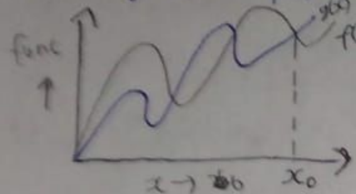
Q1) Asymtotic notation and define it's diff types.

(i) Big $O(n)$ $f(n) \Rightarrow O(g(n))$ if $f(x) \le g(x) \times c$ $\forall$ $x \ge 0$, $c > 0$ $g(x)$ is upper bound

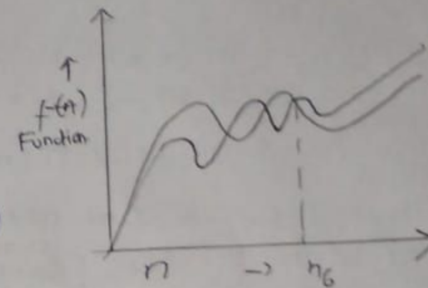


eg. $f(n) \Rightarrow n^2 + n$ $g(x) \Rightarrow n^3$

$n^2 + n \le c * n^3$,

$n^2 + n = O(n^3)$

2) Big omega Ω



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

$\Rightarrow g(n)$ is "tight" lower bound of $f(n)$ i.e. $f(n)$ can go beyond $g(n)$

i.e $f(n) = \Omega g(n)$ if and only if $f(x) \ge c \cdot g(n)$

$\forall$ $n_2 > n_0$ & $c > 0$

Ex $f(n) \Rightarrow n^3 + 4n^2$ $g(n) = n^2$ ie. $f(n) \ge c * g(n)$ $n^3 + 4n^2 = \Omega(n^2)$

3) Theta given us a range of $f(n)$ $c_1 \cdot g(n) \le f(n) \le c_2\, g(n)$

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

4) small oh (o): Upper bound of $f(n)$ $f(n) = o(g(n))$ $f(n) < c \cdot g(n)$

5) small omega (ω) lower bound of $f(n)$ $f(n) > c \cdot g(n)$

$f(n) = \omega(g(n))$,

2) What should be the time complexity of

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

1, 2, 4, 8, 16 ...... n

$\sum_{i=1}^{n} = 1 + 2 + 4 + 8 \ldots n$ k^{th} Term = $T_k = ar^{k-1}$

$a = 1, r = 2$

$T_k = 1 \times 2^{k-1} = 2^{k-1}$ Put $n = 2^{k-1}$ $2n = 2^k$

$\log_2 2n = k \log_2 2 \Rightarrow \log_2 2 + \log_2 n = k \Rightarrow \log_2 n + 1 = k =$

$\Rightarrow O(\log n)$

Q3) $T(n) = \{ 3T(n-1)$ if $n > 0$, otherwise $1 \}$

$T(0) = 1$ $T(n) = 3T(n-1)$ — (1) Put $n = n-1$

$T(n-1) = 3T(n-2)$ — (2)

(2)

Put value of $T(n-1)$ in (1)

$T(n) = 3 \times 3T(n-2) = 9T(n-2)$ — (3)   Put $n = n-2$ in (1)

$T(n-2) = 3T(n-3)$ — (4)   Put value of $T(n-2)$ in (3)

$T(n) = 27T(n-3) \Rightarrow T(n) = 3^k T(n-k)$ Put $k = n$ $T(n) = 3^n T(0)$

$T(n) = 3^n$

Q4) $T(n) = \{ 2T(n-1) - 1$ for $n > 0$, else $1 \}$

$T(n) = 2T(n-1) - 1$ — (1)

$T(0) = 1$

Put $n = n-1$ in (1)

$T(n-1) = 2T(n-2) - 1$ — (2)

Put value of $T(n-1)$ in (1)

$T(n) = 2 \times 2T(n-2) - 2 - 1$

$T(n) = 4T(n-2) - 3$ — (3)

Put $n = n-2$ in (1)

$T(n-2) = 2T(n-3) - 1$ — (4)

Put $T(n-2)$ in (3)

$T(n) = 4(2T(n-3) - 1) - 2 - 1$

$= 8T(n-3) - 4 - 2 - 1$

$T(n) = 2^k T(n-k) - (2^{k-1} + 2^{k-2} + 2^{k-3} \dots 1)$

$T(n) = 2^k T(n-k) - 2^{k-1} - 2^{k-2} - 2^{k-3} \dots 1$

$= 2^k T(n-k) - (1 + 2 + 4 \dots 2^{k-1}) \Rightarrow 2^k - \left[1 \dfrac{(2^k - 1)}{2-1} \right]$

$\Rightarrow 2^k - 2^k + 1 \Rightarrow T(n) = 1$

Que what should be the time complexity of `int i=1, s=1; while(s<=n) { i++; s=s+i; print("#"); }`

We can define the term s according to $S_i = S_{i-1} + i$.

value in s at the end of i^{th} iteration is the sum of the first 'i' positive integer. If k is the total no. of iterations taken by the program, then while loop terminates i

$1 + 2 + 3 \dots k = \dfrac{k(k+1)}{2} > N$

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

Que Time complexity of

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

$i^2 <= n \Rightarrow i <= \sqrt{n}$

$i = 1, 2, 3, 4 \dots \sqrt{n}$

$\sum_{i=1}^{n} = 1 + 2 + 3 + 4 \dots \sqrt{n}$ $T(n) = \dfrac{\sqrt{n} \times (\sqrt{n}+1)}{2} \Rightarrow T(n) = \dfrac{n \times \sqrt{n}}{2}$

$T(n) = O(n)$

7. Time complexity of

```
void function(int n)
{ int i, j, k, count=0;   for(i=n/2; i<=n; ++i)
                          for(j=1; j<=n; j*=2)
                          for(k=1; k<=n; k*=2)
                          ++count;
}
```

$\Rightarrow$ Ist loop $n/2$ times
IInd loop $\log n$ times
IIIrd loop $\log n$ times

$T(n) = \frac{n}{2} \times \log_2 n \times \log n$

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

8) Time complexity of

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

$T(n) = T(n-3) + O(n^2)$

$T(n-3) + n^2$ — (1)

$T(n-3) = T(n-6) + (n-3)^2$

from eq (1):

$T(n) = T(n-6) + (n-3)^2 + n^2$

$T(n-6) = T(n-9) + (n-6)^2$

$T(n) = T(n-9) + (n-6)^2 + (n-3)^2 + n^2$

$T(n) = T(n-3k) + (n-(3k-3))^2 + [n-(3k-6)]^2 + n^2$

$n - 3k = 0$

$k = n/3$

$T(n) = T(n-n) + 1^2 + 2^2 + 3^2 + \dots n^2$

$= T(0) + \frac{n(n+1)(n+2)}{6}$

$T(n) = \frac{n * n * n}{6}$

$T(n) = O(n^3)$

a) - Time complexity of -

```
void function (int n)
{ for (i=1 to n)
    { for (j=1; j<=n; j=j+i)
        Printf("*");
    } }
```

for $i=1 \Rightarrow j=1,2,3,4 \dots n = n$

$i=2 \Rightarrow j=1 \dots \quad n = n/2$

$i=3 \Rightarrow j=1,4,7 \dots n = n/3$

for $i=n = j=1 \quad = 1$

$\Rightarrow \sum_{j=n}^{1} = n + \frac{n}{2} + \frac{n}{3} + \frac{n}{4} + \dots 1$

$\Rightarrow \sum_{j=n}^{1} = n\left(1 + \frac{1}{2} + \frac{1}{3} \dots \frac{1}{n}\right) \Rightarrow T(n) = n \log n$

10) Foo the function n^k & C^n, what is the asymptotic relationship between these function.
Assume $K >= 1$ & $c > 1$ are constants. Find C & n for which relation hold,

n^K & c^n

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

as $n^K < a c^n$

$\forall$ $x > x_0$ & some constant $a > 0$

for $n_0 = 1$

$c = 2$

$1^n \leq a_2^n$

$n_0 = 1$ & $c = 2$.