# 习题一：找最高成绩的学生

班上组织了一次考试，全班60位同学都获得一个0-400分的分数。请问，在这些同学中最高分是多少？

输入
60个空格分割的成绩（均在0到400之间）

输出
一个数字，表示最高分。

样例
输入
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
输出
0

# 分析

方法一　定义一个大小为60的数组，存储所有的分数
再遍历得到最大值

```
int a[61];
for(i:1~60){
    输入a[i];
}
for(i:1~60){
   找到最大值
}
```

方法二　输入的时候，直接存储并更新最大值

```
int x,Max=-100;
for(i:1~60){
    输入x;
    更新Max;
}
```

Q:什么时候使用数组?

需要存储多个相同类型的数据的时候

改动一：得到全班的平均分　**不需要数组**
改动二：得到全班的平均分，并输出高于平均分的人数　**需要数组**

# 习题二：插入

依次输入n个数（从小到大排列），再输入一个数，将其插入到这组数中，使其依然有序，再输出。

输入
第一行一个数n（n<100）
第二行n个int范围以内的数，之间以空格隔开
第三行输入要插入的数

输出
插入后，仍然有序的数列

样例
输入
5
1 3 5 7 9
4

输出
1 3 4 5 7 9

# 分析

**与课上的例题不同之处：**

课上的题目：给定了插入的位置和插入的数

题单的题目：没有给定插入的位置，但是给定了插入的数

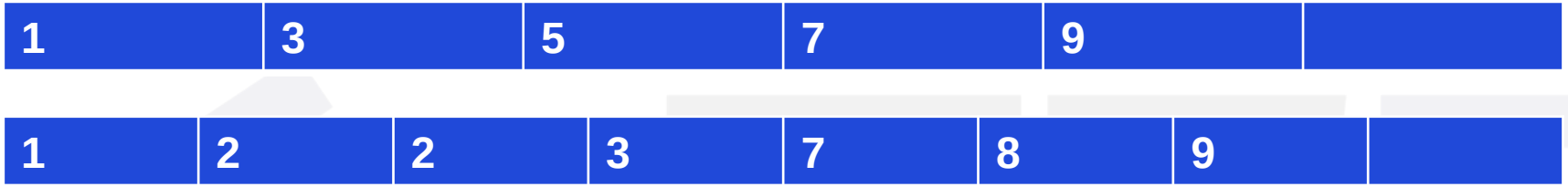
需要自己想到一个办法，找出插入的位置

**Q:如何找到插入的位置?**

切入点： n个数（从小到大排列），再输入一个数，
将其插入到这组数中，使其依然有序

# 分析

4




Q:4应该放在哪个位置?

放在5的位置
放在7的位置

放在第一个大于x的位置

找到第一个大于x的位置:

```
for(遍历整个数组a){
    if(a[i]>x){
        记录当前的下标
        break;
    }

}
```

插入x的位置

# 算法流程

## 算法

输入序列，存到数组a

利用循环找到插入的位置，并记录
(核心：找到第一个大于x的位置)

移动位置，并插入x

输出序列

(参考代码略，自行实现)

# 习题三：十进制转二进制

## 输入一个整数n（n<100000000），输出该数的二进制数。

背景知识：
二进制是逢2进位的进位制，因此它只使用到它只使用0、1两个数字符号。十进制转换为二进制的算法为：采用"除2取余，逆序排列"法。具体做法是：用2整除十进制整数，可以得到一个商和余数；再用2去除商，又会得到一个商和余数，如此进行，直到商为0时为止，然后把先得到的余数作为二进制数的低位有效位，后得到的余数作为二进制数的高位有效位，依次排列起来。
如：
789=1100010101
789/2=394 余1 第10位
394/2=197 余0 第9位
197/2=98 余1 第8位
98/2=49 余0 第7位
49/2=24 余1 第6位
24/2=12 余0 第5位
12/2=6 余0 第4位
6/2=3 余0 第3位
3/2=1 余1 第2位
1/2得0 余1 第1位

输入
第一行一个数n

输出
该数的二进制数

样例输入
789

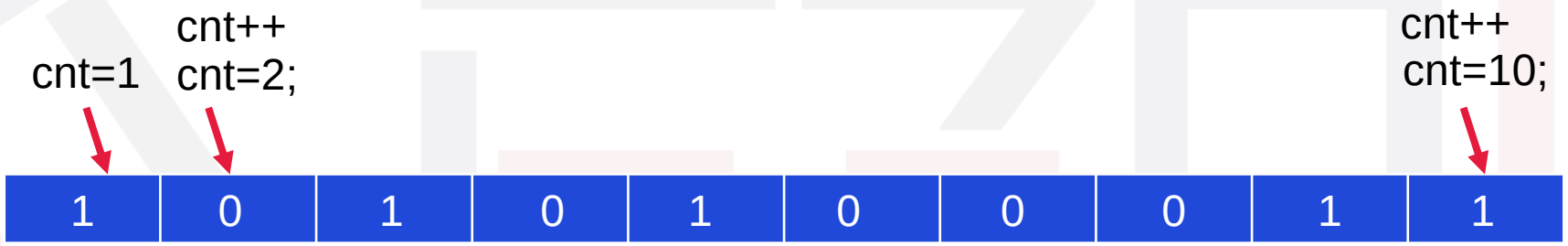
样例输出
1100010101

# 分析

789=1100010101

除2取余法

789/2=394 余1
394/2=197 余0
197/2=98 余1
98/2=49 余0
49/2=24 余1
24/2=12 余0
12/2=6 余0
6/2=3 余0
3/2=1 余1
1/2得0 余1




核心思路：

设立一个“游动”的下标cnt，实时更新放入数组的数字个数

a[cnt]=n%2;
n=n/2;
cnt++;

直到n为0为止

逆序输出答案

# 代码

```
int n,i,cnt=1; //cnt记录二进制位数
int a[50];
cin>>n;
//开始模拟
while(n!=0)
{
    a[cnt]=n%2;//余数存入数组
    n/=2;       //n/2
    cnt++;     //位数+1
}
for(i=cnt-1;i>=1;i--)//逆序输出
 cout<<a[i];
```


信息思维系列课程

# 数组之一维数组综合应用

2021.4.24

# 例一、去重(chóng)

依次输入n个数，将其去重后依次输出。

输入
第一行一个数n（n<100）
第二行n个int范围以内的数，之间以空格隔开

输出
输出去重后的数

样例输入
10
1 1 2 2 3 9 3 9 4 1
样例输出
1 2 3 9 4

思考分析

# 分析

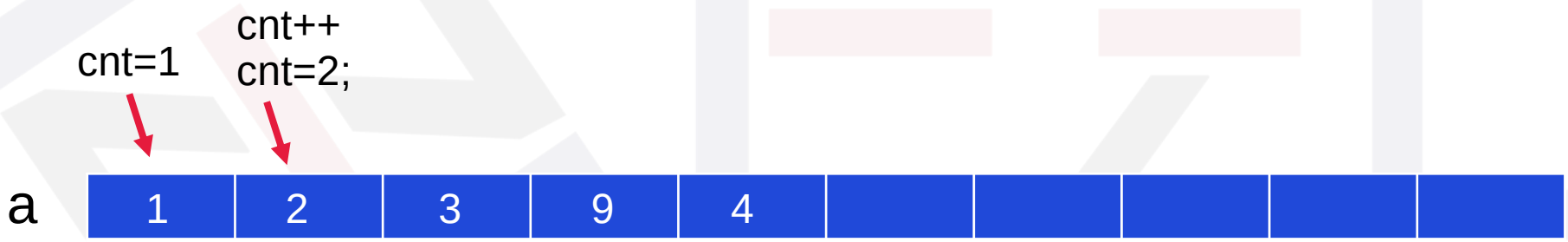
去重前：


去重后：

# 分析

方法　如果数字没有出现在去重数组中，那么就把它放入数组

1 1 2 2 3 9 3 9 4 1




也需要设立一个“游动”的下标cnt，实时更新放入数组的数字个数

所做的事情　如果x没有出现过

```
a[cnt]=x;
cnt++;
```

# 核心代码

```
int x,cnt=1,flag;
for(i=1;i<=n;i++){   //输入n个数字
    cin>>x;
    flag=1;
    for(j=1;j<=cnt;j++){   //遍历a数组
       if(a[j]==x) flag=0;
    }
     if(flag==1)  {
       a[cnt]=x;
       cnt++;
    }
}
```

# 例二、幸运数字划分

判断一个正整数n是否能被一个幸运数整除，幸运数是指一个只包含4或者7的正整数，如7,47,477等都是幸运数字，12、42则不是幸运数字。

输入
一行一个正整数n，1<=n<=1000
输出
一行一个字符串，如果能被幸运数字整除输出“YES"否则输出”NO“

样例
输入复制
47
输出复制
YES

思考分析

# 分析

**算法一**

不断拆分数字

如果有一个数字不是4和7，那么就不是幸运数字　flag=0

拆分完成都没有发现其他数字，那么就是幸运数字　flag=1

存储幸运数

查找

**标记变量**来标记是否为幸运数字

# 分析

**算法二**

题目的数据范围：1<=n<=1000

数据范围很小，其实我们自己也可以找出这样的一些数字

```
int a[14]={4,7,44,47,74,77,444,447,474,477,744,747,774,777}
```

接下来的事情就是去**查找这个数是否能被数组**里的数整除

## 这道题的启示：

- 根据**数据范围**的大小，可以采取不同算法
- 人为的在程序里计算出一些已知数据的方法，称为“**打表**”

# 核心代码

```
int a[14]={4,7,44,47,74,77,444,447,474,477,744,747,774,777}

flag=0;
for(i=0;i<14;i++){
    if(a[i]%x==0){
        flag=1;
    }

}
if(flag==1) cout<<“YES”;
else cout<<“NO”;
```

# 例三、筛选法求素数(埃氏筛法)

筛选法求素数是一种高效求素数的方法，其具体算法如下：
从2开始把连续的整数放入筛中，首先确定筛中第一个数2是素数。并从筛中晒去所有2的倍数（不包括2）；然后从2以后开始查找，找到筛中剩下的第一个数，它也是素数，并从筛中筛去它的所有倍数（不包括本身）。如此反复执行，直到无数可筛为止。这时筛中剩下的就是这一串就是素数。
提示：
可以用一个数组ss作为筛，数组的下标代表连续的整数，若储存的是0时，代表不在筛中，储存的是1时，代表在筛中。如ss[2]=1代表整数2在筛中。

输入
一行n（n<300000)

输出
2到n的所有素数

样例
输入
12

输出
2 3 5 7 11

# 分析

## 方法一、简单筛法(试除法)

1.枚举2~n之间的数字

2.判断是否为质数 → 判断该数是否含有除1和它本身以外的数

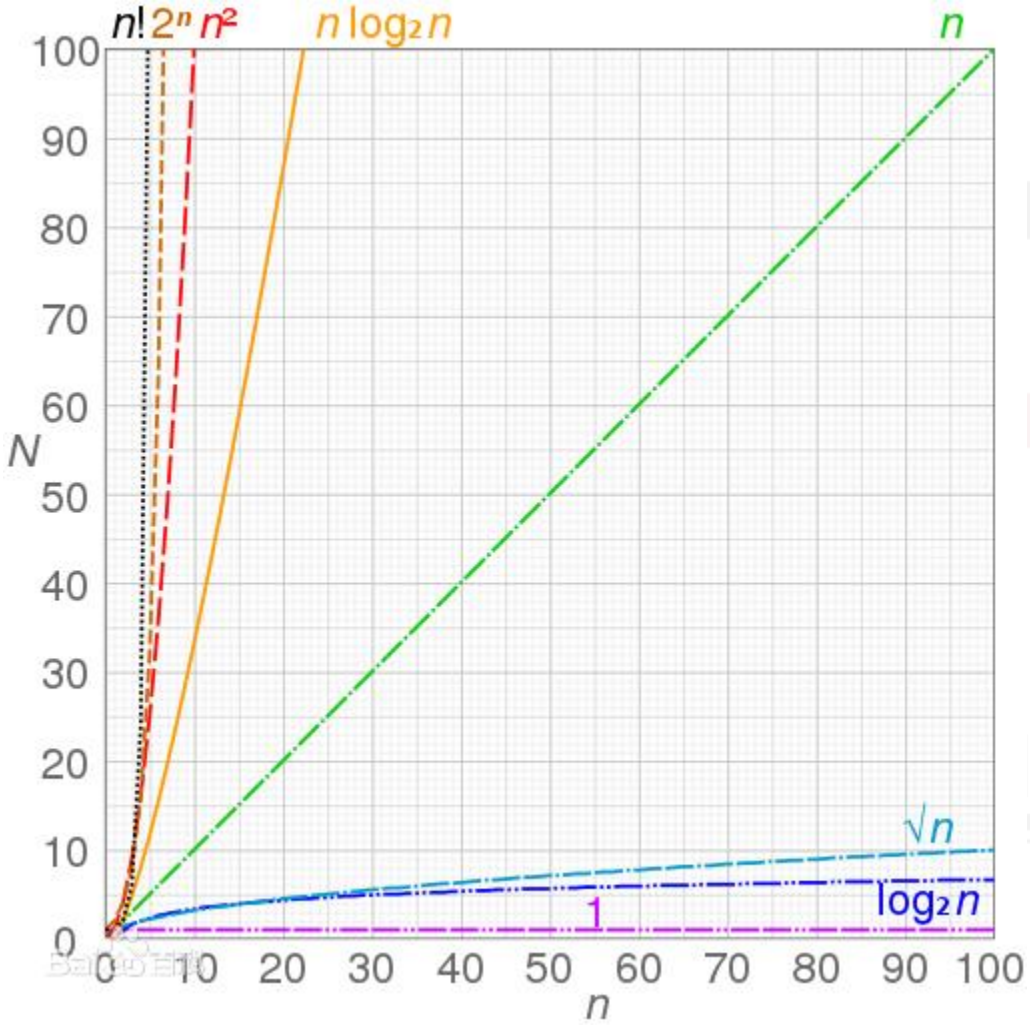
3.输出质数

# 分析

```
int n,i,j,flag;//flag为标记变量，0代表是素数，1代表是合数
    cin>>n;
    //简单筛法
    for(i=2;i<=n;i++){ //枚举2~n的数
       flag=0;
        for (j = 2; j*j <= i;j++){ //枚举因子2~sqrt(n)
          if (i%j == 0&&i!=j){  //如果为合数
                flag=1;
                break;
          }
        }
         if(flag==1) cout<<i<<" "; //是素数就输出
    }
```

sqrt函数运行效率不高
乘法更快

# 时间复杂度




含义：描述该算法的运行时间,时间复杂度常用大O符号表述，不包括这个函数的低阶项和首项系数

O( n )

n是处理的数据规模

一般我们认为，计算机一秒可以运行一亿次($10^9$)

小练习

| | |
|---|---|
| $n^3+n^2+1$ | $O(n^3)$ |
| 3*n+1 | O(n) |
| 123 | O(1) |

# 时间复杂度

```
//简单筛法
for(i=2;i<=n;i++){ //枚举2~n的数

    for (j = 2; j*j <= i;j++){ //枚举因子2~sqrt(n)


    }

}
```



**简单筛法的时间复杂度为$O(n^2)$**

**题目数据范围：n<** 300000　　300000* 300000

# 分析

## 方法二、筛选法(埃氏筛法)

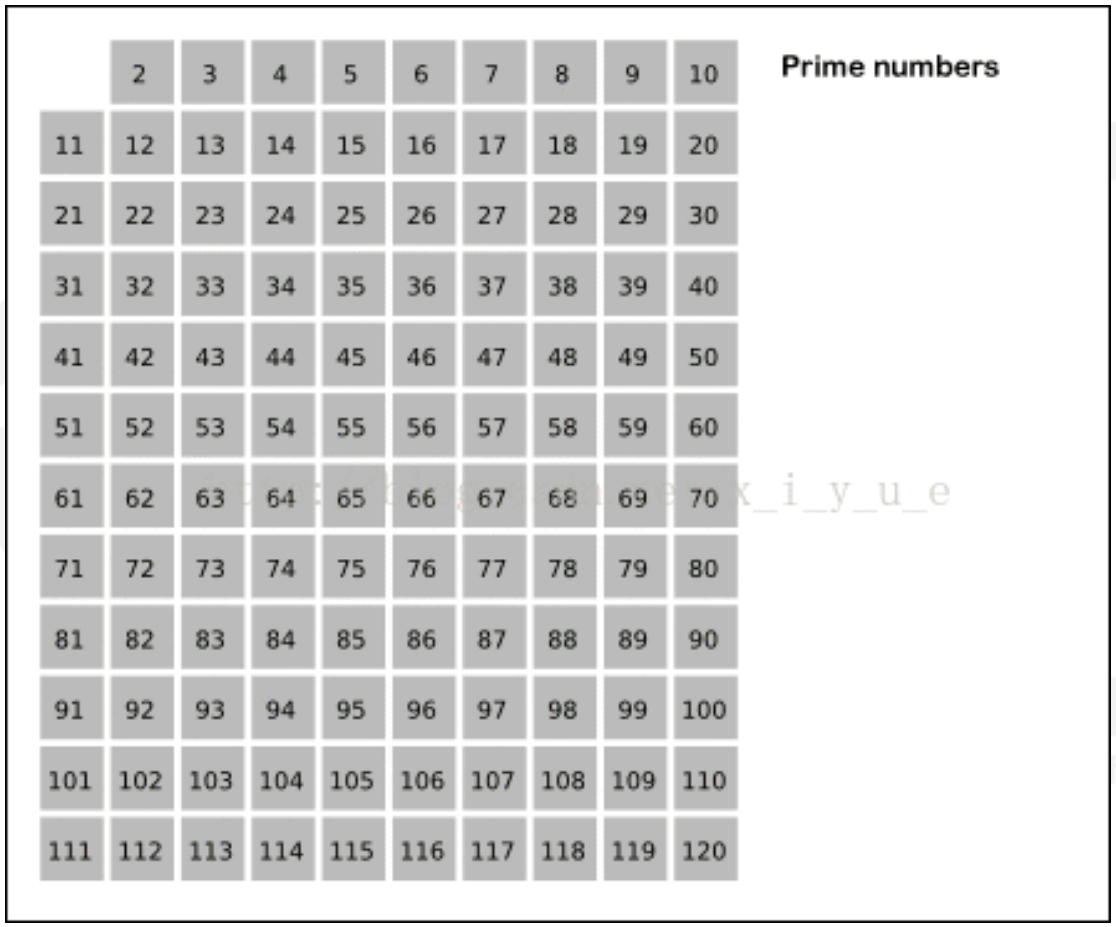
基本思想：素数的倍数一定不是素数

原理：该素数其实也就是后面合数的因子

**筛出20以内的数：**

| 步骤 | 操作 | 素数筛 |
| --- | --- | --- |
| 1 | 无 | 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 |
| 2 | 筛去2的倍数 | 2 3 5 7 9 11 13 15 17 19 |
| 3 | 筛去3的倍数 | 2 3 5 7 11 13 17 19 |
| 4 | 筛选完毕 | 2 3 5 7 11 13 17 19 |

# 动画演示




**时间复杂度O(nloglogn)**

# 分析

```
int i,j,n;
int prime[300500];//标记数组
cin>>n;
for(i=0;i<=n;i++)
      prime[i]=1; //预处理标记数组，默认0~n的数都是素数
prime[0]=0;
prime[1]=0; //0和1不是素数
//埃氏筛法
for(i=2;i*i<=n;i++)    //枚举素数(因子)2~sqrt(n)
 if(prime[i]==1){          // 该数为素数
   for(j=2;i*j<=n;j++) //筛掉它的倍数,j代表倍数
     prime[i*j]=0;
}
for(i=2;i<=n;i++)
 if(prime[i]==1)
     cout<<i<<" ";
```

# 总结

- 数组可以保存数据信息，便于后续的处理
- 将一些已知的答案存储在数组便于后续处理，称为打表
- 当需要标记一组状态的情况时，可以用到标记数组
- 随着学习的深入，大家也要逐渐重视程序的时间复杂度，学会估算时间复杂度

# Thanks

## For Your Watching