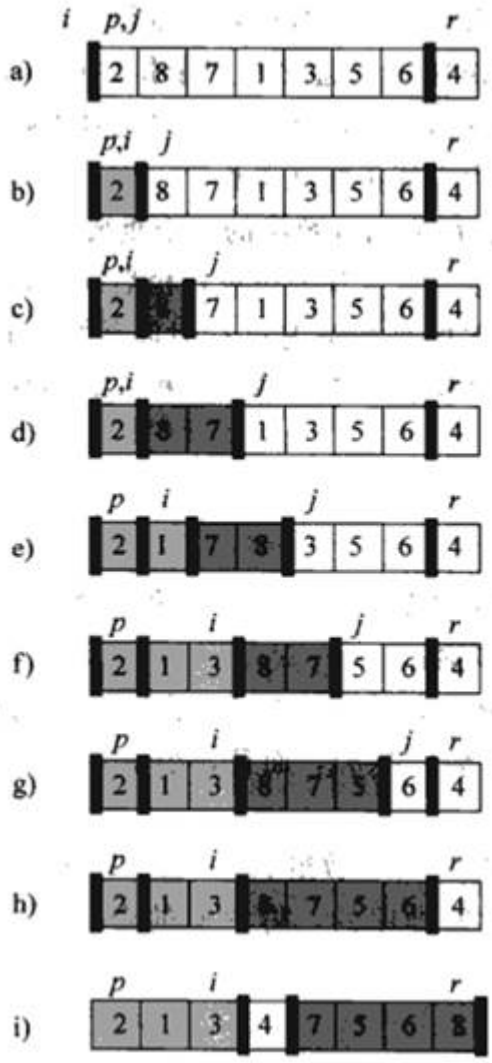
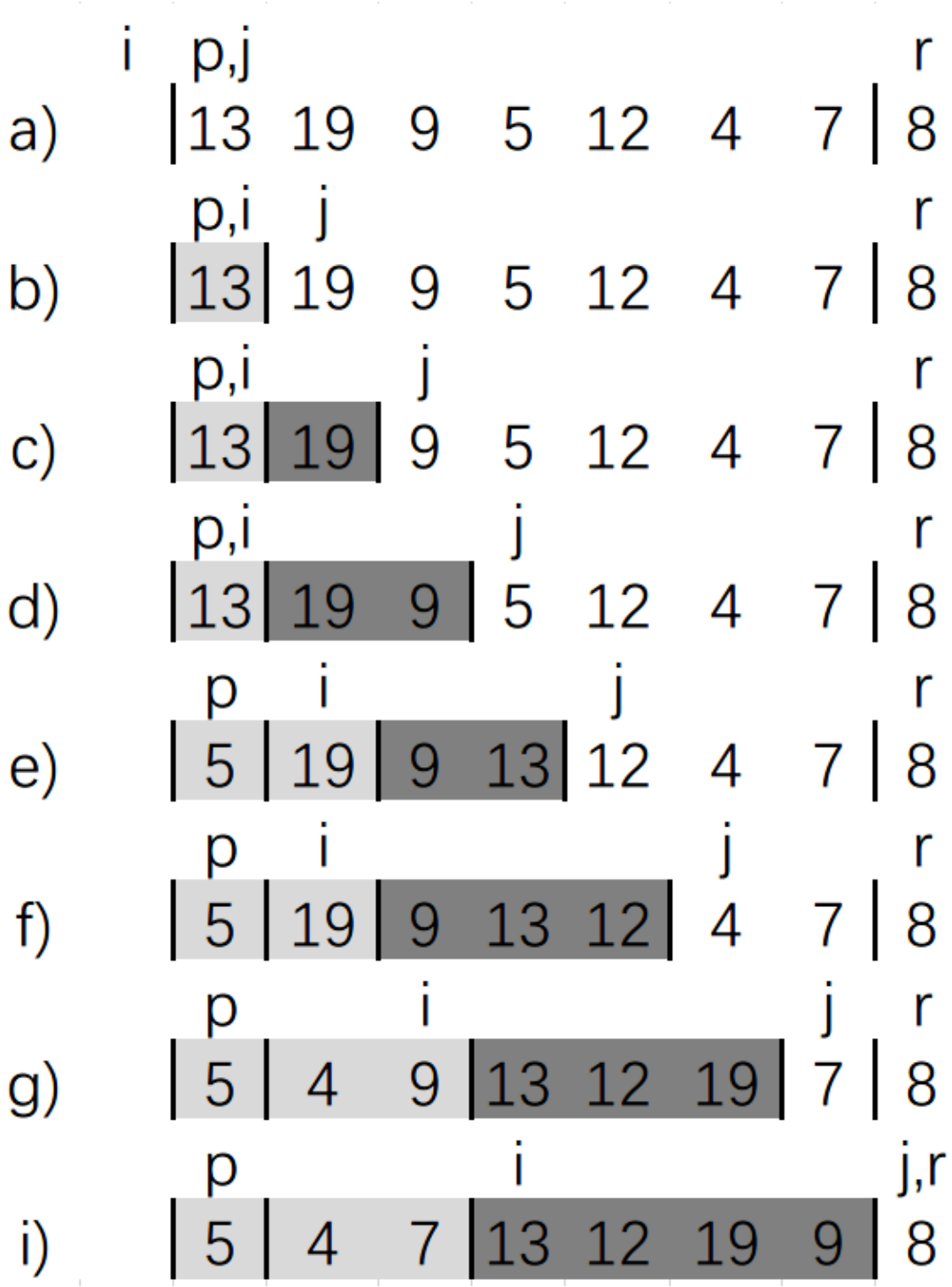
1. 以下是快速排序中的一种PARTITION方法的伪代码及过程：

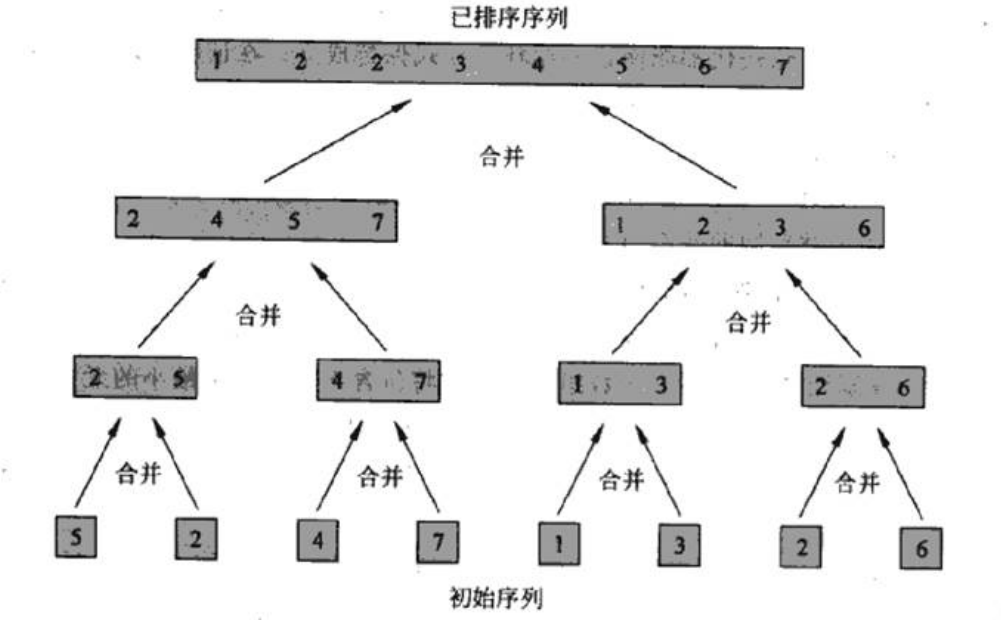
```
PARTITION(A, p, r)
1  x ← A[r]
2  i ← p−1
3  for j ← p to r−1
4        do if A[j]≤x
5              then i ← i+1
6                   exchange A[i]↔A[j]
7  exchange A[i+1]↔A[r]
8  return i+1
```

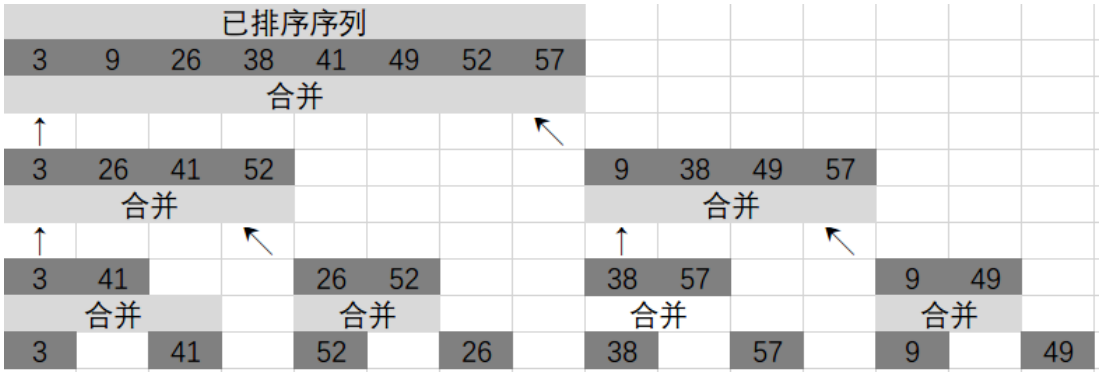


仿照上图说明PARTITION过程作用于数组$A =< 13, 19, 9, 5, 12, 4, 7, 8 >$的过程。


i p,j r
a) 13 19 9 5 12 4 7 8
p,i j r
b) 13 19 9 5 12 4 7 8
p,i j r
c) 13 19 9 5 12 4 7 8
p,i j r
d) 13 19 9 5 12 4 7 8
p i j r
e) 5 19 9 13 12 4 7 8
p i j r
f) 5 19 9 13 12 4 7 8
p i j r
g) 5 4 9 13 12 19 7 8
p i j,r
i) 5 4 7 13 12 19 9 8

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。





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。

    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。

        基本思想：在归并排序算法上优化。

        设 C 为 A、B 归并排序结果，容易知道所求中位数为 `(C[n] + C[n+1]) / 2`；而求 C[n]，可以先设 A 中取了 x 个数，那么 B 中取了 n - x 个数，`C[n] = MIN(A[x], B[n - x])`，然后可以使用二分法找到合适的 x 值。

```
FIND_N_IN_C(A, B, l_a, l_b, n)
//让 B 始终比 A 长
1. if l_a > l_b
2.       then return FIND_N_IN_C(B, A, l_b, l_a, n)
//处理空串
3. if l_a = 0
4.       then return B[0]
//递归边界条件
5. if n = 1
6.       then return min(A[0], B[0])
7. n_a ← min(⌊n/2⌋, l_a)
8. n_b ← n − n_a
9. if A[n_a − 1] < B[n_b − 1]
10.      then return FIND_N_IN_C(A[n_a ... n], B, l_a − n_a, l_b − n_b, n − p_a)
11.      else return FIND_N_IN_C(A, B[n_b ... n], l_a, l_b − n_b, n − p_b)

FIND_MEDIUM(A, B, n)
1. return (FIND_N_IN_C(A, B, n, n, n) + FIND_N_IN_C(A, B, n, n, n + 1))/2
```

2. 证明你的算法复杂度为$O(\log n)$。

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)。

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

（1）尝试用描述分治算法来解决该问题。要求写出伪代码。

（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）。