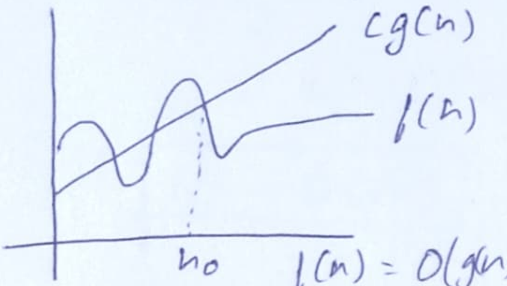
Q.1 Asymptotic notations

They are mathematical notations used to describe the running time of an algorithm when the input tends towards a particular value or a limiting value

There are mainly three types

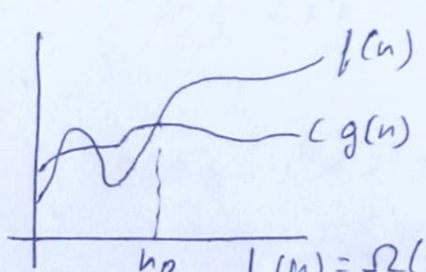
- Big O notations → It represents the upper bound of the running time of an algorithm, thus gives worst time complexity of an algorithm.



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

$O(g(n)) = \{ f(n)$ there exist positive constant c and n_0 such that $0 \leq f(n) \leq c\,g(n)$ for all $n > n_0 \}$

- Omega notations → It represents the lower bound of the running time of an algorithm thus provides best case complexity

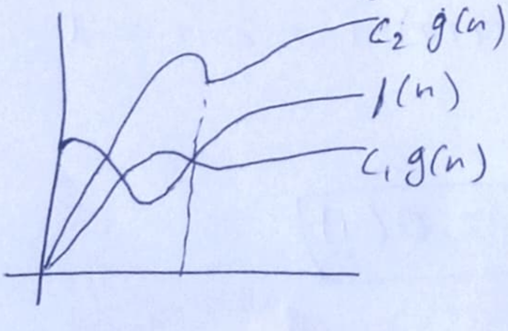


$f(n) = \Omega(g(n))$

$\Omega(g(n)) = \{ f(n)$: there exist positive constants c & n_0 such that $0 \leq c\,g(n) \leq f(n)$ for all $n \geq n_0 \}$

- Theta notations → It represents lower and upper bound of running time of an algorithm Thus gives average time complexity.

$\Theta(g(n)) = \{ f(n)$: there exists positive constants c_1, c_2 & n_0 such that $0 \leq c_1 g(n) \leq f(n) \leq c_2\,g(n)$ for all $n \geq n_0 \}$



Q.2 for (i = 1 to n) { i = i * 2 }

i	1	2	4	...	2^k
Value	2^0	2^1	2^2	...	n

$$2^k = n$$
$$k \log_2 2 = \log_2 n$$
$$k = \log n$$
$$\boxed{T.C = O(\log(n))}$$

Q3 $T(n) = \{ 3T(n-1) \quad n>0 \}$

By forward

$$T(n) = 3T(n-1)$$
$$T(1) = 3T(1-1)$$
$$= 3T(0) = 3$$
$$T(2) = 3^2$$
$$T(3) = 3^3$$
$$T(n) = 3^n$$
$$\boxed{T.C. = O(3^n)}$$

Q 4 $T(n) = \begin{cases} 2T(n-1) - 1, & n>0 \\ 1, & n=0 \end{cases}$

$$T(0) = 1$$
$$T(1) = 2T(1-1) - 1 = 2T(0) - 1 = 2 - 1 = 1$$
$$T(2) = 1$$
$$T(3) = 1$$
$$T(n) = 1$$
$$\boxed{T.C = O(1)}$$

Q5

```
int i=1, s= 1
while (s<=n)
  { i++;
    s= s+i;
    print("#")
  }
```

let

for k iteration

$$S(k) = 1+2+3+\cdots+k = \frac{(k+1)k}{2}$$

$$\frac{(k+1)k}{2} > n$$

$$k = O(\sqrt{n})$$

$$\boxed{TC = O(\sqrt{n})}$$

Q6

```
fun (int n)
{  int i, count =0;
   for (i=1; i*i <=n; i++)
     { c++; }
}
```

$$S(k) = 1^2+2^2+3^2 \cdots k^2 <= n$$

$$= \frac{k(k+1)(2k+1)}{6} \leq n$$

$$= 2k^3 + 3k^2 + k \leq 6n$$

$$\boxed{TC = \sqrt[3]{n}}$$

Q 7

```
fun (int n)
{  int i, j, k, c=0
   for (i = n/2 ; i <= n ; i++)
      for (j = 1 ; j <= n ; j = j*2)
         for (k = 1 ; k <= n ; k = k*2)
             Count ++
```

outer loop run $n/2$ time
second loop run $\log(n)$ time
third loop run $\log(n)$ time

$$T.C. = \frac{n}{2} n \log(n) * \log(n)$$

$$T.C = O(n(\log n)^2)$$

Q 8

```
fun (int n)
{  if (n==1) return;
      for (i=1 to n)
        for (j=1 to n)
            print ("*")
    fun (n-3)
  }
```

for 1^{st} loop n times
for 2^{nd} loop n times

$$T.C = n*n = O(n^2)$$

Q9

```
fun (int n)
 for (i=1 to n)
    for (j=1; j<=n; j=j+i)
           printf("*");
```

Outer loop n times
inner loop n times

T.C. = $n \times \log n = O(n \log n)$

Q10

n^k & c^n

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