Trishanth Naidu

Software Developer | Founder of Peppy UI | Author of Rootz JS

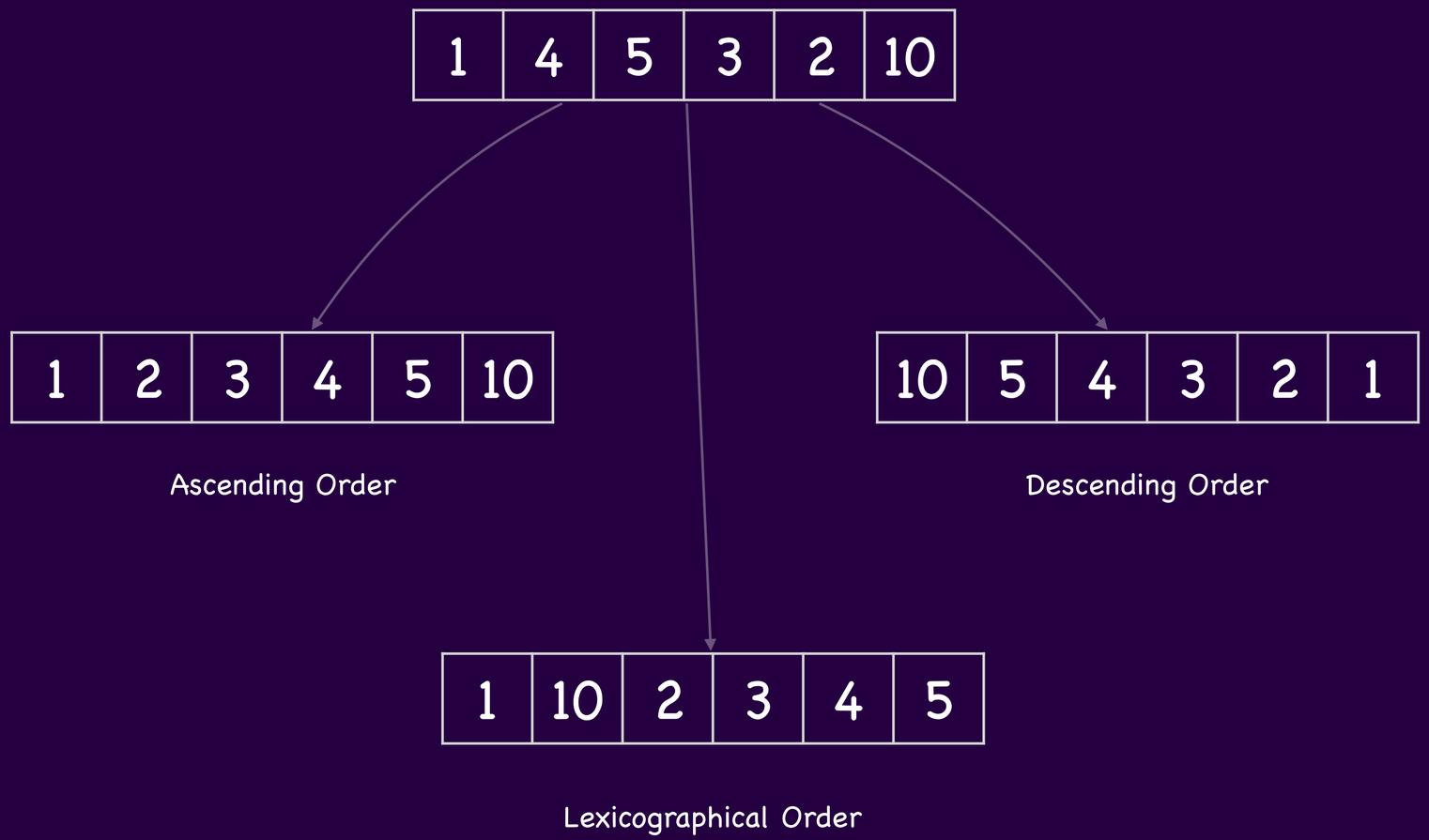
Educator at Relevel

Innovator | Fitness freak | Painter | Chef

You can Follow me on

Sorting

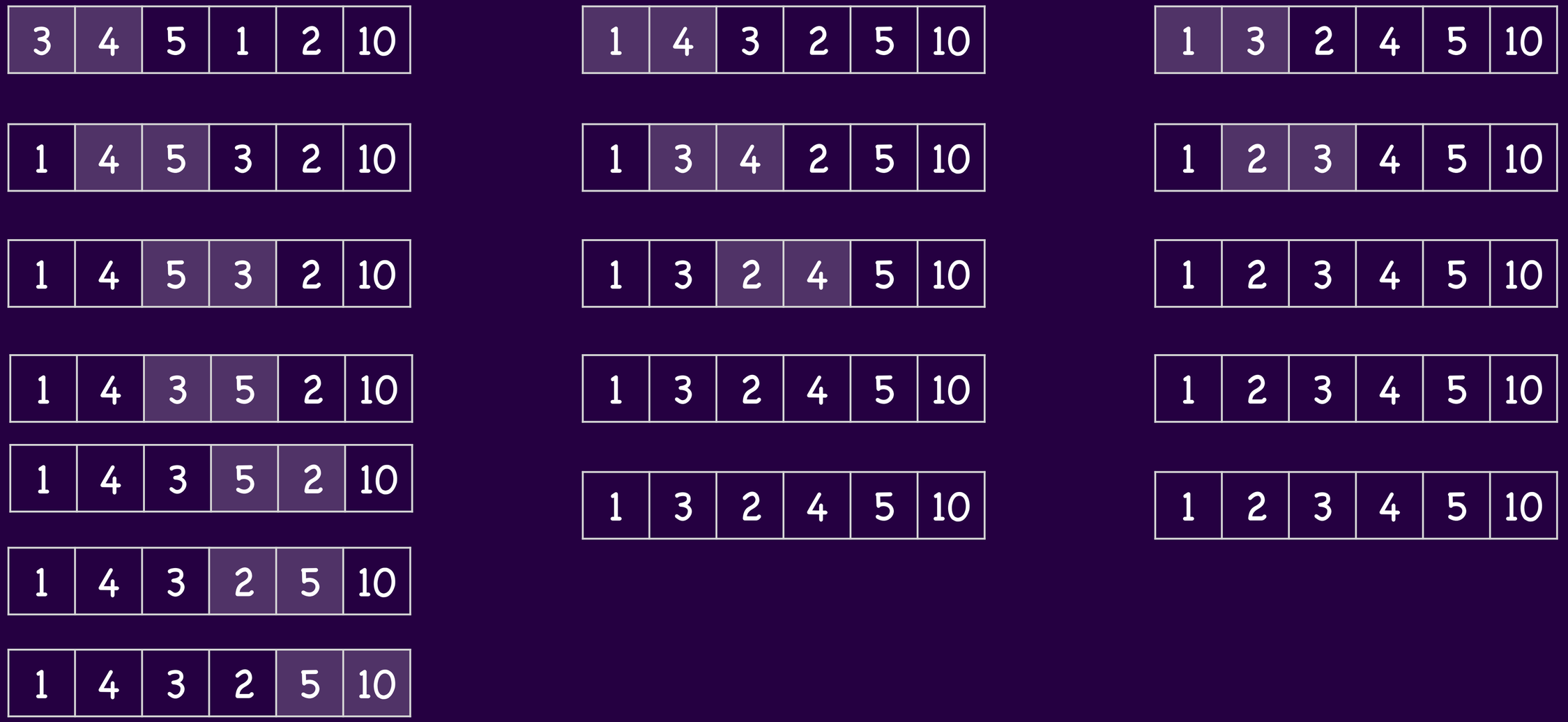
Order of Sorting



Comparison based Sorting

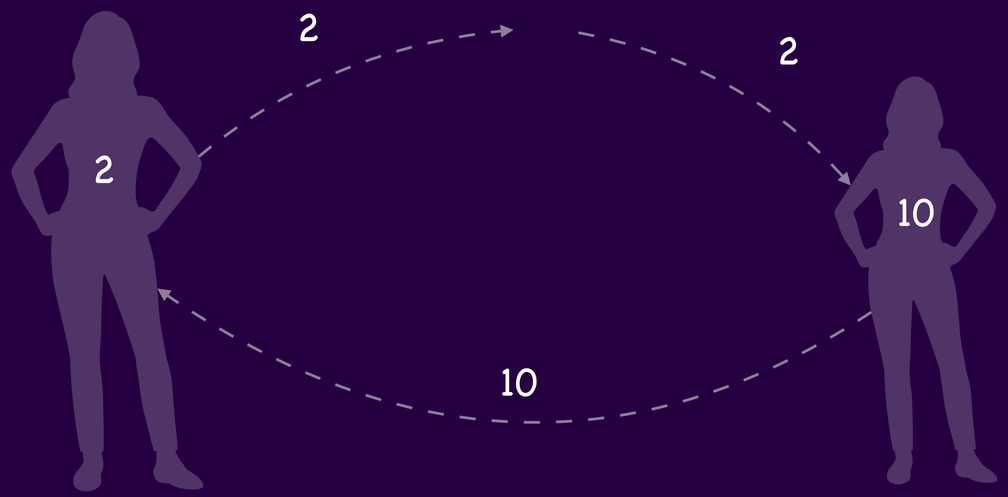
Count based Sorting

Bubble Sort

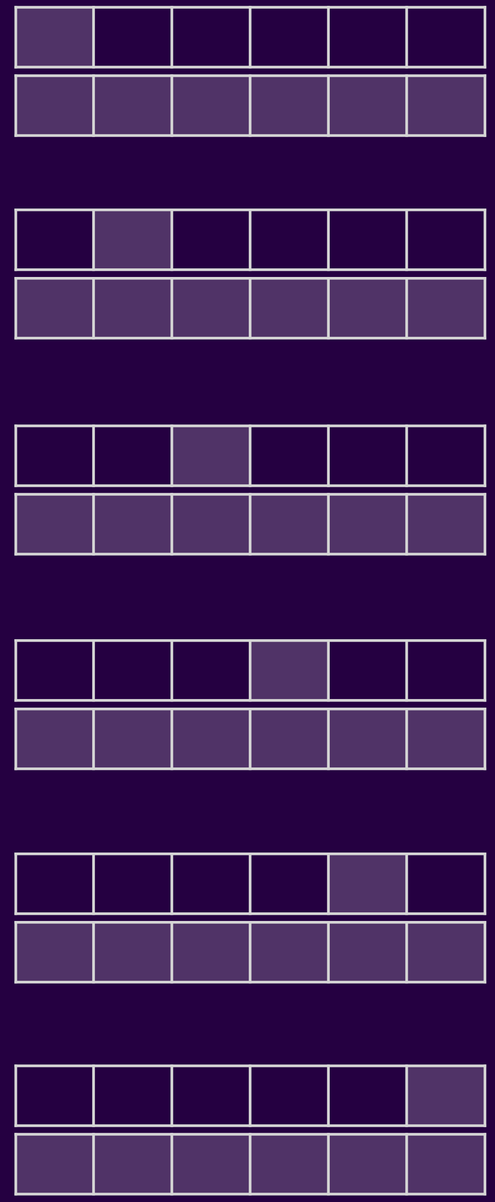

3 4 5 1 2 10
1 4 5 3 2 10
1 4 5 3 2 10
1 4 3 5 2 10
1 4 3 5 2 10
1 4 3 2 5 10
1 4 3 2 5 10
1 4 3 2 5 10
1 3 4 2 5 10
1 3 4 2 5 10
1 3 2 4 5 10
1 3 2 4 5 10
1 3 2 4 5 10
1 3 2 4 5 10
1 2 3 4 5 10
1 2 3 4 5 10
1 2 3 4 5 10
1 2 3 4 5 10

Bubble Sort Analysis

Swapping Algo (Swalgo)



Foolish looping



Bubble Sort Complexity

Time complexity

$f(n) = (n-1) + (n-2) + (n-3) + \ldots. 1$

$f(n) = n * (n-1) / 2$

$f(5) = (5-1) + (5-2) + (5-3) + \ldots. 1$

$f(5) = (4) + (3) + (2) + 1$

$f(5) = 10$

$f(5) = 5 * (5-1) / 2$

$f(5) = 10$

$f(n) = n \hat{} 2 / 2 - n / 2$

$f(n) = n \hat{} 2$

Space complexity

variable temp; O(1)

Bubble Sort Problems

Lexicographical order sorting of country names

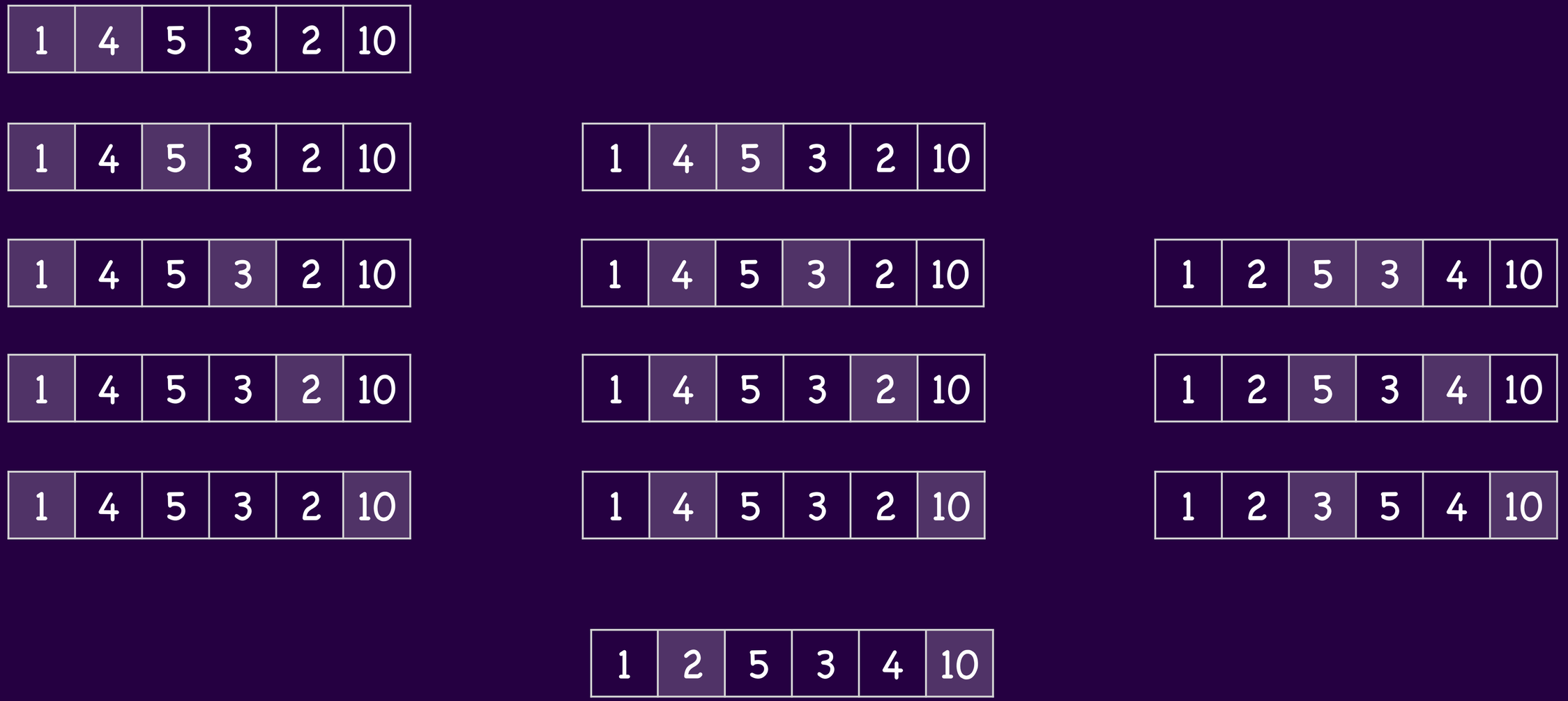
Input

India	Australia	China	Russia	Brazil	Japan

Output

Australia	Brazil	China	India	Japan	Russia

Selection Sort



Selection Sort Complexity

Time complexity

$f(n) = n + (n-1) + (n-2) + (n-3) + \ldots. 1$

$f(n) = n * (n+1) / 2$

$f(5) = 5 + (5-1) + (5-2) + (5-3) + \ldots. 1$

$f(5) = 5 + (4) + (3) + (2) + 1$

$f(5) = 15$

$f(5) = 5 * (5+1) / 2$

$f(5) = 15$

$f(n) = n \hat{} 2 / 2 + n / 2$

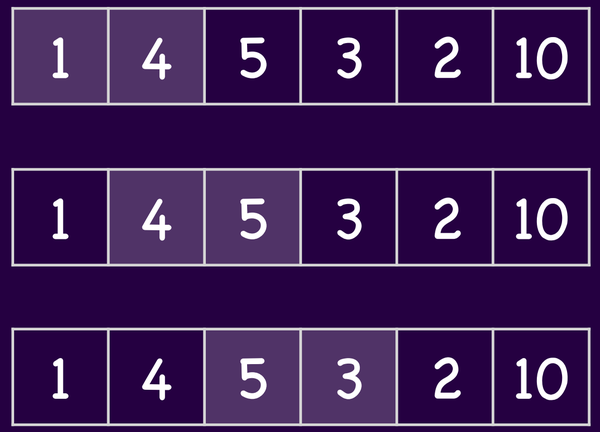
$f(n) = n \hat{} 2$

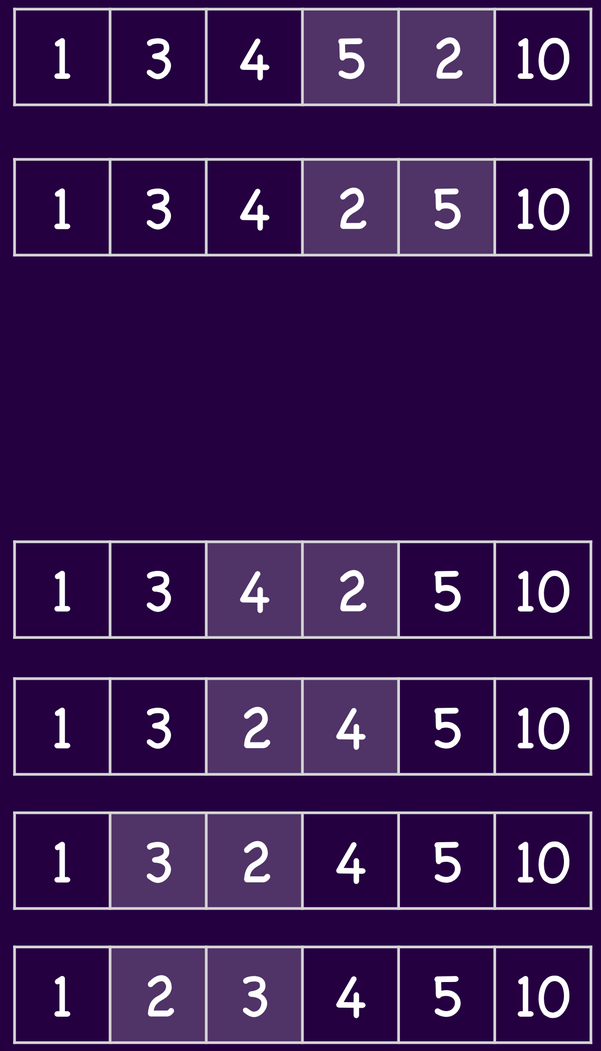
Space complexity

variable toIndex, toValue; $O(2) \sim O(1)$

Bubble Sort VS Selection Sort

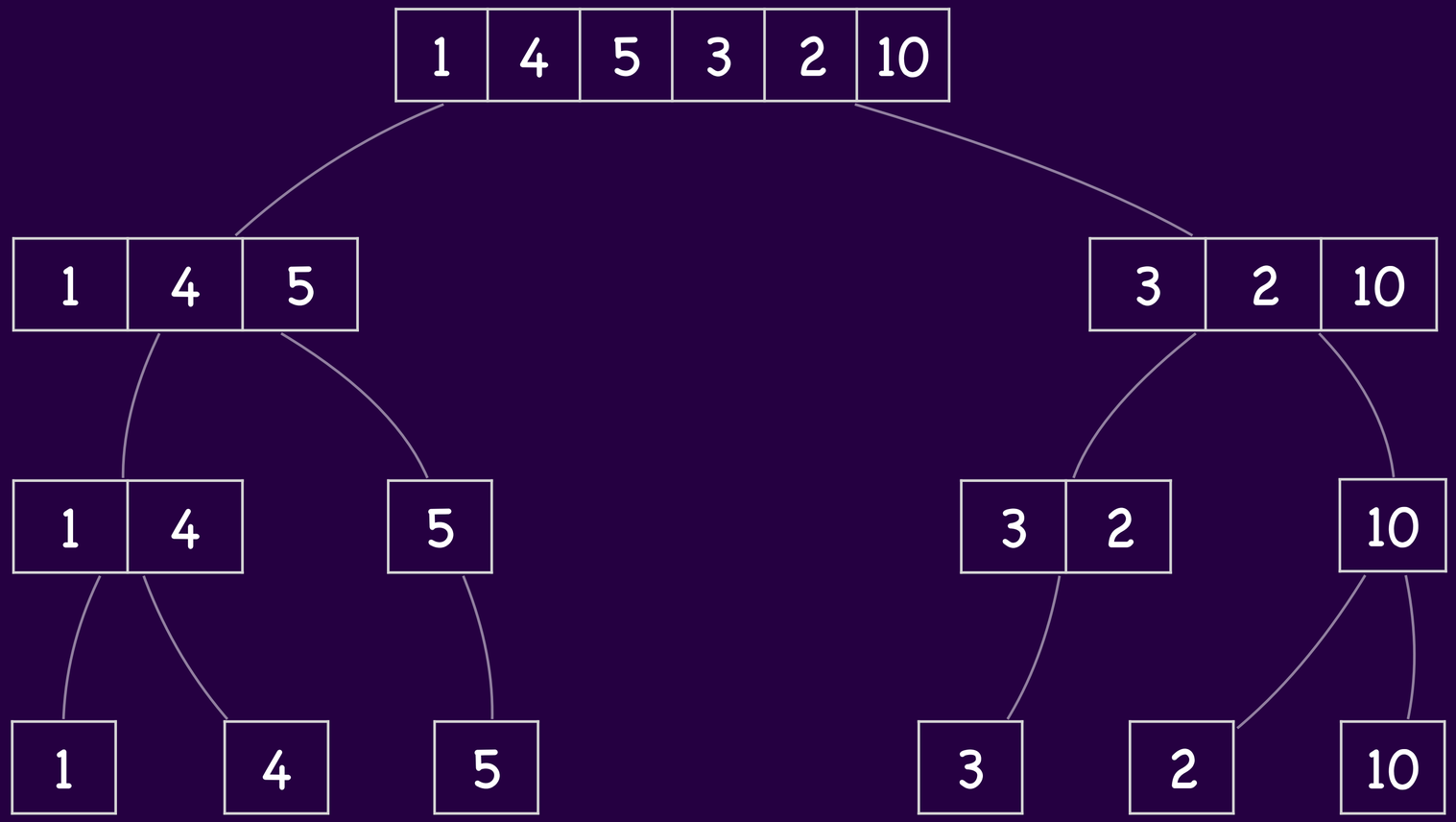
Insertion Sort



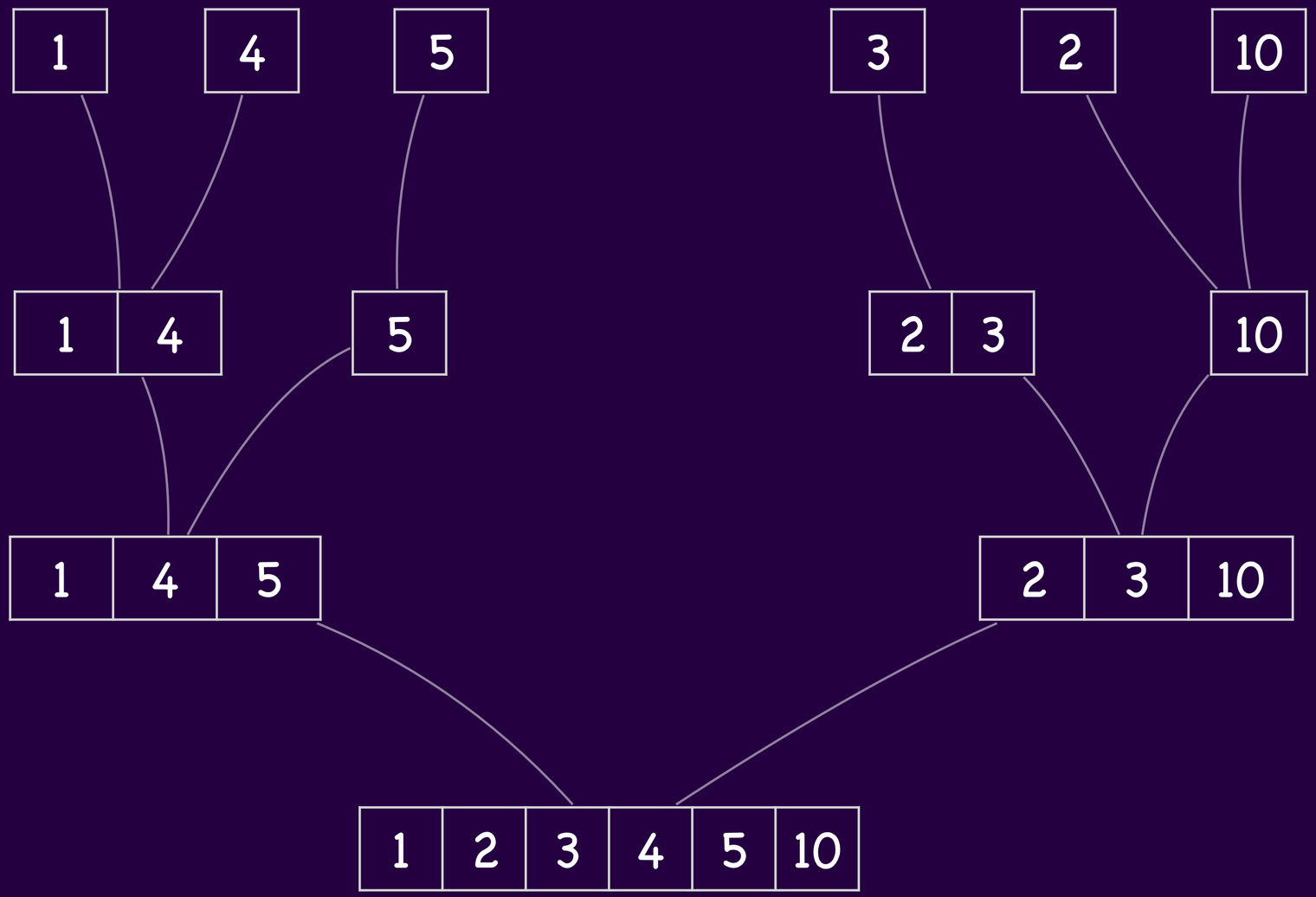


Merge Sort

Merge Sort - Division



Merge Sort - Merging

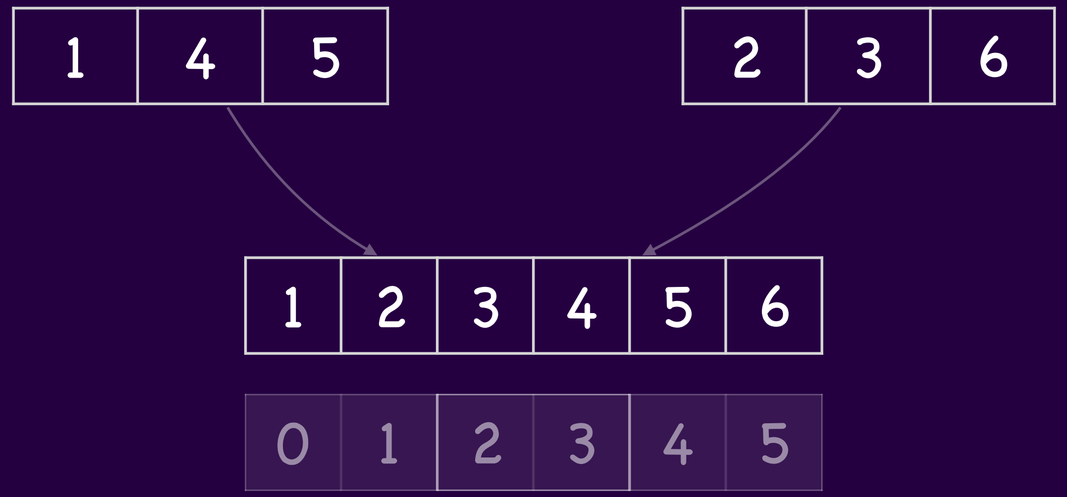


Merge Sort - Analysis

```
// Time complexity
// T(n) = 2T(n/2)
// T(n) = 4T(n/4)
// T(n) = T(n/4)
// T(n) = T(n/8)
// T(n) = T(n/16)
// T(n) = T(n/2^n)
// T(n) = T(n * 1/2^n)
// T(n) = O(nLogn) + O(n)

// T(n) = O(n Logn)
// T(n) = Omega(n Logn)

var result = [];
var arr = [14,33,27,10,35,19,42,44];
function mergeSort(arr, left, right) {
    var middle = Math.floor((left + right) / 2);

    if(left >= right) return;

    mergeSort(arr, left, middle);
    mergeSort(arr, middle + 1, right);
    var sortedArr = merge(left, middle, right)
    for(let i=left, j=0; i<=right; i++, j++) {
        arr[i] = sortedArr[j];
    }
    result = sortedArr;
}
```

```
// O(n); n>m
function merge(left, middle, right) {
    var i = 0, j = 0, result = [];
    var a1 = arr.slice(left, middle + 1);
    var a2 = arr.slice(middle + 1, right + 1);

    while(i < a1.length || j < a2.length) {
        if(a1[i] < a2[j] || a2[j] === undefined) {
            result.push(a1[i]);
            i++;
        } else if(a2[j] < a1[i] || a1[i] === undefined) {
            result.push(a2[j]);
            j++;
        }
    }
    return result;
}

mergeSort(arr, 0, arr.length - 1);
console.log(result);
```

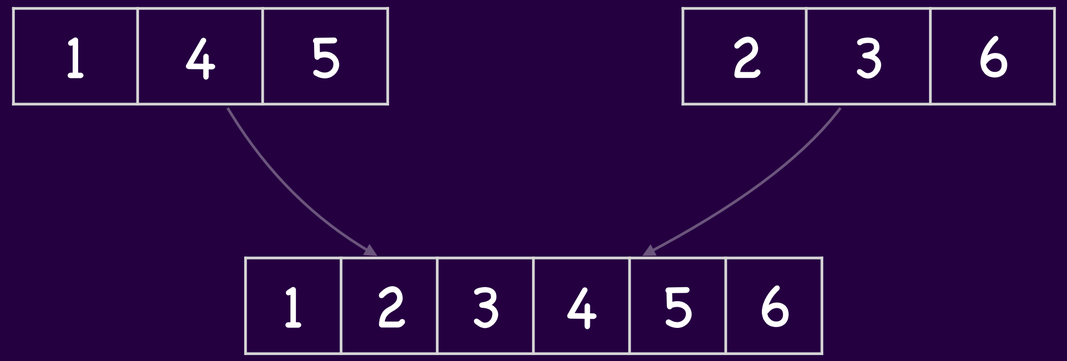
Median



M = (L + R) / 2
M = (0 + 5) / 2
M = 2.5

Median = arr[(M(floor)] + arr[M(Ceil)]) / 2
Median = (3 + 4) / 2
Median = 3.5

Sort an Array



Inversion Count

Those number/s which are not placed at the sorted location
and
the count of misplaced elements with respect to the misplaced number is called as inversion count

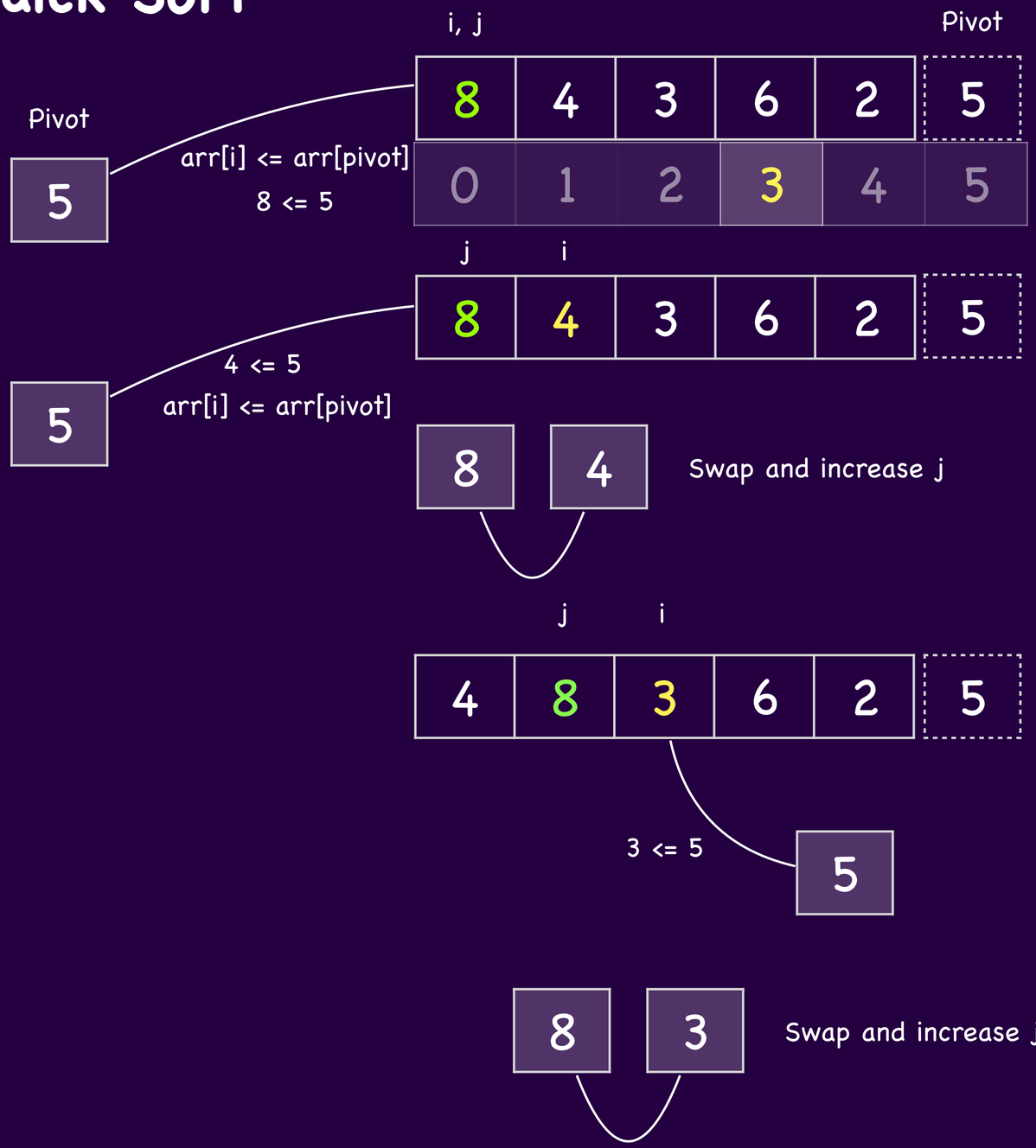
1	10	3	4	5	6

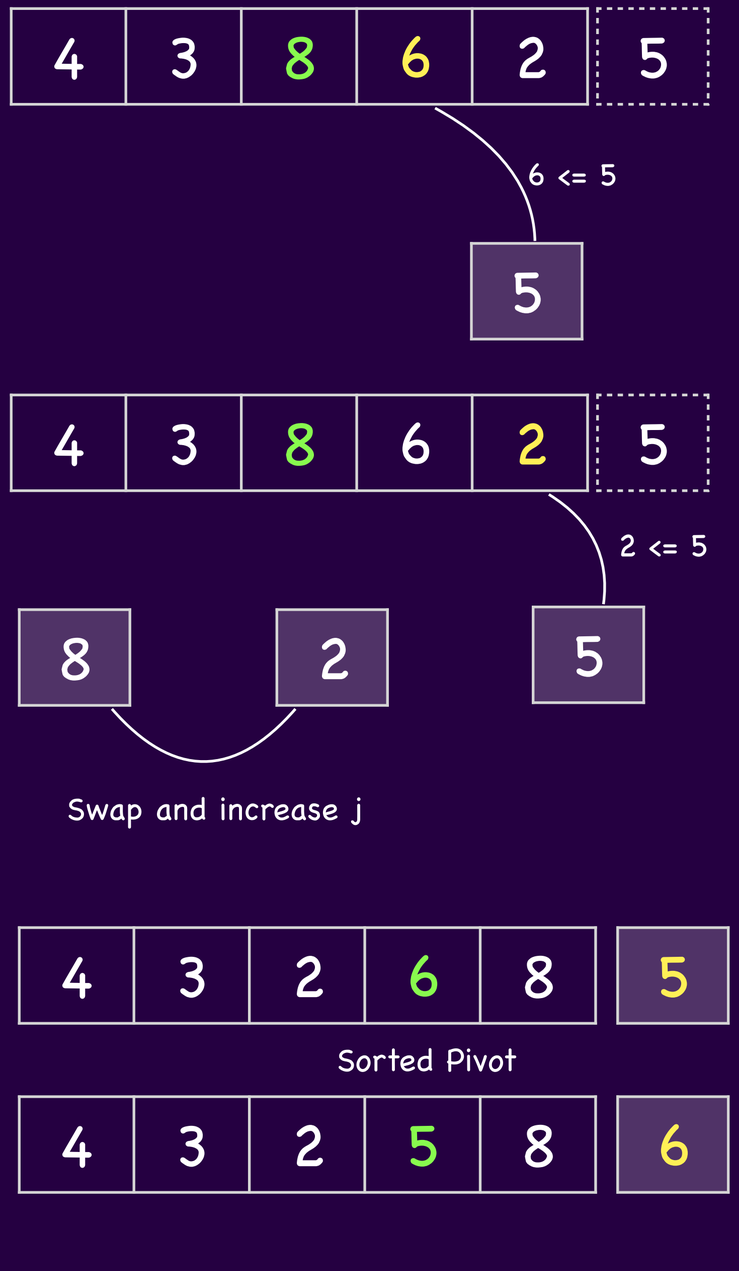
IC = [10,3] [10,4] [10,5] [10,6]
DIC = [10,6] [10,8]

Quick Sort

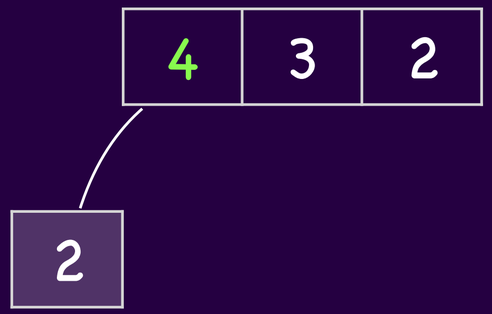
i = 0 => n	j=0	8	4	3	6	2	5	
i = 1	j=0	8	4	3	6	2	5	arr[i] <= Pivot then Swap(arr[i], arr[j]); j++
i = 2	j=1	4	8	3	6	2	5	arr[i] <= Pivot then Swap(arr[i], arr[j]); j++
i = 3	j=2	4	3	8	6	2	5	
i = 4	j=2	4	3	8	6	2	5	arr[i] <= Pivot then Swap(arr[i], arr[j]); j++
i = 5	j=3	4	3	2	6	8	5	arr[i] <= Pivot then Swap(arr[i], arr[j]); j++
		4	3	2	5	8	6	

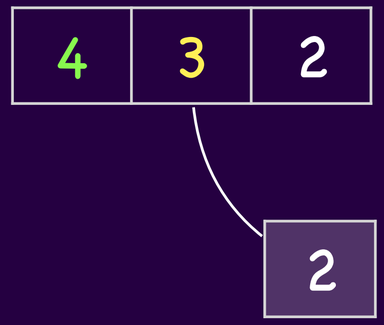
Quick Sort

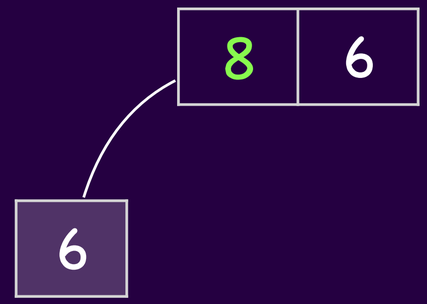

i, j
Pivot
8 4 3 6 2 5
Pivot
5
arr[i] <= arr[pivot]
8 <= 5
0 1 2 3 4 5
j i
8 4 3 6 2 5
4 <= 5
5
arr[i] <= arr[pivot]
8 4
Swap and increase j
j i
4 8 3 6 2 5
3 <= 5
5
8 3
Swap and increase j


4 3 8 6 2 5
6 <= 5
5
4 3 8 6 2 5
2 <= 5
8 2
5
Swap and increase j
4 3 2 6 8 5
Sorted Pivot
4 3 2 5 8 6

Quick Sort



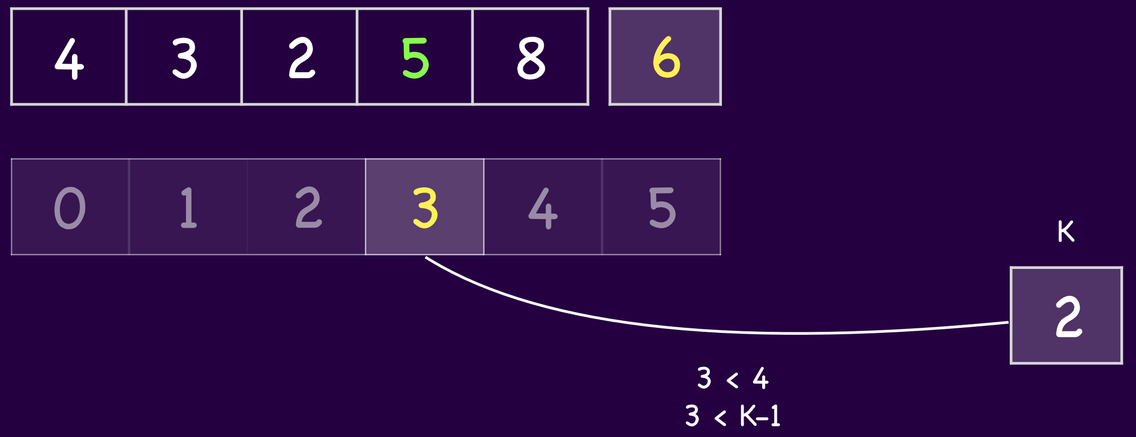




Quick Sort


4 3 2 5 8 6
0 1 2 3 4 5
4 3 2
5
8 6
L 1 P-1
P
P+1 R
L 1 R
3
L R
0 1 2
4 5
i,j 1 Pivot
i,j Pivot

Quick Select



If the pivot position is less than the K-1

Then

Search on the right had side of the sub-array

Else

Search on the left hand side of the sub-array