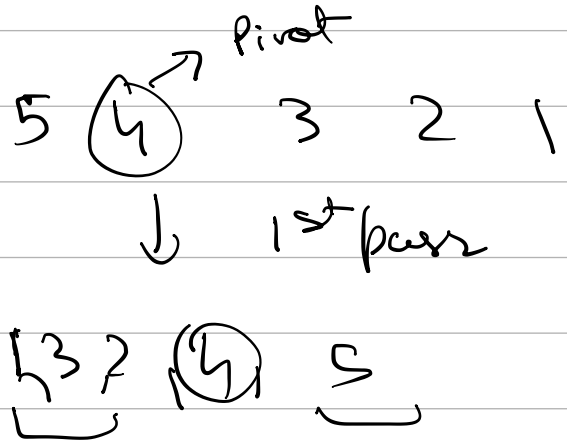
Quick Sort

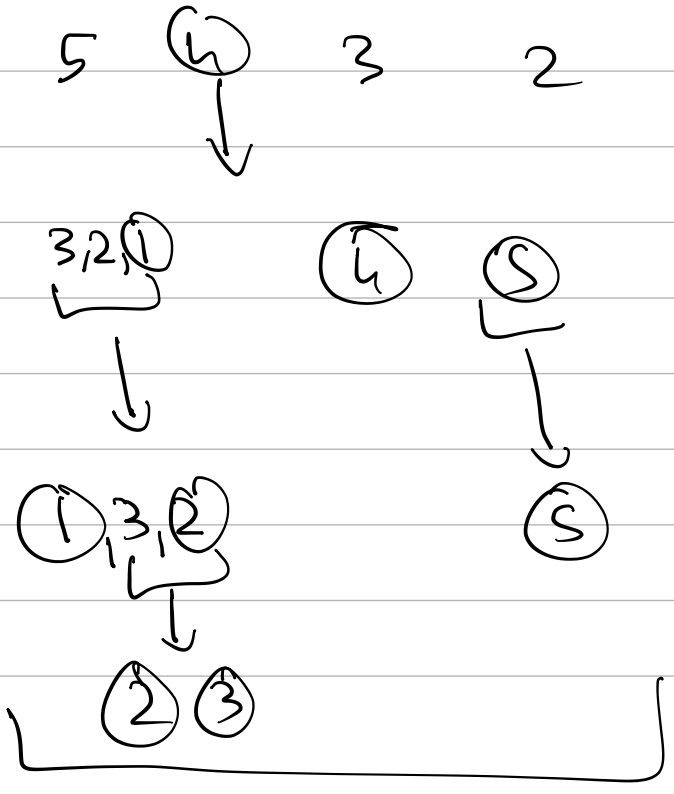
*). PIVOT :- Choose any element → after first pass all the elements < P will be on LHS of P & elements > P will be RHS of P.

Egr



Logic :- In quick sort, after each pass, the pivot element will be put in the correct position.

Diagram :-



Answer:- 1 2 3 4 5

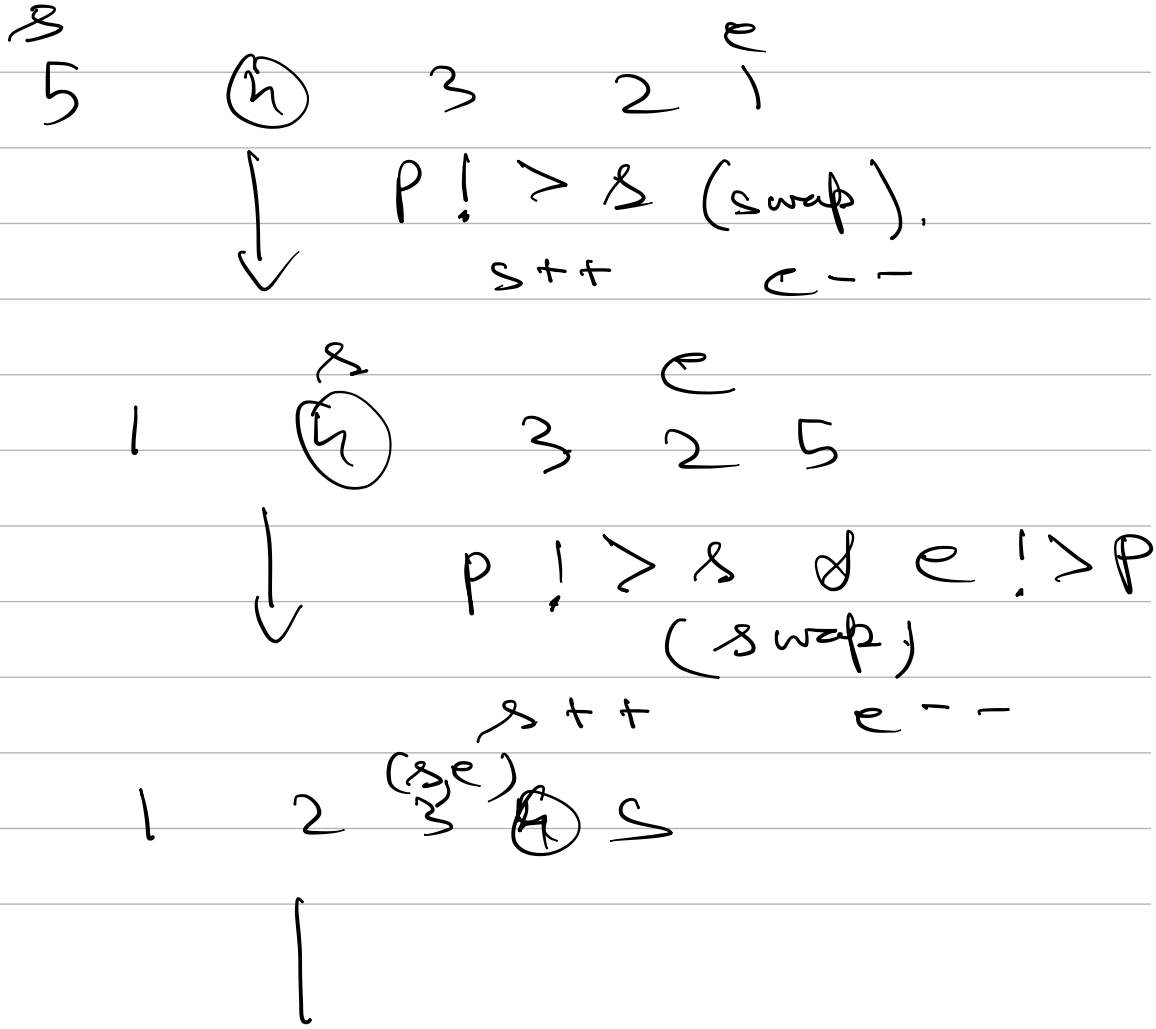
How to put pivot in correct position ?

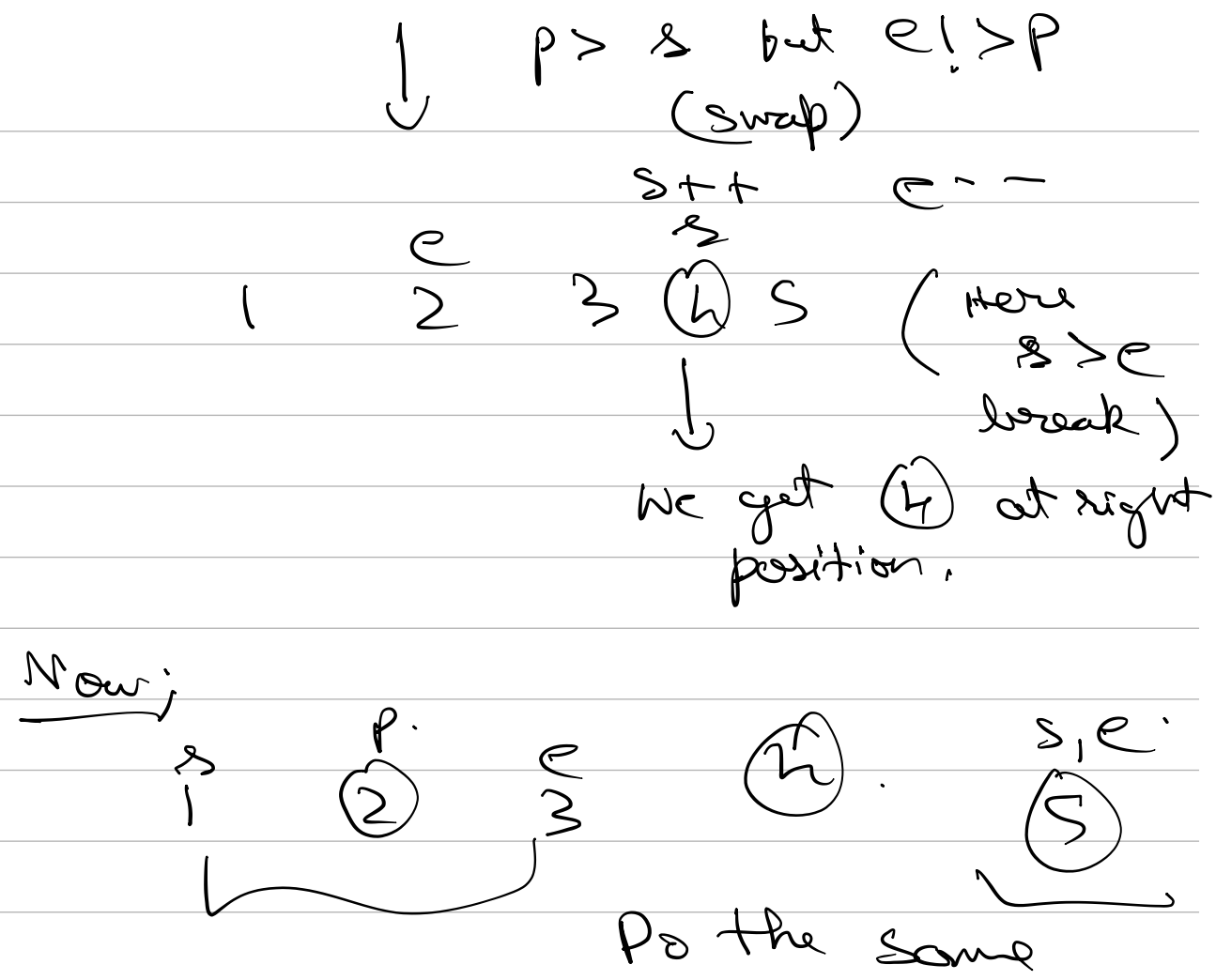
Logic:-

① pivot should be > start. (start++)

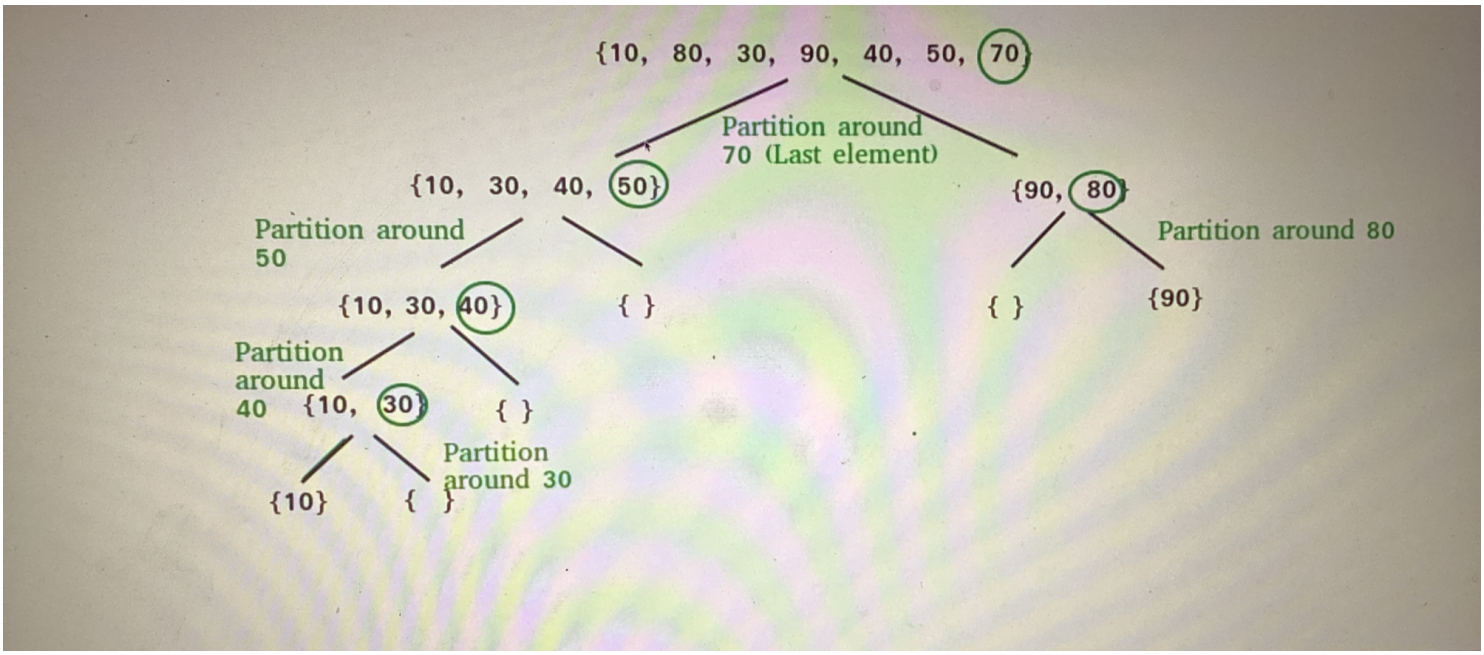
② pivot should be ≤ end. (end--)

③ if any of these conditions are violated. swap start and end and also do start++ and end--.





ANOTHER EXAMPLE:-



NOTE:- Better choose last element as pivot

Different positions the pivot can be ?

→ Random position :- 1 3 2 5 (4) 8

↓ Random.

→ Corner element :- (1) 3 2 5 4 (8)

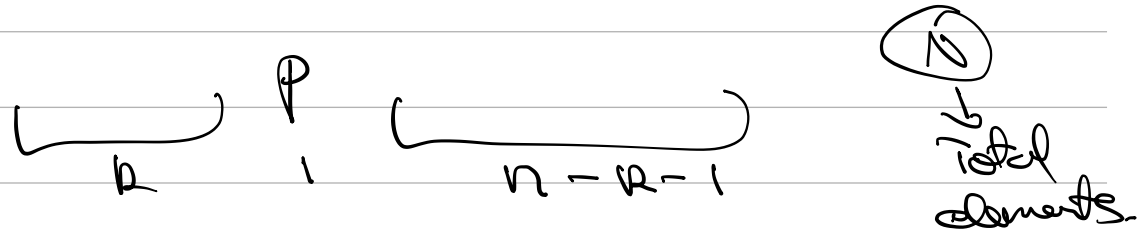
Either of them

→ Pick the middle element :-

1 3 (2) 5 4 8

↓ middle.

Complexity comparison of different positions



$$T(N) = T(K) + T(N-R-1) + O(N)$$

- $T(K)$: Time to sort R elements
- $T(N-R-1)$: Time to sort N-R-1 elements
- $O(N)$: Time to put pivot at right pos.

Recurrence relation of Quick sort:-

Worst case:-

→ When we pick highest or lowest element as pivot.

→ One side will be empty.

$k = 0$

$$T(N) = T(0) + T(N-1) + O(N)$$

$$T(N) = T(N-1) + O(N)$$

Time complexity $= O(N^2)$ → check Time and space complexity for derivation.

Best case:-

$k = \frac{N}{2}$ [we choose middle element]

$$T(N) = T\left(\frac{N}{2}\right) + T\left(\frac{N}{2}\right) + O(N)$$

$$T(N) = 2T\left(\frac{N}{2}\right) + O(N)$$

↳ $g(x)$

Akhtar Baazi-

$$f(x) = x + x \int_1^x \frac{u}{u^{p+1}}$$

To find P, $\quad 2 \times \frac{1}{2^p} = 1$

$$\boxed{p = 1}$$

$$f(x) = x + x \int_1^x \frac{u}{u^2}$$

$$= x + x \int_1^x \frac{1}{u}$$

$$= x + x \int_1^x \log u$$

$$= x + x (\log x - \log 1)$$

$$\boxed{f(x) = x + x \log x} \rightarrow \text{ignore } x,$$

$$\boxed{f(x) = x \log x}$$

$$\boxed{\text{Time complexity} = f(N) = O(N \log N)}$$

NOTES :-

*). Not stable (In case of equal elements their relative positions are not in same order as of before sorting)

*) In-place algorithm (This is better than Merge sort as. merge sort take O(N) auxiliary space but quick sort is in place algorithm).

*). Merge sort is better used in linked list, as unlike array, linked list does not have continous memory allocation.

Hybrid sorting algorithms

We use combination of sorting algorithms to get better performance.

For example :- Merge sort + Insertion sort

This work well for

partially sorted data.

Code :-

```java
public class QuickSort {
    public static void main(String[] args) {
        int[] arr = {5, 4, 3, 2, 1};
        sort(arr, low: 0, hi: arr.length - 1);
        System.out.println();
    }
```

```java
    static void sort(int[] nums, int low, int hi) {
        if (low >= hi) {
            return;
        }

        int s = low;
        int e = hi;
        int m = s + (e - s) / 2;
        int pivot = nums[m];

        while (s <= e) {
            while (nums[s] < pivot) {
                s++;
            }
            while (nums[e] > pivot) {
                e--;
            }
```

```java
            if (s <= e) {
                int temp = nums[s];
                nums[s] = nums[e];
                nums[e] = temp;
                s++;
                e--;
            }
        }

        // now my pivot is at correct index, please sort two halves now
        sort(nums, low, e);
        sort(nums, s, hi);
    }
}
```

→ again checking inside while loop as s++ and e-- is also done above.

} → while loop closed

→ passes low to e

→ s to hi

low e s high.

1 2 (3) 4 5

After putting pivot (i.e 3) in correct position, e-- and s++ is done, so. passing low to e and s to high will

pass two subarrays.

Internal sorting Algorithms

-> Each language uses its own sorting algorithm.

-> We can call this sorting function.

-> In Java, Arrays.sort(arr) function sorts 'arr' based on Dual pivot quick sort.

-> Dual pivot is more efficient than single pivot we done above.

-> Dual pivot internal implementation has hybrid algorithm (i.e combination of other algorithms like mergesort, etc).

-> Each language uses its own algorithm which we can use by calling the function.