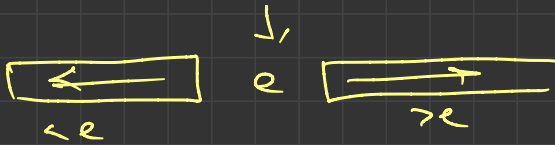





1st Step \rightarrow element

\hookrightarrow right place



Quick Sort

\hookrightarrow partition

\hookrightarrow recursion

Print \rightarrow

3	1	4	5	2
---	---	---	---	---

0 1 2 3 4

-
- A diagram of a linked list with five nodes containing the values 4, 1, 3, 5, and 2. The nodes are arranged horizontally and connected by vertical lines. Below the first two nodes (4 and 1) and the last two nodes (5 and 2), there are blue curly brackets grouping them. The node with value 3 is not grouped.

swap

4	1	3	5	2
---	---	---	---	---

2	1	3	5	4
	1		5	

2	1	3	5	4
		13		

```

int partition (vector<int> &arr, int s, int e)
{
    // get pivot
    int pivot = arr[s];

    // count elements less than or equal to pivot
    int count = 0;

    for(int i = s+1; i <= e; i++){
        if(arr[i] <= pivot){
            count++;
        }
    }

    int pivotIndex = s + count;

    // place pivot at right place
    swap(arr[s], arr[pivotIndex]);

    // use condition -> | < e | e | > e |

    int i = s, j = e;

    while(i < pivotIndex && j > pivotIndex){

        while(arr[i] <= pivot){
            i++;
        }

        while(arr[j] > pivot){
            j--;
        }

        if(i < pivotIndex && j > pivotIndex){
            swap(arr[i], arr[j]);
            i++;
            j--;
        }
    }
    return pivotIndex;
}

//Function to sort an array using quick sort
algorithm.
void solve(vector<int> &arr, int s, int e)
{
    // base case
    if(s >= e) return;

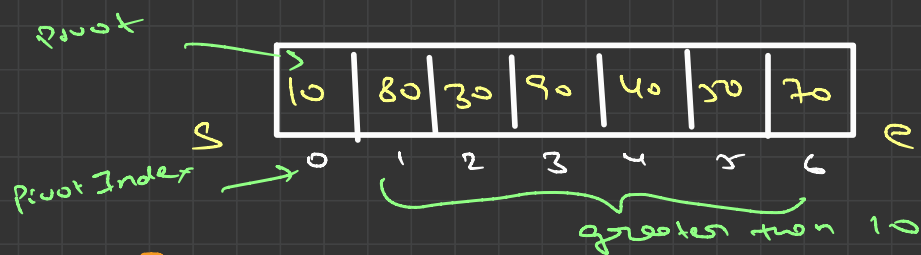
    // find pivot Index
    int p = partition(arr, s, e);

    // rec call
    solve(arr, s, p-1);
    solve(arr, p+1, e);
}

vector<int> quickSort(vector<int> arr)
{
    int s = 0;
    int e = arr.size() - 1;
    solve(arr, s, e);

    return arr;
}

```



Pivot = 10
Count = 0

Pivot Index = $s + \text{Count}$

$$\rightarrow 0 + 0 = \underline{\underline{0}}$$

while ($i < \text{pivotIndex}$ & $j > \text{pivotIndex}$)

\rightarrow return false

return pivotIndex ;

$\rightarrow 0$

Solve (arr, $p+1$, e) ;

\rightarrow index $\rightarrow 1 = s$

S	80	30	90	40	50	70	e
	1	2	3	4	5	6	

Pivot = 80

Count = 0

↳ 4

Pivot Index = S + count ;

↳ 1 + 4 = 5 ;

Swap (arr[S] , arr[Pivot Index]) ;

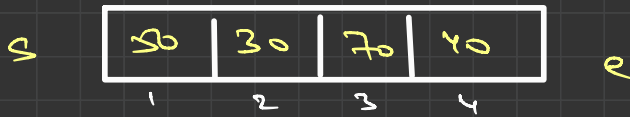
S	80	30	90	40	50	70	e
	1	2	3	4	5	6	

S	50	30	70	40	80	90	e
	1	2	3	4	5	6	

return Pivot Index ;

Solve (arr , S , P-1) ;

Solve (arr , P+1 , e) ;



Pivot = 50

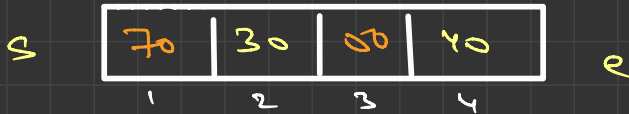
count = 0

→ 2

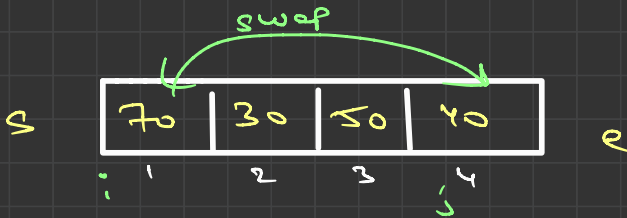
Pivot Index = $s + \text{count}$

→ $1 + 2 \Rightarrow 3$

Swap



Maintain the condⁿ :- $< e \mid e \mid > e$



⇓



return Pivot Index;

→ 3

Solve(arr, s, p-1);

Solve(arr, p+1, e);