# Sorting | Time Complexity 2

## AGENDA:

- ✓ Optimised Bubble Sort
- ✓ Selection Sort
- ✓ Insertion Sort
- ✓ Mergesort (Logic only) – No code

Mergesort requires Recursion

# Optimised Bubble Sort

1. Do we need to go all the way till the end in every iteration ?
2. What if the array is already sorted ?

| | | | | | |
|---|---|---|---|---|---|
| 4 | 2 | 5 | 6 | 1 | 3 |
| 0 | 1 | 2 | 3 | 4 | 5 |

## Time Complexity

Best Case – When the array is sorted.

Array size = N

↳ $O(N)$

Worst Case – Array is shuffled

↳ $O(N^2)$

# Selection Sort

Selection sort is a sorting algorithm that sorts an array by repeatedly finding the minimum element and putting them in ascending order.

| Iteration No.<br>i | Array | Min index | Action |
|---|---|---|---|
| 0 | 4 2 5 6 1 3<br>0 1 2 3 4 5 | 4 | Swap a[0], a[4] |
| 1 | 1 2 5 6 4 3<br>0 1 2 3 4 5 | 1 | Swap a[1], a[1]<br>Effectively does nothing |
| 2 | 1 2 5 6 4 3<br>0 1 2 3 4 5 | 5 | Swap a[2], a[5] |
| 3 | 1 2 3 6 4 5<br>0 1 2 3 4 5 | 4 | Swap a[3], a[4] |
| 4 | 1 2 3 4 6 5<br>0 1 2 3 4 5 | 5 | Swap a[4], a[5] |

1 2 3 4 5 6
0 1 2 3 4 5

$(N-1) * N$

↓

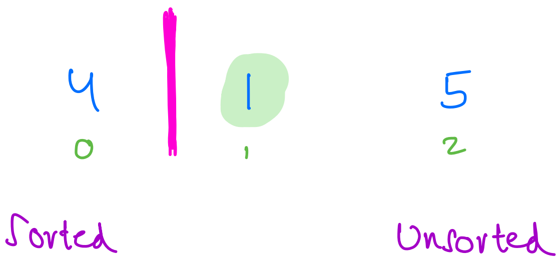
Time Complexity = $O(N^2)$

# Insertion Sort

In insertion sort, an array is divided into two sub-arrays: sorted and unsorted, where we compare and move every item from the unsorted part to the sorted part till the array is sorted.

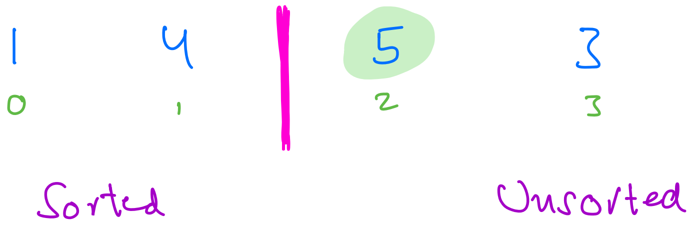
4 1 5 3 2

$i \rightarrow \text{range}(1, N)$

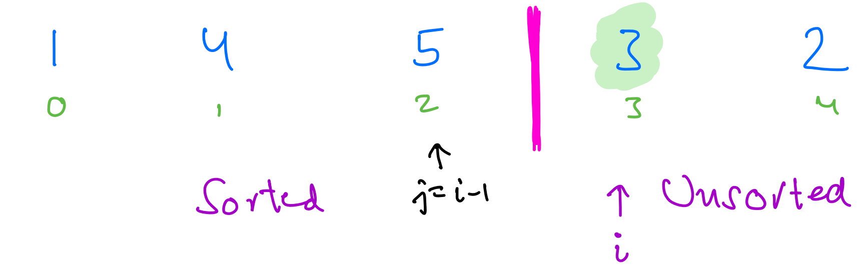


curr = a[1]

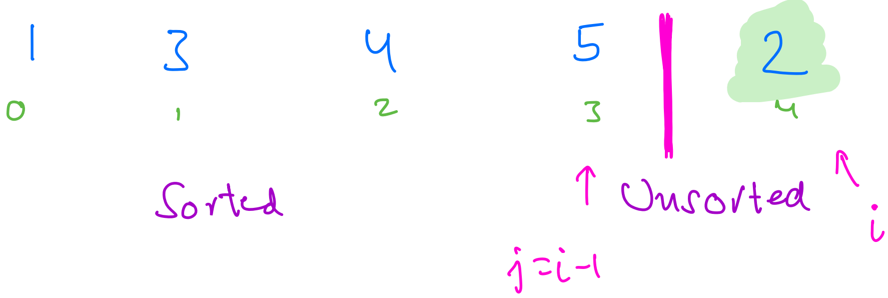
curr = 1



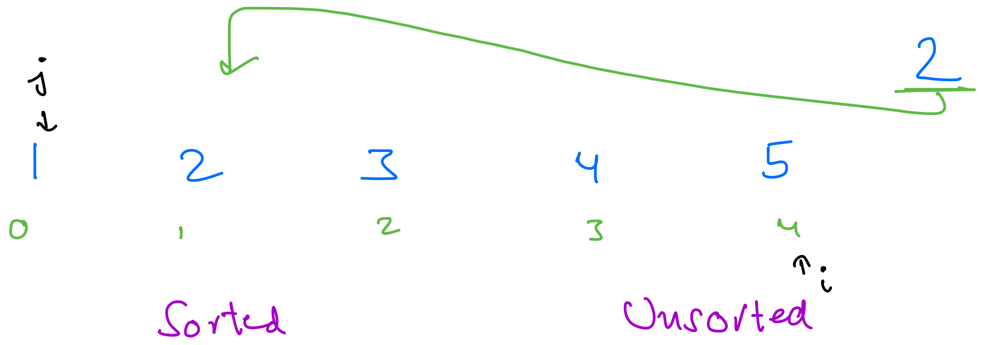
curr = a[2] = 5



curr = a[3] = 3
a[i]



curr = a[4] = 2



Move a[j] towards right side

```
while a[j] > curr :
    a[j+1] = a[j]
    j--;
a[j+1] = curr
```

Quiz

23, 12, 9, 22, 11, 10

Array after 3 iterations of insertion sort?

23 | 12, 9, 22, 11, 10

Array after 1 iteration 12, 23, | 9, 22, 11, 10

After 2nd iteration 9, 12, 23, | 22, 11, 10

After after 3rd iteration 9, 12, 22, 23 |, 11, 10

Time Complexity of Insertion Sort = $O(N^2)$

$$\left.\begin{array}{l}\text{Bubble}\\ \text{Selection}\\ \text{Insertion}\end{array}\right\} \quad O(N^2)$$

$$\left.\begin{array}{l}\text{Count}\\ \text{Radix}\\ \text{Bucket}\end{array}\right\} \quad O(N)$$

These only work in some conditions

$$\left.\begin{array}{l}\text{Most}\\ \text{other}\end{array}\right\} \quad O(N \log_2 N)$$

Mergesort

Quicksort

..... many more

# Mergesort

In merge sort, the given array is divided into roughly two equal sub-arrays. These sub-arrays are divided over and over again into halves until we end up with arrays having only one element each. At last, we merge the sorted pairs of sub-arrays into a final sorted array.

4 1 5 3 2

Pushing this to Recursion 2

# Doubts

Thank
You

Sorted Array → Binary Search

$A = [1, 3, 4, 4, 6]$

$B = 4$

$<= 4$

$A = [1, 2, 2, 2, 2, 2, 6, 6, 6]$
$B = 2$

$<= 2$ — $\boxed{6}$

$A = [1, 2, 2, 2, 2, 2, 6, 6, 6, 7]$
$B = 6$

$<= 6$

— $\boxed{9}$

---

$B = 2$

$A = [1, \underline{2, 2, 2, 2, 2}, 6, 6, 6]$

↑ First Occurrence of 2

↑ Last Occurrence of 2

Lower Bound = Index 1

Upper Bound = Index 5

To get lower bound & upper bound, you will have to slightly modify the binary search algorithm

Look for inbuilt methods in Python for binary search

---

Good Night

Thank You

Friday