What is algorithm analysis?

8 am Algorithm ~~of~~ analysis is a study to provide

9 am theoritical estimation for the required resources of an algorithm to solve a specific computation

10 am prob.

11 am i.e ⇒ Calculation efficiency ~~of~~

noon Generally the efficiency of an algorithm is related to the input length (no. of steps)

1 pm known as T.C and volume of memory is known as S.C.

2 pm

3 pm

Why do we need Algo analysis?

4 pm

:) Generally there are multiple approaches/methods

5 pm / algo to solve one problem statement.

evening Algo analysis is performed to figure out which is

7 pm better or optimal approach.

27
Saturday

26th Week • 179-187

June

MAY 2020								JUNE 2020							
WK	S	M	T	W	T	F	S	WK	S	M	T	W	T	F	S
18	31					1	2	23		1	2	3	4	5	6
19	3	4	5	6	7	8	9	24	7	8	9	10	11	12	13
20	10	11	12	13	14	15	16	25	14	15	16	17	18	19	20
21	17	18	19	20	21	22	23	26	21	22	23	24	25	26	27
22	24	25	26	27	28	29	30	27	28	29	30				

09.00 What does a BETTER Algorithm mean?

10.00 → Faster ? (Less time execution) → Time complexity → Most imp

important

11.00 → Less Memory - Space complexity

12.00 → Easy to read.

01.00 → Less ~~time~~ Line to code!

02.00 → Less hardware / software needs!

03.00

* Note: Algorithm Analysis does not give you accurate/exact values (time, space, etc), however it gives estimates which can be used to study the behavior of algorithm

04.00

05.00

06.00 What is Asymptoic Algorithm Analysis?

07.00

03.00

* Note: Algorithm Analysis does not give you accurate/exact values (time, space, etc), however it gives estimates which can be used to study the behavior of algorithm

04.00

05.00

06.00

What is Asymptotic Algorithm Analysis?

07.00

-> Defination: In mathematical analysis, asymptotic analysis of algorithm is a method of defining the mathematical boundaries of its run-time performance.

28 Sunday

-> Using the asymptotic analysis, we can easily estimate about average case, best case and worst case scenario of an algo.

NOTES : APPOINTMENTS : PHONE/E-MAIL :

09.00 Simple words:

10.00 It is used to mathematically calculate the

11.00 running time of any operation inside an algorithm

12.00

Asymptotic Algorithm analysis is to estimate the time complexity

01.00 function for arbitary large input.

02.00

03.00 **Time Complexity:** is a computational way to show how (behavior)

04.00 runtime of a program increase as the size of its input

increase

05.00

06.00

07.00

Problem statement :-

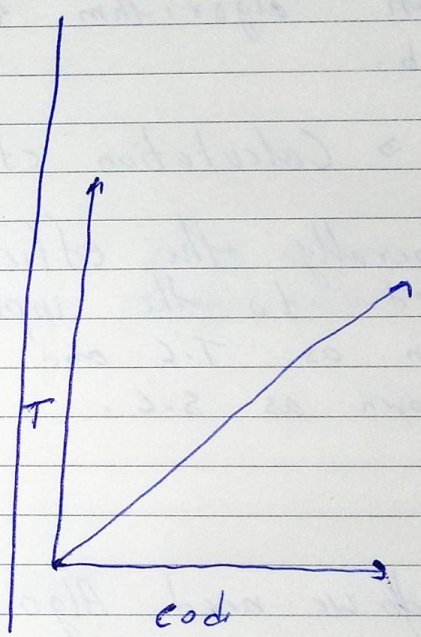
8 am Write an algo to find the sum of n numbers

9 am

Algo - 1

10 am

```
function sumofNum(N){
    sum = 0;          -> 1 usec
    for(i=0 to N){    -> n usec
        sum = sum + i;  -> n usec
    }
    print(sum)        -> 1 usec
```

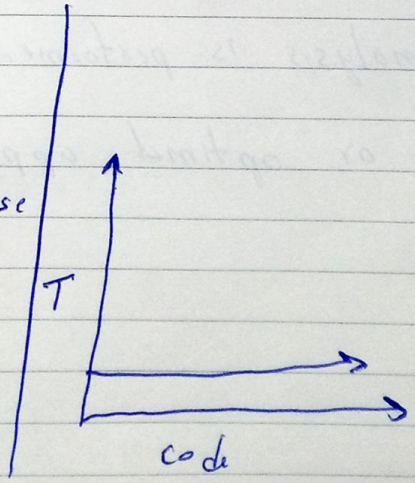


4 pm

5 pm Algo -2

evening

```
function sum(N){
    Sum = (N*(N+1))/2  -> 2usec
    print(sum)         -> 1 usec
}
```



09.00 # What is Space Complexity :-

10.00

Definition :- The space complexity of an algorithm or a computer program is the amount of memory space required to solve an instance of the computational problem as a function of the size of input.

11.00

12.00

01.00

Simple Words :- It is the memory required by an algorithm to execute a program and produce output.

02.00

03.00 similar to time complexicity, Space complexicity is often expressed asymptotically in big O notation, such as $O(n)$, $O(n\log(n))$, $O(n^2)$

04.00 etc, where n is the input size in units of bits needed to represent the input.

05.00

06.00 For any algorithm, memory is required for the following purposes -

07.00

- To store program instructions.
- To store constant values.
- To store variable values.
- and few other things like function call, jumping statements, etc.

Auxilary Space : is the temporary space (excluding the input size) allocated by your algorithms to solve the problem, with respect to input size.

NOTES : APPOINTMENTS : PHONE/E-MAIL :

Space complexity includes both Auxilary space and space used by input.

∴ Space Complexity = Input Size + Auxilary Size.

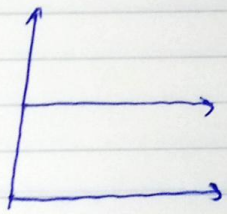
Space Complexity :-

Size of int unasigned 2 byt
asigned 4 byt

Algo - 1

```
function add(n1, n2){
    int sum = n1 + n2
    return sum
}
```

n1 = 4 byt
n2 = 4
sum = 4
Aux spa = 4
16 byt

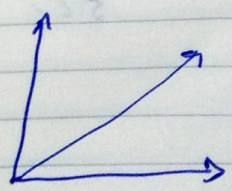


constat S.C

Algo - 2 Sum of all elements in array

```
function sumofNumbers(arr[], N){
    int sum = 0;
    for(i=0; to N){
        sum = sum + arr[i]
    }
    print(sum)
}
```

arr = N * 4 byt
sum = 4
i = 4
Aux = 4
4N + 12
=> 4N + C



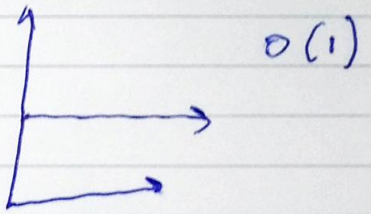
space complexity = 1 / Time comp.

Algo 3 -> Fact of no. (iterative)

```
int fact = 1
for(int i = 1; i<n; i++)
    fact *= i
return fact
```

fact = 4 but
n = 4
i = 4
Aux = 4

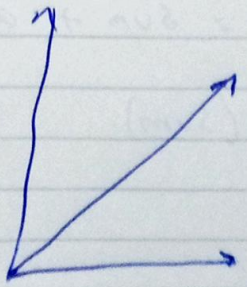
12 but 4 4 = 16



Algo-4 factorial of a number (recursive)

```
if (n<=1)
    return 1;
else
    return (n * factorial(n-1))
```

S.C: 4 + n
Aux



Analysis of Algorithm

Runtime → Time it takes to execute a piece of code.

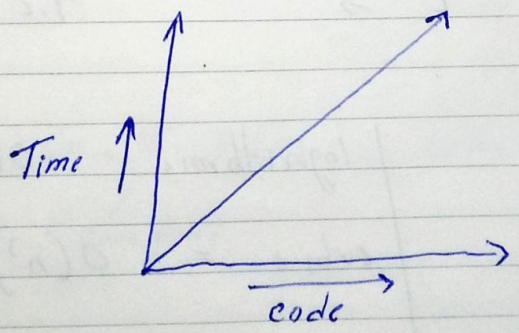
How does the runtime of a function grow?

→ Big O_n Notation and Time complexity.

Time complexity → A way of showing how runtime of a function increase as size of code increase.

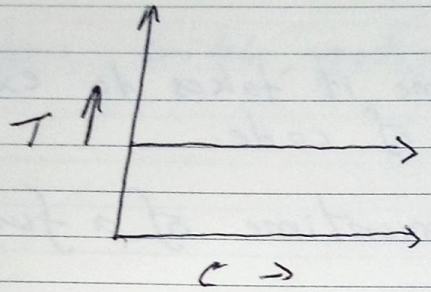
Types :-

i) Linear → if time is linearly / directly dependent on complexity of code



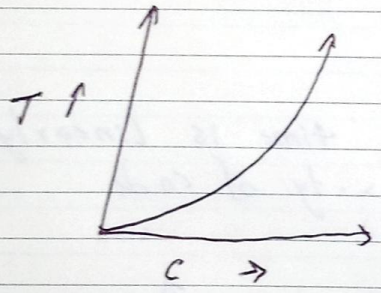
T.C → O(n)

ii) Constant → if



T.C = O(1)

iii) Quadratic →



T.C = O(n²)

Linear → O(n)

Constant → O(1)

Quadratic → O(n²)

logarithmic = O(log n)

cubic = O(n³)

How to find big Oh -> 1. Find fastest growing term
2. Eliminate const. growing term

8 am

Big Oh(O) vs Big Omega(Ω) vs Big Theta(Θ)

9 am

10 am

Big O notation(O)

$1 < \log(n) < \text{sqrt}(n) < n < n\log(n) < n^2 < n^3 < n^n$

11 am

-> Describes the worst case scenario.

noon

-> Represent upper bound running time complexity of an algo.

1 pm

2 pm

Mathematically :-

3 pm

Let f and g be functions of n -> where n is natural no denoting size or steps of algo then,

4 pm

$$f(n) = O(g(n))$$

after particular steps this holds true

5 pm

Iff

evening

$f(n) <= c \cdot g(n)$, where $n >= n_0$

c = constant

7 pm

$c > 0$

$n_0 >= 1$

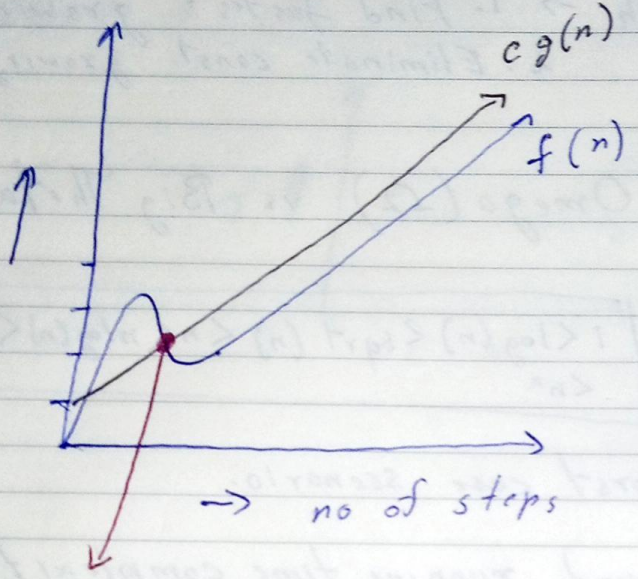
n_0 = no. of steps

JANUARY 2021

08 FRIDAY

02 Wk / 008-357

O(n) → is generally linear type



f(n) <= c g(n) satisfied

Example :-

$f(n) = 2n + 3$

$g(n) = n$

Formula → $f(n) <= c \cdot g(n)$

$2n + 3 <= c \cdot n$

let $c = 5$ $n = 1$

$2(1) + 3 <= (5)(1)$

→ $5 <= 5$

$n = 2$ $c = 5$

$2(2) + 3 <= (5)(2)$ → $7 <= 10$

But if we have multiple test case of Big oh we have to consider graph close to f(n)

Ex → n^2 vs n → n will chose

so chose value close f(n)

Big Omega (Ω) [opposite to Big oh (O)]

-> Specify the best schenarios

-> It represents the lower bound running time complexity of an algorithms.

-> Basically it tells us what is the fastest time/behavior in which the algo can run.

Mathematically :-

Let f and g be function of n, where n is natural number denoting size or steps of the algo. Then :-

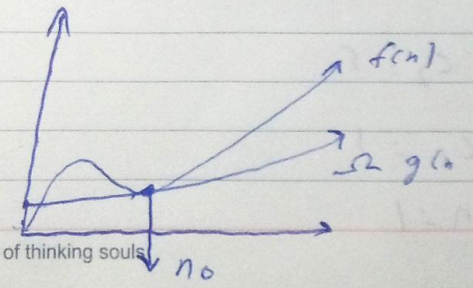
$$f(n) = \Omega(g(n)) \Rightarrow f(n) >= C \cdot g(n)$$

Sunday 10

$f(n) = 2n+3 \qquad g(n) = n$

$\Rightarrow 2n+3 >= C \times n \qquad$ let $C = 1$

$\Rightarrow 5 >= \underline{1}$



Big Theta (Θ)

8 am -> Describe average case scenario

9 am -> Represent most realastic Time complexity of an algo.

10 am

11 am Mathematically :-

noon Let f and g be functions of n, where n is natural no. denoting size or steps of algorithm.

1 pm

$$f(\Theta) = \Theta(g(n))$$

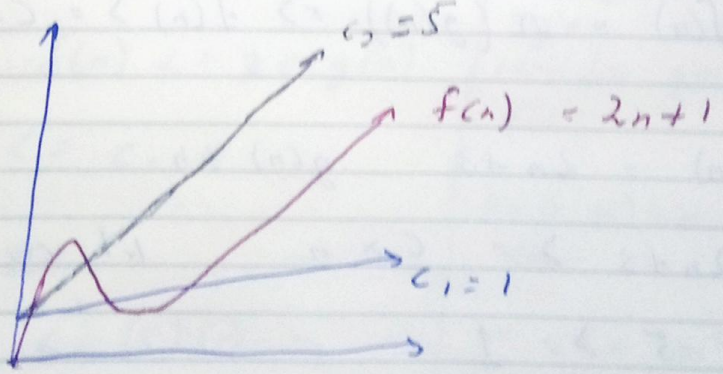
2 pm

T F F

3 pm

$$c_1 \cdot g(n) <= f(n) <= c_2 \cdot g(n)$$

4 pm



5 pm

$c_2 = 5$

evening

$c_1 = 1$

7 pm

$n = 1$