算法与复杂性　作业六

516021910528 - SHEN Jiamin

2020 年 3 月 24 日

1　0323

1. 设计算法将 1 到 n^2 按顺时针方向由内而外填入一个 $n \times n$ 的矩阵

记该矩阵为 $A[1..n][1..n]$。

算法 1 螺旋方阵

输入: 空间 $A[1..n][1..n]$

SpiralMatrix($A, 0, 0, n$)

procedure SpiralMatrix($M, top, left, size$) ▷ 向以 $M[top][left]$ 为 $A[0][0]$ 的 $size$ 维方阵中填入元素
　if $n = 1$ **then**
　　$M[top][left] \leftarrow 1$
　else if $[n \bmod 2] = 1$ **then** ▷ 去掉第一列和最后一行后，剩余的元素为 $n-1$ 维的方阵
　　SpiralMatrix($A, top, left+1, n-1$) ▷ 以左上角右侧的元素为基准
　　for $i \leftarrow 1$ to n **do**
　　　$M[n][i] \leftarrow (n-1)^2 + n - (i-1)$ ▷ 填充下边
　　　$M[i][1] \leftarrow n^2 - (i-1)$ ▷ 填充左边
　　end for
　else ▷ 去掉第一行和最后一行后，剩余的元素为 $n-1$ 维的方阵
　　SpiralMatrix($A, top+1, left, n-1$) ▷ 以左上角下侧的元素为基准
　　for $i \leftarrow 1$ to n **do**
　　　$M[1][i] \leftarrow (n-1)^2 + i$ ▷ 填充上边
　　　$M[i][n] \leftarrow (n-1)^2 + n + (i-1)$ ▷ 填充右边
　　end for
　end if
end procedure

2. 给定整数数组 $A[1..n]$，相邻两个元素的值最多相差 1。设 $A[1]=x$，$A[n]=y$，并且 $x<y$，输入 z，$x\le z\le y$，判断 z 在数组 A 中出现的位置。给出算法及时间复杂性（不得穷举）

相邻两个元素的值最多相差 1，即有 $A[i+1]-A[i]\in\{-1,0,1\}$。可以据此类比连续函数上的零点存在性定理，即

$$\text{若 } A[i]\le z\le A[j] \text{ 且 } i<j\text{，则一定存在一个 } m: i\le m\le j \text{ 使得 } A[m]=z$$

因此可用二分法，有如下算法:

算法 2

输入: 整数数组 $A[1..n]$, A 中的一个整数 z

输出: 元素 z 在数组 A 中的位置

$lidx \leftarrow 1, ridx \leftarrow n$
$mid \leftarrow \left\lfloor \frac{lidx+ridx}{2} \right\rfloor$
while $A[mid]\neq z$ **do**
    if $A[mid]<z$ **then**
        $left \leftarrow mid$ ▷ 更新后仍有 $A[left]<z$
    else ▷ $A[mid]>z$
        $right \leftarrow mid$ ▷ 更新后仍有 $A[right]>z$
    end if
    $mid \leftarrow \left\lfloor \frac{lidx+ridx}{2} \right\rfloor$
end while
return mid

该算法的时间复杂度为 $O(\log n)$

3. 假设有 k 个长度为 n 的有序序列，采用下述方法将这些序列合并成一个具有 kn 个元素的有序序列：将前两个序列合并，再并入第三个序列，然后是第四个……直至所有序列合并。

   (a) 用 k 和 n 表示该方法的时间复杂性

   当合并一个长度为 m 和一个长度为 n 的序列时

   - 赋值操作执行了 $m+n$ 次
   - 比较操作至少执行 $\min\{m,n\}$ 次，最多执行 $2\min\{m,n\}$ 次

   所以此算法合并 k 个长度为 n 的有序序列所需时间为

$$
\begin{aligned}
T(0,k,n)=T(n,k-1,n) &= 2n+cn+T(2n,k-2,n)\\
&=(2+3)n+2cn+T(3n,k-3,n)\\
&=\dots\\
&=\sum_{i=2}^{k} i\cdot n+(k-1)cn\\
&=\frac{(k+2)(k-1)}{2}n+(k-1)cn=O(k^2n)
\end{aligned}
$$

   (b) 能不能得到一个效率更高的方法来合并这 k 个序列?

   首先将第奇数个序列和第偶数个序列合并，形成 $\lceil \frac{k}{2} \rceil$ 个最大长度为 $2n$ 的序列；然后重复上一步骤，直至最终只剩 1 个序列。
   不妨设 $k=2^m$

$$
\begin{aligned}
T(k,n)=\frac{k}{2}T(2,n)+T\left(\frac{k}{2},2n\right) &= 2^{m-1}(2+c)n+T\left(2^{m-1},2n\right)\\
&=2^{m-1}(2+c)n+2^{m-2}2(2+c)n+T\left(2^{m-2},4n\right)\\
&=2\cdot 2^{m-1}(2+c)n+T\left(2^{m-2},2^2n\right)\\
&=\dots\\
&=(m-1)\cdot 2^{m-1}(2+c)n+T\left(2,2^{m-1}n\right)\\
&=m\cdot 2^{m-1}(2+c)n=O(kn\log k)
\end{aligned}
$$

4. 设 $A[1..n]$ 是一个包含 n 个不同数的数组。如果在 $i < j$ 的情况下有 $A[i] > A[j]$，则 (i, j) 为 A 中的一个逆序对。

(a) 若 $A = \{2, 3, 8, 6, 1\}$，列出其中所有的逆序对

$(2,1)$ $(3,1)$ $(8,6)$ $(8,1)$ $(6,1)$

(b) 若数组元素取自 $\{1, 2, \ldots, n\}$，那么怎样的数组含有最多的逆序对？它包含多少个逆序对？

当数组为递减序列时包含有最多的逆序对，此时序列中任取一对数，都能构成逆序对。逆序对数为 $\frac{n(n-1)}{2}$。

(c) 求任意数组 A 中逆序对的数目，并证明算法的时间复杂性为 $\Theta(n \log n)$

应用分治法，结合归并排序。首先将数组等分成左右两个子序列，分别求出两边的逆序对数（并同时进行排序）。然后再求出分属于两个子序列的逆序对，即对每一个左序列中的元素，找出右序列中大于该元素的所有元素个数。

$$T(n) = 2T(n/2) + (2 + c)n = O(n \log n)$$

算法 3 求逆序对数

输入: 数组 $A[1..n]$

输出: A 中的逆序对数 m

```
(m, _) ← MergeInversions(A[1..n])

procedure MergeInversions(A[1..n])
                  ▷ 输入一个数组 A，输出一个二元组，包含该数组中的逆序对数及一个有序的该数组
    if A.size = 1 then
        return (0, A)
    else if A.size = 2 then
        if A[1] > A[2] then
            return (1, [A[2], A[1]])
        else
            return (0, A)
        end if
    else                                                        ▷ 分治
        (nL, L) ← MergeInversions(A [1.. ⌊n/2⌋])
        (nR, R) ← MergeInversions(A [⌊n/2⌋ + 1..n])
        k ← 1, cnt ← 0                                          ▷ 归并
        while ¬L.empty ∧ ¬R.empty do
            if L.front ≤ R.front then
                A[k] = L.PopFront()
            else                                    ▷ L[1] 大于 R 的所有剩余元素
                A[k] = R.PopFront()
                cnt ← cnt + R.size              ▷ L[1] 与 R 中所有剩余元素都构成逆序对
            end if
            k ← k + 1
        end while
        if ¬L.empty then                                        ▷ 填充剩余元素
            A[k..n] = L
        else if ¬R.empty then
            A[k..n] = R
        end if
        return (cnt, A)
    end if
end procedure
```

2 0326

5. 证明二分查找是有序序列查找的最优算法

在一个长度为 n 的有序数组中查找一个元素，该元素的位置为 i 的概率为 $p_i = \frac{1}{n}$。因此，**在没有附加信息的前提下**，事件“确定一个元素的位置”能提供的信息至少为 $I_e = -\log p_i = \log n$。而一次比较只能给出真、假两种结果，即一次比较能提供的信息为 $I_c = \log 2$。所以至少需要经过

$$\frac{I_e}{I_c} = \frac{\log n}{\log 2} = \log_2 n$$

次比较方能确定长度为 n 的数组中一个元素的位置。即该问题的信息论下界为 $\Omega(\log n)$。

二分查找法的效率恰好为 $\Omega(\log n)$，因此二分查找法是最优的基于比较的有序序列查找算法。

但如果有附加信息存在（例如给定数组的分布情况）使该元素的位置为 i 的概率 $p_i' > \frac{1}{n}$，则其信息量 $I_e' < \log n$。此时利用这些附加信息可能不需要 $\log_2 n$ 次比较即可找到它的位置。

6. 写出两种建堆方法的伪代码

算法 4 自顶向下建堆

输入: $A[1..n]$

输出: 重排 A，使其每个非叶子节点的值不小于其子节点的值

```
for i ← 2 to n do
    j ← i
    while j > 1 ∧ A[j] > A[⌊j/2⌋] do
        Swap(A[j], A[⌊j/2⌋])
        j ← ⌊j/2⌋
    end while
end for
```

算法 5 自底向上建堆

输入: $A[1..n]$

输出: 重排 A，使其每个非叶子节点的值不小于其子节点的值

```
for i ← ⌊n/2⌋ to 1 do
    j ← 2i
    while j ≤ n do
        if j + 1 ≤ n ∧ A[j + 1] > A[j] then
            j ← j + 1
        end if
        if A[j] > A[⌊j/2⌋] then
            Swap(A[j], A[⌊j/2⌋])
        else
            break
        end if
        j ← 2j
    end while
end for
```

7. 对玻璃瓶做强度测试。设地面高度为 0，从 0 往上有刻度 1 到 n，相邻两个刻度间距离都相等。如果一个玻璃瓶从刻度 i 落到地面没有破碎，而在高度 $i+1$ 处落到地面时碎了，则此类玻璃瓶的强度为 i。

   (a) 若只有一个样品供测试，如何得到该类玻璃瓶的强度

   > 从刻度 1 开始逐个高度进行测试，找到使其摔碎的最低高度。

   (b) 如果样品数量充足，如何用尽量少的测试次数得到强度

   > 使用二分法。首先测试高度 $\frac{n}{2}$，若摔碎了则测试 $\frac{n}{4}$，若没摔碎则测试 $\frac{3}{4}n$，依此类推。

   (c) 如果有两个样品，如何用尽量少的测试次数得到强度

   > 第一个瓶子测试高度 $k\sqrt{n}$（k 从 1 到 $\sqrt{n}$)，可以确定一个高度为 $\sqrt{n}$ 的区间。
   > 第二个瓶子在上述区间内从最低高度开始逐渐提升高度，最多测试 $\sqrt{n}$ 次。