

# 王道考研——数据结构

WWW.CSKAOYAN.COM

## 开篇——数据结构在学什么？

1

数据结构在学什么？

- 如何用程序代码把现实世界的问题信息化
- 如何用计算机高效地处理这些信息从而创造价值



王道考研/CSKAOYAN.COM

2

人类社会的发展，迄今经历了和经历着三个浪潮：第一次浪潮为农业阶段，从约1万年前开始；第二次浪潮为工业阶段，从17世纪末开始；第三次浪潮为正在到来的信息化阶段。

——《第三次浪潮（1980版）》，阿尔文·托夫勒



信息化 →



财富

王道考研/CSKAOYAN.COM

3

人类社会的发展，迄今经历了和经历着三个浪潮：第一次浪潮为农业阶段，从约1万年前开始；第二次浪潮为工业阶段，从17世纪末开始；第三次浪潮为正在到来的信息化阶段。

——《第三次浪潮（1980版）》，阿尔文·托夫勒



信息化 →



排队

王道考研/CSKAOYAN.COM

4

人类社会的发展，迄今经历了和经历着三个浪潮：第一次浪潮为农业阶段，从约1万年前开始；第二次浪潮为工业阶段，从17世纪末开始；第三次浪潮为正在到来的信息化阶段。

——《第三次浪潮(1980版)》,阿尔文·托夫勒

信息化

友谊

王道考研/CSKAOYAN.COM

5

学完C语言，如何在计算机中表示这些信息？？？

浮点型变量(float)

数组？还是...

王道考研/CSKAOYAN.COM

6



人类社会的发展，迄今经历了和经历着三个浪潮：第一次浪潮为农业阶段，从约1万年前开始；第二次浪潮为工业阶段，从17世纪末开始；第三次浪潮为正在到来的信息化阶段。

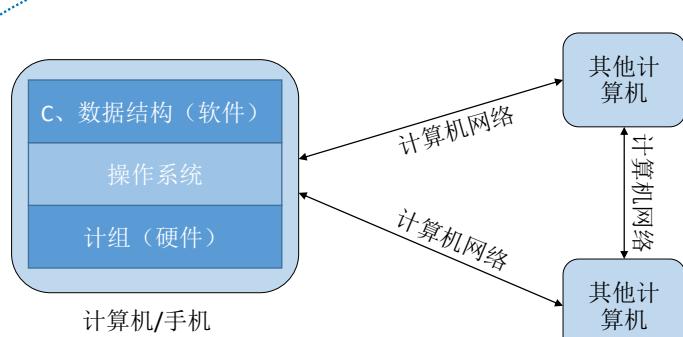
——《第三次浪潮（1980版）》，阿尔文·托夫勒

数据结构在学什么？

- 如何用程序代码把现实世界的问题信息化
- 如何用计算机高效地处理这些信息从而创造价值

王道考研/CSKAOYAN.COM

7



王道考研/CSKAOYAN.COM

8

“唯一可以确定的是，明天会使我们所有人大吃一惊。” —— 阿尔文·托夫勒  
( The sole certainty is that tomorrow will surprise us all. )

人类文明 → 农业革命 → 工业革命 → 信息革命

学会农耕，从“动物”  
到“人类”

出现枪炮与机械，  
导致古文明灭亡

高度信息化的世  
界，导致？？？



王道考研/CSKAOYAN.COM

# 王道考研——数据结构

WWW.CSKAOYAN.COM

## 第一章 绪论

1

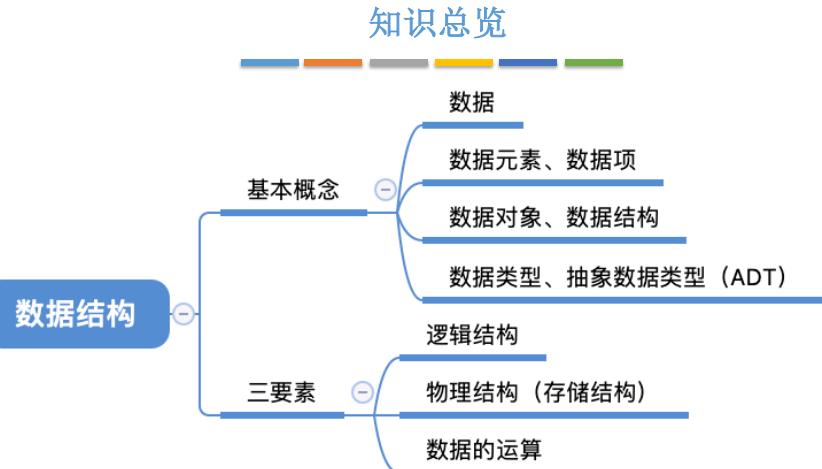
本节内容

数据结构

基本概念

王道考研/CSKAOYAN.COM

2



#### 学习建议:

- 概念多，比较无聊。抓大放小，重要的是形成框架，不必纠结于细节概念  
2. 视频结尾会把最重要的概念串一遍，勿慌

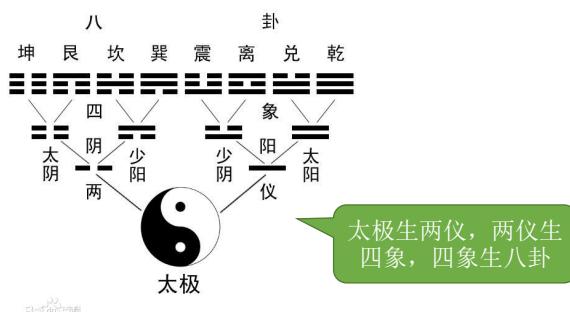
王道考研/CSKAOYAN.COM

3

## 什么是数据？

数据：

数据是信息的载体，是描述客观事物属性的数、字符及所有能输入到计算机中并被计算机程序识别和处理的符号的集合。数据是计算机程序加工的原料。



二进制0和1

A close-up photograph of a computer monitor displaying binary code. The screen shows a grid of black and white pixels representing binary digits (0s and 1s). A person's hand is visible at the bottom right corner of the screen, pointing towards the binary code.

王道考研/CSKAOYAN.COM

4

## 数据元素、数据项

**数据元素、数据项：**

**数据元素**是数据的基本单位，通常作为一个整体进行考虑和处理。  
一个数据元素可由若干**数据项**组成，数据项是构成数据元素的不可分割的最小单位。

王道考研/CSKAOYAN.COM

5

## 数据元素、数据项

**数据元素、数据项：**

**数据元素**是数据的基本单位，通常作为一个整体进行考虑和处理。  
一个数据元素可由若干**数据项**组成，数据项是构成数据元素的不可分割的最小单位。

王道考研/CSKAOYAN.COM

6

## 数据结构、数据对象

结构——各个元素之间的关系

王道考研/CSKAOYAN.COM

7

## 数据结构、数据对象

**数据结构、数据对象：**

**数据结构**是相互之间存在一种或多种特定**关系**的数据元素的集合。

**数据对象**是具有**相同性质**的数据元素的集合，是数据的一个子集。

一波顾客      另一波顾客

号数
取号时间
就餐人数

号数
取号时间
就餐人数

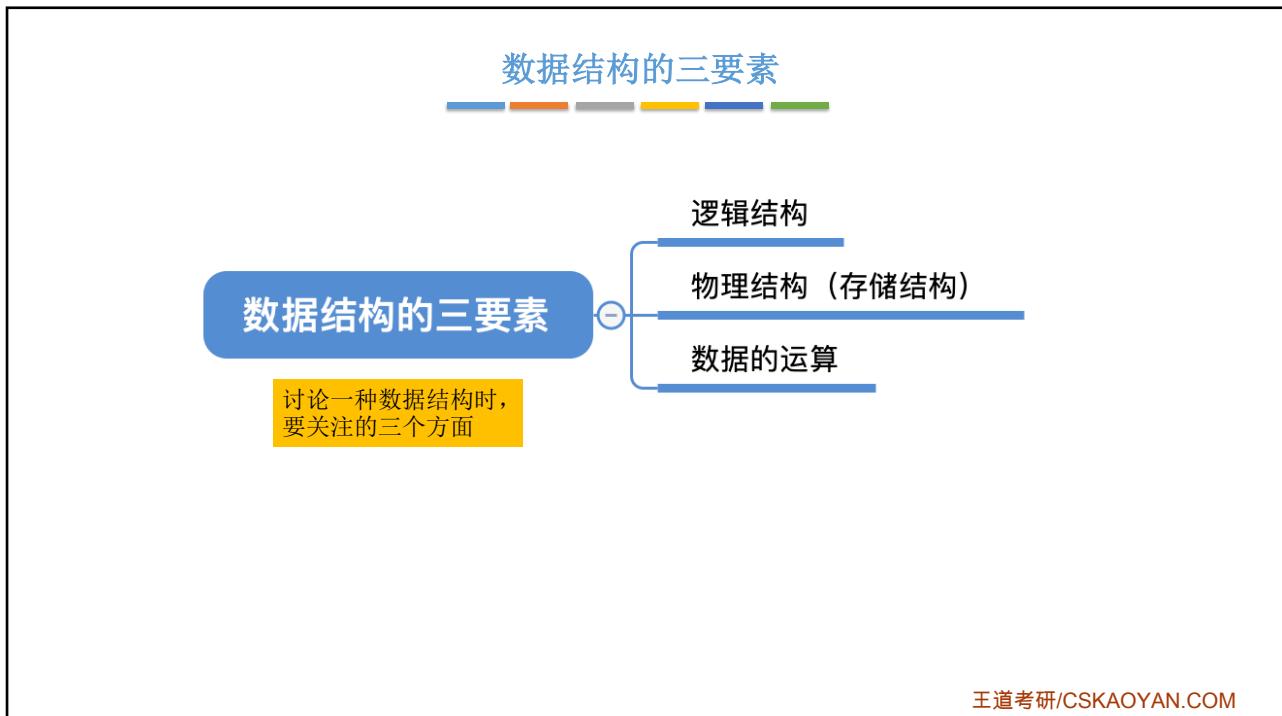
海底捞

数据结构：某个特定门店的排队顾客信息和它们之间的**关系**

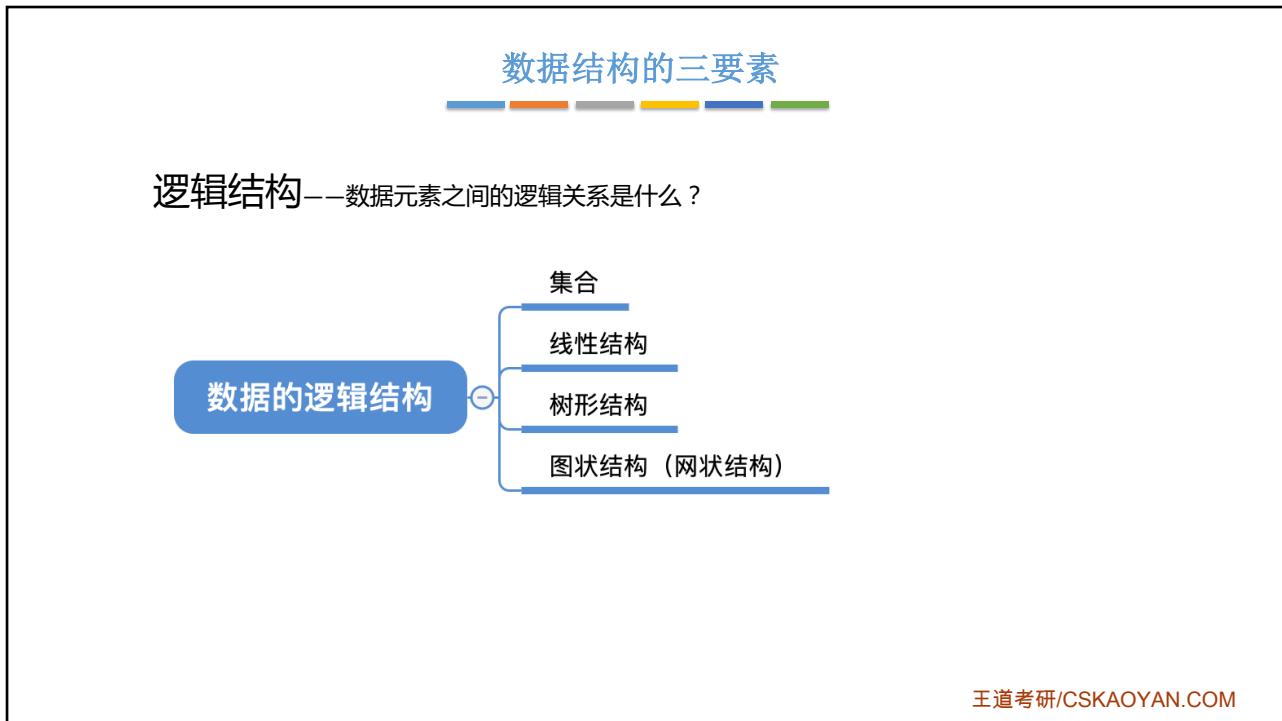
数据对象：全国所有门店的排队顾客信息

王道考研/CSKAOYAN.COM

8



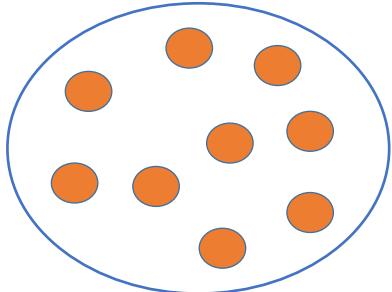
9



10

## 数据结构的三要素

数据的逻辑结构——数据元素之间的逻辑关系是什么？



**集合**

各个元素同属一个集合，别无其他关系



王道考研/CSKAOYAN.COM

11

## 数据结构的三要素

数据的逻辑结构——数据元素之间的逻辑关系是什么？



**线性结构**

数据元素之间是一对一的关系。  
除了第一个元素，所有元素都有唯一前驱；  
除了最后一个元素，所有元素都有唯一后继




995号 → 996号 → 997号 → 998号

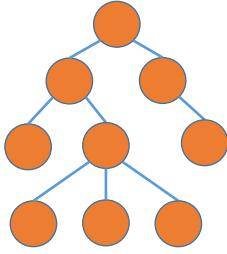
**海底捞**

王道考研/CSKAOYAN.COM

12

## 数据结构的三要素

数据的逻辑结构——数据元素之间的逻辑关系是什么？



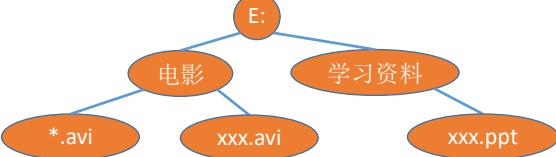
**树形结构**

数据元素之间是一对多的关系



数据结构

- 基本概念
- 数据、数据元素、数据项
- 数据对象、数据结构
- 数据类型、抽象数据类型 (ADT)
- 逻辑结构
- 三要素
- 物理结构 (存储结构)
- 数据的运算



E:

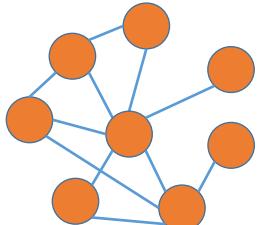
- 电影
- \*.avi
- xxx.avi
- 学习资料
- xxx.ppt

王道考研/CSKAOYAN.COM

13

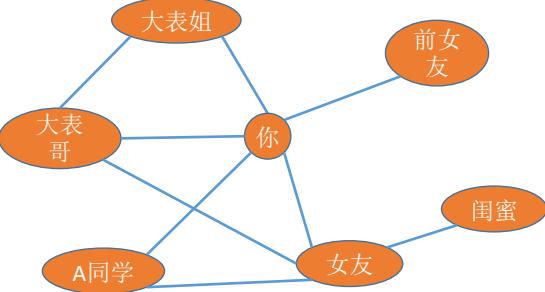
## 数据结构的三要素

数据的逻辑结构——数据元素之间的逻辑关系是什么？



**图结构**

数据元素之间是多对多的关系



微信好友

王道考研/CSKAOYAN.COM

14

## 数据结构的三要素

**数据的逻辑结构**——数据元素之间的逻辑关系是什么？

The diagram shows four examples of data structures:

- 集合 (Collection)**: Seven orange circles enclosed in a blue oval.
- 线性结构 (Linear Structure)**: Five orange circles connected by a single horizontal line, labeled "第二、三章" (Chapters 2, 3).
- 树形结构 (Tree Structure)**: A tree with one root node and three children, which each have two children of their own, labeled "第四章" (Chapter 4).
- 图结构 (Graph Structure)**: A network of seven orange circles connected by various lines, labeled "第五章" (Chapter 5).

王道考研/CSKAOYAN.COM

15

## 数据结构的三要素

**数据的物理结构 (存储结构)**——如何用计算机表示数据元素的逻辑关系？

The diagram illustrates the classification of storage structures:

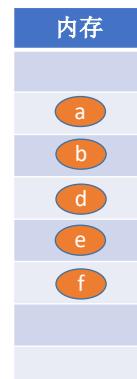
- 数据的存储结构 (Storage Structure)** (highlighted in blue)
- Four categories branching from it:
  - 顺序存储 (Sequential Storage)
  - 链式存储 (Linked Storage)
  - 索引存储 (Indexed Storage)
  - 散列存储 (Hash Storage)

王道考研/CSKAOYAN.COM

16

## 数据结构的三要素

数据的物理结构（存储结构）——如何用计算机表示数据元素的逻辑关系？



顺序存储。把逻辑上相邻的元素存储在物理位置上也相邻的存储单元中，元素之间的关系由存储单元的邻接关系来体现。

王道考研/CSKAOYAN.COM

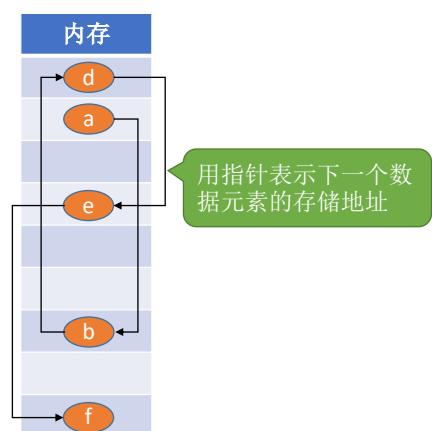
17

## 数据结构的三要素

数据的物理结构（存储结构）——如何用计算机表示数据元素的逻辑关系？



链式存储。逻辑上相邻的元素在物理位置上可以不相邻，借助指示元素存储地址的指针来表示元素之间的逻辑关系。



王道考研/CSKAOYAN.COM

18

## 数据结构的三要素

数据的物理结构（存储结构）——各个数据元素在内存中如何存储

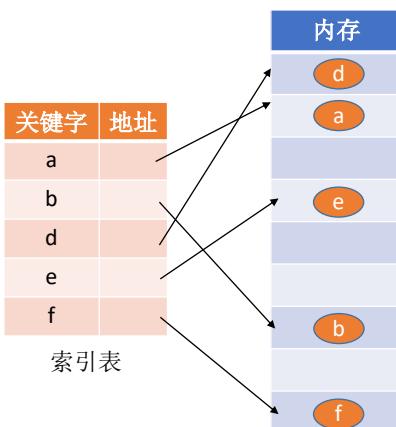


线性结构

逻辑结构

关键字	地址
a	
b	
d	
e	
f	

索引表



索引存储。在存储元素信息的同时，还建立附加的索引表。索引表中的每项称为索引项，索引项的一般形式是（关键字，地址）

王道考研/CSKAOYAN.COM

19

## 数据结构的三要素

数据的物理结构（存储结构）——各个数据元素在内存中如何存储



线性结构

逻辑结构



这真是一门  
让人头秃的科目



第六章，散列表

放轻松 你可以的

散列存储。根据元素的关键字直接计算出该元素的存储地址，又称哈希（Hash）存储

王道考研/CSKAOYAN.COM

20

## 数据结构的三要素

数据的物理结构（存储结构）——如何用计算机表示数据元素的逻辑关系？

王道考研/CSKAOYAN.COM

21

## 数据结构的三要素

**来~放轻松~**

绪论部分只需要理解两点：

1. 若采用**顺序存储**，则各个数据元素在物理上必须是**连续的**；若采用**非顺序存储**，则各个数据元素在物理上可以是**离散的**。
2. 数据的**存储结构**会影响**存储空间分配的方便程度** Eg: 有人想插队
3. 数据的**存储结构**会影响对**数据运算的速度** Eg: 想找到第三个人



顺序存储



非顺序存储（离散存储）

王道考研/CSKAOYAN.COM

22

## 数据结构的三要素

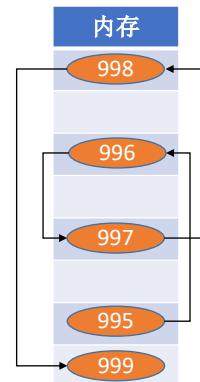
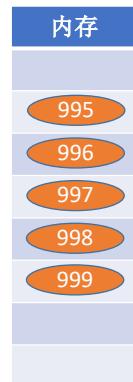
**数据的运算**——施加在数据上的运算包括运算的定义和实现。运算的定义是针对逻辑结构的，指出运算的功能；运算的实现是针对存储结构的，指出运算的具体操作步骤。



逻辑结构——线性结构（队列）

结合现实需求**定义**队列这种逻辑结构的运算：

- ①队头元素出队；
  - ②新元素入队；
  - ③输出队列长度；
- .....



王道考研/CSKAOYAN.COM

23

## 数据结构的三要素

### 数据结构的三要素

讨论一种数据结构时，要关注的三个方面

逻辑结构

物理结构（存储结构）

数据的运算

王道考研/CSKAOYAN.COM

24

## 数据类型、抽象数据类型

### 数据类型、抽象数据类型：

**数据类型**是一个值的集合和定义在此集合上的一组操作的总称。

- 1) 原子类型。其值不可再分的数据类型。
- 2) 结构类型。其值可以再分解为若干成分（分量）的数据类型。

#### bool 类型

值的范围: true、false  
可进行操作: 与、或、非...

#### int 类型

值的范围: -2147483648 ~ 2147483647  
可进行操作: 加、减、乘、除、模运算...

```
struct Customer{
    int num; //号数
    int people; //人数
    .... //其他必要的信息
};
```

定义一个具体的结构类型，表示排队顾客信息。  
根据具体业务需求来确定值的范围，可进行的操作

值的范围: num (1~9999) 、 people (1~12)  
可进行操作: 如“拼桌”运算，把人数相加合并

王道考研/CSKAOYAN.COM

25

## 数据类型、抽象数据类型

### 数据类型、抽象数据类型：

**数据类型**是一个值的集合和定义在此集合上的一组操作的总称。

- 1) 原子类型。其值不可再分的数据类型。
- 2) 结构类型。其值可以再分解为若干成分（分量）的数据类型。

**抽象数据类型** (Abstract Data Type, **ADT**) 是抽象数据组织及与之相关的操作。

### 数据结构的三要素

#### 逻辑结构

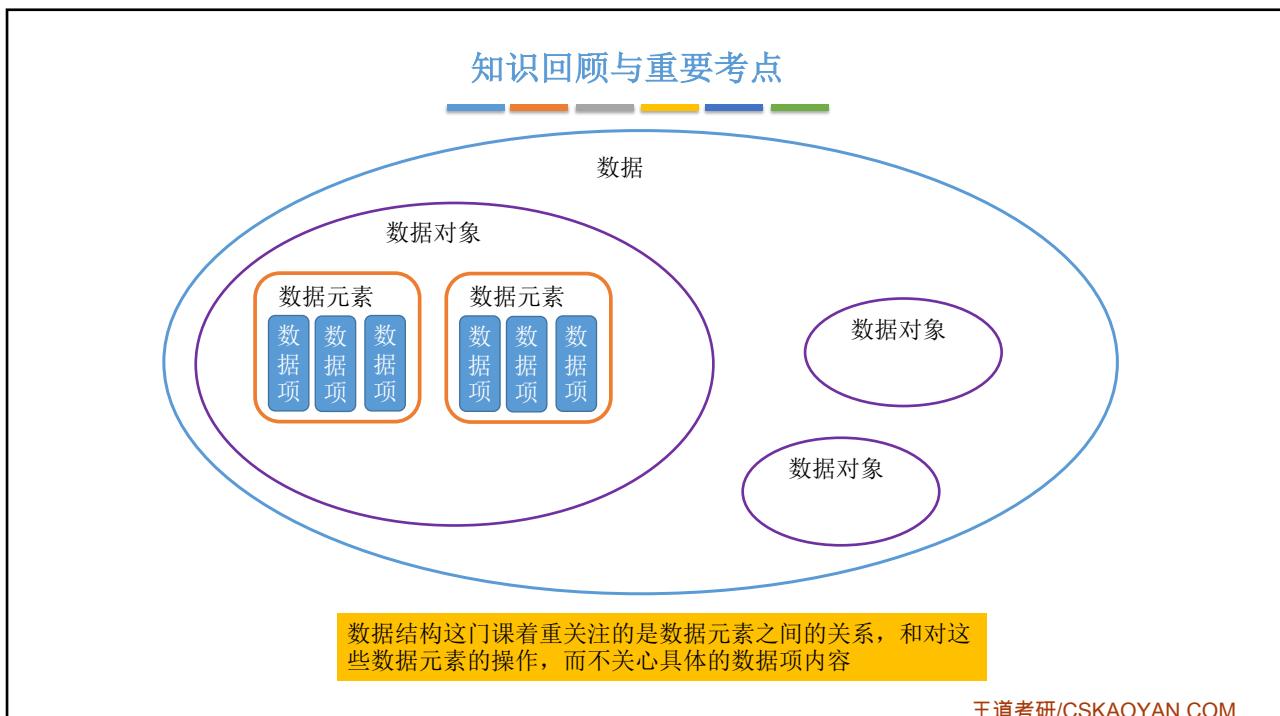
ADT 用数学化的语言定义数据的逻辑结构、定义运算。  
与具体的实现无关。

#### 物理结构 (存储结构)

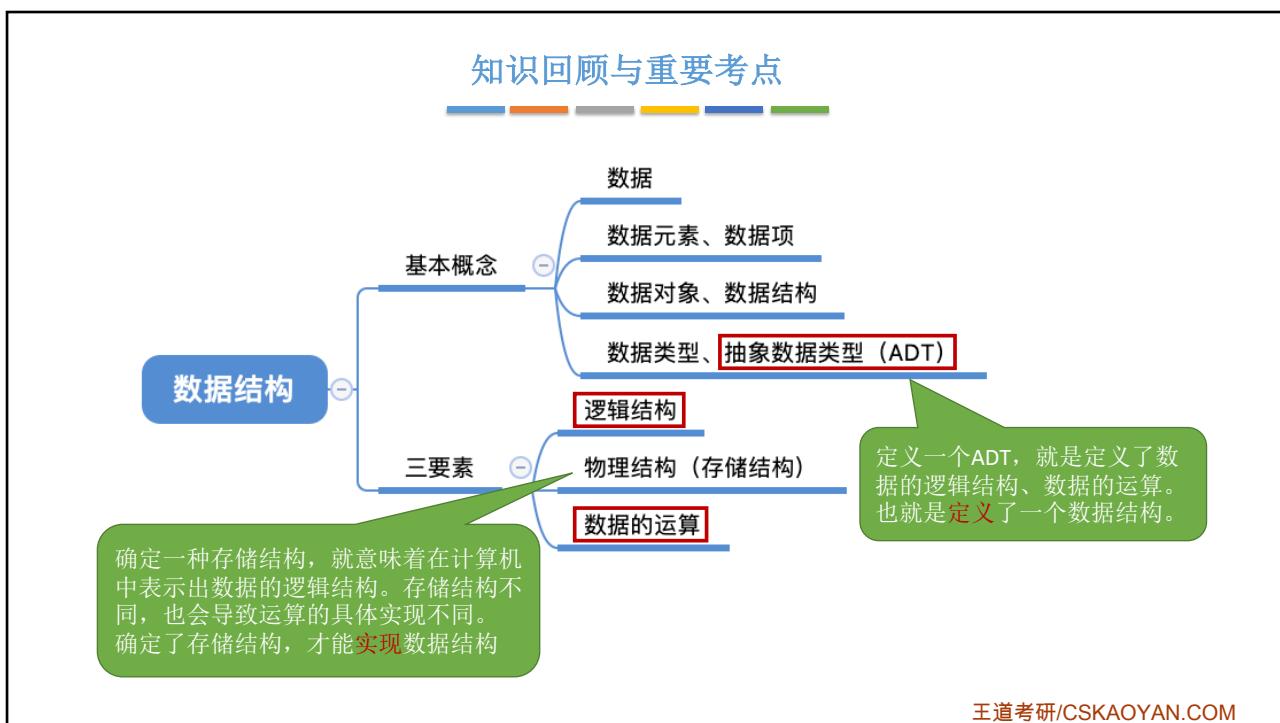
#### 数据的运算

王道考研/CSKAOYAN.COM

26



27



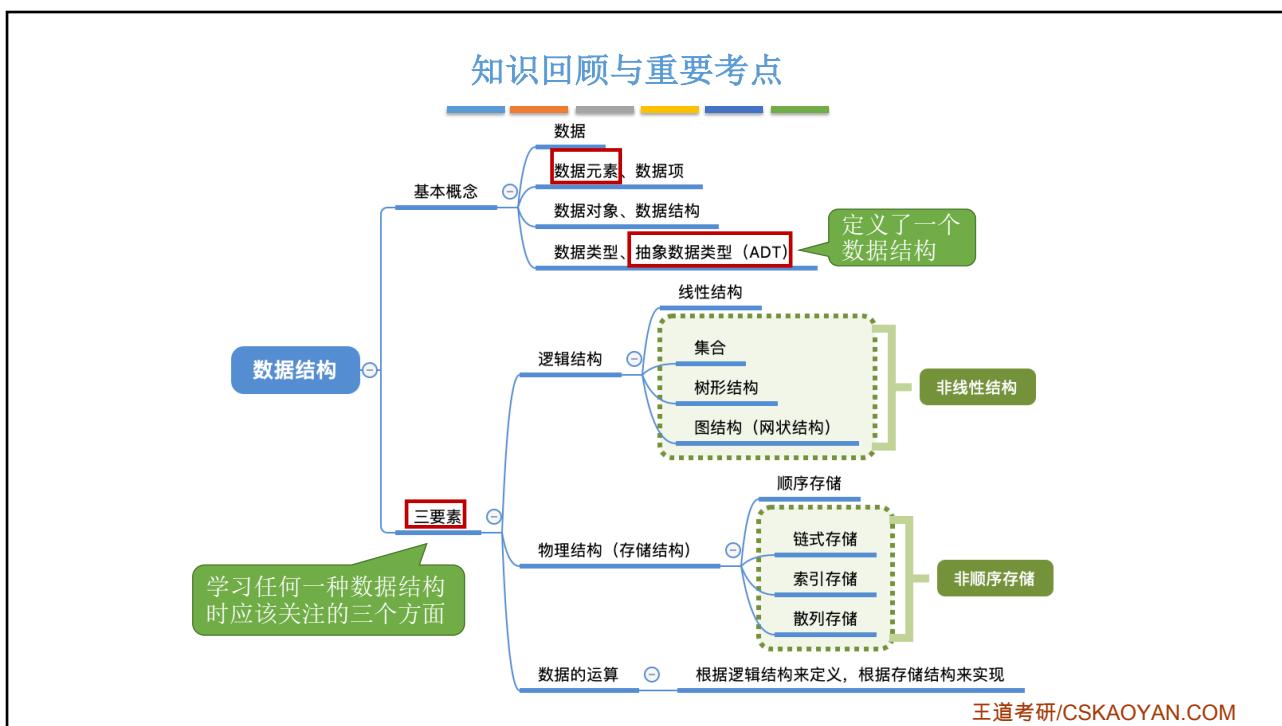
28

## 知识回顾与重要考点

<b>第2章 线性表</b> ..... 12 <sup>④</sup> 2.1 线性表的定义和基本操作 ..... 12 <sup>④</sup> 2.1.1 线性表的定义 ..... 12 <sup>④</sup> 2.1.2 线性表的基本操作 ..... 13 <sup>④</sup> 2.1.3 本节试题精选 ..... 13 <sup>④</sup> 2.1.4 答案与解析 ..... 13 <sup>④</sup> 2.2 线性表的顺序表示 ..... 14 <sup>④</sup> 2.2.1 顺序表的定义 ..... 14 <sup>④</sup> 2.2.2 顺序表上基本操作↓ 的实现 ..... 15 <sup>④</sup> 2.2.3 本节试题精选 ..... 17 <sup>④</sup> 2.2.4 答案与解析 ..... 19 <sup>④</sup> 2.3 线性表的链式表示 ..... 27 <sup>④</sup> 2.3.1 单链表的定义 ..... 27 <sup>④</sup> 2.3.2 单链表上基本操作↓ 的实现 ..... 28 <sup>④</sup> 2.3.3 双链表 ..... 31 <sup>④</sup> 2.3.4 循环链表 ..... 33 <sup>④</sup> 2.3.5 静态链表 ..... 33 <sup>④</sup> 2.3.6 顺序表和链表的比较 ..... 34 <sup>④</sup> 2.3.7 本节试题精选 ..... 35 <sup>④</sup>	<b>第3章 栈和队列</b> ..... 59 <sup>④</sup> 3.1 栈 ..... 59 <sup>④</sup> 3.1.1 栈的基本概念 ..... 59 <sup>④</sup> 3.1.2 栈的顺序存储结构 ..... 60 <sup>④</sup> 3.1.3 栈的链式存储结构 ..... 62 <sup>④</sup> 3.1.4 本节试题精选 ..... 62 <sup>④</sup> 3.1.5 答案与解析 ..... 66 <sup>④</sup> 3.2 队列 ..... 73 <sup>④</sup> 3.2.1 队列的基本概念 ..... 73 <sup>④</sup> 3.2.2 队列的顺序存储结构 ..... 73 <sup>④</sup> 3.2.3 队列的链式存储结构 ..... 75 <sup>④</sup> 3.2.4 双端队列 ..... 77 <sup>④</sup> 3.2.5 本节试题精选 ..... 78 <sup>④</sup> 3.2.6 答案与解析 ..... 81 <sup>④</sup> 3.3 栈和队列的应用 ..... 85 <sup>④</sup> 3.3.1 栈在括号匹配中的应用 ..... 85 <sup>④</sup> 3.3.2 栈在表达式求值中↓ 的应用 ..... 85 <sup>④</sup> 3.3.3 栈在递归中的应用 ..... 86 <sup>④</sup> 3.3.4 队列在层次遍历中↓ 的应用 ..... 87 <sup>④</sup> 3.3.5 队列在计算机系统中↓ 的应用 ..... 88 <sup>④</sup> 3.3.6 本节试题精选 ..... 88 <sup>④</sup> 3.3.7 答案与解析 ..... 90 <sup>④</sup>
---	--

王道考研/CSKAOYAN.COM

29



30

本节内容

# 算法

## 基本概念

王道考研/CSKAOYAN.COM

1

知识总览

```
graph LR; A[算法的基本概念] --> B[什么是算法]; A --> C[算法的五个特性]; A --> D["好"算法的特质]
```

算法的基本概念

什么是算法

算法的五个特性

"好"算法的特质

王道考研/CSKAOYAN.COM

2

## 什么是算法？



### 程序 = 数据结构 + 算法

如何把现实世界的问题信息化，将信息存进计算机。同时还要实现对数据结构的基本操作

如何处理这些信息，以解决实际问题


海  
底  
捞


海底捞排队系统的**数据结构**——队列  
已经实现的基本操作：

- ①队头元素出队；
- ②新元素入队；
- ③输出队列长度；
- ④交换第*i*号和第*j*号的位置

.....

要解决的现实问题——带小孩的顾客优先就餐  
设计一个**算法**：

- 执行基本操作②（新顾客取号）
- 依次对比前一桌信息，如果前一桌没有小孩，则使用基本操作④交换位置。

王道考研/CSKAOYAN.COM

3

## 算法的特性



**算法必须具备的特性**

用有限步骤解决某个特定的问题

海底捞的排队系统就是一个程序，可以永不停歇

设计一个算法，解决一个特定的问题——带小孩的顾客优先就餐



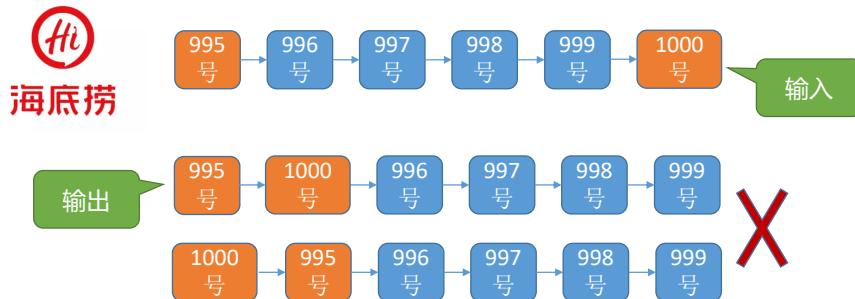
王道考研/CSKAOYAN.COM

4

## 算法的特性

**确定性。** 算法中每条指令必须有确切的含义，对于**相同的输入**只能得出**相同的输出**。

Eg: 设计一个算法，解决一个特定的问题——带小孩的顾客优先就餐



**可行性。** 算法中描述的操作都可以通过已经实现的**基本运算执行有限次**来实现。

**输入。** 一个算法**有零个或多个输入**，这些输入取自于某个特定的对象的集合。

**输出。** 一个算法**有一个或多个输出**，这些输出是与输入有着某种特定关系的量。

王道考研/CSKAOYAN.COM

5

## “好”算法的特质

设计算法时要**尽量追求**的目标

1) 正确性。算法应能够正确地解决求解问题。



王道考研/CSKAOYAN.COM

6

## “好”算法的特质

2) 可读性。算法应具有良好的可读性，以帮助人们理解。

要解决的现实问题——带小孩的顾客优先就餐  
设计一个**算法**：

- 执行基本操作②（新顾客取号）
- 依次对比前一桌信息，如果前一桌没有小孩，则使用基本操作④交换位置。



注：算法可以用伪代码描述，甚至用文字描述，  
重要的是要“无歧义”地描述出解决问题的步骤

王道考研/CSKAOYAN.COM

7

## “好”算法的特质

3) 健壮性。输入非法数据时，算法能适当地做出反应或进行处理，而不会产生莫名其妙的输出结果。



王道考研/CSKAOYAN.COM

8

## “好”算法的特质

4) 高效率与低存储量需求

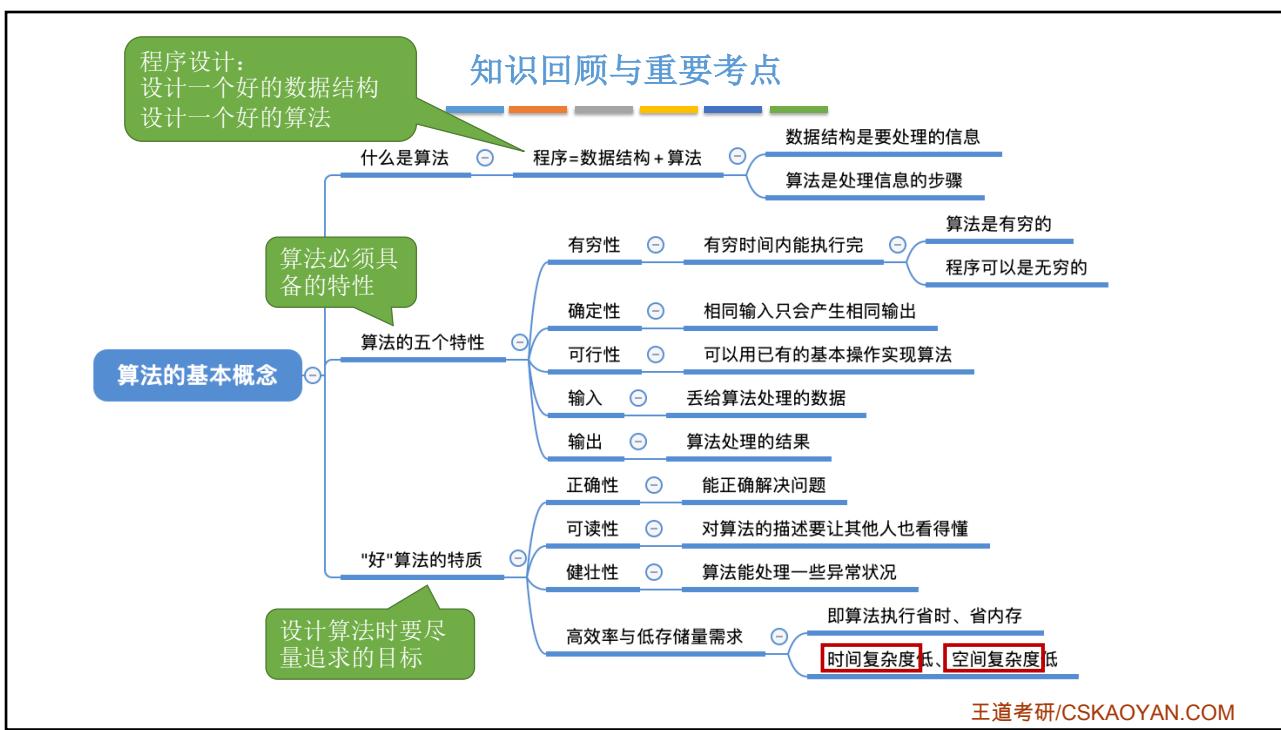
执行速度快。  
时间复杂度低

不费内存。  
空间复杂度低

算法一：从后往前对比  
算法二：从前往后对比

王道考研/CSKAOYAN.COM

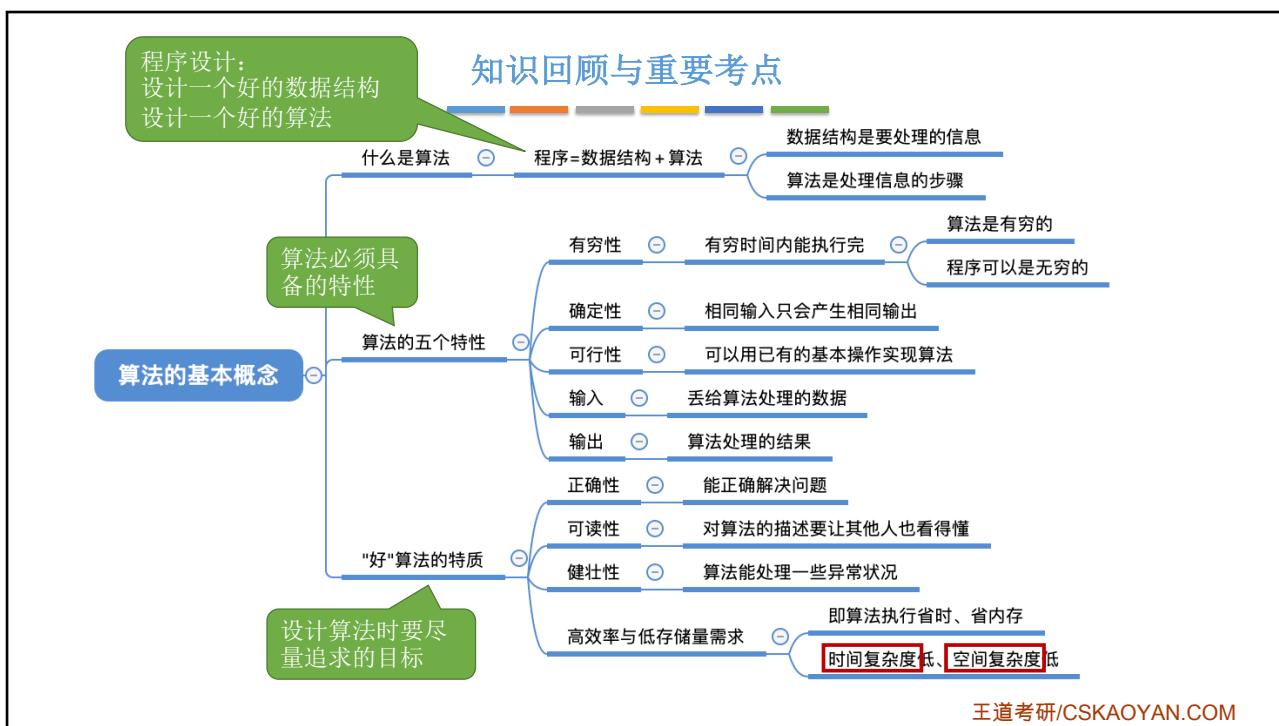
9



10



11



12

本节内容

算法

效率的度量

王道考研/CSKAOYAN.COM

1

知识总览

算法效率的度量

时间复杂度

空间复杂度

王道考研/CSKAOYAN.COM

2

## 如何评估算法时间开销？



让算法先运行，**事后统计**运行时间？



存在什么问题？

- 和机器性能有关，如：超级计算机 v.s. 单片机
- 和编程语言有关，越高级的语言执行效率越低
- 和编译程序产生的机器指令质量有关
- 有些算法是不能事后再统计的，如：导弹控制算法

能否排除与算法本身无关的外界因素
能否事先估计？



**算法时间复杂度**

事前预估算法**时间开销** $T(n)$ 与问题规模  $n$  的关系（ $T$  表示“time”）

王道考研/CSKAOYAN.COM

3

## 算法的时间复杂度



用算法表白——“爱你 $n$ 遍”

王道考研/CSKAOYAN.COM

4

## 算法的时间复杂度

```
//算法1—逐步递增型爱你
void loveYou(int n) { //n 为问题规模
    ① int i=1; //爱你的程度
    ② while(i<=n){
        ③     i++; //每次+1
        ④     printf("I Love You %d\n", i);
    }
    ⑤ printf("I Love You More Than %d\n", n);
}
```

语句频度:

①	--1次
②	--3001次
③④	--3000次
⑤	--1次

$T(3000) = 1 + 3001 + 2*3000 + 1$   
时间开销与问题规模 n 的关系:  
 $T(n)=3n+3$

```
int main(){
    loveYou(3000);
}
```

I Love You 2994  
I Love You 2995  
I Love You 2996  
I Love You 2997  
I Love You 2998  
I Love You 2999  
I Love You 3000  
I Love You 3001  
I Love You More Than 3000



思考中.....

问题1: 是否可以忽略表达式某些部分?

问题2: 如果有好几千行代码,按这种方法需要一行一行数?

王道考研/CSKAOYAN.COM

5

## 算法的时间复杂度

全称: 演进时间复杂度



思考中.....

问题1: 是否可以忽略表达式某些部分?

当问题规模 n 足够大时...

时间开销与问题规模 n 的关系:

$$T_1(n)=3n+3$$

$$T_2(n)=n^2+3n+1000$$

$$T_3(n)=n^3+n^2+9999999$$

若  $n=3000$ , 则

$3n = 9000$	$V.S.$	$T_1(n) = 9003$
$n^2 = 9,000,000$	$V.S.$	$T_2(n) = 9,010,000$
$n^3 = 27,000,000,000$	$V.S.$	$T_3(n) = 27,018,999,999$

大O表示“同阶”, 同等数量级。即: 当  $n \rightarrow \infty$  时, 二者之比为常数

$T_1(n) = O(n)$   
 $T_2(n) = O(n^2)$   
 $T_3(n) = O(n^3)$

简化

结论1: 可以只考虑阶数高的部分

当  $n=3000$  时,  $9999n = 29,997,000$  远小于  $n^3 = 27,018,999,999$   
当  $n=1000000$  时,  $9999n = 9,999,000,000$  远小于  $n^2 = 1,000,000,000,000$

结论2: 问题规模足够大时, 常数项系数也可以忽略

王道考研/CSKAOYAN.COM

6

## 算法的时间复杂度



问题1：是否可以忽略表达式某些部分？

当问题规模  $n$  足够大时...

时间开销与问题规模  $n$  的关系：

$$T_1(n) = 3n + 3$$

$$T_2(n) = n^2 + 3n + 1000$$

$$T_3(n) = n^3 + n^2 + 9999999$$

**简化**

a) 加法规则  
 $T(n) = T_1(n) + T_2(n) = O(f(n)) + O(g(n)) = O(\max(f(n), g(n)))$  多项相加，只保留最高阶的项，且系数变为1

b) 乘法规则  
 $T(n) = T_1(n) \times T_2(n) = O(f(n)) \times O(g(n)) = O(f(n) \times g(n))$  多项相乘，都保留  
Eg:  $T_3(n) = n^3 + n^2 \log_2 n$   
 $= O(n^3) + O(n^2 \log_2 n)$   
 $= ? ? ?$

王道考研/CSKAOYAN.COM

7

## 算法的时间复杂度

$O(1) < O(\log_2 n) < O(n) < O(n \log_2 n) < O(n^2) < O(n^3) < O(2^n) < O(n!) < O(n^n)$



问题：两个算法的时间复杂度分别如下，哪个的阶数更高（时间复杂度更高）？

$T_1(n) = O(n)$   
 $T_2(n) = O(\log_2 n)$

Eg:  $T_3(n) = n^3 + n^2 \log_2 n$   
 $= O(n^3) + O(n^2 \log_2 n)$   
 $= O(n^2 n) + O(n^2 \log_2 n)$

$\lim_{n \rightarrow \infty} \frac{\log_2 n}{n}$  洛必达  $\lim_{n \rightarrow \infty} \frac{1}{n \ln 2} = 0$       当  $n \rightarrow \infty$  时， $n$  比  $\log_2 n$  变大的速度快很多

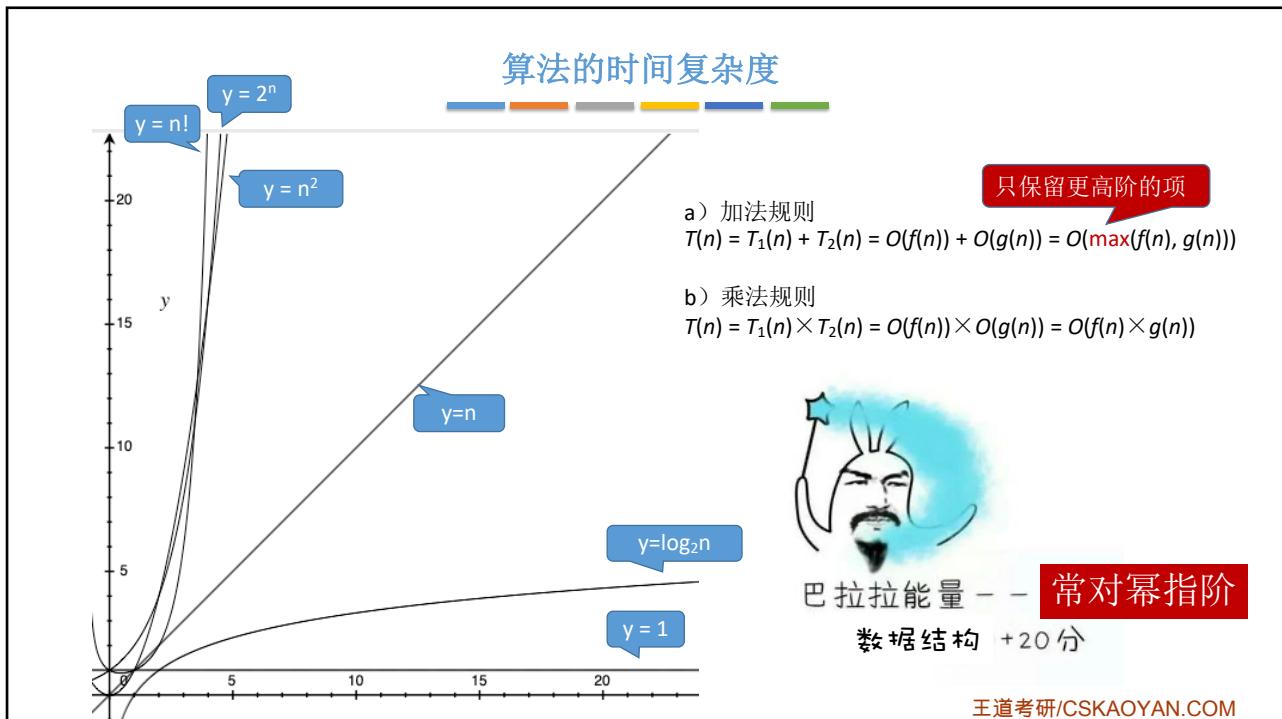
$\lim_{n \rightarrow \infty} \frac{n^2}{2^n}$  洛必达  $\lim_{n \rightarrow \infty} \frac{2n}{2^n \ln 2}$  洛必达  $\lim_{n \rightarrow \infty} \frac{2}{2^n \ln^2 2} = 0$       当  $n \rightarrow \infty$  时， $2^n$  比  $n^2$  变大的速度快很多

别紧张 放轻松 😊

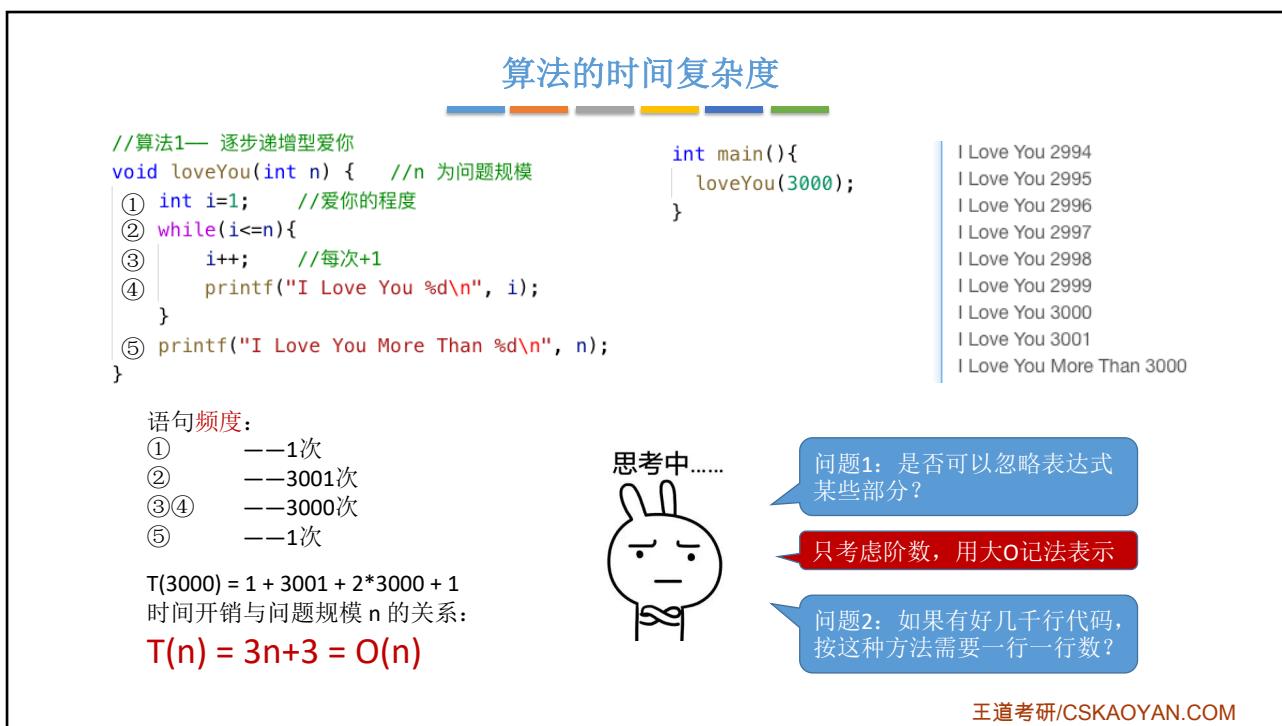


王道考研/CSKAOYAN.COM

8



9



10

## 算法的时间复杂度

`//算法1—逐步递增型爱你  
void loveYou(int n) { //n 为问题规模  
 ① int i=1; //爱你的程度  
 ② while(i<=n){  
 ③ i++; //每次+1  
 ④ printf("I Love You %d\n", i);  
 }  
 ⑤ printf("I Love You More Than %d\n", n);  
}`

语句频度：  
 ① ——1次  
 ② ——3001次  
 ③④ ——3000次  
 ⑤ ——1次

$T(3000) = 1 + 3001 + 2*3000 + 1$   
 时间开销与问题规模 n 的关系：  
 $T(n) = 3n+3 = O(n)$

此处插入1000行顺序执行的代码

结论1：顺序执行的代码只会影响常数项，可以忽略  
 结论2：只需挑循环中的一个基本操作分析它的执行次数与 n 的关系即可

  
**机智如我**

王道考研/CSKAOYAN.COM

11

## 算法的时间复杂度

`//算法2—嵌套循环型爱你  
void loveYou(int n) { //n 为问题规模  
 int i=1; //爱你的程度  
 while(i<=n){  
 ① i++; //外层循环执行n次  
 ② printf("I Love You %d\n", i);  
 ③ for (int j=1; j<=n; j++){  
 ④ printf("I am Iron Man\n");  
 }  
 }  
 ⑤ printf("I Love You More Than %d\n", n);  
}`

时间开销与问题规模 n 的关系：  
 $T(n) = O(n) + O(n^2) = O(n^2)$

外层循环执行n次  
 嵌套两层循环  
 内层循环共执行 $n^2$ 次

结论1：顺序执行的代码只会影响常数项，可以忽略  
 结论2：只需挑循环中的一个基本操作分析它的执行次数与 n 的关系即可  
 结论3：如果有多层次嵌套循环，只需关注最深层循环循环了几次

  
**机智如我**

王道考研/CSKAOYAN.COM

12

## 算法的时间复杂度

```
//算法1—逐步递增型爱你
void loveYou(int n) { //n 为问题规模
    ① int i=1; //爱你的程度
    ② while(i<=n){
        ③     i++; //每次+1
        ④     printf("I Love You %d\n", i);
    }
    ⑤ printf("I Love You More Than %d\n", n);
}
```

语句频度：

- ① ——1次
- ② ——3001次
- ③④ ——3000次
- ⑤ ——1次

$$T(3000) = 1 + 3001 + 2*3000 + 1$$

时间开销与问题规模 n 的关系：

$$T(n) = 3n+3 = O(n)$$

```
int main(){
    loveYou(3000);
}
```

```
I Love You 2994
I Love You 2995
I Love You 2996
I Love You 2997
I Love You 2998
I Love You 2999
I Love You 3000
I Love You 3001
I Love You More Than 3000
```



思考中.....

问题1：是否可以忽略表达式某些部分？

只考虑阶数，用大O记法表示

问题2：如果有好几千行代码，按这种方法需要一行一行数？

只需考虑最深层循环的循环次数与 n 的关系

王道考研/CSKAOYAN.COM

13

## 小练习1

```
//算法3—指数递增型爱你
void loveYou(int n) { //n 为问题规模
    int i=1; //爱你的程度
    while(i<=n){
        i=i*2; //每次翻倍
        printf("I Love You %d\n", i);
    }
    printf("I Love You More Than %d\n", n);
}
```

```
I Love You 32
I Love You 64
I Love You 128
I Love You 256
I Love You 512
I Love You 1024
I Love You 2048
I Love You 4096
I Love You More Than 3000
```

计算上述算法的时间复杂度  $T(n)$ :

设最深层循环的语句频度（总共循环的次数）为 x，则由循环条件可知，循环结束时刚好满足  $2^x > n$

$$x = \log_2 n + 1$$

$$T(n) = O(x) = O(\log_2 n)$$

王道考研/CSKAOYAN.COM

14

## 小练习2

```
//算法4— 搜索数字型爱你
void loveYou(int flag[], int n) { //n 为问题规模
    printf("I Am Iron Man\n");
    for(int i=0; i<n; i++){ //从第一个元素开始查找
        if(flag[i]==n){ //找到元素n
            printf("I Love You %d\n", n);
            break; //找到后立即跳出循环
        }
    }
}

//flag 数组中乱序存放了 1~n 这些数
int flag[n]={1...n};
loveYou(flag, n);
```

计算上述算法的时间复杂度  $T(n)$

很多算法执行时间与  
输入的数据有关

**最好情况:** 元素n在第一个位置

——**最好时间复杂度**  $T(n)=O(1)$

**最坏情况:** 元素n在最后一个位置

——**最坏时间复杂度**  $T(n)=O(n)$

**平均情况:** 假设元素n在任意一个位置的概率相同为  $\frac{1}{n}$  ——**平均时间复杂度**  $T(n)=O(n)$

$$\text{循环次数 } x = (1+2+3+\dots+n) \frac{1}{n} = \left(\frac{n(n+1)}{2}\right) \frac{1}{n} = \frac{1+n}{2}$$

王道考研/CSKAOYAN.COM

15

## 算法的时间复杂度

**最坏时间复杂度:** 最坏情况下算法的时间复杂度

**平均时间复杂度:** 所有输入示例等概率出现的情况下，算法的期望运行时间

**最好时间复杂度:** 最好情况下算法的时间复杂度

王道考研/CSKAOYAN.COM

16



本节内容

# 算法

效率的度量

王道考研/CSKAOYAN.COM

1

知识总览



时间开销与问题规  
模  $n$  之间的关系

时间复杂度

空间复杂度

空间开销（内存开销）与  
问题规模  $n$  之间的关系

算法效率的度量

王道考研/CSKAOYAN.COM

2

## 程序运行时的内存需求

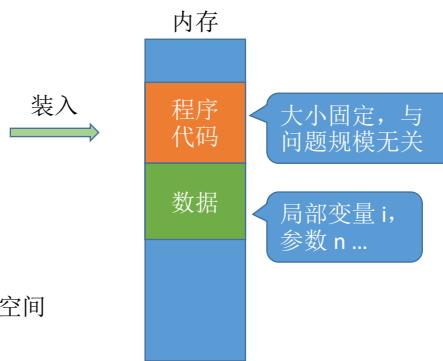
```
//算法1— 步步递增型爱你
void loveYou(int n) { //n 为问题规模
    int i=1; //爱你的程度
    while(i<=n){
        i++; //每次+1
        printf("I Love You %d\n", i);
    }
    printf("I Love You More Than %d\n", n);
}
```

无论问题规模怎么变，算法运行所需的内存空间都是固定的常量，算法空间复杂度为

$$S(n) = O(1)$$

注：S 表示 “Space”

算法原地工作——算法所需内存空间为常量



王道考研/CSKAOYAN.COM

3

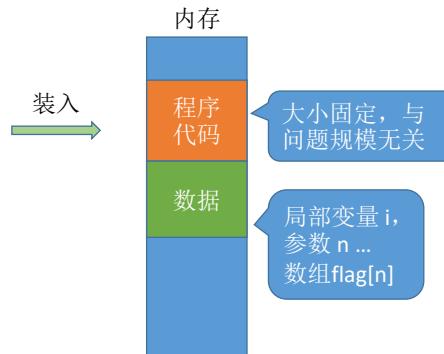
## 空间复杂度

```
void test(int n) {
    int flag[n]; //声明一个长度为n的数组
    int i;
    //.....此处省略很多代码
}
```

假设一个 int 变量占 4B...  
则所需内存空间 = 4 + 4n + 4 = 4n + 8

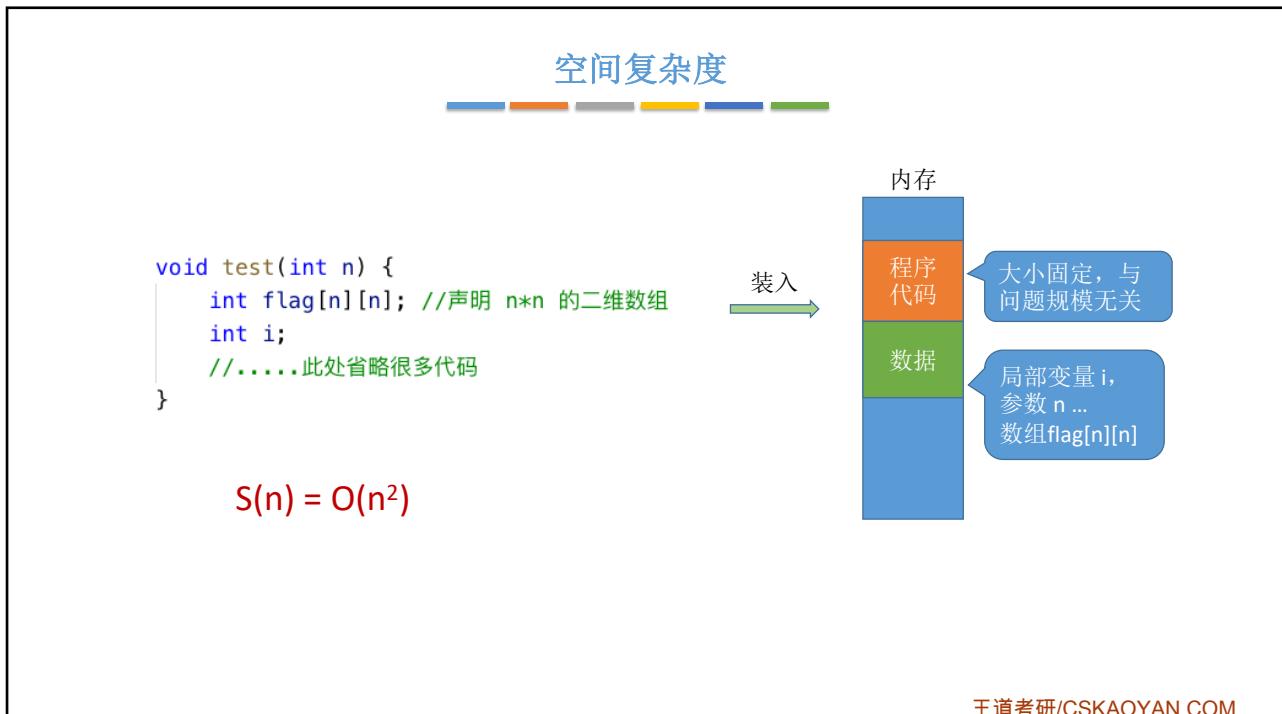
$$S(n) = O(n)$$

只需关注存储空间大小  
与问题规模相关的变量

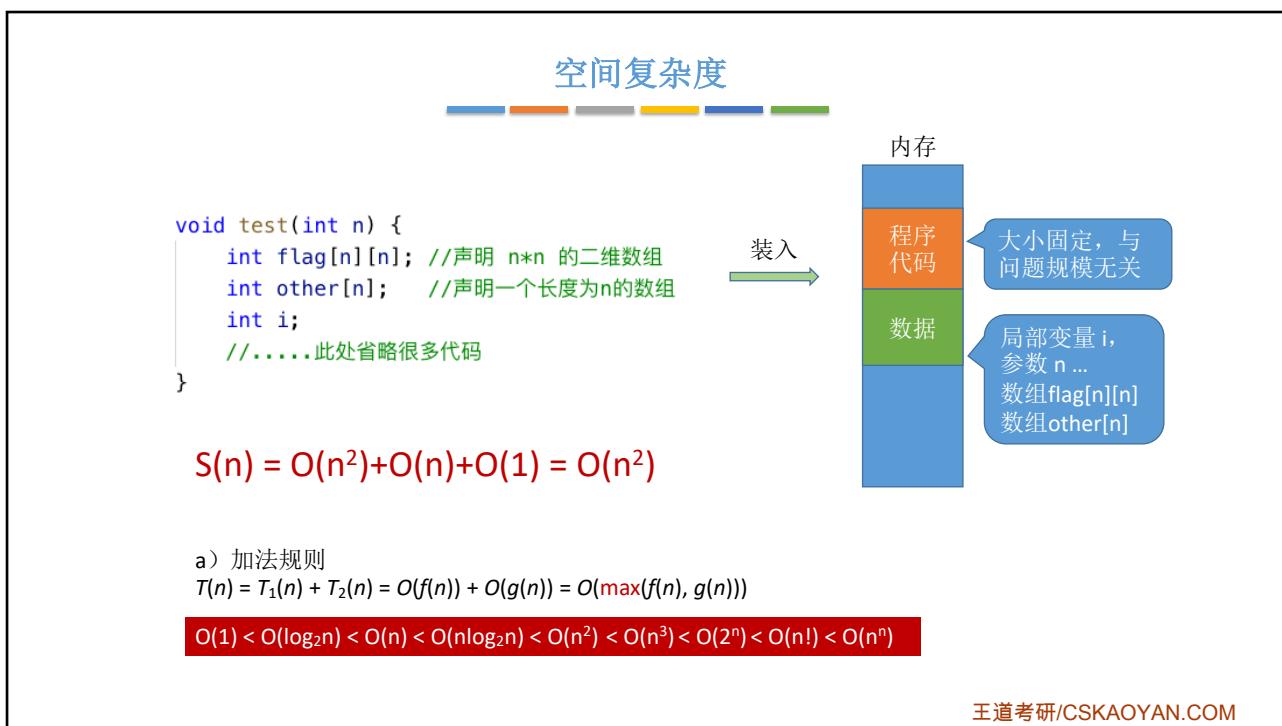


王道考研/CSKAOYAN.COM

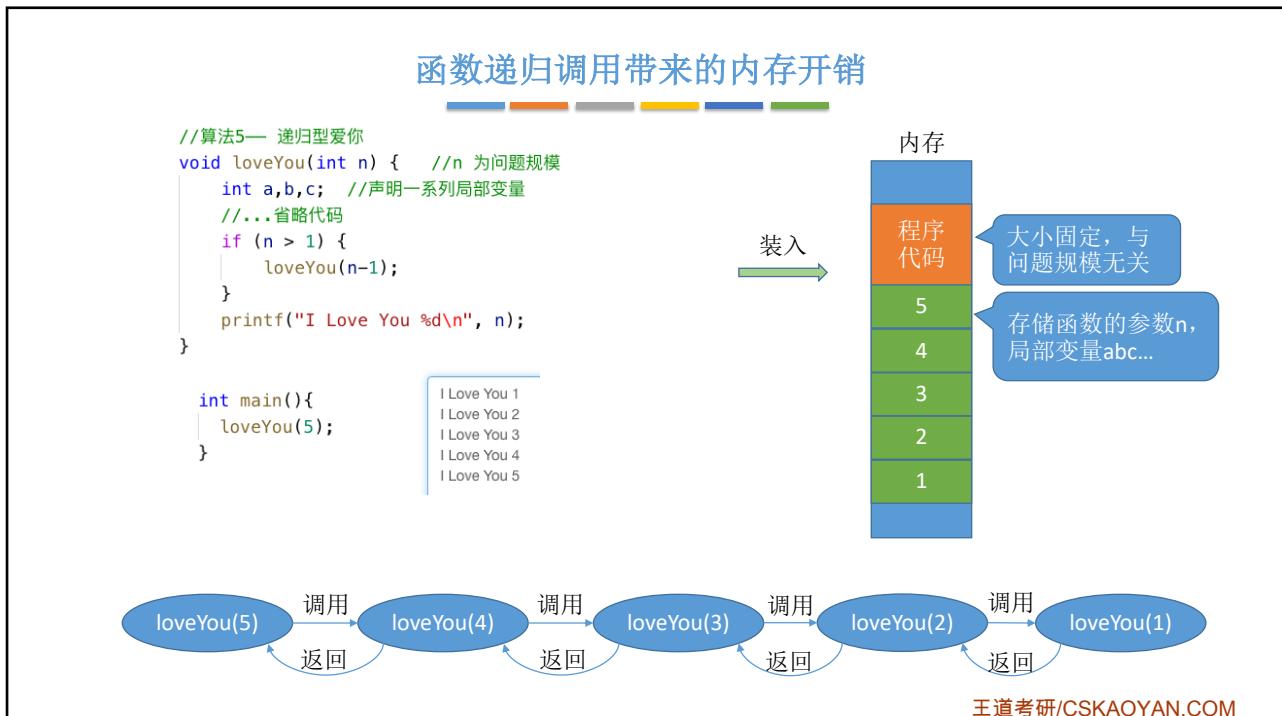
4



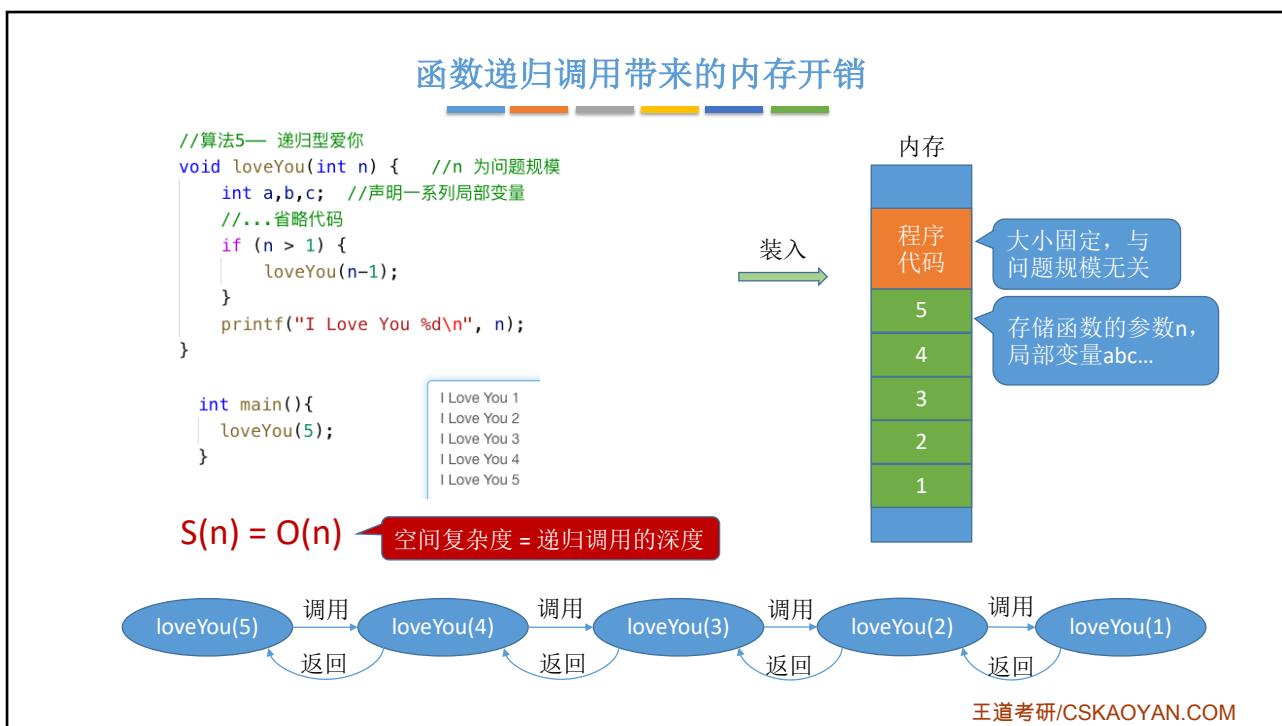
5



6



7



8

## 函数递归调用带来的内存开销

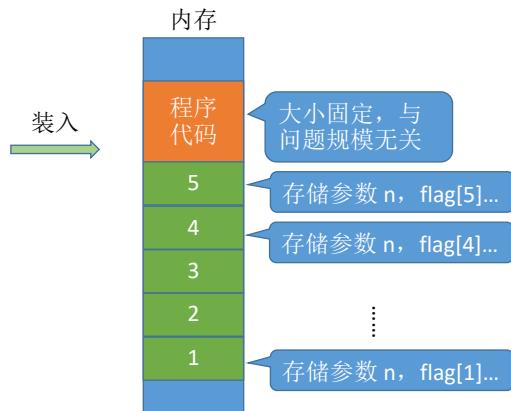
```
//算法5— 递归型爱你
void loveYou(int n) { //n 为问题规模
    int flag[n]; //声明一个数组
    //...省略数组初始化代码
    if (n > 1) {
        loveYou(n-1);
    }
    printf("I Love You %d\n", n);
}

int main(){
    loveYou(5);
}

```

I Love You 1  
I Love You 2  
I Love You 3  
I Love You 4  
I Love You 5

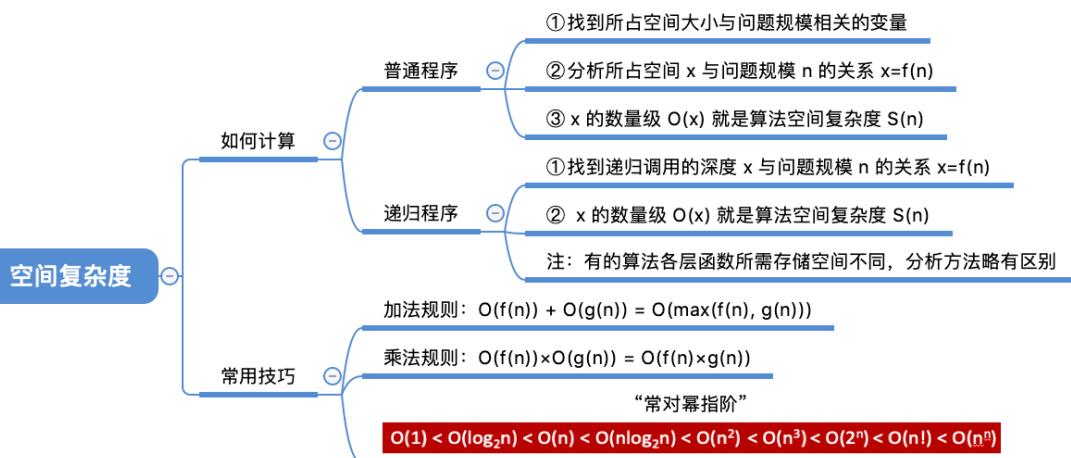
$$1+2+3+\dots+n = [n(1+n)]/2 = \frac{1}{2}n^2 + \frac{1}{2}n$$

$$S(n) = O(n^2)$$


王道考研/CSKAOYAN.COM

9

## 知识回顾与重要考点



王道考研/CSKAOYAN.COM

10