S2-Class-4[Sorting-2]

①

```
for(i=1;i<=3^n;i=i*3)
{
        print("*);
}
```

$\log_3 3^n = n \cdot \log_3 3$

$= n$

$O(n)$

②

```
for(i=1;i*i<=n;i++)
{
        print("*");
}
```

$\Rightarrow O(\sqrt{n})$

$i^2 \le n$

$\Updownarrow$

$i \le \sqrt{n}$

$\sqrt{n}$ beeter, compare to n

$\therefore O(\sqrt{n} \cdot \log_2^m)$

```
for(i=1;i<=Math.sqrt(n);i++)
{
        j=1;
        while(j<=m)
        {
                print("*");
                j=j*2;
        }
}
```

$\rightarrow \sqrt{n}$

$\Rightarrow \log_2^m$

```
j=1;
while(j<=m)
{
        print("*");
        j=j*2;
}
```

$\rightarrow \log_2^m$

$O(n^2)$

```
for(i=1;i<=n;i++)                    → n
{
        for(j=1;j<=n;j++)            → n
        {
                for(k=1;k<=100000000^1000000000^100000000;k=k*2)   ⇒ O(1)  (∵ const)
                {
                        print("*");
                }
        }
}
```

```
for(k=1;k<=100000000^1000000000^100000000;k=k*2)
{
        print("*");
}
```

In the second loop, the bound ~~100000000^1000000000^100000000~~ is written as n:

$\Rightarrow \log_2^n = \log_2 1000\cdots0$

$100\cdots0$ $100\cdots0$

very big value

constant

```
for(i=1;i<=n;i++)
{
        j=1
        while(j<=n)
        {
                Arr.sort();
                j=j*2;
        }
        print("*");
}
```

$\text{for}(i=1;i<=n;i++) \Rightarrow n$

$\text{Arr.sort();} \rightarrow n \cdot \log_2^n$

$\Rightarrow n * n \cdot (\log_2^n)^2$

$= \boxed{n^2 \cdot (\log_2^n)^2}$

```
while(j<=n)
{
        Arr.sort();
        j=j*2;
}
```

$\text{while}(j<=n) \rightarrow \log_2^n$

$\underbrace{n\log_2^n + n\cdot\log_2^n + n\cdot\log_2^n + \cdots + n\cdot\log_2^n}_{\log_2^n \text{ times}}$

$= \log_2^n * n \cdot \log_2^n = n\cdot(\log_2^n)^2$

let i, j be two indices of array, s.T $i < j$ and $(arr[i] > arr[j])$ for all i, j

Inversion :-

i

$n = 6$

	0	1	2	3	4	5
$a \to$	7	1	4	9	8	2

AILOBE

$i=0$	$i=1$	$i=2$	$i=3$	$i=4$	$i=5$
(7, 1)	(1, x)	(4, 2)	(9, 8)	(8, 2)	x
(7, 4)			(9, 2)		
(7, 2)					

① MAX inversions

$n=5$

	0	1	2	3	4
a	9	7	2	1	0

9 ⇒ 4 inv

7 ⇒ 3 inv

2 ⇒ 2 "

1 ⇒ 1

0 ⇒ x

$4+3+2+1=10$

$n-1$ Nat no's sum.

$1+2+\cdots+n-1$

$= \dfrac{n(n-1)}{2}$

$= \dfrac{5\times4}{2} = 10$ ✓

② min inversions

$n=5$

	0	1	2	3	4
a	0	1	2	7	9
	X	X	X	X	X

= 0 inv

Insertion Sort

n=1

n =7

Left	Right
sorted	Not sorted

a

0	1	2	3	4	5	6
2	9	1	7	4	8	6

a

0	1	2	3	4	5	6
9	2	1	7	4	8	6

key = ~~7~~ 4

a

0	1	2	3	4	5	6
2	9	1	7	4	8	6

a

0	1	2	3	4	5	6
1	2	7	~~7~~ 9		8	6

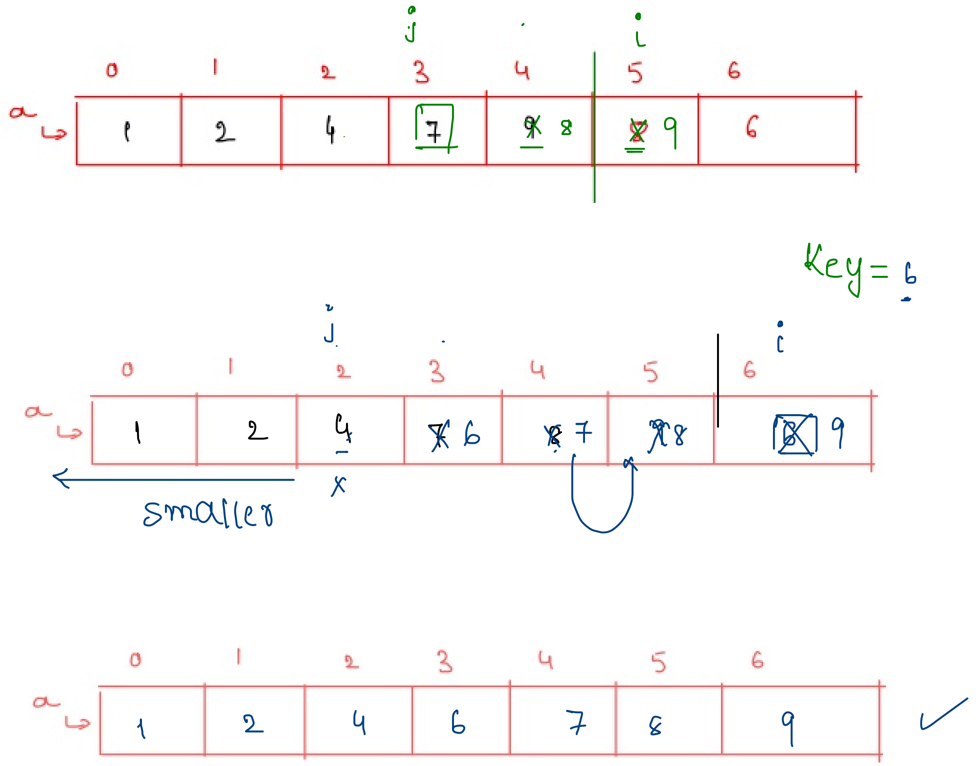
a

0	1 (j)	2	3	4	5 (i)	6
1	2	~~7~~ 4	~~9~~ 7	~~4~~ 9	8	6

a

0	1	2	3	4	5	6
1	2	4	7	9	8	6

```
for(i=1; i<n; i++)
{
    → key = arr[i] ✓
      j = i-1
                                  2 > 4
    x while( j >= 0 && arr[j] > key ) x
    {
        → arr[j+1] = arr[j]
          j--;
    }
    → arr[j+1] = key
```




```
for(i=1; i<n; i++)
{
    -> key = arr[i]
       j = i-1
                                  2 > 4
  -> x while( j >= 0 && arr[j] > key ) x
       {
           -> arr[j+1] = arr[j]
              j--;
       }
    -> arr[j+1] = key
}
```

```
void insertionSort(int arr[], int n)
{
        for(int i=1;i<n;i++)
        {
                int key=arr[i];
                int j=i-1;
                while(j>0 && arr[j]>key)
                {
                        arr[j+1]=key;
                        j=j-1;
                }
                arr[j+1]=key;
        }
}
```

$\underline{arr[j]>key}$ T

$\left.\begin{array}{l}arr[j+1]=key;\\ j=j-1;\end{array}\right\}$ moment

key = 8
7

a

0	1	2	3	4	5	6
10	9	8	7	6	5	4

WC:-

$n = 7$

a

0	1	2	3	4	5	6
4	5	6	7	8	9	10

	# of Comparisons		# of moments	
$i=1$	1	(1)	1	(1)
$i=2$	1 + 1	(2)	1 + 1	(2)
$i=3$	1 + 1 + 1	(3)	1 + 1 + 1	(3)
$i=4$		(4)		(4)
$i=5$		(5)		(5)
$i=6$		(6)		(6)

cmp's $= 1 + 2 + \cdots + n-1 \Rightarrow O(n^2)$

moments $= 1 + 2 + \cdots + n-1 \Rightarrow O(n^2)$

$O(n^2)$

```
void insertionSort(int arr[], int n)
{
        for(int i=1;i<n;i++)
        {
                int key=arr[i];
                int j=i-1;
                while(j>0 && arr[j]>key)
                {
                        arr[j+1]=key;
                        j=j-1;
                }
                arr[j+1]=key;
        }
}
```

F

x

a

0	1	2	3	4	5	6
4	5	6	7	8	9	10

n=7

a

0	1	2	3	4	5	6
4	5	6	7	8	9	10

j

	# of Comparisons	# of moments
i = 1	1	0
i = 2	1	0
i = 3	1	0
i = 4	1	0
i = 5	1	0
i = 6	1	0

comp's: 1+1+1+1+ ... +1 ⇒ O(n)

moments: 0 = 0

O(n)

Best case n=7

a →

0	1	2	3	4	5	6
4	5	6	7	8	9	10

\# of inversions (d) = 0   0 d

\# comparisons = $O(n)$

\# of moments = 0

$O(n)$ Best case

Worst case n=7

a →

0	1	2	3	4	5	6
10	9	8	7	6	5	4

\# of inversions (d) = 21 $\frac{n(n-1)}{2}$ d

\# comparisons = $O(n^2)$ ✓

\# of moments = $O(n^2)$ ✓

$O(n^2)$ W.C

can I say insertion sort Time Complexity is O(n+d) ?

where d : number of inversions in the array

Note:- * $O(n+d)$ d : # of inversions

B.C = 0

$n+0 = n$

$\therefore O(n)$

W.C : $\frac{n(n-1)}{2} \simeq n^2$

$n+n^2$

↳ n^2 (max)

$\therefore O(n^2)$

$\underline{BC}$ $\quad \overset{\underline{c}}{n} + \overset{\underline{m}}{0} \Rightarrow O(n) \quad \{ \quad d = 0$

$\underline{WC}$ $\quad \overset{c}{\underline{n^2}} + \overset{m}{\underline{n^2}} \Rightarrow \underline{O(n^2)} \quad \{ \quad d = n^2$

$\Rightarrow$ $\underline{TrC}$ $\quad O(n+d)$

$\underline{BC}$ $\quad d=0$

$O(n)$

$\overset{\underline{c}}{n} + \overset{\underline{m}}{\underline{n^2}}$

$\Downarrow$

$\underline{O(n^2)}$

$O(n+d)$

An array contains four occurrences of 0, five occurrences of 1, and three occurrences of 2 in any order. The array is to be sorted using swap operations (elements that are swapped need to be adjacent).

a. What is the minimum number of swaps needed to sort such an array in the worst case?
b. Give an ordering of elements in the above array so that the minimum number of swaps needed to sort the array is maximum.