

- 数论，是专门研究整数的纯数学的分支，而整数的基本元素是素数（也称质数），所以数论的本质是对素数性质的研究。
- 数论被高斯誉为“数学中的皇冠”。按研究方法来看，数论大致可分为初等数论和高等数论。而初等数论是用初等方法研究的数论，它的研究方法本质上说，就是利用整除性质，主要包括整除理论、同余理论、连分数理论。
- 信息学竞赛中的数论主要涉及素数、约数、同余和数论函数等相关知识。



求下数组中第2大的数。

3	1	2	5	4	7	6
---	---	---	---	---	---	---

temp=nums[low]=3,以3为枢轴点,进行快排。

第一次, low=0,high=6,temp=3,经过快排后的结果:

6	7	4	5	3	2	1
---	---	---	---	---	---	---

此时partition排序后获得的index=4, $4 > 2$, 那么第2大的数位于枢轴点的左边。

第二次, 只针对上图中左半边进行排序。

temp=nums[low]=6,以6为枢轴点进行第2次快排。

6	7	4	5	3	2	1
---	---	---	---	---	---	---

第2次, low=0, high=4-1=3, temp=6; |

排序后的结果:

7	6	4	5	3	2	1
---	---	---	---	---	---	---

此时快排后的index=1, $1 = 2 - 1$, 所以枢轴点对应的就是第k大的数。

结束!

<https://blog.csdn.net/orangefly0214>



1 基本概念

整数集合： $Z = \{\dots, -2, -1, 0, 1, 2, \dots\}$

自然数集合： $N = \{0, 1, 2, \dots\}$

整除： 若 $a = bk$ ，其中 a, b, k 都是整数，则 b 整除 a ，记做 $b \mid a$ 。

约数： 若 $b \mid a$ 且 $b > 0$ ，则称 b 是 a 的约数（因数）， a 是 b 的倍数。

- 1 整除任何数，任何数都整除 0。
- 若 $a \mid b, a \mid c$ ，则 $a \mid (b+c), a \mid (b-c)$ 。
- 若 $a \mid b$ ，则对任意整数 c ， $a \mid (bc)$ 。

传递性： 若 $a \mid b, b \mid c$ ，则 $a \mid c$ 。



1 素数与合数

素数： $a > 1$ 且除了1和本身外不能被其它数整除的数。

合数： $a > 1$ 且不是素数的数称为合数。

其他整数（0, 1负整数）既不是素数也不是合数。

素数有无穷多个，但分布比较稀疏，不大于 n 的素数约有 $n/\ln(n)$ 个。



2 素数判定 (试除法)

- 若 n 是一个合数，则 n 至少有 1 个素因子。因此其中最小的素因子一定不大于 $\sqrt{n}$ 。
- 如果 n 是合数，则一定可以为分解 $a \times b$ 的形式，其中 $a \leq b, a \neq 1, b \neq n$ ，如 $18 = 2 \times 9, 18 = 3 \times 6$ 。因 $a \times a \leq a \times b = n$ ，则可得： $a \leq \sqrt{n}$ 。
- 可得判断依据：如果 $2 \sim \sqrt{n}$ 中有 n 的约数，则 n 是合数，否则 n 是素数。



2.1 素数判定 (试除法)

- 若 n 是一个合数，则 n 至少有 1 个素因子。因此其中最小的素因子一定不大于 $\sqrt{n}$ 。
- 如果 n 是合数，则一定可以为分解 $a \times b$ 的形式，其中 $a \leq b, a \neq 1, b \neq n$ ，如 $18 = 2 \times 9, 18 = 3 \times 6$ 。因 $a \times a \leq a \times b = n$ ，则可得： $a \leq \sqrt{n}$ 。
- 可得判断依据：如果 $2 \sim \sqrt{n}$ 中有 n 的约数，则 n 是合数，否则 n 是素数。

```
1    bool isPrime(int n){
2        if (n==1) return false;
3        else{
4            for(int i=2;i*i<=n;i++)
5                if (n % i == 0) return false;
6            return true;
7        }
8    }
```



2.2 朴素筛选法

对于序列中的每个合数，一定可以写成 $p * x$ 的形式， p 是序列左边较小的数， x 是倍数，枚举倍数 x ，把 $p * x$ 标记为合数，这就是筛选法。

2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20

2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20



2.2 朴素筛选法

2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20

2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20

2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20

2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20

2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20



2.2 朴素筛选法

在筛选过程中，第一轮执行 $n/2$ 次，第二轮执行 $n/3$ 次，第三轮执行 $n/4$ 次....

时间复杂度为 $O(\frac{n}{2} + \frac{n}{3} + \frac{n}{4} + \dots) = O(n * (\frac{1}{2} + \frac{1}{3} + \dots))$
 $= O(n * \log n)$

每个i都需要遍历:

```
2 // 朴素的筛选法
3 void get_primes2(){
4     for(int i=2; i<=n; i++){
5         if(!st[i]) primes[cnt++] = i; // 把素数存起来
6         for(int j=i; j<=n; j+=i){
7             // 不管是合数还是质数，都用来筛掉后面它的倍数
8             st[j] = true;
9         }
10    }
11 }
```



2.3 埃氏(Eratosthenes)筛法

在朴素筛选法中，对每个数都进行筛选，显然存在大量重复的筛选，如筛选过2后，2的倍数就不需要再筛选。这就是埃氏筛选法。

	2	3	4	5	6	7	8	9	10	Prime numbers
11	12	13	14	15	16	17	18	19	20	
21	22	23	24	25	26	27	28	29	30	
31	32	33	34	35	36	37	38	39	40	
41	42	43	44	45	46	47	48	49	50	
51	52	53	54	55	56	57	58	59	60	
61	62	63	64	65	66	67	68	69	70	
71	72	73	74	75	76	77	78	79	80	
81	82	83	84	85	86	87	88	89	90	
91	92	93	94	95	96	97	98	99	100	
101	102	103	104	105	106	107	108	109	110	
111	112	113	114	115	116	117	118	119	120	



2.3 埃氏(Eratosthenes)筛法

埃氏筛选法只筛选素数的倍数：

$$\begin{aligned}\text{时间复杂度为 } O\left(\frac{n}{2} + \frac{n}{3} + \frac{n}{5} + \frac{n}{7} + \frac{n}{11} + \dots\right) \\ = O(n * \log \log n) \approx O(n)\end{aligned}$$



2.3 埃氏(Eratosthenes)筛法

遍历每一个素数即可:

```
5  int isprime[100005];
6  void prime(int n){
7      int i,j;
8      for(int i=2;i<=n;i++){
9          isprime[i]=1; // 初始都是素数
10         for(int i=2;i<=n;i++){
11             if(isprime[i]){
12                 for(j=i*i;j<=n;j+=i){
13                     isprime[j]=0;
14                 }
15             }
16         }
17     }
```



[1444] 埃氏筛选求素数

```
8  const int SIZE = 1e7+5;
9
10 int prime[SIZE]; // 第i个素数
11 bool is_prime[SIZE]; // true表示i是素数
12
13 int slove(int n)
14 {
15     int p = 0;
16     for(int i = 0; i <= n; i++)
17         is_prime[i] = true; // 初始化都是素数
18     is_prime[0] = is_prime[1] = false; // 0,1不是素数
19     for(int i = 2; i <= n; i++)
20     {
21         if(is_prime[i]) //
22         {
23             prime[p++] = i; // 计算素数的个数, 也记录下了素数
24             for(int j = 2 * i; j <= n; j += i) // 除掉了i的倍数的数字
25                 is_prime[j] = false;
26         }
27     }
28     return p;
29 }
```

```
31 int main()
32 {
33     int n;
34     cin >> n;
35     int res = slove(n);
36     for(int i = 0; i < res; i++)
37         printf("%d\n", prime[i]);
38 }
```



2.4 线性筛选 (欧拉筛选)

埃氏筛选中，以 $n=50$ 为例，30这个数被筛了3次，分别是：

$$2 * 15 (p = 2)$$

$$3 * 10 (p = 3)$$

$$5 * 10 (p = 5)$$

如何去掉这些重复的筛选？每个合数只会被它的最小质因子筛掉，每个合数只会被筛一次。

这就是“快速线性筛选法”，也称为欧拉筛选。快速线性筛选法没有冗余，不会重复筛除一个数，几乎是线性的。

10^7 时比埃氏筛法快一倍

10^6 时与埃氏筛法差不多

欧拉筛法（线性筛）



- 枚举 $2 \sim n$ 中的每一个数 i :
 - 如果 i 是素数则保存到素数表中;
 - 利用 i 和素数表中的素数 $\text{prime}[j]$ 去筛除 $i * \text{prime}[j]$ ，为了确保 $i * \text{prime}[j]$ 只被素数 $\text{prime}[j]$ 筛除过这一次，要确保 $\text{prime}[j]$ 是 $i * \text{prime}[j]$ 中最小的素因子，即 i 中不能有比 $\text{prime}[j]$ 还要小的素因子。

核心： i 只会被最小质数因子筛掉

欧拉筛法（线性筛）



1. 如果 i 是素数的话，素数 i 乘以不大于 i 的素数，这样筛除的数是不会重复的（线性筛选的理论基础）。筛出的数都是 $n = p_1 * p_2$ 的形式，且 p_1 和 p_2 不相等；如：

$$i = 2: n = 2;$$

$$i = 3: 2 * 3, 3 * 3;$$

$$i = 5: 2 * 5, 3 * 5, 5 * 5$$

不会有值相等

2. 如果 i 是合数，此时 i 可以表示成递增素数相乘 $i = p_1 * p_2 * \dots * p_n$ ， $p_i \leq p_j (i \leq j)$ ， p_1 也是最小的系数。



线性筛 $n=50$ 的筛选过程图示:

	素数表	筛除的数		素数表	筛除的数
<u>2</u>	{2}	{4}	<u>13</u>	{2, 3, 5, 7, 11, 13}	{26, 39}
<u>3</u>	{2, 3}	{6, 9}	<u>14</u>	{2, 3, 5, 7, 11, 13}	{28}
<u>4</u>	{2, 3}	{8}	<u>15</u>	{2, 3, 5, 7, 11, 13}	{30, 45}
<u>5</u>	{2, 3, 5}	{10, 15, 25}	<u>16</u>	{2, 3, 5, 7, 11, 13}	{32}
<u>6</u>	{2, 3, 5}	{12}	<u>17</u>	{2, 3, 5, 7, 11, 13, 17}	{34}
<u>7</u>	{2, 3, 5, 7}	{14, 21, 35, 49}	<u>18</u>	{2, 3, 5, 7, 11, 13, 17}	{36}
<u>8</u>	{2, 3, 5, 7}	{16}	<u>19</u>	{2, 3, 5, 7, 11, 13, 17, 19}	{38}
<u>9</u>	{2, 3, 5, 7}	{18, 27}	<u>20</u>	{2, 3, 5, 7, 11, 13, 17, 19}	{20}
<u>10</u>	{2, 3, 5, 7}	{20}	<u>21</u>	{2, 3, 5, 7, 11, 13, 17, 19}	{42}
<u>11</u>	{2, 3, 5, 7, 11}	{22, 33}	<u>22</u>	{2, 3, 5, 7, 11, 13, 17, 19}	{44}
<u>12</u>	{2, 3, 5, 7, 11}	{24}	...	...	...



欧拉筛法（线性筛）

```
1 void get_primes(){
2     for(int i=2;i<=n;i++){
3         if(!st[i]) primes[cnt++]=i;
4         for(int j=0;primes[j]*i<=n;j++){
5             st[primes[j]*i]=true; //用最小质因子去筛合数
6             if(i%primes[j]==0)
7                 break;
8         }
9     }
10 }
```

1) 当 $i \% \text{primes}[j] \neq 0$ 时, 说明此时遍历到的 $\text{primes}[j]$ 不是 i 的质因子, 那么只可能是此时的 $\text{primes}[j] < i$ 的最小质因子, 所以 $\text{primes}[j] * i$ 的最小质因子就是 $\text{primes}[j]$;

2) 当有 $i \% \text{primes}[j] == 0$ 时, 说明 i 的最小质因子是 $\text{primes}[j]$, 因此 $\text{primes}[j] * i$ 的最小质因子也就应该是 $\text{primes}[j]$, 之后接着用 $\text{st}[\text{primes}[j+1] * i] = \text{true}$ 去筛合数时, 就不是用最小质因子去更新了, 因为 i 有最小质因子 $\text{primes}[j] < \text{primes}[j+1]$, 此时的 $\text{primes}[j+1]$ 不是 $\text{primes}[j+1] * i$ 的最小质因子, 此时就应该退出循环, 避免之后重复进行筛选。



[1634] 线性筛素数

```
4  #define m 5800000
5  #define m1 100000000+10
6  int n,len,a[m];
7  bool vis[m1];
8  int main()
9  {
10     scanf("%d",&n);
11     for(int i = 2;i <= n;i++)
12     {
13         if(!vis[i])
14             a[++len] = i;
15         for(int j = 1;(j <= len )&&(i * a[j] <= n);j++)
16         {
17             vis[a[j] * i] = true;
18             if(!(i % a[j]))
19                 break;
20         }
21     }
22     printf("%d",len);
23     return 0;
24 }
```

素数筛法

```
int m,p[10000],r[10000]; //p存质数 r存次幂
void div(int n)
{
    for(int i=2;i*i<=n;i++)
        if(n%i==0) //能分解
        {
            p[++m]=i;
            while(n%i==0)n/=i,r[m]++;
        }
    if(n>1)p[++m]=n,r[m]=1; //如果还有一个大于sqrt(n)的质数
    for(int i=1;i<=m;i++) //输出
        cout<<p[i]<<' '<<r[i]<<endl;
}
```



[3934]Prime Distance

给定两个整数 L, R ，求闭区间 $[L, R]$ 中相邻两个质数差值最小的数对与差值最大的数对。当存在多个时，输出靠前的素数对。

$$1 \leq L < R < 2^{31} (1 \leq L < R \leq 2147483647), R - L \leq 10^6。$$

- 性质1: 若一个数 n 是一个合数, 必然存在 2 个因子 $d, \frac{n}{d}$, 假设 $d \leq \frac{n}{d}$, 则 $d \leq \sqrt{n}$, 因此必然存在一个小于等于 $\sqrt{n}$ 的因子
- 性质2: 若 $x \in [L, R]$, 且 x 是合数, 则一定存在 $P \leq \sqrt{2^{31} - 1} (< 50000)$, 使得 P 能整除 x , 其中 $P < x$.

- 1、找出 $1 \sim \sqrt{2^{31} - 1}$ (< 50000) 中的所有质因子
- 2、对于 $1 \sim 50000$ 中每个质数 P ，将 $[L, R]$ 中所有 P 的倍数筛掉(至少 2 倍)
 - 找到大于等于 L 的最小的 P 的倍数 P_0 ，找下一个倍数时只需要 $+= P$ 即可

如何求出在区间 $[L, R]$ 中最小的 p 的倍数

$$1. \left\lceil \frac{L}{p} \right\rceil \times p \Rightarrow \text{ceil}((\text{double})L / p) * p$$

$$2. \left\lfloor \frac{L+p-1}{p} \right\rfloor \times p \Rightarrow (L + p - 1) / p * p$$

由于L和R的范围很大，所以，直接通过素数筛跑出答案会超时，但是，L和R的区间 $\leq 1e6$ ，因此我们可以考虑求出[L, R]中的所有素数。

如果直接求，时间复杂度为 $[R-L] * \sqrt{x}$ ，也会超时。还是需要考虑筛选法，但常规筛选法是从[2, N]全区域筛选，也会超时，此时我们要思考如何缩小筛选的范围？

我们可以先通过素数筛跑出 $2 \sim \sqrt{R}$ 的所有素数，对于每个素数a，把[L, R]中所有能被a整除的数($i*a$) (合数)标记，所有未被标记的数就是素数，把相邻的2个素数比较，找出差最小和最大的。

```
9  bool st[N];
10  int primes[N], cnt;
11  void get_primes(int n) {
12      memset(st, 0, sizeof st);
13      cnt = 0;
14      for (int i = 2; i <= n; ++ i) {
15          if (!st[i]) primes[cnt ++ ] = i;
16          for (int j = 0; primes[j] * i <= n; ++ j) {
17              st[primes[j] * i] = true;
18              if (i % primes[j] == 0) break;
19          }
20      }
21  }
```

```
1  #include <iostream>
2  #include <cstring>
3  #include <cstdio>
4  using namespace std;
5  typedef long long LL;
6  //sqrt(2^31 - 1)
7  const int N = 1e6 + 10;
```



```
23 int main() {
24     int l, r;
25     while (~scanf("%d%d", &l, &r)) {
26         get_primes(50000);
27
28         //把[l,r]区间内所有的合数用他们的最小质因子筛掉
29         memset(st, 0, sizeof st);
30         for (int i = 0; i < cnt; ++ i) {
31             LL p = primes[i];
32             for (LL j = max(2 * p, (l + p - 1) / p * p); j <= r; j += p)
33                 st[j - l] = true;
34         }
35
36         //剩下的所有的都是素数了
37         cnt = 0;
38         for (int i = 0; i <= r - l; ++ i)
39             if (!st[i] && i + l > 1)
40                 primes[cnt ++ ] = i + l;
```

Report Window



```
if (cnt < 2) printf("There are no adjacent primes.\n");
else {
    // 计算间隔
    int minp = 0, maxp = 0;
    for (int i = 0; i + 1 < cnt; ++ i) {
        int d = primes[i + 1] - primes[i];
        if (d < primes[minp + 1] - primes[minp]) minp = i;
        if (d > primes[maxp + 1] - primes[maxp]) maxp = i;
    }
    printf("%d,%d are closest, %d,%d are most distant.\n",
        primes[minp], primes[minp + 1],
        primes[maxp], primes[maxp + 1]);
}
```



2 因式分解

- 表述：若整数 $N \geq 2$ ，那么 N 一定可以惟一地表示为若干素数的乘积。形如

$$N = p_1^{r_1} p_2^{r_2} \dots p_k^{r_k} \quad (p_i \text{ 为素数}, r_i \geq 0)$$

$$18 = 2 * 3 * 3 = 2^1 * 3^2$$

$$45 = 3 * 3 * 5$$

.....

显而易见，对于一个合数 N ，一定存在两个数，使其相乘为 N ，这两个数就是 N 的因子。

若这两个数不为素数，则可以继续将这两个数分解，直至得到素数。

综上所述，一个合数 N ，是由多个素因子相乘得到。



2 因式分解

- 表述：若整数 $N \geq 2$ ，那么 N 一定可以惟一地表示为若干素数的乘积。形如

$$N = p_1^{r_1} p_2^{r_2} \dots p_k^{r_k} \quad (p_i \text{ 为素数}, r_i \geq 0)$$

$$18 = 2 * 3 * 3 = 2^1 * 3^2$$

$$45 = 3 * 3 * 5$$

.....

显而易见，对于一个合数 N ，一定存在两个数，使其相乘为 N ，这两个数就是 N 的因子。

若这两个数不为素数，则可以继续将这两个数分解，直至得到素数。

综上所述，一个合数 N ，是由多个素因子相乘得到。



1 [3261]质因数分解

已知正整数 n 是两个不同的质数的乘积，试求出较大的那个质数。

对于100%的数据， $6 \leq n \leq 2 \times 10^9$ 。

```
1     for(int i=2;i*i <= n;i++)
2         if(n%i == 0) {
3             cout << n/i << endl;
4             break;
5         }
```



```
1  #include<bits/stdc++.h>
2  using namespace std;
3  int main ()
4  {
5      int n;
6      bool flag=0;
7      cin>>n;
8      for(int i=2;i<=sqrt(n);++i)
9          if(n%i==0)
10         {
11             flag=1;
12             cout<<n/i<<endl;
13             break;
14         }
15     if(!flag)
16         cout<<1<<endl;
17     return 0;
18 }
```



2 [2658]质因数分解2

给定 n 个正整数 a_i ，将每个数分解质因数，并按照质因数从小到大的顺序输出每个质因数的底数和指数。

第一行包含整数 n 。接下来 n 行，每行包含一个正整数 a_i 。

对于每个正整数 a_i ，按照从小到大的顺序输出其分解质因数后，每个质因数的底数和指数，每个底数和指数占一行。

每个正整数的质因数全部输出完毕后，输出一个空行。

$$1 \leq n \leq 100 \quad 2 \leq a_i \leq 2 \times 10^9$$



x 的质因子最多只包含一个大于 $\sqrt{x}$ 的质数。如果有两个，这两个因子的乘积就会大于 x ，矛盾。

i 从 2 遍历到 $\sqrt{x}$ 。用 x / i ，如果余数为 0，则 i 是一个质因子。

s 表示质因子 i 的指数， $x \% i \neq 0$ ，则 $s++$ ， $x = x / i$ 。

最后检查是否有大于 $\sqrt{x}$ 的质因子，如果有，输出。

```

19 int main()
20 {
21     int n;
22     cin >> n;
23     while (n--)
24     {
25         int x;
26         cin >> x;
27         divide(x);
28     }
29     return 0;
30 }
31

```



```

5 void divide(int x)
6 { //i <= x / i:防止越界, 速度大于 i < sqrt(x)
7     for (int i = 2; i <= x / i; i++)
8         if (x % i == 0) //i为底数
9             {
10                 int s = 0; //s为指数
11                 while (x % i == 0) x /= i, s++;
12                 cout << i << ' ' << s << endl; //输出
13             }
14     //如果x还有剩余, 单独处理
15     if (x > 1) cout << x << ' ' << 1 << endl;
16     cout << endl;
17 }

```



3 [1329] 质数因子分解

在数学课上，老师给同学们讲解了如何对一个正整数 n 进行质数因子分解，如 $60=2*2*3*5$ ，其质数因子为2、2、3、5。课后，老师给同学们布置了一道习题：求一个正整数的质数因子分解形式中，各质数因子之和（如果质数因子重复的话，则必须多次计算；注意1不是质数）。例如60的质数因子之和为12。请你通过编程的方法帮同学们解答这道习题。

输入格式

每个测试数据 n 占一行，

$$2 \leq n \leq 2 \times 10^9$$



4 [4329]: Sherlock and his girlfriend

夏洛克有了一个新女朋友。情人节快到了，他想送给她一些珠宝。他买了 n 件珠宝。第 i 件的价格等于 $i+1$ ，即珠宝的价格为2, 3, 4, ... $n+1$ 。

沃森给夏洛克一个挑战，让他给这些珠宝上色，如果其中一件的价格是另一件价格的素因子，那么两件的颜色就不能一样。

此外，沃森要求他尽量减少使用不同颜色的数量。

输入一行包含单个整数 n ($1 \leq n \leq 1000000$) – 珠宝的数量。

输出的第一行应该包含一个整数 k ，即在给定约束条件下可用于为珠宝件上色的最小颜色数。

下一行由 n 个空格分隔的整数（介于1和 k 之间）组成，这些整数按照价格上涨的顺序指定每件物品的颜色。

如果有多种方法可以使用 k 颜色为工件着色，则可以输出其中任何一种。

题目说如果一件珠宝的价格是另一件珠宝价格的素因子，那么两件珠宝就不能有相同的颜色，两者颜色不同。那么一个无论多大的非素数都可以分解为多个因子，所以题目的本质只是让我们区别素数和非素数，所以最多的颜色种类也只有两种，至于素数是1还是非素数是1，都是可以过的。

- n 个点，标号 $2..n+1$ ，给这些点染色，要求若 a 是 b 的素因子，则 a 和 b 的颜色不同
求一种颜色数最少的方案， $n \leq 1000$ 。
- 分析：
- 注意到这是二分图，一边是素数，一边是合数。
- 把素数都染成 1，合数都染成 2 即可。



```
1 #include<cstring>
2 using namespace std;
3 bool isprime(int x) {
4     for (int i = 2; i*i <= x; i++) {
5         if (x % i == 0)
6             return false;
7         //判断这个数是否为素数
8     }
9     return true;
10 }
```

```
1
2 int main()
3 {
4     int n;
5     cin >> n;
```

```
16
17
18
19
20
21 //n=1和n=2对应2和2, 3他们都是素数所以只
22 //有一种颜色情况, 需要拿出来说明
23
24
25
26
27
28
29
30
31 //当n>2时无论n多大, 都只有素数非素数两种情况
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
590
591
592
593
594
595
596
597
598
599
600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
650
651
652
653
654
655
656
657
658
659
660
661
662
663
664
665
666
667
668
669
670
671
672
673
674
675
676
677
678
679
680
681
682
683
684
685
686
687
688
689
690
691
692
693
694
695
696
697
698
699
700
701
702
703
704
705
706
707
708
709
710
711
712
713
714
715
716
717
718
719
720
721
722
723
724
725
726
727
728
729
730
731
732
733
734
735
736
737
738
739
740
741
742
743
744
745
746
747
748
749
750
751
752
753
754
755
756
757
758
759
760
761
762
763
764
765
766
767
768
769
770
771
772
773
774
775
776
777
778
779
780
781
782
783
784
785
786
787
788
789
790
791
792
793
794
795
796
797
798
799
800
801
802
803
804
805
806
807
808
809
810
811
812
813
814
815
816
817
818
819
820
821
822
823
824
825
826
827
828
829
830
831
832
833
834
835
836
837
838
839
840
841
842
843
844
845
846
847
848
849
850
851
852
853
854
855
856
857
858
859
860
861
862
863
864
865
866
867
868
869
870
871
872
873
874
875
876
877
878
879
880
881
882
883
884
885
886
887
888
889
890
891
892
893
894
895
896
897
898
899
900
901
902
903
904
905
906
907
908
909
910
911
912
913
914
915
916
917
918
919
920
921
922
923
924
925
926
927
928
929
930
931
932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
970
971
972
973
974
975
976
977
978
979
980
981
982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999
1000
1001
1002
1003
1004
1005
1006
1007
1008
1009
1010
1011
1012
1013
1014
1015
1016
1017
1018
1019
1020
1021
1022
1023
1024
1025
1026
1027
1028
1029
1030
1031
1032
1033
1034
1035
1036
1037
1038
1039
1040
1041
1042
1043
1044
1045
1046
1047
1048
1049
1050
1051
1052
1053
1054
1055
1056
1057
1058
1059
1060
1061
1062
1063
1064
1065
1066
1067
1068
1069
1070
1071
1072
1073
1074
1075
1076
1077
1078
1079
1080
1081
1082
1083
1084
1085
1086
1087
1088
1089
1090
1091
1092
1093
1094
1095
1096
1097
1098
1099
1100
1101
1102
1103
1104
1105
1106
1107
1108
1109
1110
1111
1112
1113
1114
1115
1116
1117
1118
1119
1120
1121
1122
1123
1124
1125
1126
1127
1128
1129
1130
1131
1132
1133
1134
1135
1136
1137
1138
1139
1140
1141
1142
1143
1144
1145
1146
1147
1148
1149
1150
1151
1152
1153
1154
1155
1156
1157
1158
1159
1160
1161
1162
1163
1164
1165
1166
1167
1168
1169
1170
1171
1172
1173
1174
1175
1176
1177
1178
1179
1180
1181
1182
1183
1184
1185
1186
1187
1188
1189
1190
1191
1192
1193
1194
1195
1196
1197
1198
1199
1200
1201
1202
1203
1204
1205
1206
1207
1208
1209
1210
1211
1212
1213
1214
1215
1216
1217
1218
1219
1220
1221
1222
1223
1224
1225
1226
1227
1228
1229
1230
1231
1232
1233
1234
1235
1236
1237
1238
1239
1240
1241
1242
1243
1244
1245
1246
1247
1248
1249
1250
1251
1252
1253
1254
1255
1256
1257
1258
1259
1260
1261
1262
1263
1264
1265
1266
1267
1268
1269
1270
1271
1272
1273
1274
1275
1276
1277
1278
1279
1280
1281
1282
1283
1284
1285
1286
1287
1288
1289
1290
1291
1292
1293
1294
1295
1296
1297
1298
1299
1300
1301
1302
1303
1304
1305
1306
1307
1308
1309
1310
1311
1312
1313
1314
1315
1316
1317
1318
1319
1320
1321
1322
1323
1324
1325
1326
1327
1328
1329
1330
1331
1332
1333
1334
1335
1336
1337
1338
1339
1340
1341
1342
1343
1344
1345
1346
1347
1348
1349
1350
1351
1352
1353
1354
1355
1356
1357
1358
1359
1360
1361
1362
1363
1364
1365
1366
1367
1368
1369
1370
1371
1372
1373
1374
1375
1376
1377
1378
1379
1380
1381
1382
1383
1384
1385
1386
1387
1388
1389
1390
1391
1392
1393
1394
1395
1396
1397
1398
1399
1400
1401
1402
1403
1404
1405
1406
1407
1408
1409
1410
1411
1412
1413
1414
1415
1416
1417
1418
1419
1420
1421
1422
1423
1424
1425
1426
1427
1428
1429
1430
1431
1432
1433
1434
1435
1436
1437
1438
1439
1440
1441
1442
1443
1444
1445
1446
1447
1448
1449
1450
1451
1452
1453
1454
1455
1456
1457
1458
1459
1460
1461
1462
1463
1464
1465
1466
1467
1468
1469
1470
1471
1472
1473
1474
1475
1476
1477
1478
1479
1480
1481
1482
1483
1484
1485
1486
1487
1488
1489
1490
1491
1492
1493
1494
1495
1496
1497
1498
1499
1500
1501
1502
1503
1504
1505
1506
1507
1508
1509
1510
1511
1512
1513
1514
1515
1516
1517
1518
1519
1520
1521
1522
1523
1524
1525
1526
1527
1528
1529
1530
1531
1532
1533
1534
1535
1536
1537
1538
1539
1540
1541
1542
1543
1544
1545
1546
1547
1548
1549
1550
1551
1552
1553
1554
1555
1556
1557
1558
1559
1560
1561
1562
1563
1564
1565
1566
1567
1568
1569
1570
1571
1572
1573
1574
1575
1576
1577
1578
1579
1580
1581
1582
1583
1584
1585
1586
1587
1588
1589
1590
1591
1592
1593
1594
1595
1596
1597
1598
1599
1600
1601
1602
1603
1604
1605
1606
1607
1608
1609
1610
1611
1612
1613
1614
1615
1616
1617
1618
1619
1620
1621
1622
1623
1624
1625
1626
1627
1628
1629
1630
1631
1632
1633
1634
1635
1636
1637
1638
1639
1640
1641
1642
1643
1644
1645
1646
1647
1648
1649
1650
1651
1652
1653
1654
1655
1656
1657
1658
1659
1660
1661
1662
1663
1664
1665
1666
1667
1668
1669
1670
1671
1672
1673
1674
1675
1676
1677
1678
1679
1680
1681
1682
1683
1684
1685
1686
1687
1688
1689
1690
1691
1692
1693
1694
1695
1696
1697
1698
1699
1700
1701
1702
1703
1704
1705
1706
1707
1708
1709
1710
1711
1712
1713
1714
1715
1716
1717
1718
1719
1720
1721
1722
1723
1724
1725
1726
1727
1728
1729
1730
1731
1732
1733
1734
1735
1736
1737
1738
1739
1740
1741
1742
1743
1744
1745
1746
1747
1748
1749
1750
1751
1752
1753
1754
1755
1756
1757
1758
1759
1760
1761
1762
1763
1764
1765
1766
1767
1768
1769
1770
1771
1772
1773
1774
1775
1776
1777
1778
1779
1780
1781
1782
1783
1784
1785
1786
1787
1788
1789
1790
1791
1792
1793
1794
1795
1796
1797
1798
1799
1800
1801
1802
1803
1804
1805
1806
1807
1808
1809
1810
1811
1812
1813
1814
1815
1816
1817
1818
1819
1820
1821
1822
1823
1824
1825
1826
1827
1828
1829
1830
1831
1832
1833
1834
1835
1836
1837
1838
1839
1840
1841
1842
1843
1844
1845
1846
1847
1848
1849
1850
1851
1852
1853
1854
1855
1856
1857
1858
1859
1860
1861
1862
1863
1864
1865
1866
1867
1868
1869
1870
1871
1872
1873
1874
1875
1876
1877
1878
1879
1880
1881
1882
1883
1884
1885
1886
1887
1888
1889
1890
1891
1892
1893
1894
1895
1896
1897
1898
1899
1900
1901
1902
1903
1904
1905
1906
1907
1908
1909
1910
1911
1912
1913
1914
1915
1916
1917
1918
1919
1920
1921
1922
1923
1924
1925
1926
1927
1928
1929
1930
1931
1932
1933
1934
1935
1936
1937
1938
1939
1940
1941
1942
1943
1944
1945
1946
1947
1948
1949
1950
1951
1952
1953
1954
1955
1956
1957
1958
1959
1960
1961
1962
1963
1964
1965
1966
1967
1968
1969
1970
1971
1972
1973
1974
1975
1976
1977
1978
1979
1980
1981
1982
1983
1984
1985
1986
1987
1988
1989
1990
1991
1992
1993
1994
1995
1996
1997
1998
1999
2000
2001
2002
2003
2004
2005
2006
2007
2008
2009
2010
2011
2012
2013
2014
2015
2016
2017
2018
2019
2020
2021
2022
2023
2024
2025
2026
2027
2028
2029
2030
2031
2032
2033
2034
2035
2036
2037
2038
2039
2040
2041
2042
2043
2044
2045
2046
2047
2048
2049
2050
2051
2052
2053
2054
2055
2056
2057
2058
2059
2060
2061
2062
2063
2064
2065
2066
2067
2068
2069
2070
2071
2072
2073
2074
2075
2076
2077
2078
2079
2080
2081
2082
2083
2084
2085
2086
2087
2088
2089
2090
2091
2092
2093
2094
2095
2096
2097
2098
2099
2100
2101
2102
2103
2104
2105
2106
2107
2108
2109
2110
2111
2112
2113
2114
2115
2116
2117
2118
2119
2120
2121
2122
2123
2124
2125
2126
2127
2128
2129
2130
2131
2132
2133
2134
2135
2136
2137
2138
2139
2140
2141
2142
2143
2144
2145
2146
2147
2148
2149
2150
2151
2152
2153
2154
2155
2156
2157
2158
2159
2160
2161
2162
2163
2164
2165
2166
2167
2168
2169
2170
2171
2172
2173
2174
2175
2176
2177
2178
2179
2180
2181
2182
2183
2184
2185
2186
2187
2188
2189
2190
2191
2192
2193
2194
2195
2196
2197
2198
2199
2200
2201
2202
2203
2204
2205
2206
2207
2208
2209
2210
2211
2212
2213
2214
2215
2216
2217
2218
2219
2220
2221
2222
2223
2224
2225
2226
2227
2228
2229
2230
2231
2232
2233
2234
2235
2236
2237
2238
2239
2240
2241
2242
2243
2244
2245
2246
2247
2248
2249
2250
2251
2252
2253
2254
2255
2256
2257
2258
2259
2260
2261
2262
2263
2264
2265
2266
2267
2268
2269
2270
2271
2272
2273
2274
2275
2276
2277
2278
2279
2280
2281
2282
2283
2284
2285
2286
2287
2288
2289
2290
2291
2292
2293
2294
2295
2296
2297
2298
2299
2300
2301
2302
2303
2304
2305
2306
2307
2308
2309
2310
2311
2312
2313
2314
2315
2316
2317
2318
2319
2320
2321
2322
2323
2324
2325
2326
2327
2328
2329
2330
2331
2332
2333
2334
2335
2336
2337
2338
2339
2340
2341
2342
2343
2344
2345
2346
2347
2348
2349
2350
2351
2352
2353
2354
2355
2356
2357
2358
2359
2360
2361
2362
2363
2364
2365
2366
2367
2368
2369
2370
2371
2372
2373
2374
2375
2376
2377
2378
2379
2380
2381
2382
2383
2384
2385
2386
2387
2388
2389
2390
2391
2392
2393
2394
2395
2396
2397
2398
2399
2400
2401
2402
2403
2404
2405
2406
2407
2408
2409
2410
2411
2412
2413
2414
2415
2416
2417
2418
2419
2420
2421
2422
2423
2424
2425
2426
2427
2428
2429
2430
2431
2432
2433
2434
2435
2436
2437
2438
2439
2440
2441
2442
2443
2444
2445
2446
2447
2448
2449
2450
2451
2452
2453
2454
2455
2456
2457
2458
2459
2460
2461
2462
2463
2464
2465
2466
2467
2468
2469
2470
2471
2472
2473
2474
2475
2476
2477
2478
2479
2480
2481
2482
2483
2484
2485
2486
2487
2488
2489
2490
2491
2492
2493
2494
2495
2496
2497
2498
2499
2500
2501
2502
2503
2504
2505
2506
2507
2508
2509
2510
2511
2512
2513
2514
2515
2516
2517
2518
2519
2520
2521
2522
2523
2524
2525
2526
2527
2528
2529
2530
2531
2532
2533
2534
2535
2536
2537
2538
2539
2540
2541
2542
2543
2544
2545
2546
2547
2548
2549
2550
2551
2552
2553
2554
2555
2556
2557
2558
2559
2560
2561
2562
2563
2564
2565
2566
2567
2568
2569
2570
2571
2572
2573
2574
2575
2576
2577
2578
2579
2580
2581
2582
2583
2584
2585
2586
2587
2588
2589
2590
2591
2592
2593
2594
2595
2596
2597
2598
2599
2600
2601
2602
2603
2604
2605
2606
2607
2608
2609
2610
2611
2612
2613
2614
2615
2616
2617
2618
2619
2620
2621
2622
2623
2624
2625
2626
2627
2628
2629
2630
2631
2632
2633
2634
2635
2636
2637
2638
2639
2640
2641
2642
2643
2644
2645
2646
2647
2648

```



3 筛选法

常规**试除法**，通常写一个函数对要判断的整数 x ，时间复杂度为 $O(\sqrt{x})$ 。如果要判断 n 的数是否是素数，时间复杂度为 $O(n * \sqrt{x})$ 。

如果能够实现准备好**素数表**就可以帮助我们更有效地求解素数的相关问题。筛选法可以快速列举给定范围内的所有素数。其时间复杂度不大于 $O(n * \log n)$ 。





[7549]最强素数

素数41能写成连续6个素数之和： $41=2+3+5+7+11+13$ 。

现在要求n以内的素数中，能表示为最多连续素数之和的那个数，如果有多个答案，请输出最大的那个素数。

输入仅一行，一个整数n。100%的数据， $1 \leq n \leq 1000000$

输出就一个整数，为所求的能表示为最多连续素数之和的那个素数。

输入样例 100

输出样例 41

题目要求找 $[1, n]$ 范围内能表示为**最多连续素数和的素数**并输出（如果有多个输出最大的）。

首先筛选出一定范围内所需要的素数并存储，然后在找出不大于 n 的最大素数下标 x ，暴力枚举所有 $\text{prime}[0] \sim \text{prime}[x]$ 里所有组合，就可以得到答案。



求出所有素数

```
1  #include<bits/stdc++.h>
2  using namespace std;
3  const int MAXN=1000005;
4  int p[MAXN]; // =0 是素数  =1不是
5  int prime[MAXN]; // 存所有的素数
6  int cnt; // 表示素数的个数
7  int q[MAXN]; // q[i]表示长度为i是的最大值
8
9  void allPrime(){
10     for(int i=2; i<=sqrt(MAXN); i++){
11         if(p[i]) continue;
12         for(int j=i*i; j<=MAXN; j=j+i)
13             p[j]=1;
14     }
15     for(int i=2; i<MAXN; i++){
16         if(p[i]==0)
17             prime[cnt++]=i;
18     }
19 }
```



统计最长和

```
20 int main(){
21     int n,x;
22     scanf("%d",&n);
23     allPrime();
24     for(x=0;x<cnt;x++)//找出不大于n的最大素数
25         if(prime[x]>n)break;
26     int sum,maxlen=0;
27     for(int i=0;i<x;i++){
28         //枚举所有的组合
29         sum=0; //当前组合的和
30         for(int j=i;j<x;j++){//把i~j之间的所有素数求和
31             sum=sum+prime[j];
32             if(sum>n)break;
33             if(p[sum]==0){//sum是素数
34                 q[j-i+1]=max(sum,q[j-i+1]); //最初长度为j-i+1是的最大值
35                 maxlen=max(maxlen,j-i+1);
36             }
37         }
38     }
39     printf("%d",q[maxlen]);
```



筛法优化素因数分解-1

- 利用埃氏筛法可以快速实现素因数分解，只要在判定素数时记录下每个数值的最小素因数即可。算法实现如下：

```
1  #include<bits/stdc++.h>
2  using namespace std;
3  #define MAXN 1000000
4  bool prime[MAXN];
5  int Minfac[MAXN];
6  int res[MAXN];
7  void init(int n){//初始化
8      prime[0] = prime[1] = false;//0和1都不是素数
9      Minfac[0] = Minfac[1] = -1;//两数都没有素因子
10     for( int i=2;i<=n;++i){
11         prime[i] = true;
12         Minfac[i] = i;
13     }
14 }
```



筛法优化素因数分解-2

```
16 void eratos(int n){
17     for( int i=2;i*i<=n;++i){
18         if(prime[i]==true){//如果是质数
19             for( int j=i*i;j<=n;j+=i){
20                 prime[j] = false;
21                 //将其倍数设为false
22                 if(Minfac[j]==j){
23                     //如果之前没有找打这个数的最小素因子
24                     Minfac[j] = i;
25                     //i就是它的最小素因子
26                 }
27             }
28         }
29     }
30 }
```



筛法优化素因数分解-3

```
32 void factor(int x){
33     int i=1;
34     while(x>1){
35         res[i++]=Minfac[x]; //将最小的素因子加入数组
36         x /= Minfac[x]; //除掉它
37     }
38     i--;
39     cout<<"x=";
40     for(int j=1;j<=i;j++){
41         if(j==i) cout<<res[j];
42         else cout<<res[j]<<"*";
43     }
44 }
45 int main(){
46     int n;
47     cin>>n;
48     init(n);eratos(n);factor(n);
49     return 0;
50 }
```