Algo: CS205

Algorithm

Dr: Mohamed Behiry
Eng:Mariam Hagag

Team Members

Ahmed Aboelnaga

225258

Manar Basuoni

225180

Mohamed Ashraf

225257

TOPICS

Quick sort

Binary search

Linear search

Bubble sort

Merge sort

Insertion sort

Heap sort

Selection sort

Insertion sort

Best case

O(n)

Average case

O(n2)

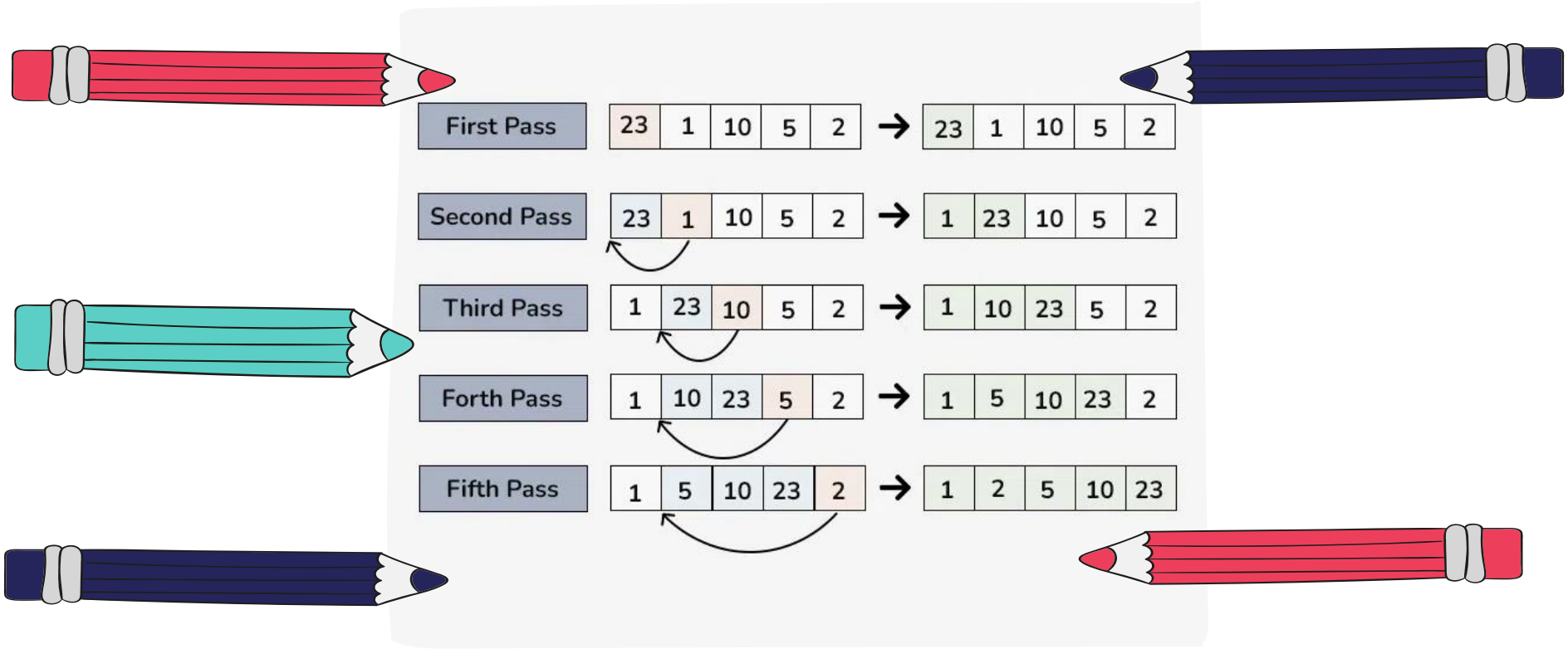
Worst case

O(n2)

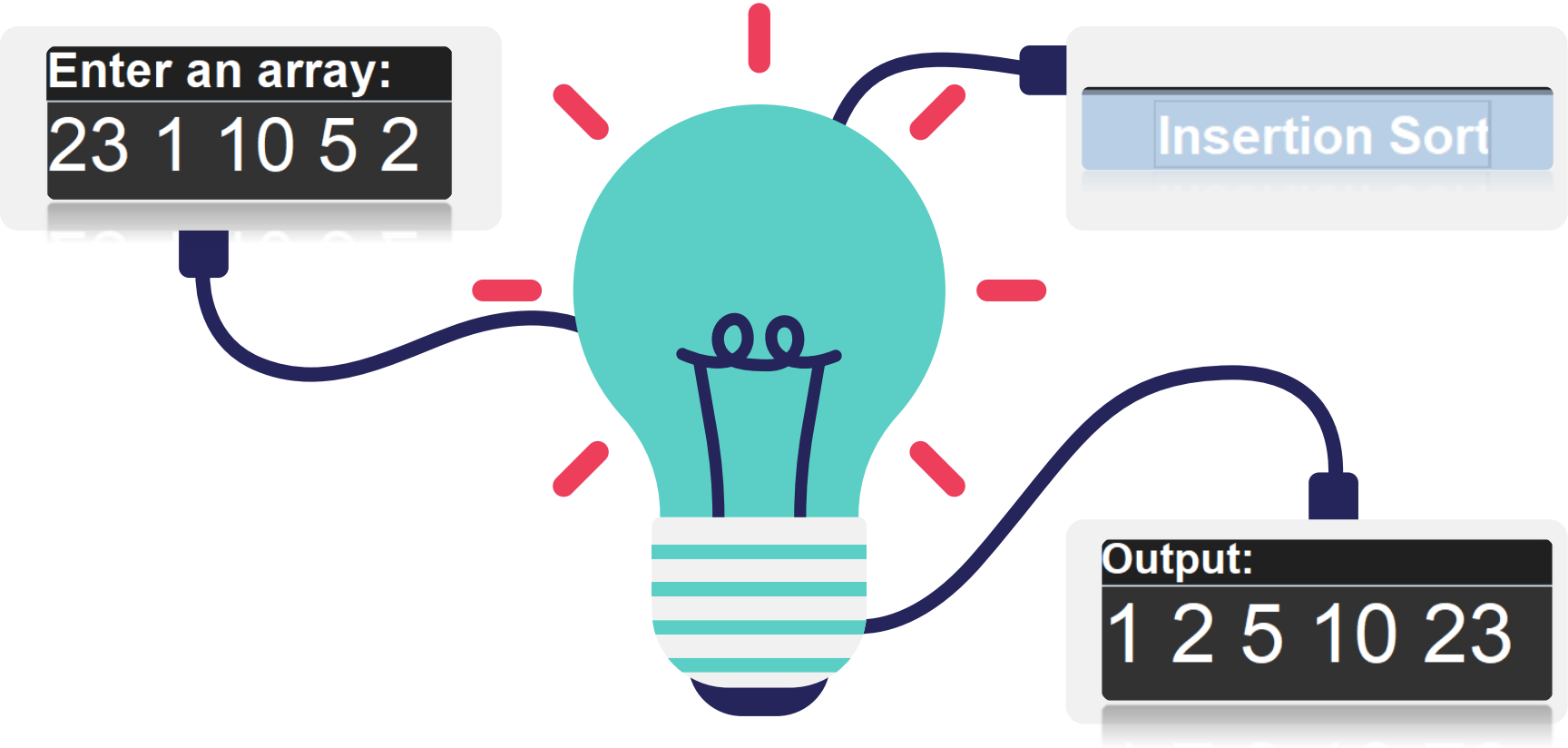
coding

```
// Insertion sort
public static void insertionSort(int[] arr)
{
    for (int i = 0; i < arr.length; i++) {
        int key = arr[i];
        int j = i - 1;
        while (j >= 0 && arr[j] > key) {
            arr[j + 1] = arr[j];
            j--;
        }
        arr[j + 1] = key;
    }
}
```

Passes


First Pass
23 1 10 5 2
23 1 10 5 2
Second Pass
23 1 10 5 2
1 23 10 5 2
Third Pass
1 23 10 5 2
1 10 23 5 2
Forth Pass
1 10 23 5 2
1 5 10 23 2
Fifth Pass
1 5 10 23 2
1 2 5 10 23

GUI



Merge sort

Best case

O(n log n)

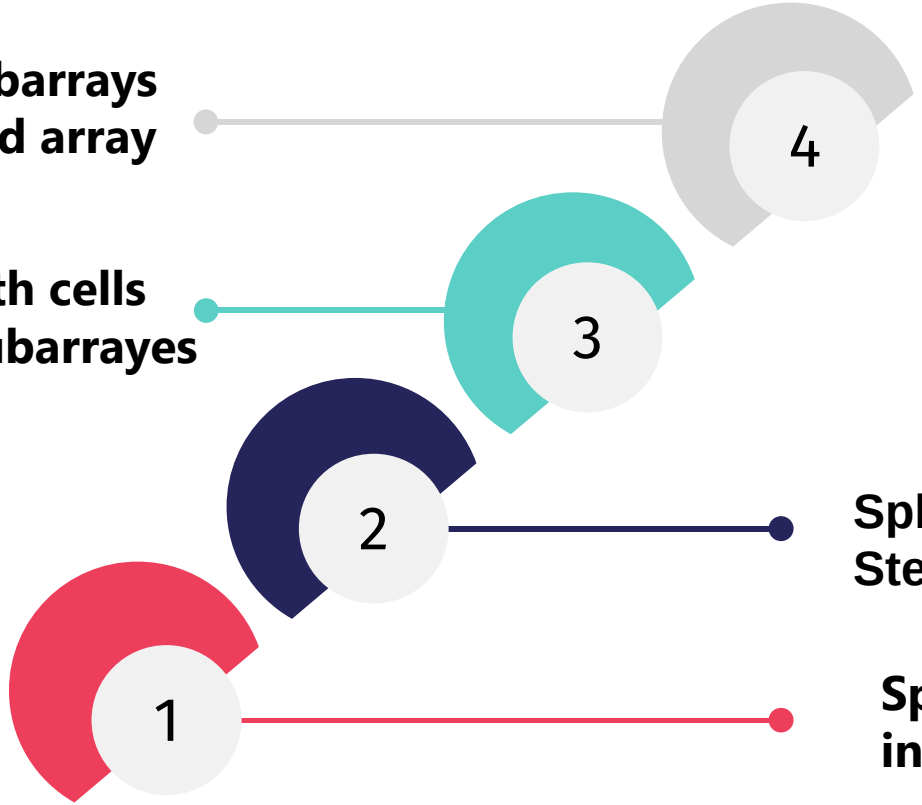
Average case

O(n log n)

Worst case

O(n log n)

STEPS



Merging sorted subarrays
Step into the sorted array

Merging unit length cells
Step into sorted subarrayes

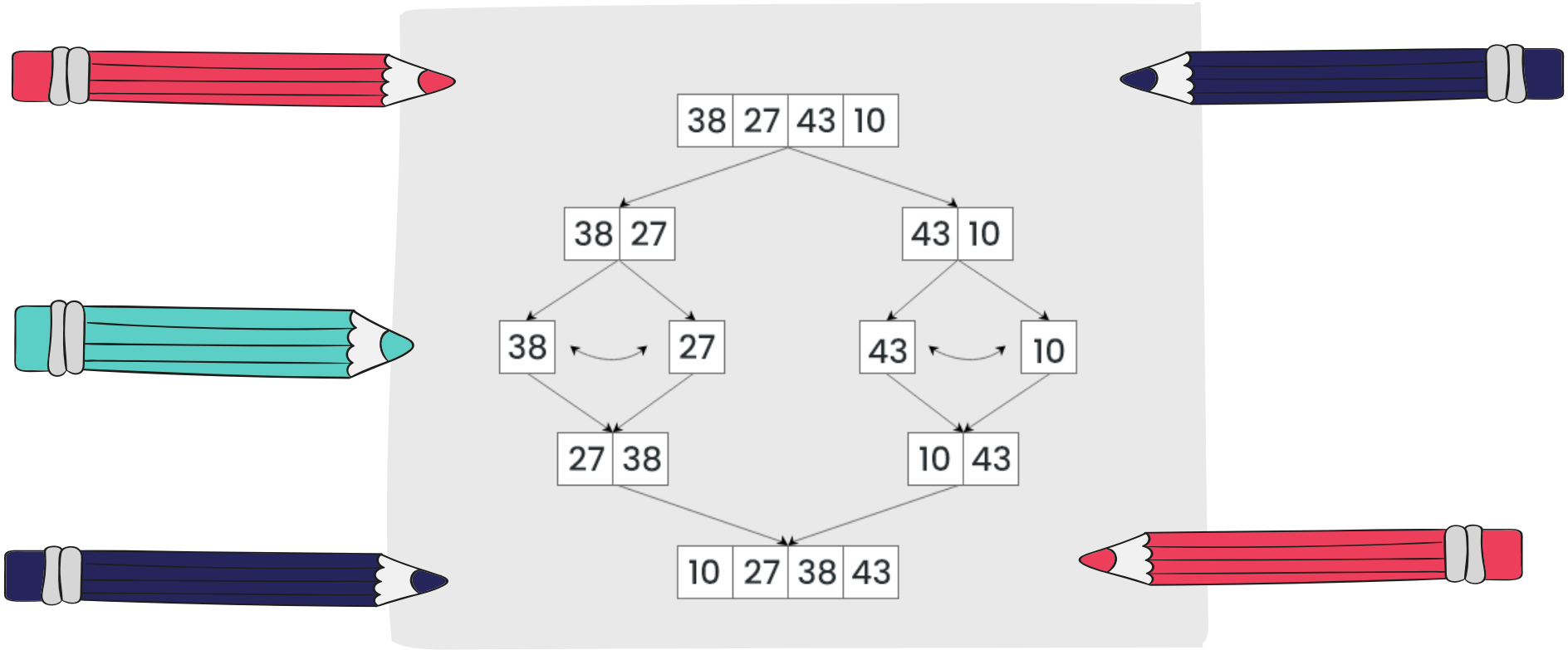
Splitting the subarrays
Step into two halves

Splitting the Array
into two equal halves

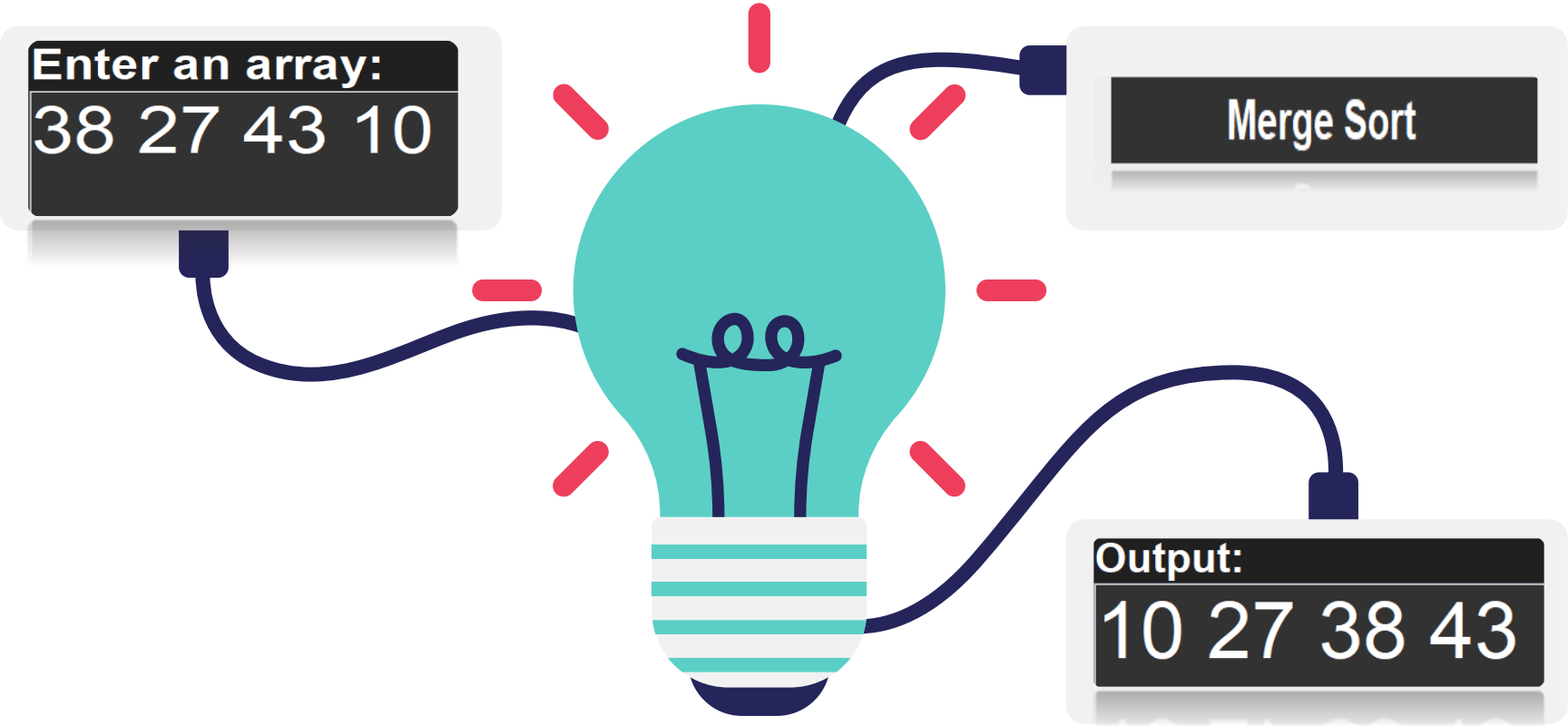
```java
// Merge Sort
public static void mergeSort(int[] arr) {
    if (arr.length <= 1) {
        return;}
    int middle = arr.length / 2;
    int[] leftArr = new int[middle];
    int[] rightArr = new int[arr.length - middle];
    int j = 0;
    for (int i = 0; i < arr.length; i++) {
        if (i < middle) { leftArr[i] = arr[i];
        } else {
            rightArr[j] = arr[i];
            j++; }}
    mergeSort(leftArr);
    mergeSort(rightArr);
    merge(leftArr, rightArr, arr);
}
```

```java
private static void merge(int[] leftArr, int[] rightArr,
int[] arr) {
    int i = 0, l = 0, r = 0;
    while (l < leftArr.length && r < rightArr.length) {
        if (leftArr[l] < rightArr[r]) {
            arr[i] = leftArr[l];
            i++;   l++;} else {
            arr[i] = rightArr[r];
            i++;
            r++;}}
    while (l < leftArr.length) {
        arr[i] = leftArr[l];
        l++;  i++ ;  }
    while (r < rightArr.length) {
        arr[i] = rightArr[r];
        r++;  i++;}}
```

Passes



GUI



Bubble sort

Best case
O(N)

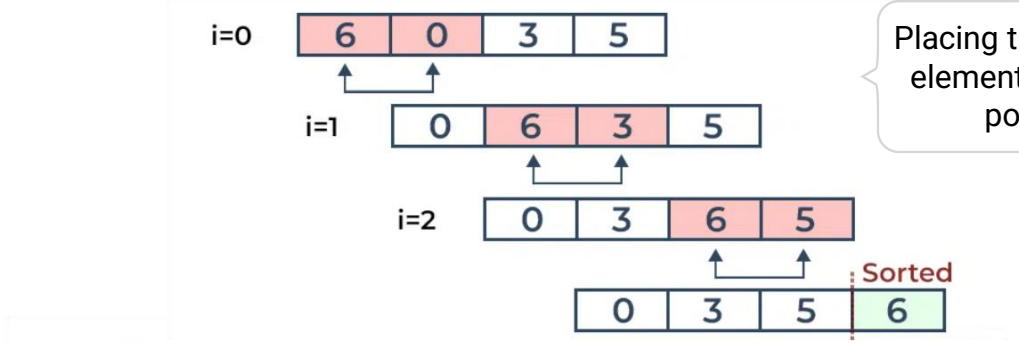
Average case
O(N2)

Worst case
O(N2)

coding

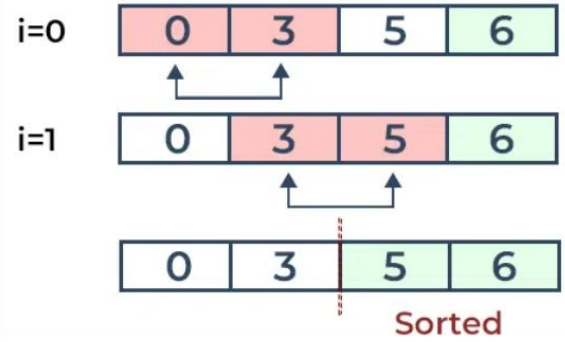
```
// Bubble Sort
public static void bubbleSort(int[]
arr) {
    for (int i = 0; i < arr.length - 1; i++)
{
        for (int j = 0; j < arr.length - 1 - i
; j++) {
            if (arr[j] > arr[j + 1]) {
                int temp = arr[j];
                arr[j] = arr[j + 1];
                arr[j + 1] = temp;
            }}}}
```

Steps



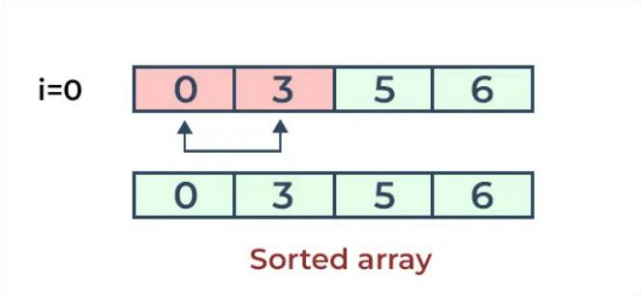
Placing the |`` largest element at Correct position

1



Placing 2"d largest element at Correct position

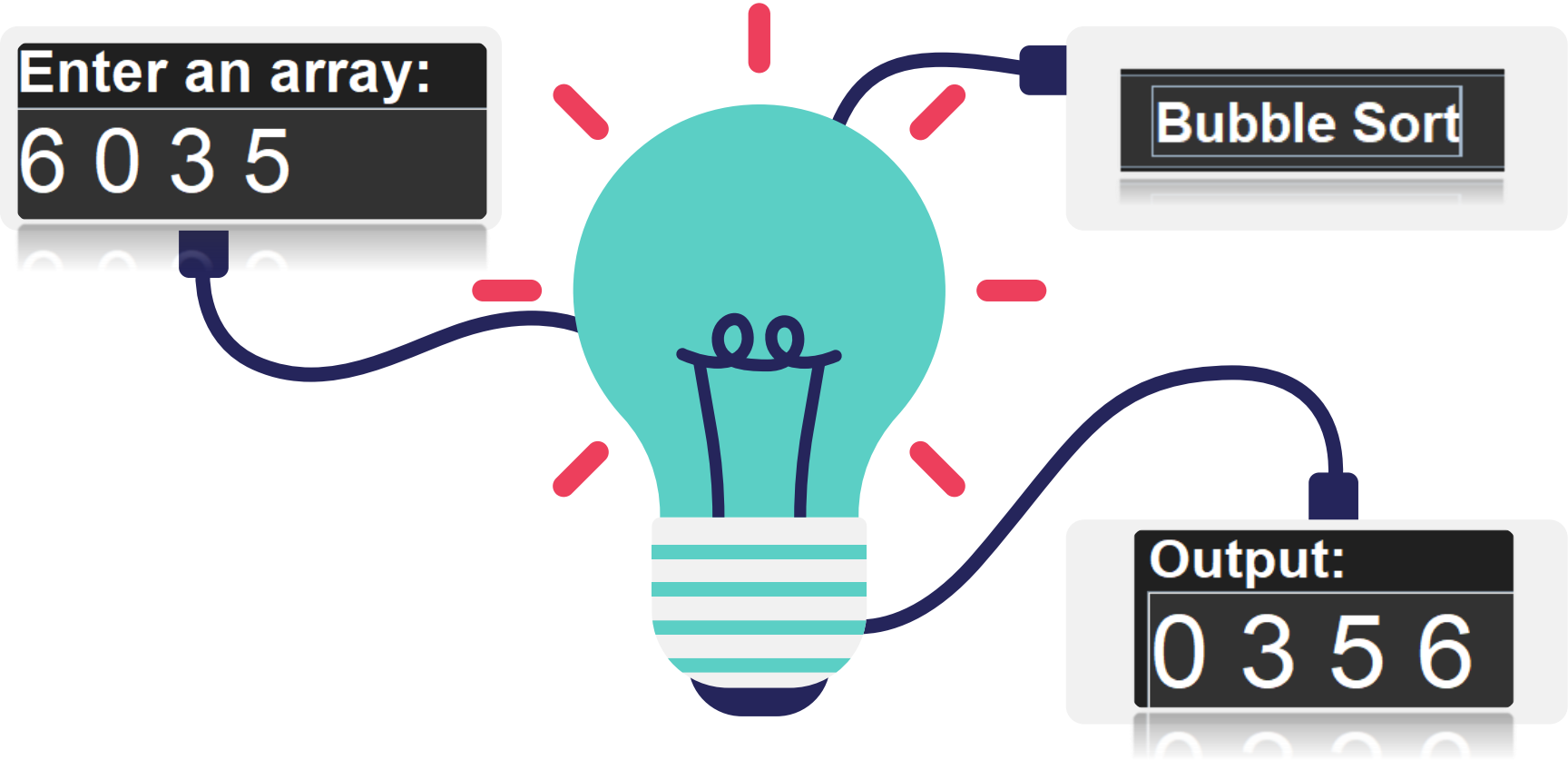
2



Placing 3'd largest element at Corrert position

3

GUI



Selection sort

Best case

O(N2)

Average case

O(N2)

Worst case

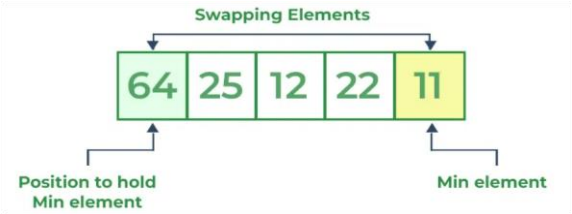
O(N2)

coding

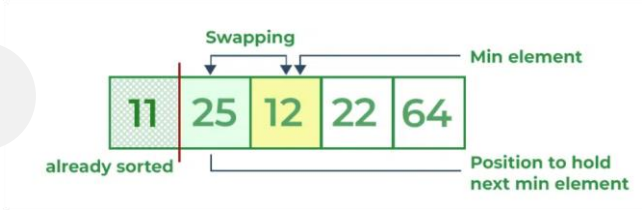
```
// Selection Sort
public static void selectionSort(int[]
arr) {
    for (int i = 0; i < arr.length; i++) {
        int minIndex = i;
        for (int j = i + 1; j < arr.length;
j++) {
            if (arr[j] < arr[minIndex]) {
                minIndex = j;}}
        int temp = arr[minIndex];
        arr[minIndex] = arr[i];
        arr[i] = temp;}}
```

STEPS

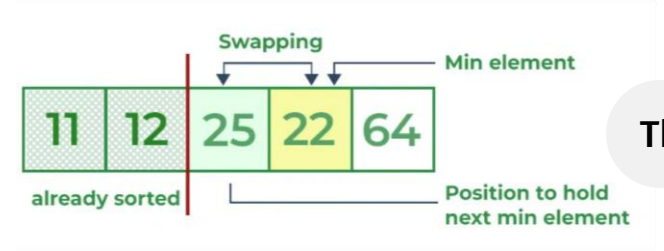
First pass: 1



2 Second Pass:

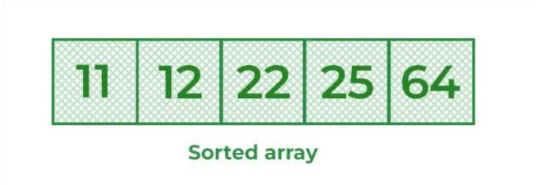


Third Pass: 3

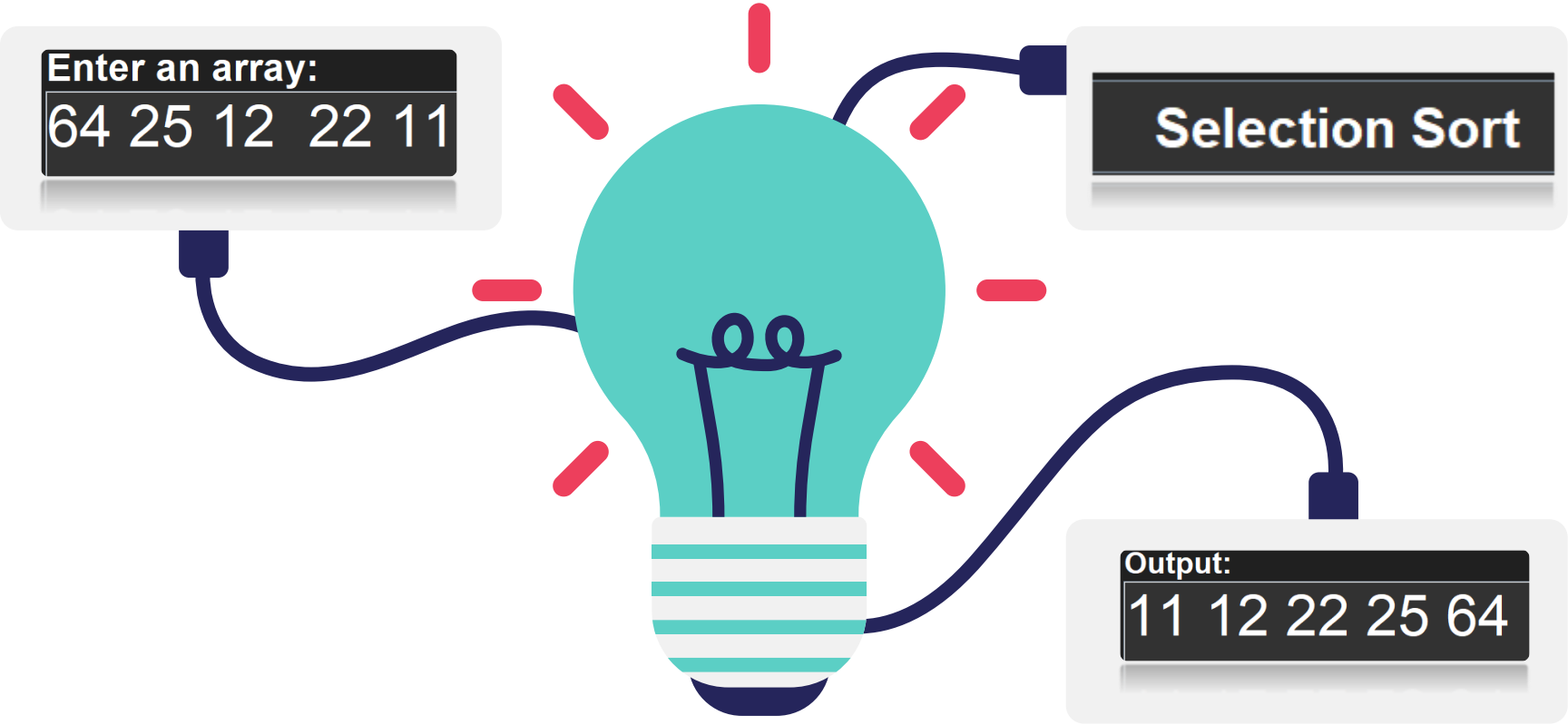


4 Fourth pass:

Fifth Pass: 5



GUI


Enter an array:
64 25 12 22 11
Selection Sort
Output:
11 12 22 25 64

Quick sort

Best case

Ω (N log (N))

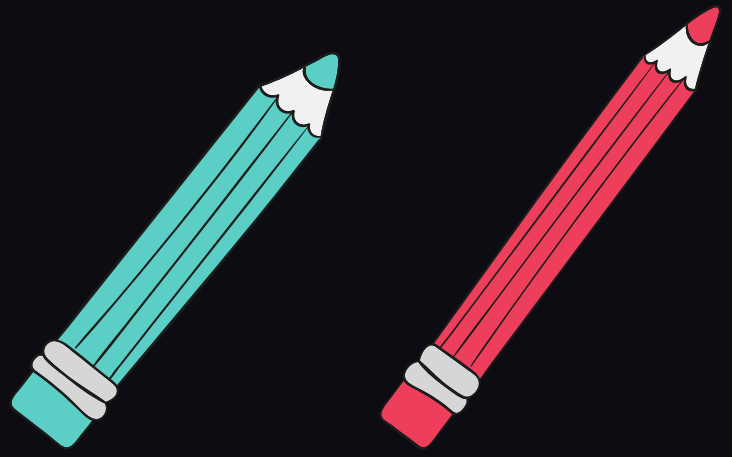
Average case

θ (N log (N))

Worst case

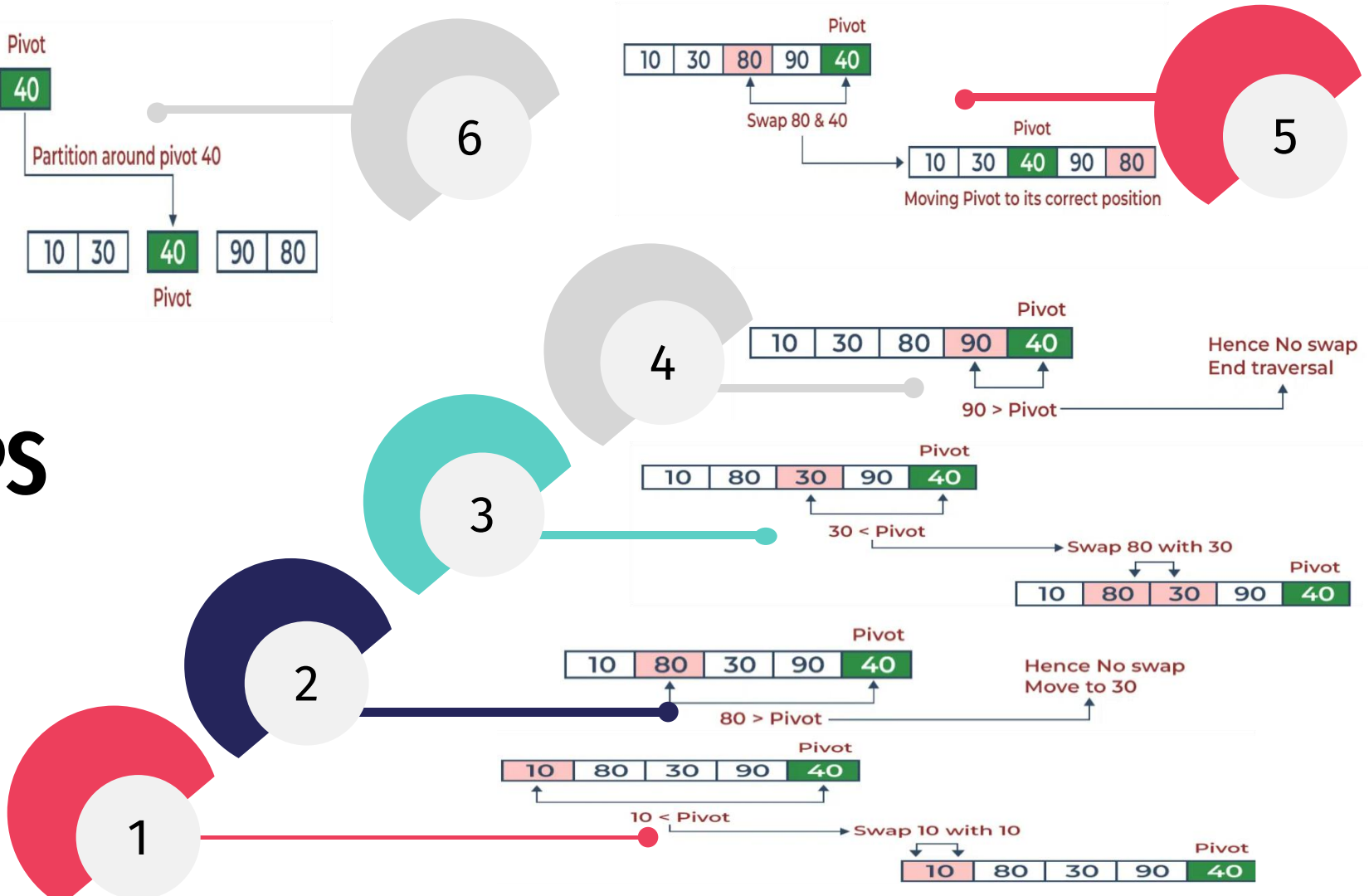
O(N2)

```java
// Quick Sort
public static void quickSort(int[] arr, int start, int end) {
    if (end <= start) {return;}
    int pivot = partition(arr, start, end);
    quickSort(arr, start, pivot -1);
    quickSort(arr, pivot + 1, end);
}
```

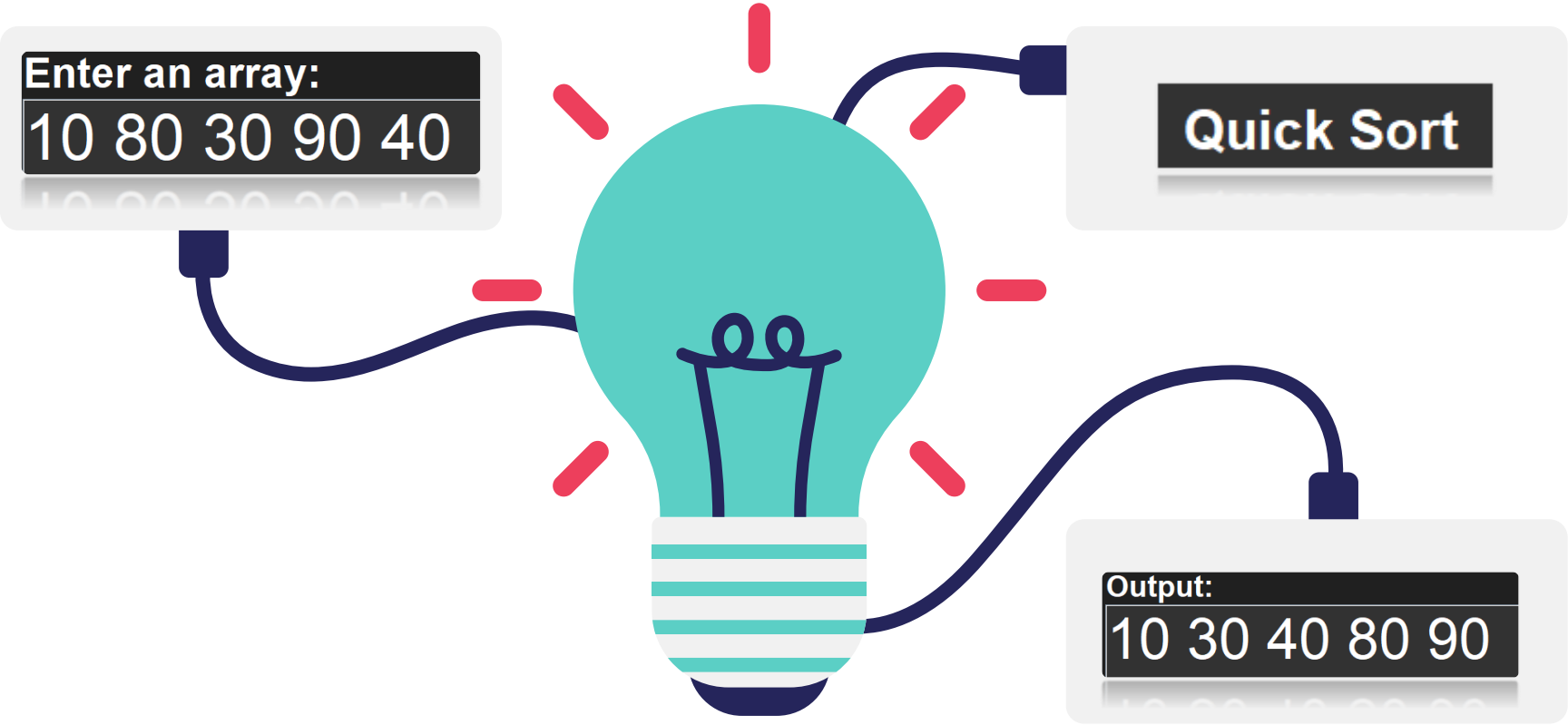


```java
private static int partition(int[] arr, int start, int end) {
    int pivot = arr[end];
    int i = start - 1;
    for (int j = start; j < end; j++) {
        if (arr[j] < pivot) {
            i++;
            int temp = arr[i];
            arr[i] = arr[j];
            arr[j] = temp;
        }  }
    i++;
    int temp = arr[i];
    arr[i] = arr[end];
    arr[end] = temp;
return i;}
```

STEPS


1
10 < Pivot
Swap 10 with 10
2
80 > Pivot
Hence No swap
Move to 30
3
30 < Pivot
Swap 80 with 30
4
90 > Pivot
Hence No swap
End traversal
5
Swap 80 & 40
Moving Pivot to its correct position
6
Partition around pivot 40
Pivot

GUI



HEAP sort

Best case

Ω (N log (N))

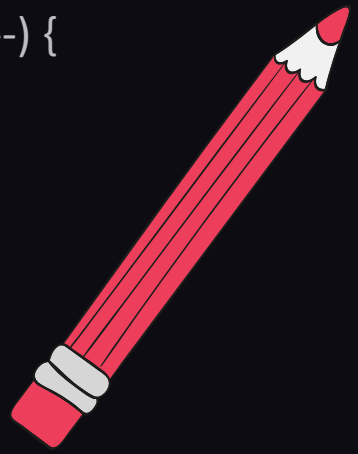
Average case

θ (N log (N))

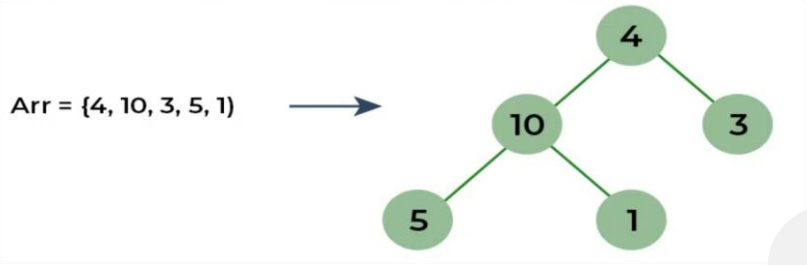
Worst case

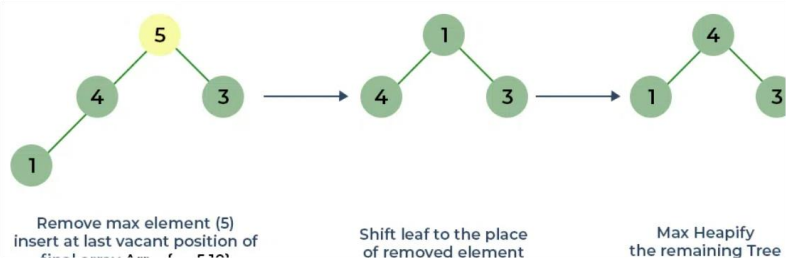
O(N log N)

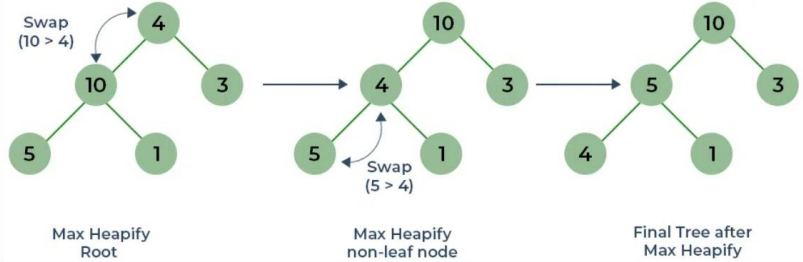
```java
// Heap Sort
public static void heapSort(int[] arr) {
    int n = arr.length;
    for (int i = n/2 - 1; i >= 0; i--) {
        heapify(arr, n,  i);
    }

    for (int i = n - 1; i >= 0; i--) {
        int temp = arr[0];
        arr[0] = arr[i];
        arr[i] = temp;
        heapify(arr, i, 0);
    }
```

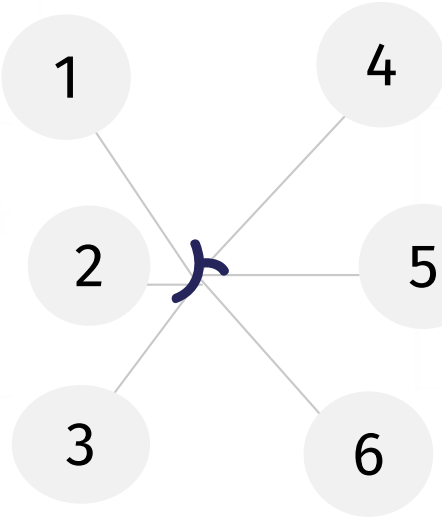


```java
private static void heapify(int[] arr,int n, int i) {
        int l = 2 * i + 1;
        int r = 2 * i + 2;
        int max = i;
        if (l < n && arr[l] > arr[max]) {
            max = l;
        }
        if (r < n && arr[r] > arr[max]) {
            max = r;
        }

        if (max != i) {
            int temp = arr[i];
            arr[i] = arr[max];
            arr[max] = temp;
            heapify(arr, n , max);
        }}}
```

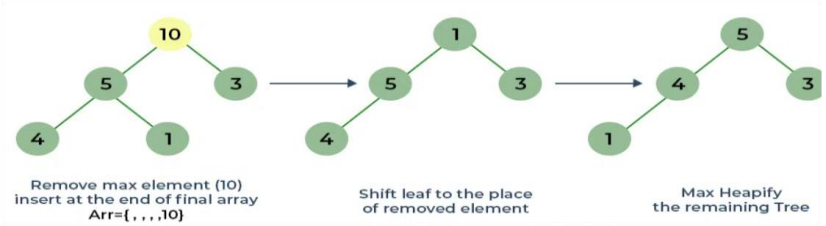

Arr = {4, 10, 3, 5, 1)
4
10
3
5
1

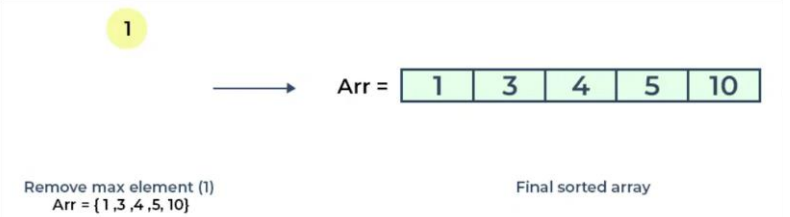

Remove max element (5)
insert at last vacant position of
Shift leaf to the place
of removed element
Max Heapify
the remaining Tree


Swap
(10 > 4)
Swap
(5 > 4)
Max Heapify
Root
Max Heapify
non-leaf node
Final Tree after
Max Heapify


1
2
3
4
5
6


Remove max element (3)
insert at last vacant position
of final array Arr = { ,3 ,4 ,5, 10}
Shift leaf to the place
of removed element.
(No heapify needed)


Remove max element (10)
insert at the end of final array
Arr={ , , , ,10}
Shift leaf to the place
of removed element
Max Heapify
the remaining Tree


Arr =
1
3
4
5
10
Remove max element (1)
Arr = { 1 ,3 ,4 ,5, 10}
Final sorted array

GUI