The Pseudocode for MergeSort is

low → start index
high → ending index.

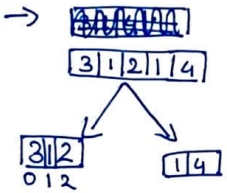
```
MergeSort (arr, low, high)
{
    mid = (low+high)/2.
    if (low >= high)
        return;
    MergeSort (arr, low, mid)
    MergeSort (arr, mid+1, high)
    Merge (arr, low, mid, high)
```

If low == high then the code breaks from the function

Recursion

Taking arr = [3, 1, 2, 1, 4] as an example.

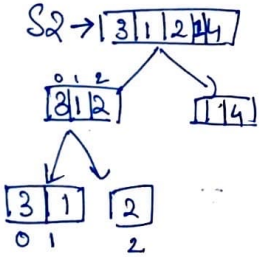
S1 →




```
MergeSort(arr, 0, 4)
{
    mid = (low+high)/2 → (2)
    (0+4)/2 → 2
    if (low >= high)
        return
    MergeSort (arr, 0, 2);
    MergeSort (arr, mid+1, high)
    Merge (arr, low, mid, high)
}
```

(1)

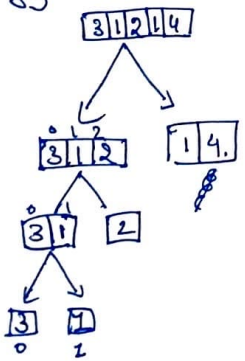
S2 →




```
MergeSort (arr, 0, 2)
{
    mid = (low+high)/2; → 1
    (0+2)/2
    if (low >= high) return;
    MergeSort (arr, 0, 1)
    MergeSort (arr, mid+1, high)
    Merge (arr, low, high, mid).
}
```

(2)

S3




```
MergeSort (arr, 0, 1)
{ mid = low+high/2 → 0
    0+1/2
    MergeSort (arr, 0, 0)
    MergeSort (arr, mid+1, high)
    Merge (arr, low, high, mid)
}
```

① After 0, 2 is sorted

This activates the MS (arr, mid+1, high) of 0, 1 is activated

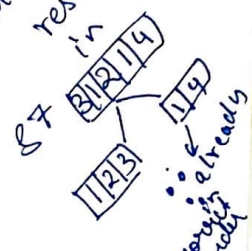
After (0, 1) is done the MS (arr, mid+1, high) activates for (0, 2)

```
MergeSort (arr, 3, 4)
{ mid = (low+high)/2 → 3
    (3+4)/2
    MergeSort (arr, low, mid)
    MergeSort (arr, mid+1, high)
    Merge (arr, low, high, mid)
```

returns: low = mid = 3

Merges indexes 3 & 4 resulting in (3, 4) is sorted

S7



S6

S8

② This activates after (3, 4) gives back

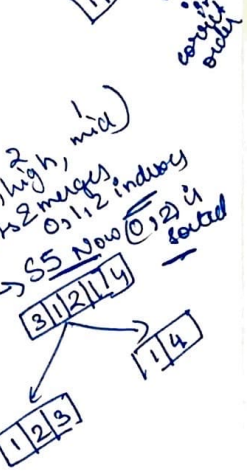
③ returns back ∵ low >= high (2 == 2) condition has been hit

④ MergeSort (arr, mid+1, high)

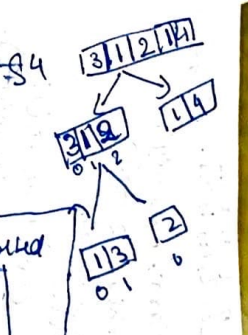
then Merge (arr, low, high, mid) will activate

S5 Now (0, 2) is sorted

sorts & merges 0, 1, 2 indexes



S4



(0, 1) is sorted

Merge (arr, 0, 1, mid)

3 → MergeSort (arr, 0, 0)

4 ← returns ∵ (low >= high)

2nd MergeSort fun is activated → MergeSort (arr, mid+1, high) (1)

returns ∵ (low >= high) (1 >= 1) (0 == 0) hence goes back

(5) 6

(7) Merges & Sorts the divided arr