Name - Bhanupriya Chilwal
Course - B.TECH (C.S.E.)
Sem - 5, Sec - A, Roll. No - 1961030

Dt. _____ B+ 1
Pg. _____

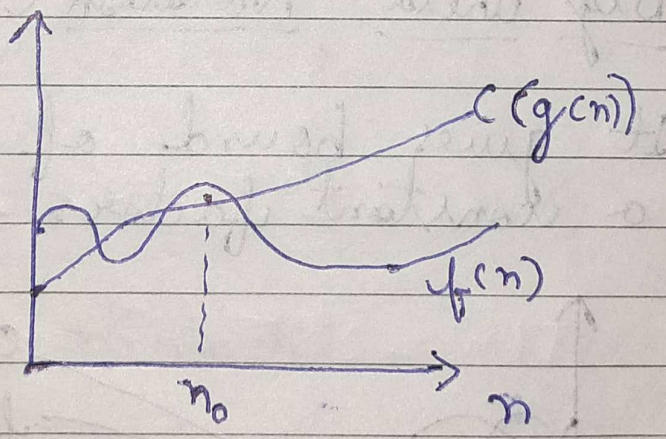
DAA - Assignment - 1

Ans. 1. = Asymptotic notations are used to represent the complexities of algorithms for asymptotic analysis.

These notations are used for very large inputs.

1. = **Big-oh (O) :-**

It gives upper bound for a function $f(n)$ to within a constant factor.



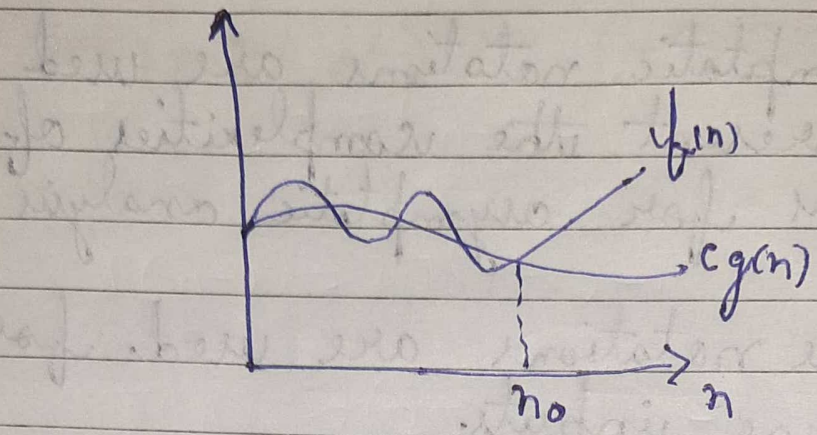
$f(n) \leq c \cdot g(n) \quad \forall\ n \geq n_0,\ c > 0$

example :- $O(n^2 + 3n) = O(n^2)$.

2. = **Big omega Notation (Ω) :-**

Big-Omega (Ω) notation gives a lower bound for a $f(n)$ to within a

constant factor.

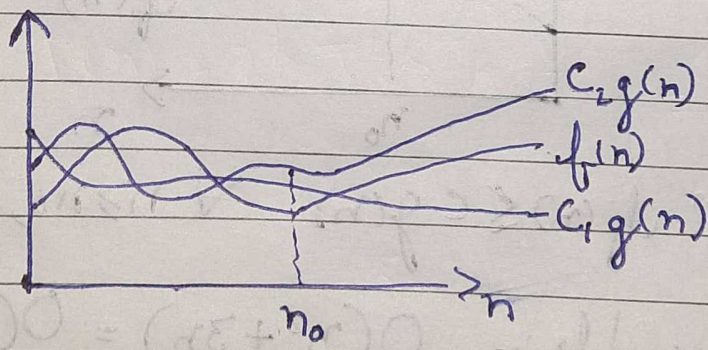


$\Omega(g(n)) = \{ f(n)$: There exist (+ve) constant c and no. such that $0 \leq c\,g(n) \leq f(n)$, $\forall\, n \geq n_0 \}$

e.g. - $\Omega(n \log n)$.

3. Big theta Notation (Θ) :-

It gives bound of function within a constant factor.



$\Theta(g(n)) = \{ f(n)$: There exist (+)ve constant c_1, c_2 and no. such that $0 \leq c_1 g(n) \leq f(n) \leq c_2 g(n)$, $\forall\, n \geq n_0 \}$

e.g. $\Theta(n^2)$

Ans.2. $O(\log_2 n)$

Ans.3. $$T(n) = \begin{cases} 3T(n-1), & n>0 \\ 1, & n=0 \end{cases}$$

$$T(n) = 3T(n-1) \quad -(1)$$

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

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

$$T(n) = 9T(n-2)$$

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

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

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

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

$$(n-k) = 0$$
$$n = k$$

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

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

$$\boxed{T(n) = O(3^n)}$$ Ans.

Dt
Pg B+ 4.

Ans. 4.

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

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

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

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

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

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

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

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

$T(n) = 2^k T(n-k) - (1+2+4 \dots +k)$

$T(n) = 2^k T(n-k) - \left(\frac{2^k - 1}{2-1}\right)$

$n-k=0$

$n=k$

$T(n) = 2^n T(0) - (2^n - 1)$

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

$T(n) = 1$

$\boxed{T(n) = O(1)}$

Ans. 5. 3, 6, 10, ... n

O(sqrt(n))

Ans. 6. O(sqrt(n))

Ans. 7. n
$\log_2 n$
$\log_2 n$

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

Ans. 8. $T(n) = T(n-3) + n^2$

$T(n-1) = T(n-4) + n^2$
$T(n-2) = T(n-5) + n^2$.
$T(n-3) = T(n-6) + (n-3)^2$

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

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

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

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

$n-k = 1$
$\Rightarrow n-1 = k$

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

$T(n) = 1^2 + 2^2 + 3^2 + \cdots + n^2$.

$T(n) = n^3$

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

Ans. 9. $i = 1, 2, \cdots$ n times

$i = 2, 4, 6, 8, \cdots$ $n/2$ times

$i = 3, 6, 9, 12, \cdots$ $\frac{n}{3}$

$\vdots$

$\frac{n}{n}$.

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

$= n\left(1 + \frac{1}{2} + \frac{1}{3} + \frac{1}{4} \cdots + \frac{1}{n}\right)$

$\Rightarrow n(\log_2 n)$.

$T(n) = O(n \log_2 n)$.

Ans. 10. For the functions n^k and a^n, what is the relation.

$k \geq 1$ & $a > 1$

relation is n^k is $O(c^n)$.

Ans.11.
```
void fun(int n)
{
  int j = 1, i = 0;
  while (i < n)
  {
    i = i + j;
    j++;
  }
}
```

0, 3, 6, 10, 15, ----, n.

kth term is $\frac{K(K+1)}{2}$

$$n = \frac{K^2+K}{2}$$

$K \simeq \sqrt{n}$

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

Ans.12. Recurrence relation of Fabonacci series :-

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

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

$$T(n) = 4T(n-4) + 3$$

$T(n) = 8T(n-6) + 7$

$T(n) = 16T(n-8) + 15$

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

for $T(n-2k) = T(0)$

$n = 2k$

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

$T(n) = 2^{n/2} T(0) + (2^{n/2} - 1)$

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

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

Hence, Space complexity = $O(n)$.
It depends on height of recursion tree.

Ans. 13. $n(\log n)$

```
void fun(n)
{ for(int j = 0; j < n; j++)
  { for(int i = 0; i < n; i = i * 2)
    { printf("*");
} } }
```

```
Void main()
{ fun();
}
```

→ n^3

```
#include <stdio.h>
void main()
{int n;
  cin>>n;
 for(int i=0; i<n; i++)
 { for(int j=0; j<n; j++)
 {for(int k=0; k<n; k++)
  { x++;
  }
 }
}
}
```

Dt
Pg
B+

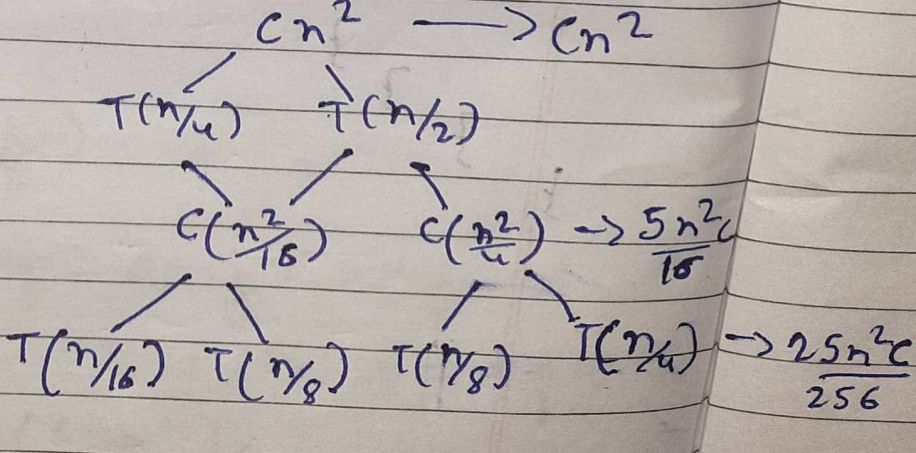
→ $\log(\log n)$

```
#include<bits/stdc++.h>

void fun(int n)
{
  if(n==2)
    return 1;
  else
    fun(sqrt(n));
}

void main()
{
  fun(100);
}
```

Ans. 14. $T(n) = T(n/4) + T(n/2) + cn^2$

$T(1) = c$

$T(0) = 0$



$T(n)$ = cost of each level.

$$T(n) = cn^2 + \frac{5cn^2}{16} + \frac{25cn^2}{256} + \text{----}$$

it is a G.P :-

with $a = n^2$

$$r = \frac{5}{16}$$

sum of g.p.

$$T(n) = cn^2 / \left(1 - \frac{5}{16}\right)$$

$$= \frac{16cn^2}{11}$$

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

Ans. 15

```
for (int i to n)
{
    for (int j = 1; j < n; j += i)
    {
        // O(1)
    }
}
```

$\Rightarrow \frac{25n^2c}{256}$

$\underbrace{n, \frac{n}{2}, \frac{n}{3}, \frac{n}{4}, \frac{n}{5}, \text{----} 1}_{K \text{ times}}$

$K = \log_2 n$

$n\left(1, \frac{1}{2}, \frac{1}{3}, \frac{1}{4}, \cdots \frac{1}{n}\right)$.

$(n(\log_2 n))$

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

Ans.16. for (int $i=2$; $i <= n$; $i = pow(i,k)$)

{
// O(1).
}

$2, 2^k, 2^{k^2}, 2^{k^3}, \cdots n$

G.P. $a = 2$

$r = 2^k$

k^{th} term $= ar^{k-1}$

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

let $k^{(k-1)} = x$

$k \log_k k = \log x$

$k = \log_k x$ — (1)

$n = 2^x$

$\log_2 n = x \log 2$

$x = \log n$

$\log x = \log(\log n)$

from ①:-

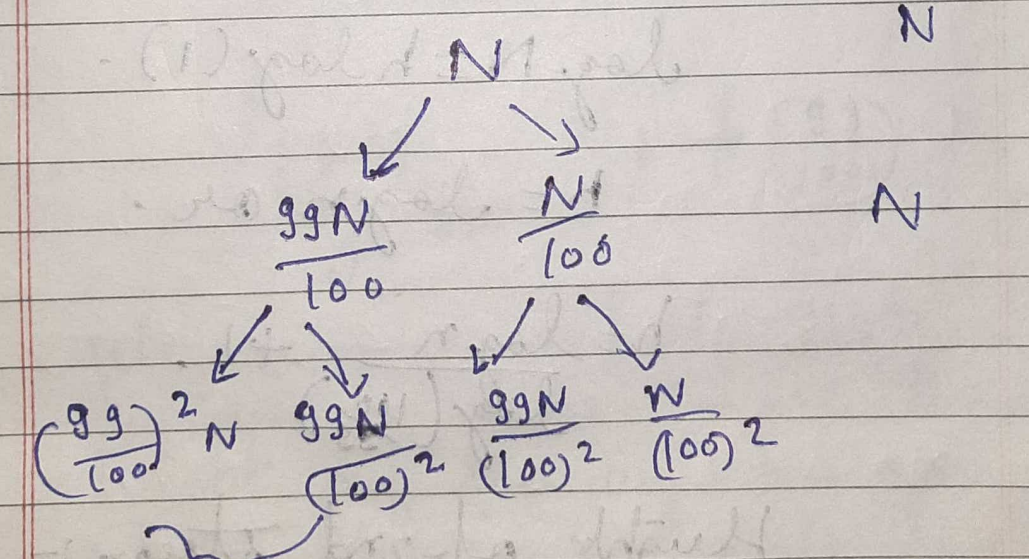
$k = \log(\log n)$

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

Ans-17) pivot is divided in 99% and 1%

$$T(n) = T\left(\frac{99}{100}N\right) + T\left(\frac{N}{100}\right) + N$$

Here we can use 2 extremes of a tree.
where starting point is N.



$$N\left(\frac{99\times99}{100*100} + \frac{99(1)}{100*100}\right) + \frac{100}{100*100}N$$

$$= \frac{99N}{100} + \frac{N}{100}$$

$= N$

cost of each level is N Only.

Total cost = height * cost of each level.

For 1st stream - $N, \frac{99N}{100}, \left(\frac{99}{100}\right)^2 N, \ldots$

$$\left(\frac{99}{100}\right)^{h-1} N = 1$$

$$\left(\frac{99}{100}\right)^{h-1} = \frac{1}{N}$$

$$N = \left(\frac{100}{99}\right)^{h-1}$$

$$\log N = h \log(1).$$

$$h = \log n \text{ or.}$$

$$h = \frac{\log n}{\log\left(\frac{100}{99}\right)} + 1.$$

Height of 2nd stream :-

$$N, \frac{N}{100}, \frac{N}{(100)^2}, \frac{N}{(100)^3}, \ldots$$

$N\left(\frac{1}{100}\right)^{n-1} = 1$

$N = (100)^{n-1}$

$(n-1)\log_{100} 100 = \log_{100} n$

$n = \frac{\log n}{\log 100} + 1$ & $h = \log_{100} N$

$T(n) = O(n \log_{100} n)$

time complexity = $O(n \log n)$

Height of both tree is

$\frac{\log n}{\log 100} + 1$ of $\left(\frac{1}{100}\right)$

$\frac{\log_{100} n}{\log_{100}\left(\frac{100}{99}\right)} + 1$ of $\left(\frac{99}{100}\right)$

We can conclude that if division is done more than height of tree will be more and when division ratio is less than height is less.

Ans.18. (a) n, $n!$, $\log n$, $\log \log n$, root (n), $n \log n$, 2^n, 2^{2n}, 4^n, n^2, 100.

Ans. $O(100) < O(\log(\log n)) < O(\log n)$ $O(\sqrt{n})$ $< O(n) < O(n \log n) < O(n^2) < O(2^n)$ $< O(2^{2n}) < O(4^{2n})$.

(b) $2(2^n)$, $4n$, $2n$, 1, $\log(n)$, $\log(\log(n))$

Ans. $O(1) < O(\log(\log(n))) < O(\log(n))$ $< O(\log 2n) < O(2 \log n) < O(n) <$ $O(n \log n) < O(\log(n!)) < O(2^n)$ $< O(4n) < O(n^2) < O(n!) < O(2(2^n))$

(c) $8^{\wedge}2n$, - - -

Ans. $O(96) < O(\log_8 n) < O \log_2 n <$ $O(\log n!) < O(n \log_8 n) <$ $O(n \log_2 n) < O(5n) < O(8n^3) <$ $O(7n^3) < O(n!) < O(8^{\wedge}2n)$.

Ans.19. void linearsearch (int arr[], int n, int key)

{ for (i=0 to i=n)
if (arr[i] == key;
cout << "found";
else

continue;
}

Ans-20 =>

Iterative Insertion sort:-

```
void Insertion sort (arr, n)
  { int i, temp, j;
    for (i = 1 to n)
  { temp = arr[i];
    j = i - 1
    while (j >= 0 && arr[j] > temp
      {arr[j + 1] = arr[j];
       j--;
      }
    arr[j + 1] = temp;
  } }
```

Recursive Insertion sort:-

```
insertion sort (arr, n)
  {if (n <= 1)
     return;
```

```
insertionsort (arr, n-1);
    last = arr[n-1];
    j = n-2;
while(j >= 0 and arr[j] > last)
    {arr[j+1] = arr[j];
     j--;
    }
    arr[j+1] = last;
}
```

Insertion sort is called online sorting because it don't know, the whole input, it might make decision that later turn out to be not optimal.

Other algorithms are off-line algorithms that are discussed in lectures.

Ans. 21.

	Time Complexity			Space
	Best	Avg.	worst	
Bubble sort	$O(n^2)$	$O(n^2)$	$O(n^2)$	$O(1)$
selection sort	$O(n^2)$	$O(n^2)$	$O(n^2)$	$O(1)$
Insertion sort	$O(n)$	$O(n^2)$	$O(n^2)$	$O(1)$
Merge sort	$O(n \log n)$	$O(n \log n)$	$O(n \log n)$	$O(n)$ recursion
Quick sort	$O(n \log n)$	$O(n \log n)$	$O(n^2)$	$O(n)$
Heap sort	$O(n \log n)$	$O(n \log n)$	$O(n \log n)$	$O(1)$

Ans. 22.

	inplace	Stable	Online sorting
Bubble sort	Yes	Yes	NO
Selection sort	Yes	NO	NO
Insertion sort	Yes	Yes	Yes
Merge sort	NO	Yes	NO
Quick sort	Yes	NO	NO
Heap sort	Yes	NO	NO

Ans.23. Binary search (arr, int n, key)

```
{ beg = 0;
  end = n-1;
  while (beg <= end)
  {mid = (beg + end) / 2
  if (arr[mid] == key)
  cout<<"found";
  else if (arr[mid] < key)
        beg = mid + 1;
  else
        end = mid - 1;
  } }
```

Time Complexity of linear search - O(n)
Space Complexity of linear search - O(1)

Time Complexity of Binary search -> O(log n).
Time Complexity of Binary search -> O(n).

Ans.24. $T(n) = T(\frac{n}{2}) + 1$.

Ans.