算法分析

郑智润

湘潭大学数学与计算科学学院，湘潭， 411100

2.8 假设我们需要生成前N个整数的一个随机排列.例如,$\{4,3,1,5,2\}$和$\{3,1,4,2,5\}$就是合法的排列,但$\{5,4,1,2,1\}$则不是,因为数1出现两次而数3却没有出现.这个程序常常用于模拟一些算法.假设存在一个随机数生成器r,它有方法$randInt(i,j)$, 该方法以相同的概率生成i和j之间的整数.下面是3个算法:

1. 如下填入从$a[0]$到$a[n-1]$的数组a.为了填入$a[i]$,生成随机数直到它不同于已经生成的$a[0],a[1],\cdots,a[i-1]$时再将 其填入$a[i]$.时间复杂度为$O(N^3)$.

2. 同算法1,但要保存一个附加的数组,称之为$used$数组.当一个随机数ran最初被放入数组a的时候,置$used[ran]=true$.这就是说, 当用一个随机数填入$a[i]$时,可以用一步来测试是否该随机数已经被使用,而不是像第一个算法那样(可能)用i步测试.时间复杂度为$O(N^2)$.

3. 填写该数组使得$a[i]=i+1$,然后

$$for(i = 1;\ i < n;\ ++i) \qquad swap(a[i],a[randInt(0,i)]);$$

时间复杂度为$O(N)$.

答案: 代码 exercise2.8.cpp

2.13 计算

$$f(x)=\sum_{i=0}^{N} a_i x^i.$$

1. 采用$x^n = x\times x\times\cdots\times x$方式计算$x$的$n$次幂.求幂运算的时间复杂度为$O(N)$,求上述多项式的时间复杂度为$O(N^2)$.

2. 采用递归算法使求幂的时间复杂度降为$O(logN)$.$N\leq 1$是这种递归的基准情形.否则,若N是偶数,我们 有$x^N = x^{N/2}\cdot x^{N/2}$;如果$N$是奇数,则$x^N = x^{(N-1)/2}\cdot x^{(N-1)/2}\cdot x$.求上述多项式的时间复杂度为 $O(NlogN)$.

3. 采用Horner法则计算上述多项式时间复杂度为$O(N)$.

$$\sum_{i=0}^{N} a_i x^i = ((((a_N)\times x + a_{N-1})\times x + a_{N-2})\cdots)\times x + a_0.$$

答案: 代码 exercise2.13.cpp

2.15 给出一个有效的算法来确定在整数$A_1 < A_2 < A_3 < \cdots < A_N$的数组中是否在整数$i$使得$A_i = i$.你的算法运行时间是多少?

答案:采用二分策略进行查找,时间复杂度为$O(logN)$,代码 exercise2.15.cpp

相关作者：

E-mail:jiangxizhengzhirun@163.com (郑智润)

2.17　给出有效的算法(及其运行时间分析)来:

1. 找出最小子序列和.

2. 找出最小正子序列和.

3. 找出最大序列乘积.

答案:

1.exercise2.17a.cpp给出三种算法分别为暴力搜索$O(N^2)$,递归分治策略$O(NlogN)$,联机算法$O(N)$.

2.exercise2.17b.cpp给出三种算法分别为暴力搜索$O(N^2)$,$O(NlogN)$.

3.exercise2.18c.cpp给出两种算法分别为暴力搜索$O(N^2)$,联机算法$O(N)$.

2.18　数值分析中一个重要的问题是对某个任意的函数f找出方程$f(x)=0$的一个解.如果该函数是连续的并有两个点low和 $high$使得$f(low)$和$f(high)$符号相反,那么在low和$high$之间必然存在一个根,并且这个根可以通过折半查找求得.写出一个 函数,以$f,low,high$为参数,并且解出一个零点.为保证能够正常终止,你必须做什么?

答案:代码 exercise2.18.cpp

2.19　正文中最大相连子序列和算法均不给出具体序列的任何指示.对这些算法进行修改,使得它们以单个对象的形式返回最大子序列的值以及 具体序列的那些相应的下标.

答案:代码 exercise2.19.cpp