# 算法设计与分析 (2017 年秋季学期)

# 第二次作业参考答案

## 1 最小 $k$ 个数问题 (共 25 分)

给定一个包含 $n$ 个数的集合，欲采用一个基于比较的排序算法找出集合中最小的 $k$ 个数。下面给出了一些解决这个问题的算法，请分析每个算法的时间复杂度 (请使用关于 $n$ 和 $k$ 的函数表示)。可以假定集合中的所有数都是互不相同的。

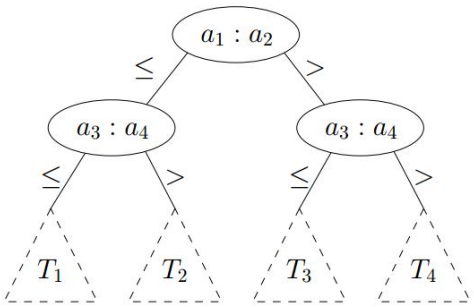
1. 把这 $n$ 个数排序，随后输出排序后的前 $k$ 个数。
2. 把这 $n$ 个数插入到一个初始为空的最小堆中，之后再调用 $k$ 次提取最小值的操作 ($Extract - Min$)。
3. 用线性时间复杂度的算法来构建一个这 $n$ 个数的最小堆，之后调用 $k$ 次提取最小值的操作 ($Extract - Min$)。
4. 请设计一种比上述三种算法更高效的算法来解决这个问题。[提示：可采用随机化用线性时间选择算法 ($RandomizedLinear - TimeSelectionAlgorithm$)]

解：

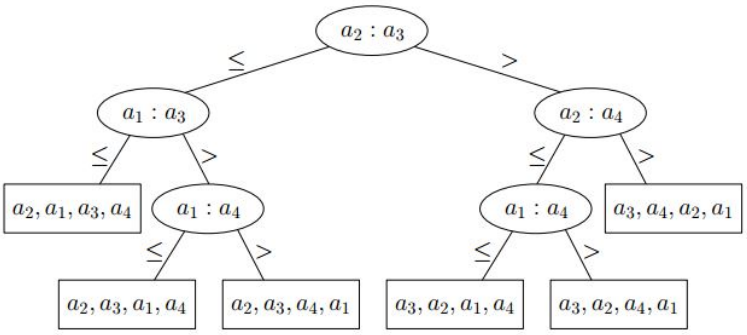
1. $O(n \log n)$(5 分)
2. 通过 n 次插入操作建堆的时间复杂度为 $O(n \log n)$，调用 $k$ 次 $Extract - Min$ 操作的时间复杂度为 $O(k \log n)$。总的时间复杂度为 $O(n \log n)$。(5 分)
3. 以线性时间建堆的时间复杂度为 $O(n)$，调用 $k$ 次 $Extract - Min$ 操作的时间复杂度为 $O(k \log n)$。总的时间复杂度为 $O(n + k \log n)$。(5 分)
4. 首先使用线性时间选择算法找出集合中的第 $k$ 小的数 (记 $a_k$)，这需要 $O(n)$ 的时间。接下来用 $O(n)$ 的时间找出集合中所有小于等于 $a_k$ 的数。最后我们仅需对这 $k$ 个数进行排序，时间复杂度为 $O(k \log k)$。总的时间复杂度为 $O(n + k \log k)$。(如果只找出了 $k$ 个数而没有进行排序也可。)(10 分)

## 2 决策树 (25 分)

下图中展示了对包含四个元素 $a_1, a_2, a_3, a_4$ 的数组进行归并排序的决策树的一部分，请你完成子树 $T_3$ 的部分。




解：




## 3 桶排序问题 (25 分)

**桶排序**是另一种不基于比较的排序，他的算法流程如下所示。

假设要给 $n$ 个数 $a_1, a_2, \ldots, a_n$ 进行排序，且这 $n$ 个数都在集合 $\{0, 1, \ldots, k-1\}$ 中取值。我们创建 10 个“桶” $B_0, B_1, \ldots, B_9$，之后对每个 $a_i$，找他的第一位 (我们通过添加前导 0 使每个数都有 $\lceil \log_{10} k \rceil$ 位)，并把他扔进相应的桶中。例如，这 $n$ 个数分别为 $69, 4, 99, 12, 65$，那么经过这一步操作，桶中所含元素的情况为：$B_0 = 4, B_1 = 12, B_6 = 69, 65, B_9 = 99$，其他桶仍为空。很容易发现，$a_i$ 所属的桶编号可以通过 $\lfloor a_i/(k/10) \rfloor$ 来计算 (假定 $k$ 为 10 的倍数)。每个桶使用链表结构来存储，因此可以用 $O(1)$ 的时间来实现向桶中插入一个元素以及从桶中删除一个元素。

在把元素放入相应桶中之后，从 $B_0$ 开始，输出 10 个桶中的所有元素到一个数组 $A$ 中，这样，所有的数已经按照他们的第一位排好序了，并且每一个非空的桶都占据了该数组中的连续一段。接下来，我们再用同样的算法继续对每一个连续段进行排序，只不过这次我们关注数的第二位。一般地，在算法的第 $i$ 次迭代中，我们关注此时每个连续段中数字的第 $i$ 位。在做完 $\lceil \log_{10} k \rceil$ 次迭代后，我们就得到了排好序的数组。

这个排序算法也叫做**邮递员算法**，因为这种算法和邮递员根据邮政编码划分信件的方式非常相似：首先根据邮编第一位进行划分，之后根据邮编第二位进行划分，...，直到分类完成。

1. 请分析这个算法在最坏情况下的运行时间。[请注意《算法导论》第 8.4 节介绍了桶排序算法，并分析了该算法的输入在服从均匀分布条件下的平均情况下的运行时间。但是，本题要求大家分析最坏情况下的运行时间。]
2. 这种算法固定使用 10 个桶来进行排序，可能并不是最好的选择。最好情况下应使用几个桶呢？请分析这种情况下的时间复杂度 (用关于 $k$ 和 $n$ 的函数表示)。可以假定 $k > n$。

解：

1. 将 $n$ 个数分入 10 个桶中的时间复杂度为 $O(n)$。将这些数从桶中输出的时间复杂度也为 $O(n)$。因此每一轮迭代都花费 $O(n)$ 的时间。迭代的次数和最大数的位数相等，为 $log_{10} k = O(\log k)$。因此总的时间复杂度为 $O(n \log k)$。(10 分)
2. 当数组均匀分布时，算法具有最好的复杂度。注意在算法在某一次迭代过程中，将桶内元素取回至数组的复杂度为 $n' + p$，其中 $n'$ 为此次迭代的元素数量，$p$ 为桶的个数。在最好的复杂度下，迭代过程中每个桶分得的元素个数为 $n'/p$，因此有递推式 $T(n) = p \cdot T(n/p) + n + p$。对于这个递推式，当 $p \leq n$ 时，递推式退化成 $T(n) = p \cdot T(n/p) + O(n)$(但是迭代次数是 $\log_p k$)，此时复杂度是 $n \cdot \log_p k$；当 $p > n$ 时，递推式变为 $T(n) = p \cdot T(n/p) + O(p)$，注意这里 $p$ 相当于一个常数，且仍需要迭代 $\log_p k$ 次，此时复杂度是 $p \cdot \log_p k$。综合两方面结果可知，当 $p = n$ 时最好情况下具有最小的复杂度 $O(n \cdot \log_n k)$。

## 4 环路问题 (22 分)

给出一个联通无向图 $G = (V, E)$，请设计一种尽可能高效的算法来判断 $G$ 中是否有环。如果有，请输出任意一个环 (按顺序给出环上的每个顶点)。请解释算法的正确性并分析算法的时间复杂度。

解：

选出任意一点 $s \in V$，从该点开始在图 $G$ 上进行深度优先搜索 ($DFS$)。当搜索过程中遇到一条反向边 $(u, v)$，我们就可以得出结论：$G$ 中存在环。接下来从 $u$ 开始，输出当前搜索栈中记录的所有点，直到 $v$ 结束，这就是 $G$ 中的一个环。

正确性：无向图中的边要么是树边，要么是反向边 (定理证明见 Lecture09)，如果图中有环，那么一定存在一条反向边。如果图中没有环，意味着该图是一个树结构，不存在反向边。

时间复杂度：该算法的时间复杂度为 $O(|V|)$。因为在该算法的执行过程中所有的点至多被访问一次，因此我们最多也仅会访问 $O(|V|)$ 条边。(如果此处复杂度分析为 $O(|V| + |E|)$ 会扣 2 分)

算法的伪代码可参考 **Algorithm 2**：

**Algorithm 1** $Visit(u)$

```
color[u] ← GRAY;
for v ∈ Adj(u) do
    if color[v] = WHITE then
        pred[v] ← u;
    else
        if v ≠ pred[u] then
            begin ← u;
            end ← v;
            return ;
        end if
    end if
    if begin ≠ NULL then
        return ;
    end if
end for
```

**Algorithm 2** $Cycle(G)$

```
for u ∈ V do
    color[u] ← WHITE;
    pred[u] ← NULL;
end for
begin ← NULL;
end ← NULL;
Visit(1);
if end = NULL then
    output No Cycle;
else
    while begin ≠ end do
        output begin;
        begin ← pred[begin];
    end while
    output end;
end if
```