DSA through C++

Time Complexity

Saurabh Shukla (MySirG)

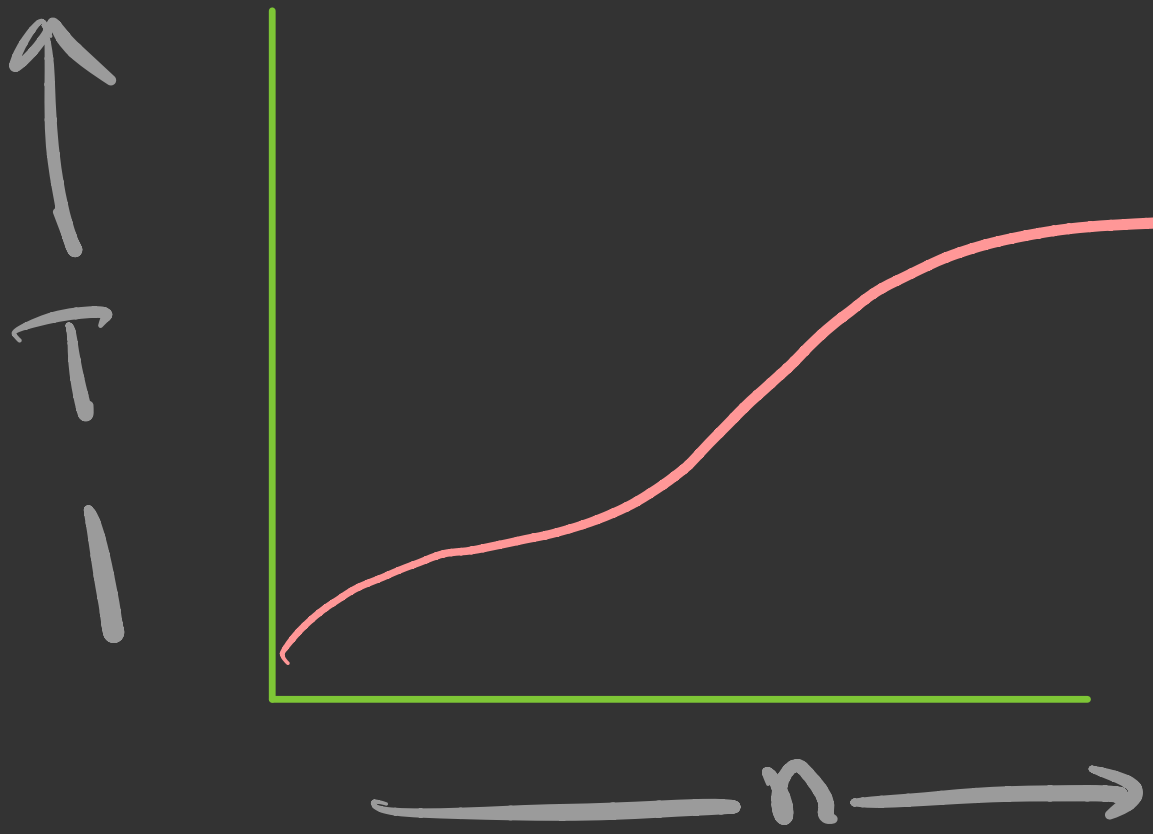
Agenda

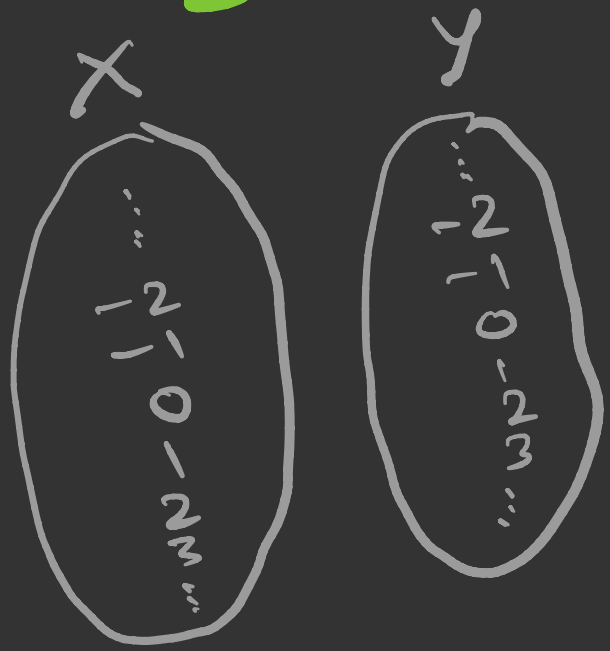
1. Basics of function
2. Measuring Algorithm Performance
3. Why analysis is required
4. Types of algorithm analysis
5. Time Complexity
6. Asymptotic Analysis
7. Asymptotic Notations
8. Calculation Big O notation

Basics of Functions

$T = f(n)$ [$y = f(x)$]

$y = 3x$





Measuring Performance of Algorithms

Algorithm analysis is an important part of computational complexity theory, which provides theoretical estimation for the required resources of an algorithm to solve a specific computational problem.

It is the determination of the amount of time and space resources required to execute it.

Why Analysis is required?

- Predicting behavior of an algorithm without implementing.
- Analysis is only approximation There are many influencing factors
- It helps us in determining the best algorithm to solve a programming problem.

Types of Algorithm Analysis

1. Best Case
2. Worst Case
3. Average case

- Certain input takes less time for an algorithm to accomplish its task. This is best case
- Certain input takes max time for an algorithm to accomplish its task. This is Worst case.
- All other cases / total no. of cases = Average case

Time Complexity

It is a measure of rate of change in time with respect to change in input size.

Various asymptotic notations are there to represent time complexity of an algorithm.

Asymptotic Analysis

पास रहते हुए भी कभी न मिलने वाली स्थिति।

It is used to analyze performance of algorithms in terms of input size

We donot calculate the actual time taken by the algorithm to solve the problem, rather we are interested in how the time taken by an algorithm increases with the input size.

Asymptotic Notation

Big O notation

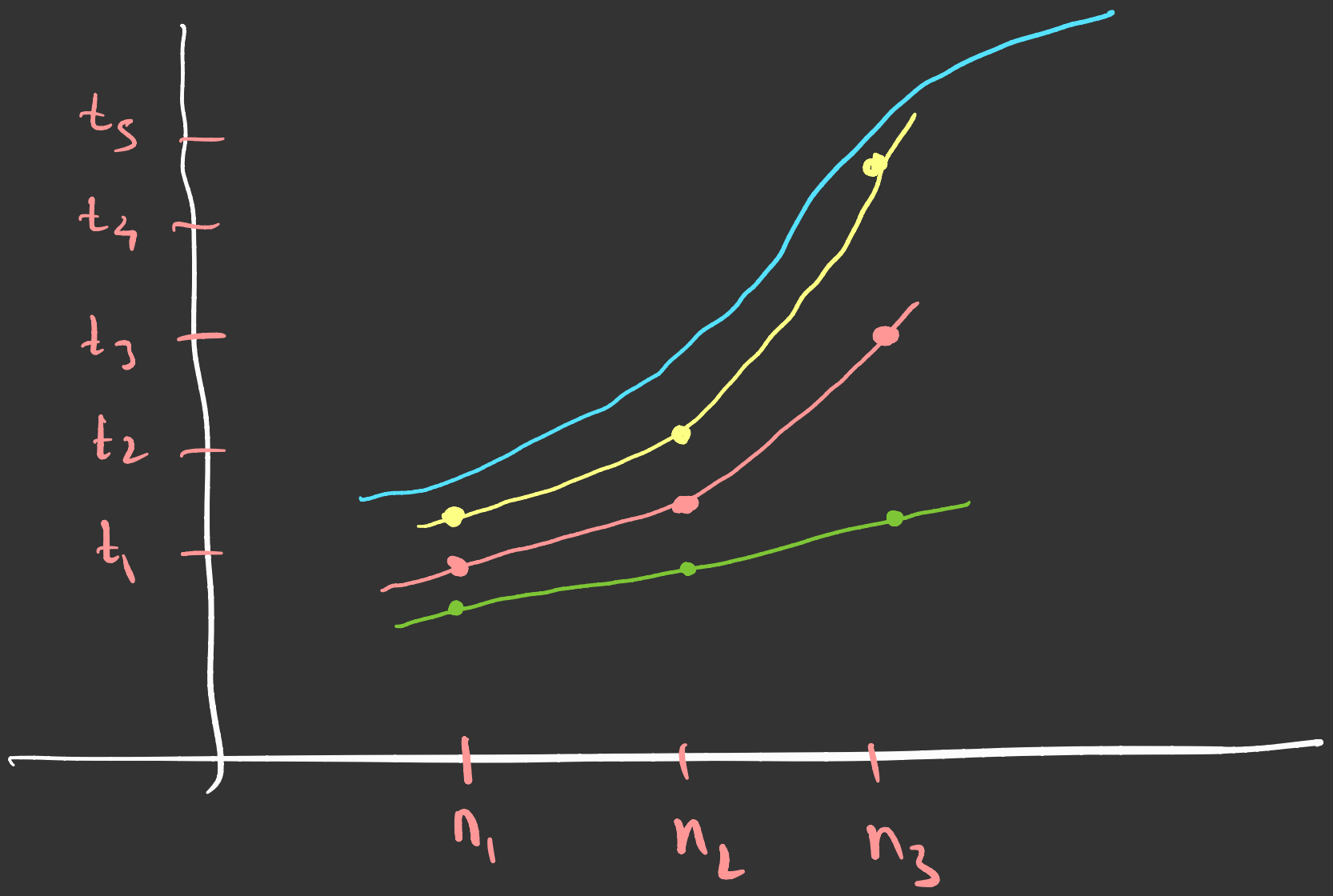
provides an upper bound on the growth rate of time. It represent the worst case scenario.

Omega notation

provides a lower bound on the growth rate of time. It represents the best case scenario

Theta notation

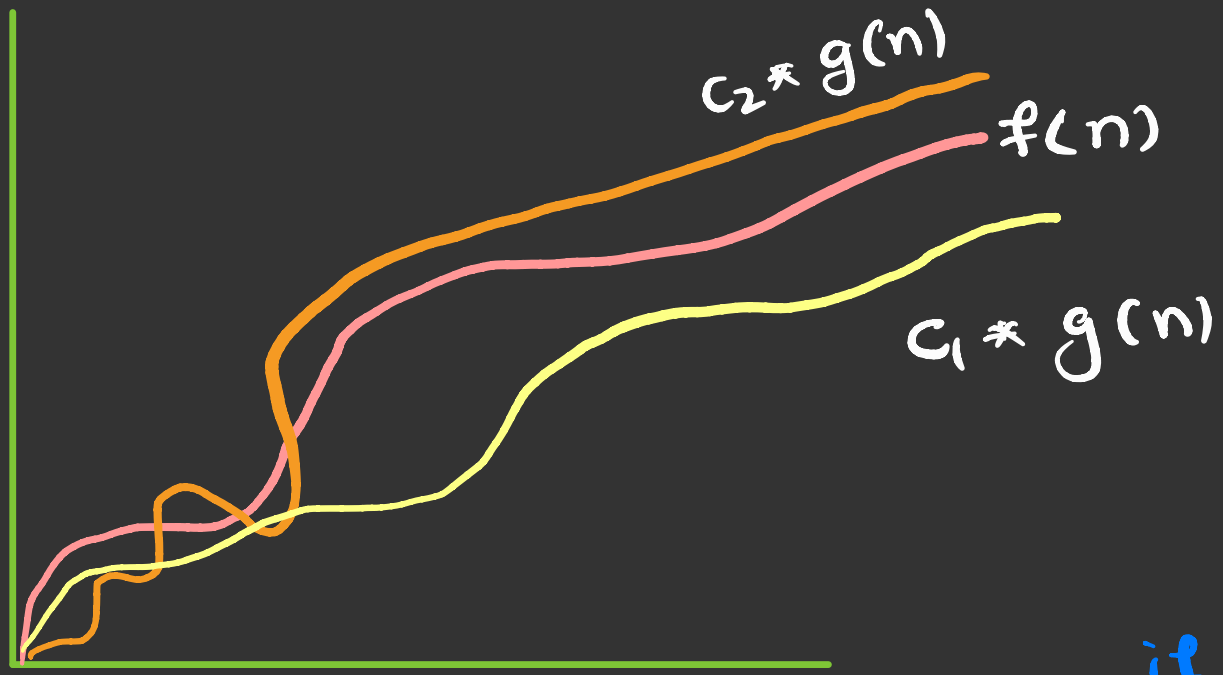
provides both an upper bound and lower bound on the growth rate of time. It represents the average case scenario.


t_5
t_4
t_3
t_2
t_1
n_1
n_2
n_3

Theta Notation

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

$c_1 * g(n) \leq f(n) \leq c_2 * g(n)$
c_1, c_2 are +ve constants
for all $n \geq n_0$

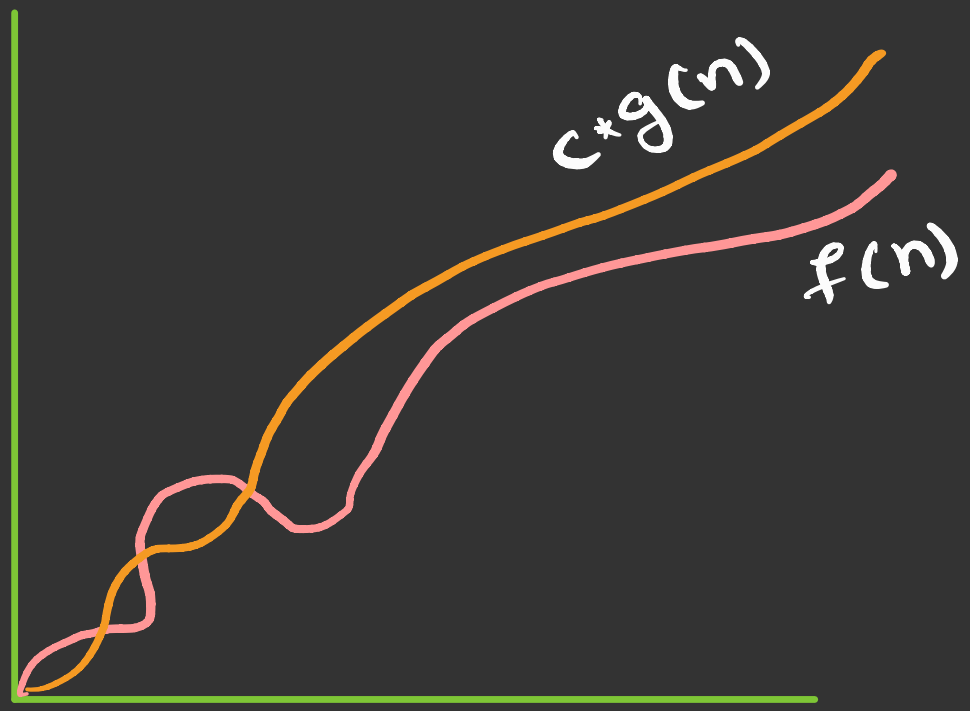


if $g(n) = 3n^3 + 5n^2 + 4$
then
$g(n) = n^3$
for some n_0
such that $n \geq n_0$

Big O notation

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

$0 \leq f(n) \leq c\,g(n)$
for all $n \geq n_0$

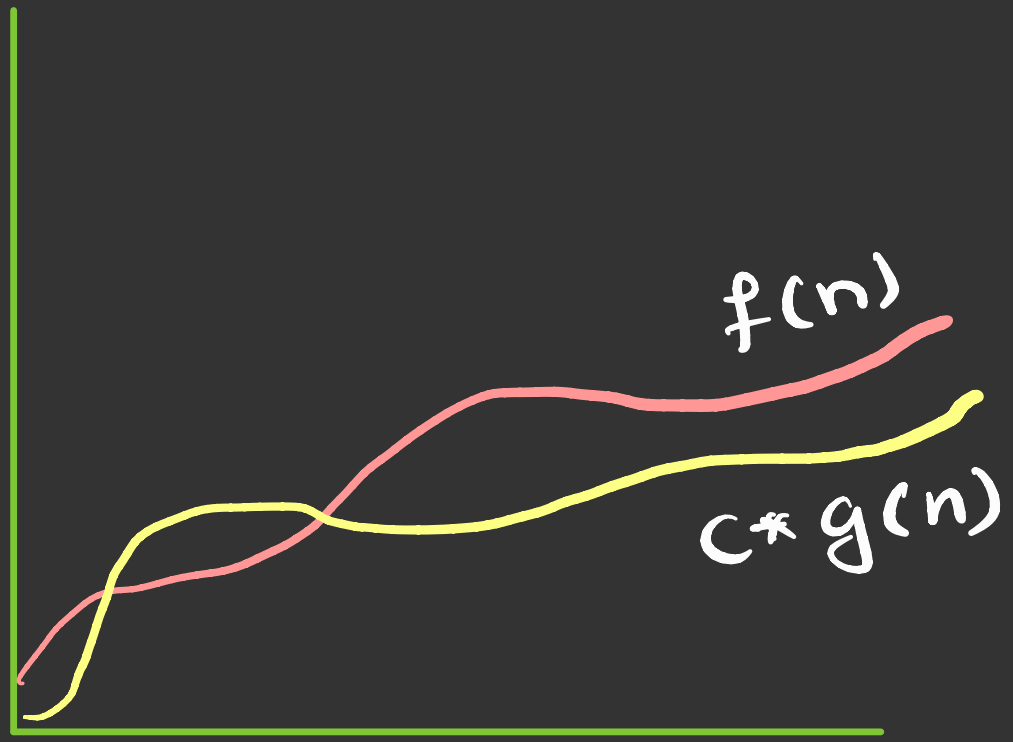


Omega Notation

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

$c * g(n) \leq f(n)$

for all $n \geq n_0$



Calculate O(n)

1. Factorial
2. Insertion Sort

```
f=1                          t1
for  i=1 to n
    f= f*i ;                 t2
```

$t_1 + n\,t_2$

$O(n)$

```
for( i=1 to n)
    temp= A[i]
    for (j= i-1 to 0)
        if (A[j] > temp)
            A[j+1]=A[j];
        else
            break;
    A[j+1] = temp;
```

$j = n(n)$

$n \times t_1$

$O(n^2)$

$O(1)$ $O(\log n)$ $O(n)$ $O(n \log n)$

$O(n^2)$ $O(n^2 \log n)$ $O(n^3)$

$O(a^n)$