

# 数论\_1

## 逆元递推

### 算法简介

线性时间递推  $O(\sim p-1)$  在模  $p$  意义下的乘法逆元。

### 算法思想

首先  $1^{-1} \equiv 1 \pmod p$

设  $p = k \ast i + r \left( 1 \leq r < i < p \right)$

所以有  $k \ast i + r \equiv 0 \pmod p$

两边同乘以  $r^{-1}$  有  $i^{-1} \equiv -k \ast r^{-1} \pmod p$

即  $i^{-1} \equiv -\lfloor \frac{p}{i} \rfloor \ast i^{-1} \pmod p$

### 算法模板

[洛谷p3811](#)

```
const int MAXP=3e6+5;
int inv[MAXP];
void get_inv(int p){
    inv[1]=1;
    _for(i,2,p)
        inv[i]=1LL*(p-p/i)*inv[p%i]%p;
}
```

## 数论分块

### 算法简介

数论分块用于解决形如  $\sum_{i=1}^n \lfloor \frac{n}{i} \rfloor$  的问题，时间复杂度  $O(\sqrt{n})$

### 算法思想

由于部分  $\lfloor \frac{n}{i} \rfloor$  的值相同，所以考虑分块计算。

设对  $\forall i \in \lfloor l, r \rfloor$  有  $\lfloor \frac{n}{r} \rfloor$  的值相同。

显然  $l$  等于上一个块的  $r$  加  $1$ ，故只需要考虑  $r$  的计算。

事实上有 
$$\lfloor \frac{n}{\lfloor \frac{n}{\lfloor \frac{n}{r} \rfloor} \rfloor} \rfloor \leq \frac{n}{\lfloor \frac{n}{\lfloor \frac{n}{r} \rfloor} \rfloor + 1}$$

移项，得 
$$\frac{n}{\lfloor \frac{n}{\lfloor \frac{n}{r} \rfloor} \rfloor + 1} \leq \lfloor \frac{n}{\lfloor \frac{n}{r} \rfloor} \rfloor$$

所以有 
$$r = \lfloor \frac{n}{\lfloor \frac{n}{\lfloor \frac{n}{r} \rfloor} \rfloor} \rfloor$$

关于算法的时间复杂度分析，有时间复杂度等于  $\lfloor \frac{n}{r} \rfloor$  的可能取值个数。

当  $\lfloor \frac{n}{r} \rfloor \leq \sqrt{n}$  时，显然这样的取值不超过  $\sqrt{n}$  个。

当  $\lfloor \frac{n}{r} \rfloor > \sqrt{n}$  时，有  $i \leq \sqrt{n}$  所以这样的取值也不超过  $\sqrt{n}$  个。

综上所述  $\lfloor \frac{n}{r} \rfloor$  的可能取值个数只有  $O(\sqrt{n})$  个，时间复杂度证毕。

## 算法模板

[洛谷p2261](#)

### 题意

给定  $n$  和  $k$  请计算  $\sum_{i=1}^n k \bmod i$

### 题解

$$\sum_{i=1}^n k \bmod i = \sum_{i=1}^n k - i \ast \lfloor \frac{k}{i} \rfloor = n \ast k - \sum_{i=1}^n i \ast \lfloor \frac{k}{i} \rfloor$$

```
#include <cstdio>
#include <iostream>
#include <vector>
#include <algorithm>
#include <cstring>
#include <cctype>
#include <queue>
#include <cmath>
#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 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=3e6+5;
LL pre_s(int k){
return 1LL*k*(k+1)/2;
}
LL cal(int n,int k){
LL ans=0;
int lef=1,rig=0;
while(lef<=n){
rig=min(n,k/(k/lef));
ans+=(pre_s(rig)-pre_s(lef-1))*(k/lef);
lef=rig+1;
}
return ans;
}
int main()
{
LL n=read_int(),k=read_int();
enter(n*k-cal(min(n,k),k));
return 0;
}

```

From:  
<https://wiki.cvbbacm.com/> - CVBB ACM Team

Permanent link:  
[https://wiki.cvbbacm.com/doku.php?id=2020-2021:teams:legal\\_string:jxm2001:%E6%95%B0%E8%AE%BA\\_1&rev=1593689438](https://wiki.cvbbacm.com/doku.php?id=2020-2021:teams:legal_string:jxm2001:%E6%95%B0%E8%AE%BA_1&rev=1593689438)

Last update: 2020/07/02 19:30