Design and Analysis of Algorithms

Assignment - 1

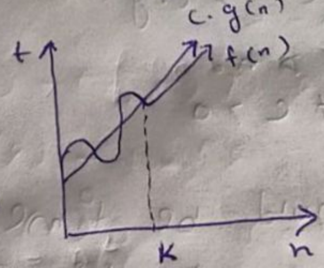
Aniket
CST SPL-1
49

Q. What do you understand by Asymptotic notations. Define different Asymptotic notation

=> Asymptotic notations are mathematical way of representing a time complexity. There are three main Asymptotic notation

1) Big-oh (O)

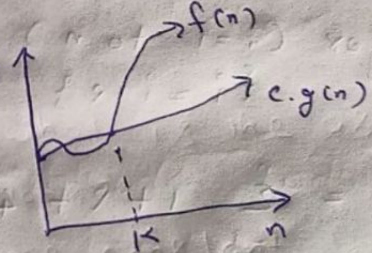
- worst case
- upper bound (at most)




$f(n) = O\,g(n)$
$f(n) \leq c \cdot g(n)$
$c > 0$

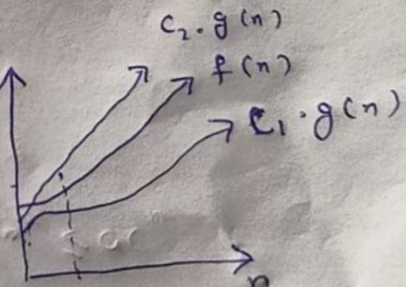
2) Big-omega ($\Omega$)

- Best case
- Lower bound (at least)




3) Theta ($\Theta$)

- Average case
- Exact time




Aniket

4) small o(o)

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

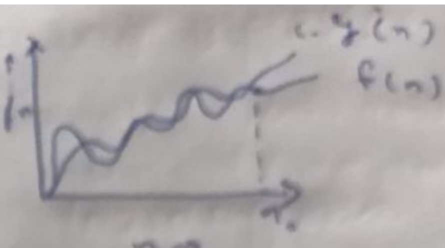
$g(n)$ is upper bound of $f^n$ for/

$f(n) = o(g(n))$

when $f(n) < c.g(n)$

$\forall\ n > n_0$

$\&\ c > 0$





5) Small omega (ω)

$f(n) = w(g(n))$

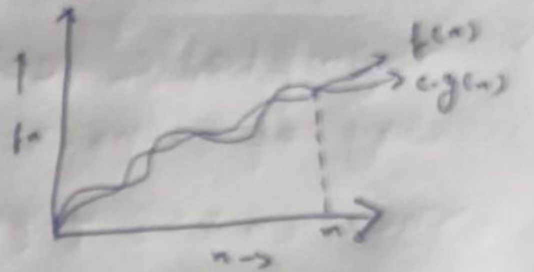
$g(n)$ is lower bound of $f(n)$

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

when $f(n) > c.g(n)$

$\forall\ n > n_0$

$\&\ \forall\ c > 0$





Q what should be time complexity

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

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

// i = 1, 2, 4, 8

// O(1)

$= \sum_{i=1}^{n} 1 + 2 + 4 + 8 + \dots + n$

GP $k^{th}$ value => $T_k = ar^{k-1}$, a = 1 and r = 2

$n = 2^k$

=> $2n = 2^k$

$= \log 2n = k \log 2$

=> $k = \log 2n$

=> $k = \log 2 + \log n$

$k = \log n + 1$

$O(k) = O(1 + \log n)$

$= O(\log n)$

Aniket

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

$T(n) = 3T(n-1)$ — ①

put $n = n-1$

$T(n-1) = 3T(n-2)$ — ②

from 1 & 2

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

$= 9T(n-2)$ — ③

Putting $n = n-2$ in ①

$T(n) = 3T(n-3)$ — ④

$\Rightarrow T(n) = 27T(n-3)$

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

Putting $n-k = 0$

$\Rightarrow n = k$

$\Rightarrow T(n) = 3^n[T(n-n)]$

$\Rightarrow T(n) = 3^n T(0)$

$T(n) = 3^n \times 1$

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

4) $T(n) = \{2T(n-1) - 1$ if $n>0$, otherwise $1\}$

$T(n) = \{2T(n-1) - 1\}$ — (i)

put $n = n-1$ in eq$^n$ - i

$n = n-1$

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

$= 2T(n-2) - 1$ — (ii)

put eq$^n$ (ii) in eq$^n$ (i)

$T(n) = \{2(2T(n-2) - 1) - 1\}$

$= 4T(n-2) - \cancel{2} - 1$ — (iii)

put $n = n-2$ in eq$^n$ (i)

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

Aniket

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

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

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

$GP = 2^{k-1} + 2^{k-2} + 2^{k-3} + \dots\dots$

$a = 2^{k-1}$

$r = 1/2$

$= \frac{a(1-r^n)}{1-r} = 2^{k-1} \frac{(1-(1/2)^n)}{-1/2}$

$= 2^k (1-(1/2)^k) = 2^k - 1$

Let $n-k=0$

$n = k$

$T(n) = 2^n T(n-n) - (2^n - 1)$

$T(n) = 2^n \cdot 1 - (2^n - 1) = 2^n - (2^n - 1)$

$T(n) = O(1)$

$\Rightarrow 2^{k-1} = 2^{n-1} - 1 \Rightarrow O(2^n)$

Q what should be time complexity of

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

$i = 1,\ 2,\ 3,\ 4,\ 5,\ 6 \dots$

$s = 1\ \ 3\ \ 6\ \ 10\ \ 15\ \ 21$

sum of $s = 1 + 3 + 6 + 10 + \dots\dots T_n$ — (1)

also $s = 1 + 3 + 6 + \dots T_{n-1} + T_n$ — (11)

from (1) — (2)

$0 = 1 + 2 + 3 + 4 \dots\dots n - T_n$

Aniket

$T_k = \frac{1}{2} k(k+1)$

for k iterations

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

$$\frac{k^2+k}{2} <= n$$

$$O(k^2) <= n$$

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

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

Q Time complexity of

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

as $i^2 <= n$

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

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

$$\sum_{i=1}^{n} 1+2+3+4+\ldots\ldots+\sqrt{n}$$

$$\Rightarrow T(n) = \frac{\sqrt{n} \times (\sqrt{n}+1)}{2}$$

$$T(n) = \frac{n\sqrt{n}}{2}$$

$$T(n) = O(n)$$

Aniket

Q Time complexity of :-

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

for $K = K*2$

$K = 1, 2, 4, 8 \ldots n$

GP => $a = 1$ , $r = 2$

$$= \frac{a(r^n - 1)}{r - 1} = 1\,\frac{(2^k - 1)}{1} = 2^k$$

$$n = 2^k$$

$$\log n = k \log 2$$

$$\log n = k$$

Below

| $i$ | $j$ | $K$ |
|---|---|---|
| | $\log n$ | $\log n \times \log n$ |
| 1 | | $\log n \times \log n$ |
| 2 | $\vdots$ | $\vdots$ |
| 3 | $\vdots$ | $\vdots$ |
| $\vdots$ | | |
| $n$ | $\log n$ | $\log n \times \log n$ |

$$O(n * \log n * \log n)$$

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

Aniket

Q) Time complexity of

```
function (int n)
{ int(n==1)
    return;                      // 0
  for(i=1 to n)                  // i = 1,2,3,4   = O(n)
  { for (j=1 to n)               // j = 1,2,3,4   = O(n^2)
      { print ('*')
      }
  }
  function (n-3);        T(n/3)
}
```

$\Rightarrow \quad T(n) = T(n/3) + n^2$

$a = 1, \quad b = 3, \quad f(n) = n^2$

$c = \log_3 1 = 0$

$\Rightarrow n^0 = 1 > f(n) = n^2$

$T(n) = \Theta(n^2)$

Q ~~Find~~ for functions, $n^k$ & $c^n$, what is the asymptotic relation between these functions? assume that $k >= 0$ & $c > 1$ are constant find out the value of c & $n_0$ for which relation holds

as given $n^k$ & $c^n$

relation b/w $n^k$ & $c^n$ is

$n^k = O(c^n)$ as $n^k \leq ac^n$ & some constant $a>0$

$\forall n \geq n_0$

for $n_0 = 1$, $c = 2$

$\rightarrow 1^k \leq a 2^1$

1 & c = 2

Aniket