算法分析与设计: 第四次作业

PB20061372　朱云沁　Apr. 18, 2023

8.1-1 在一棵比较排序算法的决策树中, 一个叶节点可能的最小深度是多少?

解: 记待排序的元素为 $a_1, a_2, \dots, a_n$. 决策树叶节点对应排列 $\langle \pi(1), \pi(2), \dots, \pi(n) \rangle$. 此时, 排序算法确定出 $n-1$ 个不等式

$$a_{\pi(1)} \le a_{\pi(2)},\ a_{\pi(2)} \le a_{\pi(3)},\ \cdots,\ a_{\pi(n-1)} \le a_{\pi(n)},$$

等价于 $n-1$ 次形如 $a_i \le a_j$ 的比较. 另一方面, 为了遍历 n 个元素, 至少需要 $n-1$ 次比较. 故叶节点的最小深度为 $n-1$.

8.2-4 设计一个算法, 它能够对于任何给定的介于 0 到 k 之间的 n 个整数先进行预处理, 然后在 $O(1)$ 时间内回答输入的 n 个整数中有多少个落在区间 $[a, b]$ 内. 你设计的算法的预处理时间为 $\Theta(n+k)$.

解: 伪代码如下:

Algorithm 1: 8.2-4

Input: an array $A[1..n]$ of n integers in $[0, k]$, and two real numbers a, b in $[0, k]$
Output: the number of integers in A that are in the interval $[a, b]$

1. Let $C[0..k]$ be an array of size $k+1$.
2. **for** $i = 0$ **to** k:
3. 　$C[i] \leftarrow 0$
4. **for** $i = 1$ **to** n:
5. 　$C[A[i]] \leftarrow C[A[i]] + 1$
6. **for** $i = 1$ **to** k:
7. 　$C[i] \leftarrow C[i] + C[i-1]$
8. **return** $C[\lfloor b \rfloor] - C[\lceil a \rceil - 1]$

第 $1 \sim 7$ 行进行预处理, 三个 **for** 循环的时间复杂度分别为 $\Theta(k)$, $\Theta(n)$, $\Theta(k)$, 故预处理时间为 $\Theta(n+k)$. 第 8 行进行查询, 时间复杂度为 $\Theta(1)$.

8.3-4 说明如何在 $O(n)$ 时间内, 对 0 到 $n^3 - 1$ 区间内的 n 个整数进行排序.

解: 伪代码如下:

Algorithm 2: 8.3-4

Input: an array $A[1..n]$ of n integers in $[0, n^3 - 1]$.
Output: a sorted array $B[1..n]$ in ascending order.

Let $C[0..n]$ be an array of size $n+1$, and $B[1..n]$ be an array of size n.
for $d = 1$ **to** 3:
　for $i = 0$ **to** n:
　　$C[i] \leftarrow 0$
　for $i = 1$ **to** n:
　　$C\left[\left\lfloor A[i]/n^{d-1} \right\rfloor \bmod n\right] \leftarrow C\left[\left\lfloor A[i]/n^{d-1} \right\rfloor \bmod n\right] + 1$
　for $i = 1$ **to** n:
　　$C[i] \leftarrow C[i] + C[i-1]$
　for $i = n$ **to** 1:
　　$B\left[C\left[\left\lfloor A[i]/n^{d-1} \right\rfloor \bmod n\right]\right] \leftarrow A[i]$
　　$C\left[\left\lfloor A[i]/n^{d-1} \right\rfloor \bmod n\right] \leftarrow C\left[\left\lfloor A[i]/n^{d-1} \right\rfloor \bmod n\right] - 1$
　$A[1..n] \leftarrow B[1..n]$

给定 n 个 3 位数, 每一位的取值范围为 0 到 n, 进行 Radix-Sort. 其中, 对每一位均使用 Counting-Sort 进行稳定排序, 耗时 $\Theta(n)$, 共有 3 轮, 故总时间复杂度为 $\Theta(n)$.

8.4-4 在单位圆内给定 n 个点, $p_i = (x_i, y_i)$ 对所有 $i = 1, 2, \dots, n$, 有$0 < x_i^2 + y_i^2 \le 1$. 假设所有的点服从均匀分布, 即在单位圆的任一区域内找到给定点的概率与该区域的面积成正比. 请设计一个在平均情况下有 $\Theta(n)$ 时间代价的算法, 它能够按照点到原点之间的距离 $d_i = \sqrt{x_i^2 + y_i^2}$ 对这 n 个点进行排序. (提示:在 Bucket-Sort 中, 设计适当的桶大小, 用以反映各个点在单位圆中的均匀分布情况.)

解: 伪代码如下:

Algorithm 3: 8.4-4

Input: an array $A[1..n]$ of n points $p_i = (x_i, y_i)$ in the unit circle, $i = 1, 2, \dots, n$.
Output: the array $A[1..n]$ sorted in ascending order by the distance to the origin.

Let $B[0..n-1]$ be an array of n linked lists that store points.
for $i = 0$ **to** $n-1$:
　$B[i] \leftarrow \varnothing$
for $i = 1$ **to** n:
　Insert $A[i]$ into $B\left[\left\lfloor n(x_i^2 + y_i^2) \right\rfloor\right]$.
for $i = 0$ **to** $n-1$:
　Sort $B[i]$ in ascending order by the distance to the origin, using Insertion-Sort.
$A[1..n] \leftarrow B[0] \oplus B[1] \oplus \cdots \oplus B[n-1]$

桶 $B[i]$ 中的点满足 $\frac{i}{n} \le x_i^2 + y_i^2 < \frac{i+1}{n}$. 记

$$X_{ij} = \mathrm{I}\left\{ \frac{i}{n} \le x_j^2 + y_j^2 < \frac{i+1}{n} \right\} \sim \text{Bernoulli}\left(\frac{1}{n}\right)$$

由于 $X_{i1}, X_{i2}, \dots, X_{in}$ 相互独立, 故 $B[i]$ 的大小 n_i 满足

$$n_i = \sum_{j=1}^{n} X_{ij} \sim \text{Binomial}\left(n, \frac{1}{n}\right),$$
$$\mathrm{E}\left[n_i^2\right] = \mathrm{E}^2\left[n_i\right] + \mathrm{Var}\left[n_i\right] = 2 - \frac{1}{n}.$$

因此, 该 Bucket-Sort 在平均情况下的运行时间为

$$\Theta(n) + nO\left(2 - \frac{1}{n}\right) = \Theta(n).$$

9.3-4 对一个包含 n 个元素的集合, 假设一个算法只使用比较来确定第 i 小的元素, 证明: 无需额外的比较操作, 它也能找到前 $i-1$ 个更小的元素和后 $n-i$ 个更大的元素.

解: 算法 Select 中, 最后一次 Partition 操作结束时, 数组中前 $i-1$ 个元素均小于 $A[i]$, 后 $n-i$ 个元素均大于 $A[i]$, 满足题意.

对于一般的算法, 记输入的元素为 $a_1, a_2, \dots, a_n$. 算法确定出第 i 小的元素, 当且仅当存在若干个形如 $a_i \le a_j$ 的比较, 使得对某一排列 $\langle \pi(1), \pi(2), \dots, \pi(n) \rangle$, 有

$$a_{\pi(i)} \ge a_{\pi(1)},\ a_{\pi(i)} \ge a_{\pi(2)},\ \cdots,\ a_{\pi(i)} \ge a_{\pi(i-1)},$$
$$a_{\pi(i)} \le a_{\pi(i+1)},\ a_{\pi(i)} \le a_{\pi(i+2)},\ \cdots,\ a_{\pi(i)} \le a_{\pi(n)}.$$

对任意 $\pi(j) < \pi(i)$, 分情况讨论: (1) 若存在比较 $a_{\pi(i)} \ge a_{\pi(j)}$, 则元素 $a_{\pi(j)}$ 能被找到; (2) 若不存在 $a_{\pi(i)}$ 和 $a_{\pi(j)}$ 间的比较, 则必有形如 $a_{\pi(i)} \ge a_{\pi(k_1)} \ge \cdots \ge a_{\pi(k_m)} \ge a_{\pi(j)}$ 的若干比较, 使得 $a_{\pi(i)} \ge a_{\pi(j)}$ 成立, 故 $a_{\pi(j)}$ 能够通过搜索算法找到. 对任意 $\pi(j) > \pi(i)$, 有类似的结论. 因此, 算法能够找到前 $i-1$ 个更小的元素和后 $n-i$ 个更大的元素.

9.3-7 设计一个 $O(n)$ 时间的算法, 对于一个给定的包含 n 个互异元素的集合 S 和一个正整数 $k \le n$, 该算法能够确定 S 中最接近中位数的 k 个元素. 请给出伪代码.

解: 伪代码如下:

Algorithm 4: 9.3-7

Input: a set S of n distinct elements, and a positive integer $k \le n$.
Output: the k elements in S that are closest to the median.

Let $A[1..n]$ be an array that stores the elements in S.
$\mathit{median} \leftarrow \textsc{Select}\left(A, 1, n, \left\lfloor \frac{n+1}{2} \right\rfloor\right)$
Initialize two arrays $\mathit{distance}[1..n]$ and $\mathit{sign}[1..n]$.
for $i = 1$ **to** n:
　$\mathit{distance}[i] \leftarrow |A[i] - \mathit{median}|$
　$\mathit{sign}[i] \leftarrow \operatorname{sgn}(A[i] - \mathit{median})$
$\textsc{Select}(\mathit{distance}, 1, n, k)$
Initialize an array $B[1..k]$.
for $i = 1$ **to** k:
　$B[i] \leftarrow \mathit{median} + \mathit{sign}[i] \cdot \mathit{distance}[i]$
return $B[1..k]$

Select 算法的最坏情况运行时间为 $O(n)$, 又由于 $k \le n$, 故算法的运行时间为 $O(n)$.