \lfloor\frac n{\lfloor\sqrt n\rfloor}\rfloor\}\end{equation}

因为首先 $m\lt n$，于是

\begin{equation}\{1,2,3\cdots \lfloor\sqrt m\rfloor\}\subset\{1,2,3\cdots \lfloor\sqrt

n\rfloor\}\end{equation}

设 $m=\lfloor\frac nd\rfloor$有

\begin{equation}\{\lfloor\frac n{2d}\rfloor,\lfloor\frac n{3d}\rfloor,\lfloor\frac n{4d}\rfloor\cdots \lfloor\frac n{\lfloor\sqrt m\rfloor d}\rfloor\} \subset \{\lfloor\frac n2\rfloor,\lfloor\frac n3\rfloor,\lfloor\frac n4\rfloor\cdots \lfloor\frac n{\lfloor\sqrt n\rfloor}\rfloor\}\end{equation}

所以记忆化搜索只需要求出最开始的 $O(\sqrt n)$ 个状态，即 $\{1,2,3\cdots \lfloor\sqrt n\rfloor\}\cup\{\lfloor\frac n2\rfloor,\lfloor\frac n3\rfloor,\lfloor\frac n4\rfloor\cdots \lfloor\frac n{\lfloor\sqrt n\rfloor}\rfloor\}$

根据整数分块，每个状态统计答案的时间复杂度为 $O(\sqrt n)$总时间复杂度为

\begin{equation}\sum_{i=1}^{\lfloor\sqrt n\rfloor}\left(O(\sqrt i)+O\left(\sqrt {\frac ni}\right)\right)=O\left(\int_{x=1}^{\sqrt n}\sqrt x+\sqrt {\frac nx}\mathrm{d}x\right)=O\left(n^{\frac 34}\right)\end{equation}

考虑线性筛预处理前 k 个前缀和 $(k\ge \sqrt n)$

总时间复杂度变为

\begin{equation}O(k)+\sum_{i=1}^{\lfloor\sqrt {\frac nk}\rfloor}O\left(\sqrt {\frac ni}\right)=O(k)+O\left(\int_{x=1}^{\sqrt {\frac nk}}\sqrt {\frac nx}\mathrm{d}x\right)=O(k)+O\left(\frac n{\sqrt k}\right)\end{equation}

发现取 $k\sim n^{\frac 23}$ 时可以达到最佳时间复杂度 $O\left(n^{\frac 23}\right)$

另外关于记忆化搜索的答案，建议用哈希表存储。

算法练习

习题一

洛谷p4213

题意

给定正整数 n求

\begin{equation}\text{ans}_1=\sum_{i=1}^n\varphi(i)\end{equation}

\begin{equation}\text{ans}_2=\sum_{i=1}^n\mu(i)\end{equation}

题解

取 $f=\varphi,g=I$则$(f\ast g)=id$根据 (1) 式，有

\begin{equation}I(1)S(n)=\sum_{i=1}^n id(i)-\sum_{d=2}^n I(d)S(\lfloor\frac nd\rfloor)\end{equation}

即

\begin{equation}S(n)=\frac {n(n+1)}2-\sum_{d=2}^n S(\lfloor\frac nd\rfloor)\end{equation}

取 $f=\mu,g=I$则$(f\ast g)=e$根据 (1) 式，有

\begin{equation}I(1)S(n)=\sum_{i=1}^n e(i)-\sum_{d=2}^n I(d)S(\lfloor\frac nd\rfloor)\end{equation}

即

\begin{equation}S(n)=1-\sum_{d=2}^n S(\lfloor\frac nd\rfloor)\end{equation}

```
#include <iostream>
#include <cstdio>
#include <cstdlib>
#include <algorithm>
#include <string>
#include <sstream>
#include <cstring>
#include <cctype>
#include <cmath>
#include <vector>
#include <set>
#include <map>
#include <stack>
#include <queue>
#include <ctime>
#include <cassert>
#define _for(i,a,b) for(int i=(a);i<(b);++i)
#define _rep(i,a,b) for(int i=(a);i<=(b);++i)
#define mem(a,b) memset(a,b,sizeof(a))
using namespace std;
typedef long long LL;
inline int read_int(){
    int t=0;bool sign=false;char c=getchar();
    while(!isdigit(c)){sign|=c=='-';c=getchar();}
    while(isdigit(c)){t=(t<<1)+(t<<3)+(c&15);c=getchar();}
    return sign?-t:t;
}
inline LL read_LL(){
    LL t=0;bool sign=false;char c=getchar();
    while(!isdigit(c)){sign|=c=='-';c=getchar();}
    while(isdigit(c)){t=(t<<1)+(t<<3)+(c&15);c=getchar();}
    return sign?-t:t;
}
inline char get_char(){
    char c=getchar();
```

```
    while(c==' '||c=='\n'||c=='\r')c=getchar();
    return c;
}
inline void write(LL x){
    register char c[21],len=0;
    if(!x)return putchar('0'),void();
    if(x<0)x=-x,putchar('-');
    while(x)c[++len]=x%10,x/=10;
    while(len)putchar(c[len--]+48);
}
inline void space(LL x){write(x),putchar(' ');}
inline void enter(LL x){write(x),putchar('\n');}
const int MAXP=5e6+5;
bool vis[MAXP];
int prime[MAXP],mu[MAXP],cnt;
LL phi[MAXP];
template <typename T1,typename T2>
struct HASH_Table{
    static const int HASH_MOD=3000017,MAXS=5e6;
    struct cell{
        T1 key;T2 val;
        int next;
    }e[MAXS];
    int head[HASH_MOD],cnt;
    void clear(){mem(head,0);cnt=0;}
    T2 insert(T1 Key,T2 Value){
        int h=Key%HASH_MOD;
        e[++cnt].key=Key,e[cnt].val=Value,e[cnt].next=head[h];
        head[h]=cnt;
        return Value;
    }
    T2 find(T1 Key){
        int h=Key%HASH_MOD;
        for(int i=head[h];i;i=e[i].next){
            if(e[i].key==Key)
            return e[i].val;
        }
        return -1;
    }
};
HASH_Table<int,int> S_Mu;
HASH_Table<int,LL> S_Phi;
void Pre(){
    vis[1]=true,mu[1]=1,phi[1]=1;
    _for(i,2,MAXP){
        if(!vis[i])mu[i]=-1,phi[i]=i-1,prime[cnt++]=i;
        for(int j=0;j<cnt&&i*prime[j]<MAXP;j++){
            vis[i*prime[j]]=true;
            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;
            }
        }
    }
    _for(i,2,MAXP)
    mu[i]+=mu[i-1],phi[i]+=phi[i-1];
}
int S_mu(int n){
    if(n<MAXP)
    return mu[n];
    if(S_Mu.find(n)!=-1)
    return S_Mu.find(n);
    int ans=1,lef=2,rig;
    while(lef<=n){
        rig=n/(n/lef);
        ans-=(rig-lef+1)*S_mu(n/lef);
        lef=rig+1;
    }
    return S_Mu.insert(n,ans);
}
LL S_phi(int n){
    if(n<MAXP)
    return phi[n];
    if(S_Phi.find(n)!=-1)
    return S_Phi.find(n);
    LL ans=1LL*n*(n+1)/2;
    int lef=2,rig;
    while(lef<=n){
        rig=n/(n/lef);
        ans-=1LL*(rig-lef+1)*S_phi(n/lef);
        lef=rig+1;
    }
    return S_Phi.insert(n,ans);
}
int main()
{
    int t=read_int(),n;
    Pre();
    while(t--){
        n=read_int();
        space(S_phi(n));
        enter(S_mu(n));
    }
    return 0;
}
```