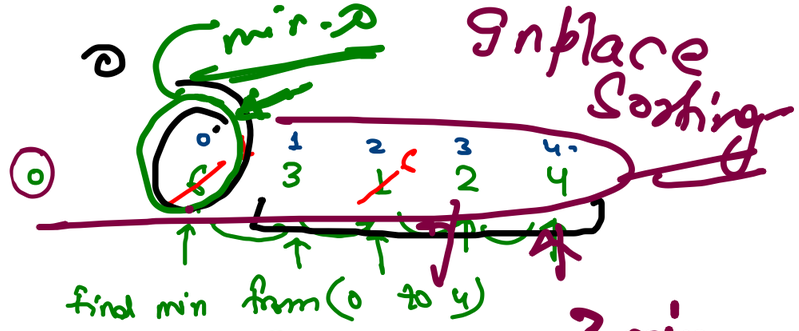
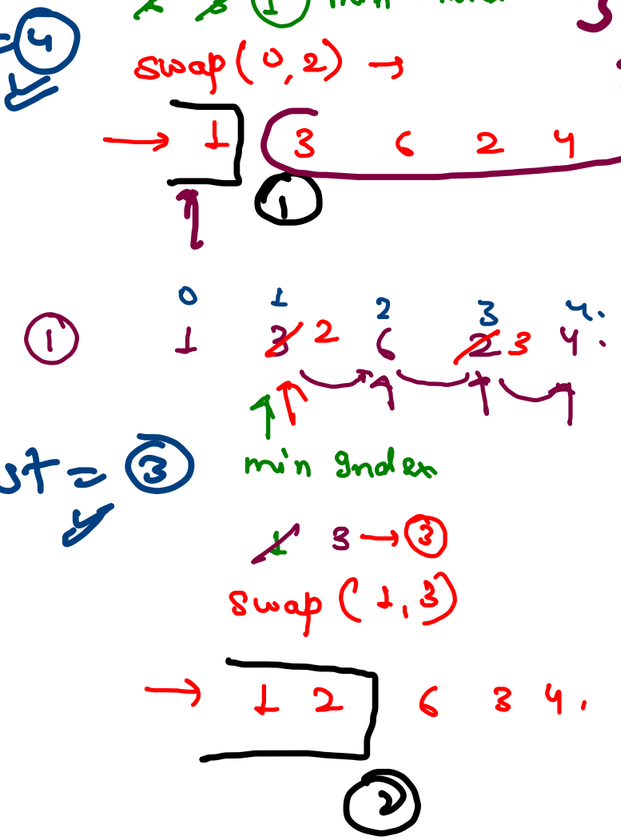
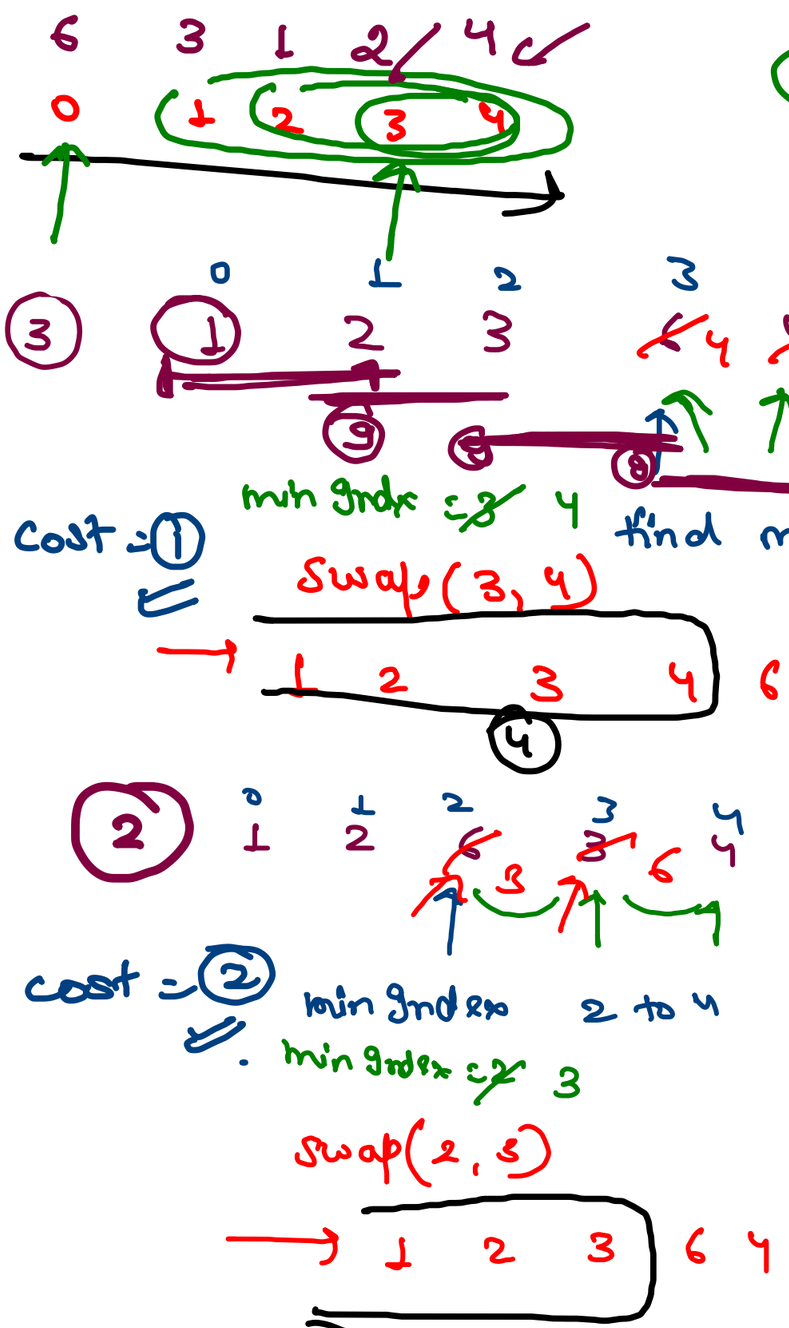
Selection Sort :

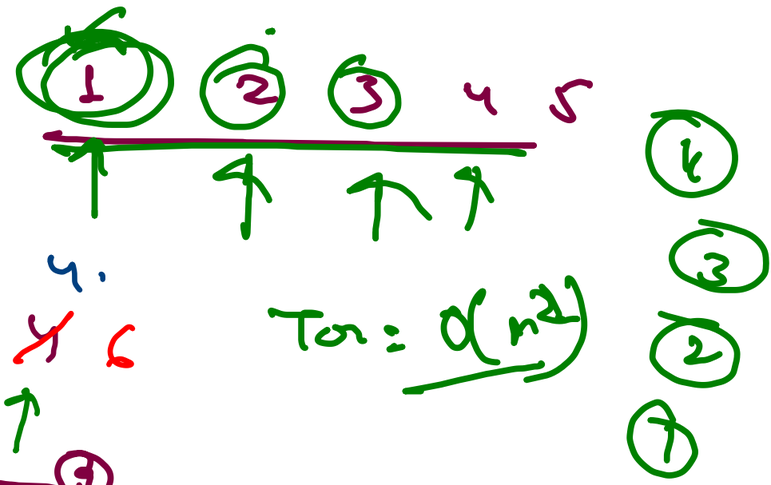
Eg - arr → 6 3 1 2 4

index → 0 1 2 3 4









Tc : $O(n^2)$

Inplace Sorting

Completn.

Space → O(1)

$T_{cost} = 1 + 2 + 3 + \dots + (n-1)$

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

$= \frac{n^2}{2} - \frac{n}{2}$

$T = O(n^2)$

Bonus Problem.

$res = f(x) = 1 \cdot x^n + 2 \cdot x^{n-1} + 3 \cdot x^{n-2} + \dots (n-1) * x^2 + n x^1$

xval = x.

Sum = 0

xval'

x * x * x * x * x,

```
// Bonus Question => 1.x^n + 2.x^(n-1) + 3.x^(n-2) + ---------- + n.X^1
public static void fun(int X, int N) {
    int xval = X;
    int sum = 0;
    for(int n = N; n >= 1; n--) {
        sum += xval * n;
        xval *= X;
    }
    System.out.println(sum);
}
```

$n x + (n-1) x^2 + (n-2) x^3 + \dots 3x^{n-2} + 2x^{n-1} + 1 \cdot x^n$

$n \rightarrow n \cdot x$

$n-1 \rightarrow (n-1) \cdot x^2$

$n-2 \rightarrow (n-2) x^3$

$\vdots$

$3 \rightarrow 3 \cdot x^{n-2}$

$2 \rightarrow 2 \cdot x^{n-1}$

$1 \rightarrow 1 \cdot x^n$

Sieve of Eratosthenes:

Range → 0 → 36

q = 10 ≡ ni

10 ≤ ni ≤ ni

q ≤ 10000

arr[i] = True, that means i is prime no.

array = new boolean[ni + 1]

Preprocessing

if 4 is not prime, then move on next element

n√n + O(q) = O(q)

O(1)

q >> n

0	1	2	3	4	5	6	7	8	9	10	11	12
F	F	T	T	F	T	F	T	F	F	F	T	F

13	14	15	16	17	18	19	20	21	22	23
T	F	F	F	T	F	T	F	F	F	T

24	25	26	27	28	29	30	31	32	33	34
F	F	F	F	F	T	F	T	F	F	F

35	36
F	F

query solve

arr[26]

O(26)

√n factor

ni = 36

Insertion Sort :

arr → 6 3 1 2 4
Index → 0 1 2 3 4.

itr: i - index that means (0 to i-1) → sorted.

j

① 1.1 6 3 1 2 4.
0 1 2 3 4.

3 6 1 2 4.

→ 3 6 1 2 4.

② 2.1 3 6 1 2 4.
0 1 2 3 4

2.2 3 1 6 2 4.

1 3 6 2 4

③ 3.1 1 3 6 2 4.
0 1 2 3 4.

3.2 1 3 2 6 4

3.3 1 2 3 6 4

1 2 3 6 4

④ 4.1 1 2 3 6 4
0 1 2 3 4.

4.2 1 2 3 4 6

if previous is not smaller stop the algorithm for that particular iteration

1 2 3 4 6

Benefit

Sorted array -

j = 1 itr; j > 0; j--

1 2 3 4 5 6

→ O(n) sort

O(n²) ✗

Bubble Sort + Selection Sort O(n²)

Count Sort

number → Range 0 lo 65 to hi 75 (10)

total no. of elements >>> hi − lo

O(n) (m)

Range → 0, 9

arr → 74 70 71 73 73 67 72 72 65 69 68 69 69 66 66

index 0 1 2 3 4 5 6 7 8 9 10 11 12 13 14

lo 1

hi 5.

array → (hi − lo + 1)

(4)

(5 − 1 + 1)

67 − 65 = 2 O(n)

frequency array.

0 (65)	1 (66)	2 (67)	3 (68)	4 (69)	5 (70)	6 (71)	7 (72)	8 (73)	9 (74)	(75)
1	1	2	2	1	1	2.	1	1	3	

i = 0

Print order

O(n) Base 265 min value

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

66

65

(1)

max complexity — O(n) → Sort

Step (1) → min & max find, max & min include,

(2) → array of frequency having size (max − min + 1)

(3) → manage frequency.

arr → 61 62 62 63 65 69 73 71 72 73 66 67 66

0 1 2 3 4 5 6 7 8 9 10 11 12

① find min & max

min → 61 , max → 73

② Make freq. array size (max - min + 1) (73 - 61 + 1) = 13

61 62 63 64 65 66 67 68 69 70 71 72 73 74.

0 1 2 3 4 5 6 7 8 9 10 11 12 13

1	2	1	0	1	2	1	0	1	0	1	1	1	0

61+0 61+1 61+2 61+4 61+5 61+6 61+8 61+10 61+11 61+12

freq → min=61

Sort

61→0

61 62 62 63 65 66 66 67 69 71 72 73

index = arr[i] - min

61 - 61 = 0

62 - 61 = 1

62 - 61 = 1

63 - 61 = 2

65 - 61 = 4

66 - 61 = 5

66 - 61 = 5

67 - 61 = 6

69 - 61 = 8

71 - 61 = 10

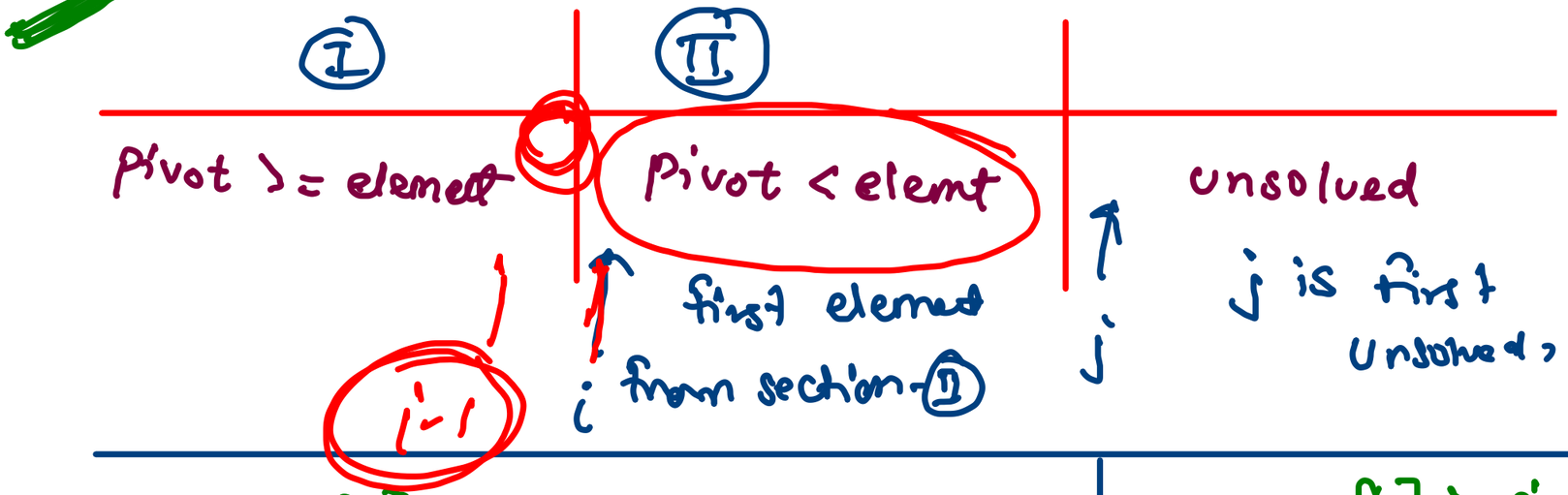
72 - 61 = 11

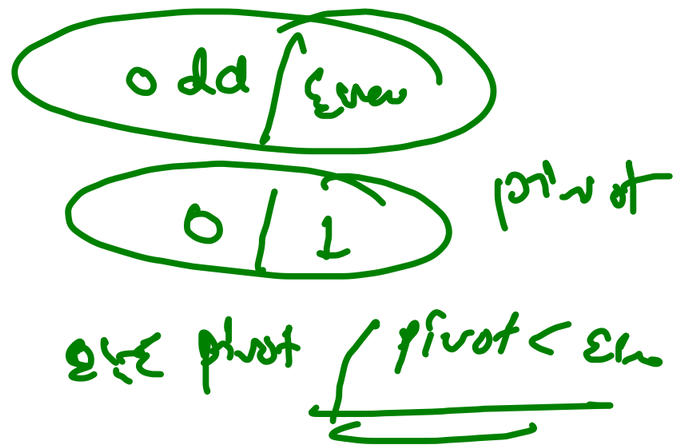
73 - 61 = 12

Partition on Array :

pivot = 12.

arr →	19	11	6	15	14	9	4	16	18	3	2	17
index →	0	1	2	3	4	5	6	7	8	9	10	12 (pivot)





arr[j] <= pivot	arr[j] > pivot

swap(i, j)
i++;
j++;
	j++;

Kth Smallest

k = 3rd small

k = 4

```java
public static int PartitionIndex(int[] arr, int pivot, int lo, int hi) {
    int i = lo;
    int j = lo;

    while(j <= hi) {
        if(arr[j] <= pivot) {
            swap(arr, i, j);
            i++;
            j++;
        } else {
            j++;
        }
    }
    return i - 1;
}

// kth Smallest
public int quickSelect(int[] arr, int lo, int hi, int k) {
    int pivot = arr[hi];
    int pi = PartitionIndex(arr, pivot, lo, hi);

    if(pi < k) {
        return quickSelect(arr, pi + 1, hi, k);
    } else if(pi > k) {
        return quickSelect(arr, lo, pi - 1, k);
    } else {
        return pivot; // arr[pi]
    }
}
```

k = 4

10 = 4

1