PAGE NO. icon
DATE : / /

Bubble Sort

Increasing

0	1	2	3	4
88	66	55	44	22

→ worst case scenario

o/p : 22, 44, 55, 66, 88

Aim :- (4) elements → if they are at correct positions then 5th element automatically would be

4

0 → ≤ 3

counter = 0

↓0	↓1	↓2	↓3	↓4
88	66	55	44	22

j, j+1

arr[j] > arr[j+1]

j=0	66	88	55	44	22
j=1	66	55	88	44	22
j=2	66	55	44	88	22
j=3	66	55	44	22	(88) → 88 reached its correct position

j = 0 → < 4

counter = 1

	66	55	44	22	88
j=0	55	66	44	22	88
j=1	55	44	66	22	88
j=2	55	44	22	(66)	(88) → correct position

j = 0 → 2

= 0 → < 3

counter = 2

	55	44	22	66	88
j=0	44	55	22	66	88
j=1	44	22	(55)	(66)	(88) → correct position
~~j=2~~					

j = 0 → < 2

PAGE NO.
DATE : / /

Counter = 3

j=0 (22) (44) (55) (66) (88) → all are in correct pos^n

44 22 55 66 88

Dim Tha

Code :

```
for (int counter = 0; counter < arr.length-1; counter++){
    for (int j=0; j< arr.length-1-counter; j++){
        if (arr[j] > arr[j+1]) {
            int temp = arr[j];
            arr[j] = arr[j+1];
            arr[j+1] = temp;
        }
    }
}
```

Selection Sort

0	1	2	3	4
88	11	44	99	55

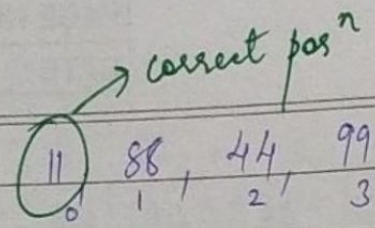
Counter = 0

min = 0
arr[min], arr[1] → min = 1
arr[min], arr[2] → min = 1
arr[min], arr[3] → min = 1
arr[min], arr[4] → min = 1

---

arr[min] ↔ arr[counter]
arr[1] ↔ arr[0]

PAGE NO. icon
DATE : / /



counter = 1

min = 1

arr[min], arr[2] → min = 2

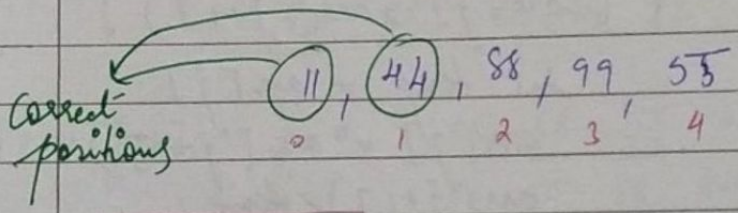
arr[min], arr[3] → min = 2

arr[min], arr[44] → min = 2

---

arr[min] ↔ arr[counter]

arr[2] ↔ arr[1]



counter = 2

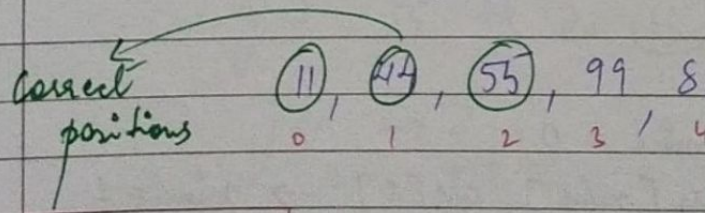
min = 2

arr[min], arr[3] → min = 2

arr[min], arr[4] → min = 4

arr[min] ↔ arr[counter]

arr[4] ↔ arr[2]



counter = 3

min = 3

arr[min], arr[4] → min = 4

arr[min] ↔ arr[counter]

arr[4] ↔ arr[3]

11, 44, 55, 88, 99 → SORTED

PAGE NO.
DATE : / /

Code :

```
for( int counter = 0; counter < arr.length -1 ; counter ++){
    int min = counter ;
    for (int j = counter + 1 ; j <= arr.length -1; j++){
        if (arr[j] < arr [min ]) {
            min = j ;
        }
    }
    int temp = arr [ min ];
    arr [ min ] = arr [ counter ];
    arr[ counter ] = temp ;
}
```

Insertion Sort

88, 11, 44, 99, 33

Counter → array till counter idx will be sorted

counter = 1

88, 11 | 44, 99, 33

Counter = 0 ——→ Not required

88 | , 11, 44, 99, 33

first element will always be sorted

```
for(int counter = 1; counter <= arr.length - 1; counter++) {
    int val = arr[counter];
    int j = counter - 1;
    while(j>=0 && arr[j] > val) {
        arr[j+1] = arr[j];
        j--;
    }
    arr[j+1] = val;
}
```

Merge Sort

merge 2 sorted arrays

arr 1 : [10, 20, 30, 40, 50] m

arr 2 : [5, 15, 30, 45, 60, 70, 80] n

merged : [5, 10, 15, 20, 30, 30, 40, 45, 50, 60, 70, 80]

i exhausted

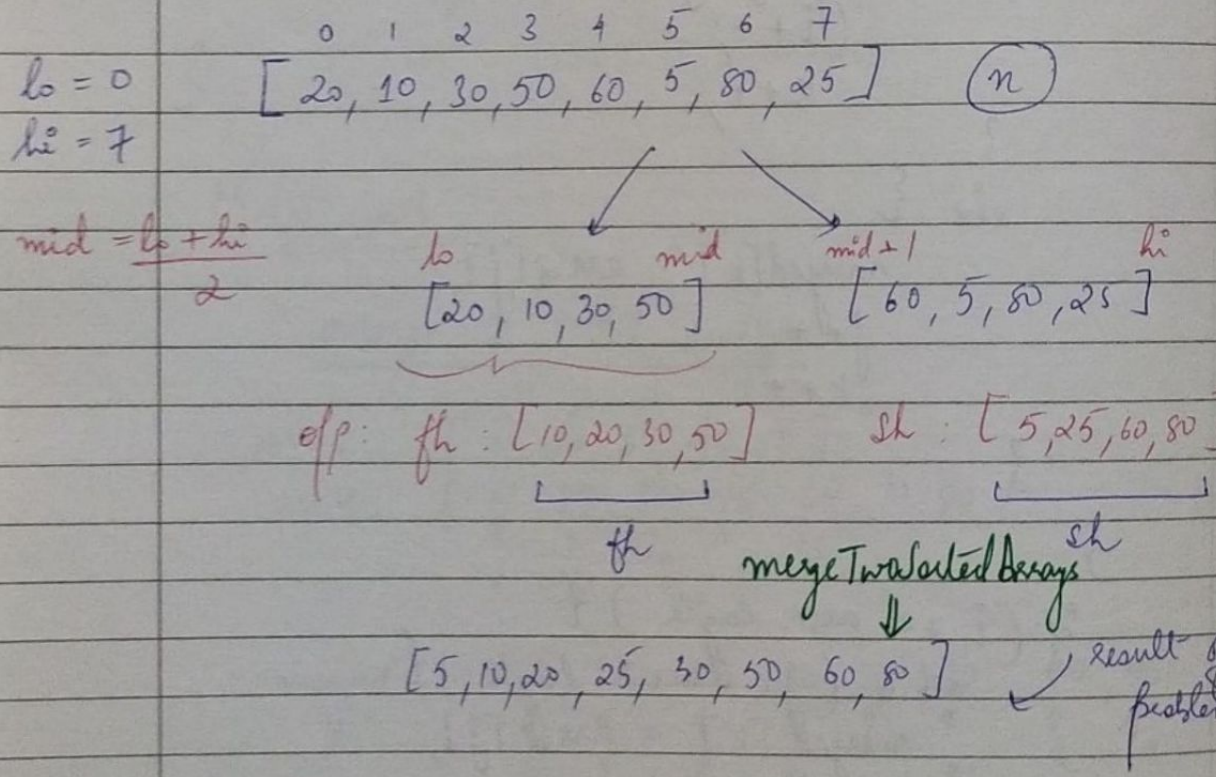
i	arr1	~~5~~	~~10~~	~~20~~	~~20~~	~~30~~	~~30~~	~~40~~	50	~~50~~	Add arr2 ke baache hue elem
j	arr2	10	15	~~15~~	30	30	~~40~~	45	~~45~~	60	

PAGE NO. icon
DATE : / /

```
public static int[] mergeTwoSortedArrays(int[] arr1, int[] arr2){
    int[] merged = new int[arr1.length + arr2.length];
    int i=0;
    int j=0;
    int k=0;

    while(i < arr1.length && j< arr2.length){
        if(arr1[i] <= arr2[j]){
            merged[k] = arr1[i];
            i++;
            k++;
        }
        else{
            merged[k] = arr2[j];
            j++;
            k++;
        }
    }
    if(i == arr1.length){
        while(j< arr2.length){
            merged[k] = arr2[j];
            j++;
            k++;
        }
    }
```

```
if (j == arr2.length) {
    while (i < arr1.length) {
        merged[k] = arr1[i];
        i++;
        k++;
    }
}
return merged;
}
```




```
public static int[] mergeSort(int[] arr, int lo, int hi) {
    BC
    int mid = (lo + hi)/2;
    int[] fh = mergeSort(arr, lo, mid);
    int[] sh = mergeSort(arr, mid+1, hi);

    int[] merge = mergeTwoSortedArrays(fh, sh);
    return merge;
}
```

br - base result

PAGE NO. icon
DATE : / /

BC

```
if( lo == hi ){
    int[] br = new int[1];
    br[0] = arr[lo];      // br[0] = arr[hi];
    return br;            ek hi baat hai
}
```

* merge sort me jo changes ho rahe hain woh original array me nhi ho rhe → naye array me bante hai

Original array me change → Inplace → Bubble, Selection, insert

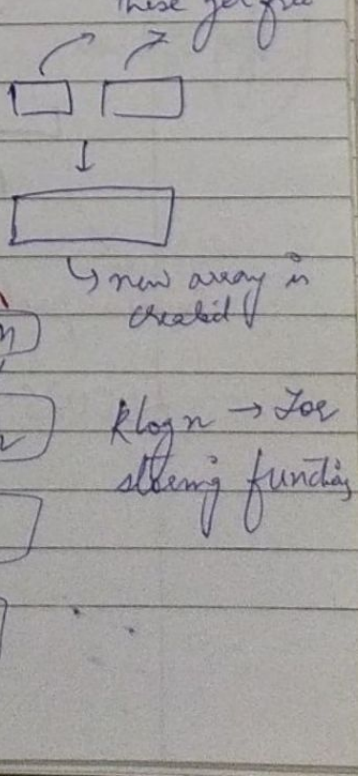
Not Inplace → merge sort

Divide & Conquer algo.

Complexity Analysis :-

Time Complexity : O(n log n)

Space Complexity : O(n)



Uss time pe ave koi array nhi hoga

: O(n)

Quick Sort

PAGE NO. icon
DATE : / /

Divide & Conquer

0	1	2	3	4	5	6	7	8
40	20	90	30	70	10	80	60	50
↑ l								↑ r

1. Partitioning
   lo = 0
   hi = 8
   mid = 4
   pivot = arr[mid]
   = 70

a) all the elements smaller than pivot would be on the left side
b) all the elements greater than pivot would be on the right side

0	1	2	3	4	5	6	7	8
40	20	~~90~~ 50	30	~~70~~ 60	10	80	~~60~~ 70	~~50~~ 90
↑ l	↑ l	↑ l	↑ l	↑ l	↑ l	↑ r	↑ r	↑ r

30 < 70

40 < 70 ✓ 20 < 70 ✓ 90 < 70 X 70 < 70 equality

problem! problem!

SWAP SWAP

[40, 20, 50, 30, 60, 10 | 80, 70, 90]

(pivot can be on any side =)

< 70 >= 70

10, 20, 30, 40, 50, 60 70, 80, 90

Smaller problem

RECURSION !!!

O(n log n) - Avg case running time

O(n²) - Worst case running time

In-place

PAGE NO.
DATE : / /

```
public static void quickSort (int[] arr, int lo, int hi){
    BC
    // 1. partitioning
    int mid = (lo + hi)/2;
    int pivot = arr[mid];

    int left = lo;
    int right = hi;

    while( left <= right ) {                          → No equality!!
        while (arr[left] < pivot) { (because we keep element equal to pivot on the right side)
            left++;
        }
        while ( arr[right] > pivot) {
            right--;
        }
problem!!   if (left <= right) {                      → important!! (when l cross r, as shown in the diagram on left page)
:. SWAP         int temp = arr[left];
            arr[left] = arr[right];
            arr[right] = temp;

            left++;
            right--;

        }
    }

    // smaller problems
    quickSort( arr, lo, right);
    quickSort( arr, left, hi);

}
```

PAGE NO.
DATE : / /

0	1	2	3	4	5
40	20	50	30	60	10

lo = 0
hi = 5
mid = 2
pivot = 50

40 < 50 ✓
20 < 50 ✓

[40, 20, 10, 30] [50, 60]

lo = 4
hi = 5
mid = 4
p = 60

lo = 0
hi = 3
mid = 1
pivot = 20

10, 20, 40, 30

10

BC

40, 30

lo = 2 hi = 3
mid = 2
pivot = 40

if (lo == hi)

problem!

Swap

[30, 40]

fh = [10, 20, 30, 40, 50, 60]

After performing quick sort for sh = [70, 80, 90]

& merge them //

main {

```
int [] arr = {20, 10, 30, 50, 60, 5, 80, 25};
quickSort (arr, 0, arr.length - 1);
for (int val : arr) {
        System.out.print (val + " ");
}
}
```

BC

```
if (lo >= hi) {
    return;
}
```

In place algorithm → all changes are made in the original array

Time Complexity

$O(n^3)$ → $O(2^n)$ (Fibonacci) → Highest Complexity

$O(n^2)$ – loop inside loop

$O(n \log n)$ – M.S., Q.S.

$O(N)$ – Linear Search, Factorial

$O(\log N)$ – B. Search

$O(1)$ – Constant

PAGE NO. icon
DATE : / /

** $n \log n + n \Rightarrow O(n \log n)$ → largest term

Time Complexity in Linear Search

$T(n) = T(n-1) + C$

$T(n-1) = T(n-2) + C$

$T(n-2) = T(n-3) + C$ (Add) $\Rightarrow T(n) = Cn$

⋮

$T(1) = C$

$O(n)$

Time complexity in Binary Search

→ check in half array & discard the other half

$T(n) = T(n/2) + C$

→ Halving array each time

$T(n/2) = T(n/4) + C$

$T(n/4) = T(n/8) + C$

⋮

$T(1) = C$

k steps

(Add)

$T(n) = kC$

$n, \frac{n}{2}, \frac{n}{4}, \frac{n}{8}, \frac{n}{16}, \cdots \frac{n}{2^k} = 1$

PAGE NO. icon
DATE : / /

$n = 2^k$

$\log_2 n = k$

$T(n) = \log_2 n \; C.$

$$\boxed{O(n) = \log_2 n}$$

Time Complexity in Dubble Sort, Selection Sort, Insertion Sort

$n + (n-1) + (n-2) + (n-3) + \ldots + (1)$ — 2nd larg. elem.

We perform 'n' operations to take the largest element towards the end.

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

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

Time Complexity of merge Sort

$T(n) = T(n/2) + T(n/2) + n$ → when we perform the merge operation

$T(n) = 2T(n/2) + n$

$2 \times T(n/2) = 2T(n/4) + n/2$

$2^2 \times T(n/4) = 2T(n/8) + n/4$

k steps

PAGE NO. icon
DATE : / /

$2^k \times T(1) = n$

$n, \frac{n}{2}, \frac{n}{4} \cdots$

$\frac{n}{2^k} = 1$

$n = 2^k$

$T(n) = Kn$

$T(n) = n \log n$

Time Complexity Array List

ops	capacity	size
0	2	0
1	2	1
1 (2)	2	2
2+1	4	3
1 (4)	4	4
4+1	8	5
1	8	6
1	8	7
1	8	8
8+1	16	9
1	16	10
1	16	11
1	16	12
1	16	13
1	16	14
1	16	15
1 (16)	16	16
16+1	32	17
⋮	⋮	⋮

to copy the prev. 2 elem

to copy the prev. 4 elem.

to copy the prev 8 elem.

16 elements

$2 + 4 + 8 + 16$

$= 30$ operations

1 element $\rightarrow \frac{30}{16}$ ops

1 element $\rightarrow$ (2) ops

add op $\rightarrow O(2)$ time (constant)

Constant time

PAGE NO. icon
DATE: / /

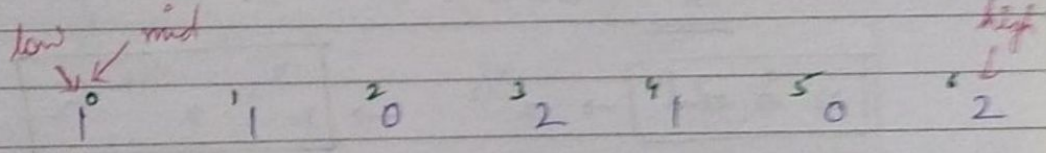
Add op. at a particular index → O(n) time.

Sort array of 0's, 1's, 2's

Dutch National Flag Algorithm

	0	1	2	3	4	5	6
arr =	1	1	0	2	1	0	2

T.C. O(n)
S.C. O(1)




```
int low = 0, int mid = 0, int high = arr.length-1;

while (mid <= high) {
  if (arr[mid] == 0) {
    swap (low, mid);

    low++;
    mid++;
  }
  else if (arr[mid] == 1) {
      mid++;
  }
  else {
      swap (mid, high);
      high--;
  }
}
```

PAGE NO. icon
DATE : / /

Power function in log(N)

```
pow = 1;
for( i=1; i<=n; i++) {
    pow = pow * x;
}
return pow;
```

O(n) ✓

x^n

n is odd → $x^{n/2}$ → $x^{n/2} \cdot x^{n/2} \cdot x$

$7^5 \rightarrow 7^{5/2} = 7^2 \rightarrow 7^2 \cdot 7^2 \cdot 7$

n is even → $x^{n/2}$ → $x^{n/2} \cdot x^{n/2}$

$7^4 \rightarrow 7^{4/2} = 7^2 \rightarrow 7^2 \cdot 7^2$

* $2^5 \rightarrow 2^{4} \cdot 2^{4} \cdot 2^{1} \rightarrow 32$

$2^4 \cdot 2^4 \cdot 2$

$2^2 \cdot 2^2 \cdot 2^1 \cdot 2^1$

$2^2 \rightarrow (2^0 \cdot 2^0) \cdot 2$

Base case

Recursion

$\log_2(5) = 2...$

PAGE NO.
DATE 5 / 1 / 1

```
public static int power (int x, int n) {
    int pnb2 = power (x, n/2);
    if ( n % 2 == 1 )
        return pnb2 * pnb2 * x;

    else
        return pnb2 * pnb2 ;

}
```

Sieve of Eratosthenis (SOE)

Prime no.

way 1

```
Scanner sc = new Scanner (System.in );
int n = sc.nextInt();
int fact = 0;
int div = 1;
while ( div <= n ) {
    if ( n % div == 0 ) {
        fact = fact + 1;
    }
    div = div + 1;
}
if (fact == 2)
    System.out ("prime");

else
    system.out ("not prime");
```

fact < n

1 2 3 4 5 6 . . . n

O(n)

PAGE NO. icon
DATE : / /

To find → numbers count which are prime till 1000

$O(n^2)$

way 2 :

18

$n/2$ → tak 18 divide ho jaayega

1
2
3
.
.
.
$n/2$

1 no. ko check karne ke liye
n nos. ko check karne ke liye

$\frac{n}{2} \cdot n$

$O(n^2)$.

way 3 :

18
2 X 9 , 3 X 6
2 X

36 $\sqrt{36} = 6$.

2 X 18
3 X 12
4 X 9
6 X 6 — $\sqrt{n}$
9 X 4
12 X 3
18 X 2.

} mirror hai

$n/2$

* jitne bhi factors milne honge → 18 tak mil jaayenge
* agar prime hua to value bhi nhi milegi

PAGE NO. icon
DATE : / /

Better → Kisi bhi number ko $\sqrt{n}$ complexity me bata sakte hain ki wo prime number hai ya nhi

'n' numbers ke liye → $n\sqrt{n}$

$O(n\sqrt{n})$

Best way → SOE

way 4 : → $\sqrt{25}$ tak table bolo → kyunki 25 ant ki jagah $\sqrt{25}$ se (last 25 ka)

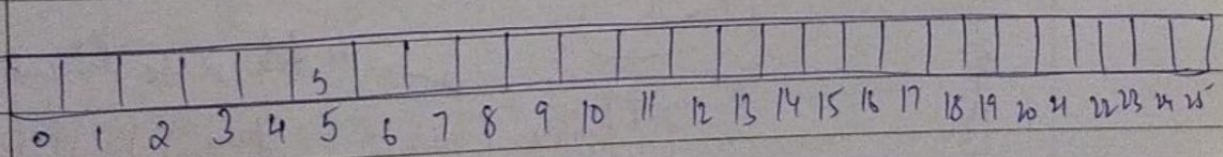
25 tak kitne prime nos. h?

Algo:

~~1~~	2	3	~~4~~	5
~~6~~	7	~~8~~	~~9~~	~~10~~
11	~~12~~	13	~~14~~	~~15~~
~~16~~	17	~~18~~	19	~~20~~
~~21~~	~~22~~	23	~~24~~	~~25~~

1D array banaya bool. ka

ant → false
not ant → true



PAGE NO. icon
DATE : / /

```
public static void SOE (int n) {

    boolean [] primes = new boolean [n+1];
    Arrays.fill (primes, true);                    } O(n)

    primes [0] = primes [1] = false;

    for (int table = 2; table * table <= n; table++) {
        for (int mult = 2; mult * table <= n; mult++) {
            primes [table * mult] = false;
        }
    }

    for (int i = 0; i < primes.length; i++) {
        if (primes [i] == true) {
            System.out (i);
        }
    }
}
```

Table Logic → ek no. ki table me 2 se start krte hain

2×2 = 4
2×3 = 6
⋮
2×13 = 26 ✗

Cond^n

2
3
5
7
11
13
17
19
23

```
if (primes [table] == false) {
    continue;
}
```

→ 4 toh 2 se hi cut ho gya tha toh no need to check for 4.

2 liye outer for loop ko 4 ke liye nhi chalayenge.

O(n)

PAGE NO.
DATE : / /

table = 2

4 6 8 10 12 14 16 18
20 22 24

$O(n/2)$

$n/2$ elements

table 3 3 6 9 12 15 18 21 24

↳ $n/3$

table 4 X

table 5 5 10 15 20 25

↳ $n/5$

table 7

11

13

$$\frac{n}{2} + \frac{n}{3} + \frac{n}{5} + \frac{n}{7} + \frac{n}{11} + \frac{n}{13} + \frac{n}{17} + \text{——}$$

$$n\left(\frac{1}{2} + \frac{1}{3} + \frac{1}{5} + \frac{1}{7} + \frac{1}{11} + \frac{1}{13} + \frac{1}{17} + \text{——}\right)$$

↳ Sum of reciprocals of prime

$\Theta(n \log\log n)$