数据结构

绪论（二）

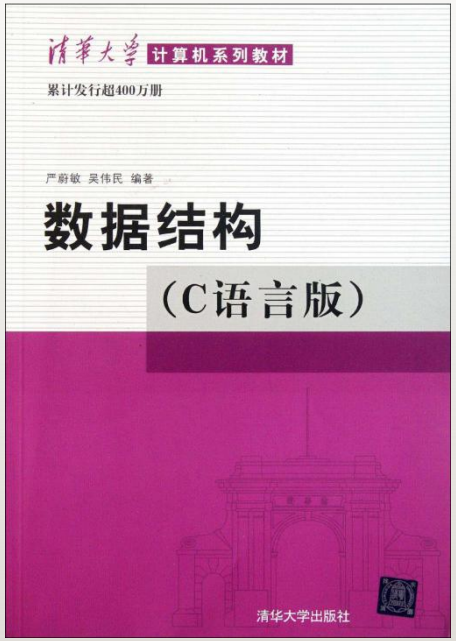
2021年09月

深圳大学电子信息工程学院 周飞

2 第一章 绪论

第三节 算法

数据结构vs算法

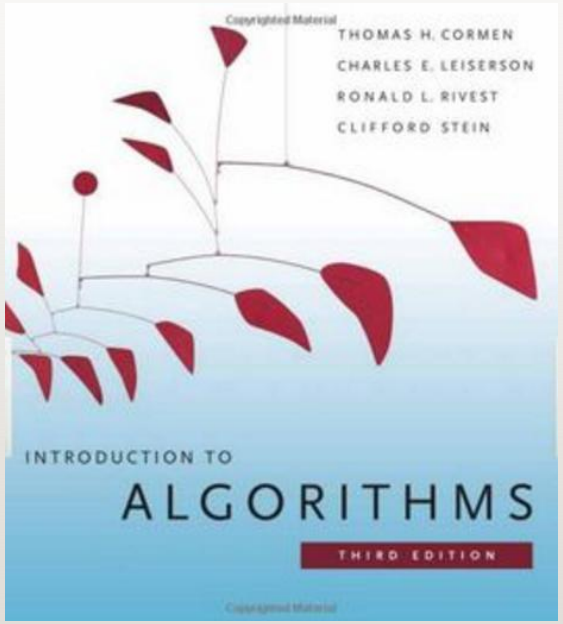


《数据结构》

- 查找 算法
- 排序 算法
- 翻看教材：各种算法

《算法导论》

- 数据结构
- 各种 树和图
- 堆栈、字符串，等等



第一章 绪论

第三节 算法

- 一、算法（Algorithm）
  - 算法是对特定问题求解步骤的一种描述，是指令的有限序列
  - 每条指令可以是一个或多个操作
  - 算法是一个长度有限的操作序列

第一章 绪论

第三节 算法

数据结构vs算法

数据结构

本质上是一个二元组{D,S}

算法

本质上是指令序列

从定义上看，他们区别是…

从定义上看，他们并没有半毛钱关系!!!!!!

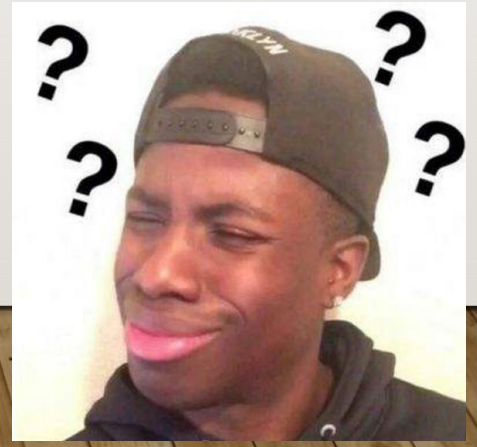
数据结构和算法有啥关系？

第一章 绪论

第三节 算法

- 二、数据结构和算法的关系
  - 算法+数据结构=程序设计
  - 算法以数据结构为基础
  - 没有数据间的关系，算法根本无法设计
  - 甚至有些数据结构就是为某类算法创造的

尼古拉斯·沃斯



第一章 绪论

第三节 算法

- 二、数据结构和算法的关系
  - 网购

商品的零部件：数据元素

搭建好的商品：数据结构

怎么使用商品来解决面对的问题：算法

第一章 绪论

第三节 算法

- 三、算法的特性
  - 1.有穷性:算法在执行有穷步后结束

最大操作次数

第一章 绪论

第三节 算法

- 三、算法的特性
  - 2.确定性每条指令的含义都是确切的，没有二义性

用自然语言或者伪代码描述一个算法时，尤其要注意！

第一章 绪论

第三节 算法

- 三、算法的特性
  - 3.可行性:操作都是可实现的，即可以通过已经实现的基本运算执行有限次实现

在工程上，算法的可行性一般指在现有的计算资源和合理的时间内可行。

第一章 绪论

第三节 算法

- 三、算法的特性
  - 4.输入：有0个或者多个输入
  - 5.输出：有1个或者多个输出

0个输入的算法？

让计算机列举素数：

2^77232917−1

2017 年 12 月 26 日

2^82589933-1

2018 年 12 月 7 日

第一章 绪论

第三节 算法

- 三、算法设计的要求
  - 有些算法比较优秀，有些算法则不太优秀

怎么评价算法？

第一章 绪论

第三节 算法

- 四、算法设计的要求
  - 1．正确性：满足具体问题的需求
    - 4个层次：
    - a)程序不含语法错误；
    - b)对于几组输出数据，能够得到满足要求的输出
    - c)对于精心选择的刁难型数据，能够得到满足要求的输出
    - d)对于所有数据，能够得到满足要求的输出

第一章 绪论

第三节 算法

- 四、算法设计的要求
  - 2．健壮性：对于非法输入，也能适当反应
    - 例如：一个算法用于判断一个数是否是素数
      - 期待的输入：实数
      - 实际的输入：字符

健壮性差，通俗地说，就是BUG多！

让程序员最痛苦的就是修BUG！

第一章 绪论

第三节 算法

- 四、算法设计的要求
  - 3．可读性：便于理解和修改

让程序员最痛苦的就是修BUG！

比修BUG更痛苦的是

修别人的BUG！

第一章 绪论

第三节 算法

- 四、算法设计的要求
  - 4．高效率和低存储量
    - 高效率：执行时间少
    - 低存储：执行中需要的最大存储空间小

第一章 绪论

第四节 算法分析

- 四、时间复杂度
  - 衡量算法的效率，主要依据算法执行所需要的时间，即时间复杂度
  - 事后统计法：计算开始时间和完成时间的差值
  - 事先分析法：撇开算法运行的软硬件环境，

分析算法的策略和问题的规模，

是评价算法时间复杂度的常用方法

第一章 绪论

第四节 算法分析

- 四、时间复杂度
  - 时间复杂度定义为问题规模n的函数f(n)，即

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

  - 什么是问题规模n？
    - 一般来说，可以理解为算法接受和处理的数据元素个数

问题规模就是队列的长度

第一章 绪论

第四节 算法分析

- 四、时间复杂度

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

O(f(n))意为f(n)阶 Order

$f_0(n) = n^2+2$　　$O(f_0(n)) = n^2$

$f_1(n) = 4n^3+ n^2$　　$O(f_1(n)) = n^3$

忽略低阶项！　　忽略倍数常量！

第一章 绪论

第四节 算法分析

- 四、时间复杂度

算法的策略和问题的规模

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

f(n)由算法的策略决定，
一般指算法中最深层循环内的基本操作重复执行的次数

第一章 绪论

第四节 算法分析

- 四、时间复杂度（举例）

```
int a[n]={1};
a[0] = a[0]*2;
```

问题规模为n

f(n)=1

算法时间复杂度：O(1)

常量阶

```
int a[n]={1};
for (i=1; i<=n; i++)
{a[i] = a[i] *2;
 a[i] =a[i] *3;}
```

问题规模为n

f(n)=2n

O(n)

线性阶

```
int a[n]={1};
for (i=1; i<=n; i++)
   for (j=1;j<=n;j++)
  {a[i] = a[i] *2;
   a[j] = a[j] *3;}
```

问题规模为n

$f(n)=2n*n = 2n^2$

$O(n^2)$

平方阶

第一章 绪论

第四节 算法分析

- 四、时间复杂度（举例）

```
int a[n]={1};
for (i=1; i<=n; i*=2)
{a[i] = a[i] *2;}
```

问题规模为n

$f(n)=\lfloor \log_2 n \rfloor +1$

算法时间复杂度：$O(\log_2 n)$

对数阶

第一章 绪论

第四节 算法分析

- 四、时间复杂度（举例）

下面程序段的时间复杂度是（ A ）。

```
i=s=0;
while(s<n){
    i++;s+=i;
}
```

A. $O(n^{1/2})$　　B. $O(n^2)$　　C. $O(\log_2 n)$　　D. $O(n^3)$

第一章 绪论

第四节 算法分析

- 四、时间复杂度

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

随着问题的规模增长快的算法，不宜使用！！

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

多项式时间内无法求解的算法一般被认为是不可行的！

第一章 绪论

第四节 算法分析

- 四、时间复杂度
  - 如果算法的执行有多种可能的操作顺序，则求其平均时间复杂度
  - 如果无法求取平均时间复杂度，则采用最坏情况下的时间复杂度

时间复杂度是衡量算法优劣的一个最重要的标准！

第一章 绪论

第四节 算法分析

- 四、空间复杂度
  - 空间复杂度指算法执行时，所需存储空间的度量。
  - 类似于时间复杂度，它也是问题规模的函数，即

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