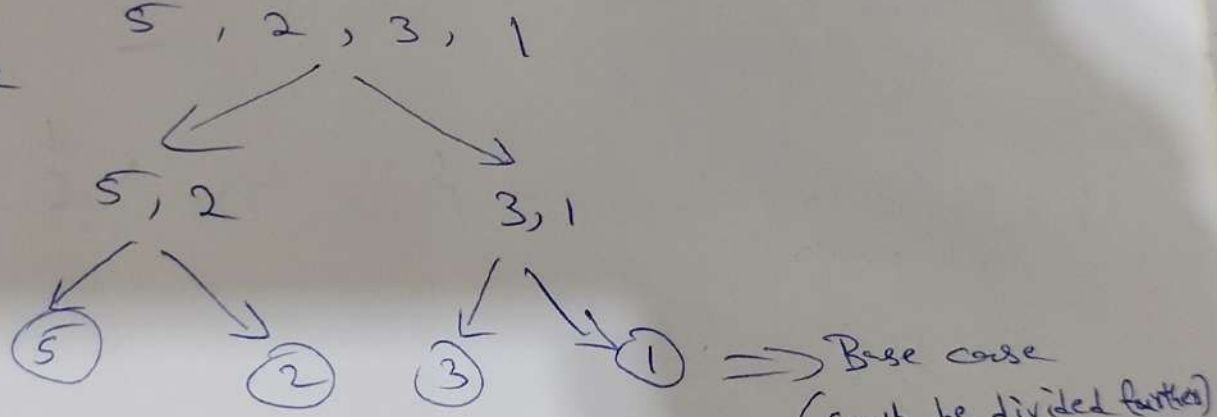
Example: i/p array = [5, 2, 3, 1]

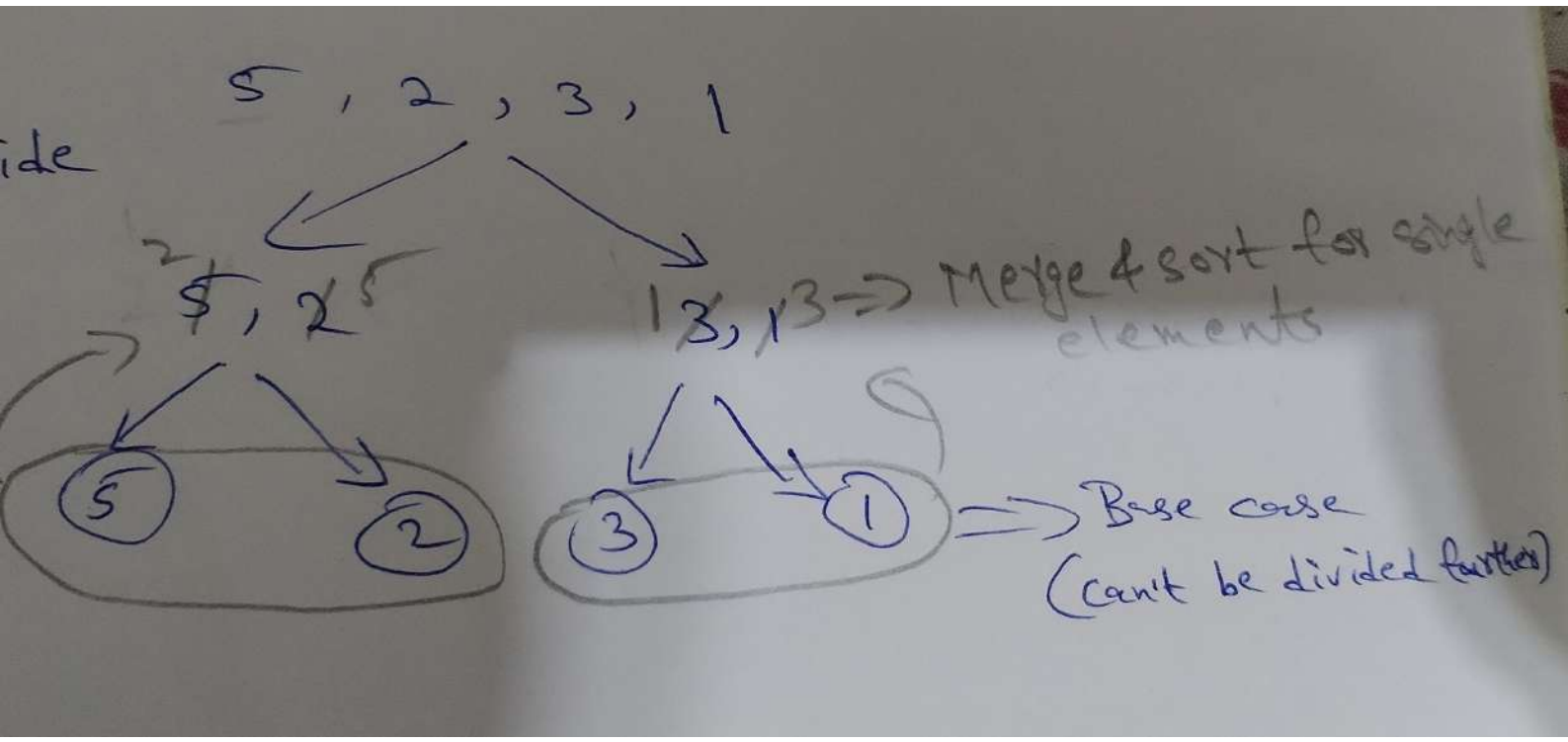
Step 1: Divide

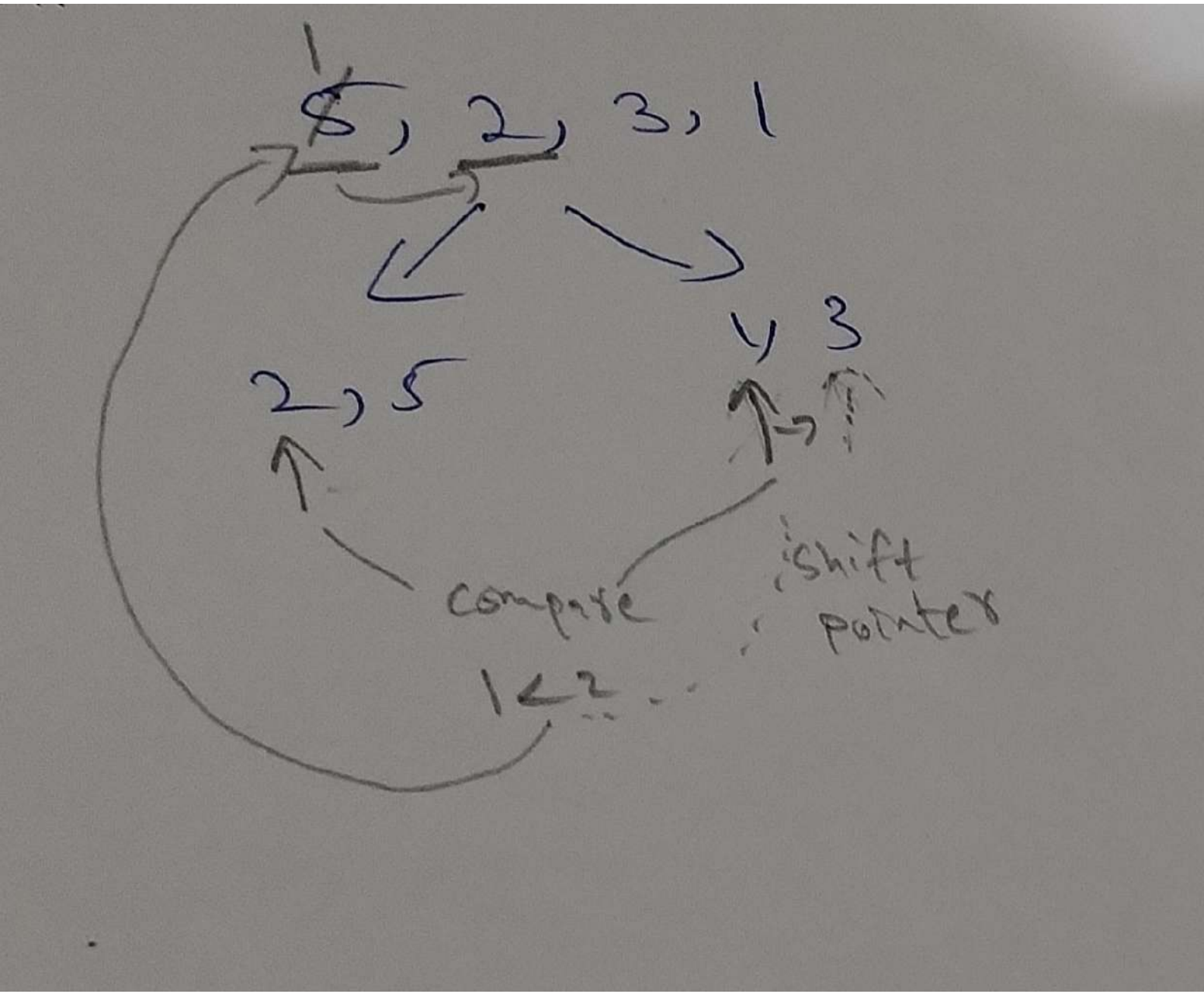


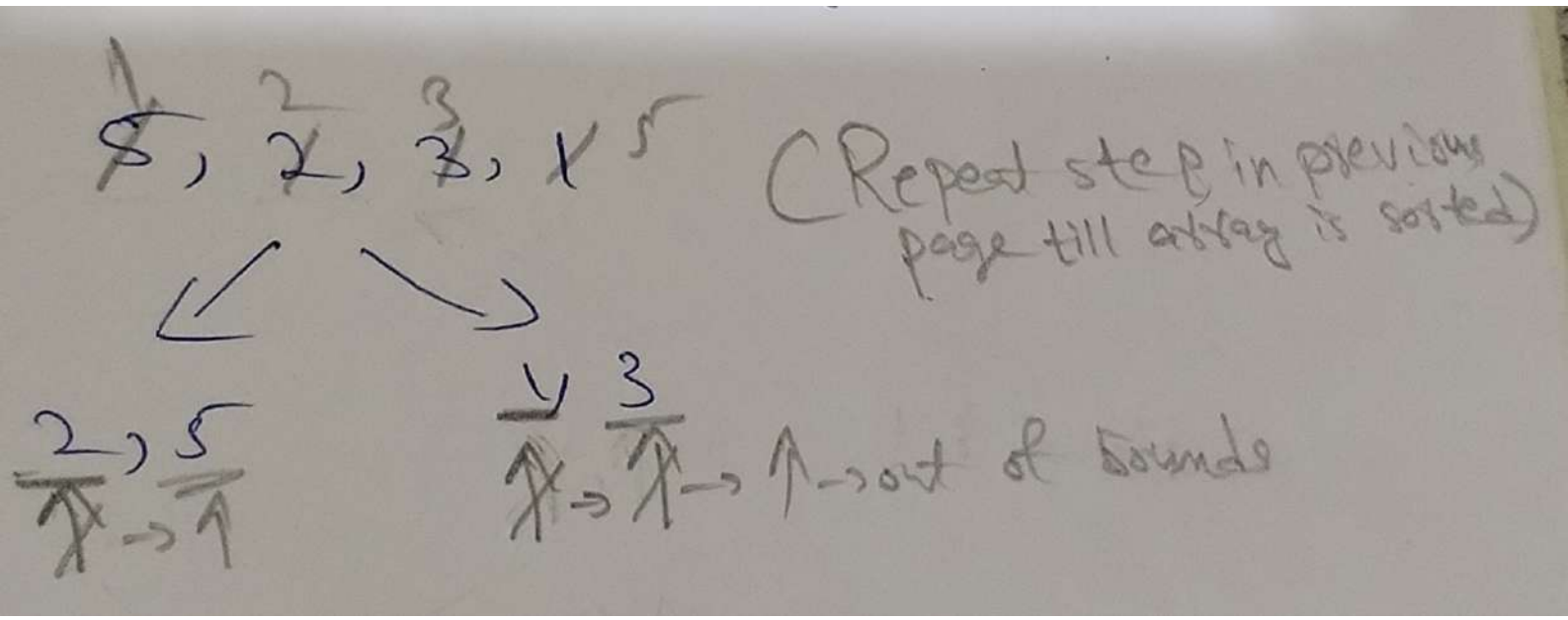
Step 2: Merge

a) When we have to merge two single elements it is easy since we have to just compare those two elements

b) When we have to merge ~~an~~ arrays containing multiple elements we have to allocate temporary arrays for them and the we are going to assign pointers for comparison.


5 , 2 , 3 , 1
5 , 2
3 , 1
Merge & sort for single elements
5
2
3
1
Base case
(can't be divided further)


5, 2, 3, 1
2, 5
4 3
compare
shift pointer
1 < 2 ...

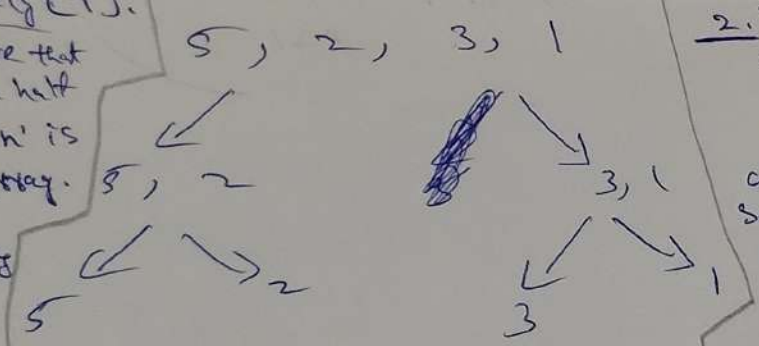

5, 2, 3, 1
(Repeat step in previous page till array is sorted)
2, 5
3
out of bounds

Time complexity (T):

1. Here we can observe that array is being divided in half each step. Assume 'n' is the length of the array.

So 'n' is divided by 2 till value is '1'.

i.e. $\frac{n}{2^x} = 1$



After taking log here we get:

$$\log n = x$$

So, height of the tree formed by dividing is $\log n$.

2. There will be 'n' values at every single level of the tree that we have to merge. So T for each level is $O(n)$.

3. total T: $O(n \log n)$
↓ height

```python
class Solution:
    def sortArray(self, nums: List[int]) -> List[int]:
        def merge(arr, L, M, R):
            left, right =arr[L:M+1], arr[M+1:R+1]
            i, j, k = L, 0, 0
            while j<len(left) and k<len(right):
                if left[j]<=right[k]:
                    arr[i]=left[j]
                    j+=1
                else:
                    arr[i]=right[k]
                    k+=1
                i+=1
            while j<len(left):
                nums[i]=left[j]
                j+=1
                i+=1
            while k<len(right):
                nums[i]=right[k]
                k+=1
                i+=1
        def mergeSort(arr, l ,r):
            if l==r:
                return arr
            m=(l+r)//2
            mergeSort(arr, l, m)
            mergeSort(arr,m+1,r)
            merge(arr,l,m,r)
            return arr
        return mergeSort(nums, 0, len(nums)-1)
```