Name: CHAKSH, Section: DS1
Roll No: 37

Ques Asymptotic notations are used to represent the complexities of algorithms for asymptotic analysis.

- These notations are mathematical tools to rep complexities

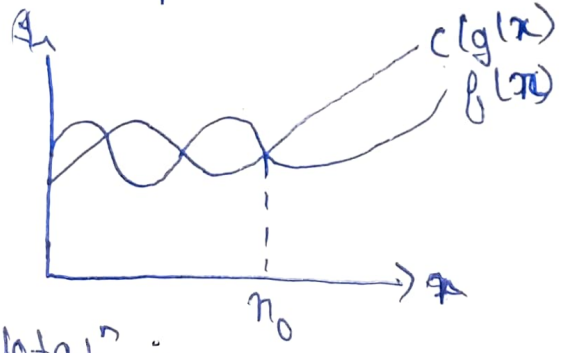
Big oh Notation:

Gives an upper bound for a f^n f(x) within a constant factor.

$f(x) = O(g(x))$

if $f(x) \leq cg(x)$

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



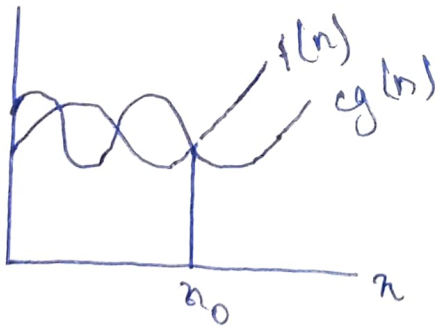
Big Omega Notat^n :

Gives lower bound for a f^n f(x) within a constant factor.

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

if $f(x) \geq cg(x)$

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



Big Theta Notation:

Gives Bound for a f^n f(n) within a const# factor

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

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

$$c_1 > c_2 > 0$$

$\forall n \geq n_0$

2. TC for ->

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

$i = 1 \quad 2 \quad 4 \quad 8 \quad \dots \quad n$

$2^0 \quad 2^1 \quad 2^2 \quad 2^3 \quad 2^k$

$GP = a r^{k-1}$

$n = 1 \cdot 2^{k-1}$

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

$2n = 2^k$

$\log 2n = k \log 2$

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

$\boxed{\log n = k}$

$\therefore T(n) = \Theta(\log(x))$ //

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

$T(0) = 1$

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

$= 3$

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

$= 9 = 3^2$

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

$T(n) = 3^n$

$= O(3^n)$ //

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

$T(n) = T(n-2) - 3$ ---②

Put $n = n-2$

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

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

Put in ②

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

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

$= 8T(n-3) - 5 = 2^k T(n-k) - 1$

$(n-k) = 1$

$k = (n-1)$

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

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

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

5. while ($S <= n$)

{ $i++$;

$S = S + i$;

printf ("#");

}

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

$S = 3$

$i = 3$

$S = 6$

$i = 4$

$S = 10$

$i = 5$

$S = 15$

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

$S+1+2 \quad S+1+2+3 \quad S+1+2+3+4 \quad S+1+2+3+4+5$

$S = S+1+2+3+4 \ldots\ldots k$

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

$= k$

$k^2 + \frac{k}{2} \leq n$

$k^2 \leq n$

$k \leq \sqrt{n}$

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

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

i :=	1	2	3	4	. . .	k^2
i^2 =	2	4	9	16	. . .	

$k^2 \leq n$

$k \leq \sqrt{n}$

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

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++;
}
}

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

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

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

8. pen (int n)

```
{
  if (n = 1)
    return;
  for (i = 1 to n) ---> n
  {
    for (j = 1 to n) } ---> n
      printf ("k");
  }
  fun (n-3);
}
```

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

$T(n) = n^2 + 7(n-3)$

$= O(n^2)$ //

9. Void fun(int n)

```
{ for(i=1 to n) ---> n
  { for(j = 1; j 2 = n; j = j + i)
    {
      print("*");
    }
  }
}
```

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

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

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

$i = n-1$, $j \; j = 1, n \rightarrow n/n = 1$

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

(maximum Series)

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

10. n^k & c^n

$n = 1$

$n^k = 1^k \; c^n = c$

$n = 2$

$nk = 2^k, \; c^n < c^2$

$n = k, \; n^k - k^k, \; c^n = c^k$

∴ we can say that

for any value of $n > 0$

$n^k >= c^n$

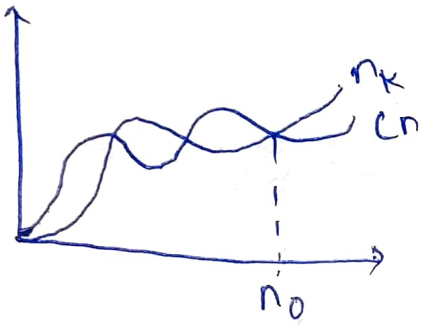
let $n^k = f(n) \cdot c^n = c\,g(n)$

$\therefore f(x) >= c\,g(n)$

$c_0 > 0, n_0 >= n_0$

$\therefore f(n) = O(c\,g(n))$

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



11. Extract Min ⟹

```
int extractMin(vector<int>& heap)
{ if (heap.empty())
  { return -1; ---> O(1)
  }
  Swap(heap[0], heap.back()); -> O(1)
  int minElement = heap.back();
  heap.popback(); ---> O(1) ---->
  heapify(heap, 0); ---> O(log n)
  return minElement;
}
```

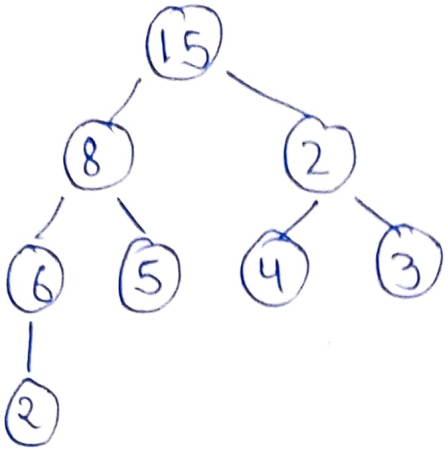
T(n) = O(log(n)) // ans.

No. of comp ⇒

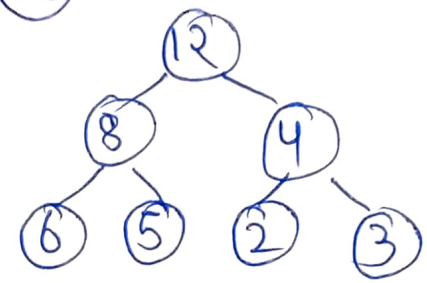
$(1 + \frac{n}{4}) + 2 \times \frac{n}{8} + 3 \times \frac{n}{16} + (n-1) * 1)$

= log(n) // harmonic mean

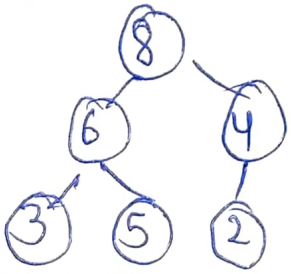
12.



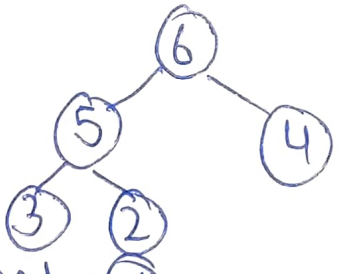
Delete Root 15



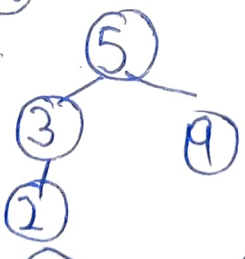
Delete Root 12



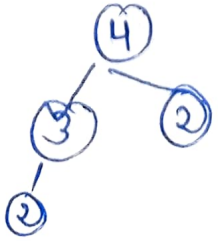
Delete Root 8



Delete Root 6



Delete Root 5



Delete Root 4

Delete 3
2

Delete 2
heap is compl