问题求解（二）作业（第十一周）

161180162 许致明

2018 年 5 月 19 日

TC 第六章

6.1-2

$$
\begin{aligned}
\exists p, s.t.\ n + p &= 2^h - 1 \\
h &= \log(n + p + 1) \\
\therefore h &< \log n \\
\text{又} \because h &> h - 1 \\
\therefore h &> \log 2^{n-1} \\
h > \log \frac{n+p+1}{2} &> \log \frac{n}{2} \\
\therefore h &> \log -1 \\
\text{综合得，} \log n - 1 < h &< \log n \\
\therefore n &= \lfloor \log n \rfloor
\end{aligned}
$$

6.1-4

在叶节点中，即数组下标为 $\lceil n/2 \rceil \sim n$ 的元素中。

6.1-7

对于下标在 $0 \sim \lfloor n/2 \rfloor$ 的元素，求出的下标仍在原数组中。而对 $\lfloor n/2 \rfloor + \sim n$ 的元素，求出的孩子下标越界，即它们都是叶节点。

6.2-2

```
MIN-HEAPIFY(A, i)
l = LEFT(i)
r = RIGHT(i)
if l ≤ A.heapsize and A[l] < A[i]
    smallest = l
else
    smallest = i
if r ≤ A.heapsize and A[r] < A[i]
    smallest = r
if smallest ≠ i
    SWAP(A[i], A[smallest])
    MIN-HEAPIFY(A, smallest)
```

运行时间不变。

6.2-5

```
MAX-HEAPIFY(A, i)
while TRUE
    l = LEFT(i)
    r = RIGHT(i)
    if l ≤ A.heapsize and A[l] > A[i]
        largest = l
    else
        largest = i
    if r ≤ A.heapsize and A[r] > A[i]
        largest = r
    if largest ≠ i
        SWAP(A[i], A[largest])
        i = largest
```

6.2-6

将最小值放置在根节点，且对于每个非叶节点，保证左孩子大于右孩子。对此堆的根做 MAX-HEAPIFY，共需 h 次（h 为树的高度），则 $h = \lfloor \log n \rfloor$。且每次运行均消耗常数时间，故最坏情况的运行时间为 $\Omega(\log n)$

6.3-3

(1) 奠基：

$$h = 1, \lceil n/2^2 \rceil = \lceil 1/4 \rceil = 1$$

(2) 归纳假设：

$$h = k, \text{ 高度为 } h \text{ 的节点数量 } m \text{ 不大于} \lceil n/2^{k+1} \rceil$$

(3)

$$h = k+1,\ m \le \lceil n/2^{(h+1)+1} \rceil = \lceil n/2^{h+2} \rceil$$

故至多有 $\lceil n/2^{h+1} \rceil$ 个高度为 h 的节点。

6.4-2

证明. (1) 初始：$i = A.length$，最大堆已经建好，$A[1..A.length]$ 包含所有元素，$A[i+1..n]$ 包含 0 个最大元素，$n - A.length = 0$。成立；

(2) 保持：假设在第 $n-i$ 次循环前，$A[1..i]$ 含全部元素中第 i 小的，$A[i+1..n]$ 含已排序的 $n-i$ 个最大元素。在第 i 次循环中，此时 $A[1]$ 是第 $(n-i+1)$ 大的元素，也是第 i 小的元素。将它与 $A[i]$ 交换，使得 $A[1..n]$ 含已排序的 $n-i+1$ 个最大元素。由此可知，第 $n-1$ 小的元素在 $A[1..i-1]$ 中，故在第 $n-(i-1)$ 次循环前，不变式成立；

(3) 终止：循环进行 $n-1$ 次后停止，此时 $i = 2$，由不变式，$A[2..n]$ 中是 $n-1$ 个最大元素排列好的序列，只有最小的元素在首位。因此，整个数组已经排好序。

□

6.4-4

$$T(\textsc{Heapsort}) = T(\textsc{Build}) + n \cdot T(\textsc{Max-Heapify})$$

$$T(\textsc{Max-Heapify}) = \Omega(\log n)$$

$$T(\textsc{Heapsort}) = \sum_{i=1}^{n-1} \Omega(\log i) = \Omega(\log(n-1)!) = \Omega(n \log n)$$

6.5-5

证明. (1) 初始：由假定，满足不变式；

(2) 保持：假设第 t 次循环前，满足不变式。

1. 若此时 $A[1..A.length]$ 满足最大堆的性质，则 $A[\textsc{Parent(i)}] > A[i]$，退出循环，不变式依然满足；
2. 若此时 $A[i] > A[\textsc{Parent(i)}]$，则进入循环，并将两者调换位置，再使得 i 成为父节点。调换后堆其他位置的性质不变，以原本的子节点为根的树满足最大堆性质，此时不确定的只有原本的父节点的位置，即现在的 $A[i]$。

故每次循环能保持不变式；

(3) 终止：每次循环后有两种情况可以终止：$i \le 1$ 或 $Ap\textsc{Parent(i)} > A[i]$。当后者满足时，由不变式可得 $A[1..A.length]$ 是最大堆。若后者不满足，每次将 i 的值减半，最终使得 $i \le 1$。前者满足时，i 的父节点不存在，不会违背条件。故 A 是最大堆。

□

6.5-7

FIFO 队列：将入队（放入优先队列）的时间作为 value，建立 value 小优先的优先队列；
栈：将入栈（放入优先队列）的时间作为 value，建立 value 大优先的优先队列。

6.5-9

利用每个链表头部的元素（共 k）个建立最小堆，每次取出堆里最小的元素。然后从取出元素所在的链表中拿出下一个元素添加进最小堆（前述过程的时间复杂度为 $O(\log k)$）。如此重复，直至 k 个链表中的元素全部被取出。因此，总时间复杂度为 $n \cdot O(\log k) = O(n \log k)$。