第 7 次书面作业答案

第八章 内排序

1. 首先，按照字符串的长度进行保序的桶排序，使得字符串由短到长的排列。该过程时间复杂度为$O(m + l_{max})$，其中$l_{max} = \max\{l_i\}$。注意到：

$$m + l_{\max} \le m + \sum_{i=1}^{m} l_i - (m-1) = O(\sum_{i=1}^{m} l_i)$$

然后,进行l_{max}次桶排序。第i次排序只对长度大于等于于$l_{max} - i + 1$的字符串的第$l_{max} - i + 1$位进行桶排序。并用计数变量 k 去记录满足条件的字符串的下标的范围[k,…,m]，使得$l_k \ge l_{max} - i + 1$并且 $l_{k-1} < l_{max} - i + 1$

假设长大于等于l的字符串数目为$n_{\ge l}$。则排序的时间代价为

$$O\left(\sum_{l=1}^{l_{max}} (n_{\ge l} + 26)\right) = O(\sum_{i=1}^{m} l_i)$$

因而总的排序时间复杂度为$O(\sum_{i=1}^{m} l_i)$

2. 与快排类似，对数组进行递归划分，找出第 k 小的元素。然后线性扫描数组即可。查找第 k 小元素的算法伪代码如下：

```
Randomized-Select (A, p, r, k)
    If p = r then
        Return A[p]
    q = Randomized-Partition (A, p, r)
    m = q – p + 1
    If k = m then
        Return A[q]
    Else If k < m then
        Return Randomized-Select (A, p, q - 1, k)
    Else
        Return Randomized-Select (A, q + 1, r, k - m)
```

```
Randomized-Partition (A, p, r)
    i = Random (p, r)
    Swap (A[r], A[i])
    Return Partition (A, p, r)
```

Partition(A, p, r)实现可以参考教材。

3. 1)

使用归纳法：k=2 时显然成立。假设对$k \le 2n/3$都能正确排序。由假设，第一个调用 sort(A, i, j-k)正确地 A[1,…,2n/3]进行了排序，使得 A[1,…,n/3]

小于 A[(n+1)/3,…,2n/3];同理，第二个调用对 A[(n+1)/3,…,n]进行了排序，使得最大的 n/3 个元素拍到了正确的位置。最后一个调用，使得剩下的 2n/3 个元素正确的进行了排序。

2)

解如下递归方程：

$$\mathrm{T(n)} = 3\mathrm{T}\left(\frac{2\mathrm{n}}{3}\right) + \mathrm{O}(1)$$

可得$\mathrm{T(n)} = \mathrm{O}(n^{log3/log1.5})$

4.

1） 因为函数 H 是单调递增的，保证了序号较大的桶的元素一定大于序号较小的桶的元素。只要分别对各个桶进行排序，即可保证最终输出的正确性。

2） 最好情况:每个桶已经排好序，则显然复杂度为 O(n)

最坏情况:所有元素都映射到一个桶中，则插入排序的代价为 O(n^2)

平均情况:每个桶的大小为$\mathrm{O}(n/m)$,因而每个桶的插入排序复杂度为$\mathrm{O}(n^2/m^2)$。而映射和查找桶的时间为 O(n),输出所有元素的时间为 O(n)

因而平均时间复杂度为：

$$\mathrm{O}(n + n^2/m)$$

严格的证明思路可以参考算法导论(p102)。它们给出了 m=n 的特殊情形的证明。

5.

原下标	0	1	2	3	4	5	6	7
数组 A	20	13	11	11’	19	89	6	4
索引 1 下标	6	4	2	3	5	7	1	0
索引 2 下标	7	6	2	3	1	4	0	5
结果	4	6	11	11’	13	19	20	89

6. 我们优化的是归并排序的插入排序部分，不开辟新的内存从而实现排序（左右各自有序的状态下）。算法流程：

1）指针 i 指向左子段的开头，指针 j 指向右子段的开头。

2）i 指针不断向后移动，直到找到第一个比 j 指向的元素大的元素或者直到和 j 相遇。

3）index 指针先代替 j 指向右端的第一个元素。

4）j 指针不断向后移动，直到找到第一个比 i 指向元素大的元素或者直到遇到数组的末尾。

5）将[i, index)段和[index, j)段进行内存反转（手摇算法），之后将 i 移动 j-index+1 空位。以 i 开始的子序列和以 j 开始的子序列又是最初的问题模型，所以继续上述操作（一旦 i，j 相遇或者 j 到达末尾，在该操作 5 结束后直接结束算法）

时间复杂度：（手摇算法时间复杂度是$\boldsymbol{O(n)}$）

最好的情况：左子段和右子段直接全部交换，此时归并排序总复杂度还是$\boldsymbol{O(nlogn)}$

最坏的情况：一段段的缓慢前进的情况，对于长度为 l 的两个有序序列，merge 时间复杂度是$\boldsymbol{O(l^2)}$，归并的长度从 1 到 n（每次乘 2），则长度为 l 的 merge 共进行$\boldsymbol{n/(2l)}$次。则$\boldsymbol{f(n)=\sum_{k=0}^{\log_2 n}\frac{n}{2l}l^2}$，其中$\boldsymbol{(l=2^k)}$，推得$\boldsymbol{f(n)=O(n^2)}$。