# Exercise 5 12310401 王子恒

5.1 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 |

| | | | | |
|---|---|---|---|---|
| | | | | r |
| 4 | 3 | 2 | 5 | 1 |

| | | | | |
|---|---|---|---|---|
| | | | | r |
| 4 | 3 | 2 | 5 | 1 |

| | | | | |
|---|---|---|---|---|
| | | | | r |
| 4 | 3 | 2 | 5 | 1 |

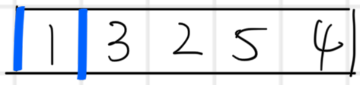
QUICKSORT(A, 7, 8)

| | | |
|---|---|---|
| i | p, j | r |
| | 7 | 8 |

| | |
|---|---|
| p, i | r |
| 7 | 8 |

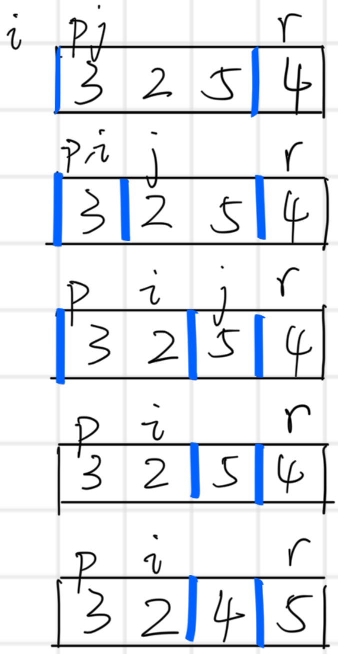
QUICKSORT(A, 7, 7)

QUICKSORT(A, 8, 8)



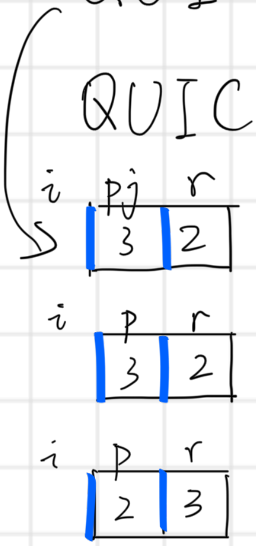
QUICKSORT(A,1,0)

QUICKSORT(A,2,5)



QUICKSORT(A,2,3)

QUICKSORT(A,5,5)



QUICKSORT(A,2,1)

QUICKSORT(A,3,3)

End

5.2

1. Base Case: If $p \geq r$, the subarray has 1 or 0 elements, it's trivially sorted.

2. Inductive Hypothesis: QuickSORT correctly sorts any array of size less than $n$.
3. Inductive Step: Consiter the subarray is $A[p \cdots r]$, $n = r-p+1$.

By the correctness of Partition, when Partition$(A, p, r)$ is called, it selects a pivot element and rearrange the array such that:
$A[p \cdots q-1] \leq A[q] < A[q+1 \cdots r]$, and $A[q]$ is placed in the correct space.
After that, it calls QUICKSORT$(A, p, q-1)$ and QUICKSORT$(A, q+1, r)$.
Because the inductive hypothesis, the two smaller array can be sorted correctly. Since $A[p \cdots q-1]$ and $A[q+1, r]$ can be sorted correctly, and $A[q]$ is sitting in the correct space, thus $A[p \cdots r]$ is sorted correctly.

5.3 Because it has distinct elements in decreasing order, thus every time the Partition would divide the subarrays with one has 1 element and the other one has $n-1$ elements
Besides, the runtime of PARTITION is $\Theta(n)$ for all the elements will be visit.

$$
\begin{aligned}
\text{Thus } T(n) &= T(n-1) + \Theta(n) \\
&= T(n-2) + \Theta(n) + \Theta(n-1) \\
&\vdots \\
&= \Theta(n) + \Theta(n-1) + \cdots + \Theta(1) \\
&= \Theta\left(\frac{n(n+1)}{2}\right) = \Theta(n^2).
\end{aligned}
$$

5.4 Because all elements are equall, the index "i" will increase until $j=r$ and $i=r-1$ thus the return $q=i+1=r$.

Because every time it left subarray with 1 and $n-1$ elements

$$T(n) = T(n-1) + \Theta(n).$$

$$\therefore T(n) = \Theta(n) + \Theta(n-1) + \cdots + \Theta(1) = \Theta\left(\frac{n(n+1)}{2}\right)$$
$$= \Theta(n^2).$$

5.5

```
PARTITION' (A, l, r):
  if (r ≤ l):
        return (l, r)
  q = l
  t = r
  p = A[l]
  i = l+1
  while i ≤ t:
        if A[i] < p then
            swap A[q] with A[i]   // from l to q is the
             q = q+1                 elements less than p
             i = i+1
        else if A[i] > p then:
             swap A[i] with A[t]  // from t to r is the
             t = t - 1               elements large than p
        else:
             i = i+1              // And A[q...t] equals p.
  return (q, t).
```

```
QUICKSORT'(A, l, r):
    if l < r:
        (q, t) = PARTITION'(A, l, r)
        QUICKSORT'(A, l, q-1)
        QUICKSORT'(A, t+1, r).
```

For the 5.4, the QUICKSORT only need to visit every elements once, so the new runtime will be $\Theta(n)$.