算法概论

复习

薛健

Last Modified: 2020.1.5

主要内容

1 算法分析

基础知识

- 逻辑 (证明)
- 概率统计 (平均时间复杂度)
- ★代数
  - 常用级数：$\sum_{i=1}^{n} i^k \quad \sum_{i=0}^{n} a^i \quad \sum_{i=1}^{n} \frac{1}{i} \quad \sum_{i=1}^{n} \lg i$
  - 用积分求近似和
  - 函数的渐进阶及分类 (Θ、Ω、O)、渐进阶比较方法 (求极限)
    * 对任意常数 $\alpha > 0$ (包括分数)，$\log n \in o(n^\alpha)$
    * 对任意常数 $k > 0$ 和 $c > 1$，$n^k \in o(c^n)$
    * $\sum_{i=1}^{n} i^d \in \Theta(n^{d+1}) \quad \sum_{i=1}^{n} \log i \in \Theta(n \log n)$
    * $\sum_{i=a}^{b} r^i \in \Theta($序列最大项$)$，$r > 0$ 且 $r \neq 1$，a 和 b 可以是 n 的函数
- ★算法正确性证明 (反例证明、反证法、数学归纳法)
- ★基本数据结构 (二叉树)

算法复杂度

- ★时间复杂度
  - 最坏情况时间复杂度：
  $$W(n) = \max\{t(I) | I \in \mathbf{D}_n\}$$
  - 平均情况时间复杂度：
  $$A(n) = \sum_{I \in \mathbf{D}_n} P(I)t(I)$$
- 空间复杂度
- ★常见复杂度的阶
  $$1 \quad \log n \ (\log^c n) \quad n \quad n \log n \quad n^2 \ (n^c) \quad | \quad c^n \quad n!$$
- ★问题的复杂度（下界）

复杂度分析方法

- ★递归（数学归纳法）
  - 递推方程 $T(n) = bT(n/c) + f(n) \Rightarrow$ 主定理
- ★递归树分析
- 对手论证法
  - 常用于分析问题复杂度下界
- 分摊时间分析法
  - 常用于分析由时间开销不定的基本操作构成的算法时间复杂度上届

2 算法设计方法

★ 分治法

- **工作原理**："解决几个小问题通常比解决一个大问题来得简单"
- **设计思想**：将一个直接难以解决的大问题分成较小规模的数个子问题；分别解决（治）这些子问题；将子问题结果合成原问题的结果
  - 由分治法产生的子问题往往是原问题的较小模式，在这种情况下，反复应用分治手段，可以使子问题与原问题类型一致而其规模却不断缩小，最终使子问题缩小到很容易直接求出其解，这自然导致递归过程的产生

- 适用条件：
  1. 问题的规模缩小到一定的程度就可以容易地解决；
  2. 问题可以分解为若干个规模较小的相同问题；
  3. 利用该问题分解出的子问题的解可以合并为该问题的解；
  4. 该问题所分解出的各个子问题是相互独立的，即子问题之间不包含公共的子子问题 (效率)

★ 动态规划

1. 使用自顶向下 (top down) 的方式分析问题，给出一般递归算法 (复杂度可能是指数级的)
2. 分析所得递归算法的子问题图，看是否可以用暂存子问题结果 (memoization) 的方式减少子问题的重复计算，若是，则可用固定的转换步骤将一般递归算法转换为其动态规划版本，此处的关键是：子问题的标识必须尽可能简洁，以便限制子问题图的规模和所需暂存空间的大小
3. 根据子问题图 (深度优先搜索) 分析动态规划版本的递归算法时间复杂度，看是否达到要求
4. 设计用于暂存子问题结果的字典数据结构
5. 分析子问题图的结构，看是否可以简单地得到结点的逆拓扑序列，若是则可以按自底向上 (bottom up) 的方式设计出动态规划算法的非递归版本
6. 确定从字典数据中提取最优解 (决策过程) 的方法

贪心方法

- 与动态规划方法一样，贪心方法一般也用来解决某种优化问题，与动态规划不同的是贪心方法并不穷尽所有子问题的最优解
- 基本思路：
  - 在一个决策序列中，每一步单独的决策其优劣有一个度量标准来衡量
  - 每一步决策总是选择在度量标准下最优的那个分支 (决定)
  - 每一步决策一旦决定便不再更改 (不同于回溯算法)
- 几点注意：
  - 贪心方法给出的解不一定是 (全局) 最优解
  - 给定具体问题设计贪心算法，一般需要证明所设计的算法求得的确实是给定问题的最优解
  - 对同一个问题，贪心算法的效率一般要高于动态规划算法

3 基本算法

排序算法

- ★分治法的具体应用：快速排序、归并排序、堆排序
- 排序算法的时间复杂度分析

算法	最坏情况	平均情况	空间占用	稳定性
插入排序	$n^2/2$	$n^2/4$	$\Theta(1)$	是
选择排序	$n^2/2$	$n^2/2$	$\Theta(1)$	否
快速排序	$n^2/2$	$1.386n\lg n$	$\Theta(\log n)$	否
归并排序	$n\lg n$	$n\lg n$	$\Theta(n)$	是
堆排序	$2n\lg n$	$\Theta(n\log n)$	$\Theta(1)$	否
快速堆排序	$n\lg n$	$\Theta(n\log n)$	$\Theta(1)$	否
希尔排序	$\Theta(n\log^2 n)$	?	$\Theta(1)$	否
基数排序	$\Theta(n)$	$\Theta(n)$	$\Theta(n)$	是

- ★以比较为基本操作的排序算法时间复杂度下界：
  - 最坏情况 $\lceil n\lg n - 1.443n\rceil$，平均情况 $n\lg n - 1.443n$

选择与检索

- 选择算法
  - 查找最大和最小元素
  - 查找第二大元素
  - 中位数问题 (查找第 k 小元素)
- 动态集合与检索算法
  - 动态集合
  - 红黑树 (二叉搜索树)
  - 动态等价关系及合并-查找程序 (设计及分析方法)
  - 优先队列和配对森林

图算法

- 图问题
  - 简单图问题：$W(n)\in O(n)$
  - 中等图问题：$W(n)\in O(n^c), c>1$
  - 困难图问题：复杂度低于多项式级别的算法还未找到
- ★图的搜索与遍历
  - 深度优先搜索：有向图和无向图 (算法框架)

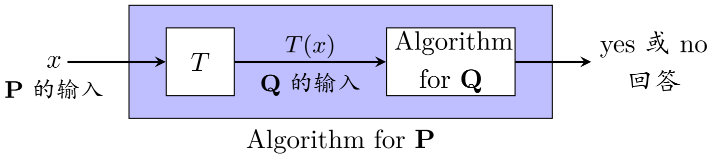
  - 广度优先搜索
  - 有向无环图

- 最小生成树问题
- 单源最短路径问题

4 $\mathcal{NP}$-完全问题

$\mathcal{NP}$-完全问题的证明

- 多项式规约 (转换函数 T 以多项式为界)



- 证明判定问题 $\mathbf{Q} \in \mathcal{NP}$ 是 $\mathcal{NP}$-完全问题：
  1. 找到一个合适的 $\mathcal{NP}$-完全问题 $\mathbf{P}$，则 $\forall \mathbf{R} \in \mathcal{NP}$，$\mathbf{R} \leq_P \mathbf{P}$
  2. 证明 $\mathbf{P} \leq_P \mathbf{Q}$
  3. 则 $\forall \mathbf{R} \in \mathcal{NP}$，$\mathbf{R} \leq_P \mathbf{Q}$，即 $\mathbf{Q}$ 也是 $\mathcal{NP}$-完全问题