\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Aho-Corasick automaton

struct node

{

node \*next[26];

node \*fail;

int sum;

};

node \*root,\*newnode,\*q[N];

int cnt,head,tail;

void insert(char \*s)

{

node \*p=root;

for(int i=0;s[i];i++){

int x=s[i]-'a';

if(p->next[x]==NULL){

newnode=(struct node\*)malloc(sizeof(struct node));

for(int j=0;j<26;j++)newnode->next[j]=NULL;

newnode->sum=0;newnode->fail=NULL;

p->next[x]=newnode;

}

p=p->next[x];

}

p->sum++;

}

void build\_fail()

{

head=0;tail=1;q[head]=root;

node \*p,\*temp;

while(head<tail){

temp=q[head++];

for(int i=0;i<26;i++){

if(temp->next[i]){

if(temp==root) temp->next[i]->fail=root;

else{

p=temp->fail;

while(p){

if(p->next[i]){temp->next[i]->fail=p->next[i];break;}

p=p->fail;

}

if(p==NULL)temp->next[i]->fail=root;

}

q[tail++]=temp->next[i];

}

}

}

}

void ac\_automation(char \*ch)

{

node \*p=root;

for(int i=0;ch[i];i++){

int x=ch[i]-'a';

while(!p->next[x]&&p!=root)p=p->fail;

p=p->next[x];

if(!p)p=root;

node \*temp=p;

while(temp!=root){

if(temp->sum>=0){

cnt+=temp->sum;

temp->sum=-1;

}

else break;

temp=temp->fail;

}

}

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Suffix Automaton

struct SAM{

int ch[N][26],tot,dis[N],fa[N],root,last,sg[N];

char a[N];

bool init(){

memset(ch,0,sizeof(ch));memset(dis,0,sizeof(dis));

memset(fa,0,sizeof(fa));memset(sg,-1,sizeof(sg));

tot=0;root=last=++tot;return true;

}

void build(){

int n=strlen(a+1);

for(int i=1;i<=n;i++)add(i);

}

void add(int pos){

int x=a[pos]-'a',p=last,np=++tot;

last=np,dis[np]=pos;

for(;p&&!ch[p][x];p=fa[p])ch[p][x]=np;

if(!p)fa[np]=root;

else{

int q=ch[p][x];

if(dis[q]==dis[p]+1)fa[np]=q;

else{

int nq=++tot;

dis[nq]=dis[p]+1;

memcpy(ch[nq],ch[q],sizeof(ch[q]));

fa[nq]=fa[q],fa[np]=fa[q]=nq;

for(;ch[p][x]==q;p=fa[p])ch[p][x]=nq;

}

}

}

//Longest Commom Substring

int ask(char b[]){

int ans=0,len=0,p=root,m=strlen(b+1);

for(int i=1;i<=m;i++){

int x=b[i]-'a';

if(ch[p][x])len++,p=ch[p][x];

else{

while(p&&!ch[p][x])p=fa[p];

if(!p)p=root,len=0;

else len=dis[p]+1,p=ch[p][x];

}ans=max(ans,len);

}

return ans;

}

//sg function

void getsg(int p){

if(sg[p]!=-1)return ;

bool vis[31];//can change 31

for(int i=0;i<=30;i++)vis[i]=false;

for(int i=0;i<26;i++){

if(ch[p][i]){

if(sg[ch[p][i]]==-1)getsg(ch[p][i]);

vis[sg[ch[p][i]]]=1;

}

}

for(int i=0;i<=30;i++) if(!vis[i]){sg[p]=i;return ;}

}

int calsg(char b[]){

int m=strlen(b+1),p=1;

for(int i=1;i<=m;i++)p=ch[p][b[i]-'a'];

return sg[p];

}

}T;

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Mobius function

void init\_mus()

{

cnt=0;memset(vis,0,sizeof(vis));

mus[1]=1;

for(int i=2;i<N;i++){

if(!vis[i]){prime[cnt++]=i;mus[i]=-1;}

for(int j=0;j<cnt&&i\*prime[j]<N;j++){

vis[i\*prime[j]]=1;

if(i%prime[j]==0){mus[i\*prime[j]]=0;break;}

else mus[i\*prime[j]]=-mus[i];

}

}

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Euler function

void init\_euler()

{

memset(prime,0,sizeof(p));

memset(phi,0,sizeof(phi));

memset(vis,0,sizeof(vis));

for(int i=2;i<N;i++){

if(!vis[i]){prime[++prime[0]]=i;phi[i]=i-1;}

for(int j=1;j<=prime[0];j++){

if(i\*prime[j]>=N)break;

vis[i\*prime[j]]=true;

if(!(i%prime[j])){phi[i\*prime[j]]=phi[i]\*prime[j];break;}

phi[i\*prime[j]]=phi[i]\*(prime[j]-1);

}

}phi[1]=1;

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

x\*(x+1)/2 <==> mult(x,x+1,p)\*modexp(2,p-2,p)%p

inline ll mult(ll a,ll b,ll p)

{

ll ans=0;

while(b){if(b&1)ans=(ans+a)%p;b>>=1;a=(a+a)%p;}

return ans;

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

IO

namespace IO {

const int MT = 10 \* 1024 \* 1024; /// 10MB 请注意输入数据的大小！！！

char IO\_BUF[MT];

int IO\_PTR, IO\_SZ;

/// 要记得把这一行添加到main函数第一行！！！

void begin() {

IO\_PTR = 0;

IO\_SZ = fread (IO\_BUF, 1, MT, stdin);

}

template<typename T>

inline bool scan\_d (T & t) {

while (IO\_PTR < IO\_SZ && IO\_BUF[IO\_PTR] != '-' && (IO\_BUF[IO\_PTR] < '0' || IO\_BUF[IO\_PTR] > '9'))

IO\_PTR ++;

if (IO\_PTR >= IO\_SZ) return false;

bool sgn = false;

if (IO\_BUF[IO\_PTR] == '-') sgn = true, IO\_PTR ++;

for (t = 0; IO\_PTR < IO\_SZ && '0' <= IO\_BUF[IO\_PTR] && IO\_BUF[IO\_PTR] <= '9'; IO\_PTR ++)

t = t \* 10 + IO\_BUF[IO\_PTR] - '0';

if (sgn) t = -t;

return true;

}

inline bool scan\_s (char s[]) {

while (IO\_PTR < IO\_SZ && (IO\_BUF[IO\_PTR] == ' ' || IO\_BUF[IO\_PTR] == '\n') ) IO\_PTR ++;

if (IO\_PTR >= IO\_SZ) return false;

int len = 0;

while (IO\_PTR < IO\_SZ && IO\_BUF[IO\_PTR] != ' ' && IO\_BUF[IO\_PTR] != '\n')

s[len ++] = IO\_BUF[IO\_PTR], IO\_PTR ++;

s[len] = '\0';

return true;

}

template<typename T>

void print(T x) {

static char s[33], \*s1; s1 = s;

if (!x) \*s1++ = '0';

if (x < 0) putchar('-'), x = -x;

while(x) \*s1++ = (x % 10 + '0'), x /= 10;

while(s1-- != s) putchar(\*s1);

}

template<typename T>

void println(T x) {

print(x); putchar('\n');

}

};

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Maxflow

struct Dinic

{

int n,m,s,t;

vector<Edge> edge;

vector<int> G[N];

bool vis[N];

ll d[N];

int cur[N];

void init()

{

for(int i=0;i<N;i++)

G[i].clear();

edge.clear();

memset(d,0,sizeof(d));

memset(vis,0,sizeof(vis));

memset(cur,0,sizeof(cur));

}

void addEdge (int from,int to,ll cap)

{

edge.push\_back((Edge){from,to,cap,0});

edge.push\_back((Edge){to,from,0,0});

m = edge.size();

G[from].push\_back(m-2);

G[to].push\_back(m-1);

}

bool BFS()

{

memset(vis,0,sizeof(vis));

queue<int> Q;

Q.push(s);

d[s] = 0;

vis[s] = 1;

while(!Q.empty())

{

int x = Q.front();

Q.pop();

for(int i=0; i<(int)G[x].size(); i++)

{

Edge & e = edge[G[x][i]];

if(!vis[e.to]&&e.cap>e.flow)

{

vis[e.to] = 1;

d[e.to] = d[x] + 1;

Q.push(e.to);

}

}

}

return vis[t];

}

ll DFS(int x,ll a)

{

if(x==t||a==0) return a;

ll flow = 0,f;

for(int & i = cur[x]; i<(int)G[x].size(); i++)

{

Edge & e = edge[G[x][i]];

if(d[x] + 1==d[e.to]&&(f=DFS(e.to,min(a,e.cap-e.flow)))>0)

{

e.flow +=f;

edge[G[x][i]^1].flow -=f;

flow +=f;

a-=f;

if(a==0) break;

}

}

return flow;

}

ll Maxflow (int s,int t) {

this->s = s;this->t = t;

ll flow = 0;

while(BFS()) {

memset(cur,0,sizeof(cur));

flow+=DFS(s,INF);

}

return flow;

}

//求最小割S,T;

void new\_BFS(int s,int n)

{

memset(vis,0,sizeof(vis));

d[s] = 0;

vis[s] = 1;

queue<int> Q;

Q.push(s);

while(!Q.empty())

{

int u = Q.front();

Q.pop();

for(int i=0;i<(int)G[u].size();i++)

{

Edge & e = edge[G[u][i]];

if(!vis[e.to]&&e.cap>e.flow)

{

vis[e.to] = 1;

d[e.to] = d[u] + 1;

Q.push(e.to);

}

}

}

int cnt = 0;

for(int i=1;i<=n;i++)

{

if(vis[i]) cnt++;

}

printf("%d\n",cnt);

for(int i=1;i<=n;i++)

if(vis[i]) printf("%d ",i);

puts("");

}

}sol;

MincotMaxflow

struct Edge

{

int from,to;ll cap,flow,cost;

Edge(int u,int v,ll ca,ll f,ll co):from(u),to(v),cap(ca),flow(f),cost(co){};

};

struct MCMF

{

int n,m,s,t;

vector<Edge> edges;

vector<int> G[N];

int inq[N];//是否在队列中

ll d[N];//距离

int p[N];//上一条弧

ll a[N];//可改进量

void init(int n)//初始化

{

this->n=n;

for(int i=0;i<n;i++)

G[i].clear();

edges.clear();

}

void addedge(int from,int to,ll cap,ll cost)//加边

{

edges.push\_back(Edge(from,to,cap,0,cost));

edges.push\_back(Edge(to,from,0,0,-cost));

int m=edges.size();

G[from].push\_back(m-2);

G[to].push\_back(m-1);

}

bool SPFA(int s,int t,ll &flow,ll &cost)//寻找最小费用的增广路，使用引用同时修改原flow,cost

{

for(int i=0;i<n;i++)

d[i]=INF;

memset(inq,0,sizeof(inq));

d[s]=0;inq[s]=1;p[s]=0;a[s]=INF;

queue<int> Q;

Q.push(s);

while(!Q.empty())

{

int u=Q.front();

Q.pop();

inq[u]--;

for(int i=0;i<(int)G[u].size();i++)

{

Edge& e=edges[G[u][i]];

if(e.cap>e.flow && d[e.to]>d[u]+e.cost)//满足可增广且可变短

{

d[e.to]=d[u]+e.cost;

p[e.to]=G[u][i];

a[e.to]=min(a[u],e.cap-e.flow);

if(!inq[e.to])

{

inq[e.to]++;

Q.push(e.to);

}

}

}

}

if(d[t]==INF) return false;//汇点不可达则退出

flow+=a[t];

cost+=d[t]\*a[t];

int u=t;

while(u!=s)//更新正向边和反向边

{

edges[p[u]].flow+=a[t];

edges[p[u]^1].flow-=a[t];

u=edges[p[u]].from;

}

return true;

}

ll MincotMaxflow(int s,int t)

{

ll flow=0,cost=0;

while(SPFA(s,t,flow,cost));

return cost;

}

}sol;

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Hamiltonian path

for(int k=1;k<=(1<<n)-1;k++)

for(int i=1;i<=n;i++)

for(int j=1;j<=n;j++)

if((1<<(i-1))&k)dp[i][k]=min(dp[i][k],dp[j][k-(1<<(i-1))]+dis[j][i]);

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

计算几何

#define P vector<double>

P three\_plane(P a,P b,P c)//三点求平面 Ax+By+Cz+D=0;

{

double x1,x2,x3,y1,y2,y3,z1,z2,z3;

x1=a[0];x2=b[0];x3=c[0];

y1=a[1];y2=b[1];y3=c[1];

z1=a[2];z2=b[2];z3=c[2];

double A = y1\*(z2-z3)+y2\*(z3-z1)+y3\*(z1-z2);

double B = z1\*(x2-x3)+z2\*(x3-x1)+z3\*(x1-x2);

double C = x1\*(y2-y3)+x2\*(y3-y1)+x3\*(y1-y2);

double D = -(x1\*(y2\*z3-y3\*z2)+ x2\*(y3\*z1-y1\*z3) + x3\*(y1\*z2 -y2\*z1) );

P ans;ans={A,B,C,D};

return ans;

}

pair<double,P> point\_area(P a,P pla)//点到面的距离和投影 pla 是平面法向量 {A,B,C}

{

double A,B,C,D;A=pla[0],B=pla[1],C=pla[2];D=pla[3];

double x0,y0,z0;x0=a[0],y0=a[1],z0=a[2];

double d=fabs(A\*x0+B\*y0+C\*z0+D)/sqrt(A\*A+B\*B+C\*C);

double t=(A\*x0+B\*y0+C\*z0+D)/(A\*A+B\*B+C\*C);

double x,y,z;

x=x0-A\*t;y=y0-B\*t;z=z0-C\*t;

pair<double,P> ans;

ans={d,{x,y,z}};

return ans;

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

base

struct L\_B{

long long d[61],p[61];

int cnt;

L\_B()

{

memset(d,0,sizeof(d));

memset(p,0,sizeof(p));

cnt=0;

}

bool insert(long long val)

{

for (int i=60;i>=0;i--)

if (val&(1LL<<i))

{

if (!d[i])

{

d[i]=val;

break;

}

val^=d[i];

}

return val>0;

}

long long query\_max()

{

long long ret=0;

for (int i=60;i>=0;i--)

if ((ret^d[i])>ret)

ret^=d[i];

return ret;

}

long long query\_min()

{

for (int i=0;i<=60;i++)

if (d[i])

return d[i];

return 0;

}

void rebuild()

{

for (int i=60;i>=0;i--)

for (int j=i-1;j>=0;j--)

if (d[i]&(1LL<<j))

d[i]^=d[j];

for (int i=0;i<=60;i++)

if (d[i])

p[cnt++]=d[i];

}

long long kthquery(long long k)

{

int ret=0;

if (k>=(1LL<<cnt))

return -1;

for (int i=60;i>=0;i--)

if (k&(1LL<<i))

ret^=p[i];

return ret;

}

}

L\_B merge(const L\_B &n1,const L\_B &n2)

{

L\_B ret=n1;

for (int i=60;i>=0;i--)

if (n2.d[i])

ret.insert(n1.d[i]);

return ret;

}

中国剩余定理 x%m=a，m为质数

ll m[maxn],a[maxn];

void gcd(ll a,ll b,ll &d,ll &x,ll &y)

{

if(b==0)

{

d=a;

x=1,y=0;

}

else

{

gcd(b,a%b,d,y,x);

y-=(a/b)\*x;

}

}

ll china(ll n,ll m[],ll a[])

{

ll M=1,d,y,x=0;

for(int i=0;i<n;i++) M\*=m[i];

for(int i=0;i<n;i++)

{

ll w=M/m[i];

gcd(m[i],w,d,d,y);

x=(x+y\*w\*a[i])%M;

}

return (x+M)%M;

}

int main()

{

ll n;

scanf("%I64d",&n);

for(int i=0;i<n;i++)

scanf("%I64d %I64d",&m[i],&a[i]);

printf("%I64d\n",china(n,m,a));

}

欧拉函数之和 S(n) = Phi(1) + Phi(2) + ...... Phi(n) n<=10^10

#define P pair<pair<int,int>,int>

int cnt,euler[N];

ll pre[N],inv;

int modexp(ll a,ll b,ll p)

{

ll ans=1;a%=b;

while(b){if(b&1)ans=ans\*a%p;b>>=1;a=a\*a%p;}

return ans;

}

void init\_euler()

{

euler[1]=1;

for(int i=2;i<N;i++)euler[i]=i;

for(int i=2;i<N;i++)

if(euler[i]==i)

for(int j=i;j<N;j+=i)

euler[j]=euler[j]/i\*(i-1);

pre[0]=0;

for(int i=1;i<N;i++)pre[i]=(pre[i-1]+euler[i])%p;

}

map<ll,ll>m;

inline ll cal(ll x)

{

if(x<N)return pre[x];

if(m.count(x))return m[x];

ll sum=0;

for(ll i=1;i\*i<=x;i++){

sum+=pre[i]\*((x/i-x/(i+1LL))%p)%p;sum%=p;

if(i!=1LL&&x/i!=i){

ll tmp=0;

if(x/i<N)tmp=pre[x/i]\*((x/(x/i)-x/(x/i+1LL))%p)%p;

else tmp=(m.count(x)==0?cal(x/i):m[x/i])\*((x/(x/i)-x/(x/i+1LL))%p)%p;

sum+=tmp;

}

sum%=p;

}

return m[x]=(((x%p)\*((x+1LL)%p)%p)\*inv%p-sum+p)%p;

}

莫比乌斯函数之和 S(a,b) = miu(a) + miu(a + 1) + ...... miu(b) a,b<=10^10

#define P pair<pair<int,int>,int>

int cnt,prime[N],vis[N],mus[N];

int pre[N];

void init\_mus()

{

cnt=0;memset(vis,0,sizeof(vis));

mus[1]=1;

for(int i=2;i<N;i++){

if(!vis[i]){prime[cnt++]=i;mus[i]=-1;}

for(int j=0;j<cnt&&i\*prime[j]<N;j++){

vis[i\*prime[j]]=1;

if(i%prime[j]==0){mus[i\*prime[j]]=0;break;}

else mus[i\*prime[j]]=-mus[i];

}

}

pre[0]=0;for(int i=1;i<N;i++)pre[i]=pre[i-1]+mus[i];

}

map<ll,ll>m;

inline ll cal(ll x)

{

if(x<N)return pre[x];

if(m.count(x))return m[x];

ll sum=0;

for(ll i=1;i\*i<=x;i++){

sum+=pre[i]\*(x/i-x/(i+1LL));

if(i!=1LL&&x/i!=i){

ll tmp=0;

if(x/i<N)tmp=pre[x/i]\*(x/(x/i)-x/(x/i+1LL));

tmp=(m.count(x)==0?cal(x/i):m[x/i])\*(x/(x/i)-x/(x/i+1LL));

sum+=tmp;

}

}

m[x]=1-sum;

return 1-sum;

}

最小公倍数计数

给出一个区间[a,b]，求最小公倍数在这个区间的不同二元组的数量。

A,b<=10^11

#define P pair<pair<int,int>,int>

int prime[N],vis[N],mus[N],cnt;

ll inv=500000004,inv1=166666668;

void init\_mus()

{

cnt=0;memset(vis,0,sizeof(vis));

mus[1]=1;

for(int i=2;i<N;i++){

if(!vis[i]){prime[cnt++]=i;mus[i]=-1;}

for(int j=0;j<cnt&&i\*prime[j]<N;j++){

vis[i\*prime[j]]=1;

if(i%prime[j]==0){mus[i\*prime[j]]=0;break;}

else mus[i\*prime[j]]=-mus[i];

}

}

}

ll query(ll n)

{

ll ans=0;

for(ll k=1;k\*k<=n;k++){if(!mus[k])continue;

ll tmp=0;

for(ll i=1;i\*i\*i<=n/k/k;i++){

for(ll j=i+1;j\*j\*i<=n/k/k;j++){

tmp+=(n/k/k/i/j-j)\*6LL+3LL;

}

tmp+=(n/k/k/i/i-i)\*3LL;

tmp++;

}

ans+=1LL\*tmp\*mus[k];

}

return ans;

}

最大公约数之和

给出一个数N，输出小于等于N的所有数，两两之间的最大公约数之和

N<=10^10

#define P pair<pair<int,int>,int>

int cnt,phi[N],vis[N],p1[N];

ll pre[N],inv;

int modexp(ll a,ll b,ll p)

{

ll ans=1;a%=b;

while(b){if(b&1)ans=ans\*a%p;b>>=1;a=a\*a%p;}

return ans;

}

void init\_euler()

{

memset(p1,0,sizeof(p));

memset(phi,0,sizeof(phi));

memset(vis,0,sizeof(vis));

for(int i=2;i<N;i++){

if(!vis[i]){p1[++p1[0]]=i;phi[i]=i-1;}

for(int j=1;j<=p1[0];j++){

if(i\*p1[j]>=N)break;

vis[i\*p1[j]]=true;

if(!(i%p1[j])){phi[i\*p1[j]]=phi[i]\*p1[j];break;}

phi[i\*p1[j]]=phi[i]\*(p1[j]-1);

}

}phi[1]=1;

pre[0]=0;

for(int i=1;i<N;i++)pre[i]=(pre[i-1]+phi[i])%p;

}

map<ll,ll>m;

inline ll cal(ll x)

{

if(x<N)return pre[x];

if(m.count(x))return m[x];

ll sum=0;

for(ll i=1;i\*i<=x;i++){

sum+=pre[i]\*((x/i-x/(i+1LL))%p)%p;sum%=p;

if(i!=1LL&&x/i!=i){

ll tmp=0;

if(x/i<N)tmp=pre[x/i]\*((x/(x/i)-x/(x/i+1LL))%p)%p;

else tmp=(m.count(x)==0?cal(x/i):m[x/i])\*((x/(x/i)-x/(x/i+1LL))%p)%p;

sum+=tmp;

}

sum%=p;

}

return m[x]=(((x%p)\*((x+1LL)%p)%p)\*inv%p-sum+p)%p;

}

ll query(ll n)

{

ll ans=0;

for(ll i=1;i\*i<=n;i++){

ll r=n/i,l=n/(i+1);

ans=(ans+((cal(r)-cal(l)+p)%p)\*i%p\*i%p)%p;

ll tmp=(cal(i)-cal(i-1))\*((n/i)%p)%p\*((n/i)%p)%p;

if(n/i!=i)ans=(ans+tmp)%p;

//ans+=euler[i]\*(n/i)\*(n/i);

}

//printf("-- %lld\n",ans );

return ans;

}

出一个数N，输出小于等于N的所有数，两两之间的最小公倍数之和

N<=10^10

#define P pair<pair<int,int>,int>

int prime[N],vis[N],phi[N];

ll inv=500000004,inv1=166666668;

ll pre[N];

void init\_euler()

{

memset(prime,0,sizeof(p));

memset(phi,0,sizeof(phi));

memset(vis,0,sizeof(vis));

for(int i=2;i<N;i++){

if(!vis[i]){prime[++prime[0]]=i;phi[i]=i-1;}

for(int j=1;j<=prime[0];j++){

if(i\*prime[j]>=N)break;

vis[i\*prime[j]]=true;

if(!(i%prime[j])){phi[i\*prime[j]]=phi[i]\*prime[j];break;}

phi[i\*prime[j]]=phi[i]\*(prime[j]-1);

}

}phi[1]=1;

pre[0]=0;

for(ll i=1;i<N;i++) pre[i]=(pre[i-1]+1LL\*i\*i%p\*phi[i]%p)%p;

}

map<ll,ll>m;

ll ask(ll x)

{

x%=p;

return x%p\*(x+1LL)%p\*(2LL\*x+1LL)%p\*inv1%p;

}

ll cal(ll n)

{

if(n<N)return pre[n];

if(m.count(n))return m[n];

ll ans=n%p\*((n+1LL)%p)%p\*inv%p;ans=ans\*ans%p;

ll ans1=0;

for(ll i=1;i\*i<=n;i++){

ll l=n/(i+1),r=n/i;

ll tmp=(ask(r)-ask(l)+p)%p\*cal(i)%p;

ans1=(ans1+tmp)%p;

if(n/i!=i&&i>1){

tmp=i\*i%p\*cal(n/i)%p;

ans1=(ans1+tmp)%p;

}

}

m[n]=(ans-ans1+p)%p;

return (ans-ans1+p)%p;

}

ll query(ll n)

{

ll ans=0;

for(ll i=1;i\*i<=n;i++){

ll l=n/(i+1LL),r=n/i;

ll tmp=i\*(i+1)%p\*inv%p;

// printf("-- %lld\n",tmp );

tmp=tmp\*(cal(r)-cal(l)+p)%p;

// printf("-- %lld %lld %lld %lld\n",r,l,cal(r),cal(l) );

ans=(ans+tmp)%p;

if(n/i!=i){

tmp=1LL\*phi[i]%p\*i%p\*i%p;

ll t=(n/i)%p;

tmp=tmp\*t%p\*(t+1LL)%p\*inv%p;

ans=(ans+tmp)%p;

}

}

return ans;

}