

# 基础排序算法

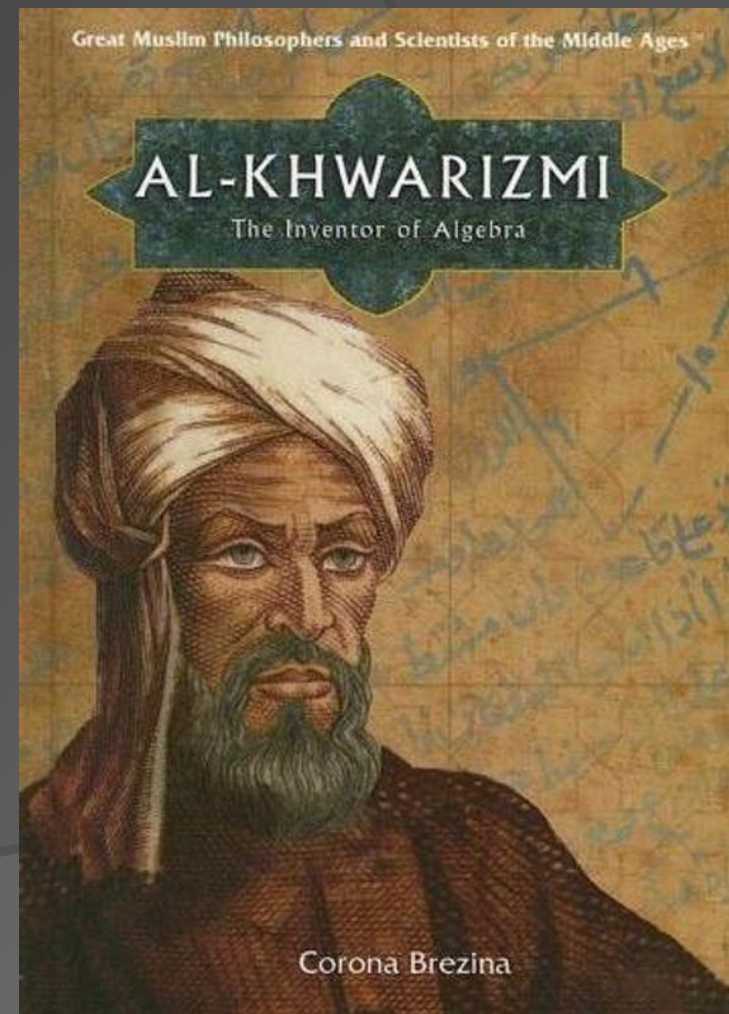
# 什么是算法？

筛选法、欧几里得算法、前缀和、分解因数和质因数.....

简单的说：解决问题的方法或步骤的描述，我们称为“算法”

算法(**algorithm**)这个单词最早出现在波斯数学家阿勒.花刺子密在公元825年所写的《印度数字算术》中。

把解决某个或某类问题的步骤描述成一定的计算机指令，这些指令序列就构成了"计算机算法"。



# 排序

将一组无序数据按由小到大或者由大到小的顺序进行排列。

比如：输入 8 5 1 7 6 3

输出 1 3 5 6 7 8

## 常用排序算法：

选择排序(Selection Sort)

冒泡排序(Bubble Sort)

插入排序(Insertion Sort)

希尔排序(Shell Sort)

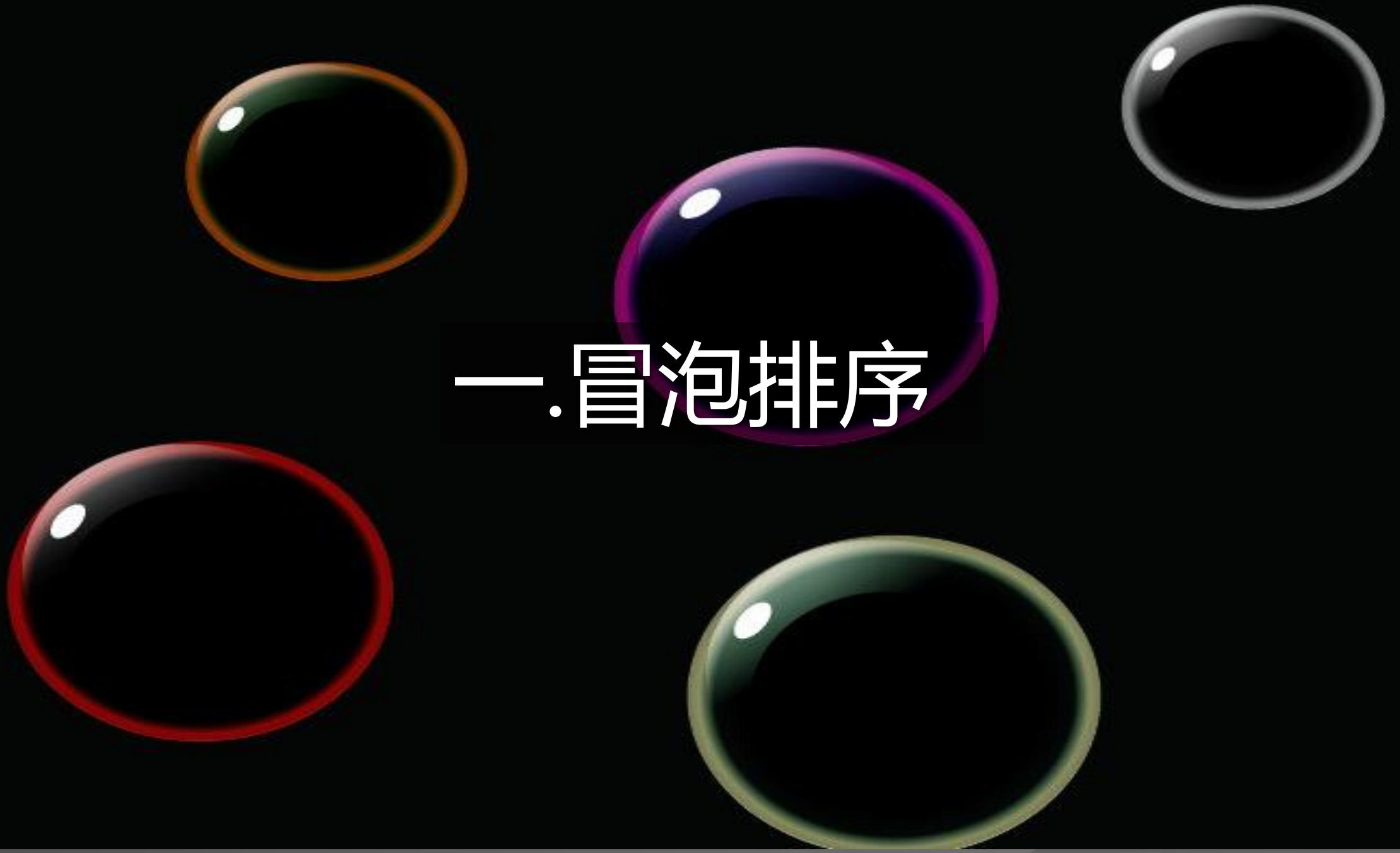
快速排序(Quick Sort)

堆排序(Heap Sort)

归并排序(Merge Sort)

基数排序(XXX Sort)

# 一.冒泡排序



# 冒泡排序

将 29 18 87 56 3 27 按由小到大排序

//总共比较n-1趟

```
for (i=1; i<=n-1; i++)
```

//j为要比较两个的相邻两数的第1个数的下标

```
for (j=1; j<=n-i; j++)
```

//若发现相邻两数反序，则交换

```
if (a[j]>a[j+1])
```

```
{
```

```
    temp=a[j];
```

```
    a[j]=a[j+1];
```

```
    a[j+1]=temp;
```

```
}
```

第1趟

18	29	56	3	27	<u>87</u>
----	----	----	---	----	-----------

第2趟

18	29	3	27	<u>56</u>	87
----	----	---	----	-----------	----

第3趟

18	3	27	<u>29</u>	56	87
----	---	----	-----------	----	----

第4趟

3	18	<u>27</u>	29	56	87
---	----	-----------	----	----	----

第5趟

3	<u>18</u>	27	29	56	87
---	-----------	----	----	----	----

冒泡排序基本思想是:对待排序的数字进行两两比较,如发现两个数字是反序的,则进行交换,直到无反序的记录为止。

```
int a[50001],n,i,j,temp;
int main()
{
    scanf("%d",&n)
    for(i=1;i<=n;i++)scanf("%d",&a[i]);
    for(i=1; i<=n-1; i++)
        for(j=1; j<=n-i; j++)
            if(a[j] > a[j+1])
            {
                temp = a[j];
                a[j] = a[j+1];
                a[j+1] = temp;
            }
    for(i=1; i<=n; i++)printf("%d  ",a[i]);
}
```

# 冒泡排序特点分析


时间复杂度：总共比较  $n(n-1)/2$  次  
时间复杂度为  $O(n^2)$

稳定性：**稳定**

**时间复杂度**简单的说就是程序循环执行的总的次数。

排序算法的**稳定性**是指 值相同的数字在排序前后其**相对位置**是否要发生改变，若要改变就是不稳定，否则就是稳定的。

排序前	15	7	12	7	3
排序后	3	7	7	12	15



**不稳定**

## 二.选择排序



# 选择排序

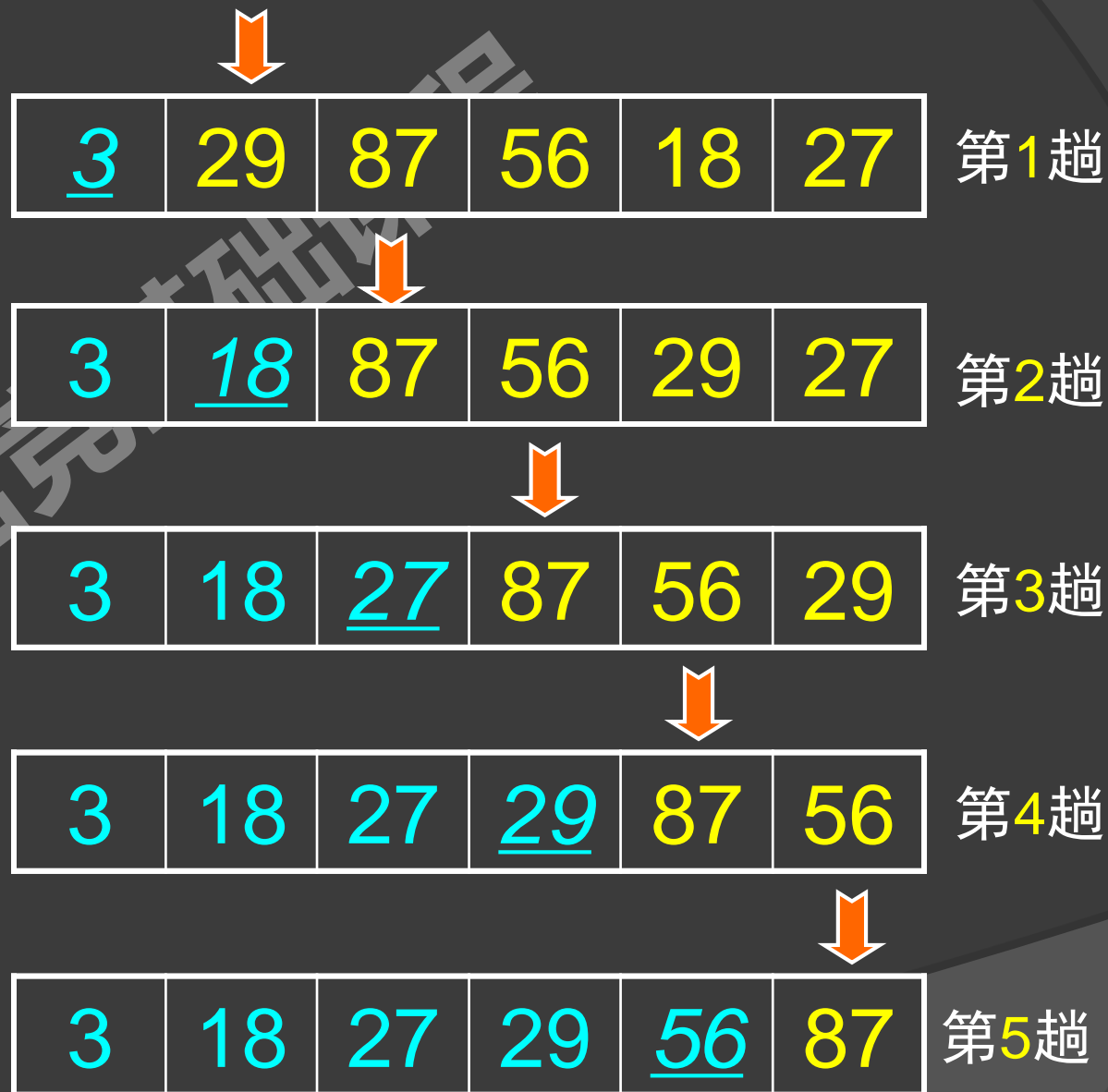
将 29 18 87 56 3 27 按由小到大排序

```
int a[1001];  
int i,j,temp;  
...输入n个数字到a中...
```

//选择排序

```
for(i=1;i<=n-1;i++) //以i号数为基准  
    for(j=i+1;j<=n;j++) //与i后每个数比较  
        if(a[i]>a[j]) //发现反序则交换  
        {  
            temp=a[i];  
            a[i]=a[j];  
            a[j]=temp;  
        }
```

...输出排序结果...



选择排序的基本思想是：

对待排序的 $n$ 个数据进行 $n-1$ 遍的处理，第1遍处理是将 $a[2..n]$ 中最小者与 $a[1]$ 交换位置，第2遍处理是将 $a[3..n]$ 中最小者与 $a[2]$ 交换位置，.....，第 $i$ 遍处理是将 $a[i+1..n]$ 中最小者与 $a[i]$ 交换位置。这样，经过 $i$ 遍处理之后，前 $i$ 个记录的位置就已经按从小到大的顺序排列好了。

```
int a[5001],n,i,j,temp;
int main()
{
    scanf("%d",&n)
    for(i=1; i<=n; i++)scanf("%d",&a[i]);
    for(i=1; i<=n-1; i++)
        for(j=i+1; j<=n; j++)
            if(a[i] > a[j])
            {
                temp = a[i];
                a[i] = a[j];
                a[j] = temp;
            }
    for(i=1;i<=n;i++)printf("%d  ",a[i]);
    return 0;
}
```

# 选择排序特点分析

时间复杂度：总共比较  $n(n-1)/2$  次  
时间复杂度为  $O(n^2)$

稳定性： 不稳定

## 三.插入排序

# 插入排序

将 29 18 87 27 23 56 按由小到大排序

数组a

0	29	18	87	27	23	56
0	1	2	3	4	5	6

下标

程序代码

```
for (i=2; i<=n; i++)  
{  
    a[0]=a[i];  
    j=i-1;  
    while (a[j]>a[0])  
    {  
        a[j+1]=a[j];  
        j--;  
    }  
    a[j+1]=a[0];  
}
```

## 算法实现

在数组中增加元素 $a[0]$ 作为临时空间,把待插入的数放到里面。第 $i$ 趟排序,即 $a[i]$ 的插入过程为:

- ① 保存 $a[0]=a[i]$
- ②  $j=i-1$
- ③ 如果 $a[j]\leq a[0]$  (即待排序的 $a[i]$ ), 则 $a[j+1]=a[0]$ , 完成插入;  
否则, 将 $a[j]$ 后移一个位置:  $A[j+1]=A[j]$ ; 继续执行③

## 程序代码

```
for (i=2; i<=n; i++)  
{  
    a[0]=a[i];  
    j=i-1;  
    while(a[j]>a[0])  
    {  
        a[j+1]=a[j];  
        j--;  
    }  
    a[j+1]=a[0];  
}
```

(1) 稳定性: 稳定

(2) 时间复杂度:  $O(n^2)$

①初始数据正序, 总比较次数:  $n-1$

②初始数据逆序, 总比较次数:  $(n^2+n-1)/2=O(n^2)$

③初始数据无序, 第 $i$ 趟平均比较次数  $(i+1)/2$ , 总次数为:  $(n^2+3n)/4=O(n^2)$

④可见, 原始数据越趋向正序, 比较次数和移动次数越少。

## 四.例题



## 课堂练习：学生成绩(nkoj1003)

某年级有 $n$  ( $n \leq 5000$ ) 个学生，学号1到 $n$ ，现给出这 $n$ 个学生的语文和数学成绩，请按数学成绩的由高到低对这 $n$ 个学生进行排序。数学成绩相同的学生，按语文成绩由高到低排序。

**输入格式(输入文件student.in):**

第一行，一个整数 $n$ ，表示 $n$ 个学生

第二行， $n$ 个空格间隔的整数，表示学号1到 $n$ 的学生的数学成绩

第三行， $n$ 个空格间隔的整数，表示学号1到 $n$ 的学生的语文成绩

**输出格式(输出文件student.out):**

排序后输出 $n$ 行，每行代表一个学生。每行两个数字，分为该生的数学和语文成绩。

**样例输入:**

```
6
67 88 91 88 99 88
80 92 69 70 85 77
```

**样例输出**

```
99 85
91 69
88 92
88 77
88 70
67 80
```

```
int yu[5001],shu[5001],n,i,j,temp;
int main()
{
    scanf("%d",&n);
    for(i=1;i<=n;i++)scanf("%d",&shu[i]);
    for(i=1;i<=n;i++)scanf("%d",&yu[i]);
    for(i=1;i<=n-1;i++)
        for(j=1;j<=n-i;j++)
            if((shu[j]<shu[j+1])||((shu[j]==shu[j+1])&&(yu[j]<yu[j+1])))
            {
                temp=shu[j];
                shu[j]=shu[j+1];
                shu[j+1]=temp;

                temp=yu[j];
                yu[j]=yu[j+1];
                yu[j+1]=temp;
            }
    for(i=1;i<=n;i++)printf("%d  %d\n",shu[i],yu[i]);
}
```

## 程序代码

```
for (i=2; i<=n; i++)  
{  
    shu[0]=shu[i];  
    yu[0]=yu[i];  
    j=i-1;  
    while ( (shu[j]<shu[0]) || (shu[j]==shu[0]) && (yu[j]<yu[0]) )  
    {  
        shu[j+1]=shu[j];  
        yu[j+1]=yu[j];  
        j--;  
    }  
    shu[j+1]=shu[0];  
    yu[j+1]=yu[0];  
}
```