

数论

主讲人：数一

本节课的内容

- 整除与同余
- 约数与倍数
- 素数与合数
- 素数筛
- 最大公约数
- 最小公倍数
- 互质
- 分解质因数
- 费马小定理
- 欧拉函数与欧拉定理

从带余除法到同余

$$a \div b = q \cdots \cdots c \Leftrightarrow a = bq + c$$



$$a \equiv c \pmod{b}$$

读作a与c在模b下同余

同余的性质

反身性: $a \equiv a \pmod{n}$

对称性: $a \equiv b \pmod{n} \Leftrightarrow b \equiv a \pmod{n}$

传递性: $\left. \begin{array}{l} a \equiv b \pmod{n} \\ b \equiv c \pmod{n} \end{array} \right\} \Leftrightarrow a \equiv c \pmod{n}$

同余的性质

线性合成: $\left. \begin{array}{l} a \equiv b \pmod{n} \\ c \equiv d \pmod{n} \end{array} \right\} \Rightarrow \begin{array}{l} a \pm c \equiv b \pm d \pmod{n} \\ ac \equiv bd \pmod{n} \end{array}$

消去公因子: $\left. \begin{array}{l} ac \equiv bc \pmod{n} \\ \gcd(c, n) = 1 \end{array} \right\} \Rightarrow a \equiv b \pmod{n}$

整除

$a \div n = q \Leftrightarrow a \equiv 0 \pmod{n} \Leftrightarrow n|a$, 读作 n 整除 a

n 称为 a 的约数

a 称为 n 的倍数

素数与合数

- 只含有1和本身两个约数的正整数，称为素数(约数个数=2)
- 除了1和本身还有其它约数的正整数，称为合数(约数个数>2)
- 1(约数个数<2)
- 正整数=1+素数+合数

素数

- 判定一个数 n 是否是素数
枚举因子: $O(\sqrt{n})$

素数筛预处理: $O(n) + O(1)$



本节课重点

枚举因子法

```
bool judge(int n) //判断一个数n是否是素数,  $O(\sqrt{n})$ 
{
    if(n==1) //把特殊的1特判掉
        return false;
    for(int i=2; i*i<=n; i++) //枚举因子
    {
        if(n%i==0) //n能被i整除 $\Leftrightarrow$  i是n的约数, n是合数
        {
            return false;
        }
    }
    return true; //在2~n-1中没有n的约数, n是素数
}
```

复杂度分析

```
bool judge(int n) //判断一个数n是否是素数
{
    if(n==1) //把特殊的1特判掉
        return false;
    for(int i=2; i*i<=n; i++) //枚举因子
    {
        if(n%i==0) //n能被i整除==>i是n的约数, n是合数
        {
            return false;
        }
    }
    return true; //在2~n-1中没有n的约数, n是素数
}
```

这个循环执行了 \sqrt{n} 次

别忘了先把
特殊的1判
断掉

容易写错的地方

这里必须小
于等于，不
能只有小于

这里必须是两个等
号表示等于，而不
是只有一个等号的
赋值

最后别忘了
要返回是素
数

```
bool isPrime(int n) //判断一个数是否是素数
{
    if (n==1) //把特殊的1特判掉
        return false;
    for (int i=2; i*i<=n; i++) //枚举因子
    {
        if (n%i==0) //n能被i整除，说明n是合数
        {
            return false;
        }
    }
    return true; //在2~n-1中没找到因子，说明n是素数
}
```

用这个方法我们能干什么？

1秒内单次判断 $1 \sim 10^{16}$ 以内的
任意一个数是否是素数

素数筛

- 可以提前处理出来 $1 \sim n$ 的全体素数，之后每次都能“瞬间”知道 $1 \sim n$ 中的某个数是否是素数。

素数筛

1. 把 $1 \sim n$ 列出来
2. 去掉不是特殊的1
3. 从小到大，枚举每一个没有删掉的数字 i
4. 把 i 的2倍，3倍，4倍，...，删掉
5. 剩下的没被删掉的都是素数

举个栗子，这里取**100**以内的数

1	2	3	4	5	6	7	8	9	10
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

去掉特殊的1

	2	3	4	5	6	7	8	9	10
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

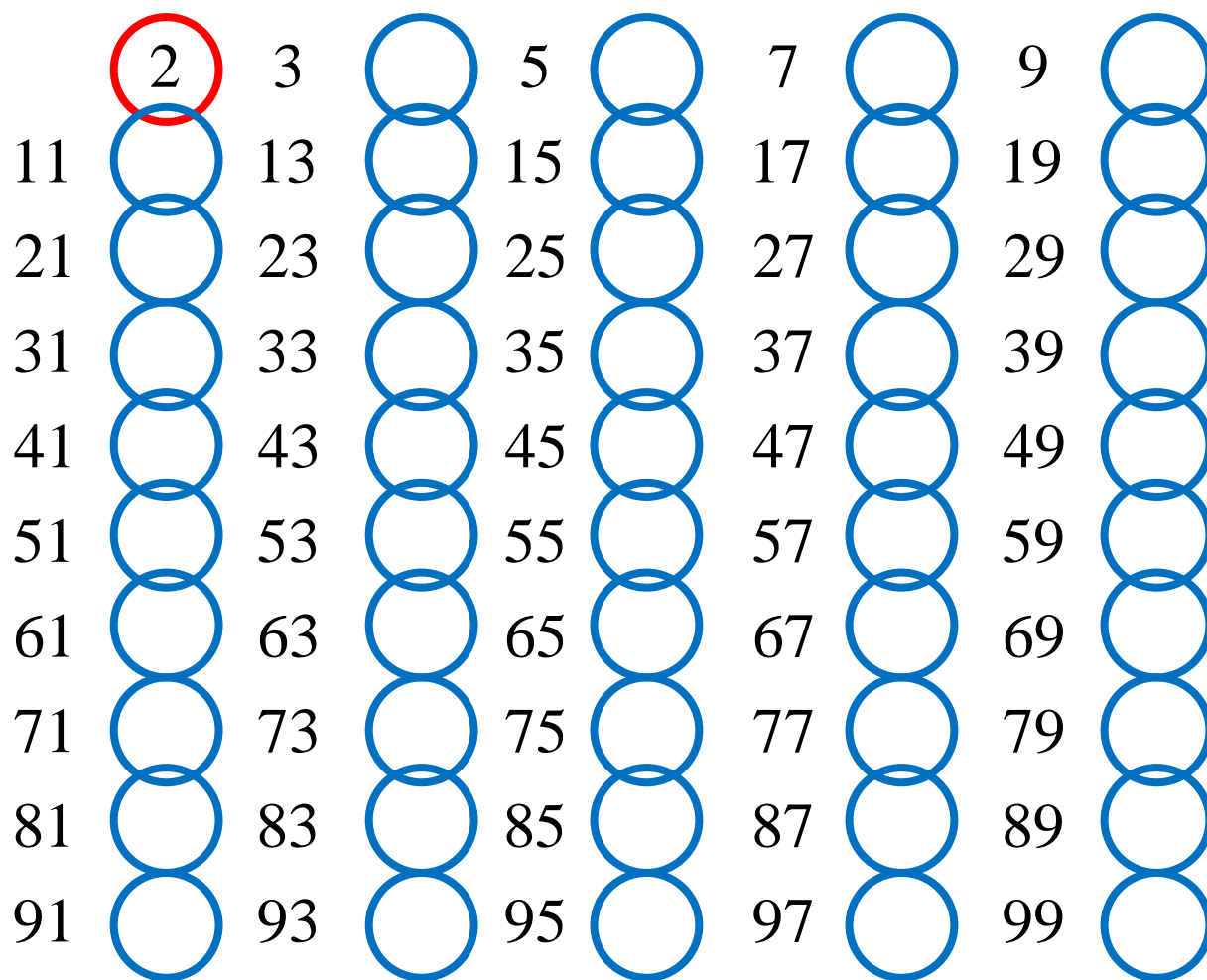
枚举*i*=2

	2	3	4	5	6	7	8	9	10
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

把i的2倍， 3倍， 4倍...删除

	2	3	4	5	6	7	8	9	10
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

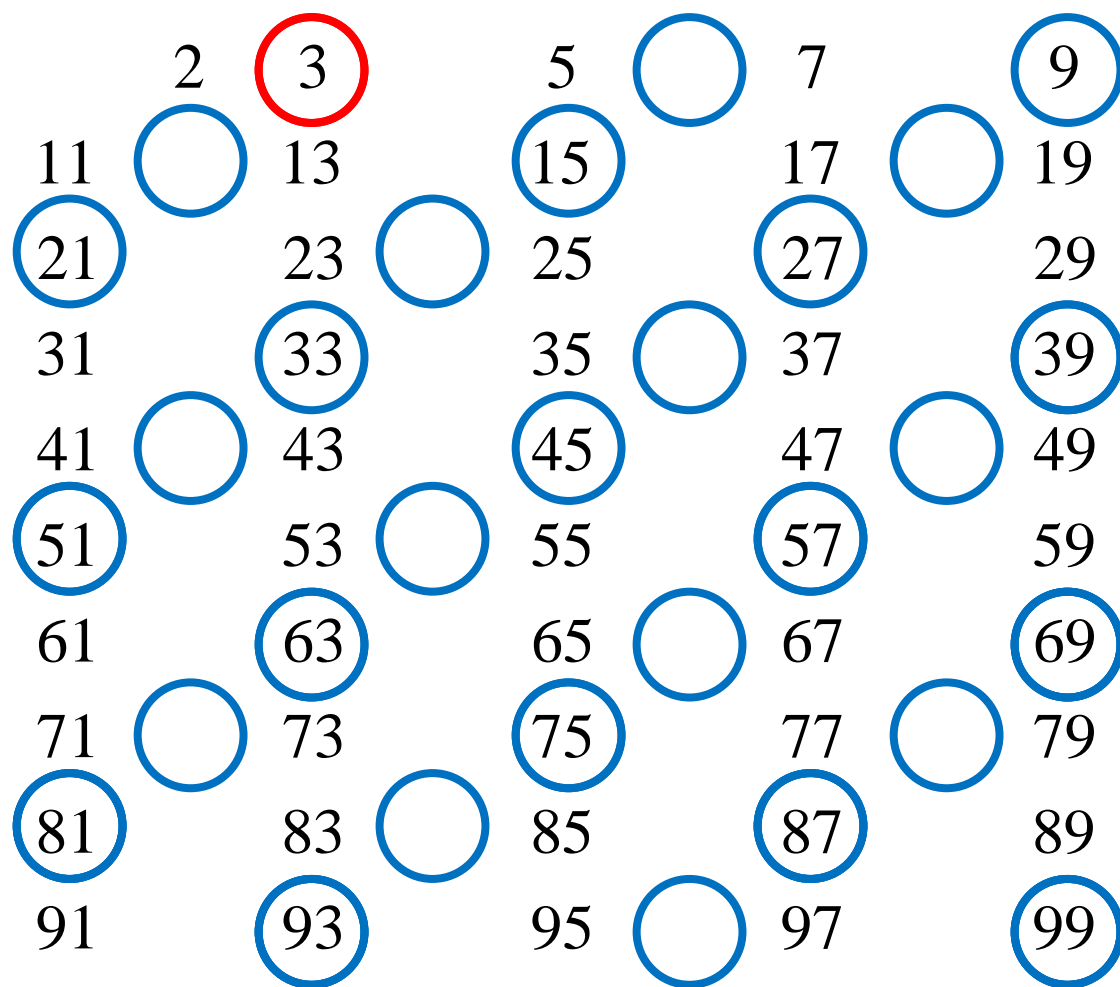
把i的2倍， 3倍， 4倍...删除



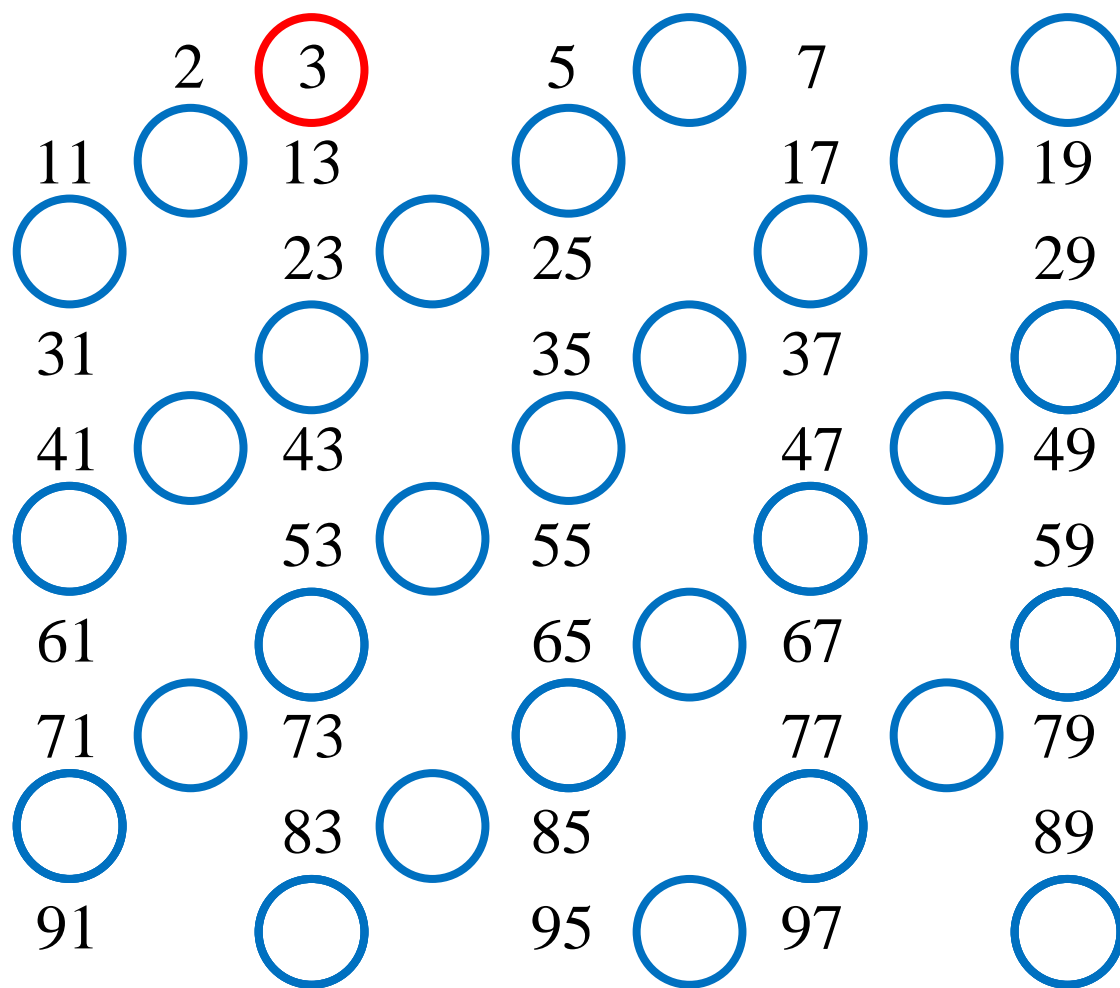
枚举下一个没有被删除的数 $i=3$

	2	3	5	7	9
11	13	15	17	19	
21	23	25	27	29	
31	33	35	37	39	
41	43	45	47	49	
51	53	55	57	59	
61	63	65	67	69	
71	73	75	77	79	
81	83	85	87	89	
91	93	95	97	99	

把i的2倍，3倍，4倍...删除



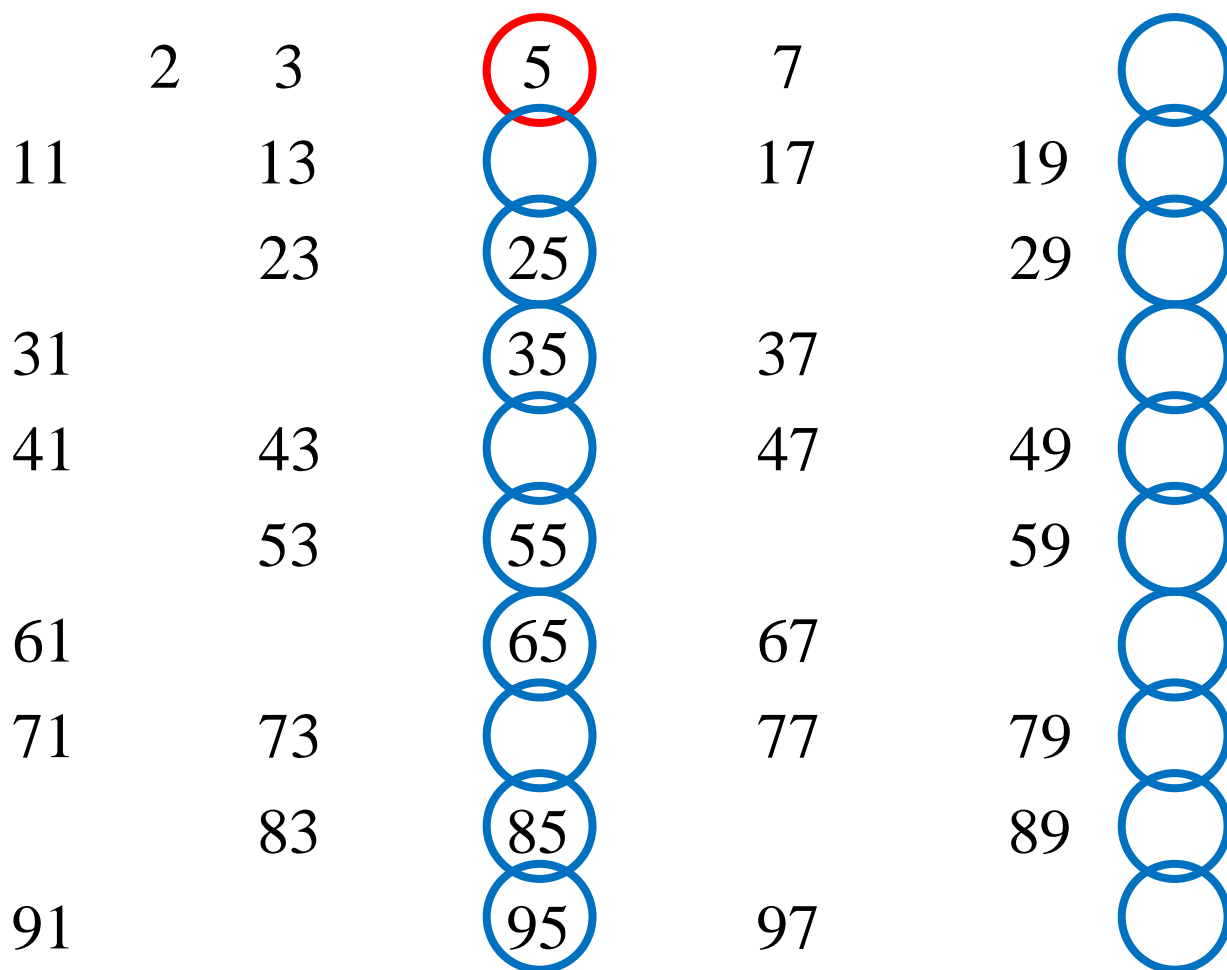
把i的2倍，3倍，4倍...删除



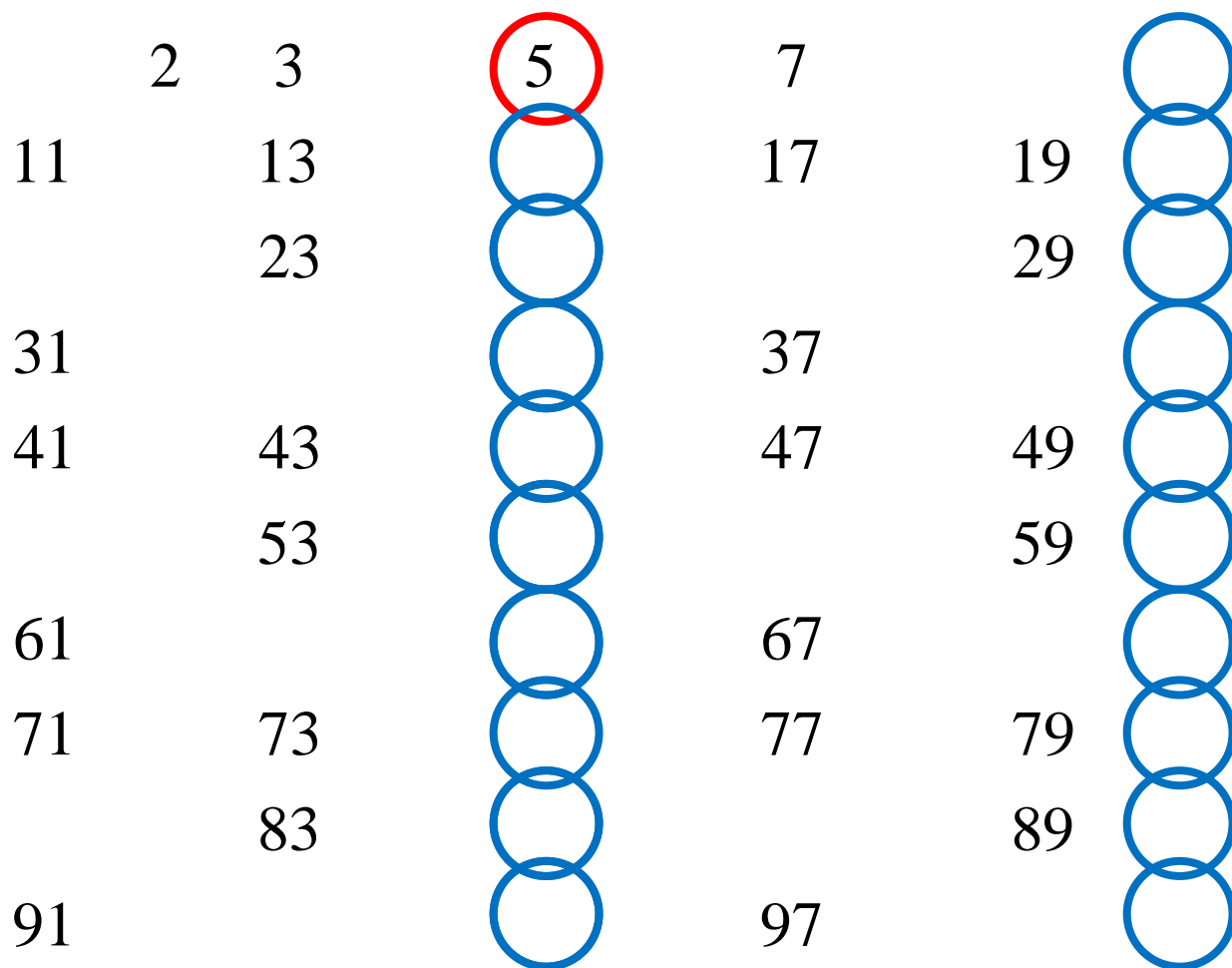
枚举下一个没有被删除的数 $i=5$

	2	3	5	7	
11		13		17	19
		23	25		29
31			35	37	
41		43		47	49
		53	55		59
61			65	67	
71		73		77	79
		83	85		89
91			95	97	

把i的2倍，3倍，4倍...删除



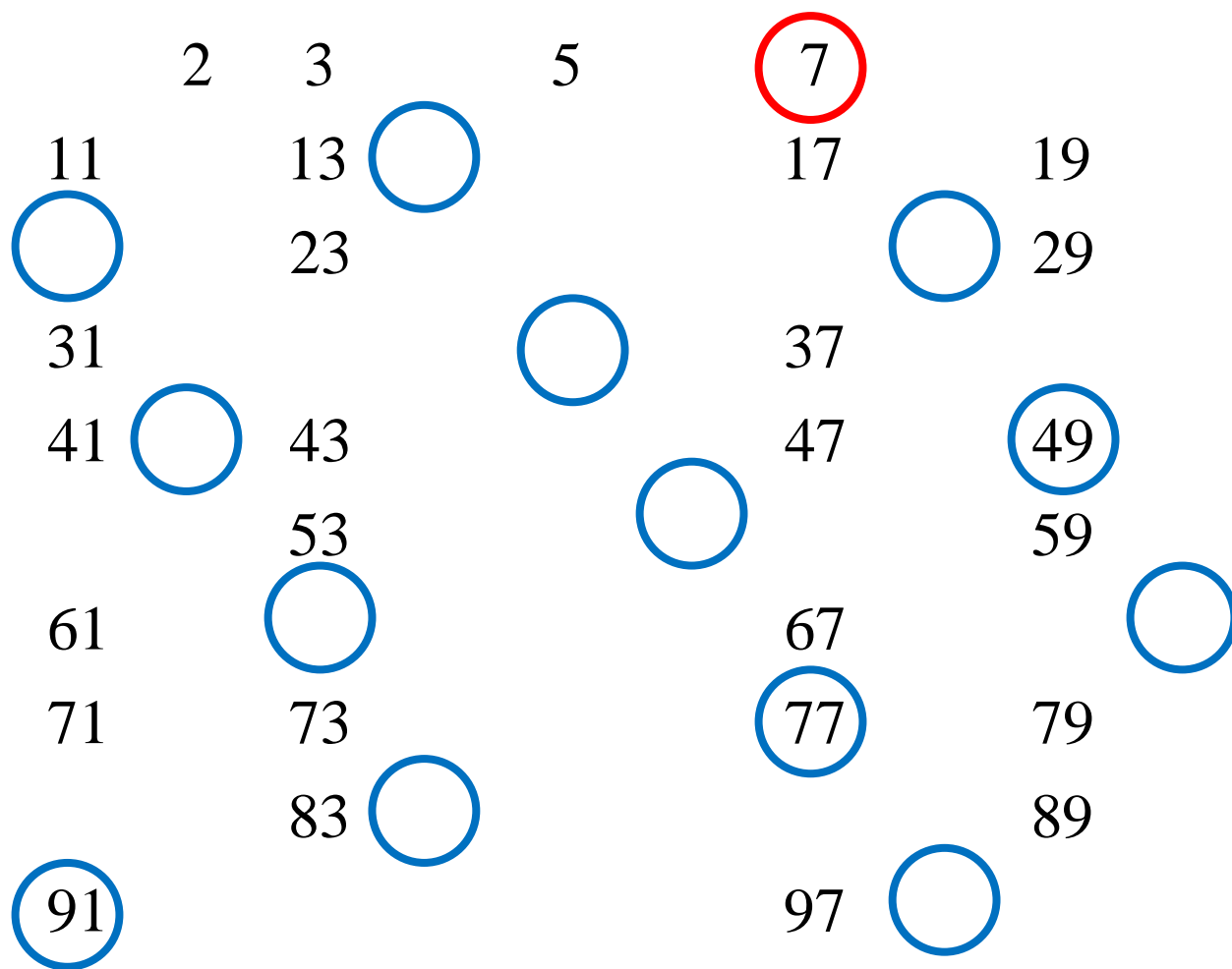
把i的2倍，3倍，4倍...删除



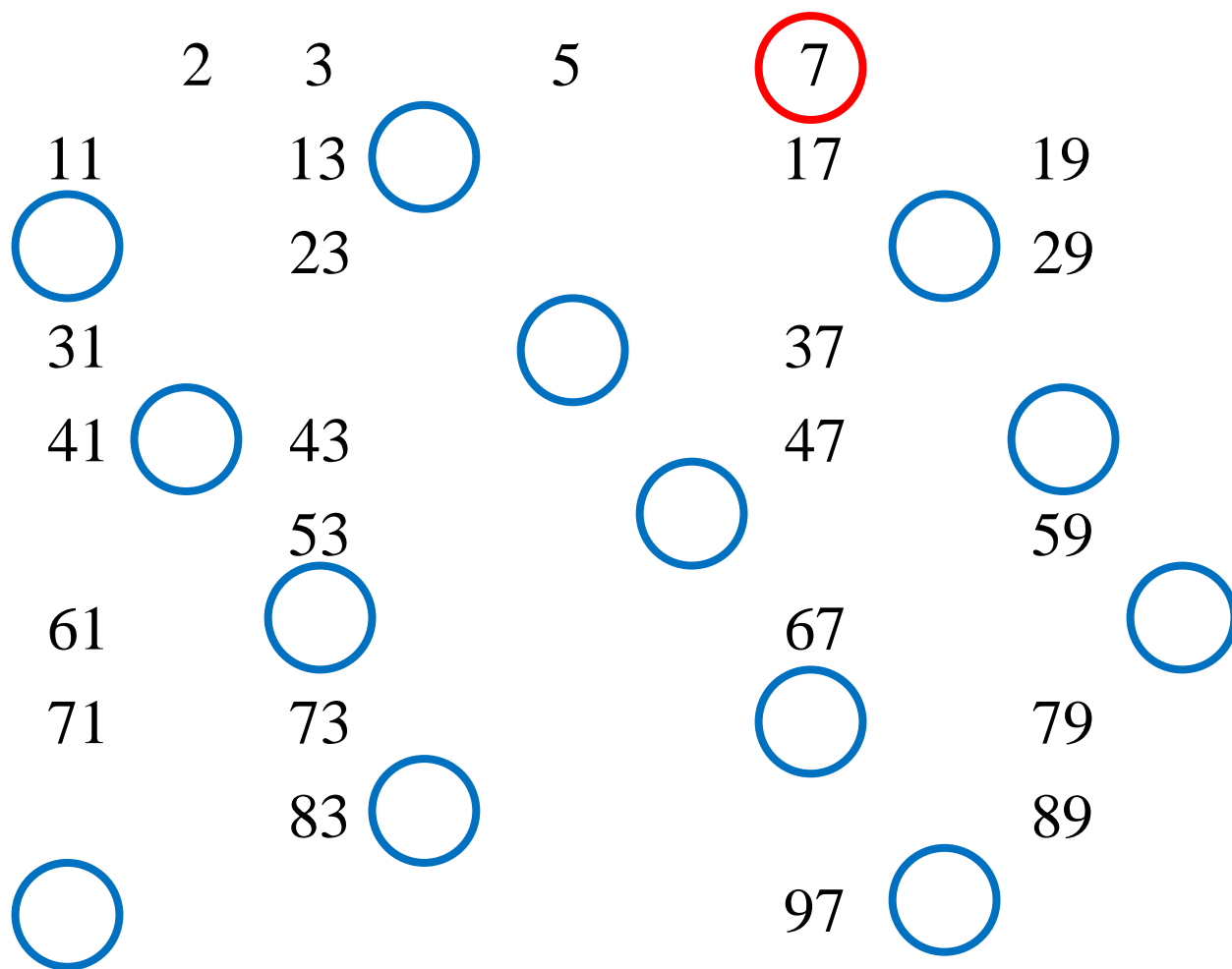
枚举下一个没有被删除的数 $i=7$

	2	3	5	7	
11		13		17	19
		23			29
31				37	
41		43		47	49
		53			59
61				67	
71		73		77	79
		83			89
91				97	

把i的2倍，3倍，4倍...删除



把i的2倍，3倍，4倍...删除



得到1~100以内的全体素数

	2	3	5	7	
11		13		17	19
		23			29
31				37	
41		43		47	
		53			59
61				67	
71		73			79
		83			89
				97	

代码实现

```
const int N=1000001; //需要找出1~N-1的质数
bool notprime[N]; //删除标记，被删除说明不是质数
void init()
{
    notprime[1]=1; //标记特殊的1不是质数
    for(int i=2; i<N; i++) //枚举i
    {
        if(notprime[i]==false) //找到一个没有被删除的数
        {
            for(int j=i+i; j<N; j=j+i) //把j=2i, 3i, 4i, ...找出来
            {
                notprime[j]=true; //把这些数标记为删除
            }
        }
    }
}
```

复杂度分析

这个循环执行了 $N-1$ 次

对于每一个 i ,
这个循环被执行了 N/i 次

```
const int N=1000001; //需要
bool notprime[N]; //删除标记
void init()
{
    notprime[1]=1; //标记特殊的1不是质数
    for(int i=2; i<N; i++) //枚举i
    {
        if(notprime[i]==false) //找到一
        {
            for(int j=i+i; j<N; j=j+i) //把j=2i, 3i, 4i, ...找出来
            {
                notprime[j]=true; //把这些数标记为删除
            }
        }
    }
}
```

复杂度分析

- 最核心的循环被执行了的次数

$$\begin{aligned} & \frac{N}{2} + \frac{N}{3} + \frac{N}{5} + \frac{N}{7} + \dots \\ &= N \left(\frac{1}{2} + \frac{1}{3} + \frac{1}{5} + \frac{1}{7} + \dots \right) \\ &= N(\ln \ln N + 0.281\dots) \end{aligned}$$

$$O(n \log \log n)$$



根据Mertens
定理

容易

想要获取1~N的素数，标记数组至

别忘了标记特殊

只对没被删除的数进行操作

每次累加的间隔是i

```
const int N=1000001;
bool notprime[N]; // 删除
void init()
{
    notprime[1]=1; // 标记特殊的1
    for(int i=2; i<N; i++) // 枚举i
    {
        if(notprime[i]==false) // 找到一个没有被删除的数
        {
            for(int j=i+i; j<N; j=j+i) // 把j=2i, 3i, 4i, ... 找出来
            {
                notprime[j]=true; //
            }
        }
    }
}
```

用这个方法我们能干什么？

1秒内把 10^7 以内的质数和合数
分隔出来

能不能再快一些？

- 有的合数被删除了几次
- 例如6在 $i=2$ 和 $i=3$ 时各删除了1次
- 例如60在 $i=2$ 和 $i=3$ 和 $i=5$ 时各被删除了1次
- 没必要删除那么多次！
- 如何让每个合数只被删除1次？

欧拉筛(线性筛)

- 特点每个合数，只会被最小素因子筛去
 1. 列出 $1 \sim n$
 2. 删除特殊的1
 3. 从2开始枚举数 i ，若没有被删掉，则加入素数表中
 4. 枚举素数表中的每一个素数 j ，并把 $i*j$ 的结果从表里删去，直到枚举到 j 能整除 i 。

还是这个栗子，这里取**100**以内的数

素数表 = {}

1	2	3	4	5	6	7	8	9	10
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

去掉特殊的1

素数表 = { }

	2	3	4	5	6	7	8	9	10
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

枚举*i*=2

素数表 = { }

	2	3	4	5	6	7	8	9	10
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

因为没有被删除，加入素数表

素数表 = {2}

	2	3	4	5	6	7	8	9	10
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

枚举素数表j=2

素数表 = {2}

	2	3	4	5	6	7	8	9	10
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

删除掉 $i*j=4$

素数表 = {2}

	2	3	4	5	6	7	8	9	10
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

因为j能整除i， 停止向后删除

素数表 = {2}

	2	3		5	6	7	8	9	10
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

枚举*i*=3

素数表 = {2}

	2	3		5	6	7	8	9	10
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

3没有被删除，加入素数表

素数表 = {2,3}

	2	3		5	6	7	8	9	10
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

枚举素数表j=2,3

素数表 = {2,3}

	2	3		5	6	7	8	9	10
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

标记 $3*2=6$, $3*3=9$

素数表 = {2, 3}

	2	3		5	6	7	8	9	10
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

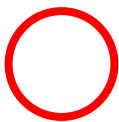
删除6和9

素数表 = {2, 3}

	2	3		5		7	8		10
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

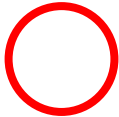
枚举*i*=4

素数表 = {2,3}

	2	3		5		7	8		10
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

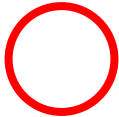
因为4被删除了，不加入素数表

素数表 = {2,3}

	2	3		5		7	8		10
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

枚举 $j=2$ ，因为2能整除4，不再枚举
后面的3

素数表 = {2, 3}

	2	3		5		7	8		10
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

标记 $4*2=8$

素数表 = {2, 3}

	2	3		5		7	8		10
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

删除掉8

素数表 = {2, 3}

	2	3		5		7		10	
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

枚举 $i=5$ ， 没被删除， 加入素数表

素数表 = {2 ,3 ,5}

	2	3		5		7			10
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

枚举j=2,3,5

素数表 = {2, 3, 5}

	2	3		5		7			10
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

标记 $5*2=10, 5*3=15, 5*5=25$

素数表 = {2, 3, 5}

	2	3		5		7			10
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

删除10,15,25

素数表 = {2, 3, 5}

2				5	7				
11	12	13	14		16	17	18	19	20
21	22	23	24		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

枚举*i*=6，因为已经被删除，不加入
素数表

素数表 = {2, 3, 5}

2, 3, 5, 7									
2	3			5	<div></div>	7			
11	12	13	14		16	17	18	19	20
21	22	23	24		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

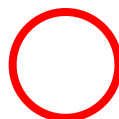
枚举j=2,因为2能整除6, 不再继续枚举3和5

素数表={2,3,5}

2

3

5



7

11	12	13	14		16	17	18	19	20
21	22	23	24		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

标记 $6*2=12$ ，并删除

素数表 = {2, 3, 5}

	2	3		5		7			
11		13	14		16	17	18	19	20
21	22	23	24		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

枚举 $i=7$ ， 没被删除， 加入素数表

素数表 = {2 ,3 ,5 ,7}

2

3

5

7

11		13	14		16	17	18	19	20
21	22	23	24		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

枚举 $j=2,3,5,7$,标记14,21,35,49,并删除






素数表 = {2, 3, 5, 7}

2

3

5

7

11		13			16		17	18	19	20
	22	23	24		26	27	28	29	30	
31	32	33	34		36	37	38	39	40	
41	42	43	44	45	46	47	48		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	

如此类推，最终枚举完*i*=97后，即可

得到100以内的素数
素数表 = {2,3,5,7,11,13,17,19,23,29,31,37,

41,43,49,53,59,61,67,71,73,79,83,89,97}

	2	3	5	7
11		13		17
		23		29
31				37
41		43		47
		53		59
61				67
71		73		79
		83		89
				97

时间复杂度

- 每一个素数都只被记录一次，每一个合数都只会被一个唯一的最小素因子标记一次(例如6只被2标记了，没有被3标记)。因此是 $O(n)$ 线性复杂度的。

代码实现

```
const int N=100000001;//需要找出1~N-1的质数
bool notprime[N]);//删除标记，被删除说明不是质数
int prime[N],pn;//素数表
void init()
{
    pn=0;//初始化素数表为空
    notprime[1]=1;//标记特殊的1不是质数
    for(int i=2;i<N;i++)//枚举i
    {
        if(notprime[i]==false)//找到一个没有被删除的数
        {
            prime[pn++]=i;//加入素数表
        }
        for(int j=0;j<pn&&prime[j]*i<N;j++)//枚举素数表
        {
            notprime[prime[j]*i]=true;//把i和素数表相乘的合数标记
            if(i%prime[j]==0)//如果遇到枚举的素数是i的约数，跳出循环
                break;
        }
    }
}
```

和之前的比较

- 1秒钟能处理的数更多了

$$10^7 \rightarrow 10^8$$

- 代码实现更复杂了

$$16\text{行} \rightarrow 21\text{行}$$

容易出错的地方

```
const int N=100000001;//需要找出1~N-1的质数
bool notprime[N]);//删除标记，被删除说明不是质数
int prime[N],pn;//素数表
void init()
{
    pn=0;//初始化素数表为空
    notprime[1]=1;//标记特殊的1不是质数
    for(int i=2;i<N;i++)//枚举i
    {
        if(notprime[i]==false)//找到质数
        {
            prime[pn++]=i;//加入素数表
        }
        for(int j=0;j<pn&&prime[j]*i<N;j++)
        {
            notprime[prime[j]*i]=true;
            if(i%prime[j]==0)//如遇到i的倍数，提前结束
                break;
        }
    }
}
```

和前一个算法不同，无论有没有标记，都要进入这个循环

这个整除提前结束的判断是保证线性的关键

素数筛应用

1. 预处理 n 以内的素数
2. 可以顺便预处理很多数论中常见的函数
3. 简化其它数论算法的操作

几乎是所有数论题的开始

最大公约数

- 欧几里得算法(辗转相除法)
`#include <algorithm>`里面有gcd函数

C++才有的
算法库

两个下划线

欧几里得算法

$$\gcd(a, b) = \begin{cases} \gcd(b, a \% b) & b \neq 0 \\ a & b = 0 \end{cases}$$

代码实现

```
int gcd(int a, int b)
{
    if (b==0)
    {
        return a;
    }
    return gcd(b, a%b);
}
```

```
long long gcd(long long a, long long b)
{
    if (b==0)
    {
        return a;
    }
    return gcd(b, a%b);
}
```



64位整数的
版本

多个数最大公约数

- 最大公约数满足结合律

$$\gcd(a_1, a_2, \dots, a_n) = \gcd(\gcd(a_1, a_2, \dots, a_{n-1}), a_n)$$

多个数最大公约数

```
int gcd(int a, int b)
{
    if (b == 0)
    {
        return a;
    }
    return gcd(b, a % b);
}

int multi_gcd(int a[], int n)
{
    int res = gcd(a[0], a[1]);
    for (int i = 2; i < n; i++)
        res = gcd(res, a[i]);
    return res;
}
```

最小公倍数

$$lcm(a_1, a_2) = \frac{a_1 \cdot a_2}{\gcd(a_1, a_2)}$$

$$lcm(a_1, a_2, \dots, a_n) = lcm(lcm(a_1, a_2, \dots, a_{n-1}), a_n)$$

代码实现

```
int gcd(int a, int b)
{
    if (b == 0)
    {
        return a;
    }
    return gcd(b, a % b);
}

int lcm(int a, int b)
{
    return a / gcd(a, b) * b; // 防止a*b溢出，先除后乘
}
```

扩展欧几里得算法

已知 a, b, c , 以及 $ax + by = c$ 求出一组解 x, y

有解条件: $\gcd(a, b) \mid c$

$$ax_1 + by_1 = \gcd(a, b)$$

$$bx_2 + (a \% b)y_2 = \gcd(b, a \% b)$$

$$ax_1 + by_1 = bx_2 + (a \% b)y_2$$

$$a \div b = a / b \dots\dots\dots a \% b \Leftrightarrow a = [a / b] \cdot b + a \% b$$

$$ax_1 + by_1 = bx_2 + (a - [a / b] \cdot b)y_2$$

$$ax_1 + by_1 = ay_2 + b(x_2 - [a / b]y_2)$$

$$\begin{cases} x_1 = y_2 \\ y_1 = x_2 - [a / b]y_2 \end{cases}$$

递归版

```
int exgcd(int a, int b, int&x, int&y)
{
    if (b==0)
    {
        x=1, y=0; // 当b=0时, ax=gcd(a, 0), x=1, y=0
        return a;
    }
    int d=exgcd(b, a%b, x, y); // 递归处理, 获取x2和y2
    int t=x;
    x=y; // x1=y2
    y=t-a/b*y; // y1=x2-a/b*y2
    return d;
}
```

应用

- 求解线性同余方程 $ax \equiv b(\text{mod } n)$
- 求逆元 $ax \equiv 1(\text{mod } n)$

解线性同余方程组

$$ax \equiv b \pmod{n} \Rightarrow ax - ny = b$$

已知 a, n , 找出一组解 x, y 满足方程

```
int solve(int a, int b, int n)
{
    int d=gcd(a, n);
    if (b%d!=0)
        return -1; //无解
    int k=b/d;
    int x, y;
    exgcd(a, n, x, y); //求解ax-ny=gcd(a, n)
    return x*k;
}
```


求逆元

$ax \equiv 1 \pmod{n}$, 就是刚才 $b = 1$ 的情况。

分解质因数

- 算术基本定理：任何一个大于1的自然数N，如果N不是素数，那么N可以唯一地表示为

$$N = p_1^{e_1} p_2^{e_2} p_3^{e_3} \cdots p_r^{e_r}$$

其中 $p_1 < p_2 < \cdots < p_r$ 为素数，

e_1, e_2, \dots, e_r 为正整数。

分解质因数

方法一：枚举因子，不断试除 ($O(\sqrt{n})$)

方法二：素数筛预处理最小素因子，循环除以最小素因子 ($O(n) + O(\log n)$)

代码实现

- 预处理素数筛

```
int p[100];
int e[100];
int cnt;
int fenjie(int n)
{
    cnt=0;
    for(int i=0;i<pn&&prime[i]<=n;i++)
        if(n%prime[i]==0)
        {
            p[cnt]=prime[i];
            while(n%prime[i]==0)
                n/=prime[i],e[cnt]++;
            cnt++;
        }
    if(n>1)
        p[cnt]=n,e[cnt++]=1;
}
```

快速幂

$$a^b = \begin{cases} a^{\frac{b}{2}} \cdot a^{\frac{b}{2}}, & b \text{ 是偶数} \\ a^{\frac{b-1}{2}} \cdot a^{\frac{b+1}{2}}, & b \text{ 是奇数} \end{cases}$$

递归版

```
long long pow_mod(long long a, long long n, long long m)
{
    if(n==0) return 1;
    long long res=pow_mod(a, n/2, m);
    if(n%2==0)
        return res*res%m;
    else
        return res*res%m*a%m;
}
```

非递归版

```
long long pow_mod(long long a, long long n, long long m)
{
    long long res=1;
    while (n>0)
    {
        if (n%2==1)
            res=res*a%m;
        a=a*a%m;
        n/=2;
    }
    return res;
}
```

快速幂时间复杂度

- $O(\log b)$ 次递归/循环

费马小定理

- 1640年费马提出的一个定理：

若 p 是一个素数，那么对于任意一个整数 a ，

$a^p - a$ 都是 p 的倍数，即 $a^p \equiv a \pmod{p}$

若 $\gcd(a, p) = 1$ ，则 $a^{p-1} \equiv 1 \pmod{p}$

一个优雅的证明

考虑二项式系数 C_p^n ，当 $n = 0$ 或 $n = p$ 时， $C_p^n = 1$ 。

当 $0 < n < p$ 时， $C_p^n = \frac{p!}{n!(p-n)!}$ ，分子是 p 的倍数，

分母不是 p 的倍数，故 $C_p^n \equiv 0 \pmod{p}$ 。

$$a^p = ((a-1)+1)^p = \sum_{n=0}^p C_p^n (a-1)^{p-n} \equiv (a-1)^p + 1 \pmod{p}$$

$$\equiv (a-2)^p + 1 + 1 \pmod{p} \equiv (a-3)^p + 1 + 1 + 1 \pmod{p}$$

$$\equiv \dots \equiv 1 + 1 + \dots + 1 + 1 \pmod{p} = a \pmod{p}$$

费马小定理的意义

- 说明了模素数的指数是有循环节的，且循环节长度为 $p-1$

欧拉函数

$\varphi(n)$ 为欧拉函数,

定义为不超过 n 的整数中与 n 互素的个数

$$\varphi(3)=2, \quad \varphi(6)=2, \quad \varphi(100)=40$$

如何求欧拉函数？

1. 枚举所有不超过 n 的数，判断是否互质。
($O(n \log n)$)
2. 欧拉函数求值公式($O(\sqrt{n})$)

欧拉函数的性质

当 n 是1时, $\varphi(n)=1$

当 p 是个素数时, $\varphi(p^k)=p^k - p^{k-1}$

当 a, b 互质时, $\varphi(ab)=\varphi(a)\varphi(b)$

欧拉函数求值公式

$$n = p_1^{e_1} p_2^{e_2} \cdots p_r^{e_r}$$

$$\varphi(n) = \varphi(p_1^{e_1}) \varphi(p_2^{e_2}) \cdots \varphi(p_r^{e_r})$$

$$\varphi(n) = n \left(1 - \frac{1}{p_1}\right) \left(1 - \frac{1}{p_2}\right) \cdots \left(1 - \frac{1}{p_r}\right)$$

单次求欧拉函数

```
int getphi(int n)
{
    int ans=n;
    for (int i=2;i*i <=n;i++)
    {
        if(n%i==0)
        {
            ans-=ans/i;
            while(n%i==0)
                n/=i;
        }
    }
    if(n>1)
        ans-=ans/n;
    return ans;
}
```

欧拉函数的应用

- 求逆元
- 欧拉降幂公式

欧拉定理的应用

- 求逆元

$$a^{\varphi(n)} \equiv 1 \pmod{n} \Leftrightarrow a \cdot a^{\varphi(n)-1} \equiv 1 \pmod{n}$$

$a^{\varphi(n)-1}$ 就是在模 n 下 a 的逆元

使用条件为 $(a, n) = 1$

欧拉定理的应用

- 欧拉降幂公式

$$a^b \equiv a^{b \% \varphi(n) + \varphi(n)} \pmod{n} (b > \varphi(n))$$

可以把大指数转化成不超过 $\varphi(n)$ 的指数

注意，这里不要求 a, n 互质！

注意，这里不要求 a, n 互质！！

注意，这里不要求 a, n 互质！！!(重要的话说三次!)

- 降幂，化简运算

例题：HDU4704 (2013 多校10)

给定 N ，定义 S_k 为 $x_1 + x_2 + \dots + x_k = N$ 的正整数解的方案数，

求 $(S_1 + S_2 + \dots + S_N) \% 1000000007$

$1 < N < 10^{20000}$

类似的题

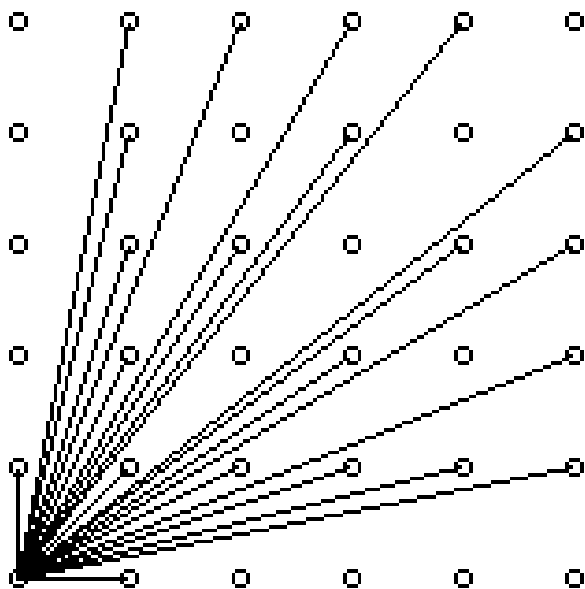
- FZU1759 Super $A^B \bmod C$
计算 $A^B \bmod C$.
($1 \leq A, C \leq 1e9, 1 \leq B \leq 1e1000000$).
- BZOJ 3884 上帝与集合的正确用法

给定 $p \leq 100000000$, 设 $a_0 = 1$, $a_n = 2^{a_{n-1}}$, 求 $\lim_{n \rightarrow \infty} a_n \% p$

即求 $2^{2^{2^{2^{2^{2^{2^{\dots}}}}}}}} \% p$ (无数个2)

- POJ3090

有一个 $n \times n$ 的二维格点，问在 $(0,0)$ 处能看到多少个格点？ ($n \leq 1000$, 1000组数据)



$n=5$ 的情况，答案是21，
如左图的21根线

$$|\{(x, y) | 1 \leq x, y \leq n, \gcd(x, y) = 1\}|$$

中国剩余定理

$$\text{已知} \left\{ \begin{array}{l} x \equiv a_1 \pmod{n_1} \\ x \equiv a_2 \pmod{n_2} \\ \dots \\ x \equiv a_k \pmod{n_k} \end{array} \right., \text{求 } x$$

孙子算经

- 今有物不知其数,三三数之剩二,五五数之剩三,七七数之剩二.问物几何?

$$\text{已知} \begin{cases} x \equiv 2(\text{mod } 3) \\ x \equiv 3(\text{mod } 5), \text{求} x \\ x \equiv 2(\text{mod } 7) \end{cases}$$

孙子算经解答

- 凡三三数之剩一,则置七十; 五五数之剩一,则置二十一; 七七数之剩一,则置十五.一百六以上,以一百五减之,即得

$$\text{已知} \begin{cases} x \equiv 2 \pmod{3} \\ x \equiv 3 \pmod{5} \\ x \equiv 2 \pmod{7} \end{cases}, \text{求} x$$

$$(2 \times 70 + 3 \times 21 + 2 \times 15) \% 105 = 23$$

中国剩余定理

1. 计算 $N = LCM(n_1, n_2, \dots, n_k)$

2. 计算 $N_i = \frac{N}{n_i}$

3. 利用 *ex gcd* 计算 $M_i N_i + m_i n_i = 1$

4. $x = \left(\sum_{i=1}^k a_i M_i N_i \right) \% N$

例题

- hdu1370

一个人有体力，感情，智商三个周期，周期分别为23天，28天，33天，然后告诉你今天分别位于三个周期的第几天，问下一次三者达到顶峰的日子。

离散对数

- 什么是对数？

已知 $a^x = b$ 的 a, b ，那么 $x = \log_a b$ ，称为以 a 为底 b 的对数

- 什么是离散对数？
在模 n 下的对数运算

已知 $a^x \equiv b \pmod{n}$ 的 a, b, n ，那么 $x \equiv \log_a b \pmod{n}$ ，
称为在模 n 下以 a 为底 b 的对数

怎么求离散对数？

- 这是一个很经典的工业问题，因为没有 $O(\log c)$ 复杂度的解法，所以可以作为公钥加密算法——DH 密钥交换算法
- 竞赛中只需要掌握 $O(\sqrt{c})$ 的算法即可
- 求解算法
- Baby Step Giant Step (竞赛中常用)
- Pollard-rho (对，又是这个人)

Baby Step Giant Step

- 一个用空间换时间的算法(hash表判重)
- 一个中间相遇法的算法(节省一半搜索状态)

算法思想

假设我们解的方程 $a^x \equiv b \pmod{n}$ 的结果是 x

把 x 写成 $x = im + j$ ($0 \leq i, j < m$), 并且取 $m = \lceil \sqrt{n} \rceil$

$$(a^m)^i a^j \equiv b \pmod{n} \Leftrightarrow a^j \equiv b (a^{-m})^i \pmod{n}$$



baby step

hash

giant step

复杂度

- 因为 i, j 都是不超过 $m = \sqrt{n}$ 的数，因此整体复杂度是 $O(\sqrt{n})$ 的

应用条件？

- n 必须是素数。
- 如果 n 不是素数会出现什么问题？

ex-baby step giant step

- 分析
- n 不是素数之所以不成立是因为 $(a,n) \neq 1$ 导致逆元不存在。

设 $g = \gcd(a, n)$

$$a^x - kn = b$$

当 g 不能整除 b 时，必然无解

两边同时除以 g

$$\frac{a^x}{g} - \frac{kn}{g} = \frac{b}{g} \Leftrightarrow \frac{a}{g} a^{x-1} \equiv \frac{b}{g} \left(\text{mod } \frac{n}{g} \right) \Leftrightarrow a^{x-1} \equiv \frac{b}{g} \cdot \left(\frac{a}{g} \right)^{-1} \left(\text{mod } \frac{n}{g} \right)$$

手算一下？

$$6^x \equiv 8 \pmod{16}$$

$$6^x \equiv 8 \pmod{16}$$

$$\gcd(6, 16) = 2$$

$$\frac{6}{2} \cdot 6^{x-1} \equiv \frac{8}{2} \left(\pmod{\frac{16}{2}} \right)$$

$$6^{x-1} \equiv 4 \cdot 3^{-1} \equiv 4 \pmod{8}$$

$$6^{x-2} \equiv 2 \cdot 3^{-1} \equiv 2 \pmod{4}$$

$$6^{x-3} \equiv 1 \cdot 3^{-1} \equiv 1 \pmod{2}$$

$$x - 3 = 0$$

$$x = 3$$

模板题

- POJ2417
找到最小的L，满足 $B^L \equiv N \pmod{P}$ ，其中P是素数。
- HDU2815 MOD Tree
求最小的D，满足 $K^D \equiv N \pmod{P}$ ，不存在输出"Orz,I can't find D!"
P.S. 注意这里的can右上角的那一撇是全角的引号.....为了减少不必要的WA，请直接复制题面...