从今天开始复习一遍408基础知识，并进行相关的拓展。今天是数据结构与算法的引言。一些基本概念。

1、基本概念

- **数据**：所有能够被计算机识别和处理的程序和符号集合。
    - 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。
- **数据元素**：是数据的基本单位，有若干个数据项组成。
    - 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。
- **数据对象**：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。
    - 例如整数数据对象是集合`N = {0, ±1, ±2···}`。
- **数据类型**：一个值的集合和定义在此集合上的一组操作的总称。
    - 原子类型：其值不可再分的数据类型
    - 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：抽象数据组织及与之相关的操作
- **数据结构**：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。
    - 在任何问题中，数据元素都不是孤立存在的，他们之间存在某种关系，这种数据元素相互之间的关系称为结构（Structure）。
    - **数据结构包括三方面的内容：逻辑结构、存储结构和数据的运算。**

2、数据结构的三要素

(1) 数据的逻辑结构

逻辑结构是指数据元素之间的逻辑关系，即从**逻辑关系上描述数据**。逻辑结构又分为线性结构和非线性结构。

> 数据的逻辑结构
>
> > 线性结构：结构中的数据元素之间只存在一对一的关系。
> >
> > > 一般线性表
> > >
> > > 受限线性表
> > >
> > > > 栈和队列
> > > >
> > > > 串
> > >
> > > 线性表推广
> > >
> > > > 数组
> > > >
> > > > 广义表
> >
> > 非线性结构
> >
> > > 集合：结构中的数据元素之间除“同属一个集合”外，别无其他关系。
> > >
> > > 树形结构：结构中的数据元素之间只存在一对多的关系。
> > >
> > > > 一般树
> > > >
> > > > 二叉树
> > >
> > > 图状结构：结构中的数据元素之间存在多对多的关系。
> > >
> > > > 有向图
> > > >
> > > > 无向图

(2) 数据的存储结构

- 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。
  - **优点：** 可以实现随机存取，每个元素占用最少的存储空间。
  - **缺点：** 只能使用相邻的一整块存储单元，因此可能产生较多的外部碎片。
- 链式存储：不要求逻辑上相邻的元素在物理位置上也相邻，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。
  - **优点：** 不会出现碎片现象，能充分利用所有存储单元。

- **缺点：** 每个元素因存储指针而占用额外的存储空间，且只能实现顺序存储。

- 索引存储：在存储信息的同时，还建立附加的索引表。索引表中的每项称为索引项，索引项一般形式是（关键字，地址）。
  - **优点：** 检索速度快。
  - **缺点：** 附加的索引表额外占用存储空间，增加和删除数据时也要修改索引表，因此话费更多的时间。
- 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。
  - **优点：** 检索、增加和删除结点的操作都很快。
  - **缺点：** 若散列函数不好，则可能出现元素存储单元的冲突，而解决冲突会增加时间和空间开销。

（3）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。

3、算法和算法的评价

（1）基本概念

算法（Alogorithm）是对特定问题求解步骤的一种描述，它是指令的有限序列，其中的每条指令表示一个或多个操作。因此，算法具有以下五个重要的特性： （1）**有穷性**：一个算法必总在执行又穷步之后结束，且每一步都可在有穷时间内完成。 （2）**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。 （3）**可行性**：算法中描述的操作都可以通过已经是实现的基本运算执行有限次来实现。 （4）**输入**：一个算法有零个或多个输入，这些输入曲子某个特定的对象的集合。 （5）**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。

设计一个好的算法应达到以下目标： （1）**正确性**：算法能够正确地解决求解问题。 （2）**可读性**：算法应该具有良好的可读性，以帮助人们理解。 （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，而不会莫名其妙的输出结果。 （4）**效率与低存储量需求**：效率是指算法执行的时间，存储量需求是指算法执行过程所需要的的最大存储空间，这两个都与问题的规模有关。

（2）算法效率的度量

算法效率的度量是通过时间复杂度和空间复杂度来描述的。

A. 时间复杂度：一个语句的频度是指该语句在算法中被重复执行的次数。

算法中所有的语句的频度之和记为`T(n)`，它是该算法问题规模`n`的函数，时间复杂度主要分析`T(n)`的**数量级。**

算法中基本运算（最深层循环内的语句）的频度与`T(n)`同数量级，因此通常采用算法中基本运算的频度`f(n)`来分析算法的时间复杂度。因此，算法的时间复杂度记为：

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

`O`的含义是`T(n)`的数量级，其严格的数学定义是：若`T(n)`和`f(n)`是定义在正整数集合上的两个函数，则存在正常数C和n_0，使得当$n >= n_0$时，都满足：

$$0 <= T(n) <= Cf(n)$$

算法的时间复杂度不仅依赖于问题的规模`n`，也取决于待输入数据的性质（如输入数据元素的初始状态）。

- **最坏时间复杂度**是指在最坏情况下，算法的时间复杂度。（一般总是考虑在最坏情况下的时间复杂度，以保证算法的运行时间不会比它更长。最常用）
- **平均时间复杂度**是指所有可能输入实例在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**是指在最好的情况下，算法的时间复杂度。

常见的渐进时间复杂度为：

$$O(1) < O(log2n) < O(nlog2n) < O(n^2) < O(n^3) < O(2^n) < O(n!) < O(n^n)$$

B. 空间复杂度

算法的空间复杂度`S(n)`定义为算法所耗费的存储空间，它是问题规模的函数。记为：

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

一个程序在执行时除需要存储空间来存放本身所用的指令、常数、变量和输入数据外，还需要一些对数据进行操作的工作单元和存储一些为实现计算所需信息的辅助空间。若输入数据所占空间只取决于问题本身，和算法无关，则只需分析除输入和程序之外的额外空间。

算法原地工作是指算法所需的辅助空间为常亮，即O(1)。