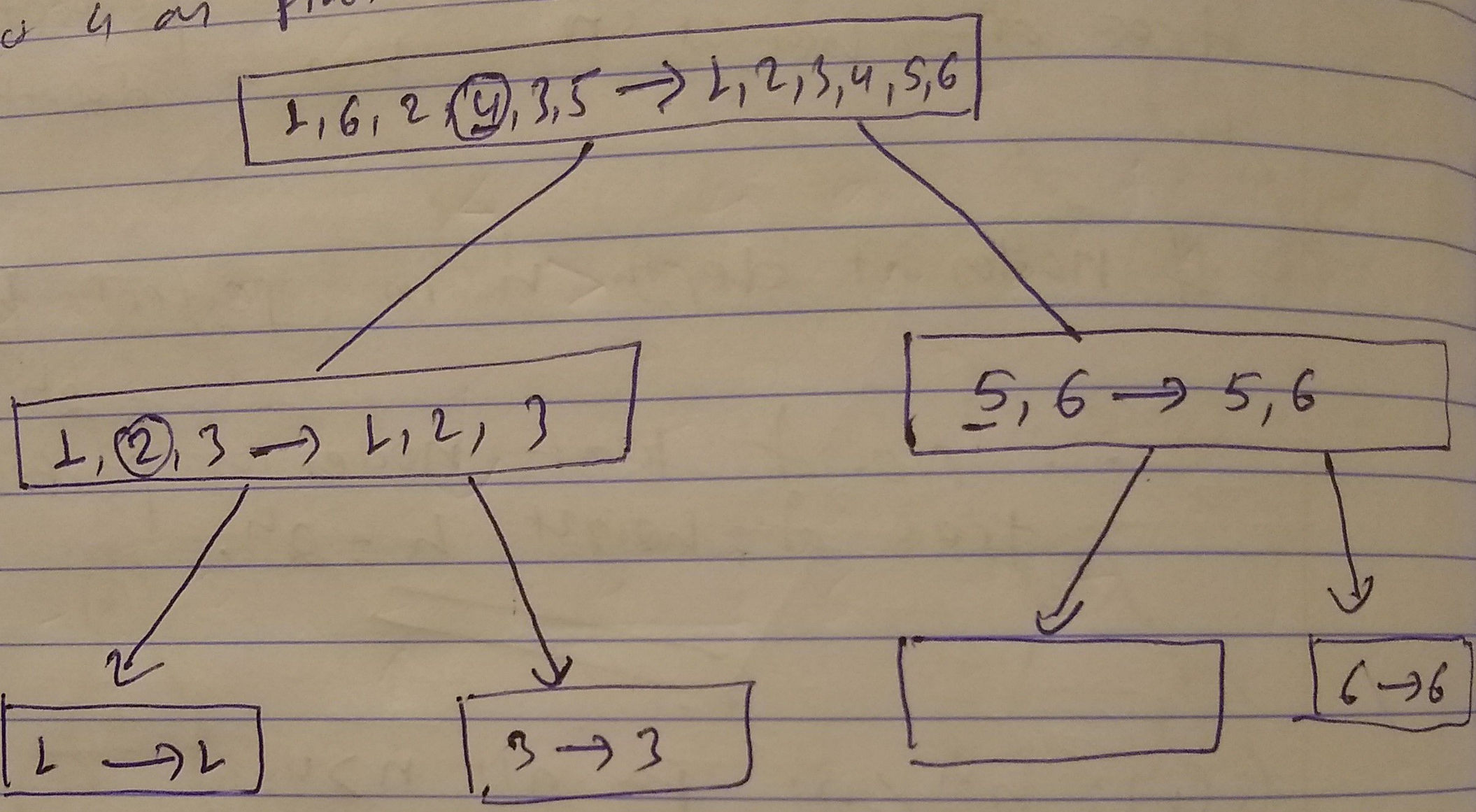
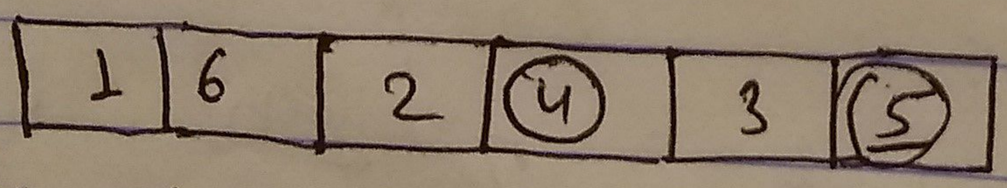
Lab - 5

1. Show all stem of quicksort in sorting the array [1, 6, 2, 4, 3, 5]. use leftmost values as pivots at each step.

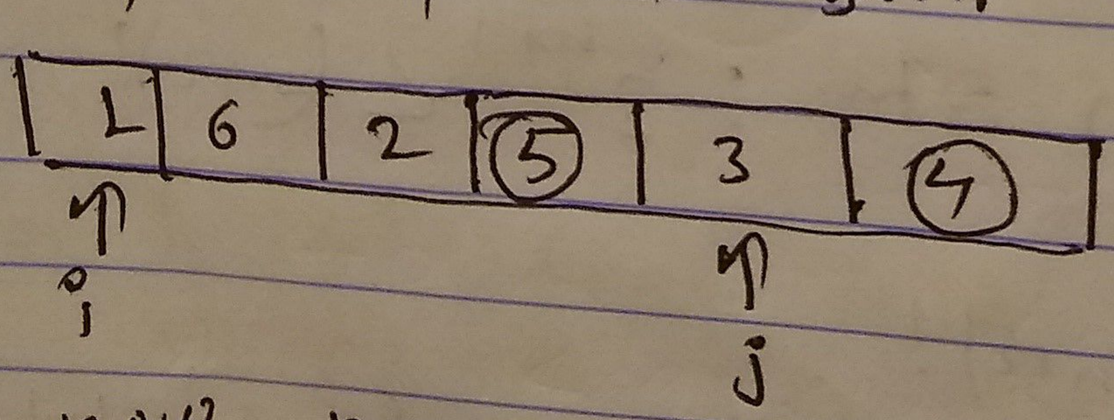
select 4 as pivot



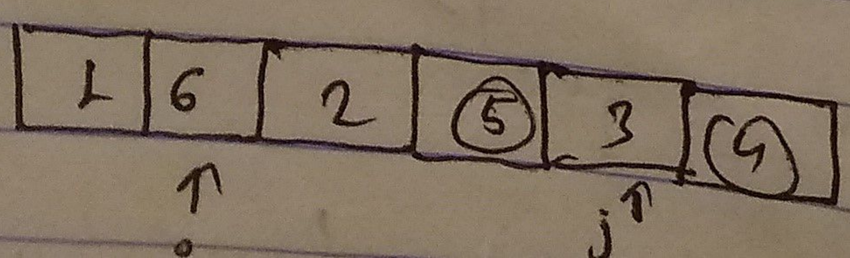
2. Show all steps of In-place QuickSort in sorting the array [1, 6, 2, 4, 3, 5] when doing first partition use leftmost values as pivots.



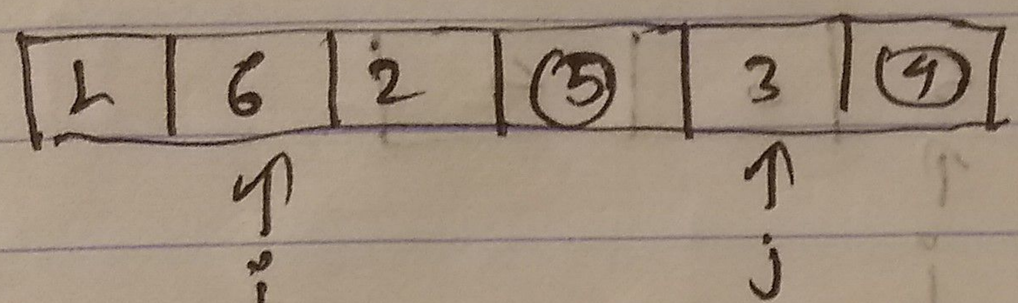
pick pivot = 4 & swap with rightmost element.



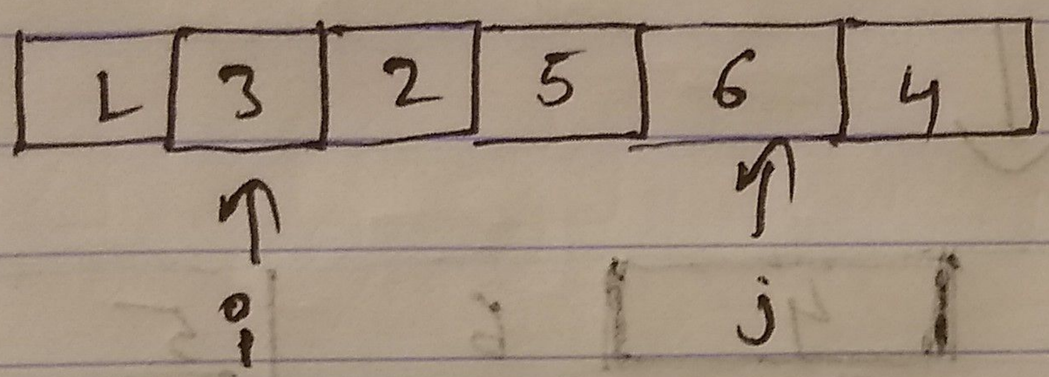
i now moves right as long as it points to the value < 4



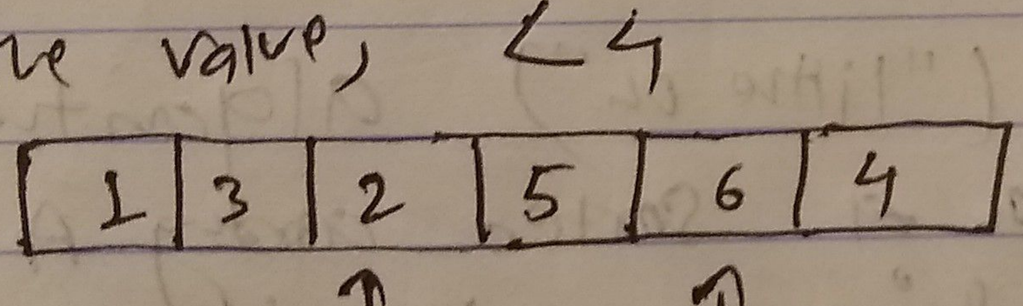
i is stuck at 6, so, move j to the left as long as it points to >4



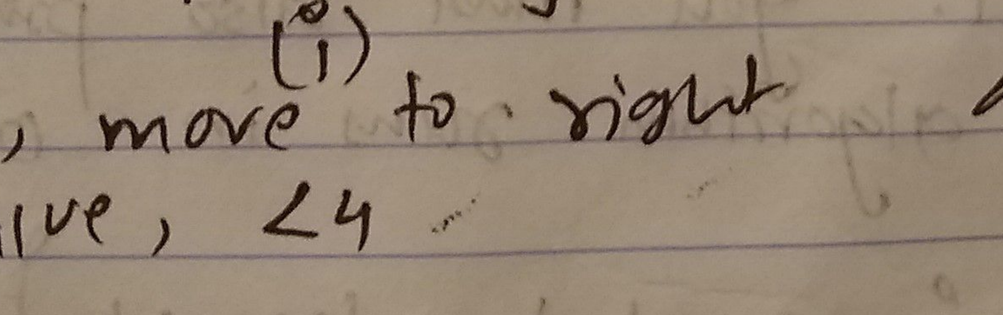
j is now stuck at 3; since i is stuck at 6; so swap 6 & 3.



Now, i is moved to right as long as points to the value, <4

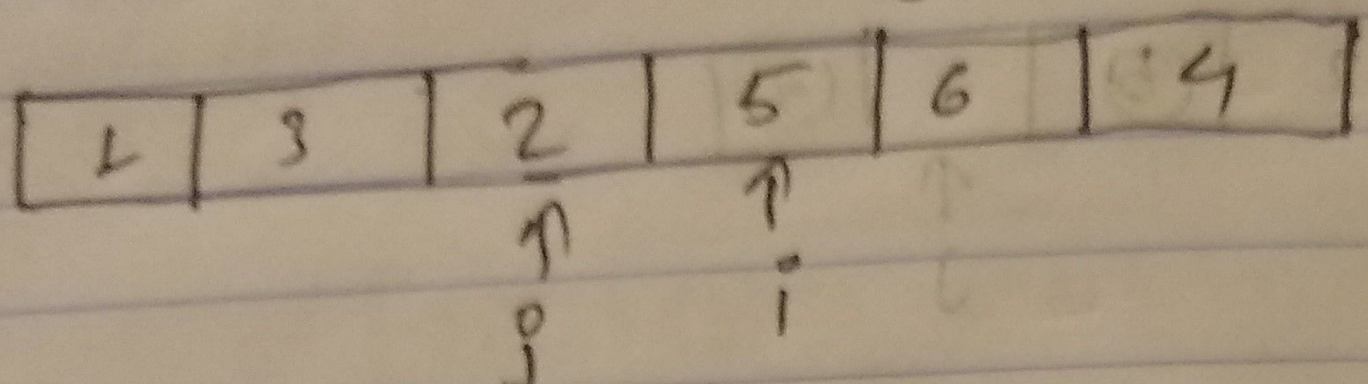


again, move to right as long as points to the value, <4

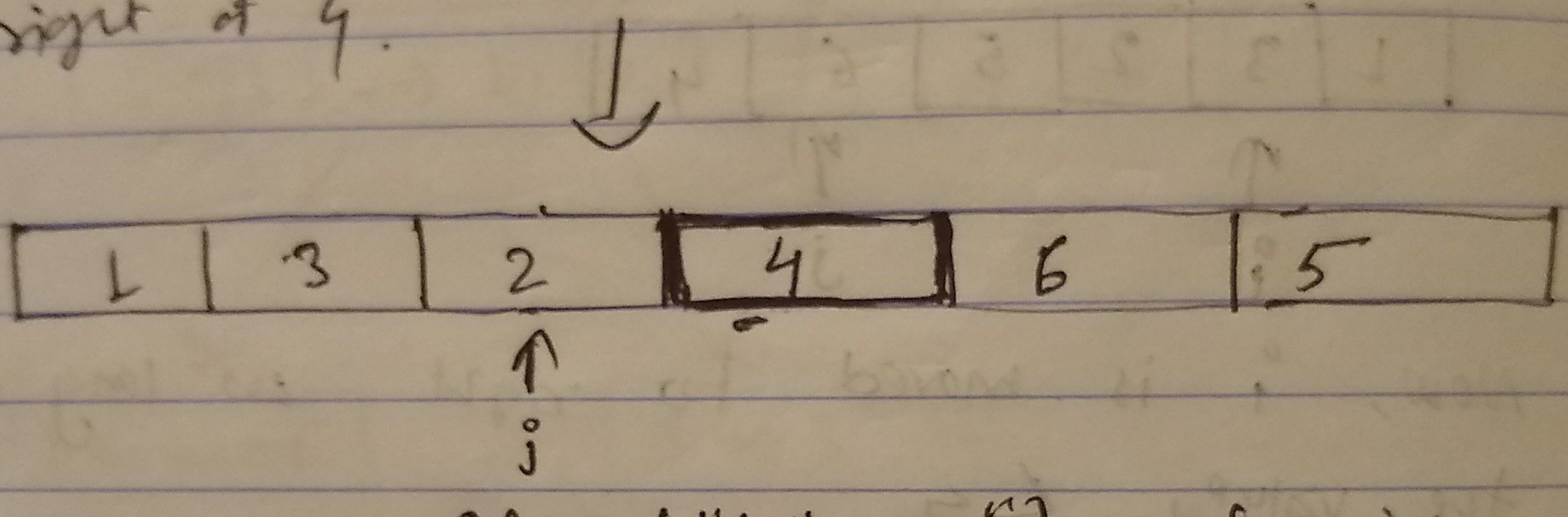


again, i is stuck at 5 so, move j to the left as long as >4

Again, i is stuck at 5; but we can move j left as long as >4.



Now element < pivot 4 are to the left of 4 and element > pivot 4 are to the right of 4.

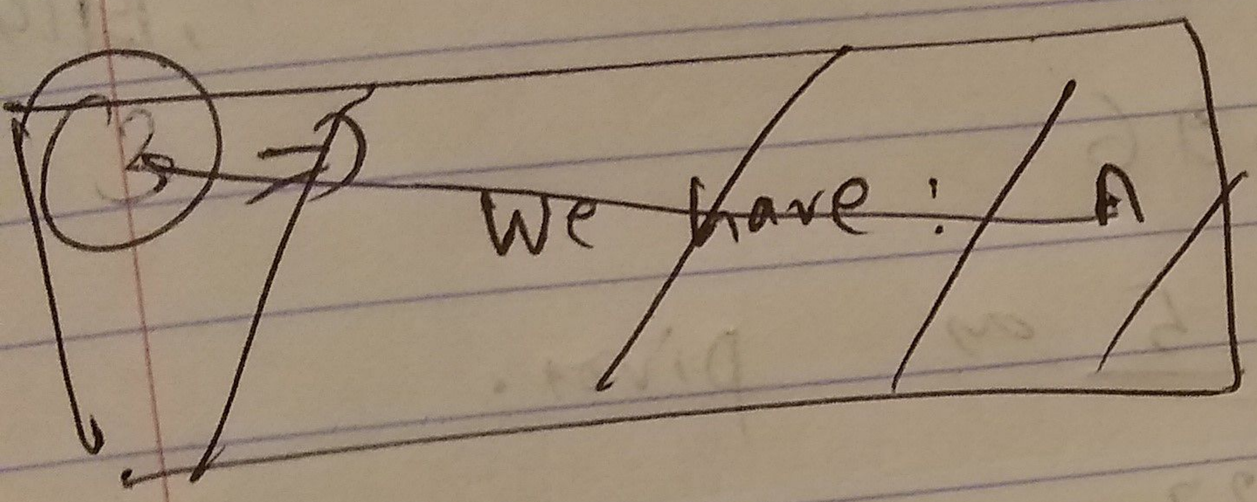


(4) Give an O(n) ("little oh") algorithm for determining Weather a Sorted array A of distinct integers contains an element m for which A[m] = m. you must also provide a proof that your algorithm runs in O(n) time.

Algoritm: Index is equal to element of array.
Input: An array of length n.
output: true or false result.

```
i ← 0                                   ← 1
while i < arr.length do                 ← O(n)
    if i = arr[i]                       ← 1
        return true;                    ← 1
    i ← i+1                             ← 1
return false                            ← 1
```

Above algoritm will run from 0 index to the length of array n. And there is only one for loop. Hence, complexity is $O(n)$.



③ ⇒ we have:

A = [5, 1, 4, 3, 6, 2, 7, 4, 3] which mean

n = 9

a[0] = 5, a[8] = 3

position: 0 ~~1 2 3 4 5 6 7 8~~

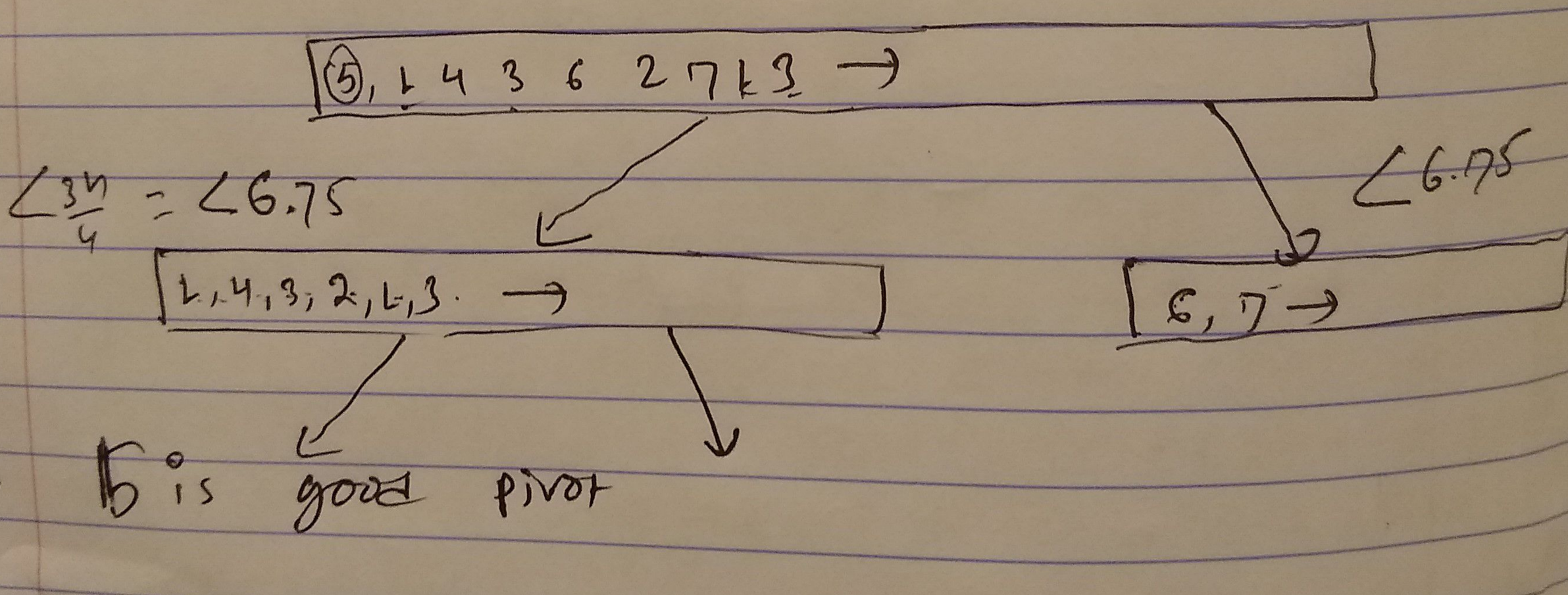
L(8+0)/2 = 4, a[4] = 6

So,

median of 5, 8, 6

for this call use 5 as pivot.

here, $\frac{3n}{4} = \frac{3 \times 9}{4} = \frac{27}{4} = 6.75 = 6.75$

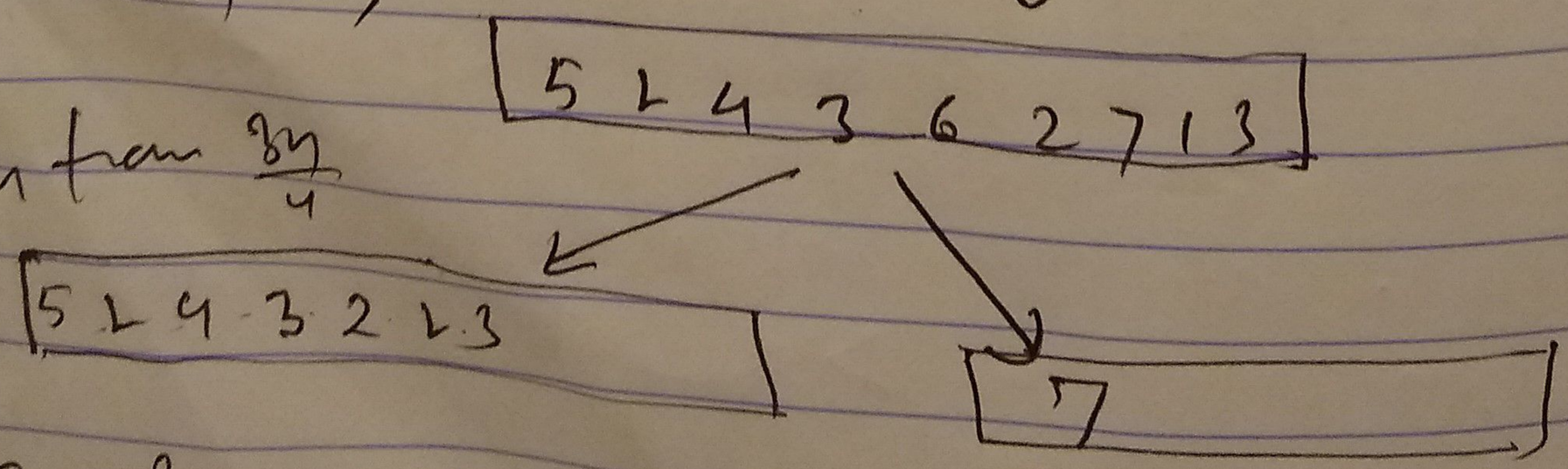


i. 5 is good pivot

again, choose pivot randomly. suppose, pivot = 6

then,

ii. Not less than $\frac{3n}{4}$

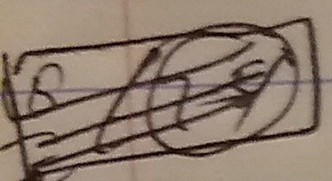


So, it is bad pivot.

(b) Is it true that at least half of the elements of A are good pivots?

Actually, if size of Land G are each less than 3n/4 then it is good self call. If L and G has size greater than or equal to 3n/4 then it is bad call.

- A self call is good with probability at least 1/2.



Given array is not in sorted order, so we can not predict that half of the pivots are good. If it is in order,

1, 1, 2, 3, 3, 4, 5, 6, 7

Then, we can say that, $\frac{9}{2} = 4.5$

1, 1, 2, 3, 3, 4, 5, 6, 7

Still we can not exactly say that half of the pivots are good.

(5) ⇒

Subset sum Algorithm:

Input: Any set with its subset and its sum.
output: return subset ~~with~~ equal to input sum.

```
IF    sum == 0    then
      return subset.
else if   sum < 0  ||  inputSetIsEmpty  then

      return ~~false~~ null

else
      firstElement ⇐ pop first element from set (subset) & its sum.

      return   subsetSum(set, sum)  ||
               subsetSum(set, sum − firstElement)
```