数据结构与算法

第 1 章 概论

主讲：赵海燕

北京大学信息科学技术学院

“数据结构与算法”教学组

国家精品课“数据结构与算法”

http://www.jpk.pku.edu.cn/pkujpk/course/sjjg/

张铭，王腾蛟，赵海燕

高等教育出版社，2008. 6, “十一五”国家级规划教材

内容提要

- 问题求解
- 数据结构及抽象数据类型
- 算法的特性及分类
- 算法的效率度量
- 数据结构的选择和评价

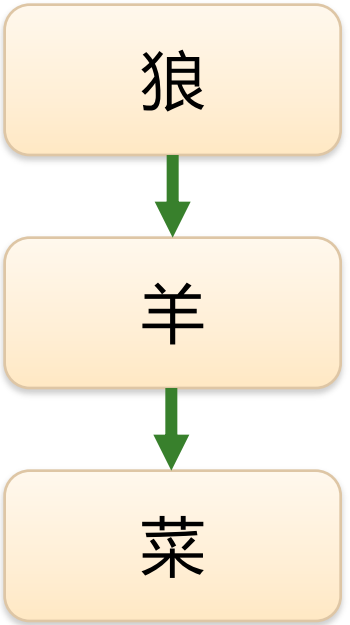
问题 & 问题求解

- *Problem*
  - *Oxford English Dictionary* as
    - a question proposed for solution or consideration
  - *Encyclopedia Britannica* as
    - a kind of thinking that facilitates question answering
- *Problem solving*
  - as *a form of thinking* that answers questions has been widely studied by many who believe that *thinking is the primary mechanism* for human *understanding and improving the world*

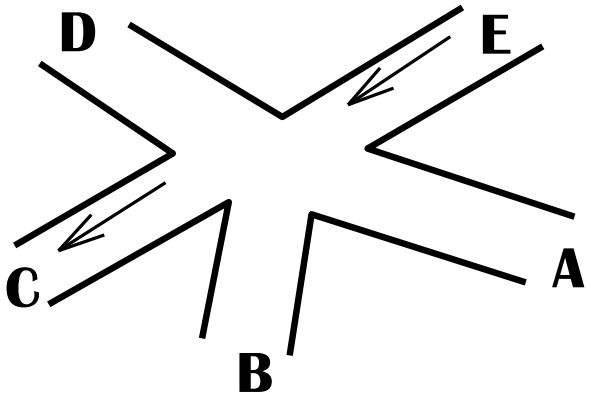
问题求解

- 实例
  - 从一组人中找出最高、最矮，及身高最适中的人
  - 12个外表完全相同的球中有一不标准，或轻或重，要求用天平以最少的次数找出该球，并判定其轻重
  - 计算机专业课程排课表
  - 农夫过河
  - 多叉路口交通灯管理问题
  - 智能交通

农夫过河



多叉路口交通灯管理问题

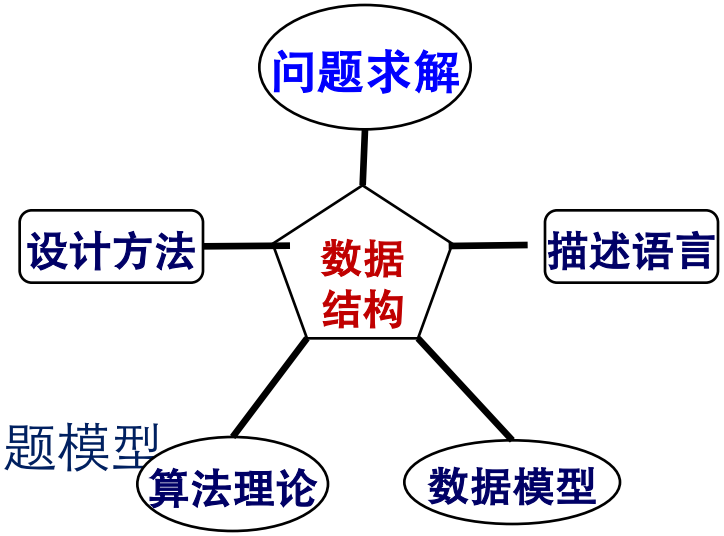


- 五叉路口
  - 右行规则
  - 道路C、E是箭头所示的单行道
- 可以同时行驶而不发生碰撞的路线用一种颜色的交通灯指示
- 用多少种颜色的交通灯，怎样分配给这些行驶路线?
  - 颜色越少则管理效率越高
  - 不考虑过渡灯（例如黄灯）

问题求解

- 阶段和步骤
  - 获取需求（问题），以保证解决的问题正是需要解决的（*solve the right problem*）；
  - 分析问题，已知/未知，输入/输出；将其分解为粒度更小的部分；
  - 针对问题（子问题）给出相应的解决方案，易于理解和修改；（*solve the problem right*）；
  - 估算解决方案的开销，以事先判断其可行性和可带来的收益（*solve the problem efficiently*）；
    - 利用数学等工具的辅助得到正确且简洁的解决方案
  - 维护和演化

利用计算机的问题求解



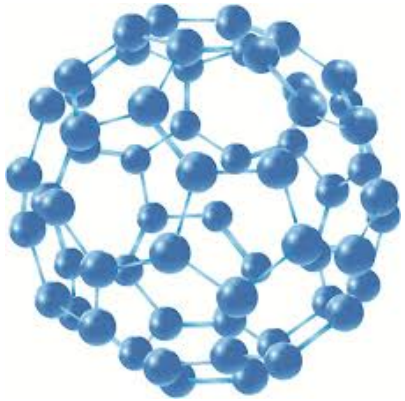
- 问题求解 编程的目标
  - 抽象和建模
    - 问题抽象
      - 分析和抽象任务需求，建立问题模型
    - 数据抽象
      - 确定恰当的数据结构表示数学模型
    - 算法抽象
      - 在数据模型的基础上设计合适的算法
  - 数据结构 + 算法； ➔ 程序设计
    - 模拟和解决实际问题

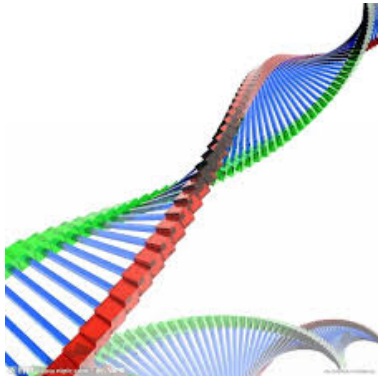
利用计算机的问题求解

- 计算机科学的不同表述
  - “信息结构转换的科学” (P. Wegner)
  - “算法的学问” (D. Knuth)
- 实质
  - 描述问题域中实际对象的数据及其相互关系
  - 映射数据及其关系到计算机的存储器上
  - 编写算法模拟对象领域中的求解过程

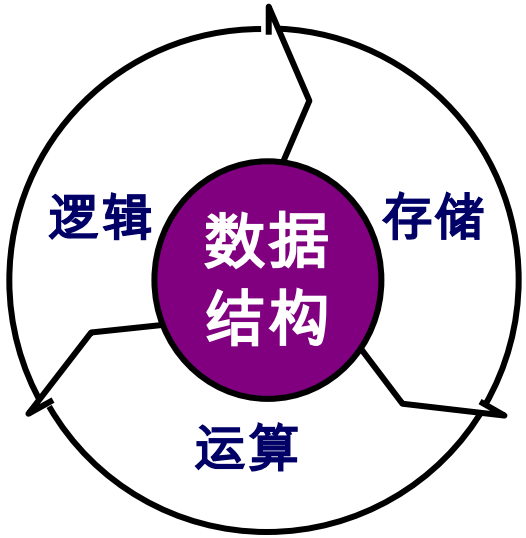
数据结构

数据结构





- 结构： 实体 + 关系
- 数据结构：
  - **逻辑**： 按照逻辑关系组织起来的一批数据
  - **存储**： 按一定的存储方法存储在计算机中
  - **运算**： 在这些数据上定义了一个运算的集合



常见的基本数据结构：线性表，字符串，栈与队列，树与二叉树，字典，图

数据的逻辑结构

- 二元组 B = (K, R)
  - K：结点（初等或组合类型）的有限集合
  - R：K上的有穷关系的集合（一组二元关系）

  - K中每个结点都代表一个数据或一组有明确结构的数据
  - 关系集 R中的每个关系(relation) r（r ∈ R）都是 K×K 上的二元关系，描述结点之间的逻辑关系
    - 例如，$r = \{<k_{i-1}, k_i> \mid k_i \in K, 1 < i < n\}$

结点类型：基本数据类型

- 整数 (integer)
- 实数 (real)
- 布尔 (boolean)
  - C++语言中0表示false，非0表示true
  - 也支持 false，true 保留字
- 字符 (char)
  - ASCII用单个字节（最高位委为0）表示字符
  - 汉字符号需用2个字节（每个字节最高位bit为1）的编码
  - Unicode，GB，Big5，HZ

结点类型： 指针类型

- 指针 (pointer)：用于表示机器内存地址，指向某一内存单元的地址
  - 32bit机器，4个字节表示一个指针
  - 64bit的机器，8个字节指针
- 指针操作
  - 分配地址
  - 赋值（另一个指针的地址值，NULL空值）
  - 比较两个指针地址
  - 指针增减一个整数量

结点类型： 复合数据类型

- 基本数据类型/复合类型组成的复杂结构
  - 例如：
    - 数组： int A[100];
    - 结构： typedef struct {} B;
    - 类： class C { };

- 复合数据类型本身，又可以参与定义结构更为复杂的结点类型

结构的分类

- 数据的逻辑结构（K, R）的讨论，重点在于数据的关系R上

- 根据 R 的性质 刻划数据结构的特点，并对数据结构进行分类：
  - 线性结构（linear structure）
  - 树结构（tree structure）
  - 图结构（graph structure）

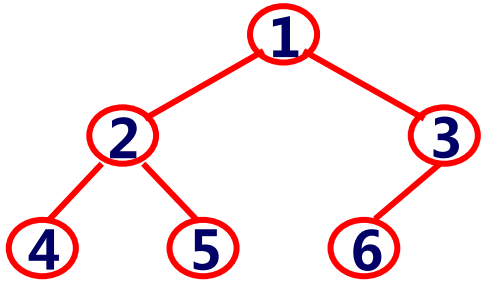
结点和结构

- 数据结构的设计可采用自顶向下的方法逐层进行
  - 先明确数据结点，及其主要关系 *r*
  - 分析关系 *r* 的同时，也要分析其数据结点的数据类型
  - 若其中数据结点的逻辑结构比较复杂，那么可将其作为下一个层次，进一步分析下一层次的逻辑结构

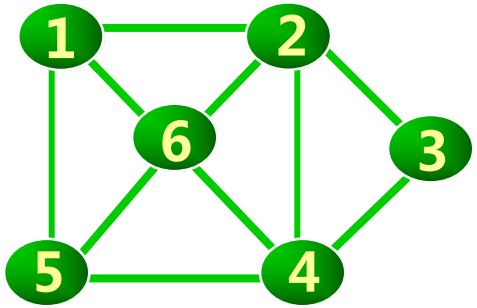
数据结构的逻辑组织

- 线性结构
  - 线性表（表，栈，队列，串等）

- 非线性结构
  - 树（二叉树，Huffman树，二叉检索树等）

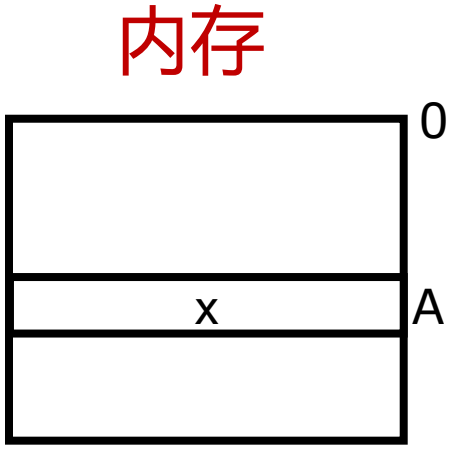


  - 图（有向图，无向图等）



数据的存储结构

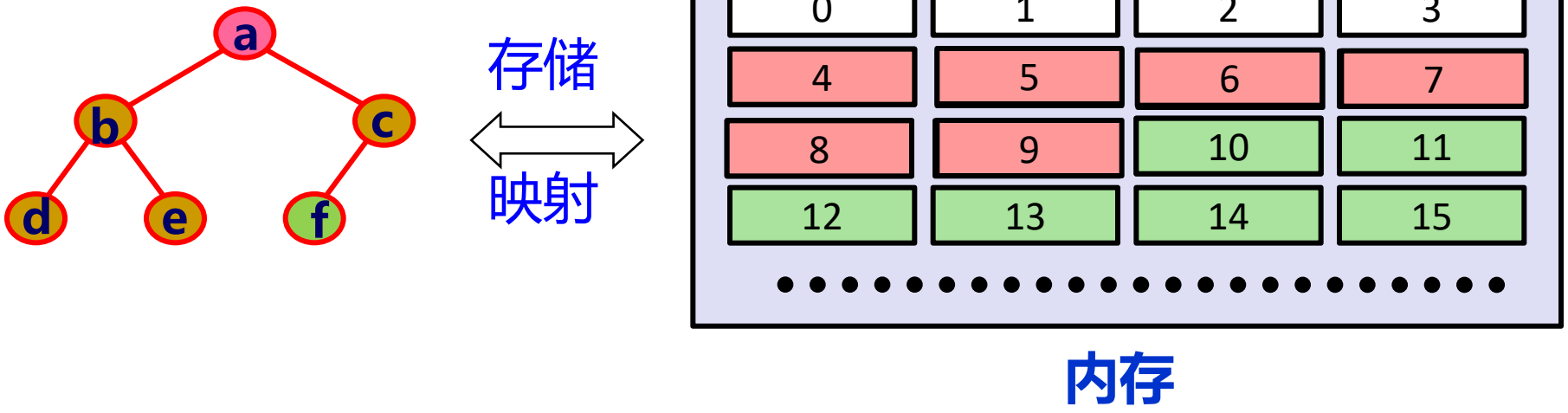
- **数据的存储**
  - 主存储器
  - 外存储器



- **计算机主存储器的特性**
  - 非负整数地址编码，相邻单元的集合
  - 基本单位是字节
  - 按地址随机访问
  - 访问不同地址所需时间基本相同

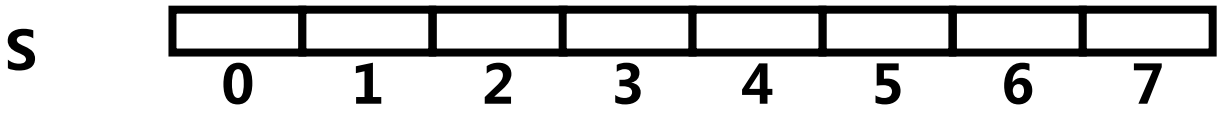
数据的存储结构

- 建立到物理内存一种映射
- 对于逻辑结构（K，r），其中 r ∈ R
  - 对结点集合 K 建立一个从K 到存储器 M 的单元的映射：K→M，对于每一个结点 j ∈ K 都对应一个唯一的连续存储区域c ∈ M

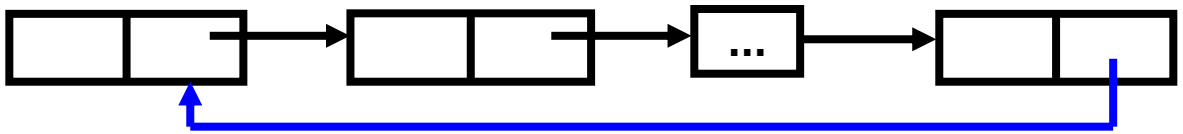


数据的存储结构

- 每一个关系元组（j_1 , j_2） $\in$ r（其中j_1， $j_2 \in$ K是结点）的映射
  - 顺序： 映射为存储单元的地址之间的顺序关系



  - 链接： 指针指向关系

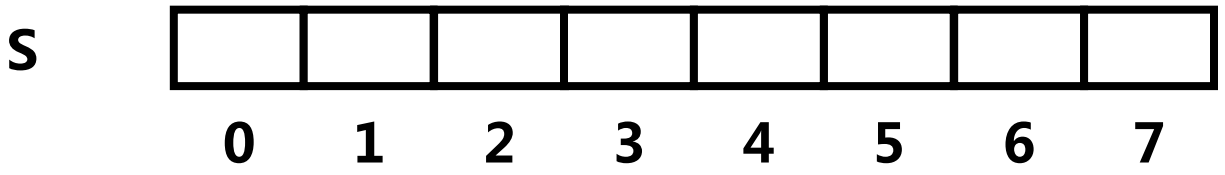


- 四类基本的存储映射方法： 顺序、链接、索引、散列

顺序方法

- 把结点存储在一块连续的按地址相邻的顺序存储单元中
  - 存储单元的顺序本身表达结点间的逻辑后继关系

- 数组（向量）：　**按下标访问**

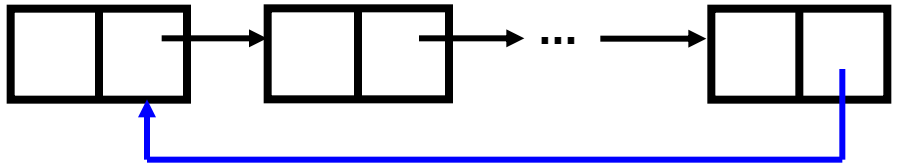


顺序方法

- 紧凑存储结构，即，存储空间除了存储有用数据外，没有用于存储其他附加的信息
  - **存储密度**：一个存储结构所存储的“有用数据”和该结构（包括附加信息）整个存储空间大小之比
  - 用空间换取时间 时，存储结构中存储一些附加信息是必要的
    - 提高算法的执行速度，或让算法实现更为简洁等

链接方法

- 结点的存储结构中附加指针字段
  - **数据域**：用于存放结点本身的数据
  - **指针域**：表达结点间的逻辑关系，某结点逻辑后继结点所在存储单元的开始地址，形成链索/链表



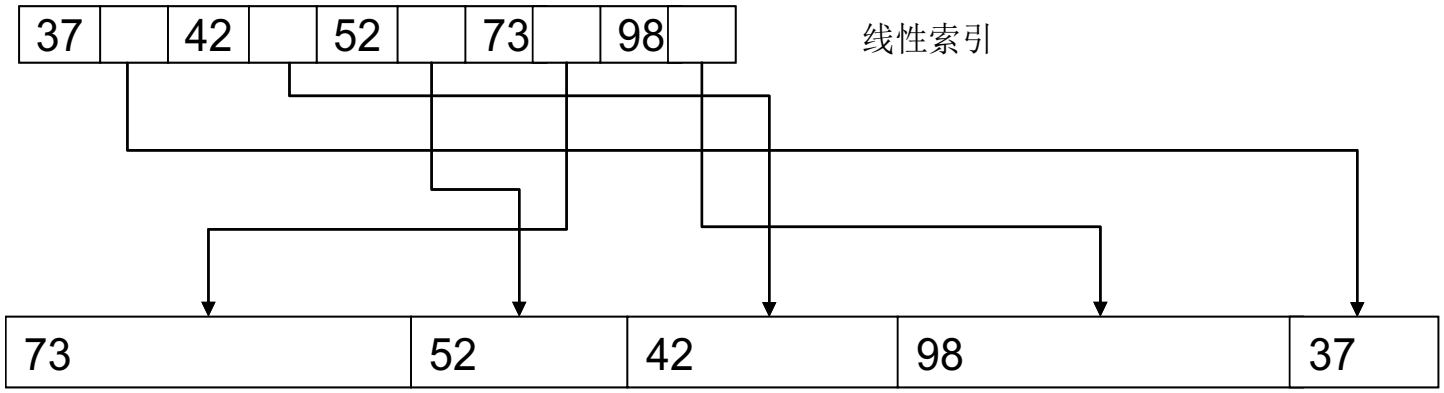
- **存储密度** $\rho < 1$

$$\rho = n \times E / n(P+E) = E/(P+E)$$

（n表示线性表中当前元素的数目）

索引方法

- 顺序存储的推广，使用整数编码来访问数据结点位置，索引映射函数Y：Z→D（往往并非简单的线性函数）
  - 结点的**索引值** z ∈ Z
  - 结点的**存储地址** d ∈ D



索引方法

- 数据表 + 索引表
- 需付出存储开销，数据结点附加用于存储指针的空间
  - 稠密索引
  - 稀疏索引
- 程序设计中经常采用，因为对于非顺序的存储结构来说，使用索引表是快速地由整数索引值找到其对应数据结点的唯一方法

散列方法

- **索引的一种延伸和扩展**
- 由散列函数（hash functions）的机制进行索引值的计算：将关键码映射到非负整数 z 的

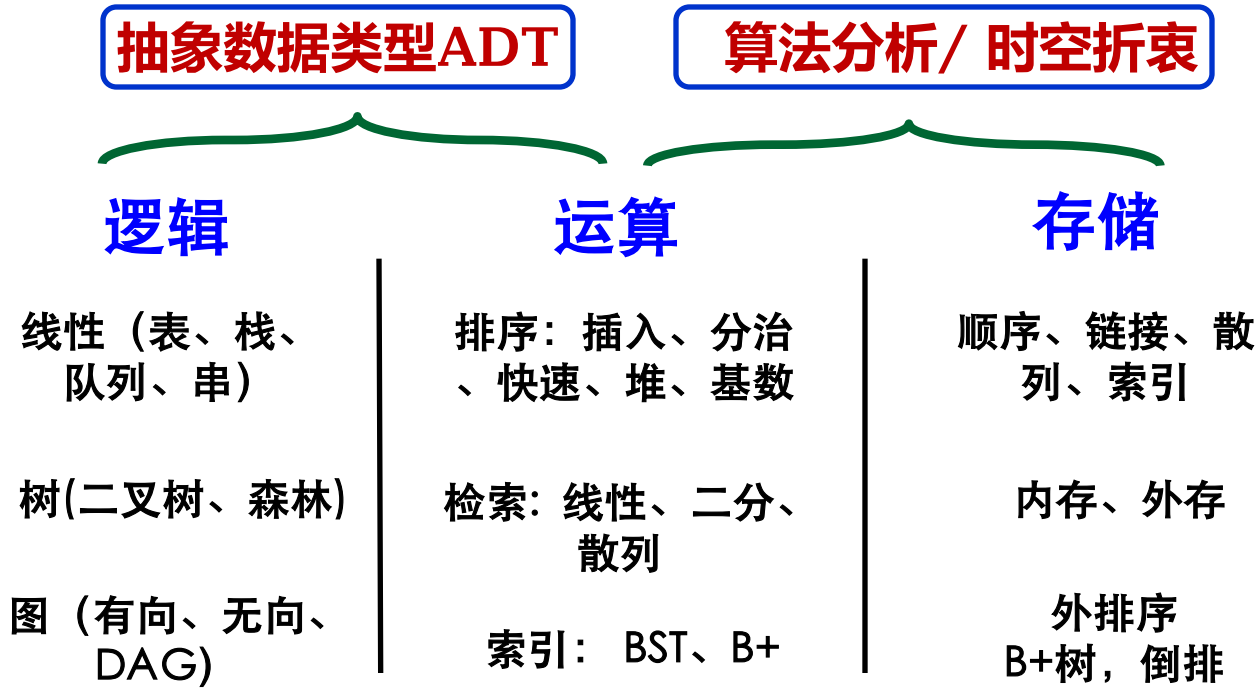
  $h: K \rightarrow Z$,

  对任意的 $k \in K$，散列函数 $h(k) = z$，$z \in Z$
- 关键
  - 如何恰当地选择散列函数
  - 如何建造散列表
  - 如何在构建散列表时解决“碰撞”问题

数据的运算

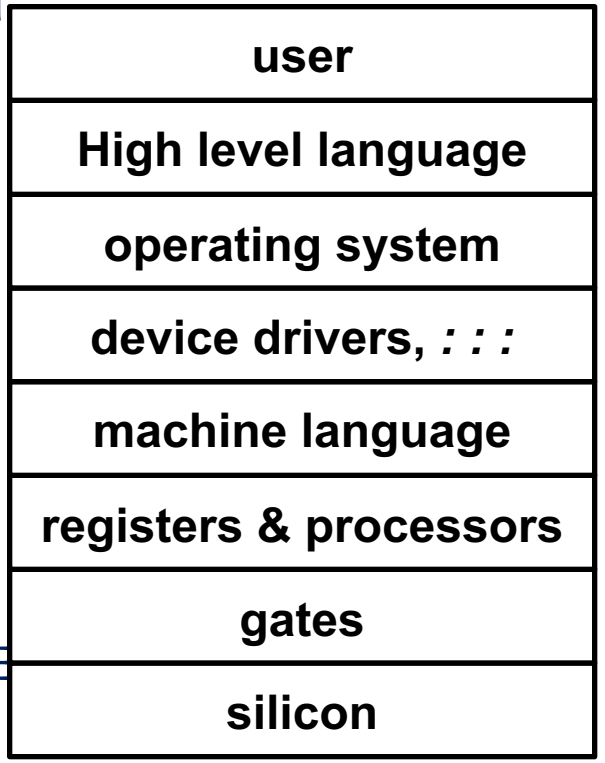
- 作用于数据上的运算
  - 查(检索）、改、增、删等
  - 排序
  - Indexing
  - ……
- 数据 + 运算
  - 不同数据
  - 不同运算

数据结构


抽象数据类型ADT
算法分析/ 时空折衷
逻辑
运算
存储
线性（表、栈、队列、串）
树(二叉树、森林)
图（有向、无向、DAG)
排序：插入、分治、快速、堆、基数
检索：线性、二分、散列
索引： BST、B+
顺序、链接、散列、索引
内存、外存
外排序
B+树，倒排

抽象

- 计算机科学本身就是抽象的科学 —— 为问题建立适当的模型并设计相应的技术解决之
  - 相对于物理学
- 计算机本身的一些抽象
- 抽象的本质?
  - 认识复杂现象所使用的思维方法
  - 简化、降低复杂度
  - 抽取出共同的、本质性特征，而忽略非本质的部分
    - **去粗取精、去伪存真**



抽象示例

摘自
《哥德尔、埃舍尔、巴赫》

- **报纸可能可被分成六个等级：**
  (1) 一个出版品
  (2) 一份报纸
  (3) 《旧金山纪事报》
  (4) 5月18日的《旧金山纪事报》
  (5) 我的5月18日的《旧金山纪事报》
  (6) 我　首次捡起时的　我的　5月18日的　《旧金山纪事报》（而现在不是我的了，因为我在几天后丢进火炉里烧了）

抽象数据类型

- **模块化思想的发展**
  - 隐藏运算实现的细节和内部数据结构
  - 软件复用（粒度 和 力度）

- **简称ADT (Abstract Data Type)**
  - 定义了一组运算的数学模型
  - 与物理存储结构无关
  - 使软件系统建立在数据之上（面向对象）

抽象数据类型

- 表示为三元组

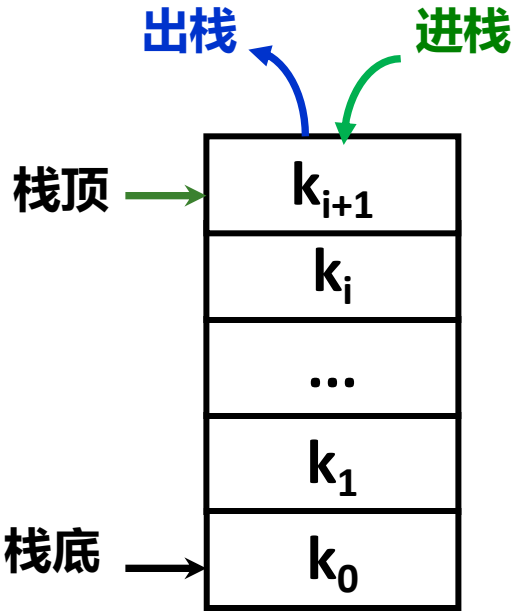
  <数据对象D，数据关系S，数据操作P>

- 逻辑结构 + 运算
  - 逻辑结构：数据对象 + 数据关系
  - 运算：数据操作

抽象数据类型

```
template <class Type>      // 模板参数为类型Type
class className {
private:                                  // 数据结构的的取值类型和取值空间
    Type  dataList;                       // 定义数据及其存储方式
    … …
public:                                   // 运算集
        methodName();                     // 定义对数据的操作
        ……
};
```

抽象数据类型：栈ADT



- 逻辑结构：线性表
- 操作特点：限制访问端口
  - 只允许在一端进行插入、删除操作
  - push、pop、top、 isEmpty

```
template <class T>              // 栈的元素类型为 T
class Stack {
public:                         // 栈的运算集
        void clear();                       // 变为空栈
        bool push(const T item);            // item入栈，成功返回真，否则假
        bool pop(T & item);                 // 弹栈顶，成功返回真，否则返回假
        bool top(T& item);                  // 读栈顶但不弹出，成功真，否则假
        bool isEmpty(;                      // 若栈已空返回真
        bool isFull();                      // 若栈已满返回真
};
```