# 00 Task 《算法导论》各章小结

Date：2024/03/16 2:02:32

---



---

## C01 算法在计算中的作用

- 概要
  - 本章介绍了算法（描述一个特定计算过程，实现所需的输入/输出关系）、数据结构（一种存储和组织数据的方式，旨在便于访问和修改）的含义，结合算法实际应用，列举各章涉及的不同算法能够解决的不同问题，这可建立一张对照图，以为实际问题的解决提供可靠的方向。其后，通过一种戏剧性的手法，比较了插入排序与归并排序效率的显著差异，指出 “应当把算法看作一种技术， 在很多看得见和看不见的地方，算法都是大多数技术的核心”，
  - 本章还介绍了NP完全问题，其中的未知性，使计算机科学家着迷，而如果一个问题并不确定是否存在最优解，或者并不存在已知的有效算法，那么给出一个近似解也很重要。
- 概念
  - 算法、数据结构
- 摘录
  - 对于数据结构，重要的是知道它们的**优势和局限**。
  - 是否具有算法知识与技术的坚实基础，是区分真正熟练的程序员与初学者的一个特征。
  - 有一个好的算法背景，你可以做的事情就多得多（决定上限）。
- 练习（P6-8）
- 2024/03/16 2:23:17

---

## C02 算法基础

- 概要
  - 本章介绍了一种重要的算法分析框架：“伪代码——算法说明——证明——分析”，实际上，最核心的内容是 **“循环不变式”**，其证明包含三步：
    - 初始化：循环第一次迭代之前，它为真；
    - 保持：如果循环的某次迭代之前它为真，那么下次迭代之前它仍为真；

    - 终止：在循环终止时，不变式为我们提供一个有用的性质，该性质有助于证明算法是正确的。
- 通过对插入排序的算法分析，展示了伪代码及循环不变式的证明过程，并指出这种证明方式的正确性。
- 其后，对**伪代码约定**，例如缩进、循环结构、条件结构（计数器保持）、注释、多重赋值、变量作用域（均为局部）、数组下标、复合数据类型（对象、属性，点标记法、串联记号）、参数传递（按值，可见性）、返回值（允许多值）、布尔运算符（短路）、异常处理。
- 进入分析算法的部分，假设了一种 **RAM 模型（随机访问机）**，是一种通用的单处理器计算模型。这部分十分巧妙地化繁为简，又不失关键部分地用一种 “够用” 的模型，讨论后续算法分析、运行的基本环境。意义：RAM模型分析**通常能很好地预测实际计算机上的性能。**关键模块包括：
    - 指令及代价：算法指令、数据移动指令、控制指令；对于每条指令所需时间均为常量；
    - 数据类型、精度取舍、数据存储（字）、索引功能；
    - 灰色区域
    - 内存层次（RAM 忽略该建模）
- 继续分析插入排序，以“步”的概念，统计了每一条指令的代价与执行次数，从而得到求和式。进而，比较了求和式在最佳情况、最坏情况、平均情况下的运行时间，由此引出了我们**真正关心的 “增长量级”** 概念，并引出了**非形式化**的记号 $\Theta$

| INSERTION-SORT($A$) | 代价 | 次数 |
|---|---|---|
| 1 **for** $j = 2$ **to** $A.length$ | $c_1$ | $n$ |
| 2 $key = A[j]$ | $c_2$ | $n-1$ |
| 3 // Insert $A[j]$ into the sorted sequence $A[1..j-1]$. | 0 | $n-1$ |
| 4 $i = j-1$ | $c_4$ | $n-1$ |
| 5 **while** $i > 0$ and $A[i] > key$ | $c_5$ | $\sum_{j=2}^{n} t_j$ |
| 6 $A[i+1] = A[i]$ | $c_6$ | $\sum_{j=2}^{n} (t_j - 1)$ |
| 7 $i = i-1$ | $c_7$ | $\sum_{j=2}^{n} (t_j - 1)$ |
| 8 $A[i+1] = key$ | $c_8$ | $n-1$ |

**该算法的运行时间是执行每条语句的运行时间之和。需要执行 $c_i$ 步且执行 $n$ 次的一条语句将贡献 $c_i n$ 给总运行时间[3]。为计算在具有 $n$ 个值的输入上 INSERTION-SORT 的运行时间 $T[n]$，我们将代价与次数列对应元素之积求和，得：**

$$T(n) = c_1 n + c_2(n-1) + c_4(n-1) + c_5 \sum_{j=2}^{n} t_j + c_6 \sum_{j=2}^{n} (t_j - 1) + c_7 \sum_{j=2}^{n} (t_j - 1) + c_8(n-1)$$

- 有了量级的比较，可以进一步探索量级更低的算法，例如 “分治法”，结构上递归，**每层递归**包含三步：
    - 分解：分解原问题为若干规模较小的子问题（重复性/自相似性）；
    - 解决：递归求解子问题，若子问题规模足够小，则直接解决；
    - 合并：子问题的解，组合为原问题的解。
- 归并排序完全遵循分治模式，着重讨论了 “合并” 步骤 $MERGE(A, p, q, r)$，并且一般化地讨论了 $n = r - p + 1$ 的范围，证明了其循环不变式的正确。随后将该模块组合为归并排序的主体，结合自底向上的图解，非常清晰地展示了归并过程。

```
MERGE(A, p, q, r)
 1  n1 = q − p + 1
 2  n2 = r − q
 3  let L[1..n1 + 1] and R[1..n2 + 1] be new arrays
 4  for i = 1 to n1
 5      L[i] = A[p + i − 1]
 6  for j = 1 to n2
 7      R[j] = A[q + j]
 8  L[n1 + 1] = ∞
 9  R[n2 + 1] = ∞
10  i = 1
11  j = 1
12  for k = p to r
13      if L[i] ≤ R[j]
14          A[k] = L[i]
15          i = i + 1
16      else A[k] = R[j]
17          j = j + 1
```

```
MERGE-SORT(A, p, r)
1  if p < r
2      q = ⌊(p+r)/2⌋
3      MERGE-SORT(A, p, q)
4      MERGE-SORT(A, q+1, r)
5      MERGE(A, p, q, r)
```

- 结合归并排序的例子，为避免使用后面章节的“主定理”，转而利用**图示与递归式重写**，分析了分治算法，这种证明思路十分清晰，非常值得模仿学习。（P21-22）

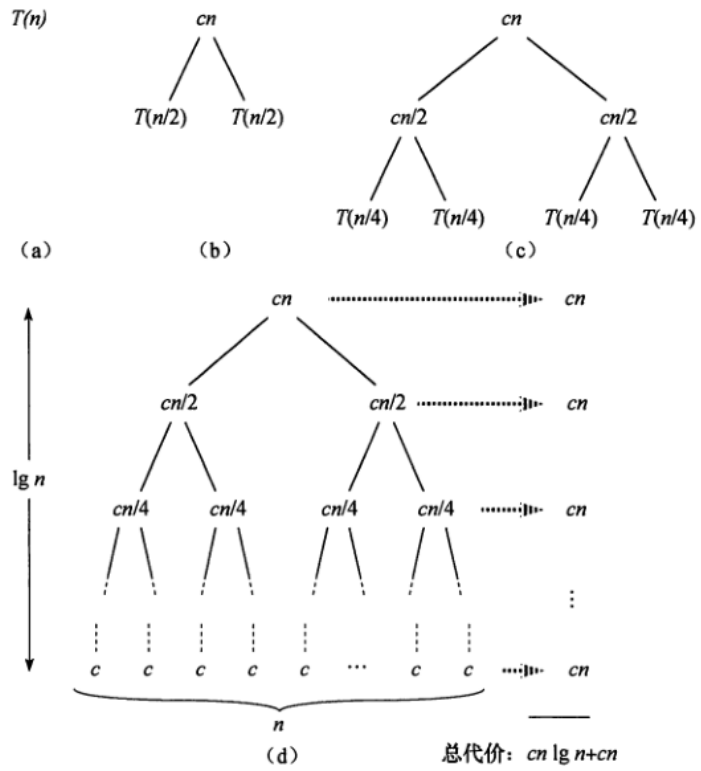


图 2-5　对递归式 $T(n)=2T(n/2)+cn$，如何构造一棵递归树。(a)部分图示 $T(n)$，它在(b)～(d)部分被逐步扩展以形成递归树。在(d)部分，完全扩展了的递归树具有 $\lg n+1$ 层(即如图所示，其高度为 $\lg n$)，每层将贡献总代价 $cn$。所以，总代价为 $cn\lg n+cn$，它就是 $\Theta(n\lg n)$

- 概念
  - 伪代码、关键词、插入排序、循环不变式、RAM模型、运行时间（“步” 单位）、循环测试次数（n + 1）、循环体执行次数（n）、线性函数、二次函数、最坏情况（上界）、平均情况（n / 2）、增长量级/增长率、$\Theta(n^2)$、增量方法、分治法、归并排序、辅助函数、哨兵、向上取整、向下取整、递归式/递归方程、递归树、树的高度、树的层数
- 摘录
  - RAM模型即使分析一个简单算法也可能是一个挑战。需要的数学工具可能包括组合学、概率论、袋代数技巧，以及**识别一个公式中最有意义的项的能力。**
- 练习（P12、16、22）
  - 选择排序、二分查找
- 思考题
  - 在归并排序中对小数组采用插入排序（使递归叶变粗）、冒泡排序的正确性（流行但低效，反复交换相邻的未按次序排列的元素）、霍纳（Horner）规则的正确性、逆序对（inversion）

```
BUBBLESORT(A)
1  for i = 1 to A.length − 1
2      for j = A.length downto i + 1
3          if A[j] < A[j − 1]
4              exchange A[j] with A[j − 1]
```

- 疑问 / 感悟
  - 如何理解 “对调用过程不可见” ？
  - 为什么循环不变式的证明，像是在说一种“显然”的废话？为什么这样的证明，可以确保其正确性？如何把握这种半形式化证明方式的精髓，例如真正理解其对正确性的保障的原理？
  - 对 RAM 模型的构建，提出了若干关键模块从复杂到简化的内容，思路十分清晰，那么是如何形成这种思考的？（从什么维度产生的？一种察觉到 “完成某个东西需要些什么” 的能力，仿佛游刃有余且心中有数。书中提到设计原则是 “根据真实计算机如何设计”）
  - 如何理解真实计算机未列出的指令，使得在 RAM 中包含的灰色区域？（由于是一种对真实计算机简化后的抽象模型，并未那么精确地考虑更多复杂的情况，例如 “指数运算是一条常量时间的指令吗？”）
  - **对于不清晰的公式，可以代入几个数去观察验证，例如递归树的层数和高度、伪代码里面的索引和取值范围。**
  - 假设 n 为 2 的幂，则递归树的层数比高度多 1，这就好像一栋三层高的房子，需要 4 层楼板，才能夹出三层的高，这里层数更强调构成的要素，高度更强调感官。
- 2024/03/18 17:14:30 1h24min

---

# C03 函数的增长