Asymptotic Analysis of algorithms (Growth of function)

Resources for an algorithm are usually expressed as a function regarding input. Often this function is messy and complicated to work. To study Function growth efficiently, we reduce the function down to the important part.

```
Let f (n) = an²+bn+c
```

In this function, the n^2 term dominates the function that is when n gets sufficiently large.

Dominate terms are what we are interested in reducing a function, in this; we ignore all constants and coefficient and look at the highest order term concerning n.

⇧

Asymptotic notation:

The word **Asymptotic** means approaching a value or curve arbitrarily closely (i.e., as some sort of limit is taken).

Asymptotic analysis

It is a technique of representing limiting behavior. The methodology has the applications across science. It can be used to analyze the performance of an algorithm for some large data set.

1. In computer science in the analysis of algorithms, considering the performance of algorithms when applied to very large input datasets

The simplest example is a function $f(n) = n^2+3n$, the term 3n becomes insignificant compared to n^2 when n is very large. The function "$f(n)$ is said to be **asymptotically equivalent** to n^2 as $n \to \infty$", and here is written symbolically as $f(n) \sim n^2$.

Asymptotic notations are used to write fastest and slowest possible running time for an algorithm. These are also referred to as 'best case' and 'worst case' scenarios respectively.

"In asymptotic notations, we derive the complexity concerning the size of the input. (Example in terms of n)"

"These notations are important because without expanding the cost of running the algorithm, we can estimate the complexity of the algorithms."

Why is Asymptotic Notation Important?

1. They give simple characteristics of an algorithm's efficiency.

2. They allow the comparisons of the performances of various algorithms.

⇧

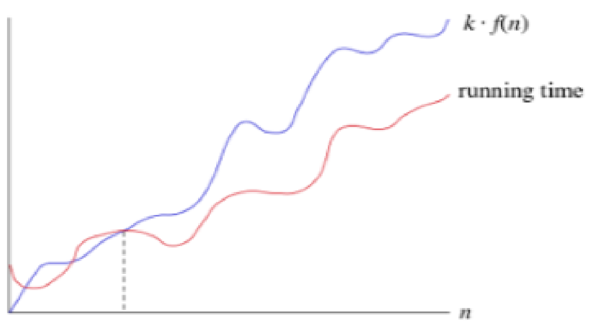
Asymptotic Notations:

Asymptotic Notation is a way of comparing function that ignores constant factors and small input sizes. Three notations are used to calculate the running time complexity of an algorithm:

1. Big-oh notation: Big-oh is the formal method of expressing the upper bound of an algorithm's running time. It is the measure of the longest amount of time. The function **f (n) = O (g (n))** [read as "f of n is big-oh of g of n"] if and only if exist positive constant c and such that

> f (n) ⩽ k.g (n)f(n)⩽k.g(n) **for** n>n0n>n0 in all **case**

Hence, function g (n) is an upper bound for function f (n), as g (n) grows faster than f (n)



ASYMPTOTIC UPPER BOUND

For Example:

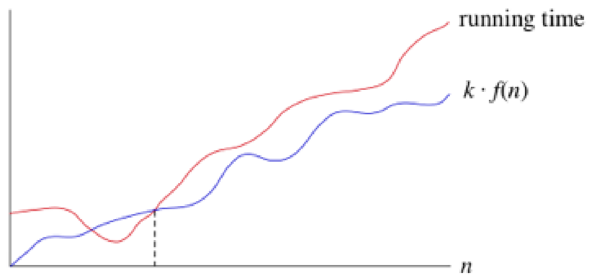
> 1. 3n+2=O(n) as 3n+2≤4n **for** all n≥2
> 2. 3n+3=O(n) as 3n+3≤4n **for** all n≥3

Hence, the complexity of **f(n)** can be represented as O (g (n))

2. Omega () Notation: The function f (n) = Ω (g (n)) [read as "f of n is omega of g of n"] if and only if there exists positive constant c and n_0 such that

```
F (n) ≥ k* g (n) for all n, n≥ n0
```

⇧



ASYMPTOTIC LOWER BOUND

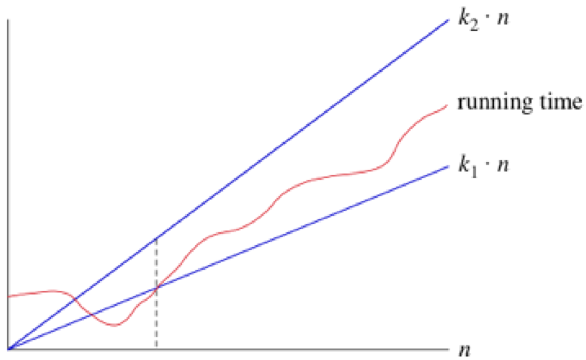
For Example:

```
  f (n) =8n²+2n-3≥8n²-3
        =7n²+(n²-3)≥7n² (g(n))
Thus, k₁=7
```

Hence, the complexity of **f (n)** can be represented as Ω (g (n))

3. Theta (θ): The function f (n) = θ (g (n)) [read as "f is the theta of g of n"] if and only if there exists positive constant k_1, k_2 and k_0 such that

```
  k₁ * g (n) ≤ f(n)≤ k₂ g(n)for all n, n≥ n₀
```



ASYMPTOTIC TIGHT BOUND

For Example:

```
3n+2= θ (n) as 3n+2≥3n and 3n+2≤ 4n, for n
    k1=3,k2=4, and n0=2
```

Hence, the complexity of f (n) can be represented as θ (g(n)).

The Theta Notation is more precise than both the big-oh and Omega notation. The function f (n) = θ (g (n)) if g(n) is both an upper and lower bound.

← prev next →

European experiences and educate who experience with children

Ad we provide the children with Academic,Behavior,Psychological skills.

Happy Jump Nurseries

Learn more

Please Share

Join Javatpoint Test Series

Placement Papers	AMCAT	Bank PO/Clerk	GATE
TCS	eLitmas	UPSSSC	NEET
HCL	Java	Government Exams	CAT
Infosys	Python	SSC	Railway
IBM	C Programming	Civil Services	CTET
Accenture	Networking	SBI	IIT JEE

Learn Latest Tutorials

NGINX

SSRS

Blockchain

ETL

Bugzilla

Hive

Jenkins

Pytorch

QA

Agile

JIRA

Tableau

⇧

Preparation

Aptitude

Reasoning

Verbal A.

Interview

Company

Trending Technologies

AI

AWS

Selenium

IoT

Cloud

Hadoop

ReactJS

React Native

Node.js

D. Science

Angular 7

B.Tech / MCA

DBMS

DS

DAA

OS

C. Network

Compiler D.	COA	D. Math.	E. Hacking	C. Graphics
Software E.	Web Tech.	Cyber Sec.	Automata	C
C++	Java	.Net	Python	Programs
Control S.				

⇧

⇧