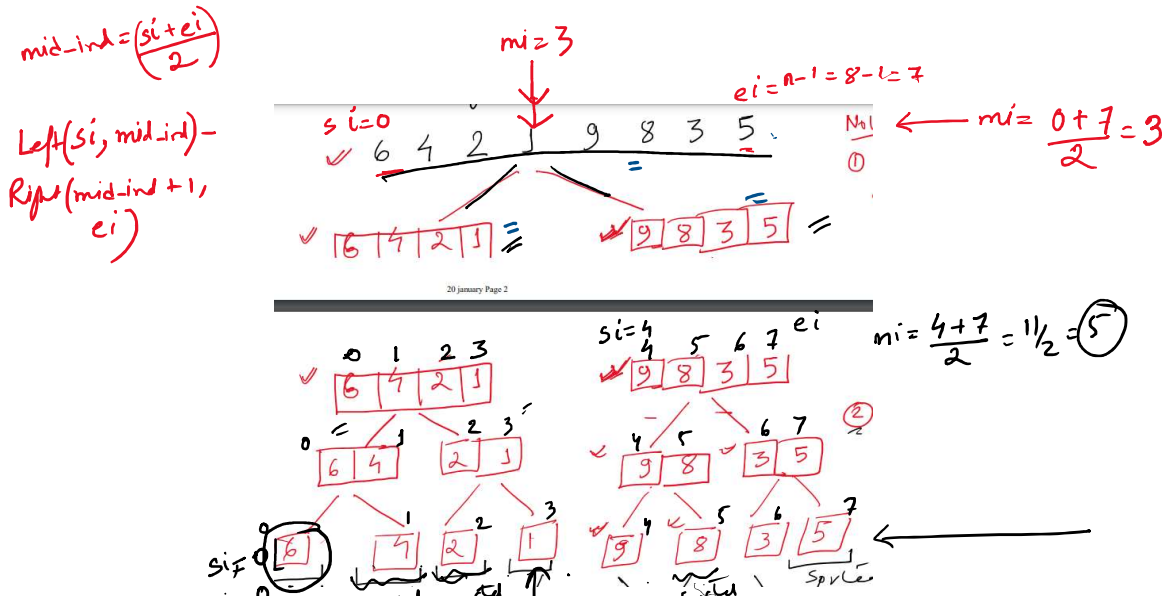
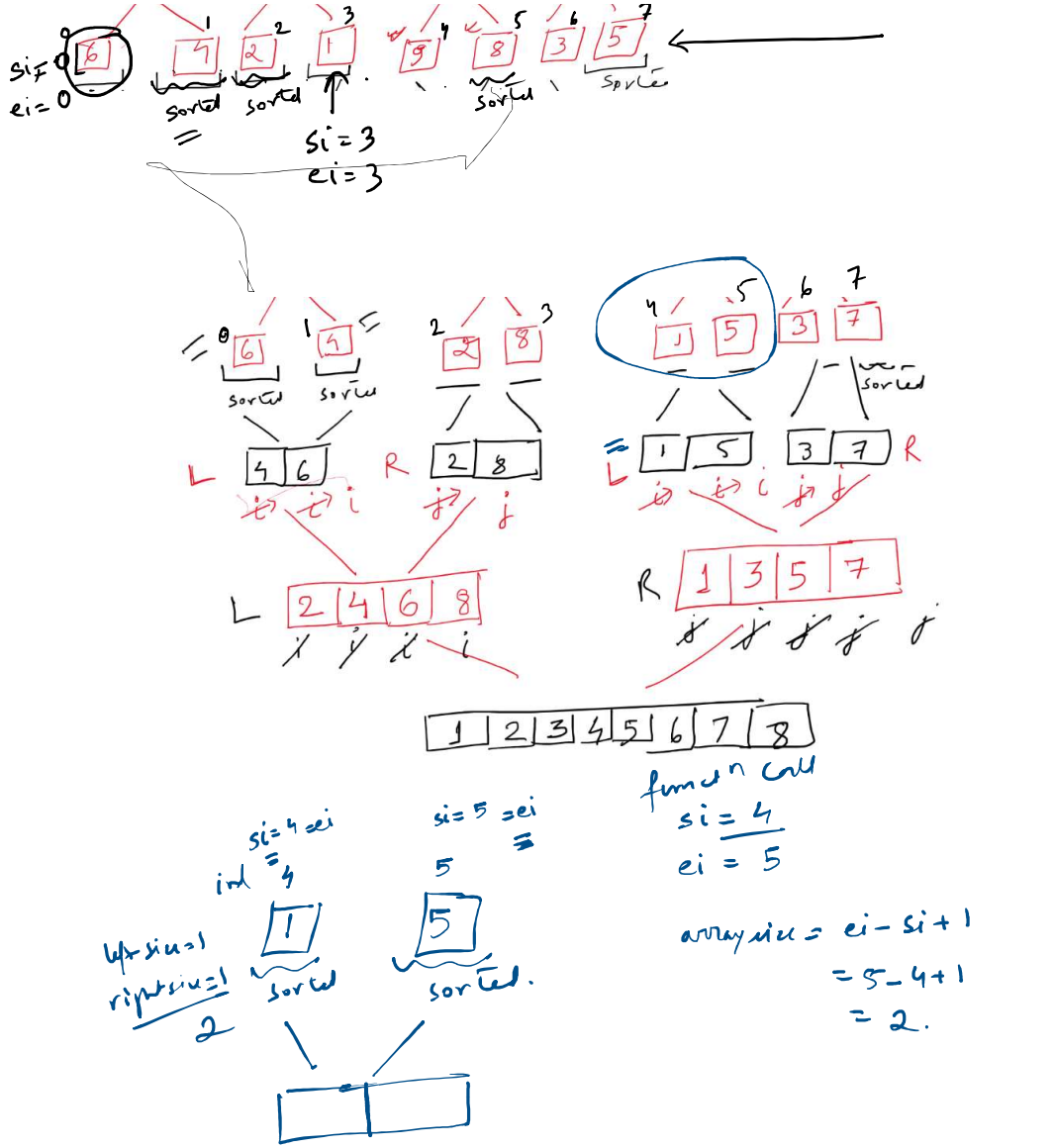
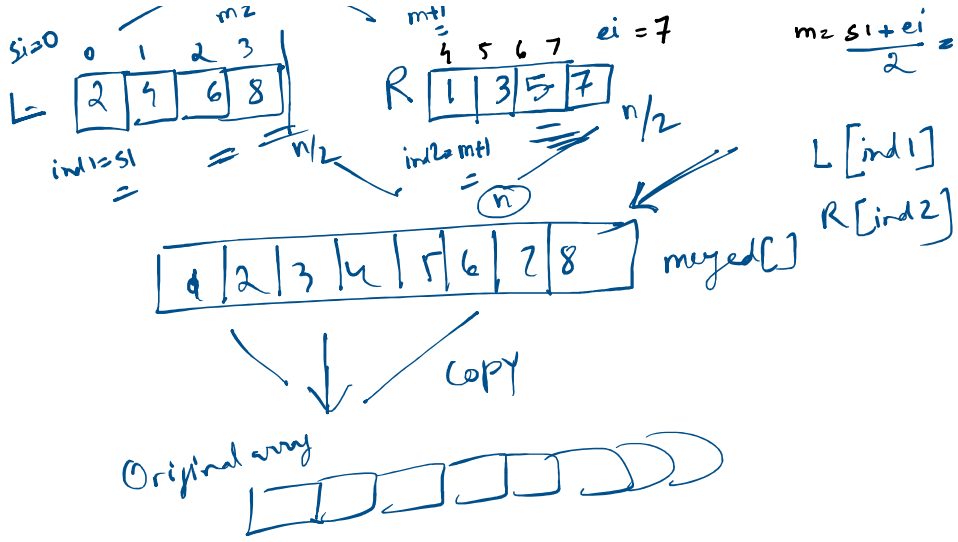
```java
public static void divide(int arr[], int si, int ei) {
    if (si >= ei) {
        return;
    }
    int mid = (ei + si) / 2;
    divide(arr, si, mid);
    divide(arr, mid + 1, ei);
    conquer(arr, si, mid, ei);
}
```

```java
divide(arr, si:0, n - 1);
```




si=0
ei=0
sorted
sorted
si=3
ei=3
sorted
sorted
L
R
2 4 6 8
1 3 5 7
1 2 3 4 5 6 7 8
function call
si = 4
ei = 5
si=4=ei
int
4
si=5=ei
5
leftsize=1
rightsize=1
2
sorted
sorted.
array size = ei - si + 1
= 5 - 4 + 1
= 2.
int merged[] = new int[ei - si + 1];
7 - 0 + 1 = 7+1
= 8
si=0
m2
m+1
ei = 7
R 1 3 5 7
m2 si+ei / 2 = 7+0 / 2 = 3



$mid = \frac{si+ei}{2} = \frac{7+0}{2} = 3$

SC → W.C → O(n)

TC →

Recursion

```
public static void divide(int arr[], int si, int ei) {
    if (si >= ei) {
        return;
    }

    int mid = (ei + si) / 2;
    divide(arr, si, mid); //left array
    divide(arr, mid + 1, ei);// right array
    conquer(arr, si, mid, ei);
    //left sorted array and right sorted array are merged into single
    sorted array
}
```

T(n/2) maths input

T(n/2) maths input.

n

Recursion → Recurrence Relation → Solve → TC

$T(n) = \begin{cases} 2T(n/2) + n \end{cases}$

$aT(n/b) + n^k \log^p n$

$T(n) =$

$a=2, b=2, k=1, p=0$

$a = b^k \quad 2 = 2$

condition (2a) →

$a = b^k \quad 2 = 2$

$p = 0$, condition 2a →

$T(n) = \Theta\left(n^{\log_b a} \log^{p+1} n\right)$

$= \Theta\left(n^{\log_2 2} \log^1 n\right)$

$= \Theta(n \log n)$ //