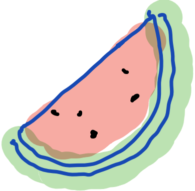
Sorting 1



What is sorting?

1, 2, 3, 4, 5 → Asc

5, 4, 3, 2, 1 → Desc

7, 2, 4, 9, 6 → Asc order → # factors

↓ ↓ ↓ ↓ ↓

2 2 3 3 4

Why sorting? → Making search easier & faster

Name	12th %
Ankur	83
Ashish	71
Nikhil	95
Rohit	70
Navneet	59
Komal	73
Abhay	93
Poornima	93
Kshitij	86

Sort Asc by 12th score →

Name	12th %
Navneet	59
Rohit	70
Ashish	71
Komal	73
Ankur	83
Kshitij	86
Abhay	93
Poornima	93
Nikhil	95

Poo | 93

Ab | 93

Stable sorting : If 2 data points have same value then their relative order in intial data should be maintained

7 2 3 4 2 8 1

Stable 1 2 2 3 4 7 8

Inplace Algo : O(1) SC

Q ⇒ Given a array of N element. Find the kth minimum

1 , 5, -1, 2 , 10 , 3 $K < \log n$

K=3 → 2

K=5 → 5

Appraoch 1 :

- Sort the array
- Ret A[K-1]

TC: (N log N)

Approach 2 :

i= 0 select 1st min → Iterate from 0 → n-1 & find min swap(0, ind_{min}) ↳ O(N) i=1 —— 2^{nd} — → ———— 1 → n-1

& find min O(N)

swap(1, ind_{min})

i=K —— K^{th} min → O(N)

O(KN)

⇓

selection sort ← O(n^2) K=n

```
for( i=0; i<N; i++)
{
    min = A[i] ;    ind = i
    for  (j= i ; j< N ; j++)
            if ( A[j] < min)
                min= A[j]
                ind = j;
    swap( A[i] , A[ind] );
}
```

O(1)

Inplace ✓

Stable No

#swaps

(n-1)

2 , 5, 2 , 1, 6

1 , 5, 2 , 2 , 6

1, 2 , 5 , 2 , 6

1 , 2 , 2, 5, 6

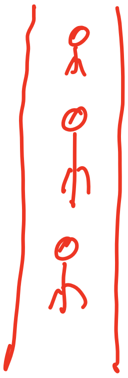
H/w : Implement stabe version of selection sort

Q ⇒ Given an array of N element. swapping of non-consequitive indexes is not allowed. Sort the array in Asc order

```
     8    6    2    2              5
3    9    9    9    9     9    4   11      11
~~9~~  ~~3~~  ~~8~~  ~~6~~  ~~7~~  ~~2~~  ~~11~~  ~~4~~  ~~5~~  → O(n)

          7    2              5        9
     6    ~~8~~    ~~8~~   8    4    ~~9~~
3    ~~8~~    ~~6~~    ~~7~~    2    ~~9~~    ~~11~~    ~~5~~    11  → O(n)

3    6    7    2    8    4    5    9    11
```



TC: $O(n^2)$

SC: $O(1)$ → Inplace

```
for( i=0;  i<N;  i++)
    for( j=0;  j< N-1-i ;  j++)
        if ( A[j] > A[j+1])
            swap( A[j], A[j+1]);
```

Stable   yes

Inplace   yes

#swaps $O(n^2)$

n logn n^2

sc: o(n)   1

θ ⇒ Given 2 sorted arrays of size N & M. Merge both & return new sorted Array

Amazon
WhatsApp

A: 2, 5, 7, 12, 20, 24, 29 ← N

B: 6, 9, 10, 14, 18, 19 ← M

C: 2, 5, 6, 7, 9, 10, 12, 14, 18, 19, 20, 24, 29

⇑ N+M

a a

N → A: 2, 5, 7, 12, 20, 24, 29

O(n+m)

M → B: 6, 9, 10, 14, 18, 19

b b b b

a = 4
b = 6
M = 6
N = 7

C: 2, 5, 6, 7, 9, 10, 12, 14, 18, 19, 20, 24, 29

A[a] B[b]

```
int[] merge (A[], N, B[], M)
{
    C[N+M] → New array
    a = 0 , b = 0 , c = 0
    while ( a < N && b < M)
    {
        if ( A[a] < B[b])
        {
```

SC: O(n+m)

```
            C[c]= A[a];
             a++;
          }
        else
          {
             C[c] = B[b];
              b++;
          }
          c++;
      }
      while( a<N)
       {
          C[c] = A[a];
           a++ ; c++;
       }
      while( b < N)
        {
           C[c]= B[b];
            b++; c++;
        }
      ret  c;
    }
```

Break: 9:00

Q ⇒ Given array of size N, 3 indexes l, y, r
l → r ⇒ sorted combined array

↓l ↓y ↓r

8, 1, $\boxed{3, 6, 11}$, $\boxed{2, 4, 9}$, 7, 6

8, 1, $\boxed{2, 3, 4, 6, 9, 11}$, 7, 6

↓l ↓y ↓r

8, 1, $\boxed{3, 6, 11}$, $\boxed{2, 4, 9}$, 7, 6

↑

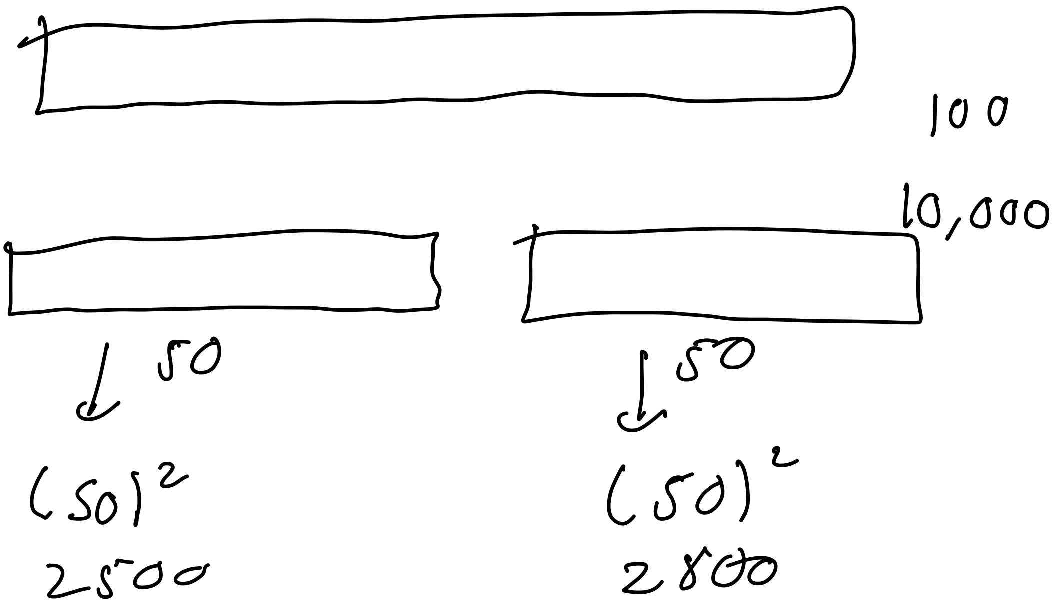
| 2 | 3 | 4 | 6 | 9 | 1 |

r − l + 1

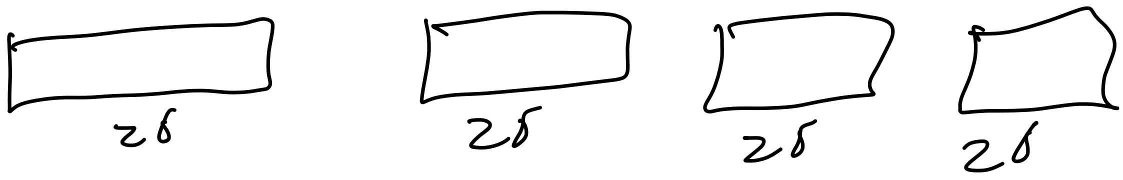
```
int[] merge (A[], l, y, r)
{
    c [r-l+1]  → New array
    a = l , b= y , c= 0
    while ( a < y && b ≤ r)
    {
        if ( A[a] < A [b])
        {
            c [c]= A[a];
            a ++;
        }
      else
        {
          c[c] = A[b];
           b++;
        }
        c++;
    }
    while( a <y)
     {
       C[c]  = A[a];
        a++ ; c++;
     }
    while( b ≤ r)
      {
        C[c]= A[b];
         b++;  c++;
     }
    for (i=0;  i< (r-l+1) ; i++)
     {
      A[i+l] = C[i]
     }
}
```

Bubble sort : $O(n^2)$

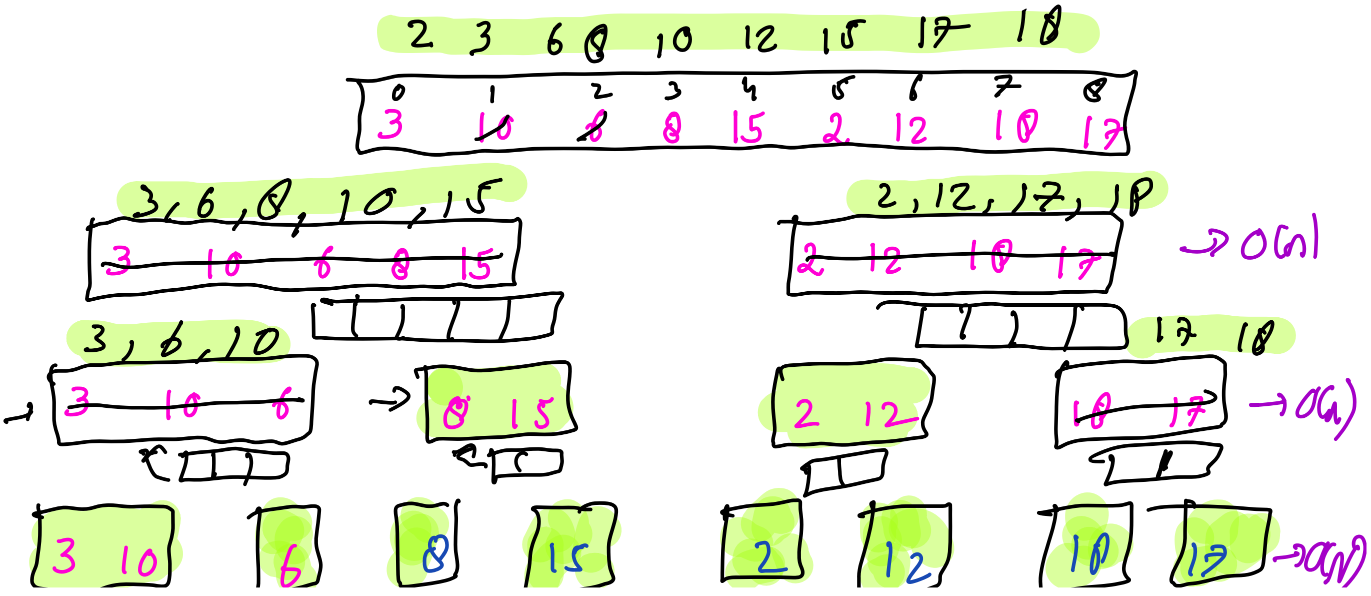
selection sort : $O(n^2)$



2500 + 2500 + 100 = 5100



$4 * (25)^2 =$ 2500 + 200



$\boxed{3}$ $\boxed{10}$

$\overline{O(N \log N)}$

```
void mergesort (A[], l, r)
 {
    if (l == r)   ret;

    mid = (l+r)/2;
    mergesort (A, l, mid);
    mergesort (A, mid+1; r);
    merge( A, l, mid+1, r);
 }
```

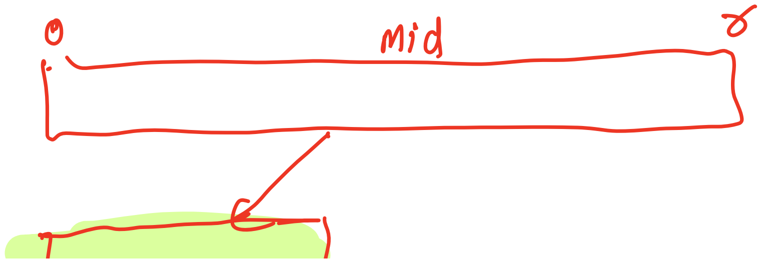
TC:

$$T(n) = T(\tfrac{n}{2}) + T(\tfrac{n}{2}) + T(n)$$

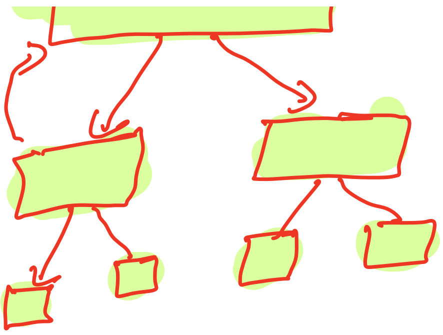
$$= 2T(\tfrac{n}{2}) + T(n)$$

SC : $O(\log N) + O(n)$

: $O(N)$



if (l == r)   ret;



```
mid=  (l+r) /2;
merge sort  ( A , l, mid);
merge sort ( A , mid+1; r);
merge( A, l , mid+1, r);
}
```