

函数



课程安排

- 1、函数的概念
- 2、定义函数
- 3、调用函数
- 4、编程题目：1507大小质数
- 5、编程题目：2091幸运数字

们

数学方程式 $y=2x-1$

$y=2x-1$ 本身就是一个函数，y的值随x的变化而变化，可以记录成： $f(x)=2x-1$ 。 $f(x)$ 表示算式中的x为变量。

$$f(x)=4x-m+1$$

请问： $m=3$ 时， x 输入的值是1, 2, 3； $f(x)$ 的输出分别是？

$$f(1)=$$

$$f(2)=$$

$$f(3)=$$

总结：函数中变量可以是多个，上题中的m,x 如果同为变量。

C++中的函数

```
#include <iostream>
using namespace std;
int main() {
    float a;
    cin>>a;
    cout<<"My height is
    "<<a<<"cm.";
    return 0;
}
```

这个程序中main就是主函数。

定义函数

例如：写一个正方形面积的
函数。输入边长，返回面积

定义函数的基本格式如下：
返回值类型 函数名(传入的参数)

```
{  
    // 要执行的代码  
    return 返回值; // 必须要有返回值  
}
```

例如：写一个正方形面积的函数。
输入边长，返回面积

```
int area(int n)  
{  
    // 要执行的代码  
    return n * n; // 必须要有返回值  
}
```

定义函数

		描述
函数名	area	
返回值类型	int	表示计算结果为整数
返回值	$n \times n$	返回 $n \times n$ ，即正方形面积
传入的参数	n	正方形边长

函数的返回值

按照返回值，可以将函数分成两类：

- 1、没有返回值的函数
- 2、有返回值的函数。



无返回值函数

```
void printHello(int n){  
    for(int i = 0; i < n; i++)  
        cout << "Hello  
World" << endl; //输出  
}
```

没有返回值的函数被void数，其通用格式如下：

```
void 函数名(传入的参数)  
{  
    //要执行的代码  
    return;      // 可选  
}
```

注意：括号中传入的参数指函数的参数类型和数量

有返回值函数

```
void printHello(int n){  
    for(int i = 0; i < n; i++)  
        cout << "Hello  
World" << endl; //输出  
}
```

函数的类型被声明为返回值的类型，其通用格式如下：
返回值类型 函数名(传入的参数)
)
{
 // 要执行的代码
 return 返回值; // 必
 需要有返回值
}

有返回值函数 注意事项

- 1、必须使用return语句，以便将值返回给调用函数
 - 2、值可以是量、变量，也可以是表达式，结果的类型必须为返回值类型或被转换为返回值类型
 - 3、声明的返回类型为double，而函数返回一个int表达式，则该int值将被强制转换为double类型
- 注意：返回值可以使任何类型，但不能是数组。

有返回值函数 例题

定义一个函数，计算从1加到100的结果：

```
int sum() {  
    int i, sum = 0;  
    for (i = 1; i <= 100; i++) {  
        sum += i;  
    }  
    return sum; //返回值  
}
```

结果保存在变量sum中，最后通过return语句返回。

sum是int型，返回值也是int类型，一一对应。

无参数的函数

计算从1加到100的结果。

```
#include <bits/stdc++.h>
using namespace std;
int sum(){
    int i, sum=0;
    for(i=1; i <= 100; i++){
        sum+=i;
    }
    return sum; //返回值
}
int main() { // 主函数
    cout << sum(); // 调用函数
}
```

上面就是一个没有参数的写法

运行结果：

5050

有参数的函数

有参数的函数写法如下：

返回值类型 函数名(参数1的类型
参数1的名称,参数2的类型 参数2
的名称,...参数n的类型 参数n的名
称)

{

// 要执行的代码

return 返回值; // 必须要有返回值

}

注意：多个参数之间用逗号分
隔

有参数的函数例题

下面看下计算矩形面积的程序：

```
#include <bits/stdc++.h>
using namespace std;
int area(int width, int height)
{
    return width * height; // 必须要有返回值
}
int main() {
    int a, b;
    cin >> a >> b;
    int c = area(a, b); // 调用函数
    cout << c;
}
```

传入的参数有两个：width和height，返回两个参数相乘的结果。

输入：10 12

输出：120

```
#include <bits/stdc++.h>
using namespace std;
void printRepeat(int n, string s) {
    for (int i = 0; i < n; i++)
        cout << s << endl;
}
int main() { // 主函数
    printRepeat(5, "ok"); // 调用函
    // 数
}
```

有参数的函数例题

运行结果：

ok

ok

ok

ok

ok

调用函数

函数也需要先定义再调用

例如：

- main函数：如果main函数调用某个名为sum的函数，要把sum放在main前面

```
#include <bits/stdc++.h>
using namespace std;
int sum() {
    int i, sum = 0;
    for (i = 1; i <= 100; i++) {
        sum += i;
    }
    return sum;
}
int main() {
    int a = sum();
    cout << "The sum is " << a << endl;
    return 0;
}
```

1507 大小质数

给出一个数n，求出比n大的第一个质数以及比n小的第一个质数。

例如：n = 5，输出7 3，比5大的第一个质数是7，比5小的第一个质数是3。

输入

一个数n ($2 < n < 1000000000$)。

输出

输出2个数，中间用空格分割，分别对应比n大的第一个质数以及比n小的第一个质数。

5

7 3

1507大小质数（解题思路）

定义一个函数，输入参数是一个整数 int，返回值是一个 bool。函数用来判断输入的数是否为质数，是质数返回 true，否则返回 false

在 main 函数里：

1. 从 n 向上枚举，调用 isprime 判断每个数是否为质数，是则输出该数。
2. 从 n 向下枚举，调用 isprime 判断每个数是否为质数，是则输出该数。



大小质数 参考答案

```
#include <bits/stdc++.h>
using namespace std;
bool is_prime(int val){
    int m = sqrt(val + 0.5);
    for(int i = 2; i <= m; i++){
        if(val % i == 0){
            return false;
        }
    }
    return true;
}
int main()
{
    int n;
    cin>>n;
    for(int j = n + 1; ; j++){
        if(is_prime(j)){
            cout<<j<<" ";
            break;
        }
    }
    for(int j = n - 1; j >= 2; j--){
        if(is_prime(j)){
            cout<<j<<endl;
            break;
        }
    }
    return 0;
}
```

2091 幸运数字们

如果一个数字的十进制表示中有7，我们就认为他是幸运数字。

输入两个整数L, R($1 \leq L \leq R \leq 100000$)

输出所有满足 $L \leq n \leq R$ 的幸运数字n。

从小到大输出所有幸运数字，一行一个。

特别注意如果区间内没有任何一个幸运数字，输出None

输入

一行两个整数L, R，其中 $1 \leq L \leq R \leq 100000$ 。

输出

从小到大输出所有幸运数字，一行一个。

特别注意如果区间内没有任何一个幸运数字，输出None

88 111

97
107

2091幸运数字们 解题思路

定义一个返回值为 bool 的 judge 函数，判断输入的 x 是否为幸运数。如果是则返回 true，否则返回 false

- 从 L 到 R 枚举所有数字，如果是幸运数字，则输出，同时做计数。

- 如果一个幸运数字都没有，则输出 None



幸运数字们 参考答案

```
#include <bits/stdc++.h>
using namespace std;
bool judge(int x) {
    while (x > 0) {
        if (x % 10 == 7) {
            return true;
        }
        x /= 10;
    }
    return false;
}
int main() {
    int L, R, cnt = 0;
```

```
scanf("%d%d", &L, &R);
for (int i = L; i <= R; i++) {
    if (judge(i)) {
        printf("%d\n", i);
        cnt++;
    }
}
if (cnt == 0) {
    printf("None\n");
}
return 0;
```

总结

我们一起学习了函数的知识点。

函数在我们以后的编程中会经常用到，函数可以大大减少我们编写的代码数量。

以大小质数为例，我们只需要定义一个判断质数的函数，之后可以在各个地方使用他，而无需每次都重新写同样的代码。



函数习题课



课程安排

- 1、编程题目：1507大小质数
- 2、编程题目：2091幸运数字们
- 3、编程题目：3320最大质因子序列

1507 大小质数

给出一个数n，求出比n大的第一个质数以及比n小的第一个质数。

例如：n = 5，输出7 3，比5大的第一个质数是7，比5小的第一个质数是3。

输入

一个数n ($2 < n < 1000000000$)。

输出

输出2个数，中间用空格分割，分别对应比n大的第一个质数以及比n小的第一个质数。

5

7 3

1507大小质数（解题思路）

定义一个函数，输入参数是一个整数 int，返回值是一个 bool。函数用来判断输入的数是否为质数，是质数返回 true，否则返回 false

在 main 函数里：

1. 从 n 向上枚举，调用 isprime 判断每个数是否为质数，是则输出该数。
2. 从 n 向下枚举，调用 isprime 判断每个数是否为质数，是则输出该数。



大小质数 (参考答案)

```
#include <bits/stdc++.h>
using namespace std;
bool is_prime(int val){
    int m = sqrt(val + 0.5);
    for(int i = 2; i <= m; i++){
        if(val % i == 0){
            return false;
        }
    }
    return true;
}
int main()
{
    int n;
    cin>>n;
    for(int j = n + 1; ; j++){
        if(is_prime(j)){
            cout<<j<<" ";
            break;
        }
    }
    for(int j = n - 1; j >= 2; j--){
        if(is_prime(j)){
            cout<<j<<endl;
            break;
        }
    }
    return 0;
}
```

2091 幸运数字们

如果一个数字的十进制表示中有7，我们就认为他是幸运数字。

输入两个整数L, R($1 \leq L \leq R \leq 100000$)

输出所有满足 $L \leq n \leq R$ 的幸运数字n。

从小到大输出所有幸运数字，一行一个。

特别注意如果区间内没有任何一个幸运数字，输出None

输入

一行两个整数L, R，其中 $1 \leq L \leq R \leq 100000$ 。

输出

从小到大输出所有幸运数字，一行一个。

特别注意如果区间内没有任何一个幸运数字，输出None

88 111

97
107

2091幸运数字们（解题思路）

定义一个返回值为 bool 的 judge 函数，判断输入的 x 是否为幸运数。如果是则返回 true，否则返回 false

- 从 L 到 R 枚举所有数字，如果是幸运数字，则输出，同时做计数。

- 如果一个幸运数字都没有，则输出 None



幸运数字们 (参考答案)

```
#include <bits/stdc++.h>
using namespace std;
bool judge(int x) {
    while (x > 0) {
        if (x % 10 == 7) {
            return true;
        }
        x /= 10;
    }
    return false;
}
int main() {
    int L, R, cnt = 0;
    scanf("%d%d", &L, &R);
    for (int i = L; i <= R; i++) {
        if (judge(i)) {
            printf("%d\n", i);
            cnt++;
        }
    }
    if (cnt == 0) {
        printf("None\n");
    }
    return 0;
}
```

3320 最大质因子序列

任意输入两个正整数m, n ($1 < m < n \leq 10000$)，依次输出m到n之间每个数的最大质因子（包括m和n；如果某个数本身是质数，则输出这个数自身）。

输入

- 一行，包含两个正整数m和n，其间以单个空格间隔。

输出

一行，每个整数的最大质因子，以空格间隔。

5 10

5 3 7 2 3 5

3320 最大质因子序列（解题思路）

定义一个函数来求 n 的最大因子，因子试探到 \sqrt{n} 即可。



3320 最大质因子序列（参考答案）

```
#include <bits/stdc++.h>
using namespace std;
int maxFac(int n) {
    int v = 0;
    for(int i = 2; i * i <= n; i++) {
        while(n % i == 0) {
            n /= i;
            v = i;
        }
    }
    return max(v, n);
}
int main() {
    int m, n;
    cin >> m >> n;
    for(int i = m; i <= n; i++)
        cout << maxFac(i) << " ";
    return 0;
}
```

函数传参



课程安排

- 1、编程题目：函数与变量作用域
- 2、参数的传递：值传递、引用传递
、指针传递
- 3、函数实例
- 4、函数命名冲突
- 5、编程题目：2639因数之和为n的
最小正数
- 6、编程题目：3410精妙数

函数与变量作用域

首先回顾下变量的知识：

- 1、局部变量：在函数或一个代码块内部声明的变量
- 2、形式参数：在函数参数的定义中声明的变量
- 3、全局变量：在所有函数外部声明的变量

注意：

- 1、参数中声明的变量，只在当前函数内部
- 2、如果函数又调用了其他函数，被调用的函数是不能使用这个变量
- 3、全局变量是所有函数都可以访问

函数与变量作用域

例题：利用全局变量求矩形
面积：

```
#include <bits/stdc++.h>
using namespace std;
int width, height, ans;
void area() {
    ans = width * height;
}
int main() {
    cin >> width >> height;
    area();
    cout << ans;
}
```

函数的传递-值传递

- 1、C++通常按值传递参数，将数值参数传递给函数，而后者将其赋给一个新的变量。
- 2、形式参数：函数定义时给出的参数
- 3、实际参数：函数调用时给出的参数（也就是传递的数据）



函数的传递-值传递

```
#include <bits/stdc++.h>
using namespace std;
void Add(int n)
{
    n += 10;
    cout << n << endl;
}
int main()
{
    int n = 5;
    Add(n);
    cout << n << endl;
    return 0;
}
```

输出结果：

15

5

函数的传递-引用传递

C++提供了一种引用参数：作用是将形参看成是实参的别名，二者指向的是同一份内存，在对函数中的引用参数进行改动时，实际上就是对实参本身进行修改

- 引用的符号是&，下面通过程序直观的感受一下引用的作用：



函数的传递-引用传递

```
#include <bits/stdc++.h>
using namespace std;
void swap1(int x, int y){ // 普通的形参
    int tmp = x;
    x = y;
    y = tmp;
}
void swap2(int& x, int& y)
{ // 引用参数
    int tmp = x;
    x = y;
    y = tmp;
}
```

```
int main()
{
    int a = 3, b = 5;
    swap1(a, b);
    cout << a << ' ' << b << endl;
    int c = 4, d = 6;
    swap2(c, d);
    cout << c << ' ' << d << endl;
    return 0;
}
```

输出结果：

3 5
6 4

函数的传递-引用传递

a b 在执行完swap1函数之后并没有发生交换，这是因为 swap1中交换的实际上是swap1 函数中x和y，而x和y的值则是a和b复制而来的。但是c和d却发生了交换，这是因为 swap2函数中使用的是引用参数，那么swap2中的x实际上就是c的别名，y 实际上是d的别名，在对x和y进行操作的时候，实际上就是对c和d本身进行操作。

函数的传递-引用 传递

```
#include <bits/stdc++.h>
using namespace std;
int area(int width, int height, int& girth)
{
    //计算周长
    girth = (width + height) * 2;
    return width * height; // 必须要有返回
    值
}
int main() {
    int a, b, s;
    cin >> a >> b;
    int c = area(a, b, s);
    cout << c << " " << s;
}
```

函数实例

组合数的公式 $C(m, n) = \frac{n!}{m!(n-m)!}$ ，其中 $m \leq n \leq 10$

```
long long factorial (int n){  
    long long m=1;  
    for (int i=1;i<=n;i++)  
        m*=i;  
    return m;  
}  
  
long long C(int n, int m)  
{  
    return  
factorial(n)/(factorial(m)*factorial(n-  
m));  
}
```

函数实例

算圆面积的函数：

```
double CircleArea(int r)
{
    double ans = 3.1415926535 * r *
r;
    return ans;
}
```

函数实例

质能方程：

```
long long E(int m)
{
    return m * 300000 * 300000;
}
```

函数实例

等差数列求和：

```
long long Gauss(int start, int end,  
                int n)  
{  
    return (start + end) * n / 2;  
}
```

函数命名冲突

函数名不能同全局变量的变量名相同。以下代码会出现编译错误。

```
#include <bits/stdc++.h>
using namespace std;
int area;
int area() {
    return 5;
}
int main() {
    cout << area;
}
```

2639 因数之和为n的最小正数

一个自然数的因数是指能被这个自然数整除的所有自然数。例如6的因数为：1,2,3,6。

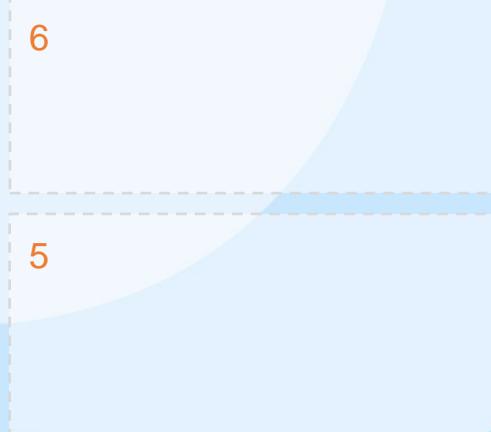
现在给出一个数n，求因数之和为n的最小的正数是多少（如果找不到这样的数，输出-1）。

输入

一个数 n ($1 \leq n \leq 10000$)

输出

一个数 a



2639因数之和为n的最小正数

写一个求约数和的函数，从1开始向上枚举，知道约数和等于n为止。n的约数和一定 $>n$ ，因为至少拥有1, n这2个约数，因此枚举的终止条件就是 $i < n$ ，如果没有枚举到任何符合条件的数，则输出-1。



2639因数之和为n的最小正数 参考答案

```
#include <bits/stdc++.h>
using namespace std;
int ysh(int a) {
    int s = 0;
    for (int i = 1; i * i <= a; i++) {
        if (a % i == 0) {
            s += i;
            if (i * i != a)
                s += a / i;
        }
    }
    return s;
}
```

```
int main() {
    int n;
    cin >> n;
    for (int i = 1; i < n; i++) {
        if (ysh(i) == n) {
            cout << i;
            return 0;
        }
    }
    cout << "-1";
    return 0;
}
```

3410 精妙数

小明正在研究趣味数学。如果一个正整数，它的二进制形式是个回文数，小明就会将它称为精妙数。如33，二进制形式为10001，则33是一个精妙数。

现在小明找到了T个数字，他想让你判断这些数是不是精妙数。是精妙数回答Yes，否则回答No。

输入

- 第一行输入一个数T，表示数字个数；
之后T行，每行输入一个数，表示每个数字。

输出

输出T行，每行一个字符串"Yes"或者"No"，以空格隔开。

3
44
33
22

No
Yes
No

3410 精妙数 解题思路

如何判断一个数是否为精妙数呢？我们先求出他的 2 进制表示，并保存至数组，再判断其是否为回文。

为了方便编写代码，我们定义一个返回值为 bool 的函数，如果是回文则返回 true，否则返回 false

- 判断回文需要利用循环遍历整个数组，假如这个数二进制表示的长度为 k，我们逐个比较 $(0, k-1), (1, k-2) \dots$ 每个对应的位置是否相等，如果不等则返回 false。如果比到最后仍然没有出现 false，则返回 true

3410 精妙数 参考答案

```
#include <bits/stdc++.h>
using namespace std;
int a[35];
bool j_exquisite(int num) {
    int cnt = 0;
    while (num > 0) {
        a[cnt] = num % 2;
        num /= 2;
        cnt++;
    }
    for (int i = 0; i < cnt; i++)
        if (a[i] != a[cnt - 1 - i])
            return false;
    return true;
}
int main() {
    int T, n;
    cin >> T;
    while (T--) {
        cin >> n;
        if (j_exquisite(n))
            cout << "Yes" << endl;
        else
            cout << "No" << endl;
    }
    return 0;
}
```

函数传参练习题课



课程安排

- 1、编程题目：2639因数之和为n的最小正数
- 2、编程题目：3410精妙数
- 3、编程题目：3051找数字

V2

2639 因数之和为n的最小正数

一个自然数的因数是指能被这个自然数整除的所有自然数。例如6的因数为：1,2,3,6。

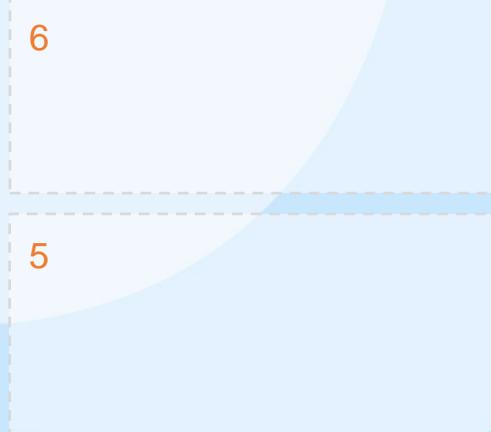
现在给出一个数n，求因数之和为n的最小的正数是多少（如果找不到这样的数，输出-1）。

输入

一个数 n ($1 \leq n \leq 10000$)

输出

一个数 a



2639因数之和为n的最小正数

解题思路

写一个求约数和的函数，从1开始向上枚举，知道约数和等于n为止。n的约数和一定 $>n$ ，因为至少拥有1, n这2个约数，因此枚举的终止条件就是 $i < n$ ，如果没有枚举到任何符合条件的数，则输出-1。



2639因数之和为n的最小正数 参考答案

```
#include <bits/stdc++.h>
using namespace std;
int ysh(int a) {
    int s = 0;
    for (int i = 1; i * i <= a; i++) {
        if (a % i == 0) {
            s += i;
            if (i * i != a)
                s += a / i;
        }
    }
    return s;
}
```

```
int main() {
    int n;
    cin >> n;
    for (int i = 1; i < n; i++) {
        if (ysh(i) == n) {
            cout << i;
            return 0;
        }
    }
    cout << "-1";
    return 0;
}
```

3410 精妙数

小明正在研究趣味数学。如果一个正整数，它的二进制形式是个回文数，小明就会将它称为精妙数。如33，二进制形式为10001，则33是一个精妙数。

现在小明找到了T个数字，他想让你判断这些数是不是精妙数。是精妙数回答Yes，否则回答No。

输入

- 第一行输入一个数T，表示数字个数；
之后T行，每行输入一个数，表示每个数字。

输出

输出T行，每行一个字符串"Yes"或者"No"，以空格隔开。

3
44
33
22

No
Yes
No

3410 精妙数 解题思路

如何判断一个数是否为精妙数呢？我们先求出他的 2 进制表示，并保存至数组，再判断其是否为回文。

为了方便编写代码，我们定义一个返回值为 bool 的函数，如果是回文则返回 true，否则返回 false

- 判断回文需要利用循环遍历整个数组，假如这个数二进制表示的长度为 k，我们逐个比较 $(0, k-1), (1, k-2) \dots$ 每个对应的位置是否相等，如果不等则返回 false。如果比到最后仍然没有出现 false，则返回 true

3410 精妙数 参考答案

```
#include <bits/stdc++.h>
using namespace std;
int a[35];
bool j_exquisite(int num) {
    int cnt = 0;
    while (num > 0) {
        a[cnt] = num % 2;
        num /= 2;
        cnt++;
    }
    for (int i = 0; i < cnt; i++)
        if (a[i] != a[cnt - 1 - i])
            return false;
    return true;
}
int main() {
    int T, n;
    cin >> T;
    while (T--) {
        cin >> n;
        if (j_exquisite(n))
            cout << "Yes" << endl;
        else
            cout << "No" << endl;
    }
    return 0;
}
```

3051 找数字 V2

将大于0的整数写成一排组成一个无限长的数，即：

12345678910111213141516...

问这个数的第n位的数字是什么？

输入

- 第一行：1个数T表示询问的数量（ $2 \leq T \leq 100000$ ）。
- 第2至n+1行：每行1个数，对应询问的 n。（ $1 \leq n \leq 10^9$ ）

输出

输出共t行，对应t次询问的答案。

3
3
10
21

3
1
5

3051找数字V2（解题思路）

如果想知道第n位是什么，需要先知道第n位对应哪个数字，以及是这个数字的第几位。

例如：第18位对应数字13的第2位，因此是3。

第14位对应11的第2位，所以是1。

求第n位对应的数字，需要先知道第n位对应的数字是几位数。

例如：第18位对应数字13，是个2位数。

所有1位数长度之和为9。

所有2位数长度之和为180。

所有3位数长度之和为2700。

.....

所有k位数长度之和为 $9k \times 10^{k-1}$

3051找数字V2（解题思路）

因此，如果 $n \leq 9$ ， n 对应1位数。如果 n 在[10,189]之间，则 n 是2位数，在[190,2890]之间，则对应3位数。以此类推。

知道了是几位数，我们可以算出第 n 位对应的整数 num 是哪一个，即 n 减去 $1 \rightarrow n-1$ 位数长度之和，设这个结果为 l ，则利用 $(l-1)/k + 1$ 就可以知道 num 是 k 位数中的第几个，从而知道 num 是多少。

举例来说，如果 $n=200$ ，根据上面的计算，我们可以知道 n 对应3位数， $200-189=11$ ，那么就是说对应3位数排列的第11位。而3位每个数字长度都是1，第11位对应第4个3位数的第2位，即103的第2位，因此是0。

3051找数字V2（解题思路）

由于需要处理的细节较多，所以专门定义一个函数getDig，来求解序列的第n位。在这个函数中：

第一步：要确定第n位对应的是几位数。

第二步：确定对应的数字是多少。

- 第三步：利用 $(l-1) \% k$ 的余数确定是数字的第几位。

为了方便编写代码，我们提前预处理好所有10的幂。



3051找数字V2

参考答案

```
#include <bits/stdc++.h>
using namespace std;
typedef long long ll;
int p10[] = {1, 10, 100, 1000, 10000, 100000,
1000000, 10000000, 100000000,
1000000000};
int getDig(int n)
{
    for (int i = 0; i < 10; i++)
    {
        ll len = 9LL * (i + 1) * p10[i];
        if (n <= len)
        {
            int num = (n - 1) / (i + 1) + p10[i];
            int s = (n - 1) % (i + 1);
            return (num / p10[i - s]) % 10;
        }
        n -= len;
    }
}
```

3051找数字V2

参考答案

```
int main()
{
    int t, v;
    cin >> t;
    for (int i = 0; i < t; i++)
    {
        cin >> v;
        cout << getDig(v) << endl;
    }
    return 0;
}
```

字符串



课程安排

- 1、字符串概述
- 2、char数组
- 3、string类型
- 4、编程题目：2148字符出现位置
- 5、编程题目：2149字符串出现位置
- 6、编程题目：2150字符替换

字符串

字符串：存储在内存中的连续字节的一系列字符。其本质相当于一个字符数组。字符串在存储上类似字符数组，所以它每一位的单个元素都是可以提取的。



```
#include <bits/stdc++.h>
using namespace std;
void printRepeat(int n, string s) {
    for (int i = 0; i < n; i++)
        cout << s << endl;
}
int main() { // 主函数
    printRepeat(5, "ok"); // 调用函
    // 数
}
```

字符串例题

注意：传入的 “ok” 就是一个字符串

输出结果：

ok

ok

ok

ok

ok

C++字符串表现形式

1、C 风格字符串；

例如：char数组

2、C++引入的 string类型；

例如：string类型



char数组

char数组是C风格的字符串，使用char[]数组存储

例如：

```
char t[5] = {'f', 'i', 'n', 'e', '\0'};
```



char数组定义与赋值

C++语言中的字符串有一个特殊的性质，以空字符结尾，空字符被写作：“\0”

例如：

```
char s[5] = {'h', 'e', 'l', 'l', 'o'};
```

//这不是一个字符串，只是一个普通的字符数组

```
char t[5] = {'f', 'i', 'n', 'e', '\0'};
```

//这是一个字符数组，这也是一个字符串；因为有‘\0’作为字符串的结束标志



单选题

下列选项哪个是字符串()

- A char a[4]={1,2,3,4};
- B char s[6]= {'h', 'e', 'l', 'l', 'o' , 'e', 'l'}
};
- C char s[5] = {'f', 'i', 'n', 'e', '\0'};
- D char s[6] = {1,2,57,@,#,%};

定义与赋值

例如：char t[5] = {'f', 'i', 'n', 'e', '\0'};

上面字符数组初始化比较麻烦，每个字符都需要一对单引号，而且还能忘记'\0'

为了解决这个问题，C++语言提供了一种简便的方式，使用双引号

- 例如：

char dog[8] = "wangcai";

char cat[] = "maomi";



定义与赋值注意事项

关于 : char dog[8] = "wangcai";

注意 :

1、使用时要保证数组足够大，能够存储字符串中的全部字符包括空字符；上面的[8]包含结束符:\0

2、使用字符串的时候，注意区分字符串常量和字符常量

例如 :

"a" != 'a'

"a"表示的是'a'和 '\0'

'a'表示的就是一个字符 a；所以，两个不可以直接画等号

字符数组的读入可以使用`cin>>`，
输出可以使用`cout<<`。

例如：

```
#include <bits/stdc++.h>
using namespace std;
void printRepeat(int n, char s[]) {
    for (int i = 0; i < n; i++)
        cout << s << endl;
}
int main() {
    char s[100];
    cin >> s;
    printRepeat(5, s);
}
```

char数组输入输出

注意：在定义s时，一定要给出一个大小（`s[100]`），否则会报错

输入：

11

输出

11

11

11

11

11

输入输出

在C++语言中，可以使用**strlen()**函数获得字符串的长度；注意：
不包括结束标识符

例如：

```
#include <bits/stdc++.h>
using namespace std;
int main() {
    char s[100];
    cin >> s;
    cout << strlen(s);
}
```

输入：

Hello

输出

5

输入输出

- 1、使用cin输入字符数组时，会默认在结尾添加'\0'。
- 2、strlen()函数从某个位置（可以是字符串开头，中间某个位置，甚至是某个不确定的内存区域）开始扫描，直到碰到第一个字符串结束符'\0'，然后返回计数器值(不包含'\0')。

```
#include <bits/stdc++.h>
using namespace std;
int main() {
    char s[100];
    cin >> s;
    cout << strlen(s);
}
```

string类型定义与赋值

string 类型是C++支持的字符串类型

char数组：定义和赋值	string类型：定义和赋值
char dog[8] = "wangcai" ;	string dog = "wangcai" ;
char cat[] = "maomi" ;	string cat = "maomi" ;

```
#include <bits/stdc++.h>
using namespace std;
int main ()
{
    string str1 = "Hello";
    string str2 = "World";
    string str3 = str1 + str2;
    cout << "str3 : " << str3 <<
endl;
    return 0;
}
```

string类型定义与赋值

输出结果：

str3 : HelloWorld

string支持加法，两个字符串相加，得到的结果是两个字符串拼在一起

string的读入可以使用**cin>>** , 输出可以使用**cout<<**

```
#include <bits/stdc++.h>
using namespace std;
void printRepeat(int n, string s) {
    for (int i = 0; i < n; i++)
        cout << s << endl;
}
int main() {
    string s;
    cin >> s;
    printRepeat(5, s);
}
```

string类型输入输出

注意：与字符数组不同，**string**类型的s 在声明阶段不需要指定大小

输入：

as

输出：

as

as

as

as

as

string类型 输入输出

```
#include <bits/stdc++.h>
using namespace std;
int main() {
    string s;
    cin >> s;
    cout << s.size();
}
```

与字符数组的strlen()函数类似，string使用size实现获取字符长度（具体用法如下）

输入：hello

输出：5

string类型 输入输出

```
#include <bits/stdc++.h>
using namespace std;
int main() {
    string s = "hello";
    s[0] = 'a';
    s[1]--;
    cout << s;
}
```

string类型可以用访问数组元素的方式，对每个字符进行赋值

运行结果为：

adllo

2148 字符出现位置

请你帮小瓜找一找某个字符在字符串中第一次出现的位置是多少。

2148 字符出现位置

输入

第一行一个整数n($1 \leq n \leq 10000$)，表示字符串的长度。

第二行一个长度为n的字符串，保证每个字符都是小写字母。

第三行一个字符，表示需要寻找的字符。

输出

一行一个整数，表示字符在字符串中第一次出现的位置（从第0位开始记）。如果字符没有出现过，则输出-1。



2148字符出现位置（解题思路）

读取字符串及字符。利用循环遍历字符数组，判断字符第一次出现的位置。



2148字符出现位 置 参考答案

```
#include <bits/stdc++.h>
using namespace std;
int main()
{
    int n;
    char a[10000], chr;
    cin >> n;
    for(int i = 0; i < n; i++)
        cin >> a[i];
    cin >> chr;
    for(int i = 0; i < n; i++)
    {
        if(a[i] == chr)
        {
            cout << i << endl;
            return 0;
        }
    }
    cout << -1 << endl;
    return 0;
}
```

2149 字符串出现位置

给你两个字符串，一个母串，一个子串，请你找出子串第一次在母串中出现的位置。如果子串没有在母串中出现过，则输出-1。

例如子串ab在母串dceab中第一次出现的位置是3，而子串abc则在dceab中没有出现过。

2149 字符串出现位置

输入

第一行一个字符串（母串），保证每个字符都是小写字母。

第二行一个字符串（子串），保证每个字符都是小写字母。

保证两个字符串的长度都不超过10000，并且大于0。

输出

一行一个整数，表示子串第一次在母串中出现的位置。假如子串没有在母串中出现过，则输出-1。
。

decdagee
age

4

2149字符串出现位置（解题思路）

读取两个字符串。利用双重循环，对两个字符进行比较。外层循环枚举子串在母串出现的位置 i ，内层循环判断在母串位置 i 上的字符是否与子串相同。



2149字符串出现位置 参考答案

```
#include <bits/stdc++.h>
using namespace std;
int main() {
    string str1, str2;
    cin >> str1;
    cin >> str2;
    for (int i = 0; i <= str1.length() - str2.length(); i++) {
        bool equal = true;
        for (int j = i; j < i + str2.length(); j++) {
            if (str1[j] != str2[j-i]) {
                equal = false;
                break;
            }
        }
        if (equal) {
            cout << i << endl;
            return 0;
        }
    }
    cout << -1 << endl;
    return 0;
}
```

2150 字符替换

小瓜有一个包含数字和字母字符的字符串，他不希望别人知道其中的数字是多少，请你帮忙把字符串中的数字字符全部替换成*。

输入

- 输入一个字符串，保证字符串长度不超过10000，并且只包含数字和字母字符。

输出

输出一个字符串，表示将输入字符串中的数字字符全部替换成*之后的字符串。

ad123de1dxb113

ad*de*dxb*

2150字符替换 解题思路

读取输入的字符 str , 遍历 str 并将其中的数字替
换为 *



2150字符替换

参考答案

```
#include <bits/stdc++.h>
using namespace std;
int main() {
    string str;
    cin >> str;
    for (int i = 0; i < str.length(); i++) {
        if (str[i] >= '0' && str[i] <= '9') {
            cout << "*";
        } else {
            cout << str[i];
        }
    }
    return 0;
}
```

字符串习题课



课程安排

- 1、编程题目：2148字符出现位置
- 2、编程题目：2149字符串出现位置
- 3、编程题目：2150字符替换

2148 字符出现位置

请你帮小瓜找一找某个字符在字符串中第一次出现的位置是多少。

2148 字符出现位置

输入

第一行一个整数n($1 \leq n \leq 10000$)，表示字符串的长度。

第二行一个长度为n的字符串，保证每个字符都是小写字母。

第三行一个字符，表示需要寻找的字符。

输出

一行一个整数，表示字符在字符串中第一次出现的位置（从第0位开始记）。如果字符没有出现过，则输出-1。



2148字符出现位置 解题思路

读取字符串及字符。利用循环遍历字符数组，判断字符第一次出现的位置。



2148字符出现位 置 参考答案

```
#include <bits/stdc++.h>
using namespace std;
int main()
{
    int n;
    char a[10000], chr;
    cin >> n;
    for(int i = 0; i < n; i++)
        cin >> a[i];
    cin >> chr;
    for(int i = 0; i < n; i++)
    {
        if(a[i] == chr)
        {
            cout << i << endl;
            return 0;
        }
    }
    cout << -1 << endl;
    return 0;
}
```

2149 字符串出现位置

给你两个字符串，一个母串，一个子串，请你找出子串第一次在母串中出现的位置。如果子串没有在母串中出现过，则输出-1。

例如子串ab在母串dceab中第一次出现的位置是3，而子串abc则在dceab中没有出现过。

2149 字符串出现位置

输入

第一行一个字符串（母串），保证每个字符都是小写字母。

第二行一个字符串（子串），保证每个字符都是小写字母。

保证两个字符串的长度都不超过10000，并且大于0。

输出

一行一个整数，表示子串第一次在母串中出现的位置。假如子串没有在母串中出现过，则输出-1。
。

decdagee
age

4

2149字符串出现位置 解题思路

读取两个字符串。利用双重循环，对两个字符进行比较。外层循环枚举子串在母串出现的位置 i ，内层循环判断在母串位置 i 上的字符是否与子串相同。



2149字符串出现位置 参考答案

```
#include <bits/stdc++.h>
using namespace std;
int main() {
    string str1, str2;
    cin >> str1;
    cin >> str2;
    for (int i = 0; i <= str1.length() - str2.length(); i++) {
        bool equal = true;
        for (int j = i; j < i + str2.length(); j++) {
            if (str1[j] != str2[j-i]) {
                equal = false;
                break;
            }
        }
        if (equal) {
            cout << i << endl;
            return 0;
        }
    }
    cout << -1 << endl;
    return 0;
}
```

2150 字符替换

小瓜有一个包含数字和字母字符的字符串，他不希望别人知道其中的数字是多少，请你帮忙把字符串中的数字字符全部替换成*。

输入

- 输入一个字符串，保证字符串长度不超过10000，并且只包含数字和字母字符。

输出

输出一个字符串，表示将输入字符串中的数字字符全部替换成*之后的字符串。

ad123de1dxb113

ad*de*dxb*

2150字符替换 解题思路

读取输入的字符 str , 遍历 str 并将其中的数字替
换为 *



2150字符替换

参考答案

```
#include <bits/stdc++.h>
using namespace std;
int main() {
    string str;
    cin >> str;
    for (int i = 0; i < str.length(); i++) {
        if (str[i] >= '0' && str[i] <= '9') {
            cout << "*";
        } else {
            cout << str[i];
        }
    }
    return 0;
}
```

字典序



课程安排

- 1、更多的输入输出方法
- 2、字典序
- 3、字符转译
- 4、编程题目：2153字典序
- 5、编程题目：3213地外遗迹

scanf ()与printf()

scanf()和 printf()是 C++语言中的一对输入输出函数，和cin和 cout相似

例如：使用scanf读取字符串

```
char dog[8];
```

```
scanf("%s", dog);
```

注意：

- 1、读取字符串时必须使用%s这个参数，表示读取的是字符串
- 2、通过键盘输入读入到字符数组中时，会默认在结尾加上空字符

A diagram illustrating the scanf function call. The code is shown in purple: `scanf(" %d , %f , %s " , &a , &b , &c);`. Three curved arrows point from the parameters to their corresponding formats in the format string:

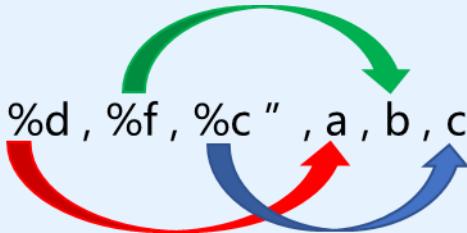
- A red arrow points from `&a` to the first `%d`.
- A blue arrow points from `&b` to the second `%f`.
- A green arrow points from `&c` to the third `%s`.

scanf()与printf()

printf的作用类似于 cout，是用来输出内容的，也可以用来输出字符串

输出时也需要使用%s这个参数，表明输出的类型是字符串。

```
printf( " %d , %f , %c " , a , b , c);
```



```
#include <bits/stdc++.h>
using namespace std;
int main ()
{
    char a[] = "hello" ;
    string b = "world" ;
    printf( "%s\n" , a); // \n 为换
行，相当于endl
    printf("%s\n", b.c_str());
}
```

printf()

printf并不能直接处理 string
类型的变量，只能处理字符数
组，所以输出变量 b的时候需
要用c_str()做一个转换

输出结果：

hello

world

ASCII码

ASCII码是基于拉丁字母的一套电脑编码系统，它是现今最通用的单字节编码系统。

电脑只认识数值，字符需要将字符对应数字，这个对应的列表，就是ASCII码表。

键盘键值表																				
Esc 27		F1 112	F2 113	F3 114	F4 115	F5 116	F6 117	F7 118	F8 119	F9 120	F10 121	F11 122	F12 123	Prt 44	Scr 145	Pau 19				
~ 192	1 49	2 50	3 51	4 52	5 53	6 54	7 55	8 56	9 57	0 48	.	= 189	— 187	— 220	— 8	Ins 45	Hom 36	PaU 33	Num 144	~ 111
TAB 9	Q 81	W 87	E 69	R 82	T 84	Y 89	U 85	I 73	O 79	P 80	[219] 221	Enter		Del 46	End 35	PaD 34	106	*	
Caps L 20	A 65	S 83	D 68	F 70	G 71	H 72	J 74	K 75	L 76	;	' 222	13 186	Shift		109	+	107	105		
	Shift 16	Z 90	X 88	C 67	V 86	B 66	N 78	M 77	,	— 188	— 190	— 191	— 16		1	2	3	Ent 97		
Ctrl 17	Win 91	Alt 18		32	Space					Alt 18	Win 92	RightK 93	Ctrl 17	38 37	39 40	0 110	.	13		

ASCII码

常见的ASCII码

'0'	48
'1'	49
'9'	57
'A'	65
'B'	66
'Z'	90
'a'	97
'b'	98
'z'	122
空格	32
回车	13

高四位		ASCII非打印控制字符								ASCII 打印字符												
		0000				0001				0010		0011		0100		0101		0110		0111		
低四位	+进制	字符	ctrl	代码	字符解释	+进制	字符	ctrl	代码	字符解释	+进制	字符	ctrl									
		0	BLANK NULL	^@	NUL 空	16	►	^P	DLE	数据链路转意	32	0	48	0	64	@	80	P	96	`	112	p
0001	1	1	☺	^A	SOH 头标开始	17	◀	^Q	DC1	设备控制 1	33	!	49	1	65	Ā	81	Q	97	ā	113	q
0010	2	2	☻	^B	STX 正文开始	18	↑	^R	DC2	设备控制 2	34	"	50	2	66	Ā	82	R	98	ā	114	r
0011	3	3	♥	^C	ETX 正文结束	19	!!	^S	DC3	设备控制 3	35	#	51	3	67	Ā	83	S	99	ā	115	s
0100	4	4	◆	^D	EOT 传输结束	20	¶	^T	DC4	设备控制 4	36	\$	52	4	68	D	84	T	100	d	116	t
0101	5	5	♣	^E	ENQ 查询	21	ƒ	^U	NAK	反确认	37	%	53	5	69	E	85	U	101	e	117	u
0110	6	6	♠	^F	ACK 确认	22	■	^V	SYN 同步空闲	38	&	54	6	70	F	86	V	102	f	118	v	
0111	7	7	●	^G	BEL 震铃	23	↕	^W	ETB 传输块结束	39	'	55	7	71	G	87	W	103	g	119	w	
1000	8	8	▣	^H	BS 退格	24	↑	^X	CAN 取消	40	(56	8	72	H	88	X	104	h	120	x	
1001	9	9	○	^I	TAB 水平制表符	25	↓	^Y	EM 媒体结束	41)	57	9	73	I	89	Y	105	i	121	y	
1010	A	10	▣	^J	LF 换行/新行	26	→	^Z	SUB 替换	42	*	58	:	74	J	90	Z	106	j	122	z	
1011	B	11	○	^K	VT 垂直制表符	27	←	^【	ESC 移意	43	+	59	;	75	K	91	【	107	k	123	{	
1100	C	12	♀	^L	FF 换页/新页	28	﹍	^＼	FS 文件分隔符	44	,	60	<	76	L	92	\	108	＼	124		
1101	D	13	♪	^M	CR 回车	29	↔	^】	GS 组分隔符	45	-	61	=	77	M	93	】	109	m	125	}	
1110	E	14	♪	^N	SO 移出	30	▲	^_	RS 记录分隔符	46	.	62	>	78	N	94	^	110	n	126	~	
1111	F	15	○	^O	SI 移入	31	▼	^-	US 单元分隔符	47	/	63	?	79	O	95	_	111	o	127	Δ	
																					Back space	

注：表中的ASCII字符可以用：ALT + “小键盘上的数字键” 输入

ASCII码

一起思考下，输出结果是什么？

```
#include <bits/stdc++.h>
using namespace std;
int main ()
{
    for(int i = 33; i <= 126; i++)
    {
        cout << i << ":" << (char)i << "
";
        if((i - 32) % 10 == 0)
            cout << endl;
    }
}
```

ASCII码

运行结果如下：

```
33:! 34:" 35:# 36:$ 37:% 38:& 39:' 40:( 41:) 42:*
43:+ 44:, 45:- 46:. 47:/ 48:0 49:1 50:2 51:3 52:4
53:5 54:6 55:7 56:8 57:9 58:: 59;; 60:< 61:= 62:>
63:? 64:@ 65:A 66:B 67:C 68:D 69:E 70:F 71:G 72:H
73:I 74:J 75:K 76:L 77:M 78:N 79:O 80:P 81:Q 82:R
83:S 84:T 85:U 86:V 87:W 88:X 89:Y 90:Z 91:[ 92:\ 
93:] 94:^ 95:_ 96:` 97:a 98:b 99:c 100:d 101:e 102:f
103:g 104:h 105:i 106:j 107:k 108:l 109:m 110:n 111:o 112:p
113:q 114:r 115:s 116:t 117:u 118:v 119:w 120:x 121:y 122:z
123:{ 124:| 125:} 126:~
```



字典序

按照 a、 b、 c..... z字母的顺序排列；

第一个字母一样，比较第二个、第三个乃至后面的字母；

如果最后两个单词不一样长（比如， sigh和sight），短者排在前

通过字典序， 可以比较任意两个字符串的大小



字典序

例如：

a < b < c < d..... y < z

aaa < b

a < aaa

两个字符的大小关系：按照ASCII 码，谁对应的数字更大，谁的值就更大



数组的字典序

数组的字典序：从两个数组的第一个值开始比较，一旦出现不相等，便可以按照那一位的大小判断整个数组的大小

```
int min = 两个数组中长度较小的;  
for(int i = 0; i < min; i++)  
{  
    if(a[i] > b[i])  
        return a更大;  
    else if(a[i] < b[i])  
        return b更大;  
}  
if(a的长度更大)  
    return a更大;  
else if(b的长度更大)  
    return b更大;  
return 一样大;
```

字符转译

```
#include <bits/stdc++.h>
using namespace std;
int main ()
{
    string s = "Hello \"world\"";
    cout << s;
}
```

字符串赋值时，用一对双引号表示赋值的内容

例如：s = "Hello world"。

如果希望s的值里面包括双引号这个字符，需要对字符进行转译处理

字符转译

转义字符	意义	ASCII码值(十进制)
\a	响铃(BEL)	7
\b	退格(BS) , 将当前位置移到前一列	8
\f	换页(FF) , 将当前位置移到下页开头	12
\n	换行(LF) , 将当前位置移到下一行开头	10
\r	回车(CR) , 将当前位置移到本行开头	13
\t	水平制表(HT) (跳到下一个TAB位置)	9
\v	垂直制表(VT)	11
\	代表一个反斜线字符\"	92
'	代表一个单引号(撇号)字符	39
"	代表一个双引号字符	34
\?	代表一个问号	63
\0	空字符(NULL)	0

2153 字典序

给你两个不同的字符串，如果第一个字符串的字典序小于第二个字符串，则输出YES，如果第一个字符串的字典序大于第二个字符串，则输出NO。

输入

- 两行。第一行一个字符串，第二行一个字符串。保证字符串的长度不超过10000。保证两个字符串不完全相等。

输出

如果第一个字符串的字典序小于第二个字符串，则输出YES，如果第一个字符串的字典序大于第二个字符串，则输出NO。

abc
abe

YES

2153 字典序（解题思路）

按照字典序的定义，逐个比较两个字符串的字符

。



2153 字典序参考答案

```
#include <bits/stdc++.h>
using namespace std;
int main() {
    string str1, str2;
    cin >> str1;
    cin >> str2;
    for (int i = 0; i <
min(str1.length(),
str2.length()); i++) {
        if (str1[i] < str2[i]) {
            cout << "YES" << endl;
            return 0;
        } if (str1[i] > str2[i]) {
            cout << "NO" << endl;
            return 0;
        }
    }
    if (str1.length() < str2.length())
    {
        cout << "YES" << endl;
    } else {
        cout << "NO" << endl;
    }
    return 0;
}
```

3213 地外遗址

题目描述：

你的朋友琼斯是一个星际冒险家，你知道的，星际冒险家总是会遇见一些奇怪的情况。今天琼斯发了一个信息向你求助。

琼斯在距离地球不远的星球上发现了一座远古遗迹，远古遗迹的大门上有着奇怪的机关。琼斯通过一段时间的摸索结合他的冒险经验得出了一个结论：机关的屏幕上每次会出现两个单词，如果第一个单词的字典序小于第二个单词的字典序，就需要按下绿色的按钮；反之则按下红色的按钮。

有一个额外的问题需要注意。因为遗迹的年代久远，遗迹主人所使用的的语言与我们的语言的字典序有一些差别。当然，见多识广的琼斯肯定是了解这些差别的，他会告诉你正确的字典序。那么你能帮他写一个程序解开大门的机关么？

PS:单词只包含小写字母

例如：uvwxyzabcdefghijklmnopqrst

表示，在遗迹语言中，u是字典序最小的，而t是字典序最大的。按照他们的字典序来排，u排在 a 的前面。

3213 地外遗迹

输入

第一行，一个长度为26的小写字母的字符串（就是琼斯告诉你的字典序，字母越前字典序越小，不会出现重复的小写字母）

第二行，一个数字 n，表示后面需要进行 n 次比较 ($1 \leq n \leq 100000$)

之后的n行，每行两个单词，中间用空格隔开

uvwxyzabcdefghijklmnopqrstuvwxyz

5

apple banana
banana blueberry
apple watermelon
vegetable banana
apple ap

输出

如果输入的两个单词，按照遗迹的字典序，第一个单词的字典序小于第二个单词的字典序，输出“green”，否则输出“red”。

green

green

red

green

red

3213 地外遗址

解题思路

可以按照给出的字典序，自定义比较函数，然后比较双方的大小，也可以简单的把 2 个单词转为正常字典序下的单词，然后直接比较双方的大小，例如在字典序：

uvwxyzabcdefghijklmnopqrst

uxc 对应正常字典序的单词为 adi

uvwxy 对应正常字典序的单词为 abcd

在正常字典序下， abcd<adi，所以在指定的字典序下， uvwx < uxc



3213 地外遗址 参考答案

```
#include <bits/stdc++.h>
using namespace std;
string table, rTable;
string replace(string s)
{
    for(int i = 0; i < s.size(); i++)
        s[i] = rTable[s[i] - 'a'];
    return s;
}
int main()
{
    int n;
    cin >> table >> n;
    rTable = table;
    for(int i = 0; i < 26; i++)
    {
        rTable[table[i] - 'a'] = (char)('a' + i);
    }
    for(int i = 0; i < n; i++)
    {
        cin >> a >> b;
        a = replace(a);
        b = replace(b);
        if(a < b)
            cout << "green" << endl;
        else
            cout << "red" << endl;
    }
    return 0;
}
```

总结

我们一起学习了字典序和字符转义。

它们是容易遗忘的知识点，请同学们多多温习~



字典序习题课



课程安排

- 1、编程题目：2153字典序
- 2、编程题目：3213地外遗迹
- 3、编程题目：1506最小字典序

2153 字典序

给你两个不同的字符串，如果第一个字符串的字典序小于第二个字符串，则输出YES，如果第一个字符串的字典序大于第二个字符串，则输出NO。

输入

- 两行。第一行一个字符串，第二行一个字符串。保证字符串的长度不超过10000。保证两个字符串不完全相等。

输出

如果第一个字符串的字典序小于第二个字符串，则输出YES，如果第一个字符串的字典序大于第二个字符串，则输出NO。

abc
abe

YES

2153 字典序（解题思路）

按照字典序的定义，逐个比较两个字符串的字符

。



2153 字典序参考答案

```
#include <bits/stdc++.h>
using namespace std;
int main() {
    string str1, str2;
    cin >> str1;
    cin >> str2;
    for (int i = 0; i <
min(str1.length(),
str2.length()); i++) {
        if (str1[i] < str2[i]) {
            cout << "YES" << endl;
            return 0;
        } if (str1[i] > str2[i]) {
            cout << "NO" << endl;
            return 0;
        }
    }
    if (str1.length() < str2.length())
    {
        cout << "YES" << endl;
    } else {
        cout << "NO" << endl;
    }
    return 0;
}
```

3213 地外遗迹

你的朋友琼斯是一个星际冒险家，你知道的，星际冒险家总是会遇见一些奇怪的情况。今天琼斯发了一个信息向你求助。

琼斯在距离地球不远的星球上发现了一座远古遗迹，远古遗迹的大门上有着奇怪的机关。琼斯通过一段时间的摸索结合他的冒险经验得出了一个结论：机关的屏幕上每次会出现两个单词，如果第一个单词的字典序小于第二个单词的字典序，就需要按下绿色的按钮；反之则按下红色的按钮。

有一个额外的问题需要注意。因为遗迹的年代久远，遗迹主人所使用的的语言与我们的语言的字典序有一些差别。当然，见多识广的琼斯肯定是了解这些差别的，他会告诉你正确的字典序。那么你能帮他写一个程序解开大门的机关么？

PS:单词只包含小写字母

例如：uvwxyzabcdefghijklmнопqrstuvwxyz

表示，在遗迹语言中，u是字典序最小的，而t是字典序最大的。按照他们的字典序来排，u排在 a 的前面。

3213 地外遗迹

输入

第一行，一个长度为26的小写字母的字符串（就是琼斯告诉你的字典序，字母越前字典序越小，不会出现重复的小写字母）

第二行，一个数字 n，表示后面需要进行 n 次比较 ($1 \leq n \leq 100000$)

之后的n行，每行两个单词，中间用空格隔开

uvwxyzabcdefghijklmnopqrstuvwxyz

5

apple banana
banana blueberry
apple watermelon
vegetable banana
apple ap

输出

如果输入的两个单词，按照遗迹的字典序，第一个单词的字典序小于第二个单词的字典序，输出“green”，否则输出“red”。

green

green

red

green

red

3213 地外遗址（解题思路）

可以按照给出的字典序，自定义比较函数，然后比较双方的大小，也可以简单的把 2 个单词转为正常字典序下的单词，然后直接比较双方的大小，例如在字典序：

uvwxyzabcdefghijklmnopqrst

uxc 对应正常字典序的单词为 adi

uvwx 对应正常字典序的单词为 abcd

在正常字典序下， $abcd < adi$ ，所以在指定的字典序下， $uvwx < uxc$



3213 地外遗址（参考答案）

```
#include <bits/stdc++.h>
using namespace std;
string table, rTable;
string replace(string s)
{
    for(int i = 0; i < s.size(); i++)
        s[i] = rTable[s[i] - 'a'];
    return s;
}
int main()
{
    int n;
    cin >> table >> n;
    rTable = table;
    for(int i = 0; i < 26; i++)
    {
        rTable[table[i] - 'a'] = (char)('a' + i);
    }
    for(int i = 0; i < n; i++)
    {
        cin >> a >> b;
        a = replace(a);
        b = replace(b);
        if(a < b)
            cout << "green" << endl;
        else
            cout << "red" << endl;
    }
    return 0;
}
```

1506 最小字典序

题目描述：

给出一个字符串S，你需要从S中挑选一对字符进行一次交换（不可以不交换！！！），并让得到的新字符串字典序最小！

例如：S = "abacc"，

如果交换字符1(a)和4(c)，得到字符"cbaac"。

如果交换字符2(a)和3(b)，得到字符"aabcc"。

其中："aabcc"的字典序小于"cbaac"。并且"aabcc"是所有交换方法中，字典序最小的。

例如：S = "aaab"，

则交换1和2，交换1和3，得到的字符都是 "aaab"，并且 "aaab" 是所有交换方法中，字典序最小的。

输出这个字典序最小的字符。

1506 最小字典序

输入

一个字符串S (S的只包括a到z的小写字符 , $2 \leq |S| \leq 500000$)。

输出

字典序最小的新字符串。

abacc

aabcc

1506 最小字典序（解题思路）

先要思考清楚应该如何换。

以 abcaba 为例：

首先要确定，换哪个字符，首字符 a 不用换，因为没有字典序比他更小的字符。

第 2 个字符 b 需要换，因为把 b 换成 a ，可以让字典序变得更小。

如果第 2 个字符需要换，那么后面的就无需考虑了，因为后面无论如何换都不能变得更小。

既然 b 需要交换，我们再考虑需要和哪一个交换， b 的后面有两个 a ，为了让字典序更小，需要尽量换后面的 a ，也就是第 6 个字符 a 。

最后还要考虑，假如无论如何换，也不能让字典序更小的情况，这时如果有某个字符出现了 2 次。交换这两个字符字典序不变。如果所有字符只出现了一次，那么交换最后两个字符，可以让字典序变大的最少。

1506 最小字典序 参考答案

```
#include <bits/stdc++.h>
using namespace std;
int main() {
    string s;
    cin >> s;
    int n = s.size();
    int minIndex = n - 1, swapA = n - 2, swapB
    = n - 1;
    for(int i = n - 2; i >= 0; i--) {
        if(s[i] < s[minIndex])
            minIndex = i;
        else if(s[i] > s[minIndex] || s[swapB] >
s[swapA]) {
            swapA = i; swapB = minIndex;
        }
    }
    swap(s[swapA], s[swapB]);
    cout << s << endl;
    return 0;
}
```

基础排序



课程安排

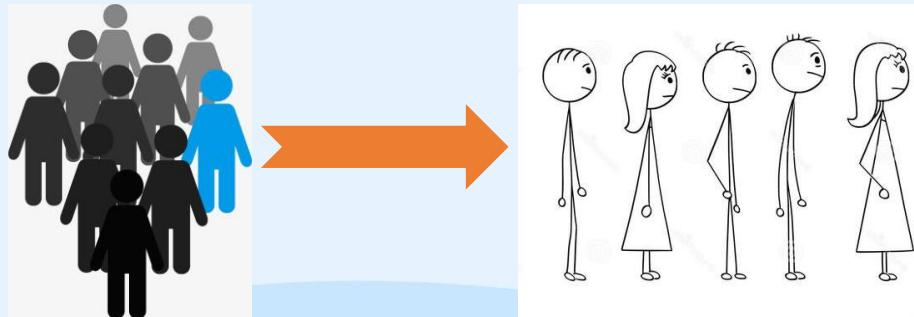
- 1、选择排序
- 2、冒泡排序
- 3、插入排序
- 4、C++的排序
- 5、编程题目：2108排序100
- 6、编程题目：3212数字变位

排序的概念

排序是将一组“无序”的序列调整为“有序”的序列

常见排序算法包括：

快速排序、希尔排序、堆排序、基数排序、冒泡排序、插入排序、归并排序、选择排序

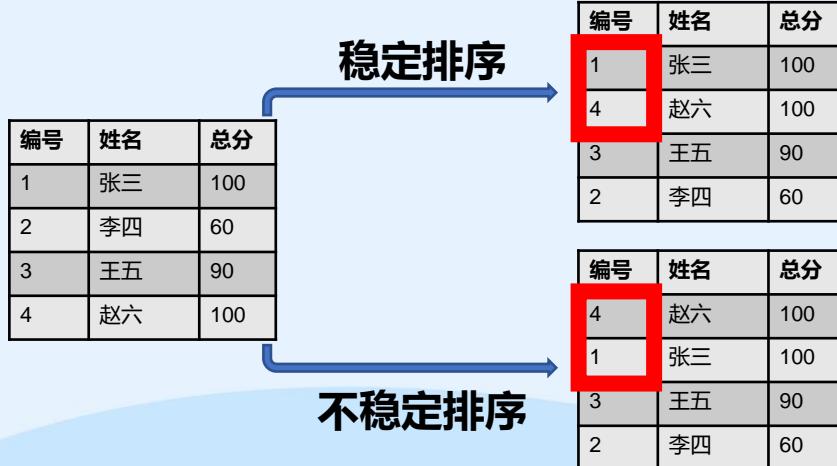


稳定排序

稳定排序：使用某种排序后，相对次序仍然不变，则排序是稳定的

冒泡排序，插入排序属于稳定排序

选择排序属于不稳定排序



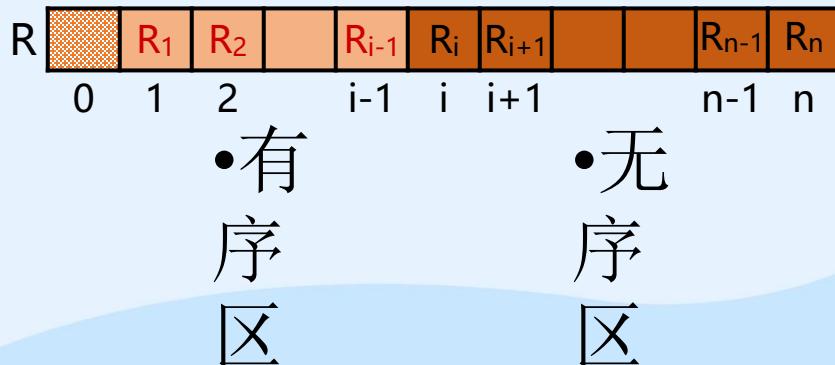
排序的效率

	效率	稳定
冒泡排序	慢	是
插入排序	慢	是
选择排序	慢	否

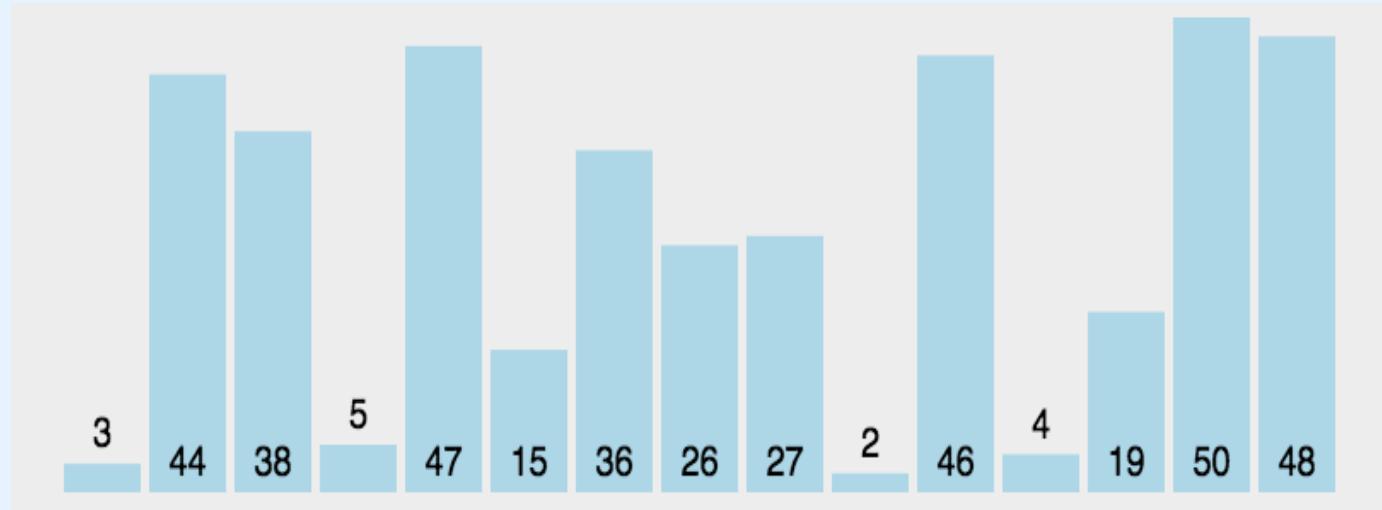
选择排序

选择排序(selection sort)是一种简单直观的排序算法。

原理：第一次从待排序的数据元素中选出最小（或最大）的一个元素，存放在序列的起始位置，然后再从剩余的未排序元素中寻找到最小（大）元素，然后放到已排序的序列的末尾。



选择排序



选择排序 算法实现

```
void sort()
{
    for(int i = 0; i < n; i++)
    {
        int minIndex = i;
        for(int j = i + 1; j < n; j++)
            if(nums[j] < nums[minIndex])
                minIndex = j;
        //交换当前和最小
        int t = nums[minIndex];
        nums[minIndex] = nums[i];
        nums[i] = t;
    }
}
```

冒泡排序

算法描述：冒泡排序(Bubble sort)是一种计算机科学领域的较简单的排序算法。

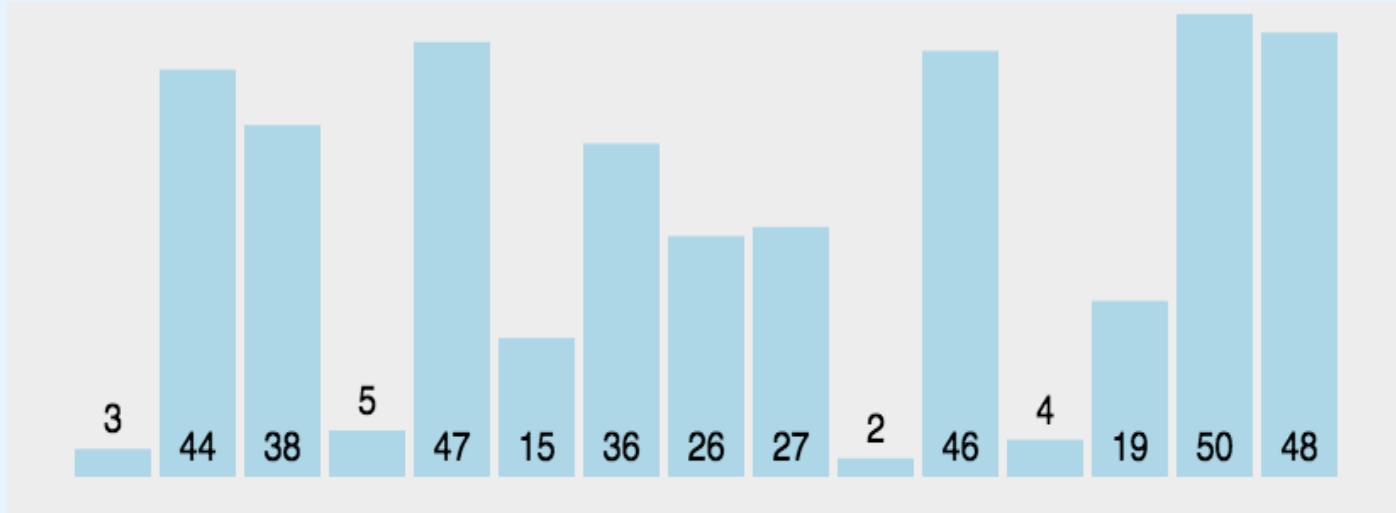
原理：重复走访过要排序的元素，依次比较两个相邻的元素，如果顺序错误就交换。

- 直到没有相邻元素需要交换，排序完成。

越小的元素经交换慢慢“浮”到数列的顶端（升序或降序排列），故名“冒泡排序”。



冒泡排序 算法演示

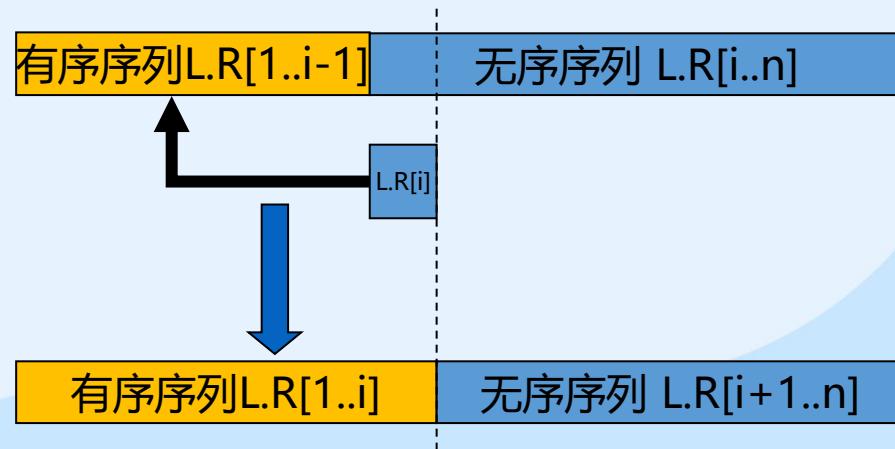


冒泡排序 算法实现

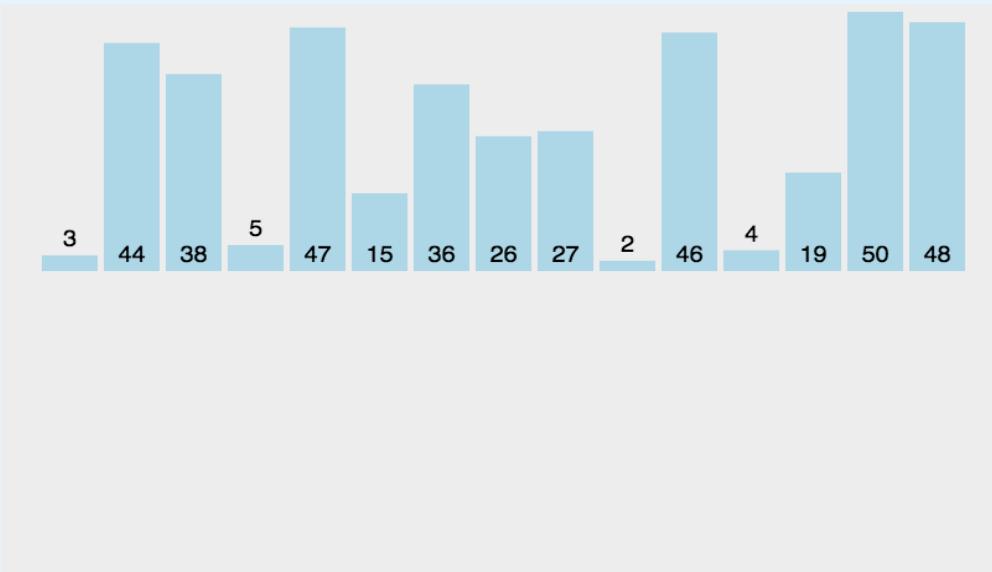
```
void sort()
{
    for(int i = 0; i < n; i++)
    {
        for(int j = 1; j < n - i; j++)
        {
            //如果后面的更小，则交换
            if(nums[j] < nums[j - 1])
            {
                int t = nums[j];
                nums[j] = nums[j - 1];
                nums[j - 1] = t;
            }
        }
    }
}
```

插入排序

插入排序的基本思想是：每步将一个待排序的数，按值的大小插入前面已经排序的数组中的适当位置上，直到全部插入完为止。
(类似于打扑克的摸牌理牌过程)



插入排序 算法演示



插入排序 算法实现

```
void sort()
{
    for(int i = 1; i < n; i++)
    {
        for(int j = i; j >= 0; j--)
        {
            //如果当前的更小，则交换
            if(nums[j] < nums[j-1])
            {
                int t = nums[j];
                nums[j] = nums[j - 1];
                nums[j - 1] = t;
            }
            else
                break;
        }
    }
}
```

C++的排序

C++的标准库里，已经提供了可以用来排序的函数
：sort

sort函数能够将数组的某一段元素按照要求进行排序，并且运行效率非常高



C++的排序

sort函数标准：sort(start,end,排序方法)

sort函数有三个参数：

第一个：排序数组的起始地址。

第二个：排序数组结束地址的下一个地址，即数组[开始, 结束]的左闭右开区间。

第三个：数组的比较函数，是规定的排序规则，可以不填，默认不填排序规则是从小到大。



sort()程序

```
#include <bits/stdc++.h>
using namespace std;
int main()
{
    int a[10] = { 9,6,3,8,5,2,7,4,1,0 };
    for (int i = 0; i<10; i++)
        cout << a[i];
    cout << endl;
    sort(a, a + 10); //第三个参数不用写，默认从小到大排序
    for (int i = 0; i<10; i++)
        cout << a[i];
    cout << endl;
    return 0;
}
```

输出结果：

9638527410

0123456789

sort()

sort(a,a+10)的意思是：对数组a的开始位置（第1个元素）到第10个元素的位置进行排序。

如果想要从大到小排序呢？

- 这时候要用到比较函数，写一个函数名叫cmp，这个比较函数需要传入两个参数a和b，类型就是要排序数组的类型，想要升序排序，只需要在函数内return $a > b$ 即可。



sort()程序

```
#include <bits/stdc++.h>
using namespace std;
bool cmp(int a, int b)
{
    return a > b; //降序就是a < b
}
int main()
{
    int a[10] = { 9,6,3,8,5,2,7,4,1,0 };
    for (int i = 0; i < 10; i++)
        cout << a[i];
        cout << endl;
    sort(a, a + 10, cmp); //排序函数，从大到小
    //排序
    for (int i = 0; i < 10; i++)
        cout << a[i];
        cout << endl;
    return 0;
}
```

运行结果：

9638527410

9876543210

sort()

sort(a,a+10,cmp)最后一个参数cmp，指向一个返回值为bool的函数，利用这个函数，可以自行定义，什么是大什么是小。

C++标准库的sort，是依照快速排序的算法实现的

。

如果使用讲过的3种排序，排序一个长度为100w的数组，大概需要十多分钟。

而使用标准库的快速排序，只需要几秒钟时间。



2108 排序100

输入一个长度为n的数组，将他排成升序，即对于任意相邻2个数字 $a[i], a[i+1]$ 来说， $a[i] \leq a[i+1]$ 。

输入

- 第一行一个整数n，表示数字长度
接下来n行，每行一个整数 $a[i]$ ，表示数组的内容。
 $1 \leq n \leq 100$ ， $1 \leq a[i] \leq 10^9$

输出

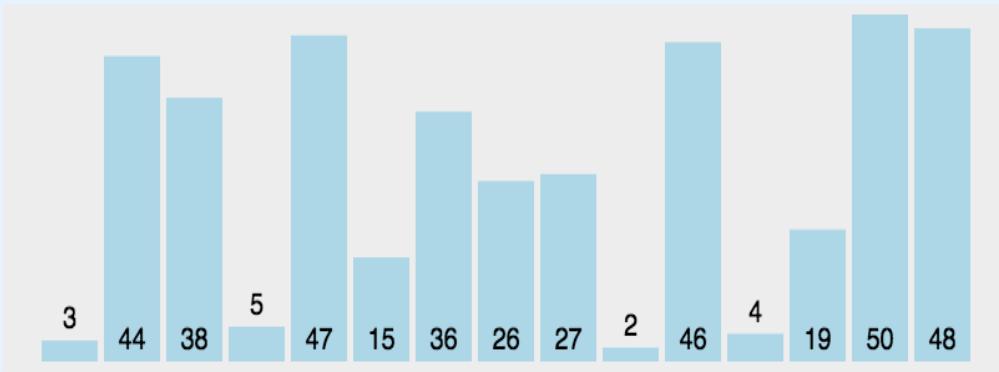
输出第一行为数组长度n
接下来n行为排序后的结果。

4
4
3
1
2

4
1
2
3
4

排序100 解题思路

可以使用选择排序来实现。



2108排序100

参考答案

```
#include <bits/stdc++.h>
using namespace std;
int nums[101], n;
void sort()
{
    for(int i = 0; i < n; i++)
    {
        int minIndex = i;
        for(int j = i + 1; j < n; j++)
            if(nums[j] < nums[minIndex])
                minIndex = j;
        //交换当前和最小
        int t = nums[minIndex];
        nums[minIndex] = nums[i];
        nums[i] = t;
    }
}
int main()
{
    cin >> n;
    for(int i = 0; i < n; i++)
        cin >> nums[i];
    sort();
    cout << n << endl;
    for(int i = 0; i < n; i++)
        cout << nums[i] << endl;
    return 0;
}
```

3212 数字变位

小明有一个数字x，现在他想把x的各个数字调换位置，得到一个最大的数和一个最小的数，你能帮帮他吗？

注：小明不希望数字的最高位是0。

输入

输入一个数x；

输出

输出两个数，分别表示调换后的最大数和最小数，以空格隔开；

9037

9730 3079

3212数字变位（解题思路）

将 x 的每一位存入数组 a ，对 a 按照从小到大排序。

倒序输出 a 就是最大的数。

最小的数需要考虑前缀 0 的问题，如果字符中有 0，我们找到最小的非 0 数字，把他跟第一个 0 进行交换，然后正序输出，就是可以组成的最小的数。



3212数字变位 参考答案

```
#include <bits/stdc++.h>
using namespace std;
int digs[20];
int main()
{
    int n, index = 0;
    cin >> n;
    while(n != 0)
    {
        digs[index] = n % 10;
        n /= 10;
        index++;
    }
    sort(digs, digs + index);
    for(int i = index - 1; i >= 0; i--)
        cout << digs[i];
    cout << " ";
```

3212数字变位 参考答案

```
if(digs[0] == 0)
{
    for(int i = 0; i < index; i++)
    {
        if(digs[i] != 0)
        {
            digs[0] = digs[i];
            digs[i] = 0;
            break;
        }
    }
    for(int i = 0; i < index; i++)
        cout << digs[i]
    return 0;
}
```

总结

我们一起学习了选择排序、冒泡排序和插入排序

。

它们在NOIP中经常考察，请同学们多多温习~



基础排序习题课



课程安排

1、编程题目：2108排序

100

2、编程题目：3212数字变
位

3、编程题目：2142第m大
的身份证号码

2108 排序100

输入一个长度为n的数组，将他排成升序，即对于任意相邻2个数字 $a[i], a[i+1]$ 来说， $a[i] \leq a[i+1]$ 。

输入

第一行一个整数n，表示数字长度
接下来n行，每行一个整数 $a[i]$ ，表示数组的内容。
 $1 \leq n \leq 100$ ， $1 \leq a[i] \leq 10^9$

输出

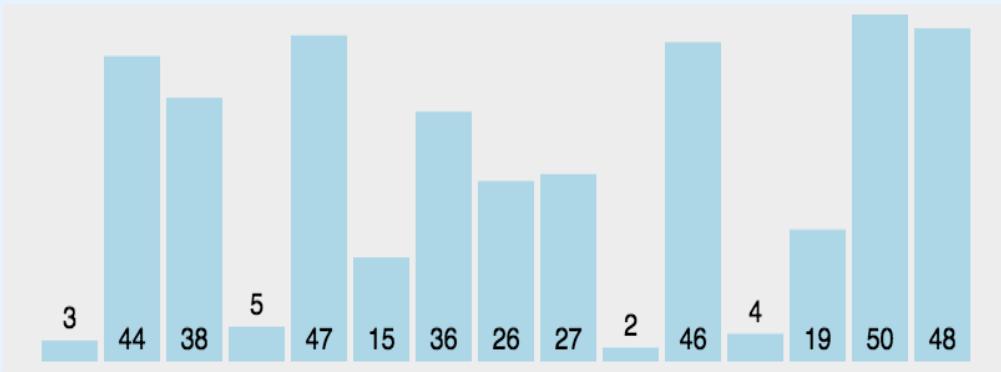
输出第一行为数组长度n
接下来n行为排序后的结果。

4
4
3
1
2

4
1
2
3
4

排序100 解题思路

可以使用选择排序来实现。



2108排序100

参考答案

```
#include <bits/stdc++.h>
using namespace std;
int nums[101], n;
void sort()
{
    for(int i = 0; i < n; i++)
    {
        int minIndex = i;
        for(int j = i + 1; j < n; j++)
            if(nums[j] < nums[minIndex])
                minIndex = j;
        //交换当前和最小
        int t = nums[minIndex];
        nums[minIndex] = nums[i];
        nums[i] = t;
    }
}
int main()
{
    cin >> n;
    for(int i = 0; i < n; i++)
        cin >> nums[i];
    sort();
    cout << n << endl;
    for(int i = 0; i < n; i++)
        cout << nums[i] << endl;
    return 0;
}
```

3212 数字变位

小明有一个数字x，现在他想把x的各个数字调换位置，得到一个最大的数和一个最小的数，你能帮帮他吗？

注：小明不希望数字的最高位是0。

输入

输入一个数x；

输出

输出两个数，分别表示调换后的最大数和最小数，以空格隔开；

9037

9730 3079

3212数字变位 解题思路

将x的每一位存入数组a，对a按照从小到大排序。

倒序输出a就是最大的数。

最小的数需要考虑前缀0的问题，如果字符中有0，找到最小的非0数字，把它跟第一个0进行交换，然后正序输出，就是可以组成的最小的数。



3212数字变位 参考答案

```
#include <bits/stdc++.h>
using namespace std;
int digs[20];
int main()
{
    int n, index = 0;
    cin >> n;
    while(n != 0)
    {
        digs[index] = n % 10;
        n /= 10;
        index++;
    }
    sort(digs, digs + index);
    for(int i = index - 1; i >= 0; i--)
        cout << digs[i];
    cout << " ";
```

3212数字变位 参考答案

```
if(digs[0] == 0)
{
    for(int i = 0; i < index; i++)
    {
        if(digs[i] != 0)
        {
            digs[0] = digs[i];
            digs[i] = 0;
            break;
        }
    }
    for(int i = 0; i < index; i++)
        cout << digs[i]
    return 0;
}
```

2142第m大的身份证号码

身份证号是我国公民的唯一识别码，它由 18 位数字或者字母组成（只能最后一位是字母）。

18 位身份证号码的含义如下：第 1~2 为省、自治区、直辖市代码；第 3~4 位为地级市、盟、自治州代码；第 5~6 位为县、县级市、区代码；第 7~14 位位出生年月日，比如 19970401 代表 1997 年 4 月 1 日；第 15~16 位为顺序号；第 17 位代表性别，男为单数，女为双数；第 18 位为校验码，0~9 和 X。

作为尾号的校验码，是把前十七位数字代入统一的公式计算出来的。

解答本题你不用关心是如何计算出来的。现在给你n个身份证号码，请你按照出生年月日的字典序（年龄从大到小）输出第m个人的身份证号。

一些解释：虽然造数据的人非常辛苦的制造各种各样的身份证号（并且让他们生日互不相同），但是你并不需要验证关于身份证号的任何合法性（包括省市区是否合法，出生年月日是否合法，校验值是否合法，你需要做的仅仅是输出年龄从大到小第m个人的身份证号）。

2142 第m大的身份证号码

输入

输入第一行包含两个正整数 n 和 m ，
两数之间用一个空格分隔，接下来的
 n 行每行为一个形如上述格式的身份
证号码（不需要关心校验码的正确性
，不影响本题解答）。($1 \leq n \leq 100$ ，
 $1 \leq m \leq n$)

```
4 2
110108196004063022
13021119640203652X
420333197902112718
210222200012036512
```

输出

输出仅包含一行，为题目要求的一个
身份证号码。

```
13021119640203652X
```

2142第m大的身份证号码（解题思路）

使用标准库自带的 sort 对所有身份证号进行排序，自己定义其中的 compare 函数，然后输出第 m 个。



2142第m大的身份证号码 参考答案

```
#include<bits/stdc++.h>
using namespace std;
typedef long long ll;
string a[105];
bool com(string a, string b)
{
    for(int i = 6; i < 14; i++)
        if(a[i] > b[i])
            return false;
        else if(a[i] < b[i])
            return true;
    return false;
}
int main()
{
    int n, m;
    cin >> n >> m;
    for(int i = 0; i < n; i++)
        cin >> a[i];
    sort(a, a + n, com);
    cout << a[m - 1];
    return 0;
}
```

位图与计数



课程安排

- 1、位图
- 2、编程题目：2113丢失的数字
- 3、使用数组计数
- 4、编程题目：2114多出的数字
- 5、编程题目：1915西湖游船

位图的概念

位图是用每一位来存放某种状态，适用于大规模数据，但数据状态又不是很多。
通常是用来判断某个数据存不存在。



2113 丢失的数字

给你m个1到n之间的整数，你能找出1到n中的哪些整数没有出现吗？

输入

- 第一行2个整数n,m，直接用空格分隔($1 \leq n \leq 100000$, $m < n$)，表示有m个1到n之间的整数。
接下来m行，每行一个整数 a_i ($1 \leq a_i \leq n$ ，数据保证m个数都不相同)。

输出

每行1个数，从小到大输出输入数据中没有出现过的1到n中的整数。

5 3
3
1
4

2
5

2113丢失的数字（解题思路）

按照一般的解题思路，我们应当先将所有数字存入数组。然后从1开始枚举，判断数组中是否存在这个数。

这需要一个双重循环来做，内层循环 $1-m$ ，外层循环 $1-n$ 。如果 n 的范围是 $10w$ ，这两重循环总的循环次数就是 $50亿$ 。

如何设计一个快速的方法来判断每个数是否存在呢？



2113丢失的数字（解题思路）

注意到n的范围是10w，我们用一个长度为10w的bool数组，来记录所有数字，记录方法如下：

例如：n=5,m=3。三个数字分别为1,3,5

对应的bool数组a的值如下：

a[0] = false、 a[1] = true、 a[2] = false、 a[3] = true、 a[4] = false、
a[5] = true

所有有数字的下标值为true，其他为false。

三个数字为1,3,5。则a[1]=a[3]=a[5]=true

用类似的方法来处理本题，我们只需要开一个长度为10w+1的数组，然后循环读入所有数字，设定对应的下标为true，然后再从1-n循环输出下标为false的下标。

2113丢失的数字 参考答案

```
#include<bits/stdc++.h>
using namespace std;
int n;
bool map[100010];
int main() {
    cin >> n;
    memset(map, false, sizeof(map));
    int a;
    for (int i = 1; i <= n; i++) {
        cin >> a;
        map[a] = true;
    }
    for (int i = 1; i <= n; i++) {
        if (!map[i]) {
            cout << i << endl;
        }
    }
    return 0;
}
```

使用数组计数

数组除了能够用来保存有或无的状态，还可以用来计数。

使用 `bool` 数组自然无法完成这个任务，需要使用 `int` 数组。



2114 多出的数字

给你m个1到n之间的整数，你能找出1到n中的哪些整数出现了多次吗？

输入

- 第一行2个整数n,m，直接用空格分隔($n \leq 100000, n < m < 2n$)，表示有m个1到n之间的整数。

接下来m行，每行一个整数 a_i ($1 \leq a_i \leq n$)。

输出

若干行，每行两个数 a_i 和 b_i ，从小到大输出输入数据中出现了超过1次的1到n中的整数 a_i 和它出现的次数 b_i 。

5 7
1
1
5
2
4
4
3

1 2
4 2

2114多出的数字（解题思路）

与位图的方法类似，用一个 int 数组 map。map[i] 中记录了数字 i 出现了多少次。

例如：

1,4,3,6,3,2,1 对应的 map 为：

0,2,1,2,1,0,1，其中 1,3 出现各两次，所以
 $map[1]=map[3]=2$ 。

逐个读入数据，设置 map 对应下标的计数，最终输出所有计数大于 1 的下标。



2114多出的数字 参考答案

```
#include<bits/stdc++.h>
using namespace std;
int n;
int map[100010];
int main() {
    cin >> n;
    memset(map, 0, sizeof(map));
    int a;
    for (int i = 1; i <= n; i++) {
        cin >> a;
        map[a]++;
    }
    for (int i = 1; i <= n; i++) {
        if (map[i] > 1) {
            cout << i << " " << map[i] << endl;
        }
    }
    return 0;
}
```

1915 西湖游船

有n名游客在西湖游玩，现在他们要上船观光。游客编号1到n。船的最大承重为W。第i个人的重量为c[i]。现在有若干次游客上下船的操作，请统计一下整个过程中船所承受过的最大总重量是多少。

1915 西湖游船

输入

单组测试数据。

第一行输入三个整数n($1 \leq n \leq 20$), m($1 \leq m \leq 500$)和 W ($1 \leq W \leq 10000$)。

接下来n行输入n名游客的重量c[i]($1 \leq c[i] \leq 1000$)

。

接下来m行，每行一个1到n之间的整数，表示游客的编号。表示该游客的上下船，如果该游客已经在船上，那么该游客就下船，反之就上船。刚开始的时候所有游客都不在船上。

输出

输出一个整数表示整个上下船过程中，船所受到的最大重量。如果最大重量超过了船的最大承重能力，输出

Oh, My God!

样例输入1

2 2 10
5
7
1
2

样例输入2

3 6 10
2
5
7
2
1
2
3
1
3

样例输出1

Oh, My God!
样例输出2

9

1915西湖游船（解题思路）

利用一个 bool 数组来记录游客在船上还是未上船

。

利用循环读取数据，同时维护船的载重即可。



1915西湖游船（参考答案）

```
#include<bits/stdc++.h>
using namespace std;
int main()
{
    int a[21],b[21]={0};
    int n,m,w,wi1=0,wi2=0,ck,c=0;
    cin>>n>>m>>w;
    for(int i=1;i<=n;i++) cin>>a[i];
    for(int i=1;i<=m;i++)
    {
        cin>>ck;
        if(b[ck]==0)
            {wi1=wi1+a[ck];b[ck]=1;}
        else {wi1-=a[ck];b[ck]=0;}
        if(wi2<=wi1) wi2=wi1;
        if(wi2>w) cout<<"Oh, My
God!";
        if(wi2>w) {c=1;break;}
    }
    if(c==0) cout<<wi2;
    return 0;
}
```

总结

我们一起学习了位图和使用数组计数。

它们是NOIP中较为重要的知识点，请同学们多多温习~



专题二十九：位图与计数



课程安排

- 1、编程题目：2113丢失的数字
- 2、编程题目：2114多出的数字
- 3、编程题目：1915西湖游船

2113 丢失的数字

给你m个1到n之间的整数，你能找出1到n中的哪些整数没有出现吗？

输入

- 第一行2个整数n,m，直接用空格分隔($1 \leq n \leq 100000$, $m < n$)，表示有m个1到n之间的整数。
接下来m行，每行一个整数 a_i ($1 \leq a_i \leq n$ ，数据保证m个数都不相同)。

输出

每行1个数，从小到大输出输入数据中没有出现过的1到n中的整数。

5 3
3
1
4

2
5

2113丢失的数字（解题思路）

按照一般的解题思路，我们应当先将所有数字存入数组。然后从1开始枚举，判断数组中是否存在这个数。

这需要一个双重循环来做，内层循环 $1-m$ ，外层循环 $1-n$ 。如果 n 的范围是 $10w$ ，这两重循环总的循环次数就是 $50亿$ 。

如何设计一个快速的方法来判断每个数是否存在呢？



2113丢失的数字（解题思路）

注意到n的范围是10w，我们用一个长度为10w的bool数组，来记录所有数字，记录方法如下：

例如：n=5,m=3。三个数字分别为1,3,5

对应的bool数组a的值如下：

a[0] = false、 a[1] = true、 a[2] = false、 a[3] = true、 a[4] = false、
a[5] = true

所有有数字的下标值为true，其他为false。

三个数字为1,3,5。则a[1]=a[3]=a[5]=true

用类似的方法来处理本题，我们只需要开一个长度为10w+1的数组，然后循环读入所有数字，设定对应的下标为true，然后再从1-n循环输出下标为false的下标。

2113丢失的数字 参考答案

```
#include<bits/stdc++.h>
using namespace std;
int n;
bool map[100010];
int main() {
    cin >> n;
    memset(map, false, sizeof(map));
    int a;
    for (int i = 1; i <= n; i++) {
        cin >> a;
        map[a] = true;
    }
    for (int i = 1; i <= n; i++) {
        if (!map[i]) {
            cout << i << endl;
        }
    }
    return 0;
}
```

2114 多出的数字

给你m个1到n之间的整数，你能找出1到n中的哪些整数出现了多次吗？

输入

- 第一行2个整数n,m，直接用空格分隔($n \leq 100000, n < m < 2n$)，表示有m个1到n之间的整数。

接下来m行，每行一个整数 a_i ($1 \leq a_i \leq n$)。

输出

若干行，每行两个数 a_i 和 b_i ，从小到大输出输入数据中出现了超过1次的1到n中的整数 a_i 和它出现的次数 b_i 。

5 7
1
1
5
2
4
4
3

1 2
4 2

2114多出的数字（解题思路）

与位图的方法类似，用一个 int 数组 map。map[i] 中记录了数字 i 出现了多少次。

例如：

1,4,3,6,3,2,1 对应的 map 为：

0,2,1,2,1,0,1，其中 1,3 出现各两次，所以
 $map[1]=map[3]=2$ 。

逐个读入数据，设置 map 对应下标的计数，最终输出所有计数大于 1 的下标。



2114多出的数字 参考答案

```
#include<bits/stdc++.h>
using namespace std;
int n;
int map[100010];
int main() {
    cin >> n;
    memset(map, 0, sizeof(map));
    int a;
    for (int i = 1; i <= n; i++) {
        cin >> a;
        map[a]++;
    }
    for (int i = 1; i <= n; i++) {
        if (map[i] > 1) {
            cout << i << " " << map[i] << endl;
        }
    }
    return 0;
}
```

1915 西湖游船

有n名游客在西湖游玩，现在他们要上船观光。游客编号1到n。船的最大承重为W。第i个人的重量为c[i]。现在有若干次游客上下船的操作，请统计一下整个过程中船所承受过的最大总重量是多少。

1915 西湖游船

输入

单组测试数据。

第一行输入三个整数n($1 \leq n \leq 20$), m($1 \leq m \leq 500$)和 W ($1 \leq W \leq 10000$)。

接下来n行输入n名游客的重量c[i]($1 \leq c[i] \leq 1000$)

。

接下来m行，每行一个1到n之间的整数，表示游客的编号。表示该游客的上下船，如果该游客已经在船上，那么该游客就下船，反之就上船。刚开始的时候所有游客都不在船上。

输出

输出一个整数表示整个上下船过程中，船所受到的最大重量。如果最大重量超过了船的最大承重能力，输出

Oh, My God!

样例输入1

2 2 10
5
7
1
2

样例输入2

3 6 10
2
5
7
2
1
2
3
1
3

样例输出1

Oh, My God!

样例输出2

9

1915西湖游船（解题思路）

利用一个 bool 数组来记录游客在船上还是未上船

。

利用循环读取数据，同时维护船的载重即可。



1915西湖游船（参考答案）

```
#include<bits/stdc++.h>
using namespace std;
int main()
{
    int a[21],b[21]={0};
    int n,m,w,wi1=0,wi2=0,ck,c=0;
    cin>>n>>m>>w;
    for(int i=1;i<=n;i++) cin>>a[i];
    for(int i=1;i<=m;i++)
    {
        cin>>ck;
        if(b[ck]==0)
            {wi1=wi1+a[ck];b[ck]=1;}
        else {wi1-=a[ck];b[ck]=0;}
        if(wi2<=wi1) wi2=wi1;
        if(wi2>w) cout<<"Oh, My
God!";
        if(wi2>w) {c=1;break;}
    }
    if(c==0) cout<<wi2;
    return 0;
}
```