20/10/2023

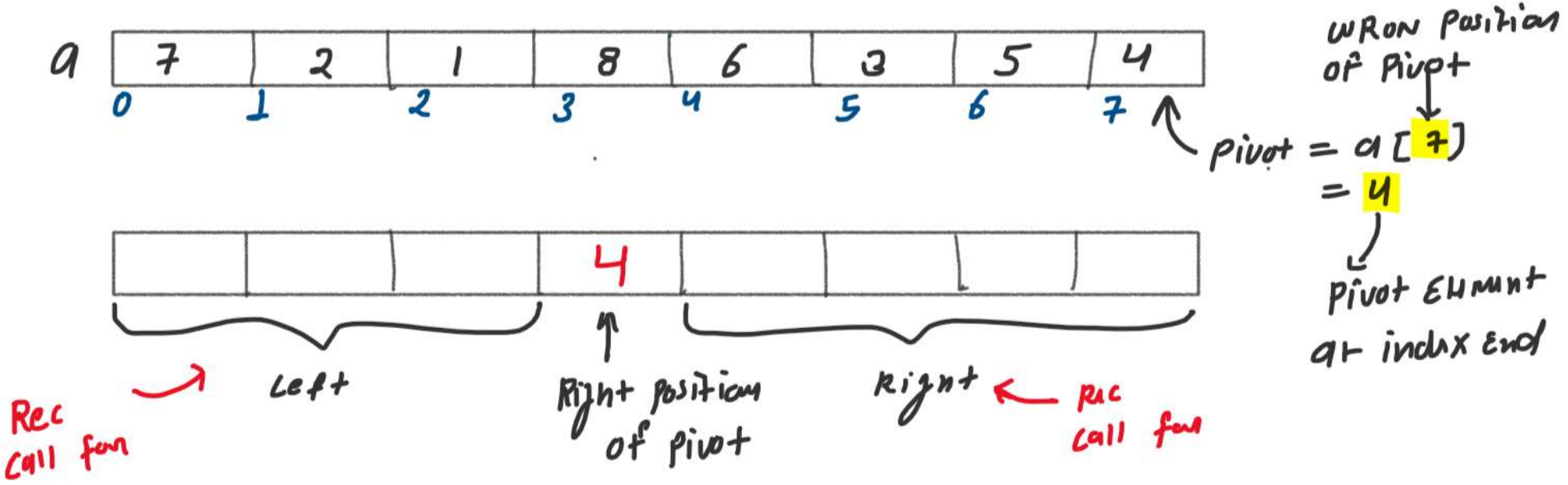
QUICK SORT ALGORITHM

QUICK SORT ALGORITHM

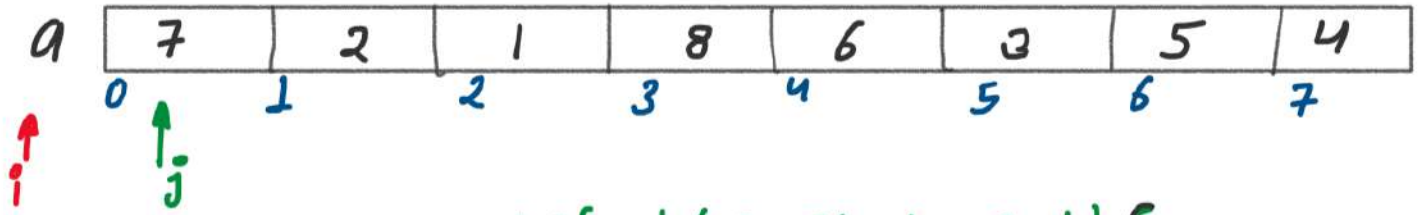
New way to partitioning of array (Ek Step Chalna Mujhe Aata hai)

Step 01: *Find pivot (It is always end element of array)*
- *To place pivot such that the element to the* ***right of pivot > a[pivot]***
- *To place pivot such that the element to the* ***left of pivot < a[pivot]***

Step 02: *Now apply recursion for left and right part of pivot (Baki ka recusion sambhal Lega)*



STEP:01



start = 0

End = 7

pivot = end

i = start - 1

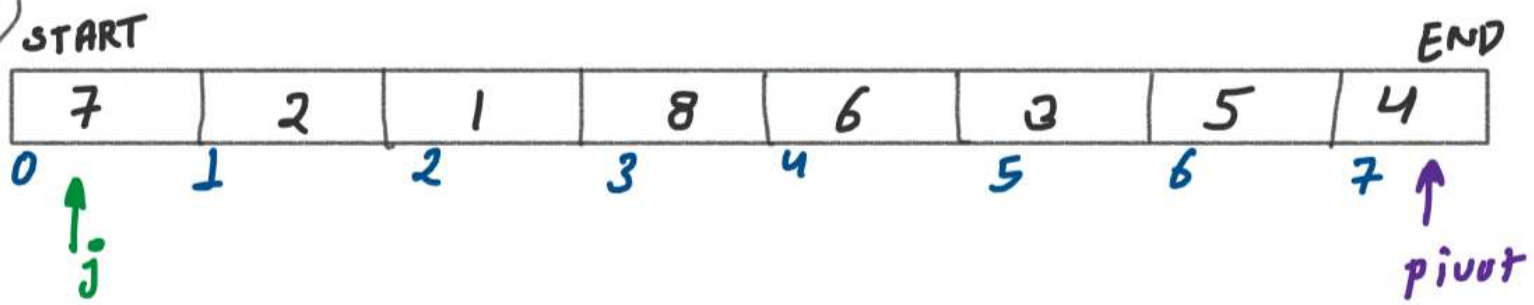
j = start

```
quickSort ( a , start , end ) {
    while ( j < pivot) {
        if ( a[j] < a[pivot] ) {
            i++;
            swap( a[j], a[i]) ;
        }
        j++;
    }
    i++;
    swap( a[i], a[pivot]) ;
}
```

MUJHE **First STEP** Chalna AATA Hai jaha par me 4 ko uski **Right Position** par RAKhana jata HU.

DRY RUN FOR

STEP 1



①

start = 0

End = 7

pivot = End

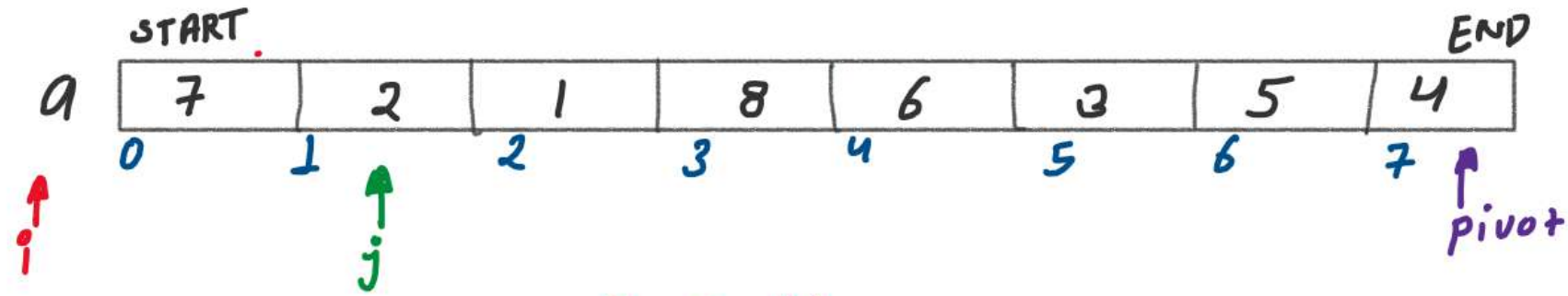
i = start - 1

j = ~~start~~ 1

a[j] > a[pivot]

7 > 4

j++



(2)

start = 0

End = 7

pivot = End

i = ~~start~~ -1 0

j = ~~1~~ 2

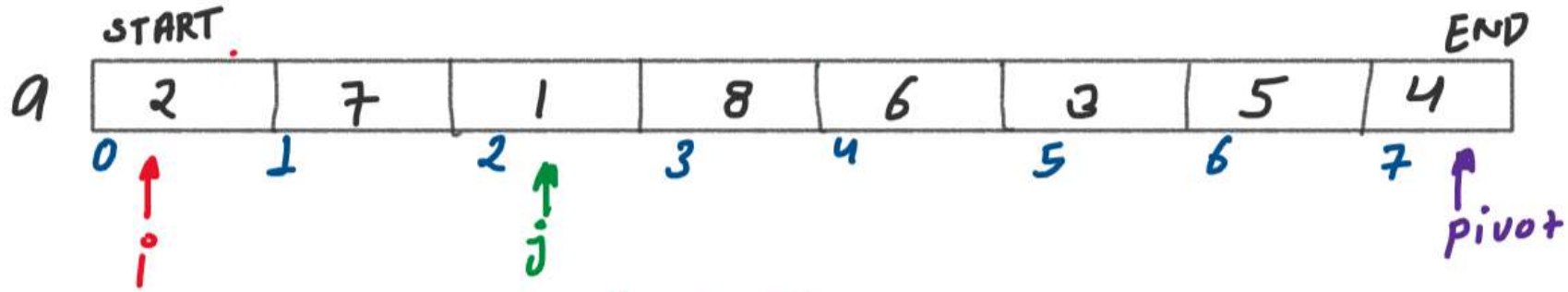
2 < 4

a[j] < a[pivot]

i++

swap(a[i] , a[j])
7 2

j++



③

start = 0

end = 7

pivot = end

i = ~~0~~ 1

j = ~~7~~ 3

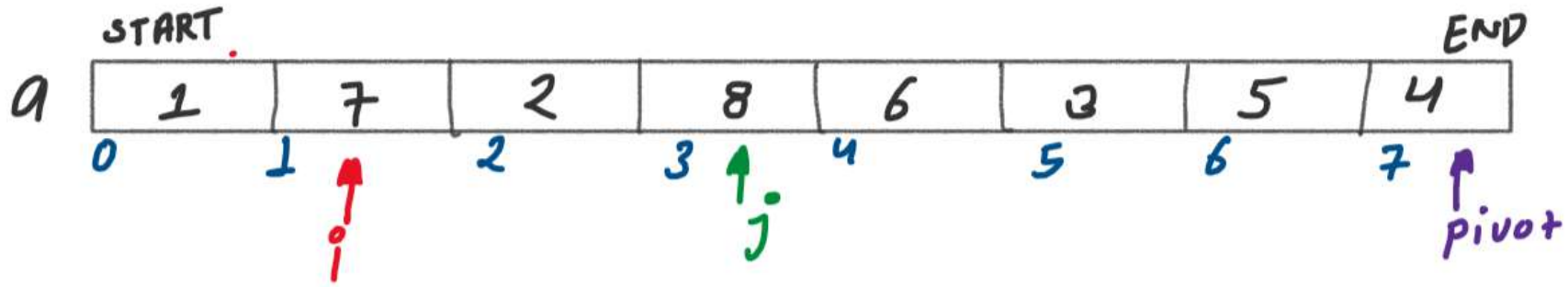
1 < 4

a[j] < a[pivot]

i++

swap(a[i], a[j])
2 1

j++



(4)

start = 0

end = 7

pivot = end

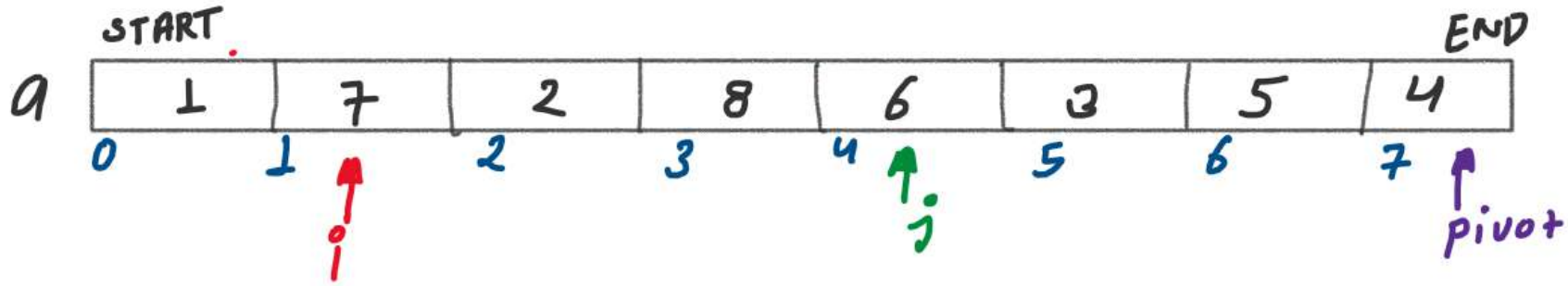
i = 1

j = ~~3~~ 4

a[j] > a[pivot]

8 > 4

j++



(5)

start = 0

End = 7

pivot = End

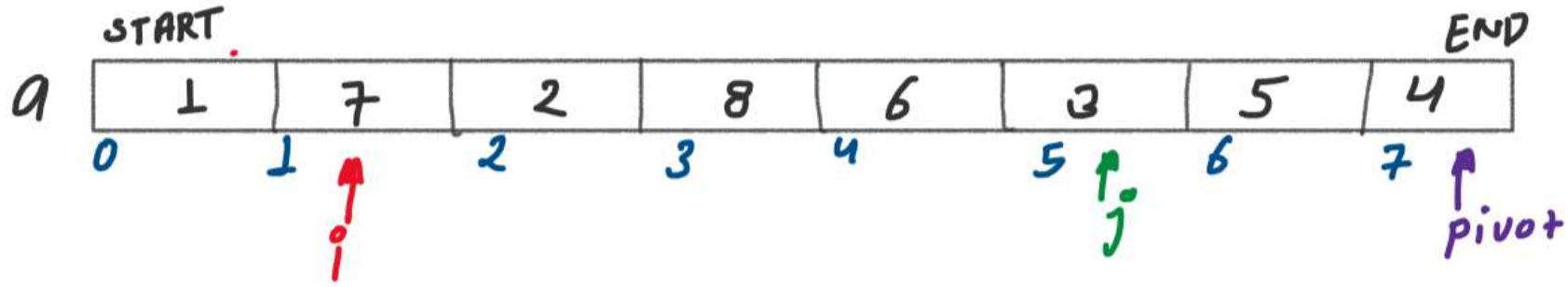
i = 1

j = ~~4~~ 5

a[j] > a[pivot]

6 > 4

j++



(6)

start = 0

End = 7

pivot = End

i = ~~1~~ 2

j = ~~5~~ 6

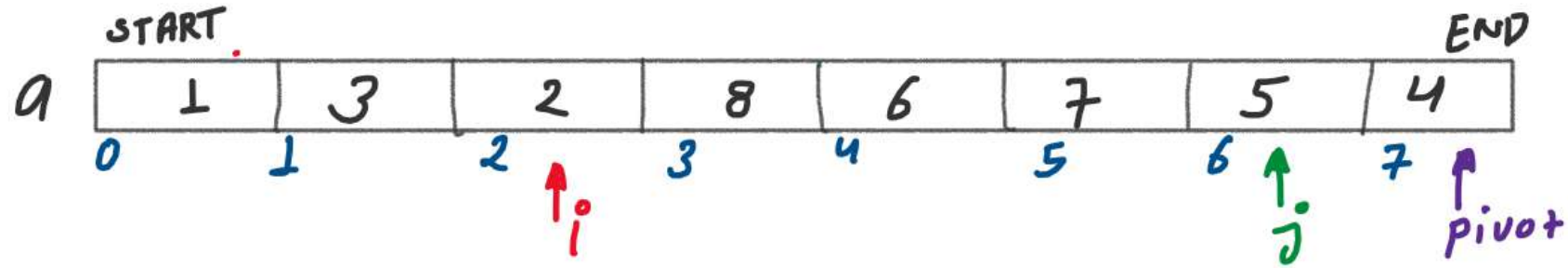
a[j] < a[pivot]

3 < 4

i++

swap(a[i] , a[j])

j++ 7 3



(7)

start = 0

end = 7

pivot = end

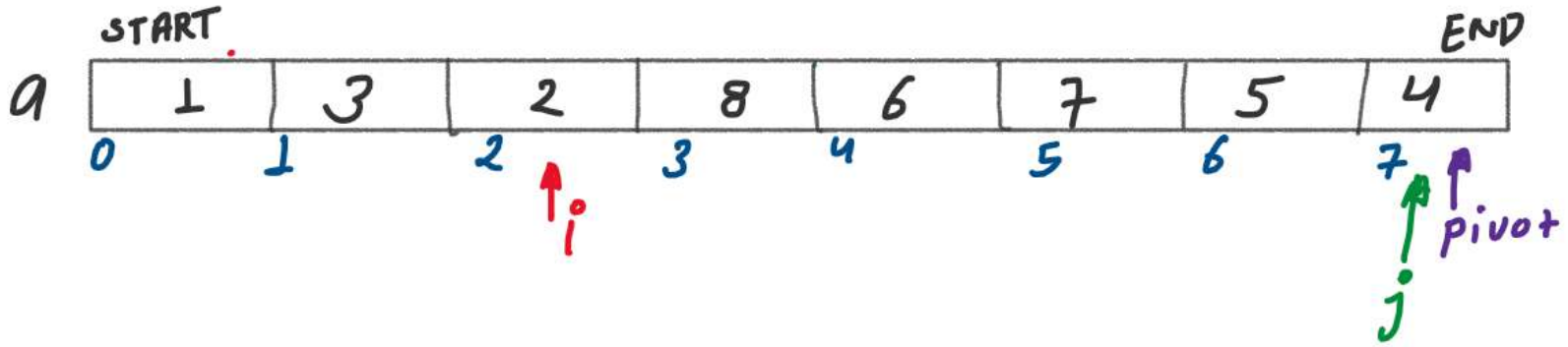
i = 2

j = ~~6~~ 7

a[j] > a[pivot]

5 7 4

j++



(8)

start = 0

End = 7

pivot = End

i = ~~2~~ 3

j = 7

while(j < pivot) {---}

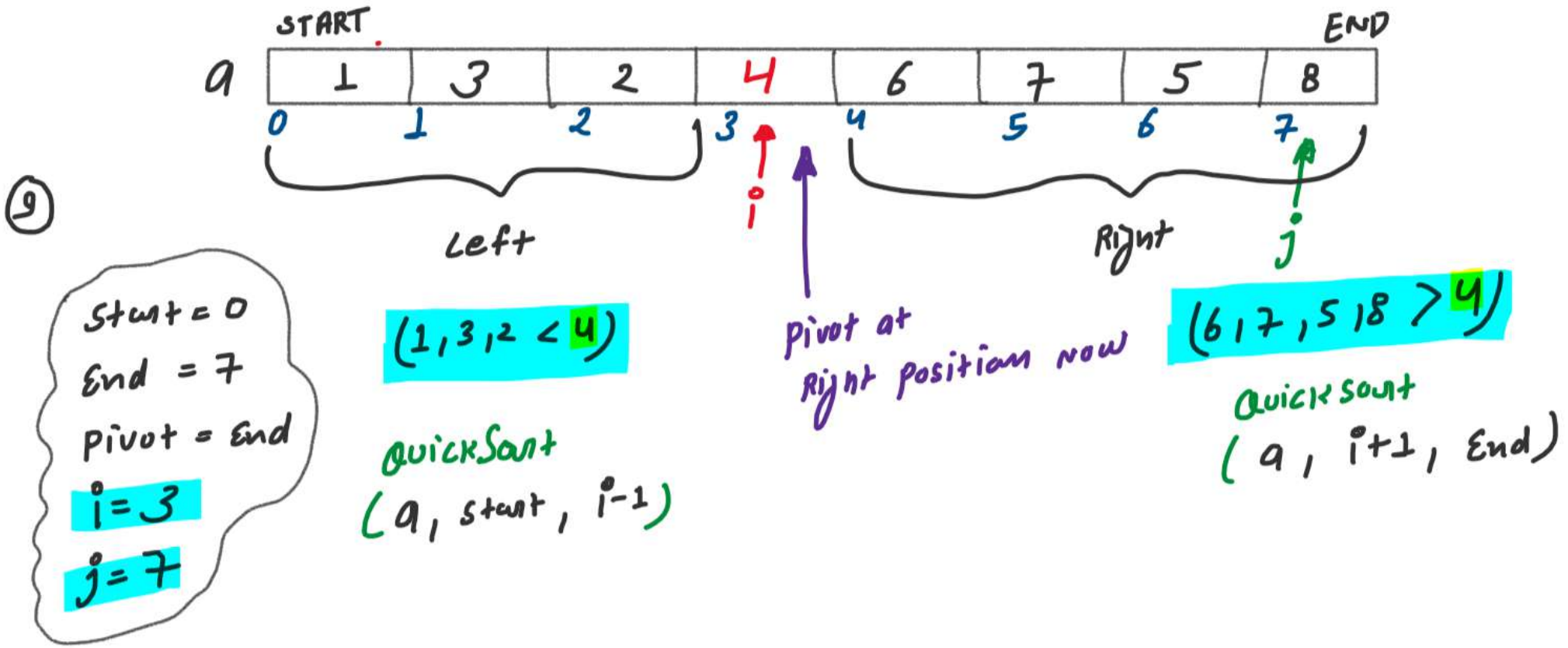
↓ 7 < 7 X while loop me entry nahi karungi

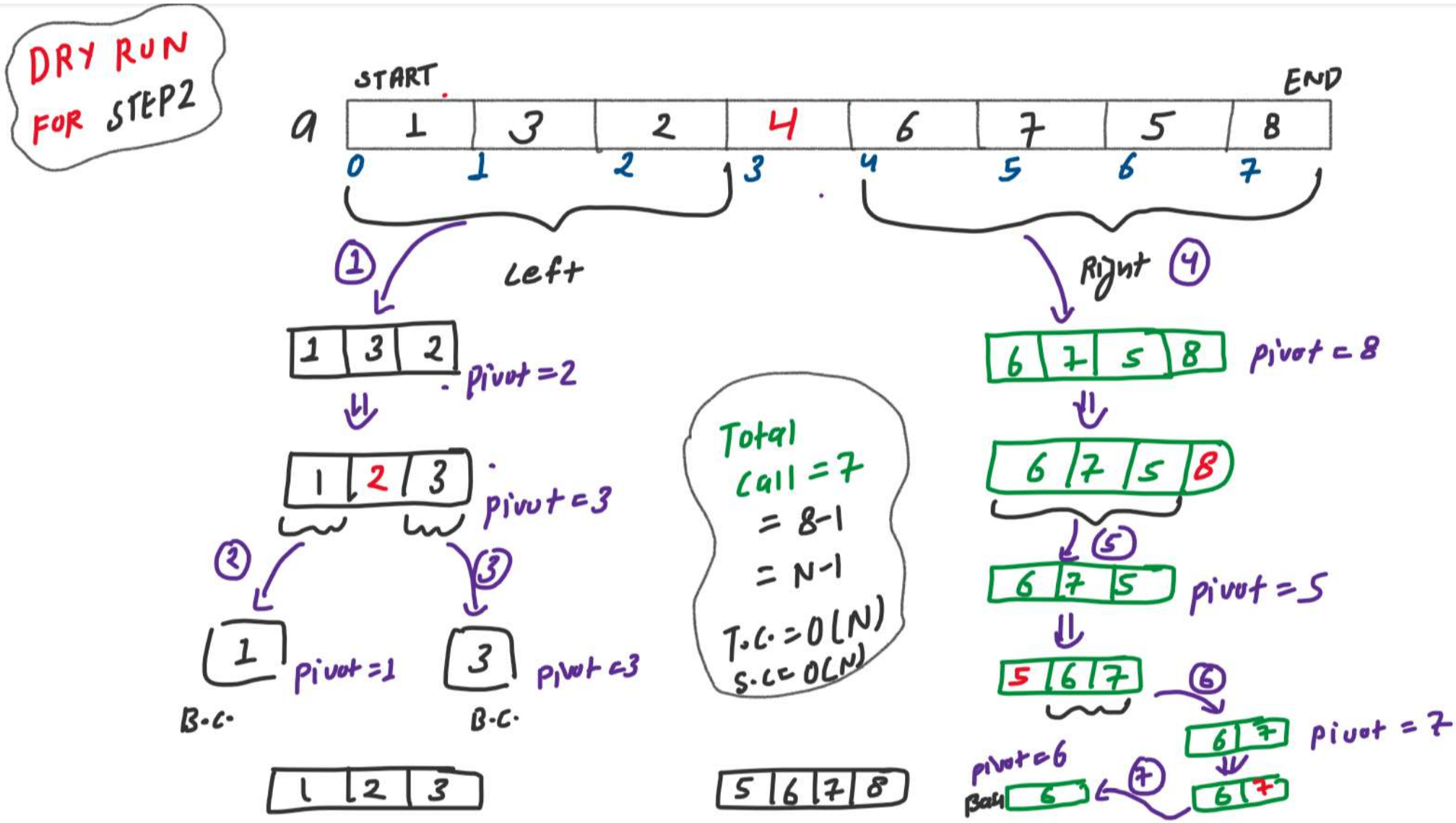
Outside of while loop

i++

swap(a[i], a[pivot]);

8 4


START
END
a
1
3
2
4
6
7
5
8
0
1
2
3
4
5
6
7
i
j
Left
Right
Pivot at
Right position now
(1,3,2 < 4)
(6,7,5,8 > 4)
quickSort
(a, start, i-1)
quickSort
(a, i+1, End)
start = 0
End = 7
pivot = End
i = 3
j = 7


DRY RUN FOR STEP2
START
END
a
1 3 2 4 6 7 5 8
0 1 2 3 4 5 6 7
Left
Right
1 3 2
pivot = 2
1 2 3
pivot = 3
1
pivot = 1
B.C.
3
pivot = 3
B.C.
1 2 3
Total call = 7
= 8-1
= N-1
T.C. = O(N)
S.C. = O(N)
5 6 7 8
6 7 5 8
pivot = 8
6 7 5
pivot = 5
5 6 7
6 7
pivot = 7
pivot = 6
Bas
6
6 7

```
//Function to sort an array using quick sort algorithm.
void quickSort(int arr[], int start, int end)
{
    // Base Case
    if(start>=end) {
        return;
    }

    int pivot = end;
    int i = start - 1;
    int j = start;

    while(j < pivot){
        if(arr[j] < arr[pivot]){
            i++;
            swap(arr[i], arr[j]);
        }
        j++;
    }
    i++;
    swap(arr[i], arr[pivot]);

    // Recursive call karlo Left part ke liye
    quickSort(arr, start, i - 1);
    // Recursive call karlo Right part ke liye
    quickSort(arr, i + 1, end);
}
```

SWAP 2 T.C. = O(N)
S.C = O(N)

SWAP: 01
T.C = O(N²)
S.C = O(1)

STEP 1 WORST CASE T.C.

EX | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 |

T.C. = O(N²) S.C. = O(1)

STEP 2 RECURSIVE CALL

S.C = O(N)
T.C. = O(N)

Total T.C. = O(N²) + O(N)
= O(N²)

Total S.C. = O(N) + (1)
= O(N)