6

GROWTH OF FUNCTIONS

rt-1. Functions for Computer Science

Functions play an important role in study of algorithms and their analysis. Here we iscuss some mathematical functions which appear very often in computer science, gether with their notations.

Floor Function : Let x be any real number. Then floor function of x means greatest teger which is less than or equal to x. Floor function is denoted by $\lfloor x \rfloor$.

Example : $\lfloor 3\cdot 14 \rfloor = 3$

$$\lfloor \sqrt{5} \rfloor = 2$$

$$\lfloor -9\cdot 2 \rfloor = -10$$

$$\lfloor 17 \rfloor = 17$$

Ceiling Function : Let x be any real number. Then Ceiling function of x means east integer which is greater than or equal to x. Ceiling function is denoted by $\lceil x \rceil$.

Example : $\lceil 3\cdot 14 \rceil = 4$

$$\lceil \sqrt{5} \rceil = 3$$

$$\lceil -9\cdot 2 \rceil = -9$$

$$\lceil 17 \rceil = 17.$$

Remarks : (1) On number line floor function is taken as left integer of x and Ceiling unction is right integer of x or it may be x itself if x is integer.

(2) If x is an integer then $\lfloor x \rfloor = \lceil x \rceil$. Otherwise $\lfloor x \rfloor + 1 = \lceil x \rceil$.

(3)
$$\lfloor x \rfloor = n \quad \Rightarrow \quad n \le x < n + 1.$$

$$\lceil x \rceil = n \quad \Rightarrow \quad n - 1 < x \le n.$$

Integer Function : Let x be any real number. Then integer of x converts x into an integer by deleting the fractional part of x. If is denoted by INT (x).

Example INT $(3\cdot 48) = 3$

INT$(-3\cdot 6) = -3$

Remark : $\text{INT}(x) = \lfloor x \rfloor$ if x is positive.

$\text{INT}(x) = \lceil x \rceil$ if x is negative.

Absolute Value Function : Let x be any real number. Then absolute value of x, denoted by $|x|$ or ABS(x) is defined as :

$$|x| = \begin{cases} x \text{ if } x \geq 0 \\ -x \text{ if } x < 0 \end{cases}$$

Example : $|-5| = 5$

$|10| = 10$

$|-6 \cdot 2| = 6 \cdot 2.$

Characteristic Function : Let A be a set and S be any subset of A. Then characteristic function denoted by $c_S : A \to \{0, 1\}$ be defined by

$$c_S(x) = \begin{cases} 1, \text{ if } x \in S \\ 0, \text{ if } x \notin S \end{cases}$$

ILLUSTRATIVE EXAMPLES

Example 1. If $A = \{a, b, c\}$, $S = \{a, b\}$ find c_S.

Sol. Since $c_S : A \to \{0, 1\}$

so $c_S(a) = 1$ $[\because a \in S]$

$c_S(b) = 1$ $[b \in S]$

$c_S(c) = 0$ $[c \notin S]$

$\therefore$ $c_S = \{(a, 1), (b, 1), (c, 0)\}$

Example 2. $A = \{a, b, c, d, e\}$ $S = \{a, c, e\}$. Find c_S also find c_ϕ and c_A.

Sol. Again $c_S : A \to \{0, 1\}$

so $c_S(a) = 1$, $c_S(b) = 0$, $c_S(c) = 1$, $c_S(d) = 0$, $c_S(e) = 1$.

$\Rightarrow$ $c_S = \{(a, 1), (b, 0), (c, 1), (d, 0), (e, 1)\}$

$c_A = \{(a, 1), (b, 1), (c, 1), (d, 1), (e, 1)\}$

$c_\phi = \{(a, 0), (b, 0), (c, 0), (d, 0), (e, 0)\}$

Example 3. Prove that if x is a real number, then $\lfloor 2x \rfloor = \lfloor x \rfloor + \lfloor x + 0.5 \rfloor$

(Pbi. U. B.C.A., 2011)

Sol. Let $x = n + \in$, where n is a positive integer and $0 \le \in < 1$.

Two case arise :

Case I $0 \le \in < \frac{1}{2}$

$$2x = 2n + 2\in$$

$$\lfloor 2x \rfloor = 2n \qquad [\because 0 \le 2\in < 1] \ldots(1)$$

also $x + \frac{1}{2} = n + \in + \frac{1}{2}$

$$\left\lfloor x + \frac{1}{2} \right\rfloor = n \qquad [\because 0 < \frac{1}{2} + \in < 1]$$

$$\lfloor x \rfloor + \lfloor x + 0.5 \rfloor = n + n = 2n \qquad \ldots(2)$$

From (1) and (2), we get

$$\lfloor 2x \rfloor = \lfloor x \rfloor + \lfloor x + 0.5 \rfloor$$

Case II $\frac{1}{2} \le \in < 1$

$$2x = 2n + 2\in = (2n + 1) + (2\in - 1)$$

$$\lfloor 2x \rfloor = 2n + 1 \qquad [\because 0 \le 2\in - 1 < 1] \ldots(3)$$

also $\left\lfloor x + \frac{1}{2} \right\rfloor = \left\lfloor n + \frac{1}{2} + \in \right\rfloor = \left\lfloor (n+1) + \left(\in - \frac{1}{2}\right) \right\rfloor$

$$= n + 1 \qquad [\because 0 \le \in - \frac{1}{2} < 1]$$

$$\lfloor x \rfloor + \lfloor x + 0.5 \rfloor = n + n + 1 = 2n + 1 \qquad \ldots(4)$$

From (3) and (4), we get

$$\lfloor 2x \rfloor = \lfloor x \rfloor + \lfloor x + 0.5 \rfloor$$

Remainder Function : Let k be any integer and let M be a positive integer. Then k (mod M) denotes the integer remainder when k is divided by M. This function is called remainder function.

k (mod M) is a unique integer such that

$$k = Mq + r \text{ where } 0 \le r < M$$

Example : For positive numbers, we simply divide k by M to obtain remainder r.

$\therefore$ 26 (mod 4) = 2

25 (mod 3) = 1

30 (mod 6) = 0

For negative numbers, divide $|k|$ by M to get remainder r' and then

$$k \text{ (mod M)} = M - r' \text{ if } r' \neq 0.$$

To find -29 (mod 5), first we divide 29 by 5 getting $r' = 4$.

So -29 (mod 5) $= 5 - 4 = 1$.

Similarly -35 (mod 9) $= 9 - 8 = 1$

-30 (mod 6) $= 0$.

Mod Relation : The term "mod" is also used for mathematical congruence relation which is denoted by $a \equiv b$ (mod M) and is defined as a is congruent to b iff M divides $b - a$.

Logarithm and Exponent Functions :

Let b be any positive number. The logarithm of any positive number x to base b is written as $\log_b x$ and it represent exponent to which b must be raised to obtain x. That is,

$$y = \log_b x \text{ iff } b^y = x$$

so
$$\log_2 16 = 4 \quad \because \quad 2^4 = 16$$

$$\log_3 81 = 4 \quad \because \quad 3^4 = 81$$

$$\log_{10} 1000 = 3 \quad \because \quad 10^3 = 1000$$

$$\log_{10} 0{\cdot}01 = -2 \quad \because \quad 10^{-2} = 0{\cdot}01.$$

For any base b, $\log_b 1 = 0$ since $b^0 = 1$

and $\log_b b = 1$ since $b^1 = b$

Note : Logarithm of negative number and logarithm of 0 and not defined.

Example : Find $\lfloor \log_2 100 \rfloor$

Sol. Since $2^6 = 64$ and $2^7 = 128$

so $6 < \log_2 100 < 7$

$\Rightarrow$ $\lfloor \log_2 100 \rfloor = 6$.

Factorial Function : Let n be any positive integer. Then factorial of n is product of positive integers from 1 to n. Factorial of n is denoted by n ! or $\underline{|n}$.

$$n! = 1 \cdot 2 \cdot 3 \ldots\ldots (n-1)\, n$$

Note : $0! = 1$.

Recursive Definition of Factorial : Factorial function can also defined in terms of recursion.

$$n! = \begin{cases} n \cdot (n-1)!, & \text{if } n \geq 1 \\ 1, & \text{if } n = 0 \end{cases}$$

Art-2. Growth of Functions

Mathematical functions are used in computer science to compare two algorithms. Algorithms are generally compared or analysed on the basis of their complexity, $f(n)$, which refers to number of key operations. Complexity is further measured in terms of size of input data n.

Suppose M is an algorithm and n is size of input data. Complexity of M is some function of n. As n grows, complexity of M also increases. We are interested in measuring this rate of growth. For this we compare $f(n)$ with some standard functions, such that

$$\log_2 n,\ n,\ n\log_2 n,\ n^2,\ n^3,\ 2^n.$$

All these functions have different rate of growths. Which is shown below

$f(n)\rightarrow$ / $\downarrow n$	$\log n$	n	$n \log n$	n^2	n^3	2^n
5	3	5	15	25	125	32
10	4	10	40	100	10^3	10^3
100	7	100	700	10^4	10^6	10^{30}
1000	10	1000	10000	10^6	10^9	10^{300}

Out of above functions, 2^n has maximum growth where as $\log_2 n$ has minimum growth. Now complexity of any algorithm is measured in terms of these standard functions. For this we use a special notation called, Big-O notation.

Definition – Big-O : Let $f(x)$ and $g(x)$ are functions defined on set of real numbers (or subset of R). Then $f(x)$ is called order of $g(x)$, written as $f(x) = O(g(x))$, if there exist a real number 'm' and a positive constant c such that for all $x \geq m$, we have

$$|f(x)| \leq c\,|g(x)|$$

then $f(x)$ is called Big-O of $g(x)$.

To show $f(x)$ is $O(g(x))$ we have to find pair c and m. Big-O gives an upper bound on number of key operations *i.e.* if we know Big-O of an algorithm we can say about maximum number of key operations. To give information about lower bound, we introduces one more function, known as Big-omega. (Ω).

Definition – Big-omega : Let $f(x)$ and $g(x)$ be functions on set of real numbers R(or subset of R). We say $f(x)$ is $\Omega(g(x))$ if there exist positive constants c and k such that

$$|f(x)| \geq c\,|g(x)|\ \forall\, x \geq k.$$

Then $f(x)$ is called big-omega of $g(x)$.

Definition – Big-theta Notation (Θ) : Let $f(x)$ and $g(x)$ be functions on set of real numbers, R (or subset of R). We say $f(x)$ is $\Theta(g(x))$ if

(*i*) $f(x)$ is $O(g(x))$

(*ii*) $f(x)$ is $\Omega(g(x))$ or we have positive constants c_1, c_2 and k such that

$$c_1|g(x)| \leq |f(x)| \leq c_2|g(x)|\ \forall\, x \geq k.$$

To show Big-Θ we have to prove both Big-O and Big-Ω.

The time efficiency of almost all of the algorithms can be characterized by only a few growth rate functions :

I. O(*l*) – Constant Time

This means that the algorithm requires the same fixed number of steps regardless of the size of the task.

Example : (Assuming a reasonable implementation of the task) :

A. Push and Pop operations for a stack (containing n elements);

B. Insert and Remove operations for a queue.

II. O(n) – Linear Time

This means that the algorithm requires a number of steps proportional to the size of the task.

Examples : (Assuming a reasonable implementation of the task) :

A. Traversal of a list (a linked list or an array) with n elements ;

B. Finding the maximum or minimum element in a list, or sequential search in an unsorted list of n elements ;

C. Traversal of a tree with n nodes ;

D. Calculating iteratively n-factorial ; finding iteratively the nth Fibonacci number.

III. O(n^2) – Quadratic Time

The number of operations is proportional to the size of the task squared.

Examples :

A. Some more simplistic sorting algorithms, for instance a selection sort of n elements;

B. Comparing two two-dimensional array of size n by n;

C. Finding duplicates in an unsorted list of n elements (implemented with two nested loops).

IV. O(log n) – logarithmic Time

Examples :

A. Binary search in a sorted list of n elements ;

B. Insert and Find operations for a binary search tree with n nodes ;

C. Insert and Remove operations for a heap with n nodes.

V. O(n log n) – "n log n" time

Example :

A. More advanced sorting algorithms – quicksort, mergesort

VI. $O(a^n)$ $(a > 1)$ – Exponential Time

Examples :

A. Recursive Fibonacci implementation

B. Towers of Hanoi

C. Generating all permutations of n symbols

The best time in the above list is obviously constant time, and the worst is exponential time which, as we have seen, quickly overwhelms even the fastest computers even for relatively small n. Polynomial growth (linear, quadratic, cubic, etc.) is considered manageable as compared to exponential growth.

Order of asymptotic behaviour of the functions from the above list :

Using the "<" sign informally, we can say that

$$O(l) < O(\log n) < O(n) < O(n \log n) < O(n^2) < O(n^3) < O(a^n)$$

Big-O when a function is the sum of several terms :

If a function (which describes the order of growth of an algorithm) is a sum of several terms, its order of growth is determined by the **fastest growing term.** In particular, if we have a polynomial

$$p(n) = a_k n^k + a_{k-1} n^{k-1} + \ldots.. + a_1 n + a_0$$

its growth is of the order n^k.

$$p(n) = O(n^k)$$

ILLUSTRATIVE EXAMPLES

Example 1. Show that $f(x) = x^2 + 2x + 1$ is $O(x^2)$.

(Pbi.U. B.C.A.-II, 2006, 2008, 2009)

Sol. Let $x \geq 1$

then $1 \leq x \leq x^2$ ...(1)

$$|f(x)| = |x^2 + 2x + 1|$$

$$\leq |x^2| + |2x| + |1| \quad [\,|x+y| \leq |x| + |y|\,]$$

$$= x^2 + 2x + 1$$

$$\leq x^2 + 2x^2 + x^2 \quad [\text{using (1)}]$$

$$= 4x^2$$

so $|f(x)| \leq 4|x^2| \ \forall x \geq 1$ $[c = 4, k = 1]$

$\therefore$ $f(x) = O(x^2)$.

Example 2. Show that $7x^2 - 9x + 4 = O(x^2)$. (Pbi. U. B.C.A. 2007)

Sol. Let $x \geq 1$

then $1 \leq x \leq x^2$ ...(1)

$$|f(x)| = |7x^2 - 9x + 4|$$

$$\leq |7x^2| + |9x| + |4| - [\,|x \pm y| \leq |x| + |y|\,]$$

$$= 7x^2 + 9x + 4$$

$$= 7x^2 + 9x + 4 \cdot 1$$

$$\leq 7x^2 + 9x^2 + 4x^2 \quad \text{[using (1)]}$$

$$= 20x^2$$

$\therefore$ $|f(x)| \leq 20\,|x^2|\ \forall x \geq 1$ [$c = 20, k = 1$]

$\Rightarrow$ $f(x) = O(x^2)$.

Example 3. Suppose $P(n) = a_0 + a_1 n + a_2 n^2 \ldots\ldots a_m n^m$. Suppose degree $P(n) = m$. Prove that $P(n) = O(n^m)$.

(Pbi. U. B.C.A. April 2008)

Sol. Let $n \geq 1$

then $1 \leq n \leq n^2 \leq n^3 \ldots\ldots \leq n^m$ ...(1)

Now $|P(n)| = |a_0 + a_1 n + a_2 n^2 \ldots a_m n^m|$

$$\leq |a_0| + |a_1 n| + |a_2 n^2| \ldots\ldots |a_m n^m| - [\,|x + y| \leq |x| + |y|\,]$$

$$= |a_0| + |a_1|\,n + |a_2|\,n^2 \ldots\ldots |a_m|\,n^m$$

$$= |a_0| \cdot 1 + |a_1|\,n + |a_2|\,n^2 \ldots\ldots |a_m|\,n^m$$

$$\leq |a_0|\,n^m + |a_1|\,n^m + |a_2|\,n^m \ldots\ldots |a_m|\,n^m \quad \text{[using (1)]}$$

$$= [\,|a_0| + |a_1| + |a_2| \ldots\ldots |a_m|\,]\,n^m$$

$$= cn^m$$

where $c = |a_0| + |a_1| + |a_2| \ldots\ldots |a_m|$

$$\Rightarrow \qquad |P(n)| \leq c\,|n^m| \;\forall\, n \geq 1$$

$$\therefore \qquad P(n) = O(n^m).$$

Example 4. Find Big-O notation for $n\,!$ and $\log n\,!$.

Sol. Let n be any natural number.

By definition $n\,! = 1 \cdot 2 \cdot 3 \cdot \ldots\ldots n$

Also $1 < 2 < 3 < 4 \ldots\ldots\ldots \leq n$

Now
$$|n\,!| = 1 \cdot 2 \cdot 3 \ldots\ldots\ldots n$$
$$\leq n \cdot n \cdot n \ldots\ldots\ldots n$$
$$\leq n^n$$

$$\Rightarrow \qquad |n\,!| \leq 1 \cdot n^n \;\forall\, n \geq 1$$

$\therefore$ $n\,!$ is $O(n^n)$ where $c = 1$, $m = 1$

Also $\log n\,!$ is $O(\log n^n)$
$$= O(n \log n)$$

$\therefore$ $\log n\,!$ is $O(n \log n)$

Example 5. Give Big-O estimate for

$$f(n) = 3\,n \log n\,! + (n^2 + 3) \log n$$

(Pbi. U. B.C.A. 2008)

Sol. First we use previous example to show

$$\log n\,! \text{ is } O(n \log n) \qquad \ldots(1)$$

Now, let
$$n \geq 1 \text{ i.e. } 1 \leq n \leq n^2 \qquad \ldots(2)$$

$$|f(n)| = |3\,n \log n\,! + (n^2 + 3) \log n|$$
$$\leq |3\,n \log n\,!| + |(n^2 + 3) \log n| \qquad \ldots[\,|x + y| \leq |x| + |y|\,]$$
$$\leq 3\,n\,.\,n \log n + (n^2 + 3 \cdot n^2) \log n \qquad \text{[Using (1) and (2)]}$$
$$= 3\,n^2 \log n + 4\,n^2 \log n$$

$$|f(n)| \leq 7 \cdot |n^2 \log n| \;\forall\; n \geq 1$$

$\therefore$ $f(n)$ is $O(n^2 \log n)$ where $c = 7$ and $m = 1$.

Example 6. Give a Big-O estimate for

$$f(x) = (x + 1) \log (x^2 + 1) + 3\,x^2$$

(Pbi. U. B.C.A. 2011)

Sol. First, a Big-O estimate for $(x+1)\log(x^2+1)$ will be found.

Note that $(x+1)$ is $O(x)$.

Furthermore, $x^2+1 \leq 2x^2$ when $x > 1$

Hence $\log(x^2+1) \leq \log(2x^2)$

$$= \log 2 + \log x^2$$
$$= \log 2 + 2\log x$$
$$\leq 3\log x \quad \text{if} \quad x > 2. \qquad [\because \log 2 < \log x]$$

This shows that $\log(x^2+1)$ is $O(\log x)$.

$\therefore$ $(x+1)\log(x^2+1)$ is $O(x\log x)$

Since $3x^2$ is $O(x^2)$

$\therefore$ $f(x)$ is $O(\max(x\log x, x^2)$

since $x\log x \leq x^2$, for $x > 1$

Hence it follows that $f(x)$ is $O(x^2)$.

Example 7. Prove that $f(x) = 8x^3 + 5x^2 + 7$ is $\Omega(g(x))$ where $g(x) = x^3$.

Sol. Let $x \geq 0$

then $x^2 \geq 0$

Now $|f(x)| = |8x^3 + 5x^2 + 7|$

$$\geq 8x^3 \qquad [\because 5x^2 + 7 \geq 0]$$

$\Rightarrow$ $|f(x)| \geq 8 \cdot |x^3| \ \forall \ x \geq 0$

$\Rightarrow$ $|f(x)| \geq 8 \cdot |g(x)| \ \forall \ x \geq 0$

Hence $f(x)$ is $\Omega(g(x))$ where $c = 8, k = 0$.

EXERCISE 6 (*a*)

1. Let f be mod-11 function. Compute.

   (*a*) $f(417)$ (*b*) $f(40)$ (*c*) $f(-253)$.

2. Let U = $\{a, b, c, \ldots\ldots, x, y, z\}$ and A = $\{a, e, i, o, u\}$. Find characteristic function of A.

3. Compute

(a) $\lfloor 2\cdot 78 \rfloor$ (b) $\lfloor -3\cdot 46 \rfloor$ (c) $\lfloor 18 \rfloor$ (d) $\lfloor \sqrt{60} \rfloor$

(e) $\lceil 9\cdot 3 \rceil$ (f) $\lceil -10\cdot 6 \rceil$ (g) $\lceil \sqrt{7} \rceil$ (h) $\lceil 9 \rceil$.

4. Calculate

(a) $\log_2 16$ (b) $\log_2 128$ (c) $\log_2 1024$

5. Show that $g(n) = n!$ is $O(n^n)$

6. Show that $g(n) = n^2 (7n-2)$ is $O(n^3)$

7. Show that $x^4 + 9x^3 + 4x + 7$ is $O(x^4)$

8. Prove that $7x^2$ is $O(x^3)$. Is it true that x^3 is $O(7x^2)$?

9. Define big omega and big theta functions.

(Pbi. B.C.A. April 2011)

10. Define floor, selling function, big-O, and big-omega function give one example of each. (Pbi. U. B.C.A. 2012)

11. Define characteristic function and absolute value function.

(Pbi. U. B.C.A. 2012)

ANSWERS

1. (a) 10 (b) 7 (c) 0
3. (a) 2 (b) −4 (c) 18 (d) 7
   (e) 10 (f) − 10 (g) 3 (h) 9
4. (a) 4 (b) 7 (c) 10