Sorting | Time Complexity

Practice

Agenda

- Understand time complexity
- Big O
- Bubble sort

Time Complexity

Linear Search , Binary Search

Sorting - Bubble , Selection , Insertion , Merge , Quicksort

which is better ??

2 criteria - Time & Space

↓ Execution time

↓ Extra space

Asymptotic Analysis of Algorithms

Observing performance of algorithms for very large inputs.

↳ Big O

Calculating its Big O time complexity

How to compute Big O ?

1) Count the no of iterations (approx)

2) Get rid of small terms

3) Get rid of constant coefficient

Quiz 1

```
for i in range(1, N+1, 2)
```

(step = 2)

Eg - N = 20 → 1, 3, 5, 7, 9, 11, 13, 15, 17, 19

N = 20 → 10 iterations

Eg - N = 11 → 1, 3, 5, 7, 9, 11

N = 11 → 6 iterations

Input N → $\frac{N+1}{2}$ iterations → $O(N)$

Quiz 2

```
for i in range(1, N+1):        → N iterations
    for j in range(1, N+1):    } N iterations
        print(i + j)
```

Total = N^2 iterations → $O(N^2)$

Nested loops

Q3

```
for i in range(1, N+1):        } N
    print(i)
for j in range(1, N+1):        } N
    print(j)
```

Total = N + N = 2N iterations

Loops one after the other

$O(N)$

Big O practice

$F(N) = 9N^2 + \cancel{8N} + \cancel{82}$

↑ Biggest term

$9N^2$

Get rid of constant coefficient

$O(N^2)$

$F(N) = 10N^3 + 5N^2 \log N + 8N + 652$

↓

$10N^3$

Get rid of 10

$O(N^3)$

Quiz

$F(N) = N^2 + 10N$

$O(N^2)$ ✗

Quiz

$F(N) = N \log N + 7N^2 + 8$

↓

$7N^2$

Get rid of 7

$O(N^2)$

Quiz

$F(N) = 50N^2 + 14N \log N + 31N^2 \log N + 8$

↓

~~31~~ $N^2 \log N$

↓

$O(N^2 \log N)$

Binary Search

Best Case - 1 iteration

Worst Case - $\log_2 N$ iterations

Time Complexity = $O(\log_2 N)$

In maths

→ Asymptotic analysis

Big-O — Theta — Big omega Ω

Small-o — Small omega ω

TLE — Time Limit Exceeded.

Online Judge

10^8 iterations — 1 second

1 second of execution time

If your code runs over 1s → TLE

Constraints

Eg - Searching

Input size

$1 <= N <= 10^{10}$

Linear Search — O(N)

↳ Worst Case — 10^{10} iterations → TLE

✓ Binary Search — $O(\log_2 N)$

↳ Worst Case — $\log_2 10^{10}$ iterations = 34 iterations

10^8 iterations → 1 sec

1 iteration → $\frac{1}{10^8}$ sec

10^{10} iterations → $10^{10} \times \frac{1}{10^8}$ sec

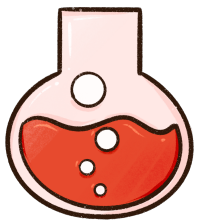
Unitary Method

$= 10^2$ sec

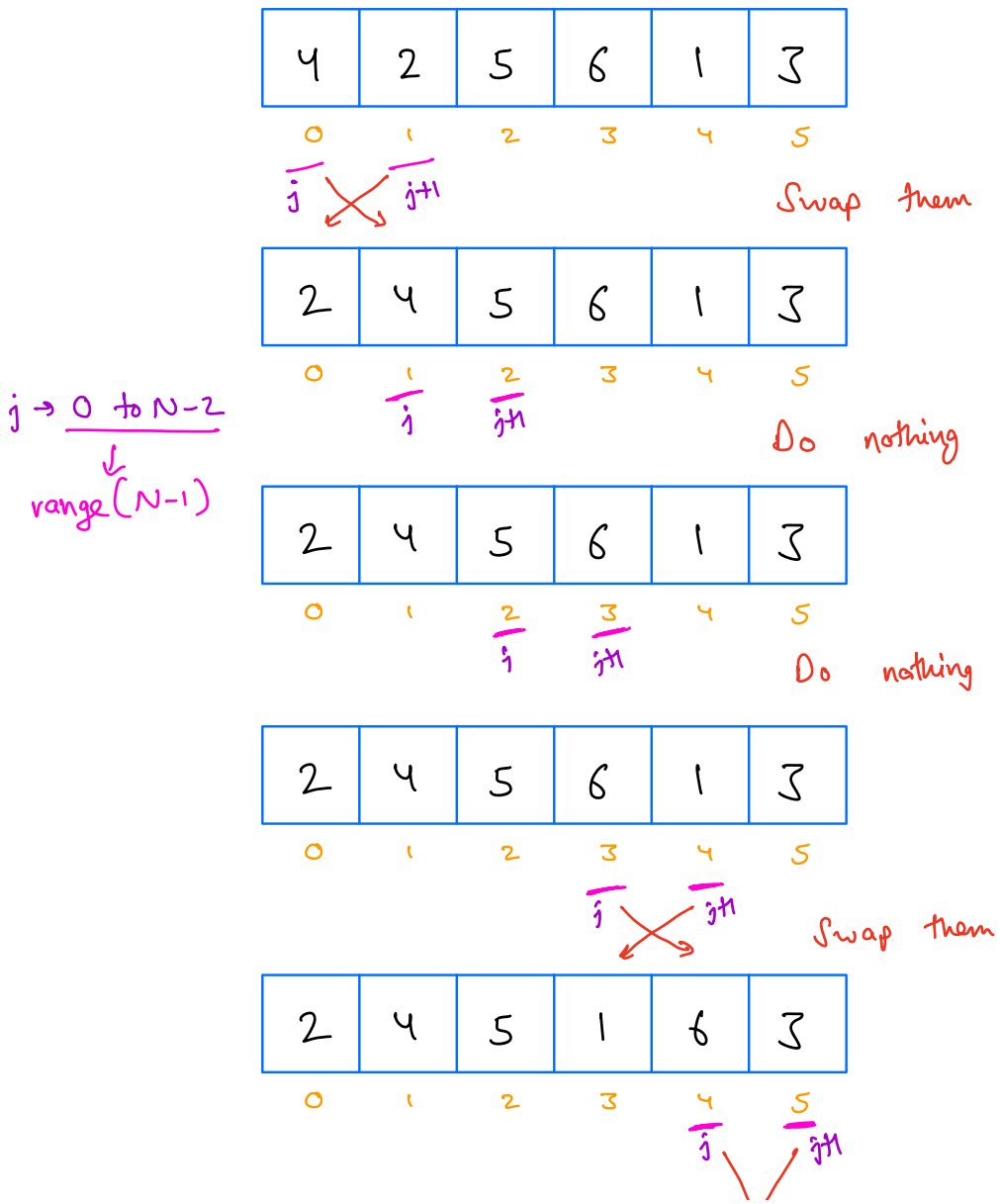
$= 100$ sec

Break till 10:13 PM

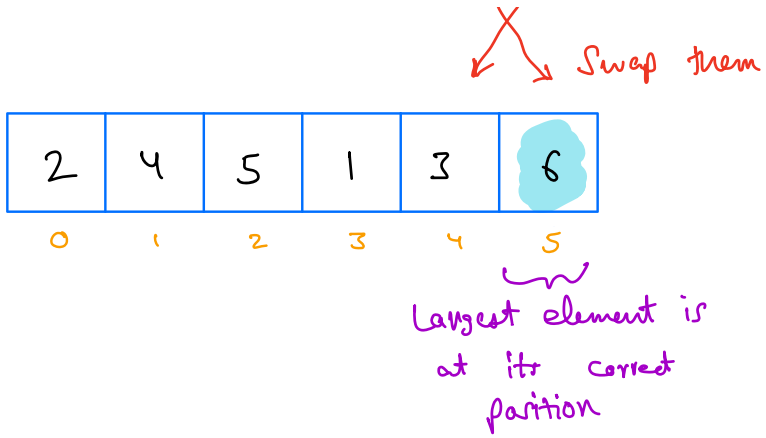
Bubble Sort



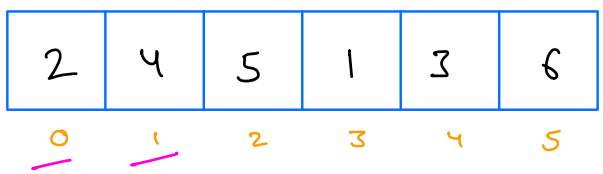
Bubble sort works on the repeatedly swapping of adjacent elements until they are not in the intended order. It is called bubble sort because the movement of array elements is just like the movement of air bubbles in the water. Bubbles in water rise up to the surface; similarly, the array elements in bubble sort move to the end in each iteration.



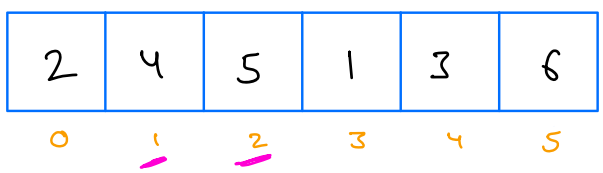
Completed 1st pass



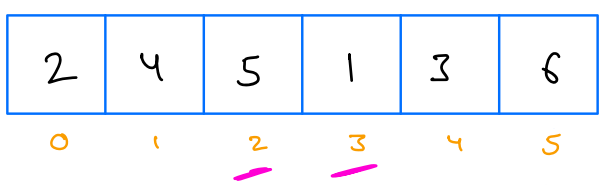
Starting 2nd pass



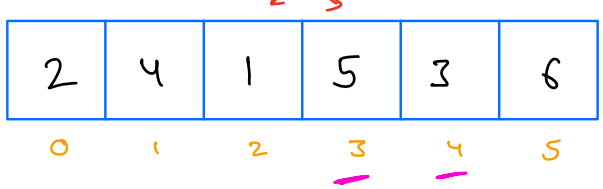
Do nothing



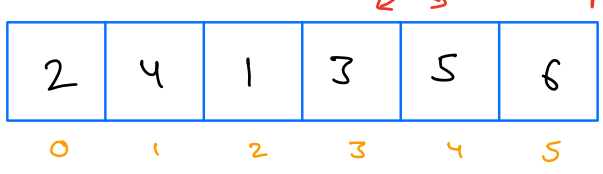
Do nothing



Swap them

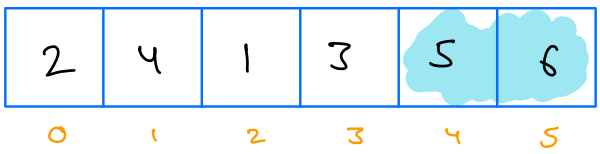


Swap them

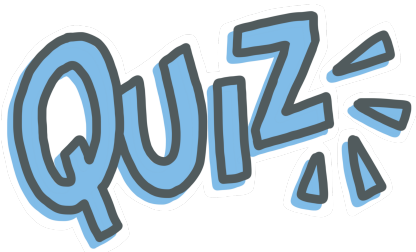


Do nothing

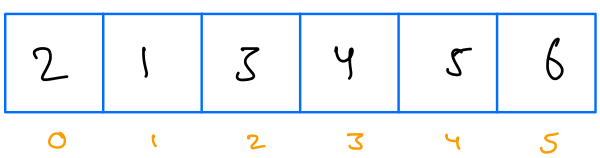
Array after 2nd pass



2 largest elements are at their correct positions



What will the array be after 3rd iteration ?



Last 3 elements at their correct positions

How many passes will be required to sort the entire array ? → N-1

```
def bubbleSort(arr):
    N = len(arr) - 1
    for i in range(N-1):
        for j in range(N-1):
            if arr[j] > arr[j+1]:
                arr[j], arr[j+1] = arr[j+1], arr[j]
```

1) 1
2) (N-1) iterations
3) N-1 iterations
4) 1
5) 1

Total iterations = $(N-1) * (N-1)$

$= N^2 - 2N + 1$ iterations

Time Complexity = $O(N^2)$

Doubts

Thank You

Interview – $O(N^2)$ Brute force

↓

$O(N \log N)$

↓

$O(N)$

Sorting – Bubble Sort → Quicksort / Mergesort

↙

Count Sort
Bucket Sort
Radix Sort

Inbuilt Python Sort – Timsort

↙ Insertion + ↘ Merge

$x^N > N^3 > N^2 > N > \log N$

Good Night

Thank You

Wednesday