数据结构与算法

人工智能与大数据学院

目　录

课程重要性

数据结构与算法是一门十分重要的专业核心课程，所有的计算机系统软件和应用软件都要用到各种类型的数据结构。

如何利用计算机求解各种问题？

- 设计并完成一个**图书管理系统**（包括采编入库、清除库存、借阅、归还、检索、预约借阅等功能）
- 设计并完成一个**邮路问题**（邮递员的投递任务是走过所有的地点再返回有句，怎样的走法可以使他的总行程最短）
- 设计并完成一个**程序相近度**的检测系统
- 设计并完成一个景区**旅游信息管理系统**（制定导游线路策略，输出任何两个景点的最短距离）

课程重要性

- 设计并完成一个图书管理系统（树+查找算法等）
- 设计并完成一个邮路问题（图+拓扑排序等）
- 设计并完成一个程序相近度的检测系统（散列表+开放地址法等）
- 设计并完成一个景区旅游信息管理系统（图+迪杰斯特拉算法或弗洛伊德算法）

程序 = 数据结构 + 算法

课程目标

- **目标1**：使学生理解和掌握表、树、图、散列等常见数据结构和算法设计，能够设计计算机应用系统中关键模块和算法，并能够在设计环节对系统中的关键模块和算法的解决方案进行比较和综合。
- **目标2**：使学生能够采用专业科学方法分析问题，并能够根据对象特征，选择数据结构、设计算法和实验方案，分析与解释数据，并通过信息综合得到合理有效的结论，具有算法设计与分析能力。
- **目标3**：使学生能够针对问题的应用背景分析，完成问题的抽象、数据的提取组织、数据结构的选择和算法的设计，设计并满足特定需求的系统、单元或开发流程，具有问题导向的数据结构的建模和算法设计能力。
- **目标4**：使学生能够选择与使用恰当的数据结构与算法技术，对实验数据和结果进行分析解释，结合用户需求，尝试从时空复杂、鲁棒性等方面对求解方法进行分析和改进，具有面向应用背景的数据结构与算法的迭代优化能力，进一步提升计算思维能力。

课程安排和考核方式

- **课程安排**：理论56学时+实验24学时+自主学习

- **考核方式**：课堂出勤+平时作业+过程考核+实验考核+自主学习 + **期末考试**

第1章　　绪 论

本章的学习目标：

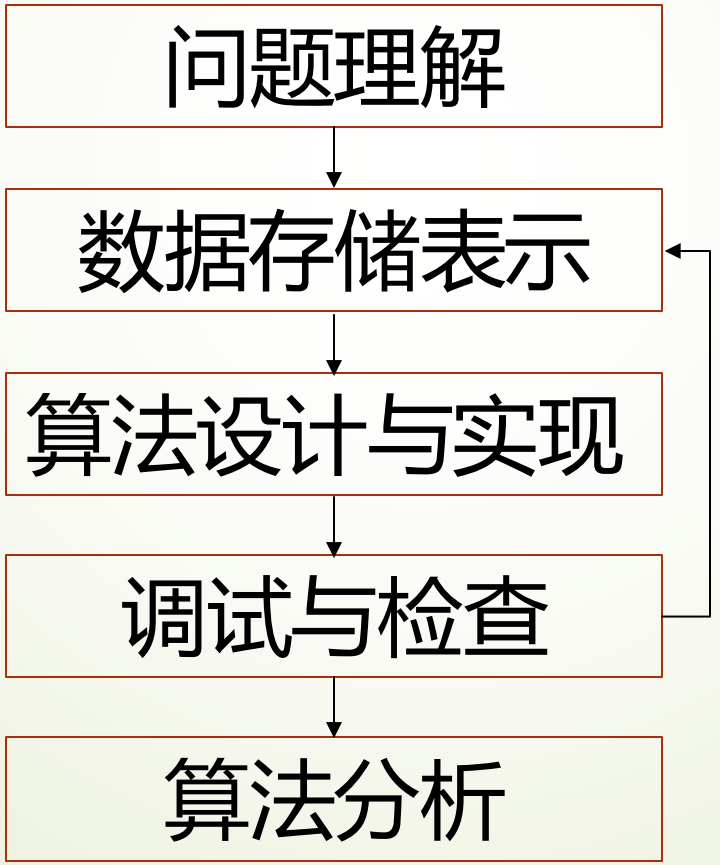
① 理解和掌握数据结构的概念

② 理解数据的逻辑结构和物理结构

③ 理解算法的概念和特性

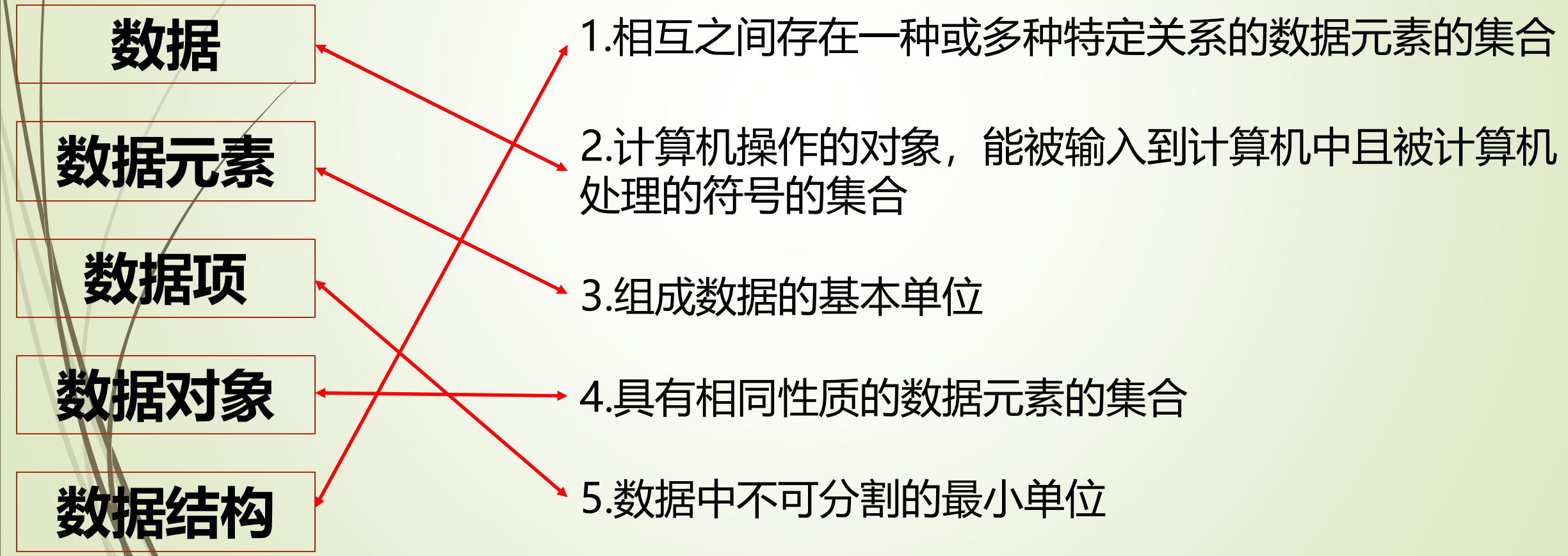
第1章　绪 论

计算机解决问题的一般步骤



第1章 绪 论

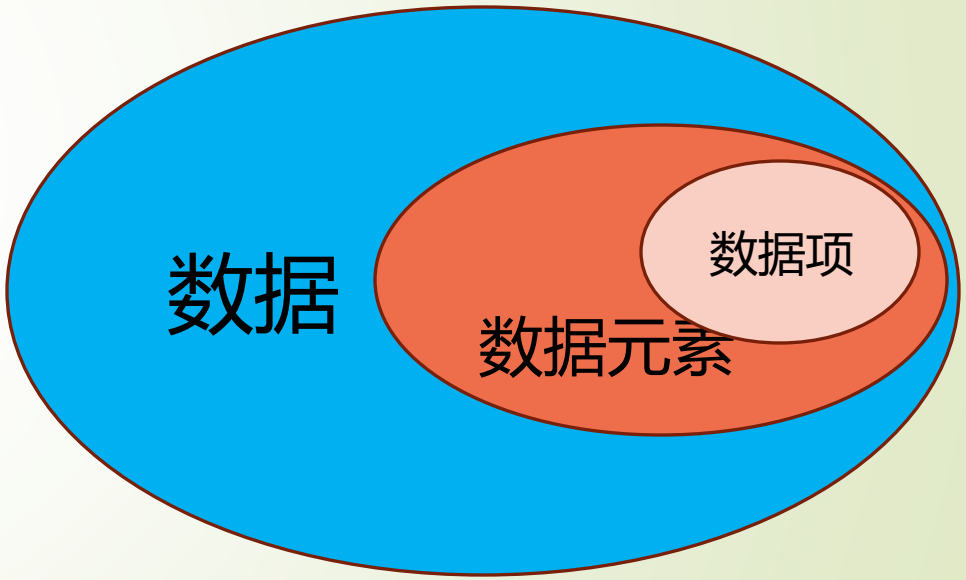
数据结构的基本概念：



第1章 绪 论

数据结构的基本概念

- 数据
- 数据元素
- 数据项
- 数据对象
- 数据结构



第1章 绪 论

数据结构的基本概念

数据对象指的是具有相同性质的数据元素的集合。

例如：每个人都有姓名、性别、年龄、籍贯等数据项，

在实际开发应用中，我们处理相同性质的数据元素时，默认将数据对象简称为数据

第1章 绪论——数据结构的基本概念

数据结构

- 逻辑结构　　数据元素之间逻辑关系的描述
- 物理结构　　数据元素及其关系在计算机内存中的表示

第1章 绪论——数据结构的基本概念

数据元素 1

数据元素 2

将数据元素之间存在的关系称为**逻辑关系**或**逻辑结构**

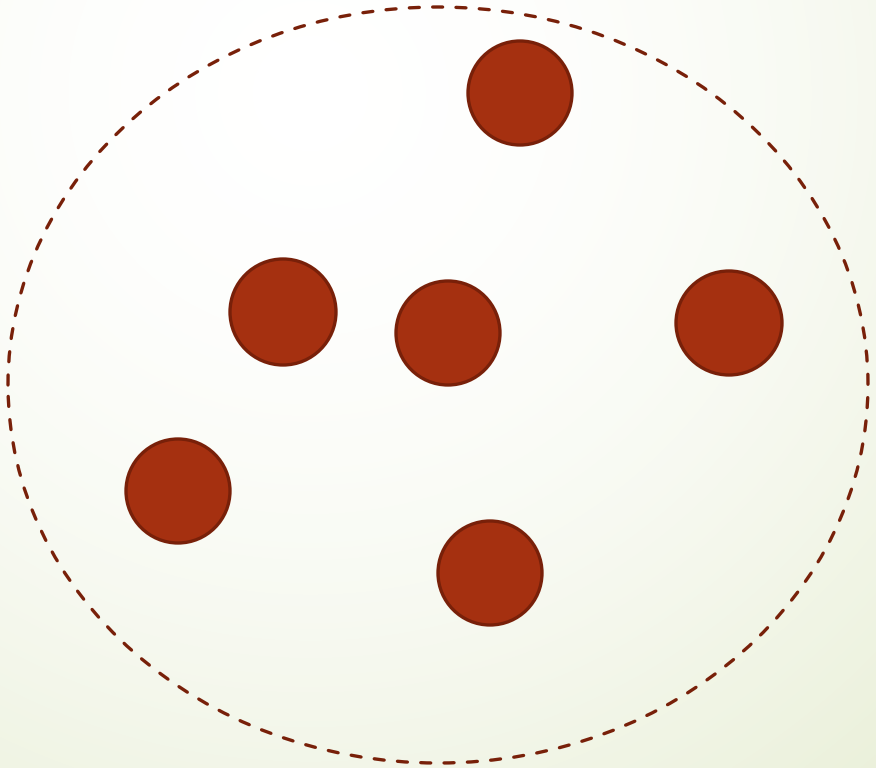
① 集合关系
② 线性关系
③ 树形关系
④ 图形关系

① 集合结构
② 线性结构
③ 树形结构
④ 图形结构

第1章 绪论——数据结构的基本概念

（1）集合结构

不同的数据元素之间没有直接的关系（举例）

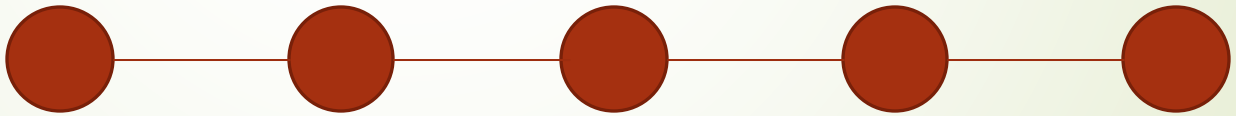


第1章 绪论——数据结构的基本概念

(2) 线性结构

不同的数据元素之间的关系是一个线性关系（举例）

每一个数据元素有且只有一个前驱元素，每一个数据元素有且只有一个后继元素

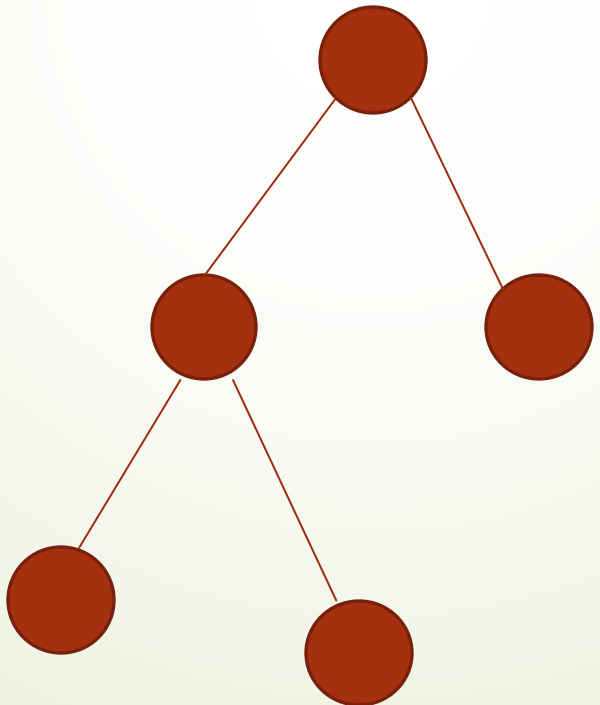


第1章 绪论——数据结构的基本概念

（3）树形结构

每一个数据元素有且只有一个前驱元素不同的数据元素之间的关系是一个线性关系，

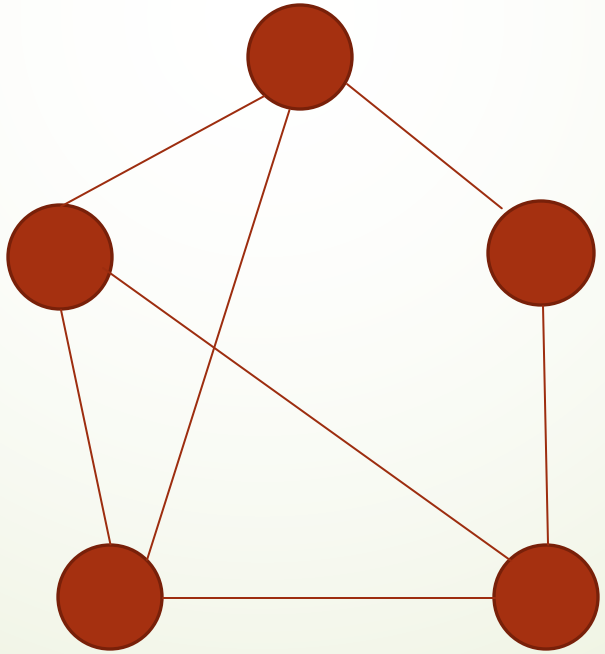
每一个数据元素有且只有一个前驱元素（除了第一个结点），但是可以有任意多个后继元素。（举例）



第1章 绪论——数据结构的基本概念

（4）图形结构

每一个数据元素有任意多个前驱元素和任意多个后继元素。（举例）



第1章 绪论——数据结构的基本概念

物理结构：数据元素及其关系在计算机内存中的表示

物理结构

顺序存储	将相邻的数据元素存在计算机地址连续的存储单元中
链式存储	逻辑上相邻的数据元素在内存中不一定相邻
索引存储	存储数据元素的同时建立索引表，存储元素间的关系
散列存储	根据数据元素的关键字直接计算出该数据元素的存储位置

习 题

1. 输入到计算机中被计算机程序处理的符号集合称为_____

2. 组成数据的基本单位是称为_____

3. 数据元素是由_____组成，并且它是数据不可分割的最小单位

4. 具有相同性质的数据元素的集合称为_____

5. 链式存储中的数据元素在内存中一定不相邻（正确或错误）

6. 生活中常用交通路线图更适合采用图这种数据结构表示（正确或错误）

第1章 绪论——算法的基本概念

算法：是指解题方案的准确而完整的描述

算法的描述

- 自然语言
- 符号语言
- 计算机程序

例如：如何描述这个问题
“从3个整数中选出最大的1个整数输出”

第1章 绪论——算法的基本概念

算法的描述

- 自然语言
- 符号语言
- 计算机程序

特性

输入、输出、有穷性、确定性、可行性

第1章 绪论——算法的基本概念

1、算法的基本性质——输入输出

一个算法可以有多个输入或者没有输入，有些算法的输入需要在执行过程中进行，有些算法则是将输入嵌入到算法之中。

```
int main(int argc, const char *argc[])
{
    int count = 0, i;
    for(i=1; i<=100; i++)
        count += i;
    printf("%d\n",count);
    return 0;
}
```

一个算法必须有一个或者多个输出，这些输出是算法对输入进行运算后的结果。

如果一个算法没有任何输出，那么该算法没有意义

第1章 绪论——算法的基本概念

1、算法的基本性质——有穷性

有穷性是指算法在执行有限次数的操作后自动停止，不会出现无限循环的情况。简单地说，算法运行一定有结束的时刻。

```
int main(int argc, const char *argc[])
{
    int s = 0, i=0, n;
    scanf(“%d”,&n);
    while(i < n);
        {s = s+i; i++; }
    printf(“%d\n”,s);
    return 0;
}
```

第1章 绪论——算法的基本概念

1、算法的基本性质——确定性

确定性是指算法的每次操作都具有确定的含义，不会出现二义性。算法在一定条件下，应该只有一条执行路径。

相同的输入在任何时刻执行都应该导向相同的结果。

1、算法的基本性质——可行性

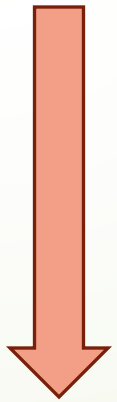
可行性是指算法中描述的操作都是可以通过将已经实现的基本运算执行有限次来实现。

简单地说，算法可以转换为程序上机运行，并可以输出正确的结果。

2、算法的设计要求

同一个问题，可以有多种解决方案，采用多种算法实现，判断一个算法是否为最优算法，可以从以下四个方面分析：

① 正确性
② 可读性
③ 健壮性
④ 高效率



2、算法的设计要求——正确性

正确性是对算法的基本要求。正确的算法能够正确反映问题的需求，经得起一切输入数据的测试。

① 算法程序没有语法错误；
② 对于合法的输入数据能够输出满足要求的结果；
③ 对于非法的输入数据能够输出满足规格的说明；
④ 对于极端的输入数据能够输出满足要求的结果；

```
int main(int argc, const char *argc[])
{
    int i, n, count = 0;
    scanf(“%d”,&n);
    for(i=0;i<n;i++)
        {count=count+i;}
    printf(“%d\n”,count);
    return 0;
}
```

2、算法的设计要求——可读性、健壮性、高效率

- 可读性：算法的设计应该尽可能的简单，便于阅读和理解
- 健壮性：算法可以对不合法的输入数据做出恰当的处理，而不是产生异常或极端的结果
- 高效率：算法的**运行时间**要尽量短，对**存储空间**的使用要尽可能少

3、算法效率的度量方法——事前分析法

在设计算法程序之前，利用统计方法对算法效率进行估算。

```
int main(int argc, const char *argc[])
{
    int i, sum=0, n=100;          执行1次
    for(i=0;i<=n;i++)             执行n+2次
        {sum=sum+i;}              执行n+1次
    printf("%d\n",sum);           执行1次
    return 0;
}
```

```
int main(int argc, const char *argc[])
{
    int i, sum=0, n=100;
    sum=(n+1)*n/2;                执行1次

    printf("%d\n",sum);
    return 0;
}
```

4、算法的时间复杂度

影响算法的执行时间有很多，例如不同算法的实现策略、不同的编程语言、不同的硬件配置、不同大小的输入数据量等等

把变量n称为问题规模，算法中语句的执行次数称为时间频度，记为T(n)，当n不断变化时，T(n)也会不断变化

如果有某个辅助函数f(n)，并且存在一个正常数c，使得$f(\mathrm{n}) \times c \geq T(n)$恒成立，记作$T(n) = O(f(n))$,将$O(f(n))$称为算法的渐进时间复杂度，简称**时间复杂度**。

4、算法的时间复杂度——$O()$

如果有某个辅助函数f(n)，并且存在一个正常数c，使得$f(n) \times c \geq T(n)$恒成立，记作$T(n) = O(f(n))$,将$O(f(n))$称为算法的渐进时间复杂度，简称**时间复杂度**。

```
int main(int argc, const char *argc[])
{
    int i, sum=0, n=100;
    for(i=0;i<=n;i++)
        {sum=sum+i;}
    printf("%d\n",sum);
    return 0;
}
```

算法执行时间为2n+5，当问题规模n变为无穷大时，该算法的执行时间可估算为n，则该算法的时间复杂度为O(n)。

4、算法的时间复杂度——$O()$

【定理】从数学的角度提供了另外一种估算算法时间复杂度的方法。可以证明符号O有如下的运算原则

(1) $O(f)+O(g)=O(\max(f,g))$
(2) $O(f)\cdot O(g)=O(fg)$
(3) 如果$g(N)=O(f(N))$，则$O(f)+O(g)=O(f)$
(4) $O(Cf(N))=O(f(N))$，其中C是一个正的常数
(5) $f=O(f)$

4、算法的时间复杂度——$O()$

常见的算法时间复杂度有常数阶$O(1)$，线性阶$O(n)$，平方阶$O(n^2)$，对数阶$O(log_2 n)$，线性对数阶$O(nlog_2 n)$，立方阶$O(n^3)$和指数阶$O(2^n)$等

常见的时间复杂度所对应的时间顺序如下：

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

思考：

1、某算法的语句执行次数为$3n + log_2 n + n^2 + 1$,其时间复杂度表示为?

2、程序段i=1;while(i<=n)i=i*3;的时间复杂度为？

本章小结

本章以概念为主，重点介绍了数据的概念、数据的逻辑结构以及算法的概念。通过本章的学习，需要对数据结构与算法建立初步的认识，重点理解逻辑结构的设计思想、算法时间复杂度的分析，为后续的学习奠定基础。