分治算法

算法设计与分析小班课

2022 年 3 月

注意: 以下内容主要是给课本作一些注解并补充习题, 可能不能准确地代表课程内容和考核的方向, 也不能替代阅读课本、听课、完成作业题、做往年题等活动; 其中的内容没有经过课程主管老师的审核, 也可能存在错误. 不过出现的所有错误都由作者本人负责.

1 递推方程的解

有关递推方程的求解我们已经知道多种方法——例如在中学阶段就已经清楚的直接迭代、换元等办法. 在这学期或者之前我们应该也已经学到了常系数线性递推方程的通用求解法以及用生成函数求解递推方程的方法. 以下对递推方程的精确求解法作出一个更多的说明.

在观察比较难看出换元的方法时, 也可以机械地算出答案. 考虑形如

$$a_n T_n = b_n T_{n-1} + c_n$$

的递推方程. 设想两边乘了一个合适的 s_n, 称为**求和因子**, 得 $s_n a_n T_n = s_n b_n T_{n-1} + s_n c_n$. 若要换元, 需要 $s_n b_n = s_{n-1} a_{n-1}$, 如果这事达成, 令 $S_n = s_n a_n T_n$, 就有

$$S_n = s_0 a_0 T_0 + \sum_{k=1}^{n} s_k c_k = s_1 b_1 T_0 + \sum_{k=1}^{n} s_k c_k,$$

进一步原递推式的解就是

$$T_n = \frac{1}{s_n a_n} \left(s_1 b_1 T_0 + \sum_{k=1}^{n} s_k c_k \right) \quad (n \geq 1). \tag{1.1}$$

但是 s_n 如何选取呢? 它的取法实际上是机械的, 对 $\frac{s_n}{s_{n-1}} = \frac{a_{n-1}}{b_n}$ 作迭代得到

$$\frac{s_n}{s_1} = \frac{a_{n-1} \cdots a_1}{b_n \cdots b_2},$$

注意 s_1 只由 s_0 约束, 后者不出现在我们的计算中, 故不妨设 $s_1 = 1$.

例 1 分析快速排序算法时差消后得到了关于比较次数 C_n 的递推式:

$$nC_n = (n+1)C_{n-1} + 2n \quad (n \geq 1), \quad C_0 = 0.$$

假设不能立刻看出换元 $\frac{C_n}{n+1}$, 也可以直接施展求和因子法, $a_n = n, b_n = n+1, c_n = 2n$, 得求和因子

$$s_n = \frac{a_{n-1} \cdots a_1}{b_n \cdots b_2} = \frac{(n-1)(n-2)\cdots 1}{(n+1)n \cdots 3} = \frac{2}{n(n+1)}.$$

最后代入式 (1.1) 得解

$$C_n = \frac{1}{\frac{2}{n(n+1)}n} \sum_{k=1}^{n} \frac{2}{k(k+1)} 2k = 2(n+1)(H_{n+1} - 1) \sim n \log n \quad (n \geq 1).$$

◇

2 主定理

主定理是用于渐近地求解一大类递推方程的方法, 它是递归树方法的特例和总结.

这里概括一下证明思路. 对于 $T(n) = aT(n/b) + f(n)$, 我们可将 a 理解为子问题的个数, b 理解为子问题的规模, $f(n)$ 为分治和合并所需要消耗的额外时间, 因此最终有 $n^{\log_b a}$ 个叶子和一些中间处理项. 这些中间项的大小是讨论的来源, 总之是要将 $f(n)$ 与 $n^{\log_b a \pm \varepsilon}$ 进行比较, 然后适当地用等比数列放缩; $f(n)$ 较小时结果取决于叶子数 (情况 1), 较大时取决于中间项 (情况 3, 需正则性条件使 $f(n)$ 是最终单调的), 否则为情况 2.

需要注意的是, 课本中声称可不妨设 $n = b^k$, 这一点 [Cor+09, §4.6.2] 进行了额外的论证. 另给简单说明如下: 算法分析中出现的递推式的解都是单调递增的 (用归纳法证明), 这种情况下总有 $T(b^{k-1}) \leq T(n) \leq T(b^k)$ (对某个 k), 这样通过不妨设的方式去掉底函数和顶函数就明显正确了. 当然, 不论怎么样, 总是可以先猜出答案再用归纳法处理.

在课本的主定理中, 我们发现 $f(n) = \Theta(n^{\log_b a} \log^k n)$ 的情形未被包括 ($\log^k n$ 的因子使得其既不属于情况 1, 也不是情况 3), 但模仿证明过程可以发现这基本属于情况 2:

问题 2 设 $a \geq 1, b > 1, k \in \mathbb{Z}_{\geq 0}$ 且有递推方程 $T(n) = aT(n/b) + f(n)$. 若 $f(n) = \Theta(n^{\log_b a} \log^k n)$, 证明: $T(n) = \Theta(n^{\log_b a} \log^{k+1} n)$.

证明 仿照课本证明的思路, 只需要计算

$$\sum_{j=0}^{\log_b n - 1} a^j f\left(n/b^j\right) \sim \sum_{j=0}^{\log_b n - 1} a^j \left(\frac{n}{b^j}\right)^{\log_b a} \log^k \frac{n}{b^j} = n^{\log_b a} \log_b n \log^k n - \sum_{j=0}^{\log_b n - 1} j^k \log b$$

$$\sim n^{\log_b a} \log^{k+1} n - \log^k n$$

就得到了结论. □

由此, 课本说例 1.19 不能求解实际上并不尽然.

如果你感兴趣的话, 可以尝试计算一下下面这个更广泛的例子.

定理 3 (Akra–Bazzi, 1998) 对每个 $1 \le i \le m$ 给定常数 $a_i \ge 1, b_i > 1$, 假设方程 $\sum_{i=1}^{m} \frac{a_i}{b_i^p} = 1$ 有唯一正实数解 p, 那么递推方程

$$T(n) = f(n) + \sum_{i=1}^{m} a_i T(n/b_i)$$

的渐近行为满足如下性质:

(i) 若 $f(n) = O(n^{p-\varepsilon})$, 则 $T(n) = \Theta(n^p)$;

(ii) 若 $f(n) = \Theta(n^p \log^k n)$, 则 $T(n) = \Theta(n^p \log^{k+1} n)$;

(iii) 若 $f(n) = \Omega(n^{p+\varepsilon})$, 则 $T(n) = \Theta(f(n))$.

问题 4 对于以下每个递推式, 利用主定理求解其渐近的阶. 如果无法使用主定理求解, 则说明理由, 尝试直接求解.

$T(n) = 3T(n/2) + n^2$	$T(n) = 4T(n/2) + n^2$	$T(n) = T(n/2) + 2^n$
$T(n) = 2^n T(n/2) + n^n$	$T(n) = 16T(n/4) + n$	$T(n) = 2T(n/2) + n \log n$
$T(n) = 2T(n/2) + n/\log n$	$T(n) = 2T(n/4) + n^{0.51}$	$T(n) = 0.5T(n/2) + 1/n$
$T(n) = 16T(n/4) + n$	$T(n) = \sqrt{2}T(n/2) + \log n$	$T(n) = 3T(n/2) + n$
$T(n) = 3T(n/3) + \sqrt{n}$	$T(n) = 4T(n/2) + cn$	$T(n) = 3T(n/4) + n \log n$
$T(n) = 3T(n/3) + n/2$	$T(n) = 6T(n/3) + n^2 \log n$	$T(n) = 4T(n/2) + n/\log n$
$T(n) = 64T(n/8) - n^2 \log n$	$T(n) = 7T(n/3) + n^2$	$T(n) = 4T(n/2) + \log n$
$T(n) = T(n/2) + n(2 - \cos n)$		

解 如下所示, 省略了 $\Theta(\cdot)$ 记号, 括号中标出了所对应主定理的情况.

n^2 (3)	$n^2 \log n$ (2)	2^n (3)
无法使用	n^2 (1)	$n \log^2 n$ (2, 课本定理无法使用)
无法使用	$n^{0.51}$ (3)	无法使用
$n!$ (3)	$\sqrt{n}$ (1)	$n^{\log_2 3}$ (1)
n (1)	n^2 (1)	$n \log n$ (3)
$n \log n$ (2)	$n^2 \log n$ (3)	n^2 (1)
无法使用	n^2 (3)	n^2 (1)
无法直接使用		

这里无法使用主定理的递推式基本可以用递归树直接算. 最后一个例子不能直接使用是因为 $n(2-\cos n)$ 不是最终单调的, 不满足正则性条件, 但是注意到 $T(n/2)+n \le T(n) \le T(n/2)+3n$, 所以仍然有 $T(n)=\Theta(n)$. □

3 分治算法的设计

主要思想: 尽可能用当前已经有的信息缩小范围, 减少运算; 可以从输入规模较小的实例中开始考虑. (考试时可能不需要像本节一样分析思路, 给出算法并证明就可以了.)

设计思想	实例	评论
缩小范围	例 2.1 芯片测试, §2.4.2 Select 算法	§2.4.2 这个算法要会使用, 例如习题 2.15, 2.21
分别求解再巧妙合并	§2.3.2 最近点对, 归并排序	主要是合并需要省时间
代数变换, 利用已有信息	例 2.3 Karatsuba 乘法, 例 2.4 Strassen 矩阵乘法, §2.4.3 卷积	目前矩阵乘法的最好上界是 $O(n^{2.3728596})$ (SODA 2021)

首先简单回顾一个分析技术.

问题 5 回答课本第 44 页倒数第二段的问题: 在 Select 算法中, 设每个分组的元素是 a 个, 那么何时得到线性时间的算法, 何时得到高于线性时间的算法?

解 对 m^*, 至少有一半一开始选出的组 (除了那个因除不尽的留下的组以及 m^* 所在的组外) 满足: 每个组都有 $\left\lceil \frac{a}{2} \right\rceil$ 个元素大于等于 m^*, 也就是说

$$\min\{\#S_1, \#S_2\} \geq \left\lceil \frac{a}{2} \right\rceil \left(\left\lceil \frac{1}{2} \frac{n}{a} \right\rceil - 2 \right).$$

于是在最坏情况下可写递推式

$$T(n) = T\left(\left\lceil \frac{n}{a} \right\rceil \right) + T\left(n - \left\lceil \frac{a}{2} \right\rceil \left(\left\lceil \frac{1}{2} \frac{n}{a} \right\rceil - 2 \right) \right) + kn.$$

容易论证 $T(n)$ 是单调递增的, 无妨去掉顶函数和常数因子, 亦即

$$T(n) = T\left(\frac{n}{a} \right) + T\left(\frac{3n}{4} \right) + kn.$$

先来考察必要条件, 假设 $T(n) = cn + o(n)$, 那么 $cn + o(n) = \frac{cn}{a} + \frac{3cn}{4} + kn + o(n)$. 匹配 n 阶项, 必须有 $c > \frac{c}{a} + \frac{3c}{4}$, 即 $a > 4$, 因此 a 至少需要为 5. 根据课本上的论证我们知道 $a = 5$ 时确实得到线性时间的算法, 而且 $a = 1, 2, 3, 4$ 时必得不到线性时间的算法. □

以下展示上面表格中说到的常见的设计思想.

问题 6 假设集合 A, B 均含 n 个实数, 这 $2n$ 个实数各不相同. 现在希望找出 $A \cup B$ 中第 n 小的数, 基本操作为查询: 选择一个 k, 可以询问 A 或者 B 中第 k 小的数. 设计一个只需要 $O(\log n)$ 次查询的解决方案.

解 考虑通过询问来缩小范围.

如果 $n = 1$, 那么只需要比较这两个数即可, 返回小的那个. 对于一般情况, 我们首先查询 A, B 各自第 $\left\lceil \frac{n}{2} \right\rceil$ 小的数, 分别记为 a, b. 鉴于这些数各不相同, 不妨设 $a < b$, 那么因为这里有至少 n 个数, 所以我们知道 A 中前 $\left\lceil \frac{n}{2} \right\rceil$ 个数都不可能是想要的答案, B 中后 $n - \left\lceil \frac{n}{2} \right\rceil = \left\lfloor \frac{n}{2} \right\rfloor$ 个数也不可能是答案. 分别把 A, B 的前 $\left\lfloor \frac{n}{2} \right\rfloor$ 和后 $\left\lfloor \frac{n}{2} \right\rfloor$ 个数 (为递归方便, 不应去掉 A 的前 $\left\lceil \frac{n}{2} \right\rceil$ 个数) 去掉, 这些数去掉的集合分别为 A', B' (一个细节: 由于只能查询, 因此具体实现时要记录 A', B' 分别对应于 A, B 的哪个部分), 算法递归进行即可.

留意到每次递归执行, 我们去掉的是 $2n$ 个数的头尾两段, 共 $2\left\lfloor \frac{n}{2} \right\rfloor$ 个数, 因此 $A \cup B$ 的第 n 小的数和 $A' \cup B'$ 的第 $n - \left\lfloor \frac{n}{2} \right\rfloor$ 是同一个数. 另外, 问题的规模每次严格递减, 所以算法是正确的. 时间复杂度分析: $T(n) = T\left(\left\lceil \frac{n}{2} \right\rceil \right) + 2$, 得出 $T(n) = O(\log n)$. □

问题 7 给定元素各不相同的序列 $a_1, \ldots, a_n$, 称 (i, j) 为 "显著的逆序对", 如果 $a_i > 2a_j$. 设计一个在 $O(n \log n)$ 时间内计算出显著的逆序对的个数的算法.

解 直接仿照归并排序求逆序数的方法即可. 在归并排序的过程中, 我们嵌入一个新的归并过程. 假设 $a_1, \ldots, a_k$ 和 $a_{k+1}, \ldots, a_n$ 分别已经有序, 这两个区间复制一份 (不一定要

复制, 仅是为了方便). 一份用于正常归并排序. 另一份用于计算显著逆序对:
(1) 指标 $i, j \leftarrow k, n$, 跨区间逆序对数 $N \leftarrow 0$.
(2) 若 $b_i \le 2b_j$, 则 $j \leftarrow j-1$, 否则 $N \leftarrow N+j-k, i \leftarrow i-1$.
(3) 当 $j = k+1$ 或者 $i = 1$ 时停止执行, 返回 N.
最后返回 N 加上两段已经排好序的区间的显著逆序对数. 新增归并过程仍然在线性时间内进行, 因此算法的渐近复杂度仍然为 $O(n \log n)$. □

问题 8 设有 n 个元素的集合 S 上存在一个等价关系 $\sim$, 每次可以选择两个元素询问二者是否等价. 设计一个查询次数 $O(n \log n)$ 的算法, 判定如下问题: 是否存在一个元素个数严格超过 $\frac{n}{2}$ 的等价类?

解 如果存在一个元素严格超过 $\frac{n}{2}$ 的等价类, 那么任意地将 S 划分为两个大小分别为 $\left\lfloor \frac{n}{2} \right\rfloor, \left\lceil \frac{n}{2} \right\rceil$ 的集合, 两边至少有一个集合是有元素个数严格超过 $\frac{n}{4}$ 的等价类的; 但反之不一定. 由此可给出:
(1) 如果 $n = 1$ 或者 2, 直接测试是否满足条件, 如果满足则返回集合中任何一个元素 a, 否则返回 `None`.
(2) 对于一般情况, 任意地将 S 划分为两个大小分别为 $\left\lfloor \frac{n}{2} \right\rfloor, \left\lceil \frac{n}{2} \right\rceil$ 的集合, 两边分别递归调用该算法.
(3) 如果任何一个递归调用返回了一个元素, 那么检查它是否确实和集合中一半的元素等价 (最坏情况下需要时间 $O(n)$), 如果是则返回之, 否则返回 `None`. 如果递归调用都没有返回元素, 则也返回 `None`.
时间复杂度分析: $T(n) \le 2T\left(\left\lceil \frac{n}{2} \right\rceil\right) + O(n)$, 得出 $T(n) = O(n \log n)$. (似乎有更快的算法.)□

问题 9 给定有 n 个顶点的完全二叉树, 其中每个顶点都有一个正权, 这些权互不相同. 称一个顶点是局部极小的, 如果其权值为其所有邻居中最小的. 以询问一个顶点的权值为基本操作, 设计算法找出一个局部极小的顶点, 要求算法运行时间为 $O(\log n)$.

解 二叉树天然地具有分治的结构, 由此立刻可以想到如下算法:
(1) 检查根结点是否比两个孩子结点都小, 如果是则返回, 否则进入较小的子结点所在的子树并递归.
(2) 到达叶子或者 (1) 返回时算法终止.
根据算法的构造, 当其终止时, 其返回的结点一定比它的父亲结点 (如果有的话) 小, 而返回内部结点时它还比两个孩子都小, 表明找到了局部极小. 因为这个二叉树只有 $O(\log n)$ 层, 因此算法满足要求. □

注 10 可以再考虑 G 为一个 $n \times n$ 的网格, 要求算法运行时间为 $O(n)$ (相当于在矩阵 $A \in \mathbb{R}^{n \times n}$ 中找局部极小), 这也容易用划分网格的分治算法解决. ♤

问题 11 以下是一个图形学中的问题. 假设平面上给定了 n 条直线 $L_i: y = a_i x + b_i$ $(i = 1, 2, \dots, n)$, 这些直线任何三条都不共点. 我们称一条直线 L_i 是 "可见的", 如果存在某个

x_0, 使得 $a_i x_0 + b_i > a_j x_0 + b_j$ 对所有的 $j \neq i$ 成立. 形象地说, 就是从 y 轴无穷远方向向下看, 至少能看到这条直线中的若干个点. 以四则运算和比较为基本操作, 给出一个在 $O(n \log n)$ 时间内计算出所有可见直线的算法.

注 12 计算几何的很多基本算法都是用巧妙的分治方法实现的. 较典型的除了课本上的最近点对之外, 还有求凸包 (连带得到最远点对), 平面上的最小生成树, Delaunay 三角化 (对点集作三角剖分, 使得任何三个点的外接圆内部没有任何其他点) 等. ♤

解 假如要采用分治算法, 那么我们的主要工作是: 给出两个可见直线的集合 $\mathscr{L}_1, \mathscr{L}_2$, 将它们合并起来, 计算最终可见的集合. 为此肯定要利用一些几何性质, 我们先在三条直线的情形下找些直观.

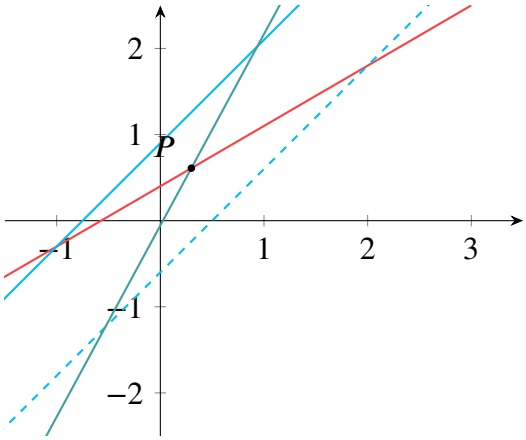


图 1: 问题 11 的基础情形的示例

当只有两条直线时, 显然两者都是可见的. 在三条直线的情况下, 通过考虑 $x \to \pm\infty$ 的情形可知, 斜率最小的和斜率最大的直线都必然可见, 关键是斜率居中的曲线. 参看图 1 中蓝色直线在不同位置下的可见情况: P 为临界点, 当它和斜率最小的直线的交点在同斜率最大的直线的交点左侧时可见, 否则不可见. 由此可以知道, 计算出一些交点的坐标是重要的, 而且我们有如下观察: 设 $L_{i_1}, L_{i_2}, \ldots, L_{i_k}$ 是斜率按递增排列的可见直线族, 令 $a_j = L_{i_j} \cap L_{i_{j+1}}$ $(j = 1, 2, \ldots, k-1)$, 则 $a_1, \ldots, a_{k-1}$ 的横坐标单调递增.

先把所有的直线按斜率排序, 然后考虑分治算法. 用 $\mathscr{L}_1 = \{L_{i_1}, \ldots, L_{i_k}\}, \mathscr{L}_2 = \{L_{j_1}, \ldots, L_{j_\ell}\}$ 分别表示 $\left\{L_1, \ldots, L_{\lceil \frac{n}{2} \rceil}\right\}$ 和 $\left\{L_{\lceil \frac{n}{2} \rceil}, \ldots, L_n\right\}$ 中的可见直线, 且分别有顺次的交点 $a_1, \ldots, a_{k-1}; b_1, \ldots, b_{\ell-1}$. 我们用归并排序的方法合并交点横坐标的递增序列 $c_1, \ldots, c_{k+\ell-2}$.

现在考虑:

$z = \min\{i \in \{1, 2, \ldots, k + \ell - 2\}$:

在点 c_i 的横坐标处, $\mathscr{L}_1$ 中纵坐标最高的直线在 $\mathscr{L}_2$ 中纵坐标最高的直线的下面$\}$.

这一下标可以在亚线性时间内计算出来, 因为 $\mathscr{L}_1, \mathscr{L}_2$ 中的交点都是排好序的, 每一个 c_i 都是某个集合中的交点, 而决定另外的集合中在此点纵坐标最高的直线可以二分搜索得到.

假设 c_z 处 $\mathscr{L}_1$ 中纵坐标最高的直线为 L_{i_s}, $\mathscr{L}_2$ 中则为 L_{j_t} (都取斜率小的那个), 二者的交点为 c^*, 那么我们断言: $\mathscr{L}_1 \cup \mathscr{L}_2$ 中的可见直线为 $L_{i_1}, \dots, L_{i_s}, L_{j_t}, \dots, L_{j_\ell}$, 它们的交点为 $a_1, \dots, a_{i_{s-1}}, c^*, b_{j_t}, \dots, b_{j_\ell}$. 这样就完成了递归求解.

断言的证明 参看图 2. 不妨设 c^* 为 $L_{i_s}, L_{i_{s+1}}$ 的交点, 那么 $L_{i_{s+1}}$ 及斜率更大的直线的可见部分只能是在 c^* 的右边, 但是因为 L_{j_t} 的存在, 它们的可见部分全部被遮住, 所以必然不可见. 根据 z 的最小性知道 L_{i_s} 仍然可见. 类似地, L_{j_t} 及斜率更小的直线的左侧可见部分都会被 L_{i_s} 遮住. 最后, 再次依照 z 的最小性, 我们知道 c^* 恰好在 $a_{i_{s-1}}, b_{j_t}$ 之间.

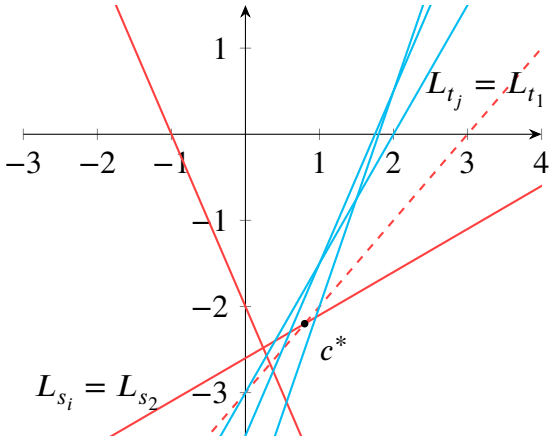


图 2: 合并解的过程的示例: 红线为 $\mathscr{L}_1$ 中直线, 蓝线为 $\mathscr{L}_2$ 中直线, 红虚线为合并后去掉的不可见直线

时间复杂度分析: $T(n) = 2T(n/2) + O(n)$, 得出 $T(n) = O(n \log n)$. □

课程中还会出现一些运用已经设计好的算法来处理问题的题目. 比较重要的是 SELECT 算法和卷积.

问题 13 给定互不相同的实数 $x_1, \dots, x_n$ 并分别赋非负权 $w_1, \dots, w_n$, 其中 $\sum_{i=1}^n w_i = 1$. 称 x_k 为带权的中位数, 如果

$$\sum_{x_i < x_k} w_i < \frac{1}{2} \quad \text{而且} \quad \sum_{x_i > x_k} w_i \le \frac{1}{2}.$$

(i) 用排序算法给出一个 $O(n \log n)$ 的找带权中位数的算法.

(ii) 给出一个线性时间的找带权中位数的算法.

(iii) 利用 (ii) 中算法解决如下**邮局问题**: 二维平面上有 n 个点 $\boldsymbol{p}_1, \dots, \boldsymbol{p}_n$ 且有非负权

$w_1, \dots, w_n$ 满足 $\sum_{i=1}^n w_i = 1$. 求以下优化问题的解:

$$\min_{\boldsymbol{p}} \sum_{i=1}^n w_i \|\boldsymbol{p} - \boldsymbol{p}_i\|_1.$$

这里 $\|(x_1, y_1) - (x_2, y_2)\|_1 \stackrel{\text{def}}{=} |x_1 - x_2| + |y_1 - y_2|$ 为 Manhattan 距离.

解 (i) 显然.

(ii) 算法在递归过程维护一个参数 k, 表示要选出的 "中位数" 将原数组划分的比例, 一开始 $k = \frac{1}{2}$. 先使用 SELECT 算法给出一组数的中位数 m 及其权 w, 计算 $W = \sum_{x_i<m} w_i$. 如果 $W < k, W + w \geq k$, 那么就返回 k. 否则:

- ◇ 如果 $W < k$, 说明想要的结果在 m 右边, 递归调用并令 $k \leftarrow k - W - w$;
- ◇ 否则, 想要的结果在 m 左边, 递归调用.

时间复杂度分析: $T(n) = T(n/2) + O(n)$, 得出 $T(n) = O(n)$.

(iii) 留意到问题可独立地划分为 x, y 两个方向. 我们指出带权中位数

$$x^* \in \operatorname*{argmin}_x \sum_{i=1}^n w_i |x - x_i|.$$

实际上, 优化函数可写为

$$\sum_{x_i>x} w_i(x_i - x) + \sum_{x_i<x} w_i(x - x_i),$$

这函数是连续的且只有有限个不可导点. 在导数存在的点, 导数为

$$\sum_{x_i<x} w_i - \sum_{x_i>x} w_i,$$

它是单调递增的, 简单论证可知对于带权中位数 x^*, 当 $x < x^*$ 时导数为负, $x > x^*$ 时为正, 因此它就是最优解. 计算带权中位数即可. □

问题 14 在一维坐标轴 $1, 2, \dots, n$ (单位为 m) 的位置上各摆放了一个点电荷, 其电量为 q_i (单位为 C, 数值可以是正数或者负数). 物理学知识告诉我们, 点 j 所受到的 Coulomb 力为

$$F_j = k\left(\sum_{i<j} \frac{q_i q_j}{|j-i|^2} - \sum_{i>j} \frac{q_i q_j}{|j-i|^2}\right) \quad \text{(N)}.$$

其中 k 为常量. 以四则运算为基本操作, 设计一个只需要 $O(n \log n)$ 时间就计算出全部 F_j $(1 \leq j \leq n)$ 的算法.

解 我们注意到

$$\begin{aligned} F_j/(kq_j) &= \sum_{i<j} \frac{q_i}{|j-i|^2} - \sum_{i>j} \frac{q_i}{|j-i|^2} \\ &= q_1 \frac{1}{(j-1)^2} + q_2 \frac{1}{(j-2)^2} + \cdots + q_{j-1} \frac{1}{1^2} + q_j \cdot 0 + q_{j+1} \frac{-1}{1^2} + \cdots + q_n \frac{-1}{(j-n)^2} \\ &= \left[(q_1, q_2, \ldots, q_n) * \left(\frac{1}{(n-1)^2}, \frac{1}{(n-2)^2}, \ldots, 1, 0, -1, \frac{-1}{(n-2)^2}, \ldots, \frac{-1}{(n-1)^2} \right) \right]_j . \end{aligned}$$

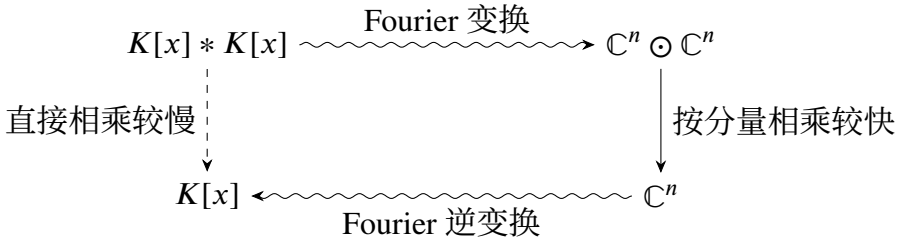
所以使用课上讲过的快速卷积方法 (Fourier 变换) 就可以完成. □

注 15 可以进一步考虑如下问题: 在 $m \times m$ 的网格上放置 n (设 $\log m \ll n$) 个点电荷, 电荷量为 q_i ($i = 1, 2, \ldots, n$), 求 $m^2 - n$ 这几个没有放置电荷的位置的势能 (取无穷远为势能零点). (可以在 $O(m^2 \log m)$ 时间内完成.) ♤

4 附: 离散 Fourier 变换

以下我们用比较干净的语言说明课本 §2.4.3 的算法.

用插值的方法确定多项式, 这相当于把多项式转换到了复数域上, 此时进行多项式的相乘是按分量乘法, 就比直接蛮力卷积要快. 进一步, 如果这个变换和逆变换的方法可以快速实现, 那么就确实得到了更快的算法.



设 $f(x) = a_0 + a_1 x + a_2 x^2 + \cdots x_{n-1} x^{n-1}$, 所作的较好的变换就是离散 Fourier 变换

$$\boldsymbol{v} = \mathscr{F}(f) = \underbrace{\begin{bmatrix} 1 & 1 & 1 & \cdots & 1 \\ 1 & \zeta_n & \zeta_n^2 & \cdots & \zeta_n^{n-1} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & \zeta_n^{n-1} & \zeta_n^{2(n-1)} & \cdots & \zeta_n^{(n-1)^2} \end{bmatrix}}_{F_n} \begin{bmatrix} a_0 \\ a_1 \\ \vdots \\ a_{n-1} \end{bmatrix},$$

其中 ζ_n 表 n 次单位根 $\mathrm{e}^{\frac{2\pi\sqrt{-1}}{n}}$. 易验证其逆变换为

$$\mathscr{F}^{-1}(\boldsymbol{v}) = \frac{1}{n}\begin{bmatrix} 1 & 1 & 1 & \cdots & 1 \\ 1 & \zeta_n^{-1} & \zeta_n^{-2} & \cdots & \zeta_n^{-(n-1)} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & \zeta_n^{-(n-1)} & \zeta_n^{-2(n-1)} & \cdots & \zeta_n^{-(n-1)^2} \end{bmatrix} \boldsymbol{v} = \frac{1}{n}\overline{\mathscr{F}(\overline{\boldsymbol{v}})}.$$

问题归结为如何快速计算矩阵 F_n 乘以一个向量. 为方便说明我们考虑 F_{2n}. 利用 $\zeta_{2n}^2 = \zeta_n$, 令 $D_n = \mathrm{diag}\{1, \zeta_{2n}, \ldots, \zeta_{2n}^{n-1}\}$ 并写出列向量形式 $F_{2n} = \begin{bmatrix} \boldsymbol{f}_1 & \cdots & \boldsymbol{f}_{2n} \end{bmatrix}$. 适当重排待作变换的向量, 我们有

$$\begin{aligned} \mathscr{F}(f) &= \begin{bmatrix} \boldsymbol{f}_1 & \boldsymbol{f}_3 & \cdots & \boldsymbol{f}_{2n-1} & \boldsymbol{f}_2 & \cdots & \boldsymbol{f}_{2n} \end{bmatrix} \underbrace{\begin{bmatrix} a_0 & a_2 & \cdots & a_{2n-2} & a_1 & \cdots & a_{2n-1} \end{bmatrix}^{\mathsf{T}}}_{\tilde{\boldsymbol{v}}} \\ &= \begin{bmatrix} F_n & D_n F_n \\ F_n & -D_n F_n \end{bmatrix} \tilde{\boldsymbol{v}} = \begin{bmatrix} I_n & D_n \\ I_n & -D_n \end{bmatrix} \left(\begin{bmatrix} F_n & \boldsymbol{0} \\ \boldsymbol{0} & F_n \end{bmatrix} \tilde{\boldsymbol{v}} \right). \end{aligned}$$

此时便可以展开分治算法. 递归算出括号中的项之后, 对角矩阵对向量的乘法可以在线性时间内完成, 即是 $T(2n) = 2T(n) + O(n)$, 这推出 $T(n) = O(n \log n)$.

参考文献

有关 SELECT 的算法是一个重点, 即是次序统计量的选择方法. 上面选择的相关问题比较少, 欲了解更多信息推荐阅读 [Cor+09, 第 9 章] 并做相关习题. 选入的习题基本上来源于 [KT06, 第 5 章]. 注 15 中的问题可在这里找到. 问题 9 是 [LTT89] 的最简单版本. Fourier 变换的更具体内容已经在数学分析 (III) 或者将要在信号与系统中接触到.

[Cor+09] T. H. Cormen et al. *Introduction to Algorithms*. MIT press, 2009. ISBN: 978-0-26-203384-8.

[KT06] J. Kleinberg and É. Tardos. *Algorithm Design*. Pearson Education, 2006. ISBN: 978-0-32-129535-4.

[LTT89] D. C. Llewellyn, C. Tovey, and M. Trick. “Local optimization on graphs”. 刊于: *Discrete Applied Mathematics* 23.2 (1989), pp. 157–178.

编写: WC
E-mail: wchang@pku.edu.cn