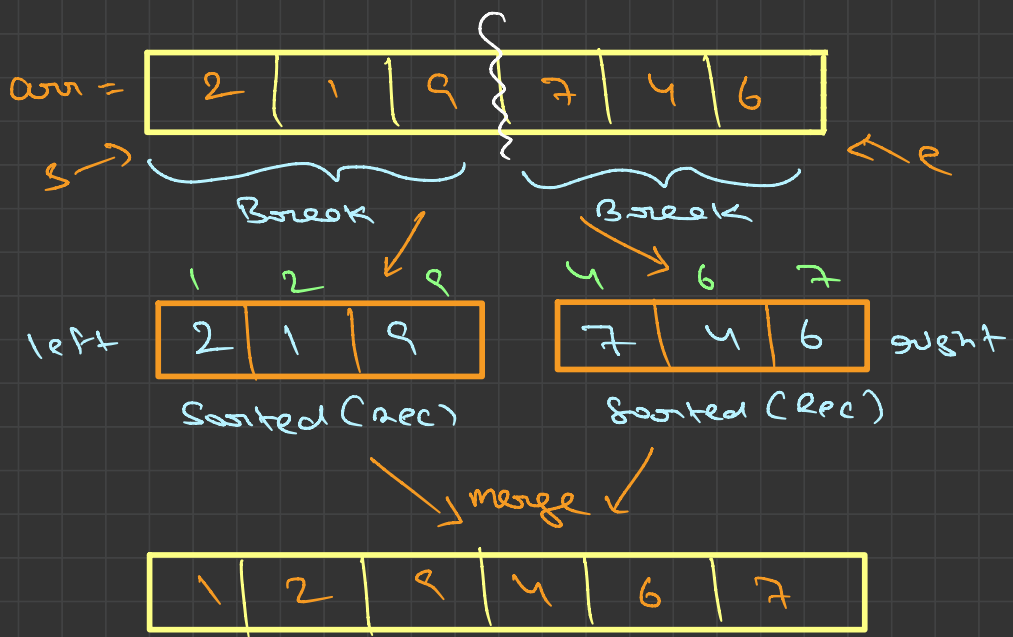
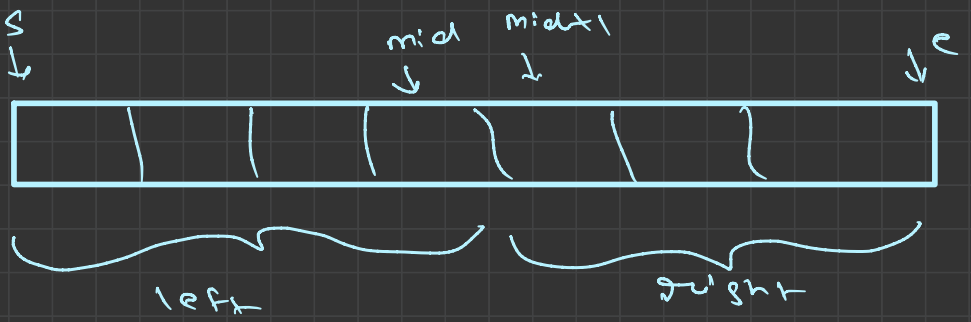
Merge Sort



H → Break
II → Recursion
III → merge 2 sorted array

Base Case Rec

```
if ( s > e ) return;
if ( s == e) return;
```



How to find length?

left = mid - s + 1

right = e - (mid + 1) + 1

↳ e - mid - ~~1 + 1~~

↳ e - mid

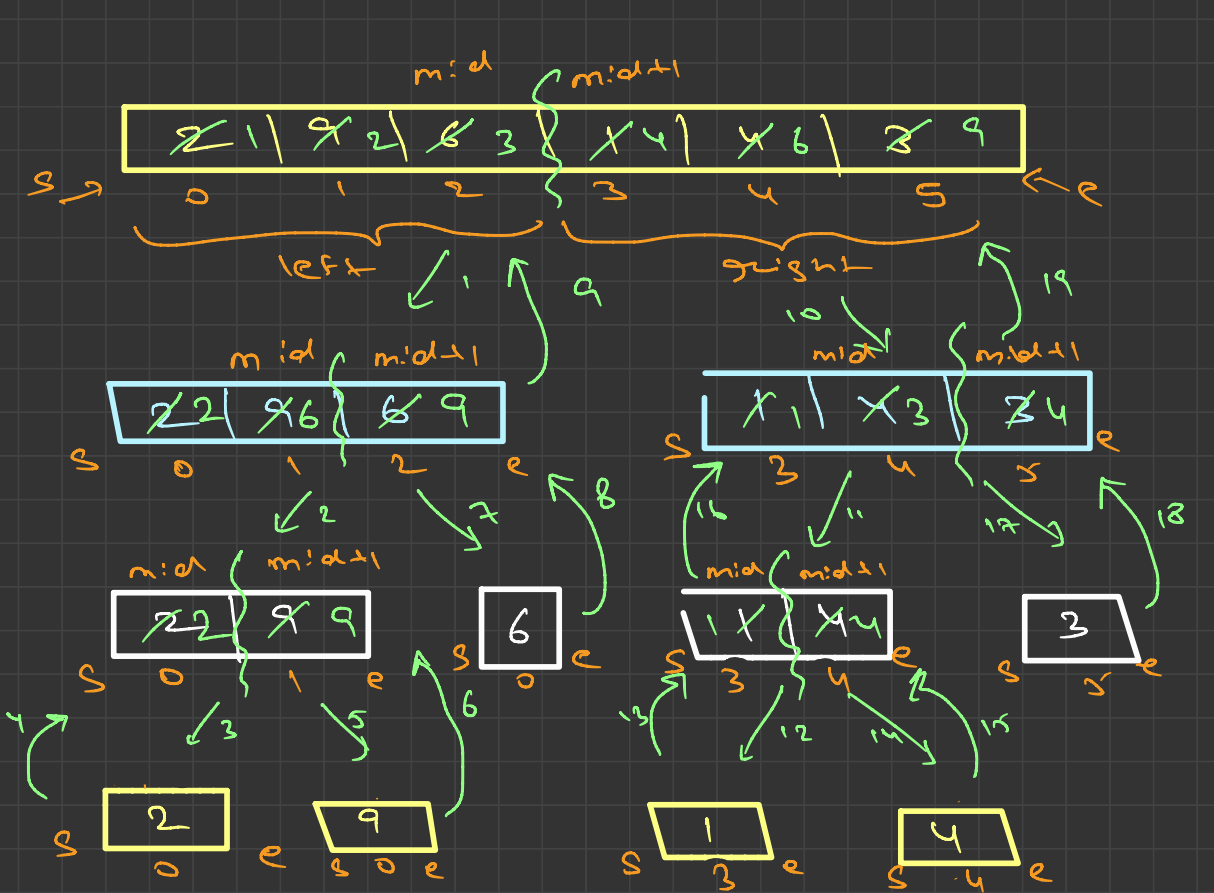
```
void mergeSort ( int arr[] , int s , int e )
{
    // base case
    if ( s > e ) return ;
    if ( s == e) return ;

    // break
    int mid = (s+e) / 2 ;

    // rec call
    mergeSort ( arr , s , mid) ;
    mergeSort ( arr , mid+1 , e ) ;

    // merge
    merge ( arr, s , e , mid ) ;

}
```


mid
mid+1
left
right
0
1
2
3
4
5
s
e

```
void merge(int arr[], int s, int e)
    {
        // find mid
        int m = (s + e) / 2;

        // find length of left and right array
        int lenLeft = m - s + 1;
        int lenRight = e - m;

        // create left and right array
        int *leftArr = new int[lenLeft];
        int *rightArr = new int[lenRight];

        // copy value in left and right array
        int k = s; // for tracking original array index

        // copy value in left array
        for(int i = 0; i < lenLeft; i++){
            leftArr[i] = arr[k];
            k++;
        }

        // copy value in right array
        for(int i = 0; i < lenRight; i++){
            rightArr[i] = arr[k];
            k++;
        }

        // compare both sorted left and right array and copy in original array in sorted way
        int leftIndex = 0;
        int rightIndex = 0;
        int orgIndex = s;

        while(leftIndex < lenLeft && rightIndex < lenRight){
            if(leftArr[leftIndex] < rightArr[rightIndex]){
                arr[orgIndex] = leftArr[leftIndex];
                leftIndex++;
                orgIndex++;
            }
            else{
                arr[orgIndex] = rightArr[rightIndex];
                rightIndex++;
                orgIndex++;
            }
        }

        while(leftIndex < lenLeft){
            arr[orgIndex] = leftArr[leftIndex];
            leftIndex++;
            orgIndex++;
        }

        while(rightIndex < lenRight){
            arr[orgIndex] = rightArr[rightIndex];
            rightIndex++;
            orgIndex++;
        }

        delete[] leftArr;
        delete[] rightArr;
    }
```

$T(n) = k_1 + 2 * T\left(\frac{n}{2}\right) + n * k$

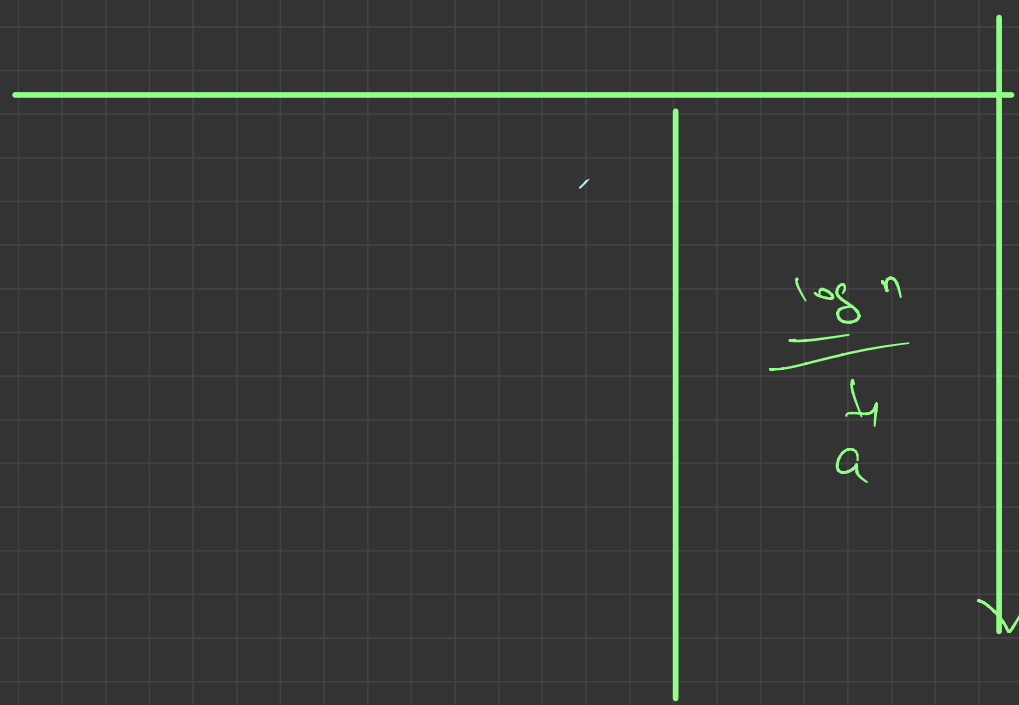
$2 * T\left(\frac{n}{2}\right) = k_1 + 4 * T\left(\frac{n}{4}\right) + \frac{n}{2} * k * 2$

$4 * T\left(\frac{n}{4}\right) = k_1 + 8 * T\left(\frac{n}{8}\right) + \frac{n}{4} * k * 4$

$8 * T\left(\frac{n}{8}\right) = k_1 + 16 * T\left(\frac{n}{16}\right) + \frac{n}{8} * k * 8$

$\vdots \qquad \vdots$

$2^{(a-1)} * T(1) = 2^{a-1} * k_1$



$$T(n) = \underbrace{k_1(1+2+4+\cdots+2^{q-1})}_{\text{G.P.}} + (q-1)n*k$$

$$= k_1 * n + q * n * k$$

$$= k_1 n + n \log n * k$$

$$= n \log n * k$$

$$= n \log n$$

Inplace