NO.

DATE

DSAA (H.) Lab 05.

Q5.1.

4 3 8 2 7 5 1 6

Quicksort (A, 1, 8)

i P,j r

⇒ |4 3 8 2 7 5 1 | 6

P,i j r

⇒ 4 | 3 8 2 7 5 1 | 6

P i j r

⇒ 4 3 | 8 2 7 5 1 | 6

P i j r

⇒ 4 3 | 8 | 2 7 5 1 | 6

P i j r

⇒ 4 3 2 | 8 | 7 5 1 | 6

P i j r

⇒ 4 3 2 | 8 7 | 5 1 | 6

P i j r

⇒ 4 3 2 5 | 7 8 | 1 | 6

P i r

⇒ 4 3 2 5 1 | 8 7 | 6

P i r

⇒ 4 3 2 5 1 | 6 | 7 8.

Quicksort (A , 1, 5).

i P,j r

⇒ |4 3 2 5 | 1

i P j r

⇒ |4 | 3 2 5 | 1

i P j r

⇒ |4 3 | 2 5 | 1

i P j r

⇒ |4 3 2 | 5 | 1

i P r

⇒ |4 3 2 5 | 1

NO.

DATE

```
   i  p                  r
⇒  | 1 | 3  2  5 | 4
Quicksort (A , 1, 0)
Quicksort (A , 1, 5)
      i  p,j             r
⇒     | 1  3  2  5 | 4
      p,i  j             r
⇒     1 | 3  2  5 | 4
      p  i   j           r
⇒     1  3 | 2  5 | 4
      p      i  j        r
⇒     1  3   2 | 5 | 4
         p      i        r
⇒     1  3   2 | 5 | 4
      p         i        r
⇒     1  3   2 | 5 | 4
         p      i        r
⇒     1  3   2 | 4 | 5.
Quicksort (A , 1, 3)
      i p,j              r
⇒     1  3 | 2
      p,i  j             r
⇒     1 | 3 | 2
      p,i                r
⇒     1 | 3 | 2
         p,i             r
⇒     1 | 2 | 3
Quicksort (A , 1, 1)
Quicksort (A , 2, 3]
      i p,j              r
⇒     2 | 3
         p,i             r
⇒     i | 3
```

```
   p,i  r
⇒  2 | 3
Quicksort (A , 2, 2]
Quicksort (A , 3, 3)
Quicksort (A , 4, 5]
      i p,j,             r
⇒     4 | 5
         p,i             r
⇒     4 | 5
         p,i             r
⇒     4 | 5
Quicksort (A , 4, 4)
Quicksort (A , 5, 5).  // A[4, ..., 5]: OK!
Quicksort (A , 6, 8]
      i p,j              r
⇒     6  7 | 8
      p,i  j             r
⇒     6 | 7 | 8
      p  i               r
⇒     6  7 | 8
      p  i               r
⇒     6  7 | 8
Quicksort (A , 6, 7)
      i  p,j             r
⇒     6 | 7
         p,i             r
⇒     6 | 7
         p,i             r
⇒     6 | 7
Quicksort (A , 8, 8] // A[6, ..., 8]: OK!
```

NO.

DATE

Q.5.2

Prof: Do induction on the length of the subarray that we need to sort in this Quicksort(A, p, r)
i.e. r-p.

(~~Ass~~ we can assume that $r \geq p$. otherwise the call of the algorithm is not right so Quicksort won't operate)

Denote $k = r-p$.

Base case: $k=0, 1$ Quicksort(A, p, r)

$\Rightarrow$ nothing to sort! the program terminates as $p \geq r$!

Induction Hypothesis: Assume it works for all: 1, 2, 3, ..., k.

Maintenance: For k+1:

Quicksort(A, p, r), $r-p = k+1$.

$\Rightarrow p < r$.

$q =$ Partition(A, p, r) $\Rightarrow$ $p \leq q \leq r$
(correct)

$\Rightarrow$ $\begin{cases} ① (q-1)-p \leq r-p-1 \leq k \\ ② r-(q+1) \leq r-p-1 \leq k \end{cases}$

$\Rightarrow$ Quicksort(A, p, q-1) and Quicksort(A, q+1, r) work!

$\Rightarrow$ Subarray $A[p, \cdots, q-1]$ and $A[q+1, \cdots, r]$ is sorted.

By the correctness of Partition(A, p, r):

we have $q =$ {greater than any element in $A[p, \cdots, q-1]$; smaller than any element in $A[q+1, \cdots, r]$}

$\Rightarrow$ $A[p, \cdots, r]$ is sorted! ✓. □.

3 2 1 | 5 6 7 8

3 2 1 | 1 4 3 2 5

1 2 3

Q5.3

In fact, this is one of the worst cases — the algorithm Partition always produces one subproblem with $n-1$ elements and one with 0 elements.

Indeed, the subarray Partition has to deal with could only be in one of the two following forms:

① $k \quad k-1, \ldots, a$. (a is the smallest element)

② $k-1, k-2, \ldots, a, k$.

The reason is as follows:

the first form is trivial. just like the original array.

Consider such an array:

$i \quad p,j \qquad\qquad r$

$k \quad k-1, \ldots\ldots \mid a$

$= i \quad p \quad j \qquad\qquad r$

$\Rightarrow \quad k \quad k-1 \quad k-2 \ldots \mid a$

$\Rightarrow \ldots\ldots$

$i \quad p \qquad\qquad j \quad r$

$\Rightarrow \quad k \quad k-1 \ldots\ldots a+1 \mid a$

$i \quad p \qquad\qquad\qquad r$

$\Rightarrow \quad k \quad k-1 \ldots\ldots a+1 \mid a$

$i \quad p \qquad\qquad\qquad r$

$\Rightarrow \quad a \mid k-1 \ldots\ldots a+1 \mid k.$

this will need us to consider the following 2 subarray:

- $A[1]$. : the trivial array $\{a\}$.
- $[k-1, k-2, \ldots, a+1, k]$. $\sim$ $A[2, \ldots, k-a]$
  $\triangleq [k-1, k-2, \ldots, a', k]$

this is actually the 2nd form we claimed before.

For this:

$i \quad p,j \qquad\qquad r$

$k-1, \quad k-2 \quad \ldots \quad a+1 \mid k$

$= \quad p, i \quad j$

$\Rightarrow \quad k-1 \mid k-2 \quad \ldots \quad a+1 \mid k$

p i j r

⇒ k-4 k-2 | k-3 ... a+1 | k

⇒ .....

p i r

⇒ k-1 k-2 k-3 ... a+1 | k

this will lead to:

Quicksort ([k-1, ..., a+1], 1, k-a-1) and Quicksort([?:], ?, ?)

trivial
0 elements

this is exactly in the form of ①!

Thus, informally: ① ⇒ ② → ① ⇒ ② ⇒ ...

⇒ There could only be 2 forms of array we need to sort!

And the above argument shows that Partition always produces one subproblem with n-1 elts. and one with 0 elts.

Thus, we have the following recurrence relation:

$$T(n) = T(n-1) + T(0) + \Theta(n)$$
$$= T(n-1) + \Theta(n)$$
$$= T(n-2) + \Theta(n) + \Theta(n)$$
$$= \cdots$$
$$= T(0) + \underbrace{\Theta(n) + \cdots + \Theta(n)}_{n \text{ times}}$$
$$= T(0) + \Theta(n^2)$$
$$= \Theta(n^2)$$

NO.

DATE

Q.5.4

When all n elts have the same value:
the judgement statement „$A[j] \le x$" will always be true.
⇒ for $j = p$ to $r-1$, „$i = i+1$" will always operate.
⇒ finally, $i = (p-1) + 1+1+\cdots+1 = r-1$.
⇒ q = Partition (A, p, r) = $i+1 = r$.
⇒ the pivot never moves its position!

$q = r$

In this case, the pivot never moves its position. $i+1 = r$.
⇒ the Partition algorithm always produces one subproblem with $n-1$ elts and one with 0 elts.
⇒ $T(n) = T(n-1) + T(0) + \Theta(n)$
Same as Q.5.3. $T(n) = \Theta(n^2)$.

Q.5.5

Algorithm: 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.                  k = k+1
7.                  exchange A[k] with A[j].
8.          else if A[j] < x then
9.                  i = i+1
10.                 k = k+1
11.                 exchange A[j] with A[k].
12.                 exchange A[k] with A[i].
13. exchange A[k+1] with A[r].
14. return i+1, k.
```

NO.

DATE

Algorithm: Quicksort (A, p, r).

1. if $p < r$, then
2. q, t = Partition (A, p, r).
3. Quicksort (A, p, $q-1$)
4. Quicksort (A, $t+1$, r)

If we still input an array with all values to be the same, then after the first Partition, $q = 1$, $t = r$.

⇒ Quicksort (A, 1, 0) and Quicksort (A, r+1, r) are trivial.

⇒ Asymptotically, the new Quicksort now has the same runtime as the new Partition in this case.

Now consider Partition (A, p, r).

The only difference (compared with the original one) is within the for loop, which will only change the constant value in front of $\Theta(n)$.

⇒ $T(n) = \Theta(n)$ still.

(In fact, $T(n) = T(0) + T(0) + \Theta(n)$
$= \Theta(n)$).