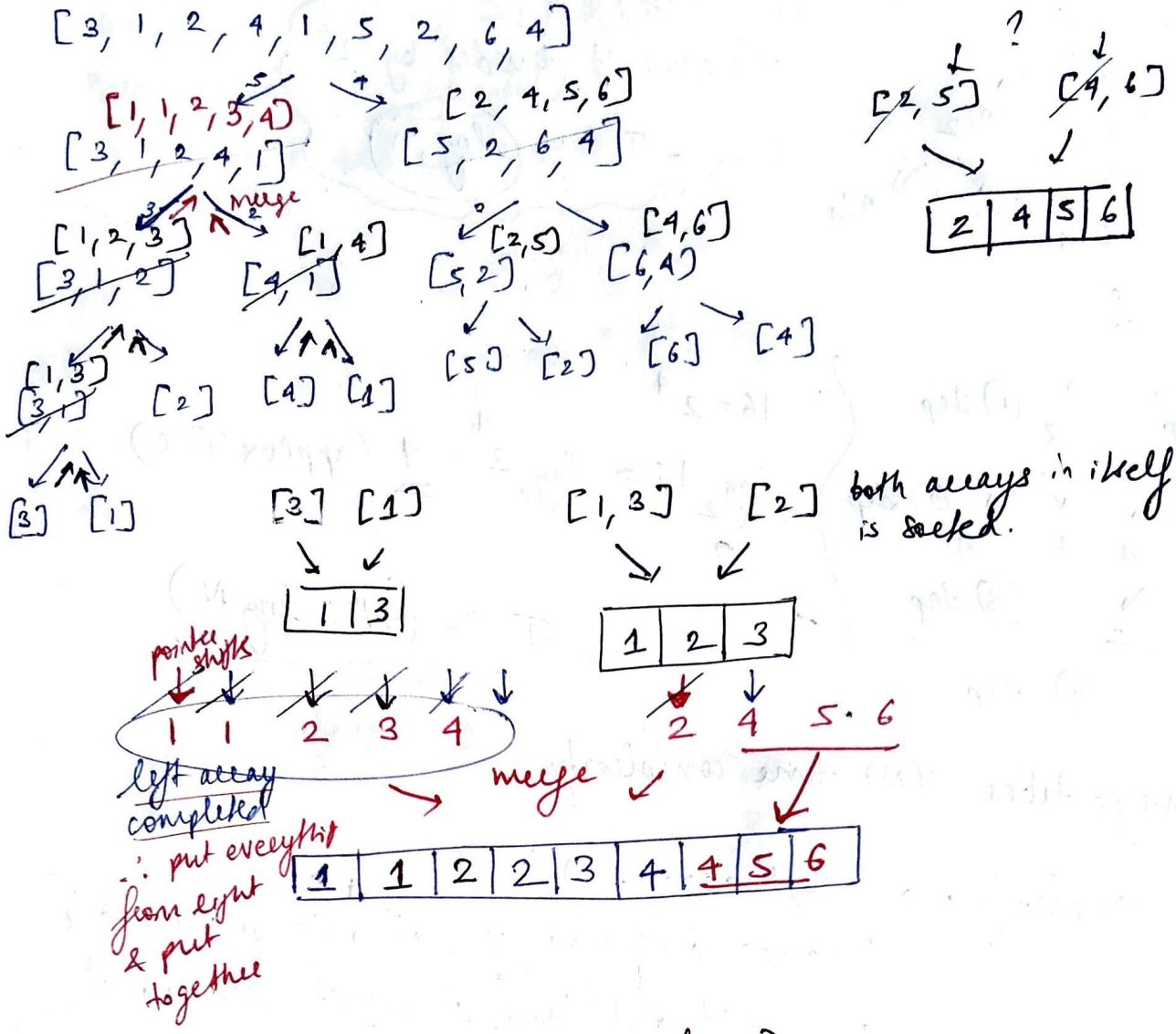
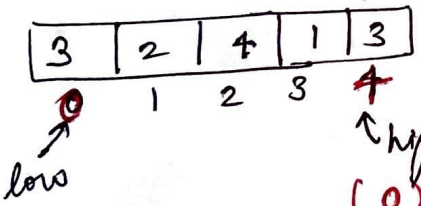
MERGE SORT → Divide & Merge



Pseudocode :- (play around with indexes)

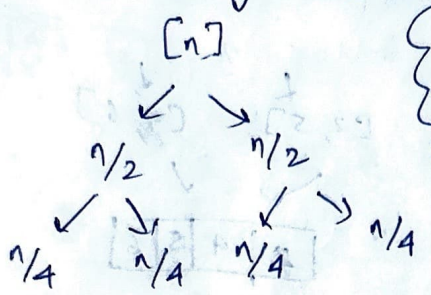


eg.

(0) (4)

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

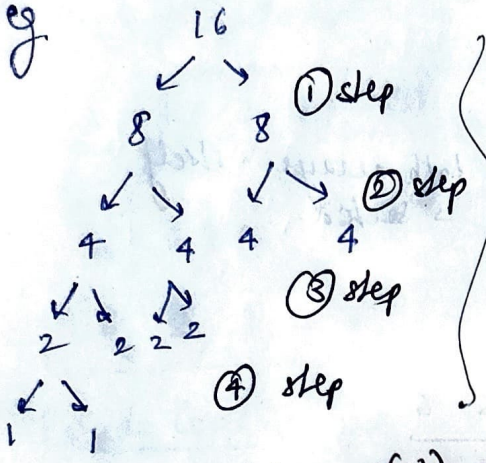
Time Complexity of Merge Sort



Note: - ~~*~~ wherever it divides by 2

T.C = $\log_2 n$

eg



$16 = 2^4$

$\log_2 16 = \log_2 2^4 = 4$ (approx T.C)

$\therefore$ T.C = $(N \times \log N)$

& merge takes O(N) time complexity

Space complexity → worst case = O(N)

because during merge, new temp variable is created and thus at worst scenario it would take space O(N)