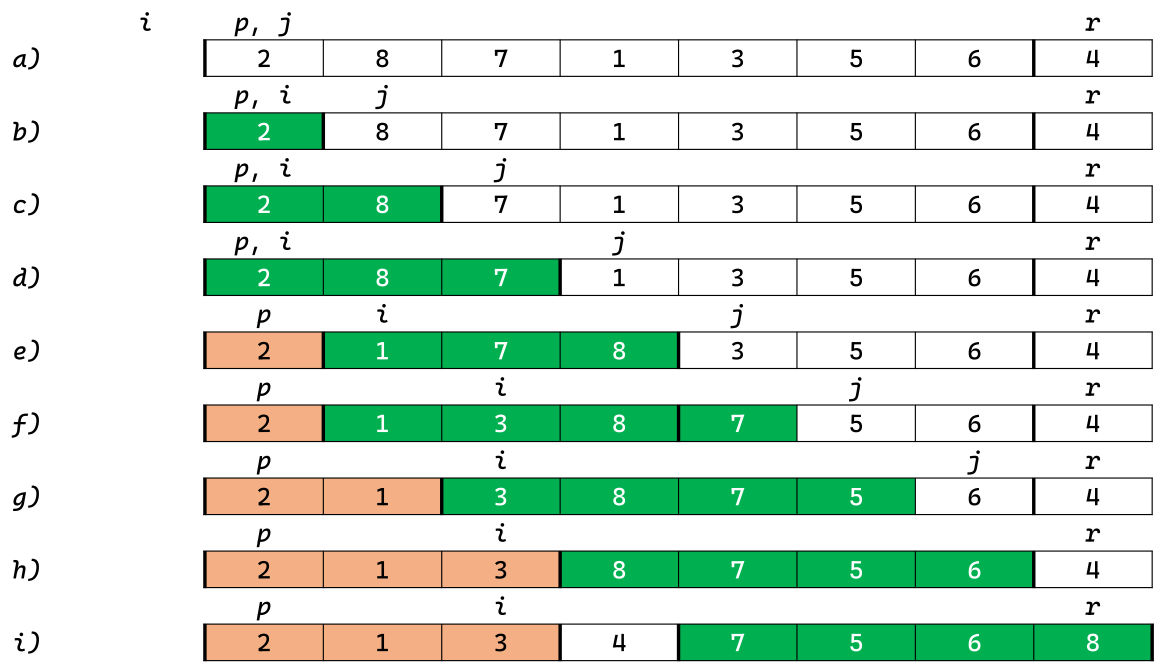
第三章-作业

1

以下是快速排序中的一种 $PARTITION$ 方法的伪代码及过程：

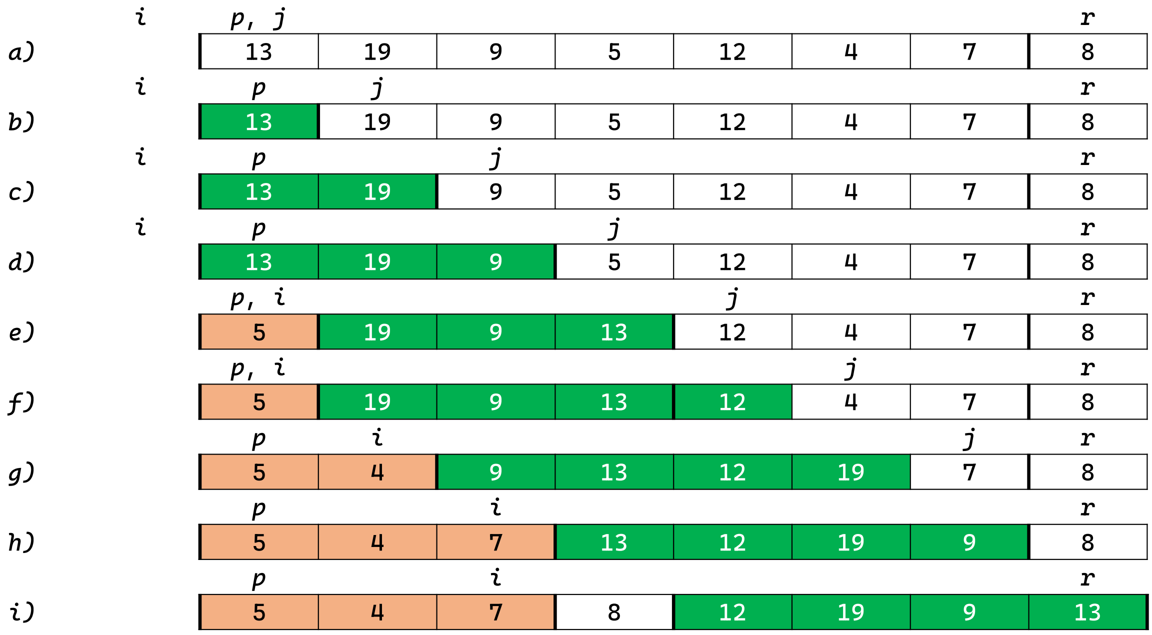
$PARTITION(A, p, r)$

```
x <- A[r]
i <- p-1
for j <- p to r-1
     do if A[j] <= x
           then i <- i+1
                exchange A[i] <-> A[j]
exchange A[i+1] <-> A[r]
return i+1
```



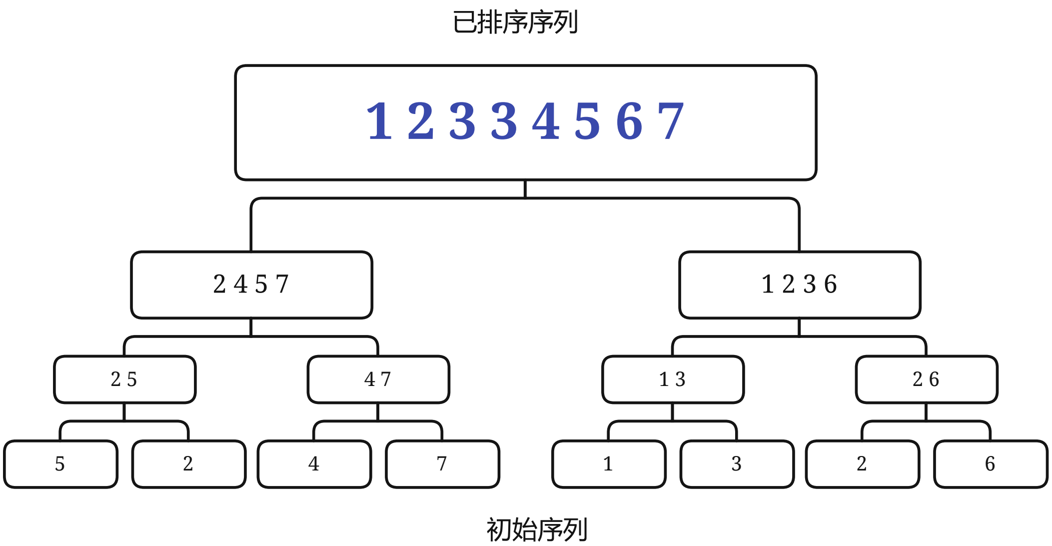
仿照上图说明 $PARTITION$ 过程作用于数组 $A =< 13, 19, 9, 5, 12, 4, 7, 8 >$ 的过程。

答：

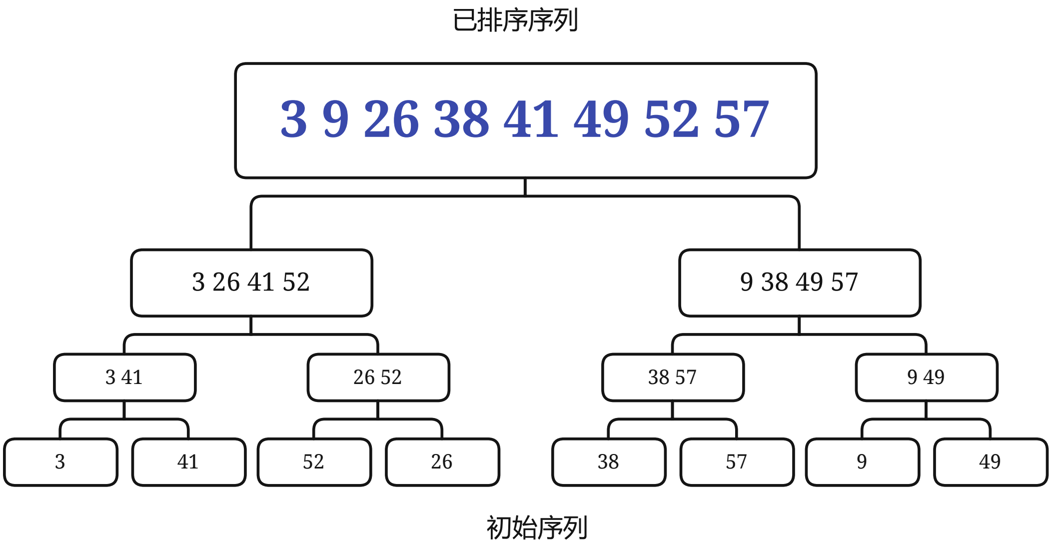


2

以下图为模型，说明合并排序在输入数组 $A =< 3, 41, 52, 26, 38, 57, 9, 49 >$ 上的执行过程。



答:



3

假设 A 和 B 是长度为 n 排好序的数组，且数组中每个数都是不同的。

3.1

设计一个算法，在 $O(\log n)$ 时间里找出这 $2n$ 个数的中位数，其中 $2n$ 个数的中位数为从小到大排序的第 n 个数。

答：

描述

假设序列 A、B 是升序序列。分别求两个升序序列 A、B 的中位数，设为 a 和 b。若 $a=b$，则 a 或 b 即为所求的中位数；否则，舍弃 a、b 中较小者所在序列之较小一半，同时舍弃较大者所在序列之较大一半，要求两次舍弃的元素个数相同。在保留的两个升序序列中，重复上述过程，直到两个序列中均只含一个元素时为止，则较小者即为所求的中位数。

伪代码

$MidSearch(A, B, n)$

```
firstA <- 0
lastA <- n - 1
firstB <- 0
lastB <- n - 1

while firstA != lastA || firstB != lastB

     midA <- firstA + (lastA - firstA)/2
     midB <- firstB + (lastB - firstB)/2

     if A[midA] == B[midB] then return A[midA]
        else if A[midA] < B[midB]
                if (lastA - firstA + 1) % 2
                   then lastB <- midB // 舍弃B中间点以后的部分且保留中间点
                        firstA <- midA // 舍弃A中间点以前的部分且保留中间点
```

```
                    else then lastB <- midB
                              firstA <- midA + 1
         else then
              if (lastB - firstB + 1) % 2
                 then lastA <- midA
                      firstB <- midB
                 else then lastA <- midA
                           firstB <- midB + 1

return min(A[firstA],B[firstB])
```

3.2

证明你的算法复杂度为 $O(\log n)$。

答:

- 分解：分解步骤仅仅计算并比较两个中位数，需要常量时间，因此，$D(n)=\Theta(1)$。
- 解决：我们递归地解决一个规模均为 $n/2$ 的子问题，将贡献 $T(n/2)$ 的运行时间。
- 合并：我们已经注意到这里不需要合并。

给出最坏情况运行时间 $T(n)$ 的递归式:

$$T(n)=\begin{cases}\Theta(1) & 若\ n=1\\ T(n/2)+\Theta(1) & 若\ n>1\end{cases}$$

使用主定理:

$$a=1,b=2,f(n)=\Theta(1),n^{log_b a}=n^{log_2 1}=1$$
$$f(n)=\Theta(1)$$

所以 $T(n)=f(n)\log n=\Theta(\log n)$，即 $T(n)=O(\log n)$

4

n 枚硬币，其中有一枚是假币，已知假币的重量较轻。现只有一个天平，要求用尽量少的比较次数找出这枚假币。我们用 $f(A,first,last)$ 函数来完成上述功能。请写出该函数的伪代码 (其中 A 表示硬币数组 $[1..n]$，$first$，$last$ 为当前考虑的硬币数组中的第一个和最后一个下标，函数返回值为假币的下标)。

答:

$f(A,first,last)$

```
n = last - first + 1 // 定义长度
if n == 1 then return first
if n == 2 then
   if A[first] > A[last] then return last
      else then return first
if n % 2 then // 奇数
   mid = (first + last)/2
   for i <- 1 to (n - 1)/2 do //排除中位数后，两边的数组长度为 (n - 1)/2
       sumleft <- sumleft + A[first - 1 + i] // first 对应 1，故从 first - 1 开始
       sumright <- sumright + A[mid + i] // mid + 1 对应 1，故从 mid 开始
   if sumleft == sumright then return mid
      else if sumleft > sumright then return f(A,mid + 1,last) //右边轻，假币在右边
```

```
      else then return f(A,first,mid - 1)
if !(n % 2) then // 偶数
   mid = (first + last - 1)/2
   for i <- 1 to n/2 do // 两边的数组长度为 n/2
       sumleft <- sumleft + A[first - 1 + i]
       sumright <- sumleft + A[mid + i]
   if sumleft > sumright then return f(A,mid + 1,last)
      else then return f(A,first,mid)
```

5

假设给定一个**不同整数**组成的**已经排好序**的数组 $A[1,\ldots,n]$，我们需要在该数组中查找是否存在索引 i，使得 $A[i]=i$。

5.1

尝试用描述分治算法来解决该问题。要求写出伪代码。

答:

描述

假设序列 A 是升序序列。计算中位数 mid 并进行 $A[mid]=mid$ 的判断，如果为真，则返回 $true$；如果 $A[mid]>mid$，**因为数组是由不同的整数组成的**，所以 mid 之后的所有元素都无法满足 $A[i]=i$，可以舍弃；如果 $A[mid]<mid$，同理，mid 之前的所有元素也无法满足 $A[i]=i$，可以舍弃。

eg：假设数组有相同元素，则:

下标	1	2	3	4	5	6	7	8	9
值	1	2	3	**4**	**6**	**6**	**6**	7	10

有相同的整数使得寻找中位数这一举动变得相对无意义，因为即使判断了 $A[mid]=mid$ 也不能以此把数组划分为有意义的子数组。你可以以该中位数为界将数组分成两个数组，重复上述过程，直到数组的长度为 1 为止。但这样做的时间复杂度和线性的查找没有区别，都是 $O(n)$。

不同的值使得情况大为不同。还是使用上面的例子，如果 $A[5]=6$，那么 $A[6]$ 必然不可能是 6，只能是大于等于 7 的数，在这样的情况下 $A[7]$ 绝对不可能为 7。这样的情况可以一直向后延伸，所以 mid 之后的所有元素都无法满足 $A[i]=i$。

伪代码

$FindEqual(A,first,last)$

```
n = last - first + 1
if n == 1 then return A[first] == first
if n % 2 then
   mid = (first + last)/2
   if A[mid] == mid then return true
      else if A[mid] > mid then FindEqual(A,first,mid - 1)
      else then FindEqual(A,mid + 1,last)
if !(n % 2) then
   mid = (first + last - 1)/2
   if A[mid] == mid then return true
      else if A[mid] > mid then FindEqual(A,first,mid - 1)
      else then FindEqual(A,mid + 1,last)
```

5.2

使用**主定理**估计第 1 小题中你所描述算法的复杂度。

注意：给出的算法应当保证在 $O(\lg n)$ 的运行时间内。

答：

假定原问题规模是 2 的幂，可以简化递归式。这时每个分解步骤将产生规模刚好为 $n/2$ 的两个子序列。

- 分解：分解步骤只有赋值和判断表达式几个步骤，需要常量时间，因此，$D(n)=\Theta(1)$。
- 解决：我们递归地解决一个规模近似为为 $n/2$ 的子问题，将贡献 $T(n/2)$ 的运行时间。
- 合并：我们已经注意到这里不需要合并。

给出最坏情况运行时间 $T(n)$ 的递归式：

$$T(n)=\begin{cases}\Theta(1) & 若\ n=1\\ T(n/2)+\Theta(1) & 若\ n>1\end{cases}$$

使用主定理：

$$a=1,b=2,f(n)=\Theta(1),n^{log_b a}=n^{log_2 1}=1$$
$$f(n)=\Theta(1)$$

所以 $T(n)=f(n)\lg n=O(\lg n)$。

附录

三道题的代码 c 语言版

[3.c](3.c) [4.c](4.c) [5.c](5.c)