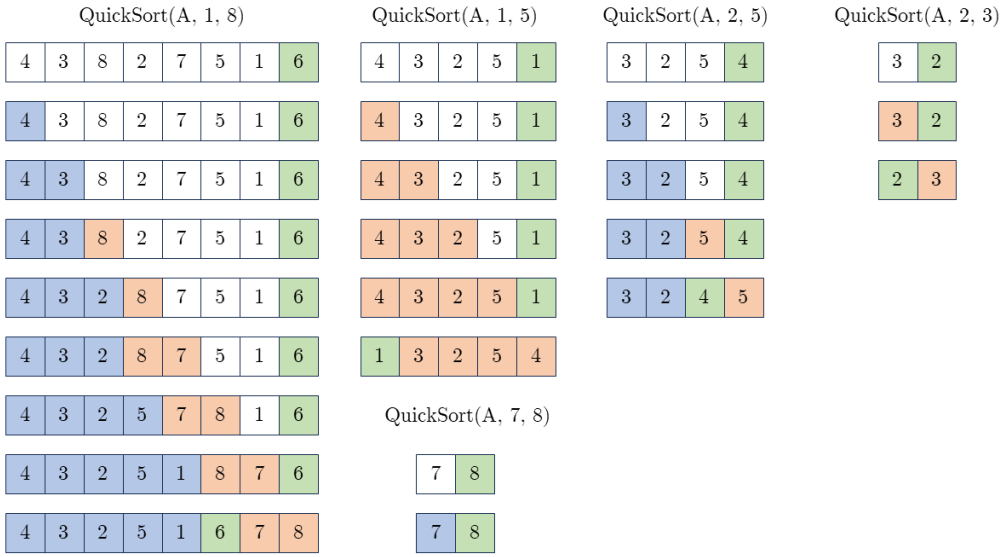
Data Structure and Algorithm Analysis(H)
Southern University of Science and Technology
Mengxuan Wu
12212006

Work Sheet 5

Mengxuan Wu

Question 5.1



Note: I omit all subarrays of size 0 or 1 in the figure.

Question 5.2

QUICKSORT(A, p, r)	
1.	**if** $p < r$ **then**
2.	$q =$ PARTITION(A, p, r)
3.	QUICKSORT$(A, p, q-1)$
4.	QUICKSORT$(A, q+1, r)$

To proof the correctness of the algorithm, we need to proof by induction.

Base case: Let $n = r - p + 1$ be the size of input. If $n = 1$, then the array is already sorted. So the base case holds.

Inductive case: Assume that the algorithm works for all input of size $n < c$. Now we need to proof that the algorithm works for input of size $n = c$.

Since PARTITION is correct, we know that the array is partitioned into two parts, $A[p \dots q-1]$ and $A[q+1 \dots r]$, where $A[p \dots q-1] \leq A[q] \leq A[q+1 \dots r]$. By the inductive hypothesis, we know that QUICKSORT$(A, p, q-1)$ and QUICKSORT$(A, q+1, r)$ will sort the two parts, since the size of the two parts are less than c. So the whole array will be sorted.

Question 5.3

When the array of distinct elements is sorted in decreasing order, the pivot will always be the smallest element in the array. Therefore, the partition will always give the pivot the index p. And the two subarray will always be $A[p \dots p-1]$ (which is an empty array) and $A[p+1 \dots r]$ (which excludes only the pivot). So the size of the two subarray are 0 and $n-1$.

Hence, the recurrence relation is $T(n) = T(0) + T(n-1) + \Theta(n)$, which is the same as that of array sorted in increasing order. Therefore, the runtime of QUICKSORT when the array contains distinct elements in decreasing order is $\Theta(n^2)$.

Question 5.4

When all n elements have the same value, the partition will always give the pivot the index r.

For such an input, the recurrence relation is $T(n) = T(n-1) + T(0) + \Theta(n)$, which is the same as that of array sorted in increasing order. Therefore, the runtime is $\Theta(n^2)$.

Question 5.5

PARTITION'(A, p, r)

```
1.  x = A[r]
2.  i = p − 1
3.  k = p − 1
4.  for j = p to r − 1 do
5.      if A[j] < x then
6.          i = i + 1
7.          k = k + 1
8.          exchange A[i] with A[j]
9.          if i ≠ k then
10.             exchange A[k] with A[j]
11.     else if A[j] = x then
12.         k = k + 1
13.         exchange A[k] with A[j]
14. exchange A[k + 1] with A[r]
15. return i + 1, k + 1
```

QUICKSORT'(A, p, r)

```
1.  if p < r then
2.      q1, q2 = PARTITION'(A, p, r)
3.      QUICKSORT'(A, p, q1 − 1)
4.      QUICKSORT'(A, q2 + 1, r)
```

The modified PARTITION' will run in $\Theta(n)$ time, since it runs through the array once and every operation in or out of the for loop takes constant time.

The runtime of QUICKSORT' on the input of question 5.4 is $\Theta(n)$, since the partition will be executed once and both of the two subarrays will be of size 0.