

算法设计与分析开箱手册

该手册为面向计算学部的算法设计与分析的开箱手册。该手册仅用于同学初步了解该课程内容。一切内容以上课老师所讲为准，该手册仅供参考。

建议同学们充分利用学习资源，如网上博客等，学习一些问题的求解方法。

一、引言

这部分主要是涉及一些算法和计算机理论的定义和概念。

1.2

既然我们要研究算法设计与分析，那么我们就需要先确定我们的研究对象是什么，也就是什么是算法？算法是满足若干条件的计算。算法的目的是求解问题，那么什么是问题？这样就引入了问题的定义，问题定义了输入输出的关系。最后我们需要一个实际例子来了解算法长什么样子，这就是算法的实例。

1.3

对算法的分析包括，算法的正确性分析，我们想要知道它能不能求解出来我们想要的东西，这里又涉及了算法是正确的，不正确算法和近似算法。循环不变量方法用来证明主要结构是循环结构的算法的正确性。

算法的复杂性分析，我们不只想要知道它能不能计算出来，我们还想要知道这个算法的复杂度是什么样的。这个分析的目的是预测算法对不同输入所需的资源量，测度又包含时间，空间，I/O等。

1.4

想要设计算法，也需要了解算法怎么设计，以及设计完怎么分析，这就是算法的设计方法和算法的分析方法。对于不同的设计方法以及设计的算法，也有着不同的分析方法。

二、数学基础

第二章主要涉及计算复杂性函数的阶和递归方程

2.1

在考量计算时的复杂度，如果我们不是那么关心常数的话，显然我们认为 $2N$ $3N$ 都其实差别不是那么大，但是如果缺少这样的工具来一起度量这两个量的话，就会造成我们需要分别都记录下来这几个，并且我们估计的时候也会更加费力。那么就自然想要用这样一个工具可以描述 $2N$ 和 $3N$ 之间核心的共同点，这就是引入了阶。容易发现 $2N$ 和 $3N$ 就是同阶的，这一点和我们的直觉是相互吻合的。

理解了这一点之后，我们只需要记住这些同阶，低阶，高阶，严格低阶，严格高阶的定义，我们就掌握了一个最基础的能够度量算法复杂性的工具了。比如 n^3 和 n^2 在阶上的关系。这部分的题目，简单的题目可以感觉出来结论是否正确，然后按照定义去证明即可。复杂一些的可能需要一些证明技巧。

对于不容易直观看出来的阶，或者阶之间的比较的结果，需要利用函数阶的性质，如传递性，自反性，对称性，反对称性等。而且并非所有函数都是可比的。

2.2

这部分是一些标准符号和通用函数，比如 Floors, ceiling。

一些算法的复杂度的估计往往会包含和式的估计和界限。这里讲了线性和相关的结论。

我们的算法也会有涉及递归方程的时候，写出复杂性方程，其实是把算法的整个过程以计量的方式体现出来，以mergesort为例，当 $n > 1$ 的时候，考虑这个 n 的时候，需要一次最坏情况 $\theta(n)$ 的操作，然后分解为两个子问题，复杂度都是 $T(n/2)$ ，因为这里关心的是元素数量这些时，算法的复杂度是多少。

求解递归方程有三个主要方法，分别是迭代方法，替换方法，master方法。

迭代方法思想是既然我们这个式子目前还没有给出最后的结果，那么我们不妨去尝试循环地展开这个式子，最后把这个递归方程转化到一个和式计算上，这样就能用求和技术来求解。当然它有自己的局限性了。

替换法，这个想法是主要建立在我们现在的这个方程不好直接处理，我们就要想办法把它进行转换。如果我们有一些经典形式的方程的结果是已知的，那么我们就可以尝试把要求解的式子转换到已知的这些方程上。技巧包括先猜后证等，需要避免一些常见的陷阱。

master定理是一个很有用的工具。master定理也就存在能够处理的情况和不能处理的情况以及master定理怎么使用的也是比较有用的。

三、分治算法

从分治算法到后面都是具有一定难度的了，所以这部分内容需要同学们自己多去摸索，多去理解，这样才能学的更加扎实一些，开箱手册这里也只能帮助同学们大致的了解这几个模块都有什么，以及介绍一个大致的思路。

首先是分治算法的原理，在我们日常生活中，以及学习中，常常当遇到一个问题处理不了的时候，我们就习惯去将这个问题拆解开来，拆成若干个子问题，这样把子问题解决了之后，再去解决原问题。而分治这里，就是先将整个问题划分为多个子问题，再去求解各个子问题，最后把子问题的解合并，再结合这个原始问题的特征，求解原始问题的解。对于一部分分治算法的分析，可以采用建立递归方程然后求解的思路。

课件涉及了3.2最大值和最小值 3.3大整数乘法 3.4矩阵乘法 3.5快速傅里叶变换 3.6线性时间选择算法 3.7最邻近点对3.8凸包算法3.9数据剪除方法

minmax问题，就是考虑原来这 n 个的最大值和最小值是可以由前一半，和后一半的最大值和最小值来求解得到的，复杂性也自然就能求解得到。

n位二进制整数XY，求乘积。

$$XY = (A2^{n/2} + B)(C2^{n/2} + D) = AC2^n + (AD + BC)2^{n/2} + BD \text{ 而}$$

$AD + BC = (A - B)(C - D) + AC + BD$,然后写出递归方程求解或者用master定理都可以了。

同学们根据这两个例子应该已经感受到分治算法的思想了，根据这样的思想可以继续去理解这章的其余的例子。可能会对同学们来说有点不容易理解，慢慢体会，不懂的地方可以多问问老师这样慢慢就会了。

四、动态规划

这部分的课程内容涉及了动态规划原理，以及最长公共子序列，矩阵链乘法，0/1背包问题，最优二叉搜索树，凸多边形的三角剖分。

动态规划是指一种解决问题的方法，在这种方法中，我们预先计算和存储更简单、更相似的子问题，以便构建复杂问题的解决方案。它类似于递归，在递归中，计算基本情况允许我们归纳确定最终值。当新值仅依赖于先前计算的值得时候，这种自下而上的方法非常有效。--

<https://brilliant.org/wiki/problem-solving-dynamic-programming/>

动态规划把原始问题划分成一系列子问题，然后求解每个子问题仅一次，并将其结果保存在一个表中，以后用到时直接存取，不重复计算，节省计算时间。

下面以最长公共子序列为例，展示一下这个方法能够得到的结果。

参考这里<https://blog.csdn.net/SYK000/article/details/120598073>给出一个最长公共子序列的一个定义。

若给定一个序列 x_1, x_2, \dots, x_m ,则另外一个序列 z_1, z_2, \dots, z_k , 是x的子序列是指存在一个严格递增下标序列 i_1, i_2, \dots, i_k ,对于所有的 $j = 1, \dots, k$ 有 $z_j = x_{i_j}$

如果序列Z是X的子序列也是Y的子序列，则称Z是序列X与序列Y的公共子序列。

最长公共子序列问题为： 输入 $X = (x_1, x_2, \dots, x_m), Y = (y_1, y_2, \dots, y_n)$

输出 $Z = X$ 与 Y 的最长公共子序列

首先我们能够想到，如果我们去蛮力枚举X的每个子序列，再检查Z是否是Y的子序列，这样肯定是没有问题的，但是复杂度太高了。

我们接着去分析，既然 X_m 和 Y_n 之间有一个LCS，那么观察 x_m 和 y_n ,可以发现它们在决定LCS的时候是发挥了作用的。

1) $x_m = y_n$: $LCS_{XY} = LCS_{X_{m-1}Y_{n-1}} + 1$ 考虑这个Z是什么情况，如果 $z_k \neq x_m$,则把 x_m 加入到Z，就能得到更长的一个公共序列和现在Z是X与Y的LCS矛盾，于是 $z_k = x_m$,下面考虑 Z_{k-1} 是不是 X_{m-1} 和 Y_{n-1} 的LCS，显然 Z_{k-1} 是公共序列，但是不是LCS呢？如果不是的话，那就存在这样的一个 X_{m-1} 和 Y_{n-1} 的公共子序列 Z' ，它的长度还给比k-1大，不然就能否定上面那个假设了，但是

我们现在又把 x_m 加进去，就会发现出现了一个长度大于k的X与Y的公共序列，这就与Z是LCS相互矛盾了，所以 Z_{k-1} 是 X_{m-1} 和 Y_{n-1} 的LCS。

2) $x_m \neq y_n, z_k \neq x_m: LCS_{XY} = LCS_{X_{m-1}Y}$ 首先可以看到 $z_k \neq x_m$, 那么Z是 X_{m-1} 与Y的公共子序列，那么它是不是LCS呢？假如它不是的话，设 X_{m-1} 与Y有一个公共子序列W，W的长度大于k，而W又是X与Y的公共子序列，这就与Z是LCS相互矛盾了，假设不成立，所以Z是LCS

3) $x_m \neq y_n, z_k \neq x_m: LCS_{XY} = LCS_{XY_{n-1}}$

最后结论如下：

设 $C[i, j] = X_i$ 与 Y_j 的LCS长度 可以得到递归过程如下：

$$C[i, j] = 0 \quad \text{if } i = 0 \text{ 或 } j = 0$$

$$C[i, j] = C[i - 1, j - 1] + 1 \quad \text{if } i, j > 0 \text{ and } x_i = y_j$$

$$C[i, j] = \text{Max}(C[i, j - 1], C[i - 1, j]) \quad \text{if } i, j > 0 \text{ and } x_i \neq y_j$$

通过这个例子希望能够帮助初学者初步理解动态规划的思想，后面的几个例子同学可以自己去研究，或者上课认真听老师讲解。

五、贪心算法

贪心算法这一章包括了5.1贪心算法原理 5.2活动选择问题 5.3哈夫曼编码 5.4最小生成树问题

5.1贪心算法原理，包括了贪心算法的基本概念，贪心选择性，优化子结构，与动态规划方法的比较，贪心算法正确性证明方法。

首先理解贪心算法，求解最优化问题的算法包含了一系列的步骤，在每一步都会有一组选择可以来选，我们希望能够通过做出局部最优的选择，来得到全局最优的选择。贪心算法就是描述了这样的一种希望，除了做题之外，我们会用到它的时候并不一定总是希望它真的能得到真正的全局最优解，而是往往想要通过这种方法先得到一个可行解。至于说贪心算法得到的是不是优化解，那需要我们进行严格的证明。

与动态规划方法相比，这里我理解是建立在求解优化解的前提下进行比较，动态规划可用的条件为1.优化子结构2.子问题重叠性3.子问题空间小 贪心方法可用的条件1.优化子结构2.贪心选择性。

5.2 这里以活动选择问题为例介绍一下贪心算法的思想和应用方法。

定义活动：设 $S = \{1, 2, \dots, n\}$ 是n个活动的集合，各个活动使用同一个资源，资源在同一时间只能为一个活动使用，每个活动i有起始时间 s_i , 终止时间 f_i , $s_i \leq f_i$

相容活动 活动i和活动j是相容的，若 $s_j \geq f_i$ 或 $s_i \geq f_j$ ，这个简单理解，就是活动i和活动j不能有冲突，那么就是要么活动i在活动j开始前结束，要么活动i在活动j结束后开始。

该问题定义为：

输入: $S = \{1, 2, \dots, n\}$, $F = [s_i, f_i]$, $n \geq i \geq 1$

输出: S的最大相容集合

先直接讲答案的话, 根据贪心思想, 每次选择 f_i 最小的活动, 能让我们选择更好的情况, 另外一个选择时间最短的, 也可以得到结果, 这里重点讲解第一种。

首先, 直观理解的话, 在这一次选择的时候, 如果我们选择了一个 f_i 更大的, 不如现在选择最小的这个的, 因为这一步的目前的收益都是一样--指集合里面多了一个元素, 但是选择 f_i 更小的那个会让在后面选择的可能收益更高至少不会低于选择较晚的那个。在这两个结束时间之间, 还可能存在着某些活动可以选择, 而选择了更晚结束的话, 这些活动可能本来有可能会加入进去, 但是这样就加不了了。

课件里面用优化解结构分析以及三个引理来证明,

引理1: 设 $S = \{1, 2, \dots, n\}$ 是n个活动集合, $[s_i, f_i]$ 是活动的起始终止时间, 且 $f_1 \leq f_2 \leq \dots \leq f_n$, S的活动选择问题的某个优化解包括活动1

引理2: 设 $S = \{1, 2, \dots, n\}$ 是n个活动集合, $[s_i, f_i]$ 是活动i的起始终止时间, 且 $f_1 \leq f_2 \leq \dots \leq f_n$, 设A是S的调度问题的一个优化解且包括活动1, 则 $A' = A - \{1\}$ 是 $S' = \{i \in S | s_i \geq f_1\}$ 的调度问题的优化解。这个引理说明了活动选择问题具有优化子结构。

引理3: 设 $S = \{1, 2, \dots, n\}$ 是n个活动集合, $f_0 = 0$, l_i 是 $S_i = \{j \in S | s_j \geq f_{i-1}\}$ 中具有最小结束时间 f_{l_i} 的活动, 设A是S的包含活动1的优化解, 其中 $f_1 \leq \dots \leq f_n$, 则 $A = \bigcup_{i=1}^k l_i$

引理3表示了贪心选择性。

这一章以活动选择问题为例, 介绍了贪心算法的思想和内容, 后面两个例子留给同学上课认真听讲和下课研究了。

六、平摊分析

平摊分析, 包括6.1平摊分析原理 6.2聚集方法 6.3会计方法 6.4势能方法 6.5动态表操作的平摊分析 6.6斐波那契堆性能平摊分析 6.7并查集性能平摊分析

这里简单的介绍平摊分析的基本思想和方法, 后面的详细的方法留在同学们上课或课下研究。

平摊分析是一种算法分析方法, 能够用于计算一系列数据结构操作的最坏情况下的平均时间。

这一章可以参考的网上资料有,

<https://www.wikiwand.com/zh-hans/%25E5%25B9%25B3%25E6%2591%258A%25E5%2588%2586%25E6%259E%2590>

<https://blog.csdn.net/rebortt/article/details/111772607>

<https://blog.csdn.net/demon24/article/details/8474439>

下面根据课件整理得到: 本章的学习目标包括, 目标1: 掌握平摊分析技术 目标2: 积累三种有用的数据结构--动态表, 斐波那契堆, 并查集, 理解现有算法分析结果 目标3: 理解高级程序设计语言中的抽

象实现

算法往往会在某个或一系列的数据结构上执行一系列操作，这里的课程中我们主要是针对在一个数据结构上的分析进行讨论。每个操作的代价也都不一样，那么我们就想要知道1.从平均效果看，每个操作的代价是什么样的2.操作序列的时间复杂度怎么分析。这里就引入了平摊分析，平摊分析将操作序列的总代价分摊到每个操作上，而它并不涉及每个操作被执行的概率，不同于平均复杂度分析。

聚集方法，对应的是每个操作的代价，分析操作序列的每个操作的代价上界，然后求得操作序列的总代价的上界,每个操作平摊 $T(n)/n$ 。

会计方法，对应的是整个操作序列的代价，考虑到一个操作序列中有不同类型的操作，而不同类型的操作的实际操作代价也可能各不相同。于是这里我们为每种操作分配不同的平摊代价，这个分配的可能比实际代价大也可能比实际代价小。

势能方法，对应的是整个操作序列的代价，和会计方法不同的是，在会计方法中，如果操作的平摊代价比实际代价大，我们将余额与数据结构的数据对象相关联，而在势能方法中，我们把余额和整个数据结构相关联，所有的余额之和构成了数据结构的势能。

这里简单介绍了这部分的内容，详细内容请同学们上课认真听讲或课下研究。

七、MAXMIN方法

这一章包括网络流算法，匹配算法，以及一些补充阅读材料。

根据课件PPT：对于这一章的内容，难点为MAX-MIN关系以及其在算法设计与而分析中的应用，MAX-MIN关系是一个很重要，除了这一章的网络流的例子，还会应用在性能分析，算法优化，问题约束等场景中。重点为在基本算法层面，1) 最大流-最小割算法 2)最大匹配-最小覆盖算法3) 基本图论算法的总结与复习，前两个问题是这一章着重讲解的问题，对于初学者难度可能不容易理解，所以需要课上认真听讲课下多研究研究。 算法设计技术层面 1) MAX-min方法，这一部分可以通过结合课程中给的例子来加以理解，和掌握。2)精益求精的算法设计过程 问题特征分析能力层面 1) 准确渐进

关于网络流的这些问题，除了教材之外，网上也有不少博客讲的比较好，个人建议为,在掌握规范的证明方法之前，可以先建立直觉上的认识和理解，这样就知道往什么方向上思考，然后再把这些感受上的结果用比较规范的方式书写下来。

八、树的搜索策略

这一章主要包括了8.1为什么引入搜索策略 8.2基本的搜索策略 8.3优化的搜索策略 8.4人事安排问题 8.5旅行商售货问题 8.6 0-1背包问题 8.7 A*算法

很多问题可以转化为树上的问题，这样可以通过树上的方法来解决，这些方法里面就包含了搜索策略。有很多问题，它不用转化就是树上的问题，而这些问题其中有一部分就能通过搜索策略来解决，这就是为什么要学习树上搜索策略的原因。

基本的搜索策略又包括广度优先搜索和深度优先搜索，直观理解的话，广度优先搜索就是先尽量把这一层的搜索情况先覆盖到，然后再考虑下一层的情况，这就是为什么叫广度优先的原因，而深度优先搜索则希望能尽量把一个情况先尽量往深层搜索，搜索结束后回溯，再去搜索其他的情况。

优化的搜索策略包括爬山策略(Hill Climbing算法)，爬山策略是对深度优先搜索进行改进，用贪心的方法确定了在遇到多个节点可以扩展的情况下的搜索的方向。Best-First Search Strategy 结合了深度优先和广度优先的优点，在目前产生的所有节点里面选择具有最小评价函数的节点扩展，与爬山策略相比，该方法进行了全局优化，而爬山策略仅具有局部优化。Branch-and-Bound Strategy可以有效地求解组合优化问题，能够发现优化解的一个界限，可以缩小解空间，提高求解效率。

人事安排问题，旅行商售货问题，0-1背包问题，这些内容留给同学们上课认真听讲或课下预习或复习了。