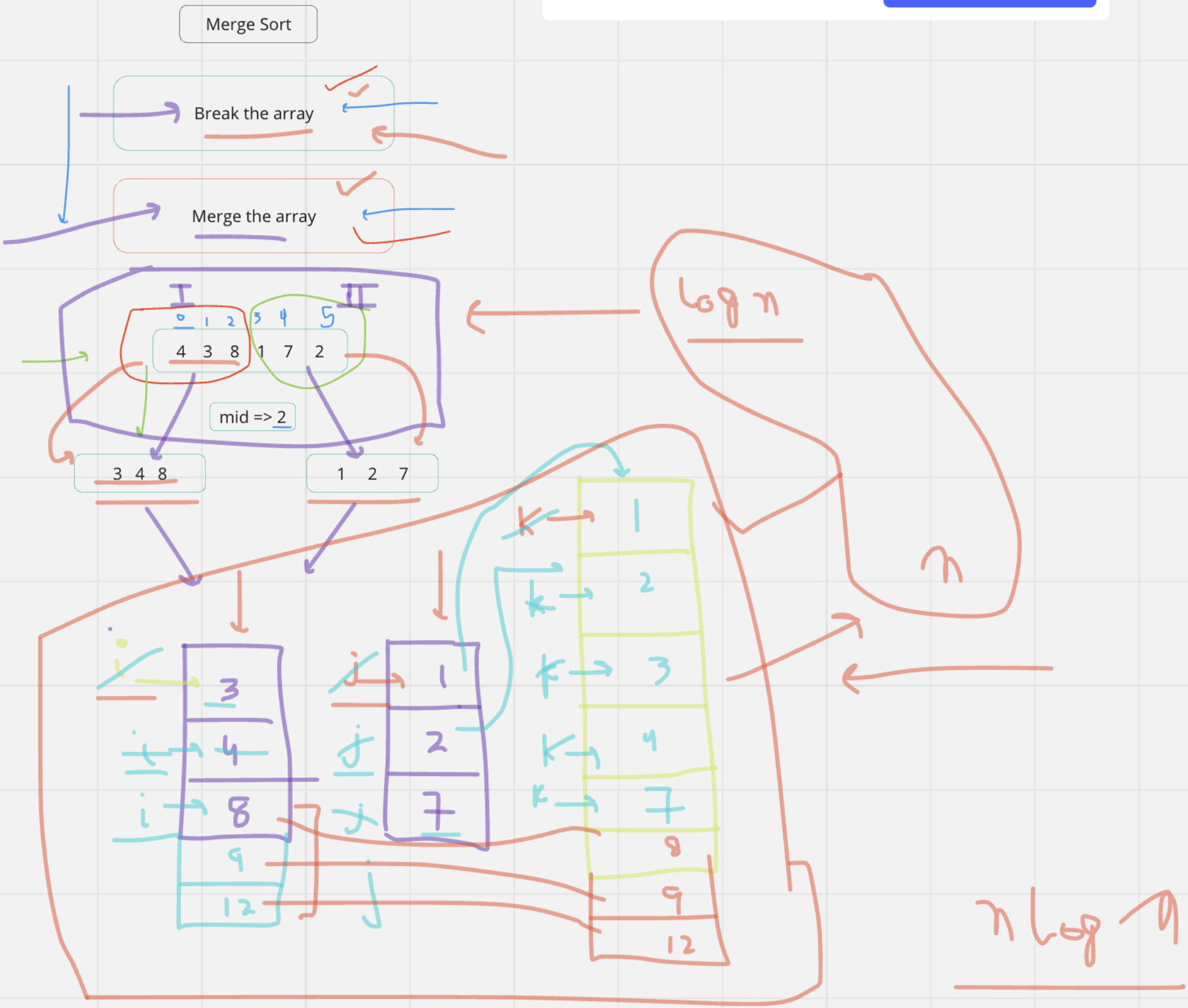
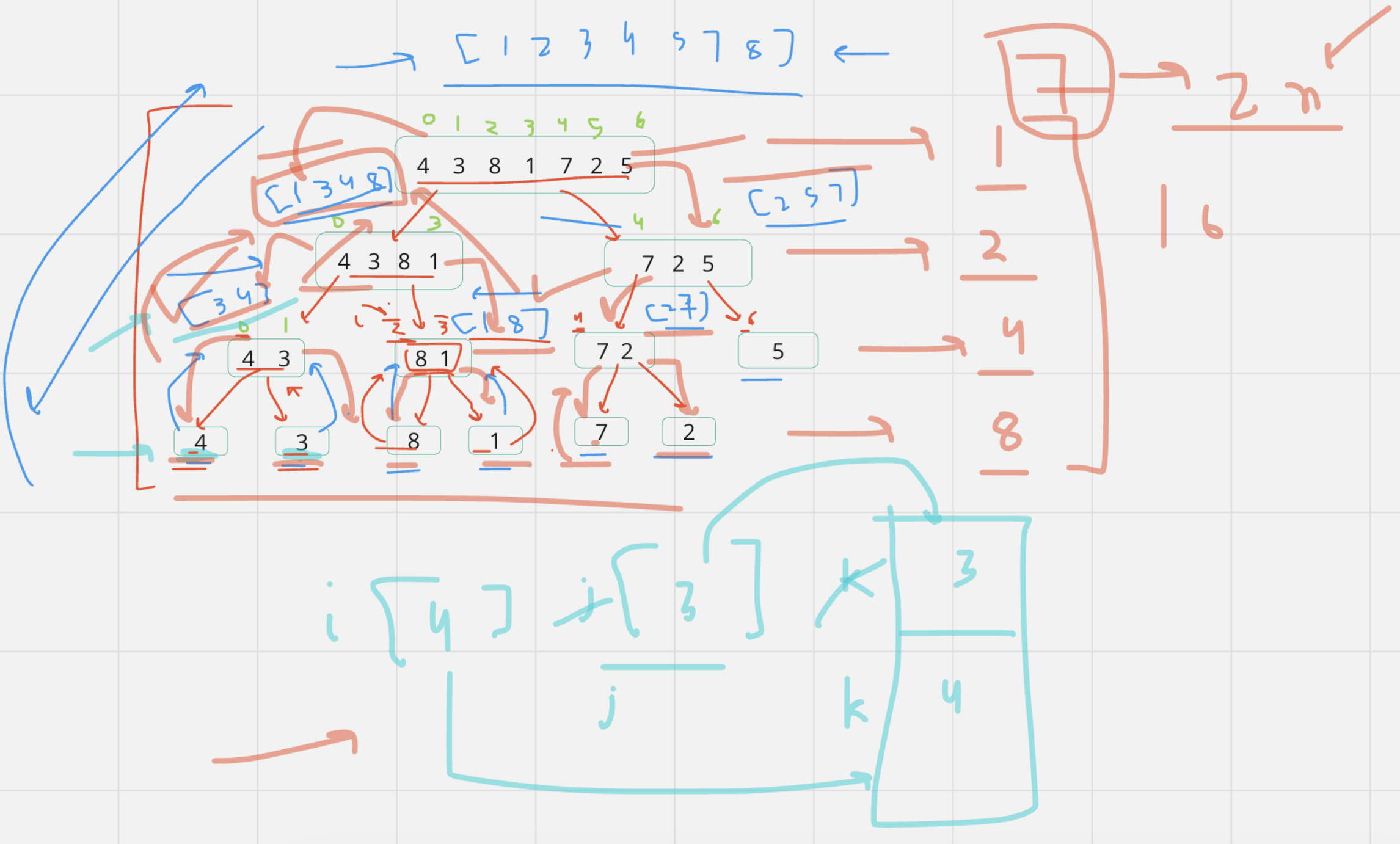

Merge Sort
Break the array
Merge the array
I
II
0 1 2 3 4 5
4 3 8 1 7 2
mid => 2
3 4 8
1 2 7
log n
n
n log n

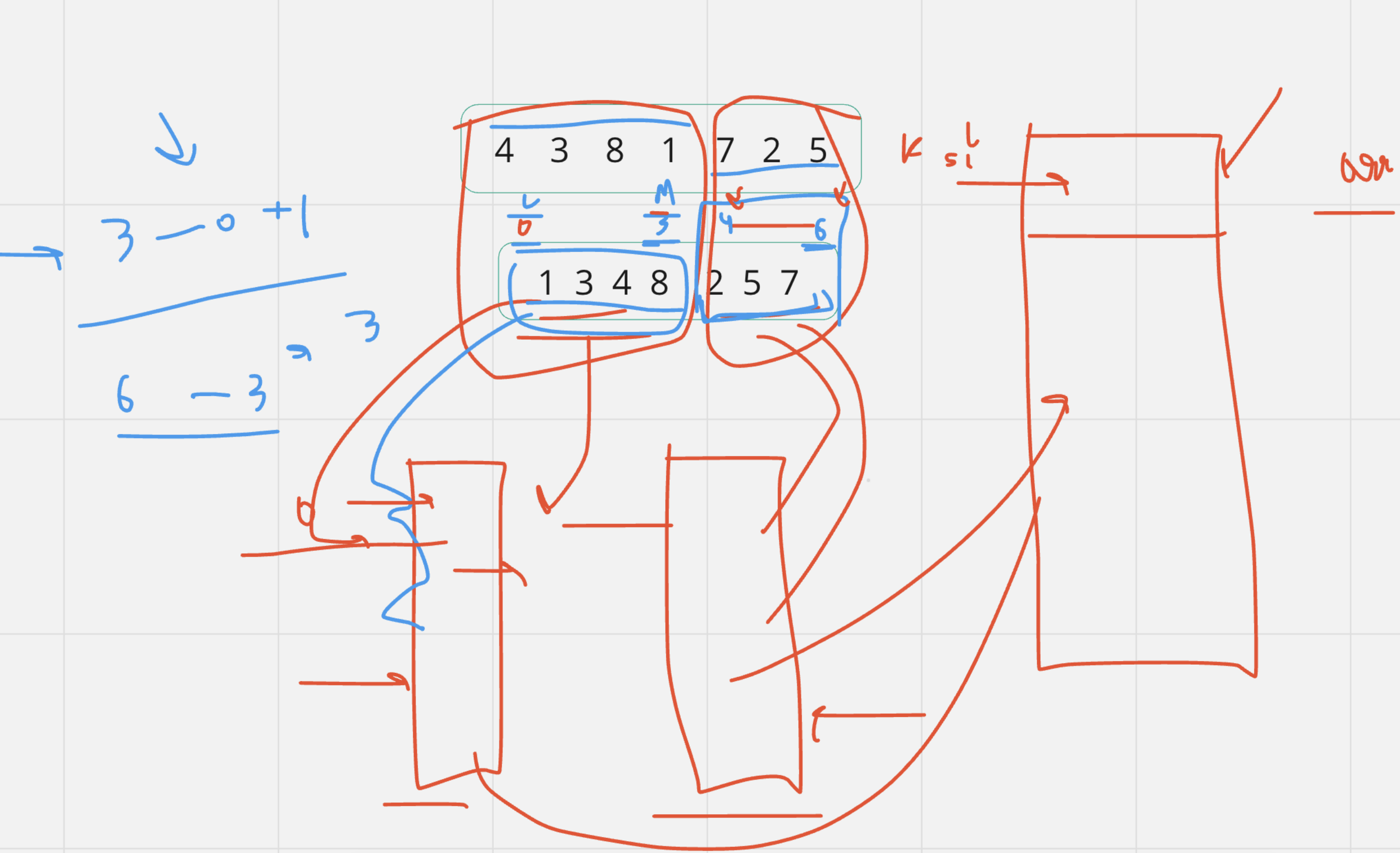

[1 2 3 4 5 7 8]
0 1 2 3 4 5 6
4 3 8 1 7 2 5
[1 3 4 8]
[2 5 7]
4 3 8 1
7 2 5
[3 4]
[7]
4 3
8 1
7 2
5
4
3
8
1
7
2
2n
1
2
4
8
16
i [4] j [3] k
3
4
j
k

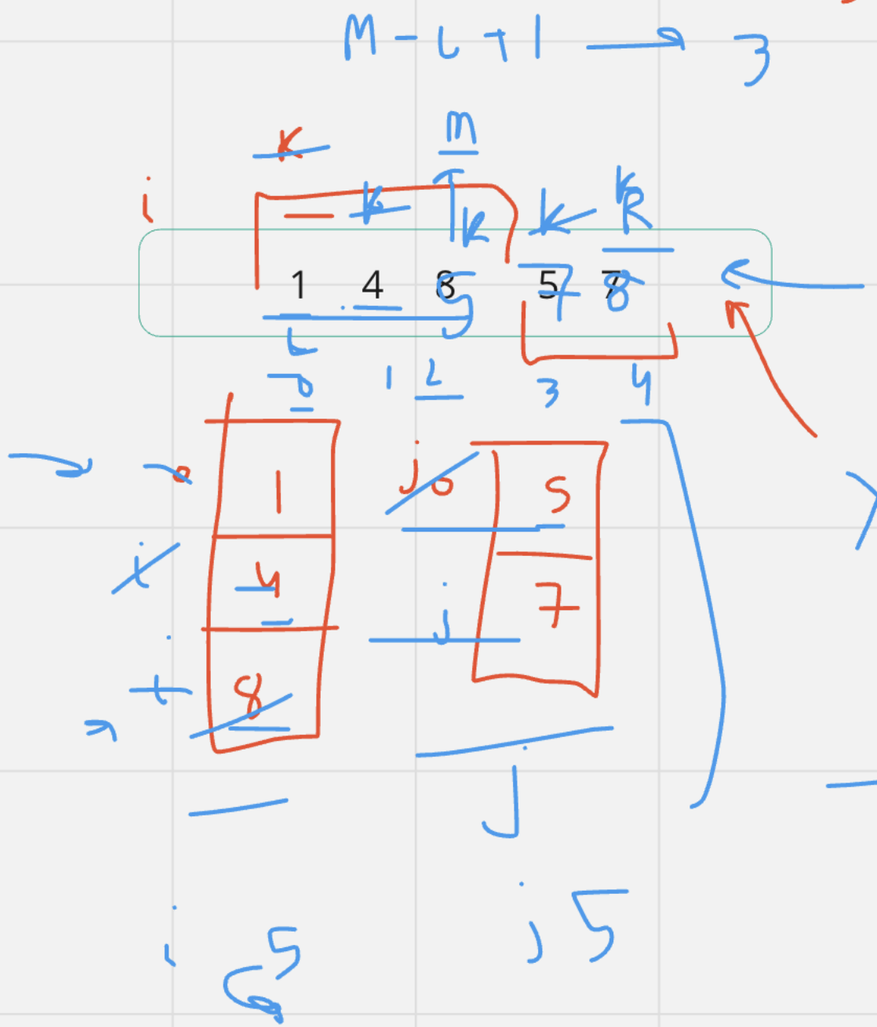
$3 - 0 + 1$

$6 - 3 = 3$

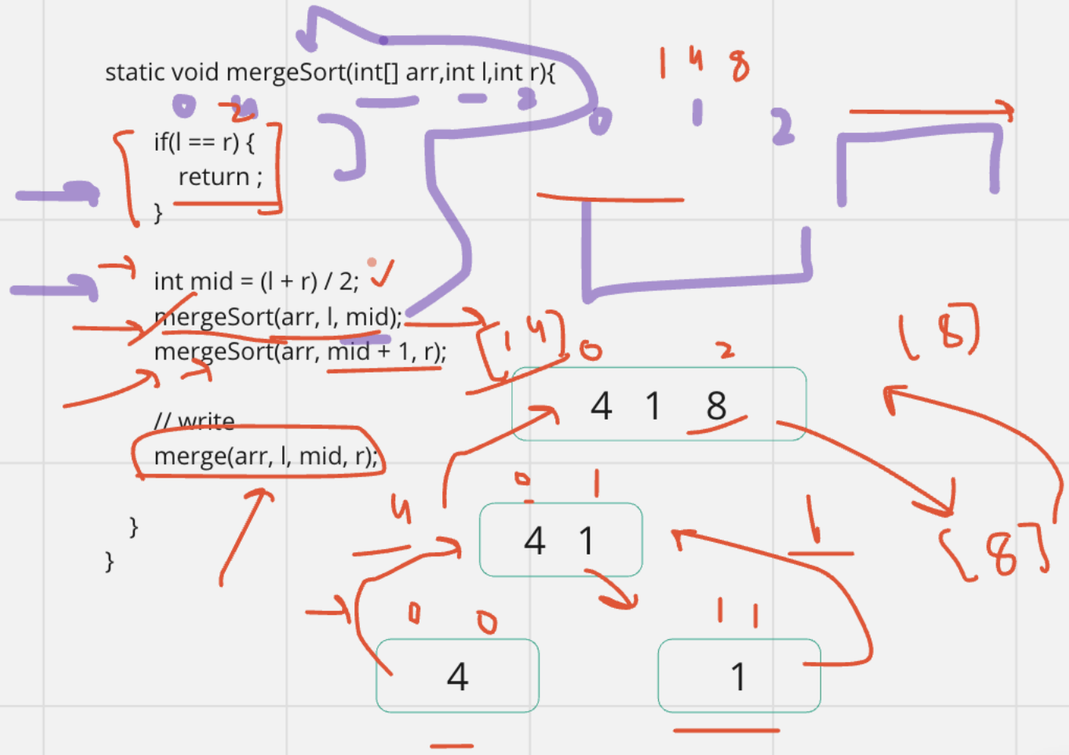
4 3 8 1 7 2 5

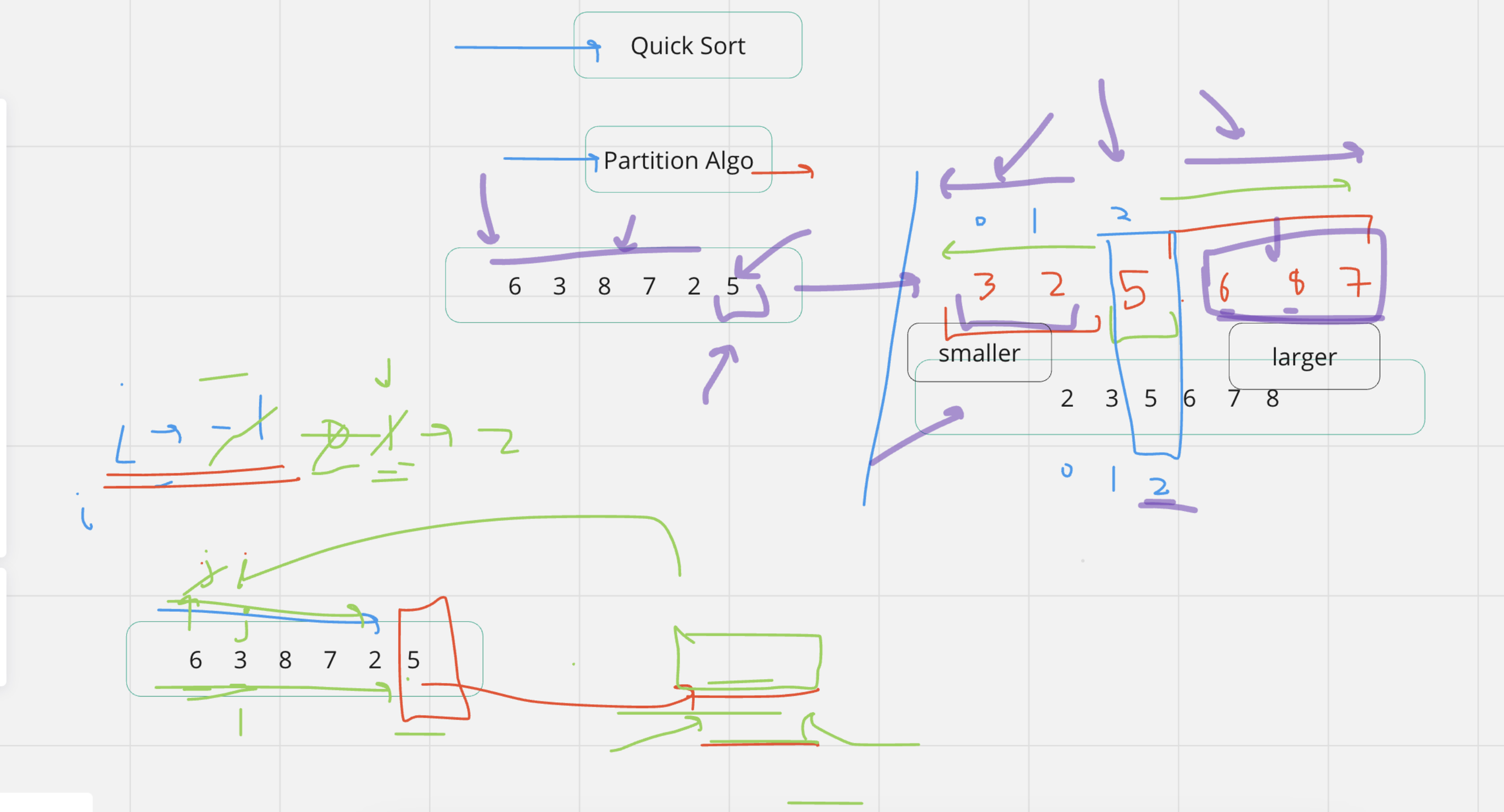
1 3 4 8 2 5 7

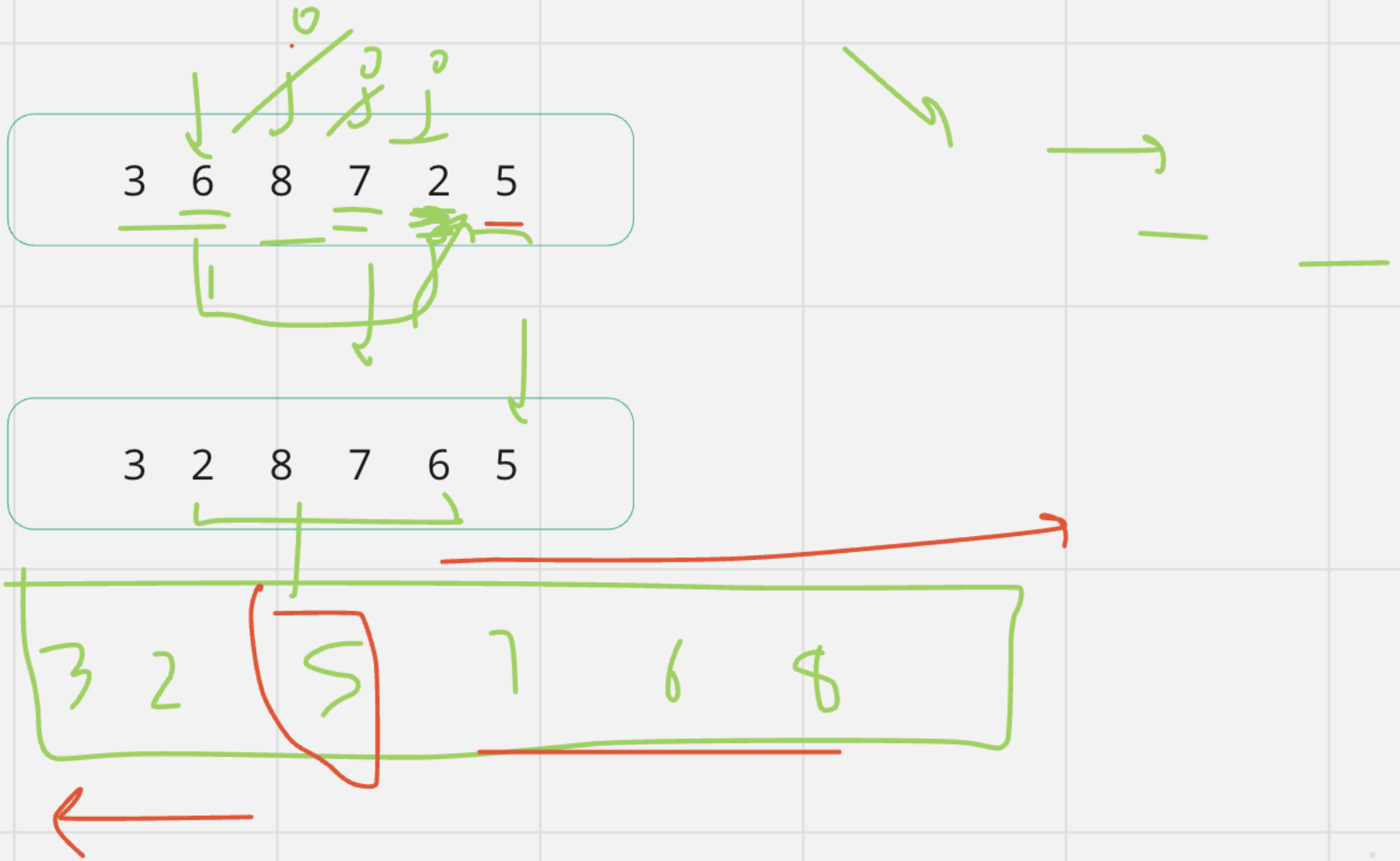


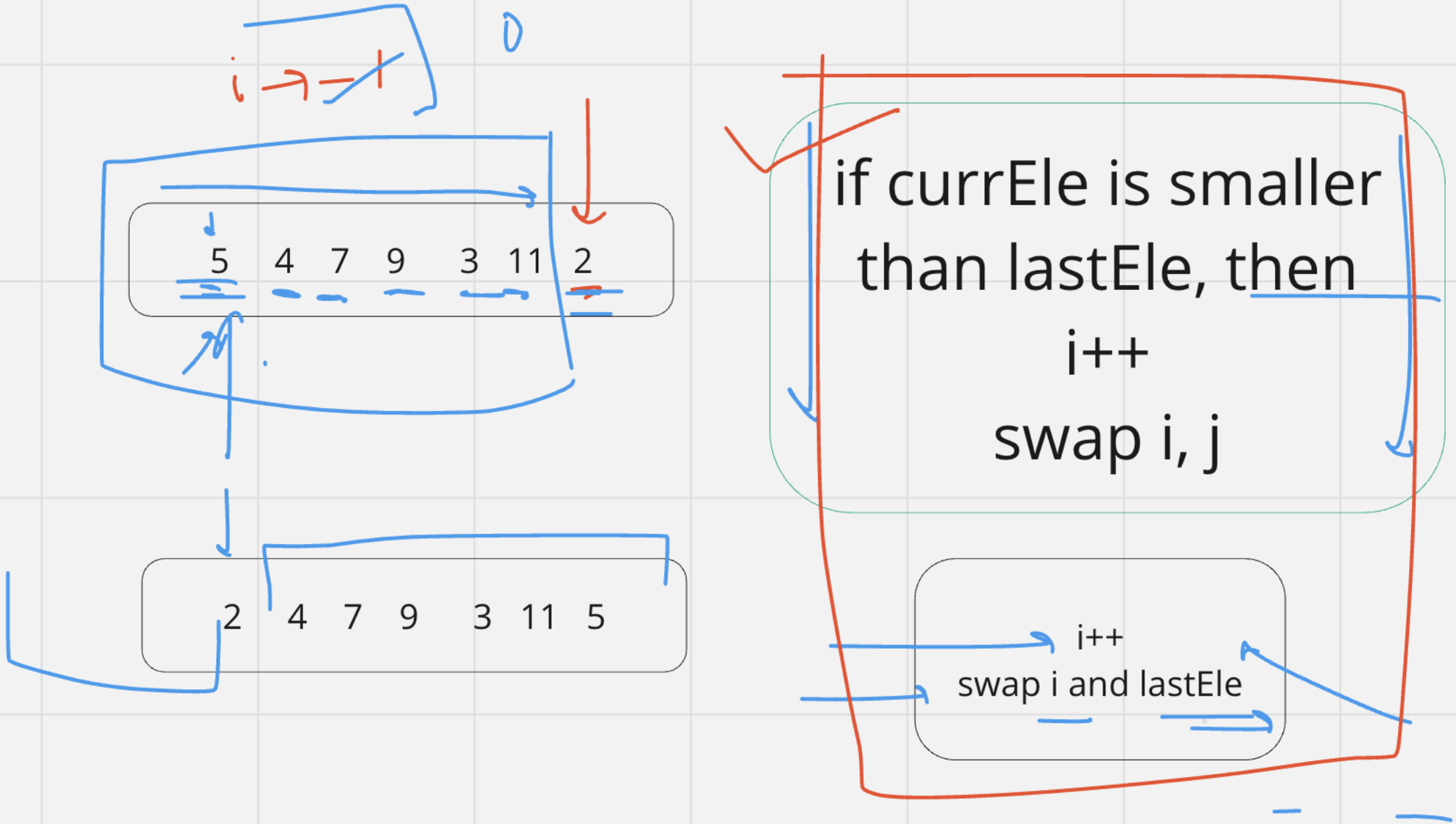



```
static void merge(int arr[], int l, int mid, int r) {

   int leftArr[] = new int[mid - l + 1];
   int rightArr[] = new int[r - mid];
   int k = 0;
   for(int i = l; i <= mid; i++) {
      leftArr[k] = arr[i];
      k++;
   }

   k = 0;
   for(int i = mid + 1; i<= r; i++) {
      rightArr[k] = arr[i];
      k++;
   }

   int i = 0;
   int j = 0;
   k = l;

   while( i < leftArr.length && j < rightArr.length) {

      if(leftArr[i] > rightArr[j]) {
         arr[k] = rightArr[j];
         j++;
         k++;
      }else {
         arr[k] = leftArr[i];
         i++;
         k++;
      }

   }

   while(i < leftArr.length ) {
      arr[k] = leftArr[i];
      i++;
      k++;
   }

   while(j < rightArr.length ) {
      arr[k] = rightArr[j];
      j++;
      k++;
   }

}
```




Quick Sort
Partition Algo
6 3 8 7 2 5
smaller
larger
2 3 5 6 7 8


3 6 8 7 2 5
3 2 8 7 6 5
3 2 5 7 6 8


i
0
5 4 7 9 3 11 2
2 4 7 9 3 11 5
if currEle is smaller than lastEle, then
i++
swap i, j
i++
swap i and lastEle