



**教师、助教考核方式**

- 教师
  - ◆ 林广艳 18612627398, [lingy@buaa.edu.cn](mailto:lingy@buaa.edu.cn), 新主楼G304
- 助教
  - ◆ 余书昊 (18967220636), 王嘉骏 (15869111809)
  - ◆ 请大家扫码入群
- 考核方式
  - ◆ 作业及考试均采用上机方式 (在线考试)
  - ◆ 作业: 20%, 期中考试: 30%, 期末考试: 50%

教师: 学生-数据结构与程序设计

扫码入群

**教材及参考书**

- 教材
  - ◆ 《C程序设计导引 (第2版)》, 尹宝林编著, 机械工业出版社, 2020
  - ◆ 《数据结构教程 (第3版)》 唐发根 北京航空航天大学出版社 2017
- 参考书
  - ◆ 《C程序设计语言》, B.W.Kernighan, D.M.Ritchie, 机械工业出版社 2012, 徐宝文等译
  - ◆ 《C程序设计思想与方法》, 尹宝林编著, 机械工业出版社, 2009
  - ◆ 《数据结构与算法分析 (第二版) —C语言描述》, Mark Allen Weiss著 冯舜玺译, 机械工业出版社 2013
  - ◆ 《C++数据结构与算法分析(第4版)》, M.Drozdek著 徐丹 吴伟敏译, 清华大学出版社 2014
  - ◆ 《大话数据结构》, 程杰著, 清华大学出版社 2011
  - ◆ 《数据结构(C语言版)》, 严蔚敏 吴伟民 编著, 清华大学出版社

**教学网站 (课件、作业、答疑和考试)**

- 网址 (建议用Chrome浏览器访问)  
<http://judge.buaa.edu.cn> (<http://10.251.0.206>)
- 用户名: 学号 密码: 学号后四位
- 选择课程: 数据结构与程序设计 (信息大类)
- 请登录尽快修改密码!

查看及提交作业 在线考试 下载课件和相关资料 课程论坛

**关于本课程: 数据结构与程序设计 (信息类)**

- 程序设计的基本步骤
  - ◆ 明确需求, 针对给定的问题, 明确输入、输出要求
  - ◆ 分析问题, 输入数据如何表示, 要做怎样的处理得到输出, 输出数据怎么表示, 怎么输出
  - ◆ 设计算法, 设计解题的方法和具体步骤
  - ◆ 编写程序, 将算法用某一计算机语言实现
  - ◆ 运行、调试, 直至得到满意的结果

**数据结构**

**编程语言运用能力的熟练和加强**

学会分析数据结构的特征, 掌握数据的组织方法和计算机的表示方法, 以便为应用所涉及的数据选择适当的逻辑结构、物理结构和相应的算法, 初步掌握算法时间及空间复杂度分析的技巧, 培养良好的程序设计技能。

学习数据结构, 必须经过大量的实践, 在实践中体会构造性思维方法, 掌握数据组织与程序设计技巧。

**编程没有捷径----练! 练! 练!**

### 关于本课程：数据结构与程序设计（信息类）

- □ 总学时：32上课学时+32上机学时（3学分）
- 计算机、软件工程、网络安全等学科**最重要专业基础课之一**
- 后续专业课程学习的必要知识与技能准备
  - ◆ 编译技术要使用栈、散列表及语法树
  - ◆ 操作系统中使用队列（优先队列）、存储管理表及目录树
  - ◆ 数据库系统运用线性表、多链表、及索引树
  - ◆ ...
- 全国计算机类硕士研究生的入学考试的统科目
- 成为一名专业程序员应具备的基本技能
  - ◆ 大部分公司技术笔试或面试的必考内容

### 教学内容安排

- 第0章 绪论（1）
- 第1章 数据结构程序设计基础（5）
- 第2章 线性表（4）
- 第3章 堆栈和队列（4）
- 第4章 树和二叉树（6）
- 第5章 查找（3）
- 第6章 排序（3）
- 第7章 图（6）

### 课程作业要求

- □ **普通作业（15%）**
  - ◆ 共七套（基础1、基础2、线性表、栈和队、树、图、查找与排序），每套有一组选择填空概念题，多个应用编程题
  - ◆ 普通应用题主要考查学生对**单一知识点的应用能力**
- **综合性能（Project）作业（5%）**
  - ◆ 综合编程题，占整个作业20分中的5分，其测试数据规模较大，主要考查学生对数据结构综合应用的能力。有3道要求完全一样的题目，但测试数据的规模不一样：
    - **小测试数据题评判要求**：正确占100%，不考查性能。同时为了方便学生调试，给出了跟测试数据基本一样的样例数据。**占分：2.5分。**
    - **大测试数据题评判要求**：正确占20%，性能占80%。即在给定时间（100秒）内正确完成，得20%分，性能部分分数以运行最快的前10%程序为基准（其为满分），依次计算得分。运行结果不正确或在给定时间内没有运行结束，则不得分。**占分：2.0分。**
    - **期末评测题**：平时只提交，不评判，课程结束后评判。测试数据比大测试数据题更大一些，只考查正确性（可扩展性），不考查性能。**占分0.5分。**

### 课程作业要求（续）

- □ 应在**规定时间**内提交
  - ◆ 每套普通作业大约开放2~4个星期
- **一定要按照题目要求提交**，比如：输入、输出数据格式，提交文件名称等等
- **严禁抄袭**，编程作业查出抄袭要处理！
  - ◆ 不能拷贝提交他人源代码（提交同学的代码）
  - ◆ 不能下载提交网上源代码（提交网上的代码）
  - ◆ 不要将自己的源码拷贝给他人（可能被抄袭）

### 学习目标和方法

动力能力非常重要，MIT的校训是“脑与手”（Mens et Manus）。

- 目标：掌握数据处理的基本原理和方法，更好地进行算法设计与算法分析，提高程序设计的水准和能力（即**程序设计能力**）
- 方法：
  - ◆ 中国著名桥梁专家、工程教育家茅以升先生提出了：“**习而学**”、“**做中学**”（Learning by Doing）的工程教育思想
  - ◆ 获得程序设计能力的唯一途径

**上机实践(编程)!**  
**(Try!!!)**



## 第0章 绪论

### 目录 CONTENTS

- 01. 课程介绍
- 02. 什么是数据结构
- 03. 算法及其描述
- 04. 算法分析基础

### 软件中的数据结构

软件? 程序 + 文档  
程序? 数据结构 + 算法


**Programs = Data Structure + Algorithm**

主要研究数据之间的关系和数据的组织方式

主要研究常用算法的设计和算法优劣的分析 (性能)

“算法 (algorithm) 是解决特定问题求解步骤的描述, 在计算机中表现为指令的有限序列, 并且每条指令表示一个或多个操作。”

**好数据结构 + 好算法 + 好风格 = 好程序**

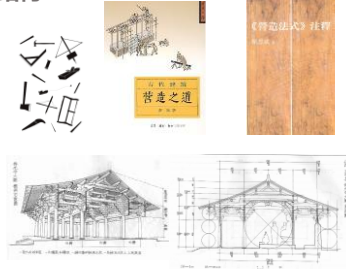


图灵奖得主  
Niklaus Wirth 设计过 (Pascal, Modula-2, 欧几里德)

### 程序设计与数据结构

**程序设计:**  
如何使用程序设计语言 (工具) 来解决问题

**数据结构:**  
研究数据的构造方法



### 什么是数据结构

**数据:** 描述客观事物的数字、字符以及一切能够输入到计算机中, 并且能够被计算机程序处理的符号的集合

**数据元素:** 数据集合中的一个元素

**数据对象:** 具有相同特性的数据元素的集合

**结构:** 数据元素之间具有的关系

**数据项:** (25, 78, 36, 100, 28, 45)

**字母表:** ('A', 'B', 'C', ..., 'Z')

学号	姓名	性别	年龄	其他
a <sub>1</sub> 99001	张三	女	17	.....
a <sub>2</sub> 99002	李四	男	16	.....
a <sub>3</sub> 99003	王五	女	18	.....
a <sub>4</sub> 99004	周六	女	17	.....
⋮	⋮	⋮	⋮	⋮
a <sub>5</sub> 99035	刘宋	男	19	.....

### 数据结构的正式定义

□ 数据元素之间的联系称之为结构, **数据结构**就是具有结构的数据元素的集合

□ 数据结构是一个二元组

$$\text{Data-Structure} = (D, R)$$

其中, D是数据元素的有限集合, R是**D**上的关系的集合

某数据对象

### 数据结构研究的主要内容

**逻辑结构 + 存储结构 + 算法**

□ **数据的逻辑结构** — 客观关系

- 通过抽象的方法研究被处理数据之间存在何种逻辑关系, 即逻辑结构, 有两大类: 线性结构 (线性表、数组、栈、队列、字符串等) 和非线性结构 (树、图等)

□ **数据的物理 (存储) 结构**

- 研究每种逻辑关系在计算机内部如何表示, 即存储结构或者物理结构, 如线性存储结构、链式存储结构、索引结构和散列结构


□ **数据的操作 (算法)**

- 研究在数据各种结构 (逻辑结构和存储结构) 的基础上如何对数据实施有效的操作或处理 (创建、清除、插入、删除、搜索、更新、访问、遍历等)

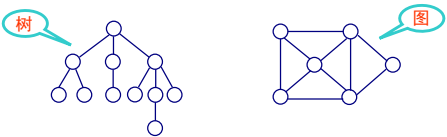
学号	姓名	性别	年龄	其他
99001	张三	女	17	.....
99002	李四	男	16	.....
99003	王五	女	18	.....
99004	周六	女	17	.....
⋮	⋮	⋮	⋮	⋮
99035	刘宋	男	19	.....

### 逻辑结构: 线性结构和非线性结构

**1. 线性结构** 如线性表、堆栈、队列、串、文件等



**2. 非线性结构** 如树、二叉树、图、集合等



## 物理（存储）结构：顺序、链式

### 1. 顺序（sequential）存储结构

用一组**地址连续**的存储单元依次存放数据元素，数据元素之间的逻辑关系通过元素的地址**直接反映**。也就是说数据间的逻辑关系与物理关系是一致的



### 2. 链式（linked）存储结构

用一组**地址任意**的存储单元依次存放数据元素，数据元素之间的逻辑关系通过**指针间接**地反映。在这种结构中，每个数据节点由两部分组成，一部分是数据本身（数据字段）；另一部分是指针，用于存放后续结构的地址（指针字段）



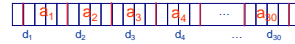
19

## 示例：逻辑结构和物理结构

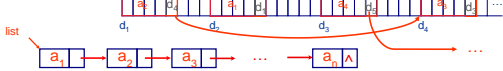
逻辑结构：线性结构（线性表）

存储结构：

### 1. 顺序存储结构



### 2. 链式存储结构



20

## 物理（存储）结构：索引和散列（后续讨论）

### 3. 索引（indexing）存储结构

构造原理：利用数据元素的**索引关系**来确定数据元素的存储位置，由数据元素本身与索引表两部分组成

特点：诸如查找、插入和删除等操作的时间效率较高，但存储空间开销较大



### 4. 散列（hashing，哈希）存储结构

构造原理：通过事先准备好的**散列函数**与处理冲突的方法来**确定数据元素的存储位置**

特点：诸如查找、插入和删除等操作的时间效率较高，主要缺点是确定好的散列函数比较困难

## 操作（算法）

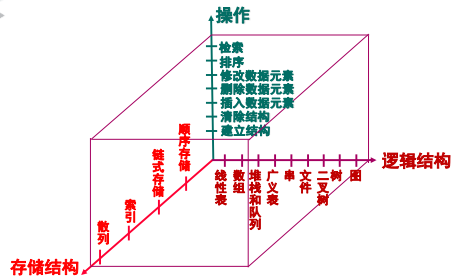
**操作**：对具有某种逻辑结构的数据所实施的一系列操作（算法）

**基本操作**：

- 构造：构造具有某种逻辑结构的数据集，如构造一个线性表、链表、树和图等
- 检索：在已有数据集中查找某一指定元素
- 遍历：访问数据集中所有元素
- 插入：在已有数据集中插入一指定元素
- 删除：在已有数据集中删除一指定元素
- 修改：在已有数据集中修改一个指定元素
- 排序：对一数据集中元素按照某一顺序进行排列
- ...

22

## 数据结构的基本问题空间



23

## 第0章 绪论

### 目录 CONTENTS

- 01. 课程介绍
- 02. 什么是数据结构
- 03. 算法及其描述
- 04. 算法分析基础

## 算法的定义

- 算法是用来解决某个特定问题的**指令的集合**
- 算法是由人们组织起来准备加以实施的一系列**有限的基本步骤**
- 算法是一组解决问题的清晰指令，它能够对符合一定**规范的输入**，在**有限的时间**内获得所需要的**输出**

26

## 算法的性质

- 一个完整的算法应该具有下面五个基本特性：

**输入：**由算法的外部提供 $n \geq 0$ 个有限量作为算法的输入  
**输出：**由算法的内部提供 $n > 0$ 个有限量作为算法的输出  
**确定性：**组成算法的每一条指令必须有清晰明确的含义  
**有穷性：**算法必须在有限的步骤内能够结束  
**可行性：**算法的每一条指令必须具有可执行性

28

## 算法的描述方法

- 1.采用自然语言来描述

**问题：**求两个正整数M与N的最大公因子

- (1) M除以N，将余数赋给中间变量R；
- (2) 判断余数R是否等于零？
  - a) 若R等于零，求得的最大公因子为当前N的值，算法到此结束。
  - b) 若R不等于零，则将N赋给M，将R赋给N，重复步骤(1)和(2)。

27

## 算法的描述方法

**问题：**求两个正整数M与N的最大公因子

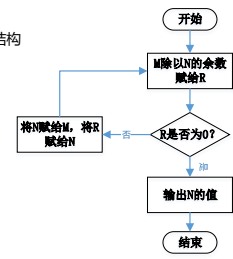
- 2.采用图形化符号来描述

- ◆ 程序流程图：表示顺序、选择、循环结构
- ◆ UML活动图

- 3.采用某种程序设计语言来描述

**C, C++, Java, Python ...**

```
int comfactor(int m, int n){
    int r;
    while(1){
        r = m % n;
        if(r == 0) return n;
        m = n;
        n = r;
    }
}
```



28

## 算法的描述方法

- 1.采用自然语言来描述
- 2.采用图形化符号来描述
- 3.采用某种程序设计语言来描述

- 4.采用伪代码来描述
  - ◆ 类似于某种高级语言，但又没有严格的语法和格式要求
  - 如：SPARKS语言（一种类Pascal语言）

```

1  void Unweighted Table T; // Assume T is initialized w/
2  int CurDist;
3  Vertices V, W;
4  for( CurDist = 0; CurDist < NumVertices; CurDist++ )
5  {
6    for each vertex V
7    {
8      if( T[V][V].Known && T[V][V].Dist == CurDist )
9      {
10         T[V][V].Known = True;
11         for each W adjacent to V
12         {
13           if( T[W][V].Dist == Infinity )
14           {
15             T[W][V].Dist = CurDist + 1;
16             T[W][V].Path = V;
17           }
18         }
19       }
20     }
21   }

```

Dijkstra算法伪代码（类C）

29

## 第0章 绪论

### 目录 CONTENTS

01. 课程介绍
02. 什么是数据结构
03. 算法及其描述
04. 算法分析基础

### 算法分析

**目的** 改进算法的质量  
**前提** 算法必须正确

算法分析是指对算法质量（效率）优秀的评价

除**正确性**外，通常从三个方面分析一个算法：

- 依据算法编写的程序在计算机中运行时间多少的度量，称之为**时间复杂度**（反映算法运行的快慢）
- 依据算法编写的程序（运行时）在计算机中占存储空间多少的度量，称之为**空间复杂度**（反映算法需要的额外空间的多少）
- 其他方面，如算法的可读性、可移植性以及易测试性的好坏

时空效率高的算法才是一个好的算法，它常常是不懈努力和反复修正的结果

### 时间复杂度

一个程序在计算机中运行时间的多少与诸多因素有关，其中主要有：

1. **问题的规模**（几乎所有算法的时间效率都与问题的规模有关）
2. 编译程序功能的强弱以及所产生的机器代码质量的优劣
3. 机器执行一条指令的时间长短

★ **程序中那些基本语句的执行次数**---与算法策略有关（对算法运行时间贡献最大的语句）

### 时间复杂度的表示法：频度统计法

□ **频度统计法**：以语句执行的次数的多少作为算法的时间度量的分析方法

- ◆ 一条**语句的频度**是指该语句被执行的次数
- ◆ 整个**算法的频度**是指算法中所有语句的频度之和

```
#define M 1000
void MATRIX(int A[ ][M],int B[ ][M],int C[ ][M],int n)
{
    int i, j, k;
    for(i=0;i<n;i++)
        for(j=0;j<n;j++){
            C[i][j]=0;
            for(k=0;k<n;k++){
                C[i][j]=C[i][j]+A[i][k]*B[k][j];
            }
        }
}
```

语句频度

算法的频度： $t(n) = 2n^3 + 3n^2 + 2n + 1$

### 频度统计法的分析

□ 假设算法A的语句频度为 $2n^2$ ，算法B的频度为 $3n+1$ ，算法C的语句频度为 $2n^2+3n+1$ ，则有下表：

次数	算法 A ( $2n^2$ )	算法 B ( $3n+1$ )	算法 C ( $2n^2+3n+1$ )
n = 1	2	4	6
n = 2	8	7	15
n = 5	50	16	66
n = 10	200	31	231
n = 100	20 000	301	20 301
n = 1,000	2 000 000	3 001	2 003 001
n = 10,000	200 000 000	30 001	200 030 001
n = 100,000	20 000 000 000	300 001	20 000 300 001
n = 1,000,000	2 000 000 000 000	3 000 001	2 000 000 300 001

从中可以看出：  
判断一个算法的效率时，算式中的常数和次要项通常可以忽略，而更应该关注主要项（最高阶项）的阶数

### 时间复杂度的表示法：大O表示法(渐近时间复杂度)

□ 当且仅当存在正整数c和 $n_0$ 使得 $t(n) \leq cf(n)$ 对所有的 $n \geq n_0$ 成立，则称函数 $t(n)$ 与 $f(n)$ 同阶，或者说， $t(n)$ 与 $f(n)$ 同一个数量级，记作

$t(n) = O(f(n))$

称上式为算法的渐近时间复杂度，或称该算法的时间复杂度为 $O(f(n))$ 。

其中，n为问题的规模(大小)的度量

当问题的输入规模n趋于无穷大时，算法的运行时间表现出固定的增长次数：

$t(n) = n^3 + n^2$      $f(n) = n^3$   
称算法的时间复杂度为 $O(n^3)$

渐近时间复杂度：当问题规模n趋于无穷大时，算法最坏时间代价的上界的数量级

### 大O (Order) 表示法

□ 大O表示法关注的是问题规模n增长时，算法执行次数增长的数量级。因此，可以只关注算法中**基本语句**执行频度，不必对算法的每一个步骤都进行详细的分析。

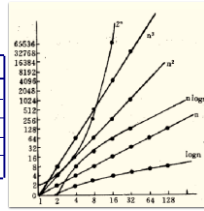
```
i=1; (1)
while (i<=n) (2)
    i=i*2; (3)
```

可只关注语句(3)的频度，设为 $t(n)$ ，则有： $2^t(n) \leq n$   
即： $t(n) \leq \log_2 n$ ，取最大值 $t(n) = \log_2 n$   
即：时间复杂度为： $O(\log_2 n)$

## 常见的的时间复杂度

对于算法分析具有重要意义的常见函数值

n	log <sub>2</sub> n	n	n log <sub>2</sub> n	n <sup>2</sup>	n <sup>3</sup>	2 <sup>n</sup>	n!
10	3.3	10 <sup>1</sup>	3.3×10 <sup>1</sup>	10 <sup>2</sup>	10 <sup>3</sup>	10 <sup>3</sup>	3.6×10 <sup>6</sup>
100	6.6	10 <sup>2</sup>	6.6×10 <sup>2</sup>	10 <sup>4</sup>	10 <sup>6</sup>	1.3×10 <sup>30</sup>	9.3×10 <sup>157</sup>
1000	10	10 <sup>3</sup>	1.0×10 <sup>4</sup>	10 <sup>6</sup>	10 <sup>9</sup>		
10000	13	10 <sup>4</sup>	1.3×10 <sup>5</sup>	10 <sup>8</sup>	10 <sup>12</sup>		
...	...	...	...	...	...		



常见的的时间复杂度:

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

$O(1)$ : 表示算法的复杂度为常量, 不随问题规模n的大小而改变

## 示例: 求时间复杂度

求两个n阶方阵的相加 $C=A+B$ 的算法如下, 分析其时间复杂度

```
#define MAX 20 /*定义最大的方阶*/
void matrixadd(int n, int A[MAX][MAX],
               int B[MAX][MAX], int C[MAX][MAX]) {
    int i, j;
    for (i=0; i<n; i++)
        for (j=0; j<n; j++)
            C[i][j] = A[i][j] + B[i][j];
}
```

该算法中的基本运算是两重循环中最深层的语句 $C[i][j] = A[i][j] + B[i][j]$

分析频度

$$T(n) = \sum_{i=0}^{n-1} \sum_{j=0}^{n-1} 1 = \sum_{i=0}^{n-1} n = n \sum_{i=0}^{n-1} 1 = n \cdot n = n^2$$

时间复杂度:  $O(n^2)$

## 示例: 求时间复杂度 (续)

分析以下算法的时间复杂度

```
int fun(int n)
{
    int i, j, k, s;
    s=0;
    for (i=0; i<n; i++)
        for (j=0; j<=i; j++)
            for (k=0; k<=j; k++)
                s++;
    return(s);
}
```

该算法的基本操作是语句 $s++$ , 其频度:

$$T(n) = \sum_{i=0}^{n-1} \sum_{j=0}^i \sum_{k=0}^j 1 = n^3$$

时间复杂度:  $O(n^3)$

## 示例: 求时间复杂度 (续)

分析以下算法的时间复杂度

```
void func(int n)
{
    int i=0, s=0;
    while (s<n)
    {
        i++;
        s=s+i;
    }
}
```

对于while循环语句, 设执行的次数为m, i从0开始递增1, 直到m为止, 有:

$$s = 0+1+2+\dots+m-1 = m(m-1)/2, \text{ 并满足}$$

$$s = m(m-1)/2 < n,$$

$$\text{则有 } m \leq \sqrt{2n}$$

即: 时间复杂度:  $O(\sqrt{n})$

## 示例: 求时间复杂度 (续)

调用算法 $\text{fun}(a, n, 0)$ , 求其时间复杂度

```
void fun(int a[], int n, int k)
//数组a共有n个元素
{
    int i;
    if (k==n-1)
        for (i=0; i<n; i++)
            printf("%d\n", a[i]);
    else
    {
        for (i=k; i<n; i++)
            a[i] = a[i] + i*i;
        fun(a, n, k+1);
    }
}
```

设 $\text{fun}(a, n, 0)$ 的时间复杂度为 $T(n)$ ,  $\text{fun}(a, n, k)$ 执行时间为 $T_1(n, k)$ , 由 $\text{fun}()$ 算法可知:

$$T_1(n, k) = n \quad \text{当 } k = n-1 \text{ 时}$$

$$T_1(n, k) = (n-k) + T_1(n, k+1) \quad \text{其他情况}$$

则:

$$\begin{aligned} T(n) &= T_1(n, 0) = n + T_1(n, 1) = n + (n-1) + T_1(n, 2) = \dots \\ &= n + (n-1) + \dots + 2 + T_1(n, n-1) = n + (n-1) + \dots + 2 + n = n^2 \end{aligned}$$

调用 $\text{fun}(a, n, 0)$ 的时间复杂度:  $O(n^2)$

## 时间复杂度: 最坏情况和平均情况

对算法的分析, 一种是平均时间复杂度; 另一种是最坏时间复杂度。一般在没有特殊说明的情况下, 都是指最坏时间复杂度

最坏情况运行时间是一种保证, 那就是运行时间将不会再坏了。在实际应用中, 这是一种最重要的需求。如机载软件关键软件中必须要做最坏情况的时间分析

平均运行时间是所有情况中最有意义的, 因为它是期望的运行时间

### 延伸学习\* - 算法空间复杂度

- 算法空间复杂度通过计算算法运行时所需的存储空间来实现
- 算法空间复杂度是衡量算法效率的另一个重要指标
- 程序设计时，经常会用空间来换时间

43

### 本章小结

- 了解课程的总体安排
  - ◆ 认识课程的定位，理解课程要求和目标
- 掌握数据结构的基本概念
  - ◆ 逻辑结构、物理（存储）结构和算法
- 掌握算法的基本概念，能够分析算法的时间复杂度
  - ◆ 算法的性质、描述方法
  - ◆ 时间复杂度
  - ◆ 空间复杂度

44



休息一会了

45