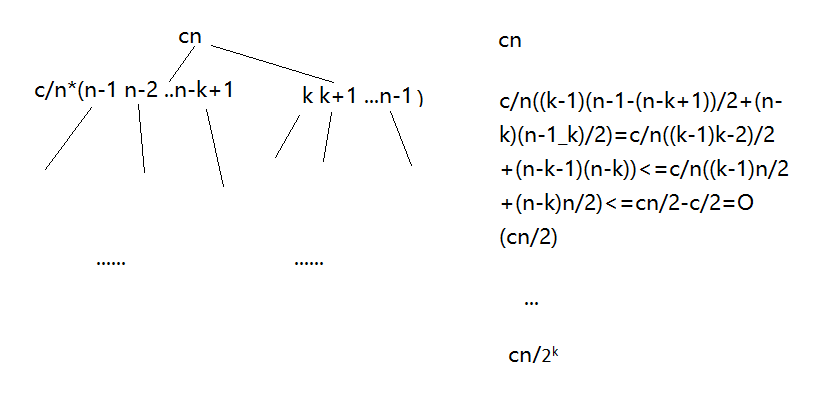
第三章练习参考答案

一、讲义习题三

4 证:

（1）证明见第九章概率算法之ppt No.17 舍伍得算法选择算法。

（2）可使用递归树方法:

所以，Cak(n)<=cn+cn/2+…+cn/2k=cn(1-(1/2)k)/(1-1/2) <2cn

Cak(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<=log(i-1)+1，因此总比较次数为：（但移动次数没节省）



1. 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=2n-1+2n-2+…+1=2n-1

1. 解：因为L中存在峰顶元素，因此|L|>=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=⎣n/2⎦，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(logn)。

1. 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 ）

1. 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(mod 3)≠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:= ⎣n/3⎦

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)≠W(Y) //W(X)、W(Y)是X、Y的重量

Then A:=X∪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(logn)

1. 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:=Sort(B)

For i:=1 to n do {

X:=A[i]

J:=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∪{x}

}

算法复杂度：排序O(mlogm)，for循环运行n次，内部二分查找 O(logm)，for循环关键操作O(nlogm)，于是

T(n)=O(mlogm)+O(nlogm)=O((m+n)logm)=O(nlogm)

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S1，x本身也放到S1，其余的放到S2，由于n是偶数，|S1|=|S2|，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\*⎡⎣n/3⎦/2⎤ >= ⎣n/3⎦=n/3个不大于v， 即A中至多n-⎣n/3⎦<=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(nlogn)。与直接排序方法的复杂度一样。

(2)问题变为4\*⎡⎣n/7⎦/2⎤ >=2 ⎣n/7⎦，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+…+(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是前半个数组，L2是后半个数组，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(nlogn)

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(logn)。

(3) 为简单起见，不妨设为整数，将高度1,2,..,n分为个组，每组高度，取第一个瓶子从下到上测试每组的最大高度，即高度,2…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，再经至多次测试，就可以得到瓶子的强度。

T(n)=O()+O()=O()

12. 解：

(1)a=9,b=3,f(n)=n，log39=2,f(n)的阶低于nlogba,符合情况1，T(n)=Θ( nlogba)= Θ(n2)。

(2)a=5,b=2,f(n)=n2log2n=O(nlog25-ε)，T(n)= Θ(nlog25)

(3)a=2,b=2,f(n)=n2logn,取c=3/4则

af(n/b)=2(n/2)2log(n/2)=(n2/2)(logn-1)≤(n2/2)logn≤cn2logn=cf(n)

于是，符合情况3，T(n)= Θ(n2logn)

13. 解：使用递归树可得：

T(n)=cn+3cn/4+(3/4)2cn+(3/4)3cn+…..=[1+3/4+(3/4)2+(3/4)3+…]cn=Θ(n)

14. 解：

(1)T(n)=nlog3+(n-1)log3+T(n-2)=log3(n+(n-1)+(n-2)+…+1)= Θ(n2)

(2)T(n)=T(n-1)+1/n=T(n-2)+1/(n-1)+1/n=…

=1/n+1/(n-1)+…+1/2+1=Θ(logn)