《算法设计与分析》课程作业一

尹达恒

2020/10/14

1 快速排序

1.1 问题描述

Description

给定一维 `int` 型数组 `a[0,1,...,n-1]`，使用快速排序方法，对其进行从小到大排序，请输出递归过程中自顶自下第二层的划分结果，其中最顶层为第一层，即最终的排序结果层。

划分时请用第 1 个元素作为划分基准，并使用课件上的方法进行一次扫描实现划分。

Input

输入第 1 行有一个 `int` 型正整数 `m`(`m`<100)，表示有 `m` 行输入。每行输入的第一个数为 `int` 型正整数 `n`(8<`n`<1000)，后面接着输入 `n` 个 `int` 型整数。

Output

对每组数据, 输出自顶自下第二层的划分结果。

Sample Input

```
2
11 6 3 7 8 5 1 4 2 4 9 10
12 6 3 7 8 4 5 1 11 2 4 9 10
```

Sample Output

```
2 3 1 4 4 5 6 7 8 9 10
2 3 1 4 4 5 6 10 8 7 9 11
```

1.2 算法思路

1. 划分：以数组中的第一个元素为基准值，从数组中的第二个元素开始扫描，比基准值小的放右边，比基准值大的放左边。

2. 处理：

   - 初始时基准坐标 `p=` 数组开头位置；

Algorithm 1: 快速排序算法伪代码

Function *QuickSort(S, n)* **begin**

  Input: 未排序的数组 S、数组长度 n、递归层数 l

  Output: 排好序的数组 S

  if $(l \leq 2) \wedge (n > 1)$ **then**

    $p = 0$;

    for $i \in \{1, 2, 3, ..., n\}$ **do**

      if $S_0 > S_i$ **then**

        交换 S_{p+1} 和 S_i;

        p 自增 1;

      end

    end

    交换 S_0 和 S_p;

    QuickSort(S, p, $l + 1$);

    QuickSort($S' = \{S_i | i \in [p + 1, n - 1]\}$, $n - (p + 1)$, $l + 1$);

  end

  return

end

- `i` 从数组第二个元素开始遍历，若 `i` 位置的值大于基准值，则与 `p` 位置后一位的值交换，并令 `p` 自增；
- 最后令数组开头的值与 `p` 位置的值交换。

3. 递归：使用同样的方法递归地处理左边和右边的子数组。
4. 输出第二层结果：设置一个层标记 `l=1`，每次递归都将此值加 1 后作为参数传入，递归函数内若检测到 `l>2` 则退出并输出结果。

1.3 算法伪代码

见算法 1。

2 找第 2 大数

2.1 问题描述

Description

给定一维 `int` 型数组，请找到第 2 大的数。

Input

输入第 1 行有一个 `int` 型正整数 `m`(`m`<100)，表示有 m 行输入。

每行输入的第一个数为 int 型正整数 `n`(0<`n`<1000)，后面接着输入 `n` 个 `int` 型整数。

Output

输出 `m` 行，每行为找第 2 大数。

Sample Input

```
2
8 3 8 4 1 6 7 3 2
9 2 4 5 9 8 7 6 4 3
```

Sample Output

```
2
3
```

2.2 算法思路

由于题干并没有要求找任意第 k 大的数，因此本题可以直接使用线性查找，时间复杂度为 $O(n)$。

2.3 算法伪代码

见算法 2。

Algorithm 2: 找第二大数算法伪代码

```
Function FindNumber(S, n) begin
    Input: 数组 S、数组长度 n
    Output: 数组 S 中第二大的数
    R = { {S_0, S_1}(S_0 < S_1) ; {S_1, S_0}(S_0 > S_1) };
    for i ∈ {2, 3, ..., n} do
        if S_i < R_0 then
            R_1 = R_0;
            R_0 = S_i;
        end
        else
            R_1 = S_i;
        end
    end
    return R_1;
end
```

3 寻找两个正序数组的中位数

3.1 问题描述

Description

给定两个大小为 `m` 和 `n` 的正序（从小到大）数组 `nums1` 和 `nums2`。

请你找出并返回这两个正序数组的中位数。

进阶：你能设计一个时间复杂度为 $O(log(m+n))$ 的算法解决此问题吗？

Input

第一行输入 `nums` 表示有 `nums` 组测试

每组测试输入 `n` 和 `m`，分别表示数组 `nums1` 和 `nums2` 的长度

然后输入正序数组 `nums1`

接着输入正序数组 `nums2`

Output

对每组测试数据输出两个正序数组的中位数

Sample Input

```
2
2 1
1 3
2
2 2
1 2
3 4
```

Sample Output

```
2.00000
2.50000
```

3.2 算法思路

设要查找的两个正序数组分别为 $\mathbb{M}$ 和 $\mathbb{N}$，其组合成的正序数组为 $\mathbb{U}$。满足：

$$(\forall M_i \in \mathbb{M})(M_{i+1} \geq M_i) \wedge (\forall N_i \in \mathbb{N})(N_{i+1} \geq N_i) \wedge$$
$$\mathbb{U} = \{U_i | U_i \in \mathbb{M} \cup \mathbb{N}\} \wedge (\forall U_i \in \mathbb{U})(U_{i+1} \geq U_i) \wedge$$
$$k < |\mathbb{M}| \wedge k < |\mathbb{N}|$$

则有如下定理：

$$M_k < N_k \Rightarrow M_k < U_{2k}$$

证明：反证法。

$$\begin{aligned}
& M_k < N_k \wedge M_k \geq U_{2k} \\
\Leftrightarrow & U_{2k} \leq M_k < N_k \\
\Rightarrow & U_{2k} \in \{M_i \in \mathbb{M} | 1 \leq i \leq k\} \cup \{N_i \in \mathbb{N} | 1 \leq i \leq k-1\} \\
\Rightarrow & |\{U_i \in \mathbb{U} | 1 \leq i \leq 2k\}| < |\{M_i \in \mathbb{M} | 1 \leq i \leq k\} \cup \{N_i \in \mathbb{N} | 1 \leq i \leq k-1\}| \\
\Rightarrow & 2k \leq 2k-1 \Rightarrow false
\end{aligned}$$

$\therefore M_k < N_k \rightarrow M_k < U_{2k}$ 为真命题。

查找两个正序数组 $\mathbb{M}$ 和 $\mathbb{N}$ 的中位数的算法可以等价为一个查找其组合成的正序数组 $\mathbb{U}$ 中第 $2k$ 大数 U_{2k} 的算法：

1. 若两数组长度和为偶数，则 $2k = (|\mathbb{M}| + |\mathbb{N}|)/2$，查找 U_{2k} 和 U_{2k+1} 取平均值；

2. 若两数组长度和为奇数，则 $2k = (|\mathbb{M}| + |\mathbb{N}| + 1)/2$，查找 U_{2k}。

而根据前述定理 $M_k < N_k \Rightarrow M_k < U_{2k}$，若 $M_k < N_k$，则 U_{2k} 必然不在 $\{M_i \in \mathbb{M} | 1 \leq i \leq k\}$ 中，因此可以直接舍弃该部分，在 $\{M_i \in \mathbb{M} | k+1 \leq i \leq |\mathbb{M}|\}$ 和 $\mathbb{N}$ 中查找第 k 大的元素即可。显然，对于 $M_k > N_k$ 时类似的情况也成立。此过程可以递归进行，直到 $M_k = N_k$ 或某一轮的 $\mathbb{M}$ 或 $\mathbb{N}$ 为空。算法的时间复杂度为 $O(log|\mathbb{M}| + log|\mathbb{N}|)$。

3.3 算法伪代码

见算法 3和算法 4。

4 搜索二维矩阵

4.1 问题描述

Description

编写一个高效的算法来搜索 `m x n` 矩阵 `matrix` 中的一个目标值 `target`。该矩阵具有以下特性：

1. 每行的元素从左到右升序排列。

2. 每列的元素从上到下升序排列。

Input

第一行输入 `nums` 表示有 `nums` 组测试。
每组测试输入 `m`、`n`，`target`，分别表示矩阵的行列数以及目标值。
接下来输入 `m * n` 的二维矩阵。

Output

对每组测试数据输出能否在矩阵中找到 target。
若能找到，输出 `true`。
若找不到，输出 `false`。

Sample Input

```
1
5 5 5
1 4 7 11 15
2 5 8 12 19
3 6 9 16 22
10 13 14 17 24
18 21 23 26 30
```

Sample Output

```
true
```

提示

`m` $<= 1000$
`n` $<= 1000$

4.2 算法思路

4.3 算法伪代码

Algorithm 3: 在两个正序数组中找第 k 大数

Function *FindK*($\mathbb{M}$, $\mathbb{N}$, k) **begin**
    Input: 数组 $\mathbb{M}$ 和 $\mathbb{N}$、整数 k
    Output: 数组 $\mathbb{M}$ 和 $\mathbb{N}$ 中第 k 大的数
    if $\mathbb{M} = \emptyset$ **then return** N_k;
    if $\mathbb{N} = \emptyset$ **then return** M_k;
    if $k = 1$ **then return** $min(M_1, N_1)$;
    $d = \lfloor \frac{k}{2} \rfloor$;
    if $d > |\mathbb{M}|$ **then**
        if $M_{|\mathbb{M}|} \leq N_d$ **then**
            return $FindK(\emptyset, \mathbb{N}, k - |\mathbb{M}|)$;
        else
            return $FindK(\mathbb{M}, \{N_i \in \mathbb{N} | i > d\}, k - d)$;
        end
    end
    if $d > |\mathbb{N}|$ **then**
        if $N_{|\mathbb{N}|} \leq M_d$ **then**
            return $FindK(\mathbb{M}, \emptyset, k - |\mathbb{N}|)$;
        else
            return $FindK(\{M_i \in \mathbb{M} | i > d\}, \mathbb{N}, k - d)$;
        end
    end
    if $M_d = N_d$ **then**
        if k 为偶数 **then return** M_d(或 N_d);
        if k 为奇数 **then return** $min(M_{d+1}, N_{d+1})$;
    end
    if $M_d < N_d$ **then**
        return $FindK(\{M_i \in \mathbb{M} | i > d\}, \mathbb{N}, k - d)$;
    else
        return $FindK(\mathbb{M}, \{N_i \in \mathbb{N} | i > d\}, k - d)$;
    end
end

Algorithm 4: 找两个正序数组的中位数

Function *FindMedium*($\mathbb{M}$, $\mathbb{N}$) **begin**

  Input: 正序数组 $\mathbb{M}$、$\mathbb{N}$

  Output: 数组 $\mathbb{M}$ 和 $\mathbb{N}$ 的中位数

  if $|\mathbb{M}| + |\mathbb{N}|$ 为奇数 **then**

    if $\mathbb{M} = \emptyset$ **then return** $N_{\frac{|\mathbb{M}|+|\mathbb{N}|+1}{2}}$;

    if $\mathbb{N} = \emptyset$ **then return** $M_{\frac{|\mathbb{M}|+|\mathbb{N}|+1}{2}}$;

    return $FindK(\mathbb{M}, \mathbb{N}, \frac{|\mathbb{M}|+|\mathbb{N}|+1}{2})$;

  else

    if $\mathbb{M} = \emptyset$ **then return** $(N_{\frac{|\mathbb{M}|+|\mathbb{N}|}{2}} + N_{\frac{|\mathbb{M}|+|\mathbb{N}|}{2}+1})/2$;

    if $\mathbb{N} = \emptyset$ **then return** $(M_{\frac{|\mathbb{M}|+|\mathbb{N}|}{2}} + M_{\frac{|\mathbb{M}|+|\mathbb{N}|}{2}+1})/2$;

    return $(FindK(\mathbb{M}, \mathbb{N}, \frac{|\mathbb{M}|+|\mathbb{N}|}{2}) + FindK(\mathbb{M}, \mathbb{N}, \frac{|\mathbb{M}|+|\mathbb{N}|}{2} + 1))/2$;

  end

end