05-09 Algorithmic Analysis

05 September 2022 21:52

What is an algorithm?

→ An algo. is step by step instructions to solve a given problem.

Q Prepare Maggi

1. Get a frying pan.
2. Take oil
3. Turn on the Stove
4. Add maggi masala

} Algo.

Q Reverse an array.

arr = 1 2 3 4 5

O/P = 5 4 3 2 1

1. Swap 1st & last elem.
2. Swap 2nd & 2nd last elem.

} algo.

---

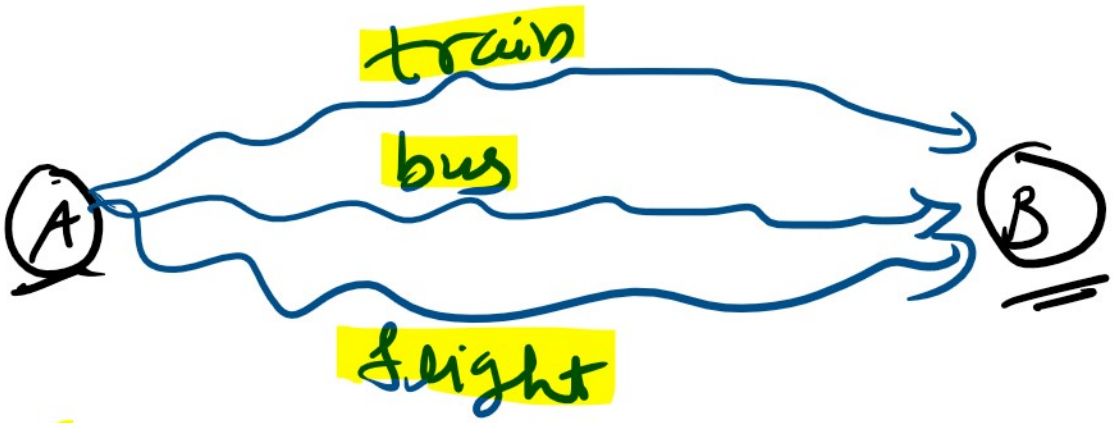
1. Read a problem Stat.
2. Devise an algo
3. Code an algo.

90%.

10%.

J → 3 days

VA → 15 mins



Speed	Cost	Comfort
flight	bus	flight
train	train	train
bus	flight	bus

* Goal of algo. analysis!-

→ The goal of algo. and. is to compare algo- in terms of.

↳ Time

↳ Space / memory.

Coding round

1. Correct Solⁿ.
2. Time Complexity.
3. Space Complexity.

100%

Q. How do we compare algo.?

(1) Experimental analysis XXX

Jitendra — Alg 1 → Compute (4 ms) → 8gb, i5, No GC
comp. time to exec. algo

Aneesh — Alg2 → Comp (3.65 ms) → 16gb, i7, Nvidia 4 gb

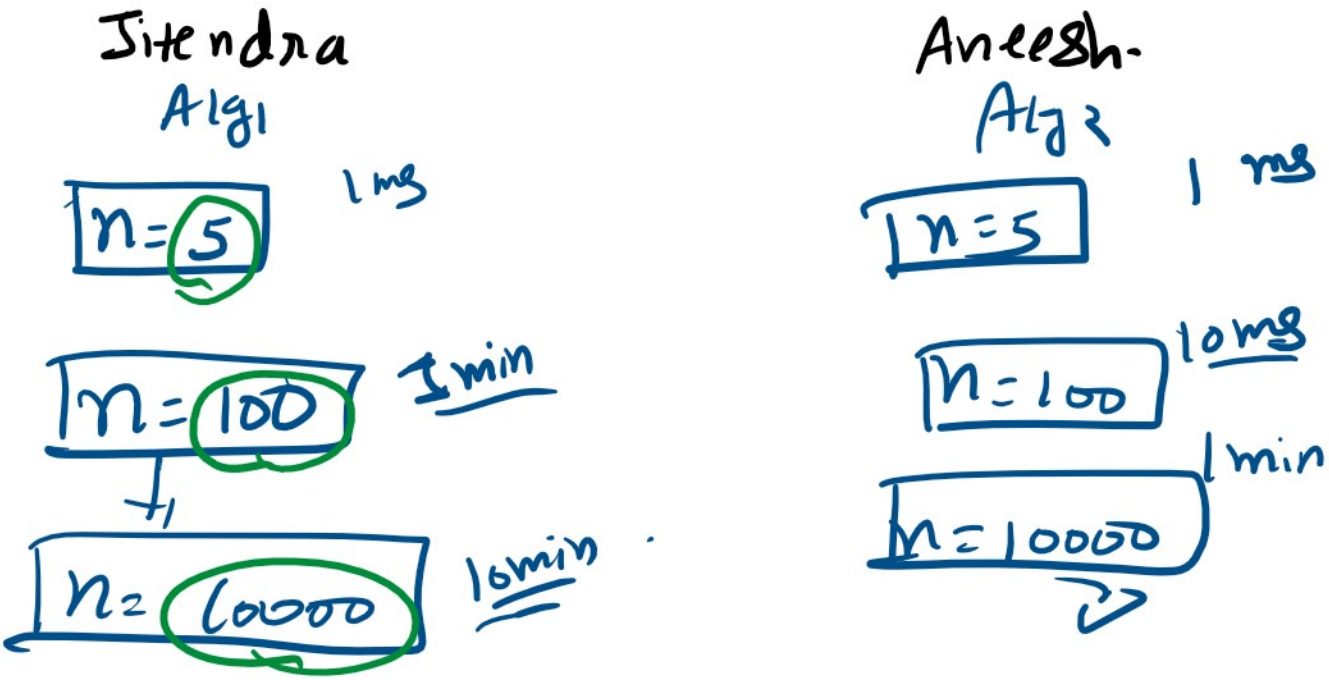
Abhishek. — Alg3 → 3 gms → 32gb, i5, No GC.

XXX

Cons:-
→ Solⁿ of machine
not good to evaluate.

2 Asymptotic analysis:-

↳ We do not depend on any machine



n=5 1 2 3 4 5 → 5 4 3 2 1

1. iterate through array → 5 steps
2. Compare 2 values. if(x > y)
   ↳ 1 step

Pros:-

1. time it will take / complexity is in terms of input size
2. machine-independent

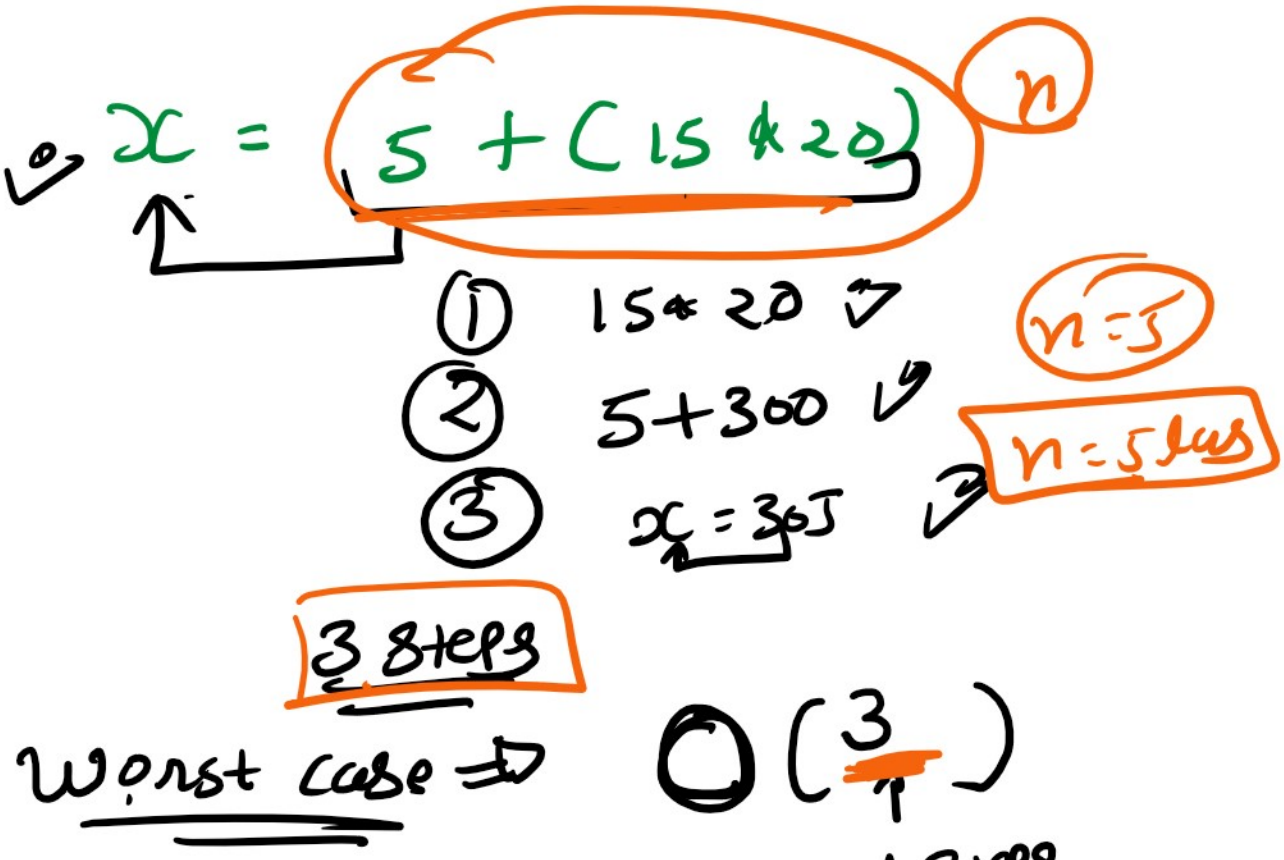
Types of measurement.

1. Worst case.
2. Best case.
3. Avg. case

best ⇒ flight always on time.

Worst ⇒ 1.5 hr late / 1 hr extreme delays

Avg ⇒ 10 ⇒ (5) ⇒ delay
(5) ⇒ Punctual



No. of steps

$O(3) \approx O(1)$

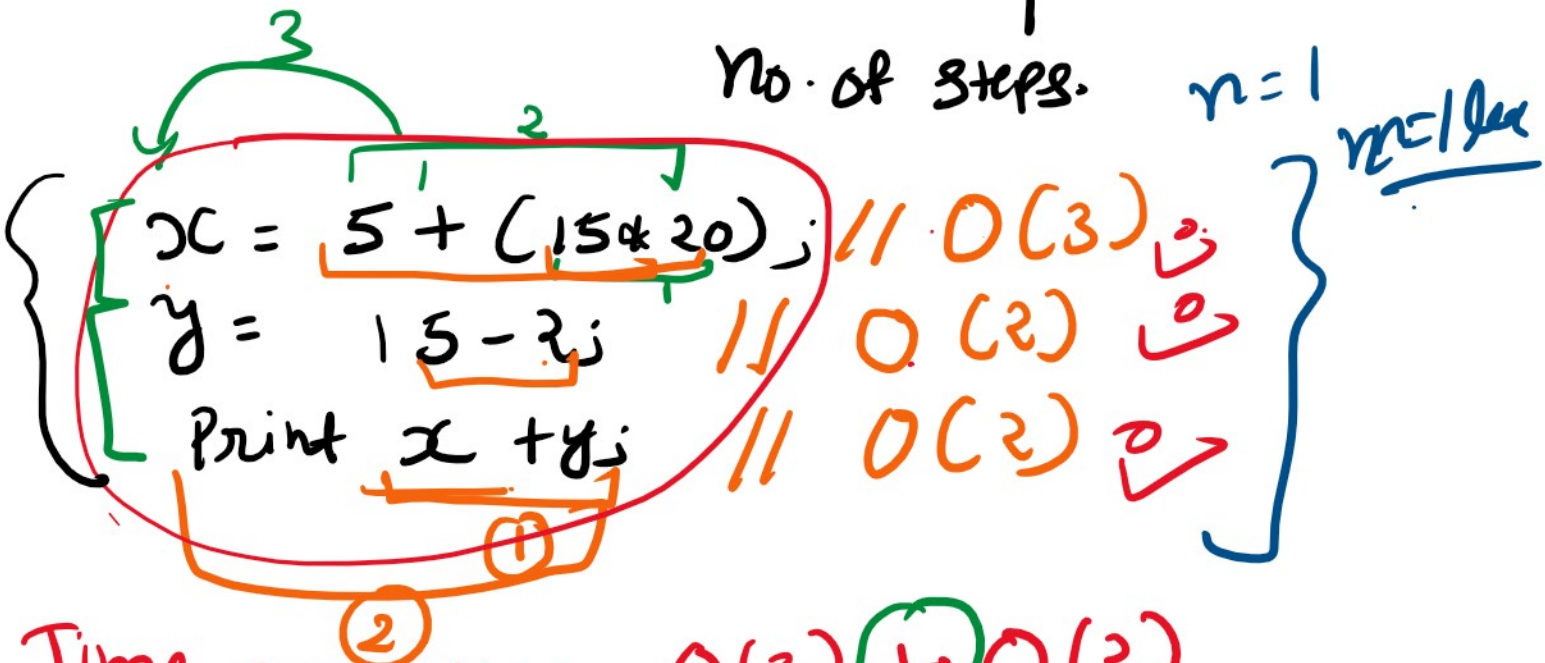
(2)

```
x = 5 + (15 * 20);
y = 15 - 2;
Print x + y;
```

Time Complexity :- Worst

$O(\quad)$

No. of steps.



Time complexity :- $O(3) + O(2) + O(2)$

$= O(7)$

$\approx O(1)$

1000000

2 O(1) 1

Q. for(i=0; i<n; i++){ } Steps?

Print(i)

}

Time Complexity. O( )

(?)

n=5

0
1
2
3
4

5 steps.

n=8

0
1
2
3
4
5
6
7

8 steps

n ⇒ no. of steps n

(n) => no. of steps n

O(n)

3 } 2 times
4 }

i=3 i<n; i++   n=5

```
for(i=1; i<n; i++) {
    Print(i)
}
```

n-3

O(n)

n+1

O(n+1)

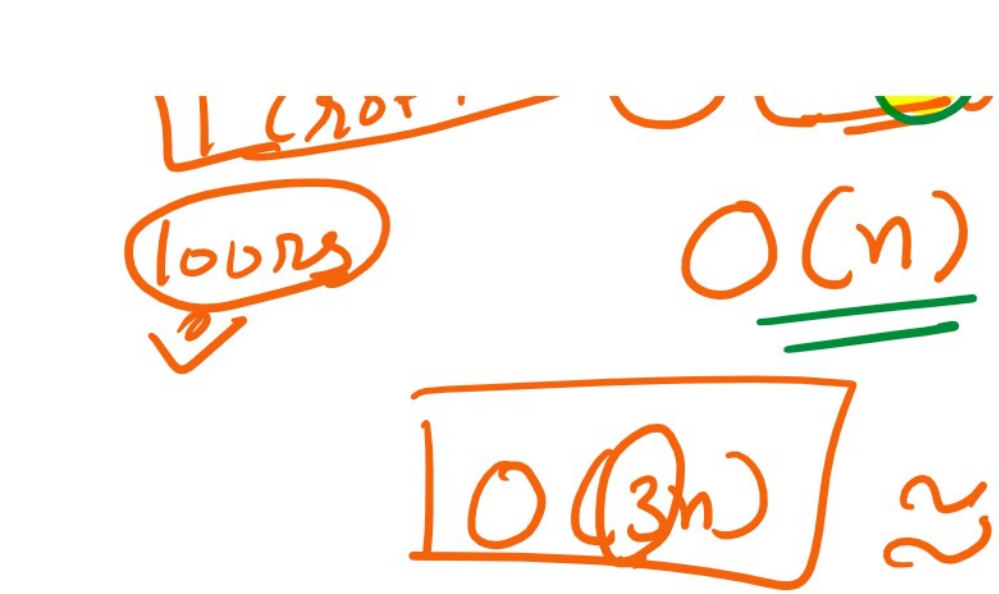
O(n)

n=5   1, 2, 3, 4 } 4 steps

n=8   1, 2, 3, 4, 5, 6, 7 } 7 steps

n => n-1 steps

1 (not loss) O(n-1) steps


hours
O(n)
O(3n) ≃ O(n)