第一章 绪 论

1.1 什么是数据结构
1.2 基本概念和术语
1.3 抽象数据类型的表示与实现
1.4 算法和算法分析
- 1.4.1 算法
- 1.4.2 算法设计的要求
- 1.4.3 算法效率的度量
- 1.4.4 算法的存储空间的需求

- **内容提要及考试重点：**
  - 相关基本概念
  - 算法时间复杂度的度量

1.1什么是数据结构

- 数据结构与程序设计密切相关，1969年，Niklaus Wirth提出：
  - Algorithm + Data Structure = Program
- 程序设计：为计算机处理问题编制一组指令集
- 如何进行处理？------**算法**
- 对处理的信息如何表示？-----问题的数学模型----**数据结构**

数值计算的程序设计问题:

- 桥梁结构压力---线性方程组

  预报人口增长---常微分方程

是**计算数学**研究内容

非数值计算的程序设计问题：

- 图书馆的书目检索系统自动化问题
- 人机对弈
- 多叉路口交通灯的管理问题

是**数据结构**的研究内容

- 在大多数情况下，计算机要处理的信息（数据）并不是没有组织。这些信息（数据）之间往往具有重要的结构关系。那么，什么是数据结构呢？先看以下几个例子。

例1 电话号码查询问题

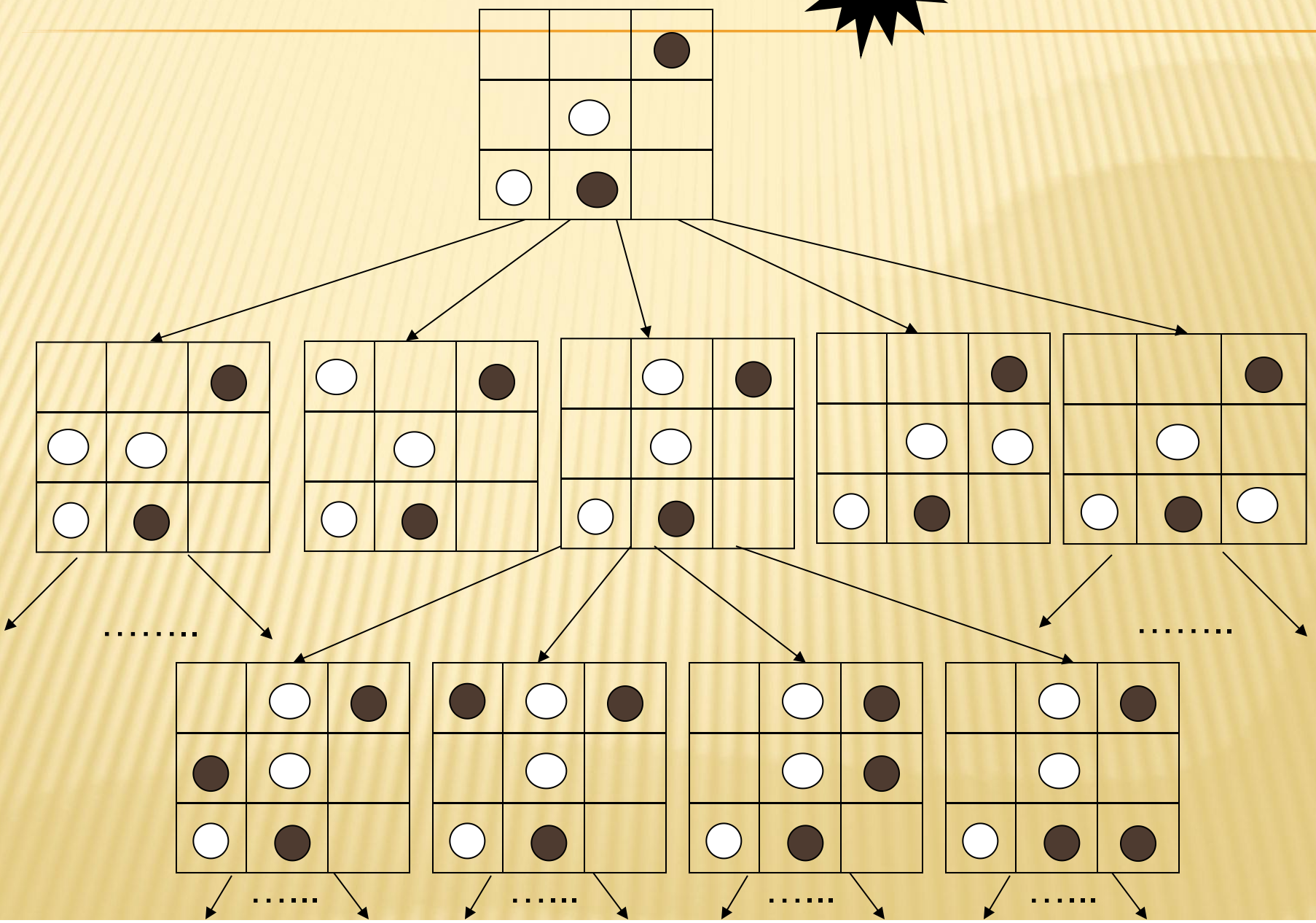
二维表...

姓名	电话号码
张三	…
李四	…
…	…
…	…

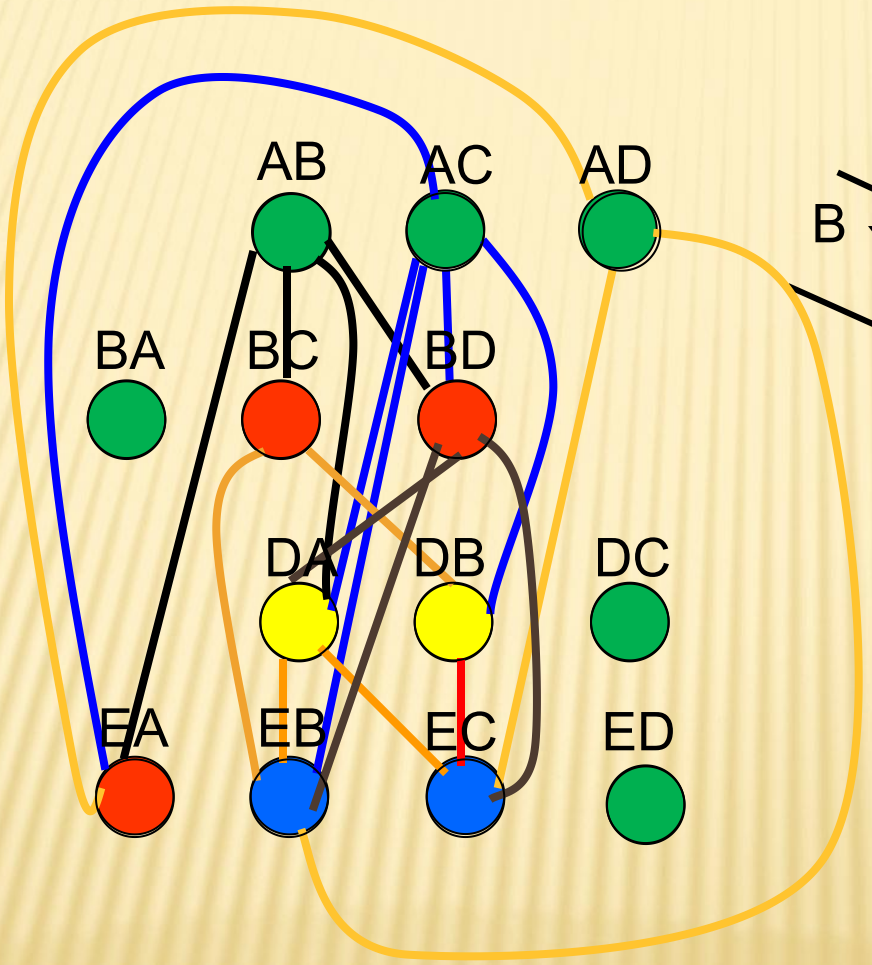
- 算法的设计，依赖于计算机如何存储人的名字和对应的电话号码，或者说依赖于名字和其电话号码的结构。
- 数据的结构，直接影响算法的选择和效率。
- 上述的问题是一种数据结构问题。可将名字和对应的电话号码设计成一个查找表。

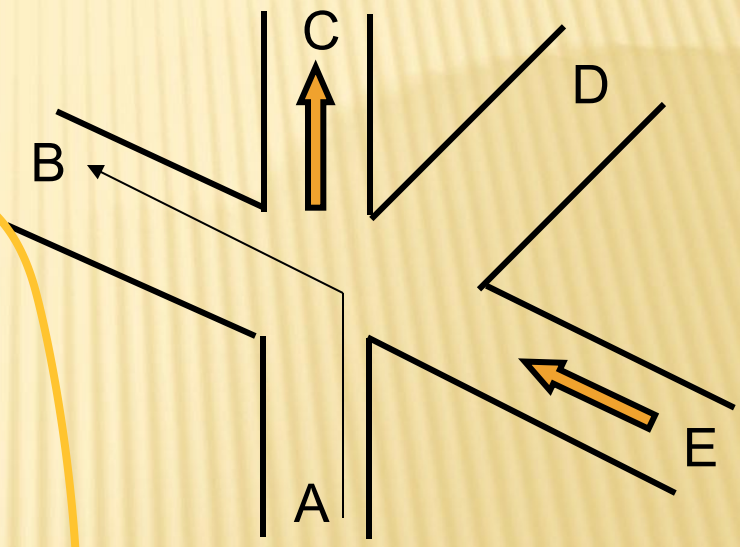
- 概括地说，数据结构描述现实世界实体的数学模型（非数值计算）及其上的操作在计算机中的表示和实现。

+ 例2 人机对奕问题



例3 多叉路口交通灯管理问题





数据结构就是研究数据的逻辑结构和物理结构以及它们之间相互关系，并对这种结构定义相应的运算，而且确保经过这些运算后所得到的新结构仍然是原来的结构类型。

1.2 基本概念和术语

- 数据和数据结构
- **数据**（Data）是对客观事物的符号表示，在计算机科学中是指所有能输入到计算机中并能被计算机处理的符号的总称。包括字符、图形、声音等等。

1.2 基本概念和术语

- **数据元素**（Data Element），是数据的基本单位，在计算机程序中通常作为一个整体进行考虑处理。
- 一个数据元素可由若干个**数据项**（Data Item）组成。
- **数据项**是数据的不可分割的最小单位。

姓名	俱乐部名称	出生日期 年 月 日	入队日期	职位	业绩
C罗	皇马	1985.2.5	2009.10	中锋	好

1.2 基本概念和术语

例：2行3列的二维数组（a1，a2，a3，a4，a5，a6）

a1	a2	a3
a4	a5	a6

行的次序关系

row={<a1，a2>，<a2，a3>，<a4，a5>，<a5，a6>

列的次序关系：

col={<a1，a4>，<a2，a5>，<a3，a6>}

1.2 基本概念和术语

- **数据对象**（Data Object）：是性质相同的数据元素的集合，是数据的一个子集。

整数数据对象是集合N = {…,-2,-1,0,1,2,…}

字母字符数据对象是集合C = {‘A’,’B’,…,’Z’}

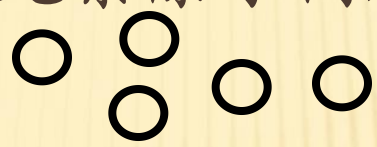
1.2 基本概念和术语

- **数据结构**，是带结构的数据元素的集合，是相互之间存在一种或多种特定关系的数据元素的集合。数据元素相互之间的关系称为**结构**。

- 逻辑结构和物理结构
- 数据之间的相互关系称为**逻辑结构**。通常分为四类基本结构：

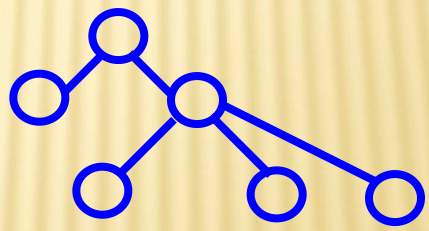
1.2 基本概念和术语

一、**集合** 结构中的数据元素除了同属于一种类型外，别无其它关系。

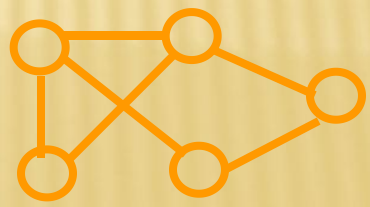


二、**线性结构** 结构中的数据元素之间存在一对一的关系。

三、**树型结构** 结构中的数据元素之间存在一对多的关系。



四、**图状结构或网状结构** 结构中的数据元素之间存在多对多的关系。



1.2 基本概念和术语

- 概括地说，数据结构分为线性结构和非线性结构。
- 线性结构包括：线性表、向量、栈、队列、链表、字符串
- 非线性结构包括：树、图、多维数组、稀疏矩阵等。

1.2 基本概念和术语

- 数据结构的形式定义为：数据结构是一个二元组：

  Data-Structure=(D，S)

其中：D是数据元素的有限集，S是D上关系的有限集。

- 例　复数的数据结构定义如下：

  Complex=(C，R)

其中：C是含两个实数的集合{C1，C2}，分别表示复数的实部和虚部。R={P}，P是定义在集合上的一种关系{〈C1，C2〉}。

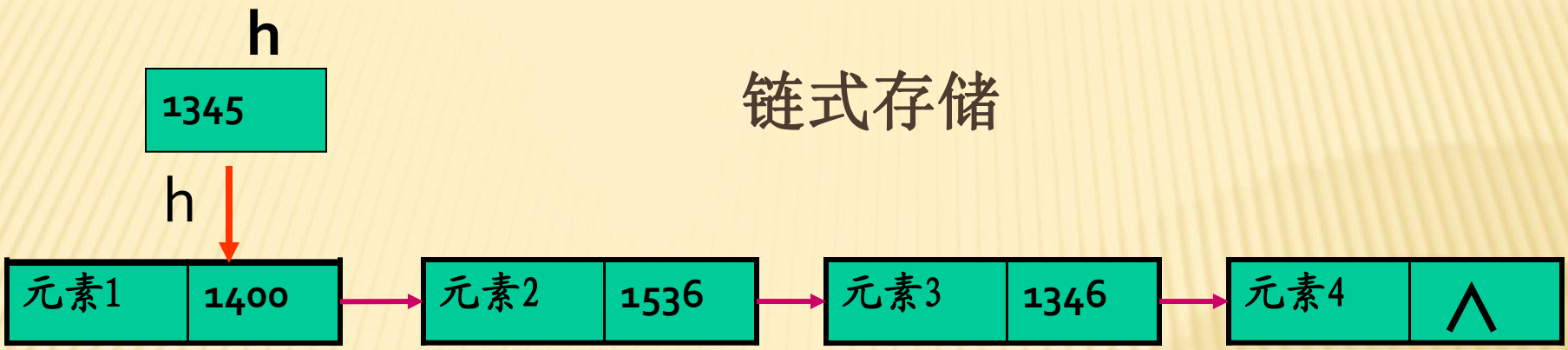
1.2 基本概念和术语

- 数据结构在计算机中的表示称为数据的**物理结构**，又称为**存储结构**。
- 数据结构在计算机中有两种不同的表示方法：
  顺序表示和非顺序表示
- 由此得出两种不同的存储结构：**顺序存储结构**和**链式存储结构**
- **顺序存储结构:**用数据元素在存储器中的相对位置来表示数据元素之间的逻辑关系。
- **链式存储结构：**在每一个数据元素中增加一个存放地址的指针，用此指针来表示数据元素之间的逻辑关系。

顺序存储

Loc(元素i)=Lo+（i-1)*m

链式存储



存储地址	存储内 容	指针
1345	元素1	1400
1346	元素4	∧
…….	……. .	…….
1400	元素2	1536
…….	……. .	…….
1536	元素3	1346

数据结构的三个方面：

- 数据的逻辑结构
  - 线性结构
    - 线性表
    - 栈
    - 队
  - 非线性结构
    - 树形结构
    - 图形结构
- 数据的存储结构
  - 顺序存储
  - 链式存储
- 数据的运算：检索、排序、插入、删除、修改等

1.2 基本概念和术语

- **数据类型**：在一种程序设计语言中，变量所具有的数据种类。
- 例　在C语言中

数据类型：基本类型和构造类型

基本类型：整型、浮点型、字符型

构造类型：数组、结构、联合、指针、枚举型、自定义

- **数据对象**：某种数据类型元素的集合。
- 例3、整数的数据对象是{…-3，-2，-1，0，1，2，3，…}

英文字符类型的数据对象是{A，B，C，D，E，F，…}

1.2 基本概念和术语

- 数据对象可以是有限的，也可以是无限的。
- 数据结构不同于数据类型，也不同于数据对象，它不仅要描述数据类型的数据对象，而且要描述数据对象各元素之间的相互关系。

1.3 抽象数据类型的表示和实现

- **抽象数据类型（ADT）**：指一个数学模型以及定义在该模型上的一组操作。
- ADT有两个重要特征：
- **数据抽象**：用ADT描述程序处理的实体时，强调的是其本质特征，其所能完成的功能以及它和外部用户的接口（即外界使用的方法）。
- ADT的第2个特征：**数据封装**，将实体的外部特征和内部实现细节分离，并且对外部用户隐藏其内部实现细节。

- 和数据结构的形式相对应，抽象数据类型可用以下三元组表示:
- （D，S，P）
- D---数据对象
- S---D上的关系集
- P---对D的基本操作集

1.4 算法和算法分析

1.4.1 算法

- **算法**：是对特定问题求解步骤的一种描述

算法是指令的有限序列，其中每一条指令表示一个或多个操作。

- 算法具有以下五个特性：

1）**有穷性** 一个算法必须总是在执行有穷步之后结束，且每一步都在有穷时间内完成。

2）**确定性** 算法中每一条指令必须有确切的含义。不存在二义性。且算法只有一个入口和一个出口。

3）**可行性** 一个算法是可行的。即算法描述的操作都是可以通过已经实现的基本运算执行有限次来实现的。

4）**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集合。

5）**输出** 一个算法有一个或多个输出，这些输出是同输入有着某些特定关系的量。

1.4 算法和算法分析

算法：是对特定问题求解步骤的一种描述
算法是指令的有限序列，其中每一条指令表示一个或多个操作。

1.4 算法和算法分析

- 例：选择排序（说明如何把一个具体问题变为一个算法，采用自顶向下，逐步求精的结构化程序设计方法）
- 首先，**问题定义**：把存放在一个整数数组中的n个乱七八糟的数据按自小到大的顺序排列起来。可能有的数据具有相同的值，因此，最后的排列结果应是数据的非递减排序。
- 其次，**考虑解决方案**：n个数据存放在数组a[0]到a[n-1]中，需要一个个排列。先考虑第一个数据，存于a[0]，从a[0]到a[n-1]中选择一个最小的，把它交换到a[0]中，然后考虑第2个元素，从a[1]到a[n-1]中选择一个最小的并交换位置。当第n-1个数据，排在它应在的位置上时，第n个数据可以不再排了。

1.4 算法和算法分析

5	8	9	7	2	3	6
2	8	9	7	5	3	6
2	3	9	7	5	8	6
2	3	5	7	9	8	6
2	3	5	6	9	8	7
2	3	5	6	7	8	9
2	3	5	6	7	8	9

1.4 算法和算法分析

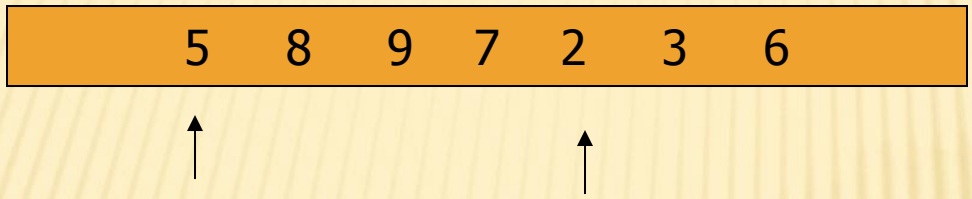
- 根据以上思路，写出算法的框架：

```
for ( int i=0;i<n-1;i++)
{
    从a[i]检查到a[n-1];
    若最小的数在a[k]，交换a[i]和a[k];
}
```

- 然后，将其细化。这时需要解决两个问题：一是如何选择值最小的数据；二是如何交换两个数据的值。

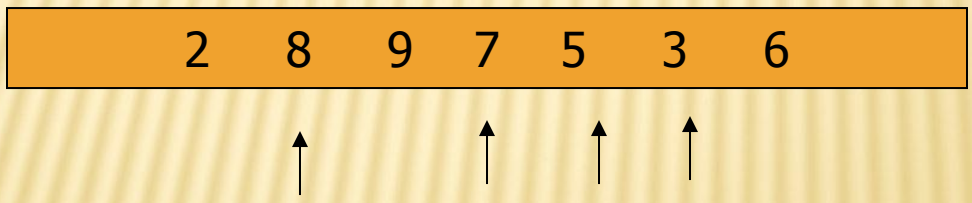
1.4 算法和算法分析

- 从第i 个元素到第n-1个元素中选择最小值元素可以采取以下做法:
- 先假定第i 个元素值最小，用k标示，然后顺序检查第i+1，i+2，…n-1，若检测到还有比刚才最小的还要小的元素，用k标示，在检查结束后k标示的就是值最小的数据。



K=0，a[k]=5　　K=4，a[k]=2

交换两个数据中的值



K=1，a[k]=8

```
Void selectsort(int a[ ],const int n)
{
   for (int i=0; i<n-1; i++)
     {
        int k=i;
        for (int j=i+1;j<n;j++)
           {
              if ( a[j]<a[k])    k=j;
            }
        int temp=a[i];a[i]=a[k];a[k]=temp;
      }
}
```

1.4.3 算法效率的度量

- 度量一个程序的执行时间通常有两种方法：
- 事后测试 收集此算法的执行时间和实际占用空间的统计资料。√
- 事先分析 求出该算法的一个时间界限函数

算法选用何种策略；
问题的规模；
书写程序的语言；
编译程序所产生的机器代码的质量；
机器执行指令的速度。

一般情况下，算法中基本操作重复执行的次数是问题规模n的某个函数，算法的时间量度记作

T(n)=O(f(n))

称作算法的渐近时间复杂度。

例

```
void Mult_matrix( int c[ ][ ], int a[ ][ ], int b[ ][ ]，  int n)
{
for(i=1,i<=n;++i)
                for(j=1;j<=n;++j)
                 {
                    c[i][j]=0;
                    for(k=1;k<=n;++k)
                           c[i][j]+=a[i][k]*b[k][j];
                  }
}// Mult_matrix
```

- 由于是一个三重循环，每个循环从1到n，则总次数为: $n \times n \times n = n^3$
- 时间复杂度为 $T(n)=O(n^3)$

- “原操作”指的是固有数据类型的操作，显然每个原操作的执行时间和算法无关，相对于问题的规模是常量。
- 在算法中考虑“起主要作用”的原操作即可，称这种原操作为“**基本操作**”
- 语句频度：是指该语句重复执行的次数

- 例 {++x;s=0;}
- 将x自增看成是基本操作，则语句频度为1，即时间复杂度为O(1)
- 如果将s=0也看成是基本操作，则语句频度为2，其时间复杂度仍为O(1)，即常量阶。

- 例、for(i=1;i<=n;++i)
- {++x;s+=x;}
- 语句频度为：2n　其时间复杂度为：O(n)
- 即时间复杂度为线性阶。

例、for(i=1;i<=n;++i)

for(j=1;j<=n;++j)

{++x;s+=x;}

语句频度为：$2n^2$

其时间复杂度为：$O(n^2)$ 即时间复杂度为平方阶。

定理：若$A(n)=a_m n^m+a_{m-1} n^{m-1}+\ldots+a_1 n+a_0$是一个m次多项式，则$A(n)=O(n^m)$

例 for(i=2;i<=n;++i)

for(j=2;j<=i-1;++j)

{++x;a[i][j]=x;}

语句频度为:

$1+2+3+\ldots+n-2=(1+n-2)\times(n-2)/2$

$=(n-1)(n-2)/2$

$=n^2-3n+2$

∴时间复杂度为$O(n^2)$

即此算法的时间复杂度为平方阶.

- 一个算法时间为O(1)的算法，它的基本运算执行的次数是固定的。因此，总的时间由一个常数（即零次多项式）来限界。而一个时间为$O(n^2)$的算法则由一个二次多项式来限界。
- 以下六种计算算法时间的多项式是最常用的。其关系为：

$O(1)<O(\log n)<O(n)<O(n\log n)<O(n^2)<O(n^3)$

- 指数时间的关系为：

$O(2^n)<O(n!)<O(n^n)$

- 当 n 取得很大时，指数时间算法和多项式时间算法在所需时间上非常悬殊。因此，只要有人能将现有指数时间算法中的任何一个算法化简为多项式时间算法，那就取得了一个伟大的成就。

- 有的情况下，算法中基本操作重复执行的次数还随问题的输入数据集不同而不同。例如：

```
Void bubble-sort(int a[]，int n)
    for(i=n-1;change=TURE;i>1 && change;--i)
    {
        change=false;
        for(j=0;j<i;++j)
            if (a[j]>a[j+1]) {
                a[j] ←→a[j+1];
                change=TURE}
    }
```

- 最好情况：0次
- 最坏情况：1+2+3+…+n-1=n(n-1)/2
-

- 对这类算法的分析，一种解决的办法是计算它的平均值，即平均时间复杂度。Bubble排序平均时间复杂度为:O(n^2)
- 另一种更可行也更常用的方法是讨论算法在最坏情况下的时间复杂度。
- 例如bubble排序的最坏情况为a中初始序列为自大至小排序，则冒泡排序算法在最坏情况下的时间复杂度为
- T（n）=O（n^2）

1.4.4 算法的存储空间需求

- 空间复杂度:算法所需存储空间的度量，记作:
- $S(n)=O(f(n))$
- 其中n为问题的规模(或大小)

复习

- **【重点和难点】**

  本章讨论的都是一些基本概念，因此没有难点，重点在于了解有关数据结构的各个名词和术语的含义，以及语句频度和时间复杂度、空间复杂度的估算。

- **【知识点】**

  数据、数据元素、数据结构、数据类型、抽象数据类型、算法及其设计原则、时间复杂度、空间复杂度

- **【学习指南】**

  1. 熟悉各名词、术语的含义，掌握基本概念，特别是数据的逻辑结构和存储结构之间的关系。分清哪些是逻辑结构的性质，哪些是存储结构的性质。

  2. 理解算法五个要素的确切含义和对算法正确性的理解。

  3. 掌握计算语句频度和估算算法时间复杂度的方法。

作业

- 1. 简述下列术语：数据、数据元素、数据对象、数据结构、存储结构、数据类型和抽象数据类型。
- 2.设n为3的倍数，试分析以下程序段中第②、③、④语句的语句频度。

```
①for i=1 to n do
②     if 3*i <=n  then
③          for j=3*i  to n do
④               { x=x+1; y=3*x+2;}
```