

第三章练习参考答案

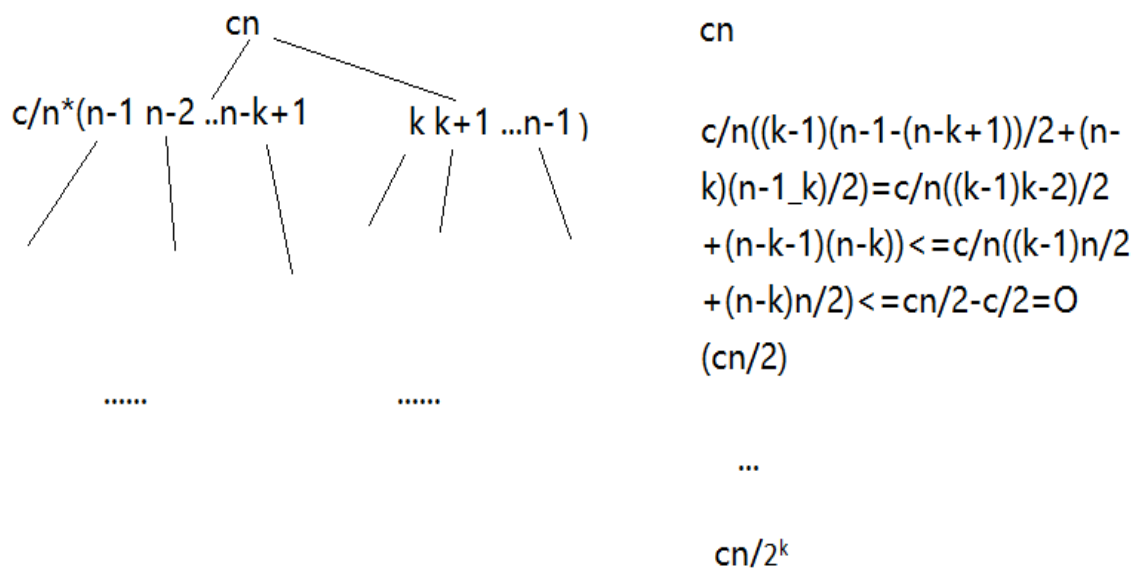
一、讲义习题三

4 证:

(1) 证明见第九章概率算法之 ppt No.17 舍伍德算法选择算法。

(2) 可使用递归树方

法:



所以, $C_a^k(n) \leq cn + cn/2 + \dots + cn/2^k = cn(1 - (1/2)^k) / (1 - 1/2) < 2cn$

$$C_a^k(n) = O(n)$$

二

1. 解: 算法伪码:

ModInsertSort(A[1..n])

for i:=2 to n do

x:=A[i]

k:=BinarySearch(A[1..i-1],x)

//在 A[1..i-1]查找 x 应该插入的位置 k

A[k]:=x //

复杂度分析：外 for 循环 $n-1$ 次，在 $i-1$ 规模的数组中二分比较次数为 $\leq \log(i-1)+1$ ，因此总比较次数为：（但移动次数没节省）

$$\sum_{i=2}^n \log(i-1)+1 = n-1 + \sum_{i=1}^{n-1} \log i \leq n-1 + \sum_{i=1}^{n-1} \log n = n-1 + (n-1) \log n = O(n \log n)$$

2. 解：类似快速排序的划分过程。从后向前把每个数与 0 比较，找到第一个负数 A[p];从前向后把每个数与 0 比较，找到第一个正数 A[q]，如果 $p > q$ ，则将 A[p]与 A[q]交换。交换后如果 $p-q=1$ ，算法停止，否则继续这个过程。

3. 解：Hanoi(A,C,n)

If $n=1$ then move(A,C)

Else Hanoi(A,B,n-1)

Move(A,C)

Hanoi(B,C,n-1)

算法复杂度： $T(n)=2T(n-1)+1=4T(n-2)+2+1=2^{n-1}+2^{n-2}+\dots+1=2^n-1$

4. 解：因为 L 中存在峰顶元素，因此 $|L| \geq 3$ 。使用二分查找算法。
如元素数等于 3，则 L[2]是峰顶元素，当元素数 $n > 3$ 时，令 $k = \lfloor n/2 \rfloor$ ，比较 L[k]与它左边和右边相邻的项，如果 $L[k] > L[k-1]$ 且 $L[k] > L[k+1]$ 则 L[k]为峰顶元素；否则，如果 $L[k-1] > L[k] > L[k+1]$ ，则继续搜索 L[1..k-1]，如果 $L[k-1] < L[k] < L[k+1]$ 则继续搜索 L[k+1..n]的范围。每比较两次，搜索范围减半，直到元素数小

于 3 停止递归调用。

时间复杂度 $T(n)=T(n/2)+2$, 根据主定理, $T(n)=O(\log n)$ 。

5. 解: 在 A 中使用二分查找算法找 L, 如果 $L=A[i]$, 找到 L 的位置 i, 然后把 i 加 1; 如果 L 不在 A 中, 那么找到大于 L 的最小数的位置 i。类似地, 找到 U 的位置 $A[j]$, $j=j+1$, 或小于 U 的最大数 $A[j]$ 。输出 A 中 i 到 j 的全体数。

(例: 关于 L 的返回条件:

```
if (left==right) return left+1

Else if left+1==right{

    If L=a(left) return right

    Else L=a(right) return right+1

    Else return right

}

End if )
```

6. 解: 采用分治策略。如果 $n<3$, 那么将拿走的硬币与剩下的硬币比较, 不等的是坏币; 将 n 枚硬币分为大致相等的 3 份, 如果 $n \bmod 3 \neq 0$, 那么令两份少的硬币相等。取两份相等的硬币放到天平上, 如果不等, 那么这两份硬币中含有坏币, 否则坏币在第 3 份中。递归处理, 直到 $n<3$ 为止。

算法: Coin(A,n)

1: $K:=\lfloor n/3 \rfloor$

将 A 中硬币划分为 X、Y、Z 三个集合, 使得 $|X|=|Y|=k, z=n-2k$

If $W(X) \neq W(Y)$ // $W(X)$ 、 $W(Y)$ 是 X 、 Y 的重量

Then $A := X \cup Y$

Else $A := Z$

$n := |A|$

if $n > 2$ then goto 1

else 将 A 中硬币与拿走的比较，不等者为坏币

算法复杂度： $T(n) = T(2n/3) + O(1)$, $T(1) = 0$, $T(2) = 1$

根据主定理， $T(n) = O(\log n)$

7. 解：在 A 与 B 中选择一个数组排序，然后循序对另一个数组的每个元素使用二分查找法，看它是否在这个数组中出现。如果出现，则将它放入 C 。算法的主要消耗是排序，所以选择较小的数组 B 进行排序。

算法： $C := \{\}$

$L := \text{Sort}(B)$

For $i := 1$ to n do {

$X := A[i]$

$J := \text{binarySearch}(L, x)$

// 在 L 中二分查找 x ，若 x 在 L 中， j 为序标，否则 j 为 0。

If $j > 0$ then $C := C \cup \{x\}$

}

算法复杂度：排序 $O(m \log m)$ ，for 循环运行 n 次，内部二分查找 $O(\log m)$ ，for 循环关键操作 $O(n \log m)$ ，于是

$$T(n)=O(m\log m)+O(n\log m)=O((m+n)\log m)=O(n\log m)$$

8. 解：算法设计思想：规定 S 的中位数 x 是从小到大排序的第 $n/2$ 个数，用 x 划分 S ，比 x 小的整数属于 S_1 ， x 本身也放到 S_1 ，其余的放到 S_2 ，由于 n 是偶数， $|S_1|=|S_2|$ ，易见这样的集合满足要求。

算法复杂度：找中位数和划分都是 $O(n)$ ，所以 $T(n)=O(n)$ 。

9. 解：(1) $r=3$ ，不妨设 n 是 3 的倍数。每组至少 2 个元素不大于 u ， A 中至少 $2*\lceil n/3 \rceil / 2 \geq \lfloor n/3 \rfloor = n/3$ 个不大于 v ，即 A 中至多 $n - \lfloor n/3 \rfloor \leq n - n/3 = 2n/3$ 个元素大于 v 。同理，至多有 $2n/3$ 个元素小于 v 。即子问题的规模小于 $2n/3$ 。所以， $T(n)=T(n/3)+T(2n/3)+O(n)$ ，得 $T(n)=O(n\log n)$ 。与直接排序方法的复杂度一样。

(2)问题变为 $4*\lceil n/7 \rceil / 2 \geq 2\lfloor n/7 \rfloor$ ，子问题规模小于 $n - 2n/7 = 5n/7$ （不妨设 n 是 7 的倍数）
 $T(n)=T(n/7)+T(5n/7)+O(n)$
 $=n(1+6/7+(6/7)^2+\dots+(6/7)^k)+O(n)=O(n)$

10. 解：算法的主要思想：在二分归并排序算法中附加计数逆序的工作。在递归调用算法分别对数组 $L1$ 和 $L2$ 排序时，分别计数每个子数组内部的逆序；在归并排好序的子数组 $L1$ 与 $L2$ 的过程中，附带计算 $L1$ 的元素和 $L2$ 的元素之间产生的逆序。假设 $L1$ 是前半数组， $L2$ 是后半数组，如果 $L1$ 的最小元素 x 大于 $L2$ 的最小元素 y ，那么算法将从 $L2$ 中取走 y ，这时 $L1$ 中的每个元素都和 y 构成逆序，所增加的逆序数正好等于此刻 $L1$ 中的元素数；相反，如果 $L1$ 中最小元素 x 小于 $L2$ 最小元素 y ，从 $L1$ 中取走 x ，这时 $L2$ 中的任何元素都与 x 不构成逆序，逆序数不增加。算法描述：初始 $N:=0$

1)将 L 从中间划分为 L1, L2

2)递归处理 L1;

3)递归处理 L2;

4)在归并 L1、L2 时计数 L1 与 L2 产生的逆序数 m, $N:=N+m$

算法复杂度: $T(n)=2T(n/2)+n-1=O(n\log n)$

11. 解: (1)只好顺序从下到上测试, 一次一个高度, 最坏 $T(n)=O(n)$

(2)二分查找法。取 $n/2$ 高度进行第一次测试, 如果瓶子没有摔碎, 则强度在 $[n/2+1, n]$ 之间, 否则在 $[1, n/2]$ 之间。每次测试后可能一个瓶子的代价, 测试范围减半, 最坏时间复杂度 $T(n)=O(\log n)$ 。

(3) 为简单起见, 不妨设 \sqrt{n} 为整数, 将高度 $1, 2, \dots, n$ 分为 \sqrt{n} 个组, 每组 \sqrt{n} 高度, 取第一个瓶子从下到上测试每组的最大高度, 即高度 $\sqrt{n}, 2\sqrt{n} \dots n$, 如果 $k-1$ 组没碎, k 组碎了, 那么玻璃瓶子的强度在第 k 组内, 于是, 再经至多 \sqrt{n} 次测试, 就可以得到瓶子的强度。

$$T(n)=O(\sqrt{n})+O(\sqrt{n})=O(\sqrt{n})$$

12. 解:

(1) $a=9, b=3, f(n)=n$, $\log 3^9=2$, $f(n)$ 的阶低于 $n \log b^a$, 符合情况 1,

$$T(n)=\Theta(n \log b^a)=\Theta(n^2)。$$

(2) $a=5, b=2, f(n)=n^2 \log^2 n=O(n \log 2^5 - \epsilon)$, $T(n)=\Theta(n \log 2^5)$

(3) $a=2, b=2, f(n)=n^2 \log n$, 取 $c=3/4$ 则

$$af(n/b)=2(n/2)^2 \log(n/2)=(n^2/2)(\log n - 1) \leq (n^2/2) \log n \leq cn^2 \log n = cf(n)$$

于是, 符合情况 3, $T(n)=\Theta(n^2 \log n)$

13. 解: 使用递归树可得:

$$T(n) = cn + 3cn/4 + (3/4)^2 cn + (3/4)^3 cn + \dots = [1 + 3/4 + (3/4)^2 + (3/4)^3 + \dots] cn = \Theta(n)$$

14. 解:

$$(1) T(n) = n \log 3 + (n-1) \log 3 + T(n-2) = \log 3 (n + (n-1) + (n-2) + \dots + 1) = \Theta(n^2)$$

$$(2) T(n) = T(n-1) + 1/n = T(n-2) + 1/(n-1) + 1/n = \dots$$

$$= 1/n + 1/(n-1) + \dots + 1/2 + 1 = \Theta(\log n)$$