ANANYA PANDEY
ROLL 17 DS1

classmate
Date
Page
23/02

ASSIGNMENT 1.

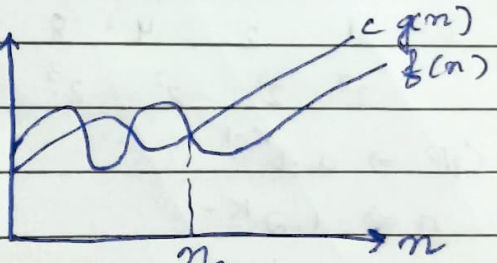
1. Asymptotic notations are used to represent the complexities of algorithms for asymptotic analysis.
— These notations are mathematical tools to rep. complexities.

Big oh notation :
Gives an upper bound for a fⁿ f(n) within a constant factor.

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

if $f(n) \leq c\,g(n)$

for $c > 0$ & $n \geq n_0$.

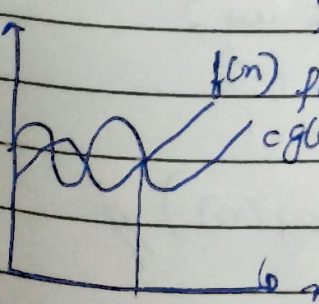


Big omega notation :
Gives lower bound for a fⁿ f(n) within a const factor c

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

if $f(n) \geq c\,g(n)$

f(n) for $c > 0$ & $n \geq n_0$



Big Theta Notation :

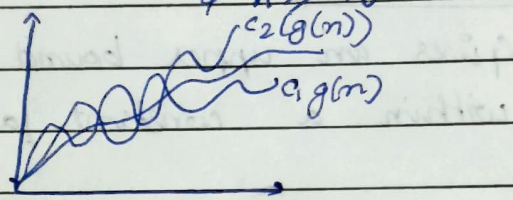
Gives bound for a f^n. $f(n)$ within a const^t factor.

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

$$c_1 g(n) \leq f(n) \leq c_2 g(n)$$

for $c_1, c_2 > 0$

& $n \geq n_0$



2. TC for →

```
for( i=1 to n )
    i = i*2;
```

i =	1	2	4	8	. .	n
	2^0	2^1	2^2	2^3		2^K

GP $\Rightarrow a r^{K-1}$

$n \Rightarrow 1.2^{K-1}$

$n \Rightarrow \frac{2^K}{2}$

$2n = 2^K$

$\log 2n = K \log 2$

$\log 2 + \log n = K \log 2$

$\boxed{\log n = K}$

$\therefore T(n) = O(\log(n))$ //

3. $T(n) = 3T(n-1)$, $n>0$, otherwise 1

$T(0) = 1$

$T(1) = 3T(0)$

$= 3$

$T(2) = 3T(1)$

$= 9 = 3^2$

$T(3) = 3T(2) = 27 = 3^3$

$T(n) = 3^n$

$= O(3^n)$ //

4. $T(n) = 2T(n-1) - 1$ — ① , $n>0$, otherwise 1

Let $n = n-1$

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

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

Put $T(n-1)$ in ①

$T(n) = 4T(n-2) - 3$ — ②

Put $n = n-2$.

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

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

Put in ②

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

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

$= 8T(n-3) - 5$ $= 2^k T(n-k) - 5$

$(n-k) = 1$

$k = (n-1)$

$T(n) = 2^{n-1} T(n-n+1) - 5$

$= 2^{n-1} T(1) - 5$

$= \frac{2^n}{2} = 2^n = O(2^n)$ //

So

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

$i = 1 \Rightarrow i++, i = 2$

$s = 3$

$i = 3$

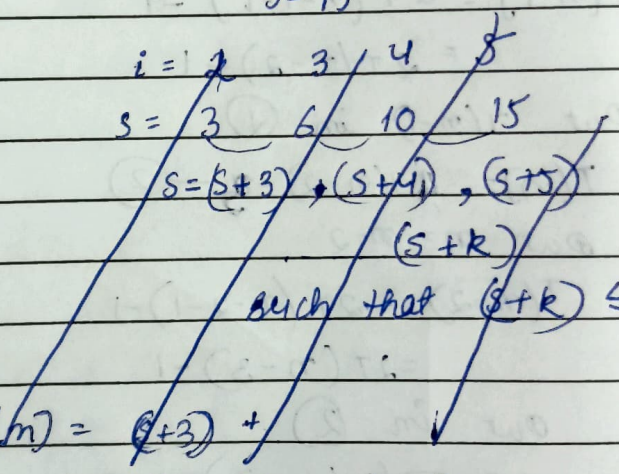
$s = 6$

$i = 4$

$s = 10$

$i = 5$

$s = 15$



$T(n) = (s+3) +$

$i = 2 \quad 3 \quad 4 \quad 5$

$s = s+1+2 \quad s+1+2+3 \quad s+1+2+3+4 \quad s+1+3+4+5$

$s = s+1+2+3+4+\dots k$

$S(k) \quad (i=k) = k(k+1)/2 \leq n$

$k^2 + k/2 \leq n$

$k^2 \leq n$

$k \leq \sqrt{n}$

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

6.
```
void fn(int n)
{ int i, count=0;
  for(i=1; i*i<=n; i++)
  {   count++;
  }
}
```

$i = 1 \quad 2 \quad 3 \quad 4 \quad \cdots$

$i^2 = 1 \quad 4 \quad 9 \quad 16 \quad \cdots \quad k^2$

$k^2 \leq n$

$k \leq \sqrt{n}$

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

7.
```
void fn(int n)
{ int i, j, k, count=0;
  for(i=n/2; i<=n; i++) → T(n/2)
  {
    for(j=1; j<=n; j=j*2) → log(n)
    { for(k=1; k<=n; k=k*2) → log(n)
        count++;
    }
  }
}
```

$T(n) = T(n/2) * \log(n) * \log(n)$

$= \frac{n}{2} * \log n^2$

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

$= O(n \log n) = O(n \log n)$ //

8.
```
fun (int n)
{
  if (n == 1)
    return;
  for (i = 1 to n)          ⟶ n
  {
    for (j = 1 to n)        ⟶ n
      printf ("*");
  }
  fun (n - 3);
}
```

$T(n) = n * n * [T(n-3)]$

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

$= O(n^2)$ //

9.
```
void fun(int n)
{ for (i = 1 to n)          ⟶ n
  { for (j = 1; j <= n; j = j + i)
    {
      print ("*");
    }
  }
}
```

$i = 1$, $j = 1, 2, 3, 4, \ldots n \rightarrow n/1$

$i = 2$, $j = 1, 3, 5, 7, \ldots n \rightarrow n/2$

$i = 3$, $j = 1, 4, 7, 11, \ldots n \rightarrow n/3$

$\rightarrow n/(n-1)$

$i = n-1$, $j = 1, n$

$n + \frac{n}{2} + \frac{n}{3} \cdots \frac{n}{n} = \log(n)$

(Harmonic series)

$$T(n) = n \times \log n = O(n \log n) //$$

10.

n^K & c^n

i) $n = 1$,

$n^K = 1^K \quad c^n = c$

$n = 2$

$nK = 2^K, c^n = c^2$

:

$n = K, n^K = K^K, c^n = c^K$

∴ we can say that for any value of $n > 0$

$\therefore n^K \geq c^n$

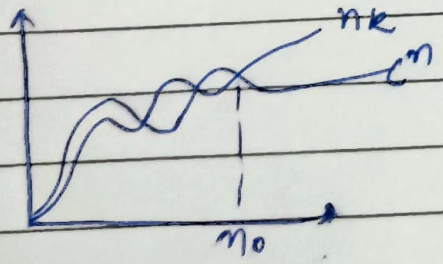
let $n^K = f(n)$, $c^n = c_0 g(n)$

$\therefore f(n) \geq c_0 g(n)$

$c_0 > 0, n_0 \geq n_0$.

$\therefore f(n) = O(c_0 g(n))$

$\therefore n^K = O(c^n) //$



11. extractMin ⇒

```
int extractMin (vector<int> &heap)
{   if (heap.empty())
    {  return -1;          → O(1)
    }
    swap (heap[0], heap.back());  → O(1)

    int minElement = heap.back();
    heap.pop_back();          → O(1)
    heapify(heap, 0);         → O(log n)
    return minElement;
}
```

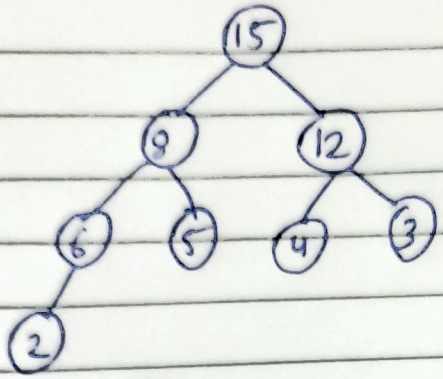
T(n) = O(log(n)) // ans.

No. of comp ⇒

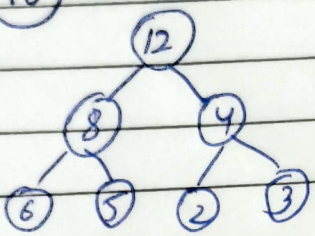
$(1 * \frac{n}{4}) + 2 * \frac{n}{8} + 3 * \frac{n}{16} + (h-1) * 1$

= log(n) // harmonic mean

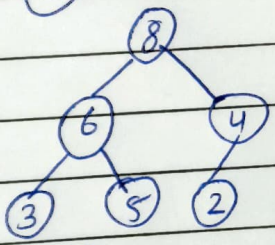
12.



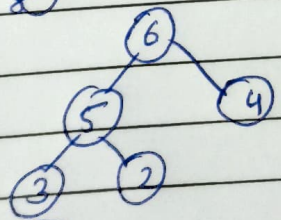
Delete root (15)



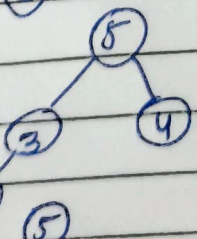
Delete Root (12)



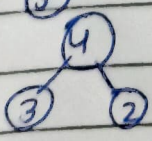
Delete Root (8)



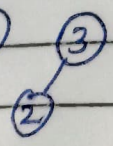
Delete Root (6)



Delete Root (5)



Delete Root (4)



Delete (2)

Delete (2)

Heap is empty.