“高级算法设计与分析”课程学习报告

21215122, 何峙，大数据与人工智能

======================

作业：

1. 求解算法时间复杂度递归方程求解方法（PPT第1讲）

递推法

换元法

生成函数法

特征方程法

1. 分治与递归，整数划分问题（PPT第2讲）
2. n位大整数的乘法
3. 动态规划，完全加括号的矩阵连乘
4. 用动态规划求解矩阵链
5. 构建最优二叉树/ 说明矩阵连乘不满足四边形不等式条件
6. 证明贪心算法的性质（最优解）/ N皇后问题
7. 将N皇后递归问题改为迭代实现
8. 实现局部搜索算法具体应用

本课程把算法原理跟实际问题相结合，将原本枯燥的理论通过一些实际例子阐述明白，十分有趣！不是讲解一个个算法的实现过程，而是深入具体分析一类类算法背后的设计思想，并对算法的正确性和合理性进行论证，很具有启发性，对日常工作学习有很大帮助！现对课程内容进行总结性回顾，以加深对算法分析设计思想的理解。

1. 算法复杂度。

这是衡量算法优劣的重要指标。我们不仅要用算法解决问题，而且要越来越快的解决问题，这也是我们学习算法设计最主要驱动力。算法复杂度通常用数量阶表示，如我们熟悉的时间复杂度从小到大的排列依次是：O(1) < O(logn) < O(n) < O(nlogn) < O(n2) < O(n3) < O(2n) < O(n!) < O(nk)。我们可以跑代码，通过统计监控得出算法的执行时间，但这种统计方式具有很大局限性：1）测试结果受硬件的影响较大；2）测试结果受数据规模的影响很大。于是，课程介绍了一些快速分析出算法的时间复杂度的方法，如：

1. 递推法：利用递推式不断的循环解嵌套；
2. 换元法：通过代换参数可将复杂的递推式转换为较为简单的递推式；
3. 生成函数法：构造数列的生成函数，再利用数列的性质求解N的表达式方法；
4. 特征方程法：通过求解特征方程的根从而求解递推方程的解的方法；

这对快速衡量各种算法的执行效率很有帮助。了解了算法的复杂度，不难认识到随着算法的改进，相同时间内能够处理的数据量也大大提升。

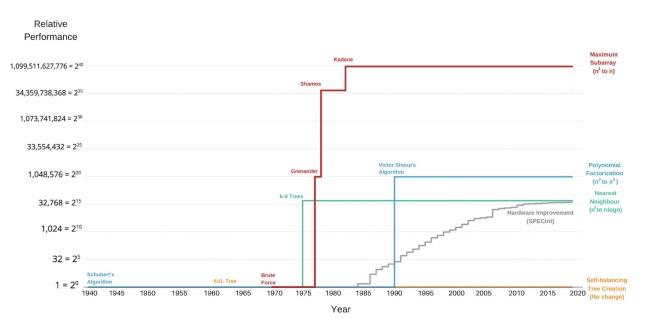


图1 随年份增长而获得改进的算法对于处理数据量的影响

（from MIT Computer Science & Artificial Intelligence Laboratory1）

图1列出的4个算法族，它们随着年份增长，获得了改进，处理的数据量呈直线式的上升。图中还标有根据摩尔定律的硬件（灰色线）随年份的改进趋势。对比算法的改进，硬件的改进对处理数据量的提升显得相当平滑。由此可得：

1. 算法改进带来的改变问题的可操作性，是硬件改进不能比拟的；
2. 随着数据量的大量提升，算法的改进比硬件的改进显得更加重要。

这些发现表明算法的改进在例如数据分析和机器学习等领域上相当重要，因为它们都依赖于大数据。

1. 分治与递归

分治是一种算法思想，从字面意思即可以理解：当一个问题规模较大且不易求解的时候，就可以考虑将问题分成几个小的模块，逐一解决。而递归是一种算法实践方法，通常跟迭代法相对比：迭代使用的是循环结构，递归则使用选择结构。递归的优点：能使程序的结构更清晰、更简洁、更容易让人理解，从而减少读懂代码的时间。其缺点：但大量的递归调用会建立函数的副本，会消耗大量的时间和内存，而迭代则不需要此种付出。

分治通常跟递归搭配使用会比较让人容易理解，其思维及使用步骤一般为：

（1）分解：将原问题分解为若干个规模较小，相互独立，与原问题形式相同的子问题；

（2）解决：若子问题规模较小而容易被解决则直接解，否则递归地解各个子问题；

（3）合并：将各个子问题的解合并为原问题的解。

例如作业中的整数划分问题：将正整数划分成一系列小的正整数之和。其递归方程为：

q(n, m) =

其中q(n, m)表示最大加数n不大于m的划分个数，只要我们将正整数划分为小整数的4种情况梳理出来，即可以利用递归方法顺利求解。

1. 动态规划
2. 贪心策略
3. 回溯与分枝界限策略
4. 局部与随机搜索策略
5. 遗传算法