Sorting

Arranging data in a particular order!

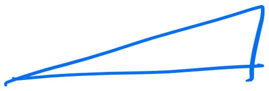
9, 8, 7, 6, 5 : Sorted in DESC order

Q Given an Arr. Sort it in ASC order!

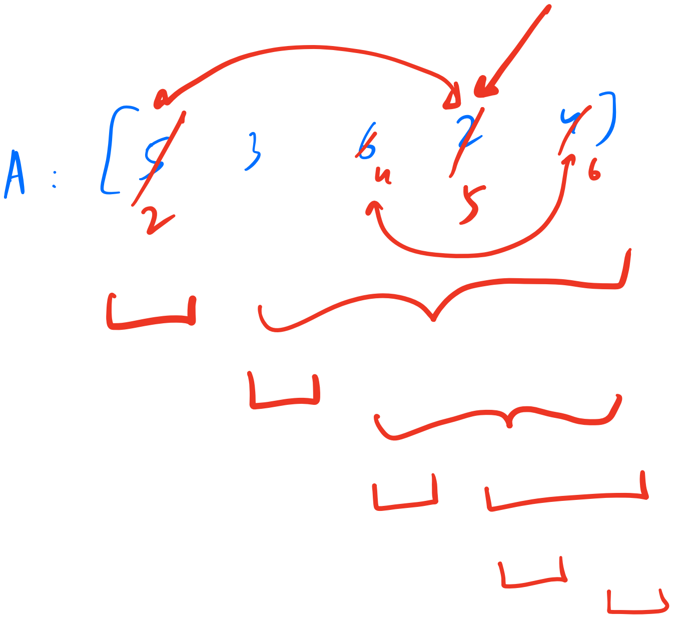
A : [5 3 6 2 4]

↓

[2 3 4 5 6]



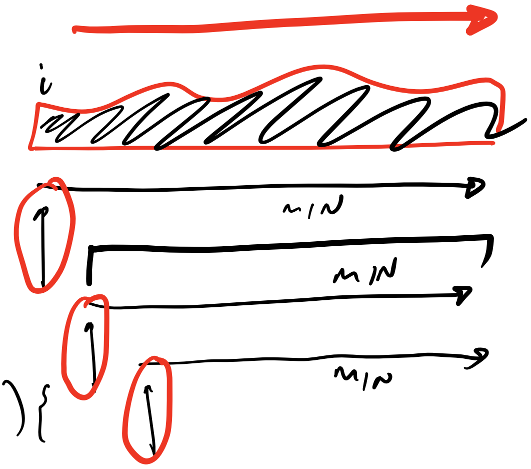
Idea: find smallest & put it at first place!

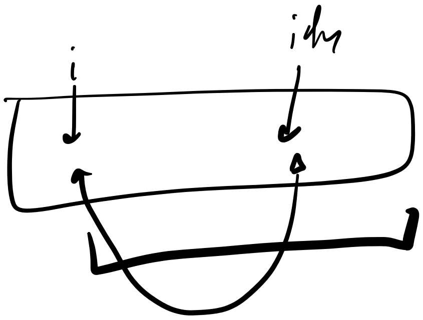


SELECTION SORT

// A[], N

```
f(i=0; i<N-1; i++){
    idx = i;
    f(j=i+1; j<N; j++){
        if(A[j] < A[idx]){
            idx=j;
        }
    }
    swap(A[i], A[idx]);
}
```





$\boxed{TC = O(N^2)}$

$\boxed{SC = O(1)}$

✓

* INPLACE SORTING ALGOS

→ No extra space req'd for sorting!

BUBBLE SORT

A: ~~6~~ ~~5~~ ~~3~~ ~~2~~ ~~4~~ ~~1~~

~~1~~ 4

~~1~~

2

pass 1: 1 ~~1~~ ~~1~~ ~~1~~

6 5 3

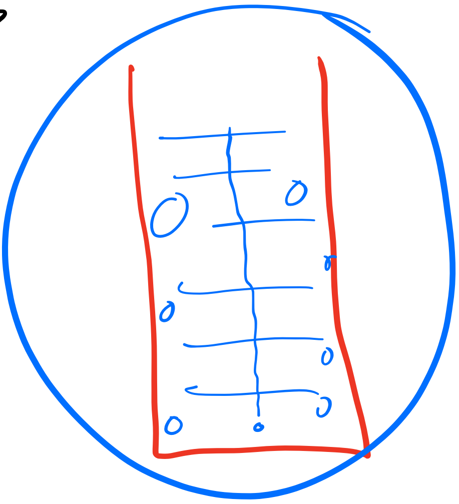
1 | ~~6~~ ~~5~~ ~~3~~ ~~2~~ 4

2 ~~2~~ ~~2~~ 3

6 5

pass 2:

1 2 6 5 3



$$\boxed{TC = O(N^2)}$$

$$\boxed{SC = O(1)}$$

pass N-1 :

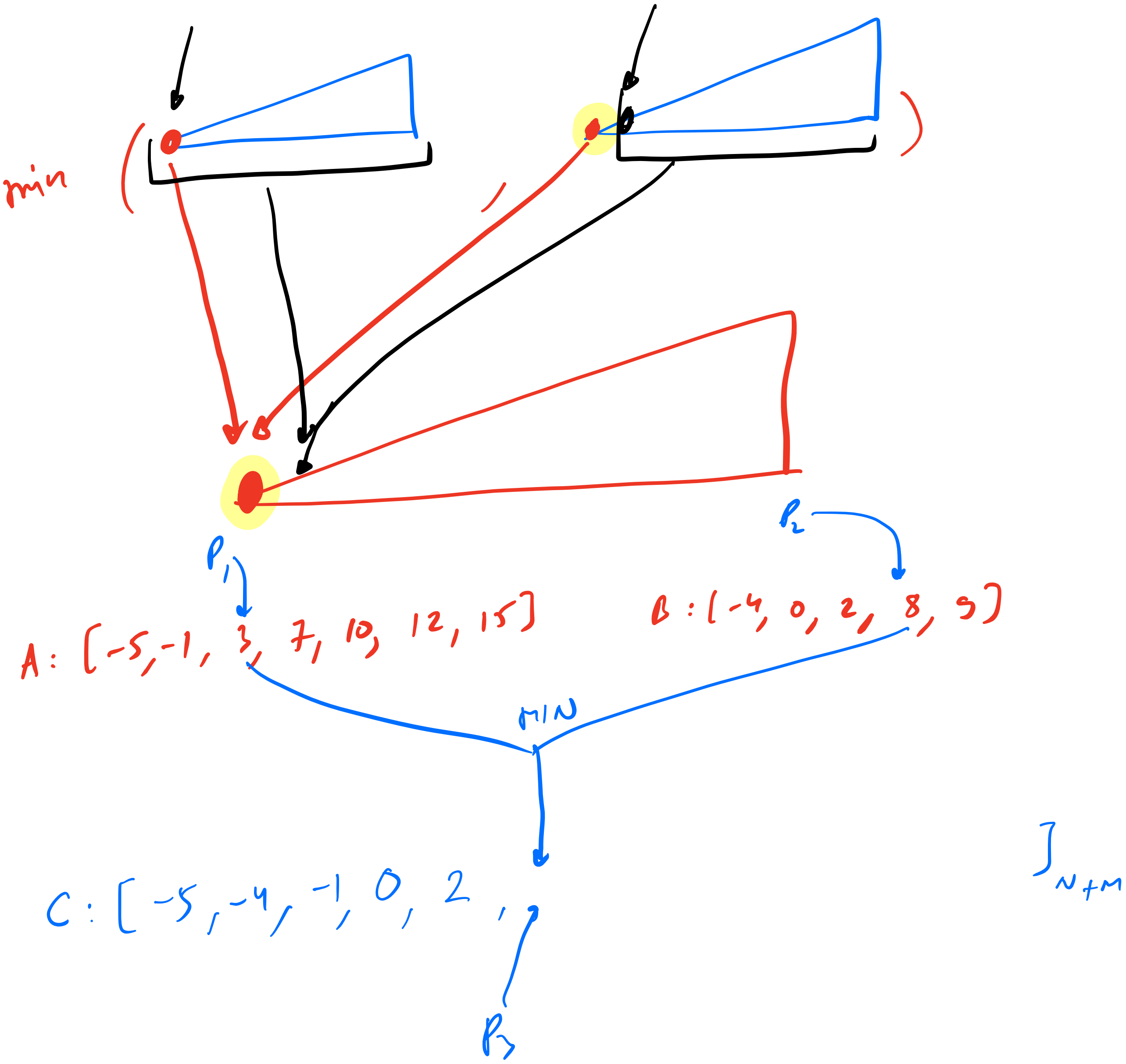
Q Given 2 sorted Array A[N], B[M].
Merge & create a new sorted array C[N+M];

A : [-1, 4, 8]
B : [1, 2, 7, 10]

I/P

O/P

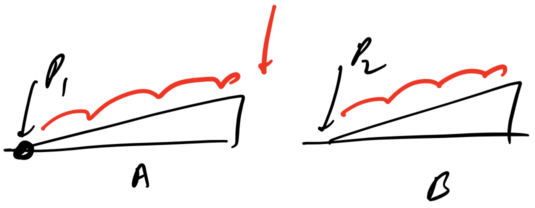
C= [-1, 1, 2, 4, 7, 8, 10]

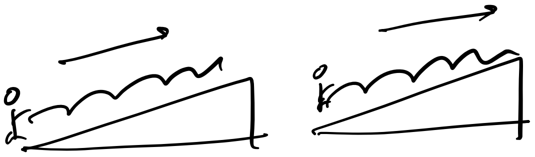

min
p1
p2
A: [-5, -1, 3, 7, 10, 12, 15]
B: [-4, 0, 2, 8, 9]
MIN
C: [-5, -4, -1, 0, 2, ,
]N+M
p3

CoDe:

```
// A[N]   , B[M] ;
int C[N+M];
  P1 = 0, P2 = 0 ,  P3 = 0;
  while(P1 < N && P2 < M){
     if(A[P1] < B[P2]) {
           C[P3] = A[P1];
           P1 ++, P3 ++;

     }
     else {
           C[P3] = B[P2];
           P2 ++, P3 ++;
     }
  }
  while( P1 < N) {
        C[P3] = A[P1];
        P1 ++, P3 ++;
  }
  while( P2 < M) {
        C[P3] = B[P2]
        P2 ++, P3 ++;
  }
```





C : N+M

$\boxed{TC = O(N+M)}$

$\boxed{SC = O(1)}$

Q Given an array A of size N.

& 3 indexes. s, m, e. //0<=s<=m<e<N

Given: A[s, m] is sorted!

→ A[m+1, e] is sorted!

Sort the A[s, e]

s m e

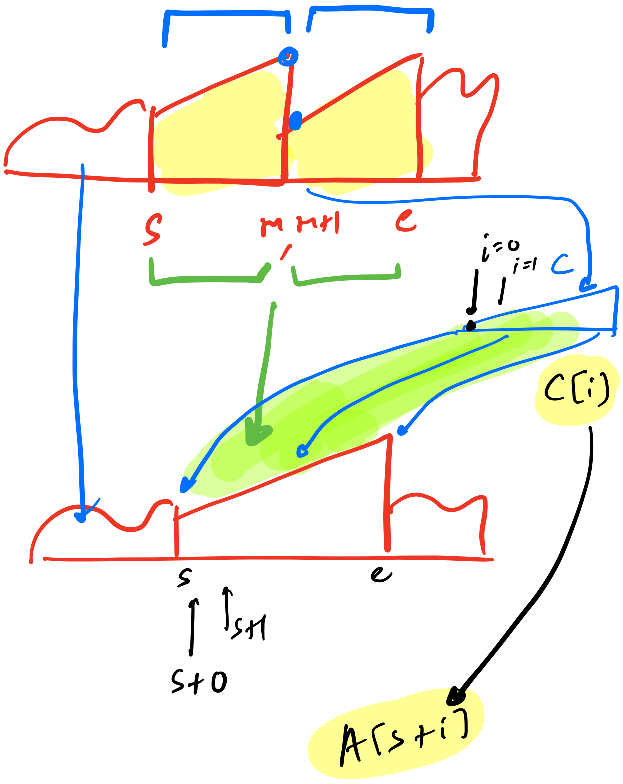
0 1 2 3 4 5 6 7 8 A

A: 5 10 1 2 11 8 10 15 6

↓

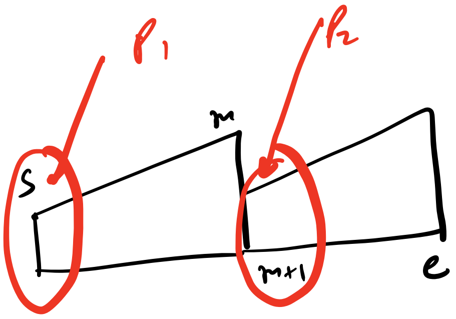
A: 5 10 1 2 8 10 11 15 6

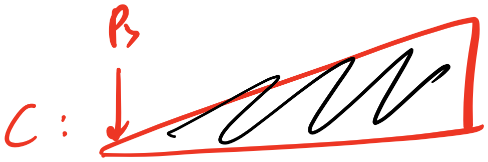
A



CoDE:

```
// A[], N,   s, m,  e
int C[e-s+1];
P1 = s, P2 = m+1, P3 = 0;
while(P1 <= m && P2 <= e {
    if( A[P1] < A[P2]) {
        C[P3] = A[P1];
        P1++, P3++;
    }
    else {
        C[P3] = A[P2];
        P2++, P3++;
    }
}
while( P1 <= m) {
    C[P3] = A[P1];
    P1++, P3++;
}
while( P2 <= e) {
    C[P3] = A[P2]
    P2++, P3++;
}
f( i=0; i< e-s+1; i++) {
    A[s+i] = C[i];
}
```



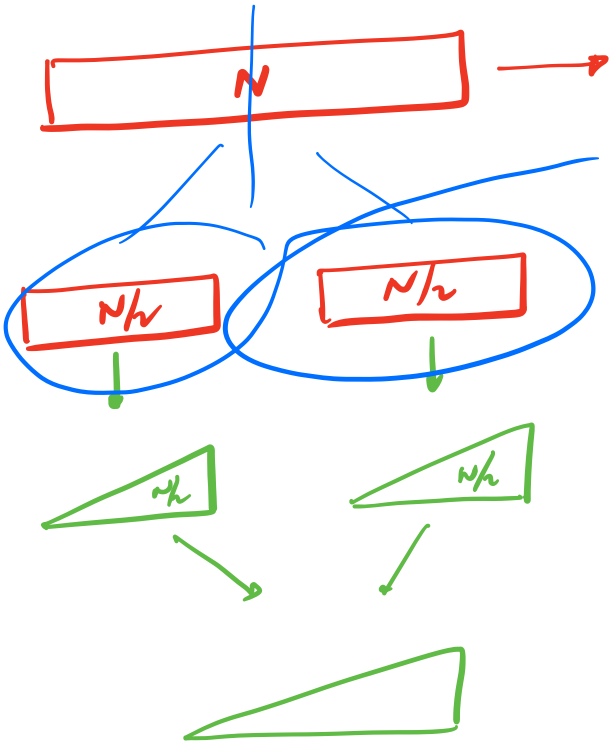


$\boxed{TC = O(e-s)}$

SC

Q Given an arry of size N. Sort it in Asc order!

$N = 10^5$



$\boxed{N^2} \rightarrow 10^{10}$

$2 \times (N/2)^2 + N$

$2 \cdot \frac{N^2}{4} + N$

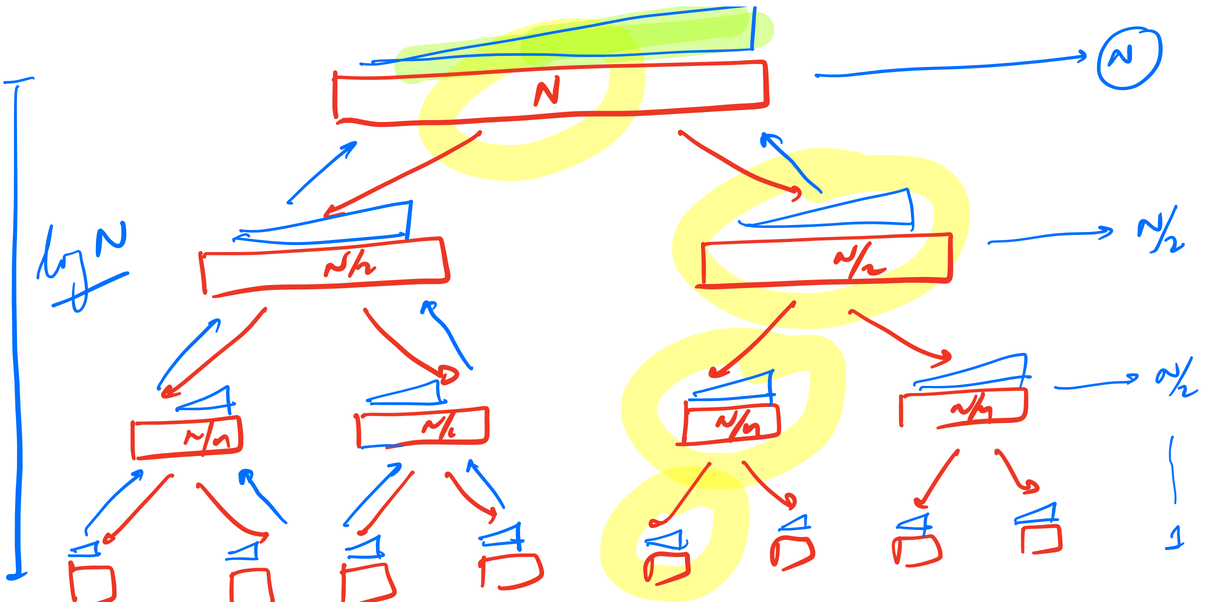
$\boxed{\frac{N^2}{2} + N}$

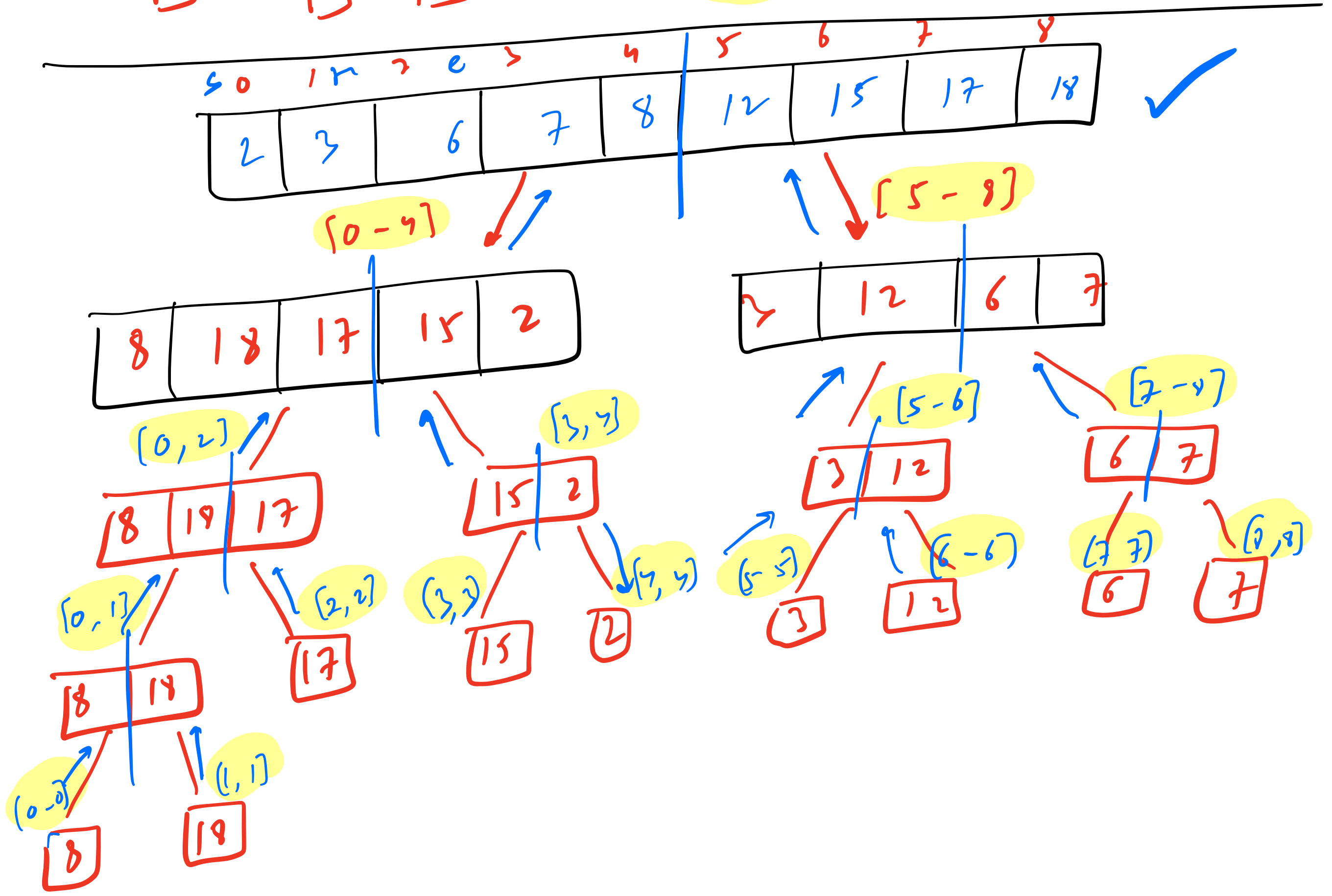
$\frac{10^{10}}{2} + 10^5$

$5 \cdot 10^9 + 10^5$

$\sim 5 \cdot 10^9$

MERGE SORT

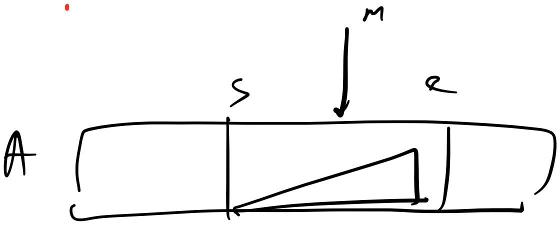

N
N
log N
N/2
N/2
N/2
N/4
N/4
N/4
N/4
N/2
1


0
1
2
3
4
5
6
7
8
2
3
6
7
8
12
15
17
18
[0 - 4]
[5 - 8]
8
18
17
15
2
2
12
6
7
[0, 2]
[3, 4]
[5 - 6]
[7 - 8]
8
18
17
15
2
3
12
6
7
[0, 1]
[2, 2]
(3, 3)
(4, 4)
[5 - 5]
[6 - 6]
[7, 7]
[8, 8]
8
18
17
15
2
3
12
6
7
[0 - 0]
[1, 1]
8
18

(0, -1)

```
void mergeSort( A[], s, e){
// Ass: Sort the Array A[s, e]!
        if (s >= e) { ret; }
        m = (s+e)/2;
        mergeSort( A[], s, m);
        mergeSort( A[], m+1, e);
        merge( A[], s, m, e);
}
```

(A, 0, N-1)



$$\boxed{T(N) = 2 \cdot T(N/2) + N}$$

N = N/2

$T(N/2) = 2\ T(N/4) + N/2$

N = N/4

$T(N) = 2\left[2\ T(N/4) + N/2\right] + N$

$T(N) = 4\ T(N/4) + 2N$

$T(N/4)$

$T(N) = 8\ T(N/8) + 3N$

$T(N) = 16\ T(N/16) + 4N$

$$T(N) = 2^k \underbrace{T(N/2^k)}_{} + \underbrace{k}_{} . N$$

$$\underline{T(1) = 1}$$

$$N/2^k = 1$$

$$2^k = N$$

$$k = \log_2 N$$

$$T(N) = N\,T(1) + \underline{\log N . N}$$

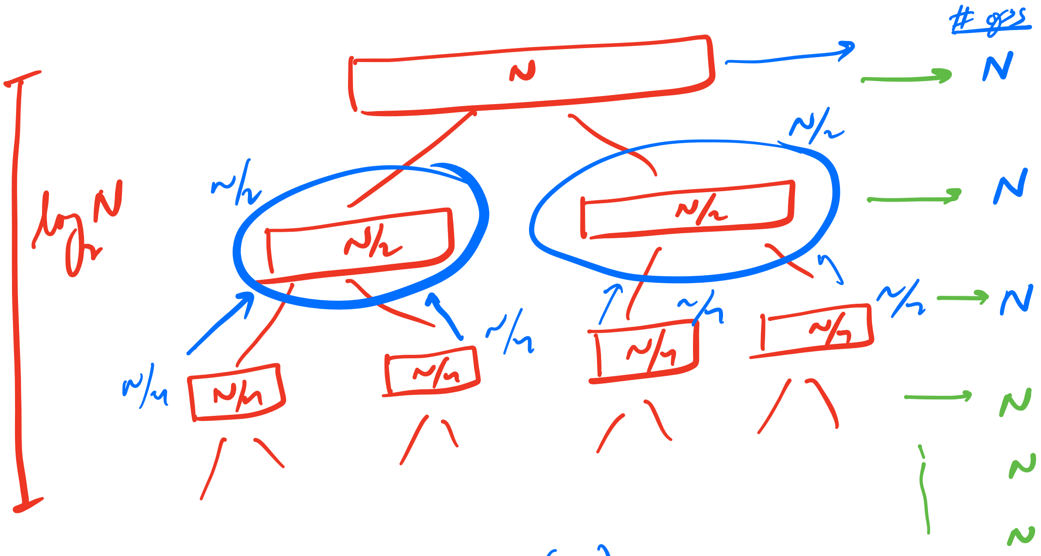
$$N \times 1 + N \log N$$

$$T(N) = N \log N$$

$$\boxed{TC : O(N \log N)}$$

$$SC = \underset{\substack{\swarrow \\ C[\,] \\ (\text{merge})}}{N} + \underset{\substack{\downarrow \\ \text{recursion}}}{\log N}$$

$$\boxed{SC = O(N)}$$

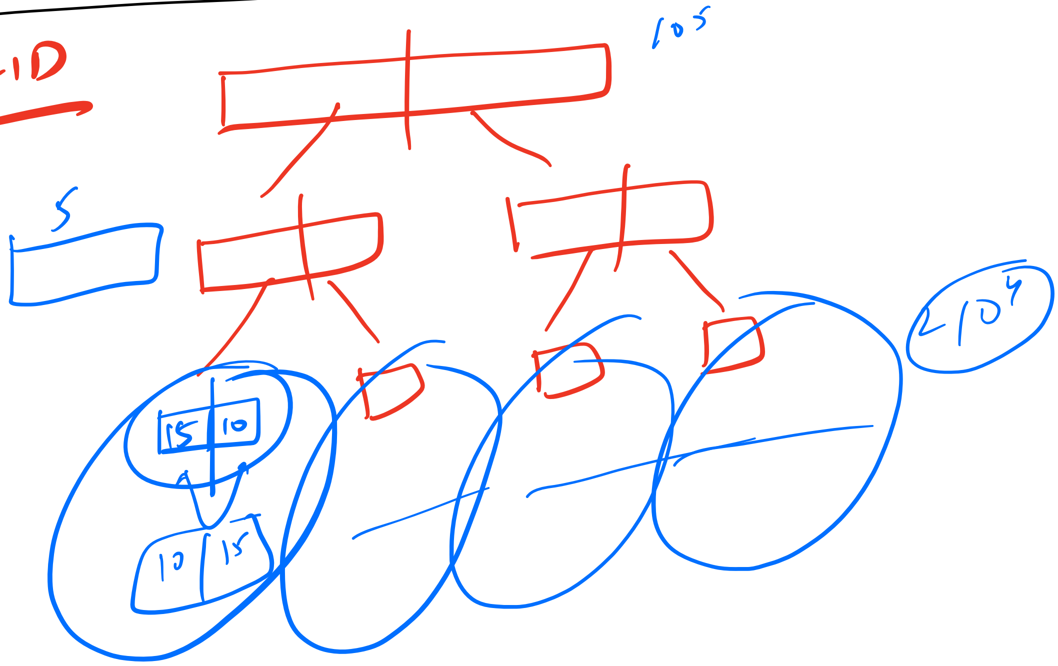


1 level → $O(N)$

log N levels → $O(N \log N)$

$$\boxed{TC = O(N \log N)}$$

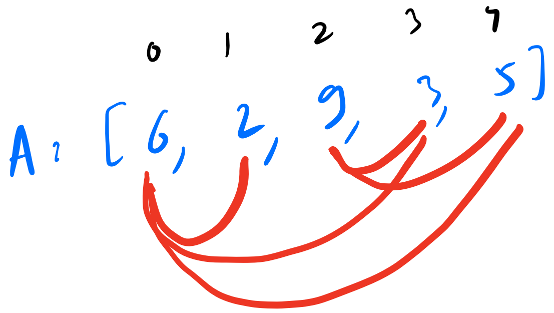
HYBRID



```
void mergeSort( A[], s, e){
  // Ass: Sort the Array A[s, e]!
  if( e-s+1 <= 5){ selectionSort( A[], s, e); }
  m = (s+e)/2;
  mergeSort( A[], s, m);
  mergeSort( A[], m+1, e);
  merge( A[], s, m, e);
}
```

Q Given an Array. find the # of pairs (i, j)

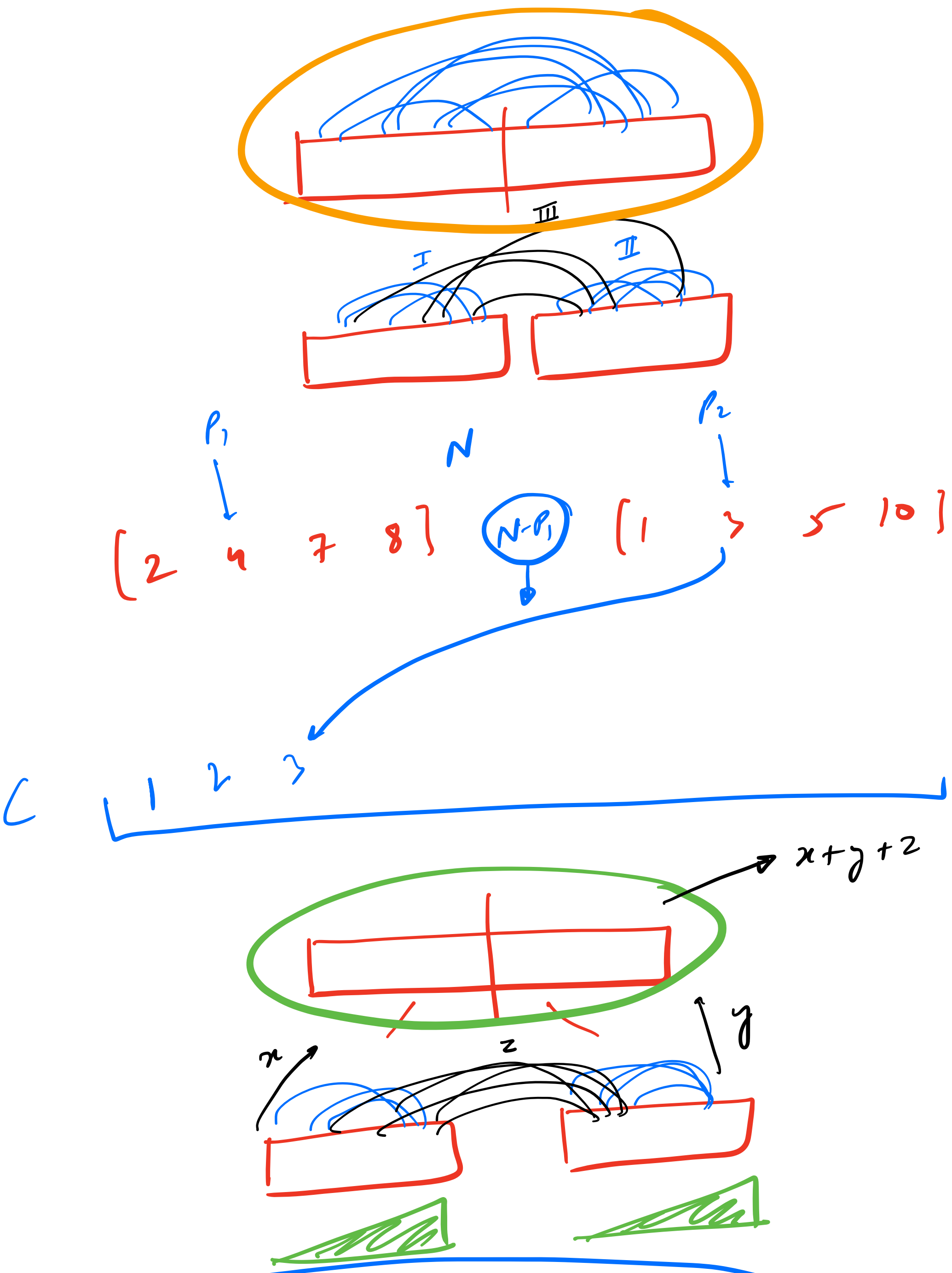
: $(i < j)$ & $(A_i > A_j)$ Inversion Count!



1) BF)

```
for(i: 0 → N-1)
    for(j: i+1 → N-1)
        if(A[i] > A[j]) cnt++
```

$O(N^2)$



Ass :

1. Ret the INV count!
2. Sort the Arrays as well!

```
int    invCnt ( A[], s, e) {
// Ass: Sort the Arry A[s, e], ret the inv cnt;
  if( s >= e ){ret 0;}
    m = (s+e)/2;
    x = invCnt ( A[], s, m);
    y = invCnt( A[], m+1, e);
     z = merge( A[], s, m, e);
    ret x + y + z;
}
```