THEORY and PRACTICE

ARRAY : 40, 70, 20, 60, 90, 10, 50, 30.

Position : 0 1 2 3 4 5 6 7

① USING INSERTION SORT , $n = 8$

FOR $i = 1 \Rightarrow i < n$

$i = 1$, $j = i - 1 = 1 - 1 = 0$.

$X = a[i] = a[1] = 70$.

$a[j+1] = X \Rightarrow a[0+1] = 70$.

40 , 70, 20, 60, 90, 10, 50, 30.

$i = 2$, $j = i - 1 = 2 - 1 = 1$

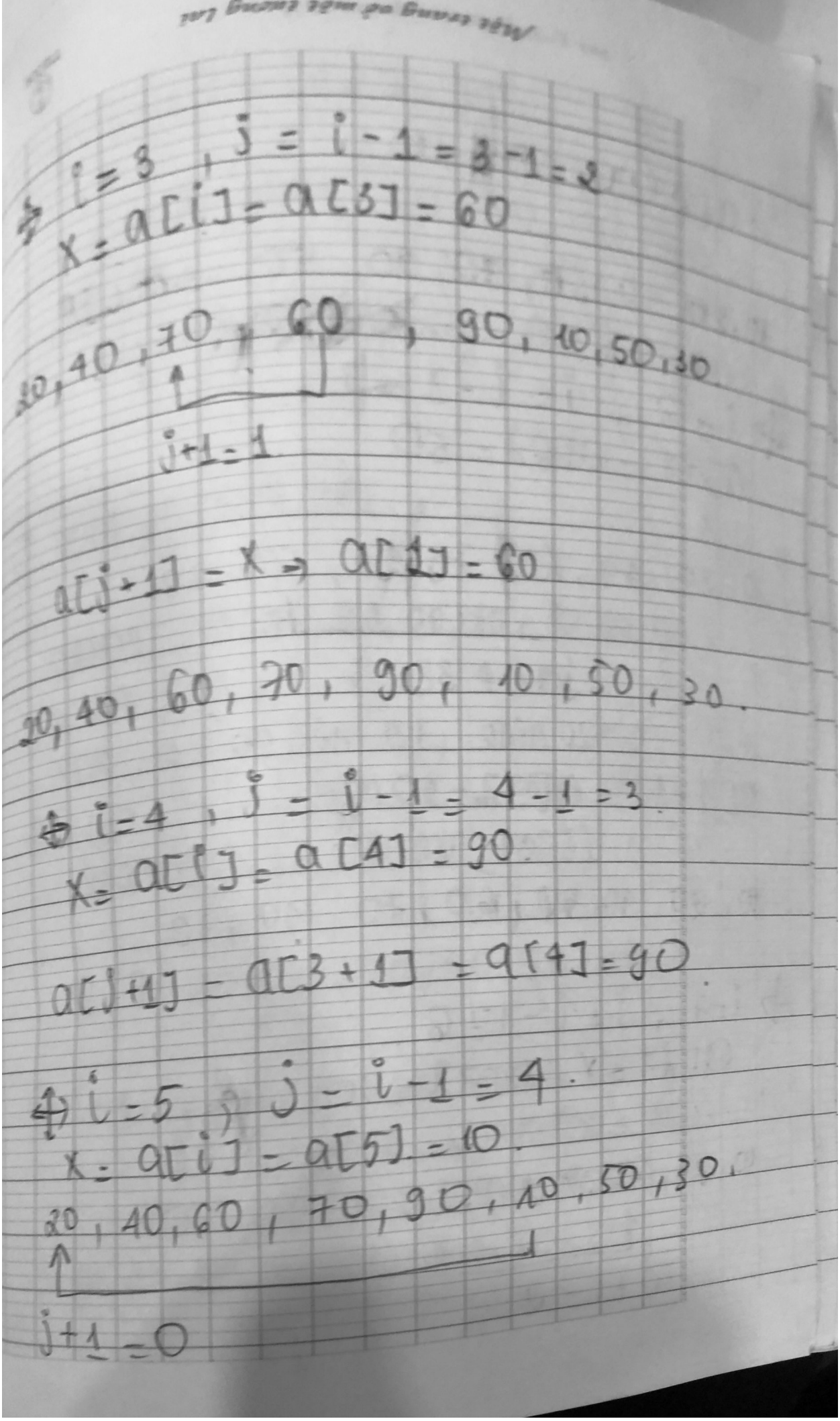
$X = a[i] = a[2] = 20$.

40 , 70, 20, 60, 90, 10, 50, 30.

$j + 1 = 0$

$a[j+1] = a[0] = 20$.

20, 40, 70, 60, 90, 10, 50, 30



$a[j+1] = a[0] = 10$

10, 20, 40, 60, 70, 90, 50, 30

↔ $i = 6$, $j = i - 1 = 5$

$a[i] = a[6] = 50$

10, 20, 40, 60, 70, 90, 50, 30

$j + 1 = 3$

$a[j+1] = a[3] = 50$

10, 20, 40, 50, 60, 70, 90, 30

↔ $i = 7$, $j = i - 1 = 6$

$a[i] = x = a[7] = 30$

10, 20, 40, 50, 60, 70, 90, 30

$j + 1 = 2$

a[j+1] = a[2] = 30

10, 20, 30, 40, 50, 60, 70, 90.

Algorithm Complexity $O(n^2) = O(64)$

② USING SELECTION SORT

ARRAY - 10, 70, 20, 60, 90, 10, 50, 30.

POSITION	0	1	2	3	4	5	6	7

FOR i=0 → i<n-1 | min_POS = i = 0.

↕ i=0 → j=i → n-1 → min_POS = 5 - 5

- SWAP (a[i], a[min_POS])

10, 70, 20, 60, 90, 40, 50, 30

↕ i=1 → min_POS = i = 1

j = i → n-1 → minPOS = 2.

SWAP (a[i], a[min_POS])

10, 20, 70, 60, 90, 40, 50, 30.

- i = 2, min_POS = i = 2
  j = i → n-1 → min_POS = 7
  SWAP(a[2], a[7])
  10, 20, 30, 60, 90, 40, 50, 70.
- i = 3, min_POS = i = 3
  j = i → n-1 → min_POS = 5
  SWAP(a[3], a[5])
  10, 20, 30, 40, 90, 60, 50, 70.
- i = 4, min_POS = i = 4
  j = i → n-1 → min_POS = 6
  SWAP(a[4], a[6])
  10, 20, 30, 40, 50, 60, 90, 70
- i = 5, min_POS = 5
  j = i → n-1 → min_POS = 5
  SWAP(a[5], a[5])
  10, 20, 30, 40, 50, 60, 90, 70
- i = 6, min_POS = 6
  j = i → n-1 → min_POS = 7
  SWAP(a[6], a[7])
  10, 20, 30, 40, 50, 60, 70, 90

Algorithm 6(64)

(5) Using Interchange sort

array: 40, 70, 20, 60, 30, 10, 50, 30.

FOR $i = 0 \rightarrow < n-1$

* $i = 0$, $j = i+1 = 1 \rightarrow < n$

↳ (a[i] > a[j]) ⟶ swap(a[i], a[j])

→ j = 2

20, 70, 40, 60, 90, 10, 50, 30.

→ j = 5

10, 70, 40, 60, 90, 20, 50, 30

* $i = 1$, $j = i+1 = 2 \rightarrow < n$

→ j = 2

10, 40, 70, 60, 90, 20, 50, 30.

→ j = 5

10, 20, 70, 60, 90, 40, 50, 30

* $i = 2$, $j = i+1 = 3 \rightarrow < n$

→ j = 3

10, 20, 60, 70, 90, 40, 50, 30.

→ j = 5

10, 20, 40, 70, 90, 60, 50, 30

→ j =

→ j = 7

10, 20, 30, 70, 90, 60, 50, 40

* i = 3 j = i + 1 = 4 → < n

→ j = 5

10, 20, 30, 60, 90, 70, 50, 40.

→ j = 6

10, 20, 30, 50, 90, 70, 60, 40

→ j = 7

10, 20, 30, 40, 90, 70, 60, 50.

* i = 4. j = i + 1 = 5 → < n.

→ j = 5

10, 20, 30, 40, 70, 90, 60, 50

→ j = 6

10, 20, 30, 40, 60, 90, 70, 50

→ j = 7.

10, 20, 30, 40, 50, 90, 70, 60

* i = 5 j = i + 1 = 6 → < n

→ j = 6

10, 20, 30, 40, 50, 70, 90, 60

→ j = 7

10, 20, 30, 40, 50, 60, 90, 70

* i = 6 , j = i + 1 = 7 → < n

→ j = 7

10, 20, 30, 40, 50, 60, 70, 90

Algorithm Complexity $O(n^2) = O(6^2) = O(6)$

④ USING BUBBLE SORT

ARRAY: 40, 70, 20, 60, 90, 10, 50, 30

FOR i = 0 → i < n − 1

i = 0 j = n − 1 → i

a[j − 1] > a[j] swap(a[j], a[j − 1])

10, 40, 70, 20, 60, 90, 30, 50

i = 1

10, 20, 40, 70, 30, 60, 90, 50

i = 2

10, 20, 30, 40, 70, 50, 60, 90

i = 3

10, 20, 30, 40, 50, 70, 60, 90

i = 4

10, 20, 30, 40, 50, 60, 70, 90

i = 5

10, 20, 30, 40, 50, 60, 70, 90

i = 6

10, 20, 30, 40, 50, 60, 70, 90

⑤ QUICK SORT

0 1 2 3 4 5 6 7
10 70 20 60 90 10 50 30

X = 60, LEFT = 0, RIGHT = 7

i j

10 30 20 60 90 10 50 70

i j

10 30 20 50 90 10 60 70

10 30 20 50 90 60 70

A B

A

LEFT = 0, RIGHT = 4, X = 20 = a[2]

i j

10 30 20 50 40

10 20 30 50 40

A_1 A_2

A1

LEFT = 0 j = RIGHT = 1, X = 10 = a[0]

10 20

A2:

i = LEFT = 2, RIGHT = 4, X = a[3] = 50

30 50 40

i j

[30 40][50]

B

X = a[6] = 60, i = 5, j = 7

90 60 70

i j

[60] [90 70]

B1 B2

B1 i = 6, j = 7, X = a[6] = 90

90 70

70 90

10 20 30 40 50 60 70 90

→ Result

Complexity O(8log8)

(7) When searching for 90 in array using sequential search → algorithm complexity : $O(n) = O(90)$

using binary search → algorithm complexity $O(\log n)$
$= O(\log 90)$

⇒ Binary search faster than sequential search