算法设计与分析课程期中报告

19335286 郑有为

April 2021

目录

1 Introduction **2**

2 对四种技术的总结 **2**

2.1 设计思想与原理 2

2.1.1 分治算法的基本思想 2

2.1.2 动态规划的基本思想 2

2.1.3 贪心法的基本思想 3

2.1.4 回溯与分支限界的基本思想 3

2.2 适用范围和条件 3

2.2.1 适用范围分类 3

2.2.2 使用条件异同 3

2.3 相互的联系与异同 4

2.3.1 联系和异同 4

2.3.2 自顶向下与自底向上 5

2.4 优缺点比较 5

2.5 算法改进途径 6

3 应用实例与多技术解决 **6**

3.1 案例 I: 6

3.2 案例 II: 6

3.3 案例 III: 6

1 Introduction

在半学期的《算法设计与分析》课程的学习中，我们主要学习了**分治策略**、**动态规划**、**贪心法**、**回溯与分支限界**，共四种算法算法设计技术。本报告对上述四种算法设计思路进行归纳和总结。

报告的第一部分将归纳上述四种算法设计技术的**基本原理**、给出每种算法设计技术的**使用条件限制（适用范围）**和不同算法设计思路之间的**联系**，并分析比较每种算法设计技术的**优缺点**，和**算法改进途径**；在报告的第二部分，将结合几个现实生活中的问题，研究基于不同算法设计技术的解决方案，并给出相应的算法空间和时间复杂度分析，最后选出给定问题的最优算法设计技术。

2 对四种技术的总结

2.1 设计思想与原理

2.1.1 分治算法的基本思想

分治算法的核心思想：分而治之，具体可分为三个步骤。

1. **Divide**，将规模一定的问题规约成若干个子问题，每个子问题**互相独立且与原问题形式相同**，递归分解子问题；

2. **Conquer**，若子问题规模小到可以直接获得结果，则返回结果，否则递归求解各个子问题；

3. **Merge**，递归返回上一级时对子问题的解进行综合，最后一步一步得到原问题的解。

分治算法旨在分析问题本身与规模无关的普遍性质，从而得到分治的**递推方程**，再以此设计递归的算法来解决实际问题。

2.1.2 动态规划的基本思想

动态规划技术将目标问题**划分出多个求解阶段**，划分的结果需满足**最优化原理**，随后从最小的子问题开始逐层向上求解，每一层计算需用到下面层的结果，有效利用前面的获得的结果，避免对子问题的重复计算，提高算法速度。

动态规划技术的本质是**用空间换取时间**。

使用动态规划一般将问题规约成：**在约束条件限制为 R_k 的情况下，仅使用前 k 个元素所取得的最优值为 F_k。**，然后再建立备忘录对结果进行存储、重用和追溯最优解。

2.1.3 贪心法的基本思想

贪心法通常采用迭代的方式一步一步缩小问题的规模，采取“只顾眼前”的局部最优策略来进行每一步取舍，最后得到问题的最优解（有时候只能得到局部最优解），在设计过程中注重贪心策略的正确性证明。

问题的**最优子结构性质**和是否具有**无后效性**是判断一个问题是否可用贪心算法求解的关键特征。

2.1.4 回溯与分支限界的基本思想

回溯法用于解决搜索问题和优化问题，在遍历一个树状的搜索空间以获得所有解或最优解时，回溯法利用某些规则和条件限制进行跳跃式地遍历解空间已得到问题结果。

分支限界是对回溯法的进一步改进，是回溯法的变种，通过建立**约束函数**和**界函数**增加算法搜索时的约束条件，进行剪枝以提高算法的速度。

2.2 适用范围和条件

2.2.1 适用范围分类

在算法适用范围上，我将四种算法设计技术划分为两类，第一类是分治法，第二类是动态规划、贪心法、回溯与分支限界，划分的标准是**是否面向组合优化问题**。

组合优化问题一般有对应的目标函数和约束条件，求解组合优化问题就是求出可行解集中的最优解，最经典的组合优化问题就是背包问题。

在一般的组合优化问题中，经分解得到**子问题往往即不是相同的，也不是互相独立的**，故一般不使用分治法解决此类问题，而是使用**动态规划、贪心法、回溯与分支限界**。

分治法分解得到的**子问题往往是相同的，且互不影响**，分治法在查找、排序等**经典数学问题**上有广泛应用，如二分查找和二分排序，汉诺塔，快速幂等算法，在实际生活中，分治算法可以解决**筛选问题**，如筛选次品等。

2.2.2 使用条件异同

在解决组合优化问题上，动态规划需要遵循无后效性和优化原则（又表现为最优子结构性质）；贪心法强调最优子结构性质；回溯法需满足多米诺性质；下面给出他们的定义：

1. **无后效性**：指如果在某个阶段上过程的状态已知，则从此阶段以后过程的发展变化仅与此阶段的状态有关，而与过程在此阶段以前的阶段所经历过的状态无关。[1]

2. **最优化原理**：这个原理的实质是多阶段决策过程具有这样的性质，即不管过去的过程如何，只从当前的状态和系统的最优化要求出发，作出下一步的最优决策。[2]

3. **最优子结构性质**：指当一个问题的最优解包含其子问题的最优解时，称此问题具有最优子结构性质。[4]

4. **优化原则**：一个最优决策序列的任何一个子序列本身一定是相对于子序列的初始状态和结束状态的最优决策序列，与最优子结构性质是等价的。

5. **多米诺性质**：假设 $P(x_1, x_2, ..., x_i)$ 是向量 $< x_1, x_2, ..., x_i >$ 的某个性质，那么有

$$P(x_1, x_2, ..., x_i, x_{i+1}) \to P(x_1, x_2, ..., x_i)$$

等价于 $\neg P(x_1, x_2, ..., x_i) \to \neg P(x_1, x_2, ..., x_i, x_{i+1})$，其含义是若遍历到某步已经不满足要求，在此基础上往下遍历也不会满足要求，此时应进行回溯。

基于上述定义，我认为无后效性、最优化原理和最优子结构性质在一定程度上是一致的。三者从不同的侧面总结出了我们在解决组合优化问题时划分步骤需要遵循的原理。无后效性和最优化原理从**各个阶段/步骤的时序关系**出发：当前步骤的结果已经总结了过去的步骤的结果，未来的步骤无需考虑过去步骤结果的获得过程。无后效性还有一层意思，就是 “所有各阶段都确定时，整个过程也就确定了”。最优性原理强调问题的最优策略的子策略也是最优的。最优子结构性质则从**分析最优解的结构**出发：原问题的最优解由子问题的最优解能推导而来。

一般情况下，贪心法与动态规划或回溯与分支限界不同，前者需要对贪心策略进行严格的证明，这是由于贪心法的使用条件相对受限，且不直观的原因。

使用贪心法可以实现高速度，低空间损耗的算法，但贪心法有时会陷入局部最优解，需要加以约束才能获得全局最优。在一般情况下，贪心策略适用的前提是：局部最优策略能导致产生全局最优解。但在有些用动态规划或回溯与分支限界解决不够理想的完全 NP 问题，如旅行商问题上，我们可以通过贪心策略得到近似最优解，在实际应用中有广泛的应用。

2.3 相互的联系与异同

2.3.1 联系和异同

1. 动态规划法能解决的组合优化问题，一般用回溯与分支限界也能解决；贪心策略能够解决的问题，使用动态规划或回溯与分支限界也都能解决。

2. 一般情况下，对于能使用贪心策略的问题，使用贪心策略的时间和空间复杂度比使用动态规划或回溯与分支界限要低，并且更容易实现。

3. 相较于暴力求解，四种算法能提高效率的共同原因是减少了不必要的计算。分治法和动态规划提高效率的原因是**减少了重复的计算**，而贪心和回溯与分支限界提高效率的原因是**提前判断并省去了某些非最优解的计算**。

4. 考虑四种算法设计技术的选择与求解次序，分治法和贪心算法是先进行选择，在逐步解决子问题；而动态规划是先解决子问题在对其进行选择；在回溯与分支限界中，选择与子问题的解

决交替进行。

5. 不论是上述哪一种方法，对数据的预处理（如排序）可以提高算法的速度，很多时候，贪心策略实现的基础就是将数据进行某种排列（如按照对应的值从小到大），数据的预处理也有助于分支限界中代价函数的设计。

6. 四种算法设计技术不是完全分离独立的，在具体的算法设计过程中，我们往往可以结合几种算法设计技术来提高算法的性能，如在回溯法中使用分治算法，将原问题先分解成子问题，分别进行树的遍历最后在整合结果。

2.3.2 自顶向下与自底向上

从子问题解决原问题，无非是两种方法，自底向上 (Bottom-Up) 与自顶向下 (Top-Down)，形式上前者对应迭代，利用循环将结果存在数组里，从数组起始位置向后计算；后者对应递归，即利用函数调用自身实现。[5]

分治算法是典型的自顶向下设计过程，将问题规模一步一步缩小，实现上一般采取递归的方式。

动态规划一般采用自底向上，一步一步扩大问题的规模，直到与原问题规模一致。

贪心算法采用自顶向下，以迭代的方法做出相继的贪心选择，每做一次贪心选择，就将所求问题简化为一个规模更小的子问题，通过每一步贪心选择，可得到问题的一个最优解[3] 。

回溯与分支限界在此方面没有明显的区分。

2.4 优缺点比较

分治法：优点是易理解，适用于分布式计算的实现；缺点是对问题的划分有严格的要求，在实现时对系统堆栈要求高。

动态规划：优点是在每一步的计算时能充分利用过去的结果，避免重复计算，一般可以得到多项式时间复杂度；缺点是对空间的要求高，特别是在问题规模比较大时，空间消耗往往难以忍受，与此同时，问题的维数的增加将是的储存空间的维数增加，使得对空间的需求急剧增加。

贪心法：优点是算法的时间复杂度非常低，对空间的损耗也非常少；缺点是贪心策略不一定正确，需要证明或加以条件约束。

回溯与分支限界：优点是对空间要求不高，容易理解；缺点是平均时间复杂度难以计算，一般需要对解进行统计才能估算平均复杂度。最坏时间复杂度几乎是指数级别，难以接受。

在时间复杂度上，一般是贪心法最优，其次是动态规划和分治法，对于回溯与分支限界需要考虑回溯和限界条件的优劣。

在空间复杂度上，只有动态规划有较大的空间需求，空间复杂度较大。

2.5 算法改进途径

分治法的改进途径包括：(1). 寻找子问题的依赖关系，减少子问题的个数，进一步避免重复计算；(2). 数据预处理，以减少递归内部的计算量。

对于动态规划法，有时候得推导出的递推方程可能有多种形式，每种形式对应的复杂度是不同的，需要提前分析以选出最合适的算法。

对于贪心法，在面对不能得到最优解的情况下需要做特殊的处理，一种方法是分析输入，归纳贪心不出最优结果的情况，进而对输入加以限制，或判断这些输入出现的概率；另一种对贪心结果进行统计分析，分析误差的范围，考虑是否接该策略。

对于回溯与分支限界法，分支限界的过程就是对回溯发的改进，除此之外，根据数据预处理优化树的结构也能改进算法。

在教材中，根据影响回溯算法效率的三个因素给出了对应的改进途径，包括：安排树的分支搜索循序策略，利用搜索树的堆成性裁剪子树，分解为子问题。

3 应用实例与多技术解决

3.1 案例 I:

3.2 案例 II:

3.3 案例 III:

```
输入：待求解的问题 P 和问题的规模 n
输出：问题的解 S
Divide_and_Conquer
```

参考文献

[1] 陈克式，陈开周．经济数学辞典：中国经济出版社，1991 年 09 月第 1 版，432 页

[2] 萧浩辉．决策科学辞典：人民出版社，1995

[3] 谢翌，江渝川主编，大学计算机计算思维与应用，重庆大学出版社，2017.01，第 58 页

[4] 补录，计算机算法设计与分析研究，新华出版社，2015.09，第 88 页

[5] 自底向上与自顶向下(递归与动态规划),https://blog.csdn.net/QilanAllen/article/details/100806257