Pooja Bisht
BTECH, CSE, V SEM
U.RN 1961127 Section - C

Assignment - 1

Q1. What do you understand by asymptotic notation. Define different asymptotic notations with example.

Ans. Asymptotic notations are the mathematical notations used to describe the running time of an algo when the i/p tends towards a particular value or a limiting value.

⇒ There are mainly 3 asymptotic notations:

① Big-O notation

⇒ It represents the upper bound of running time of an algo.

⇒ This notation is called as upper bound of the algo, or a worst case of an algo.

⇒ $O(g(n)) = \{ f(n) :$ there exists +ve constants c & n_0 such that $0 \leq f(n) \leq cg(n)$ for all $n > n_0$ where c > 0 & $n > n_0$

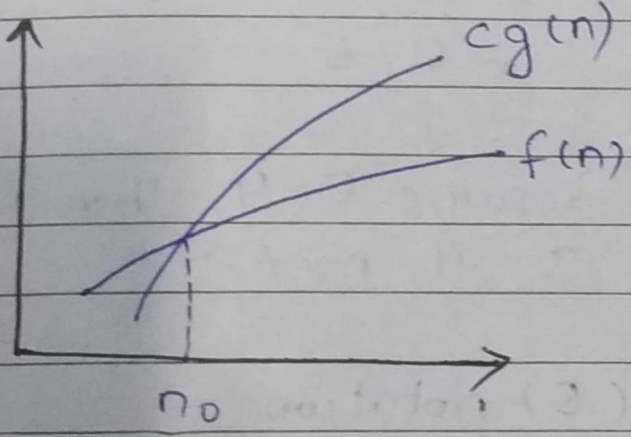
e.g. $f(n) = 3\log n + 100$

$g(n) = \log n$

$3\log n + 100 <= c * \log(n)$

$c = 150$ & $n > 2$

(undefined at $n = 1$)



② Big Omega (Ω) Notation

⇒ It represents the lower bound of the running time of an algo.

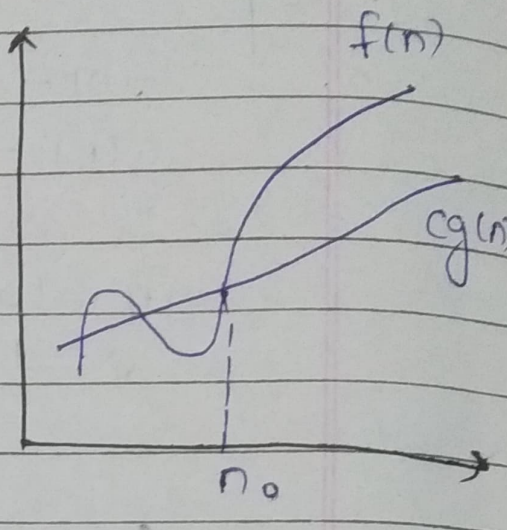
⇒ This notation is known as lower bound of an algo or best case of an algo

⇒ $\Omega(g(n)) = \{ f(n) :$ there exist +ve Constants c & n_0 such that $0 < cg(n) \leq f(n)$ $\forall n, n > n_0$

e.g

$f(n) = 3n + 2$

$c \cdot g(n) \leq f(n)$



$cn \leq 3n + 2$

$cn - 3n \leq 2$

$n(c-3) \leq 2$

$\Rightarrow n \leq \frac{2}{c-3}$

if we assume $c = 4$, then $n_0 = 2$

$c = 4, n_0 = 2$

③ Theta (Θ) notation

⇒ It enclose the function from above and below. It represents the upper & lower bound of running time of an algo.

⇒ Known as tight bound of an algo, an average case of algo

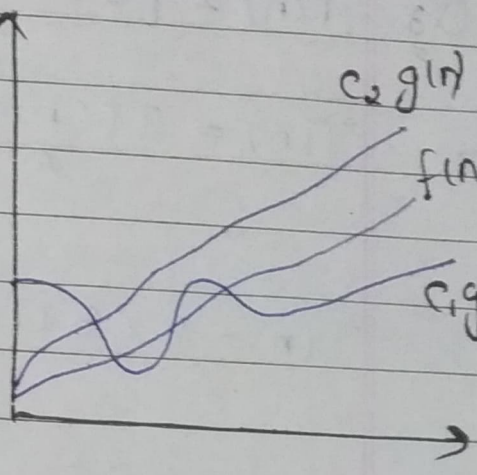
⇒ $\Theta(g(n)) = \{ f(n) :$ there exists +ve Constant c_1, c_2 & n_0 such that

$0 \leq c_1 * g(n) \leq f(n) \leq c_2 * g(n) \; \forall \; n > n_0$

Page:
Date:

⇒ eg -

$f(n) = 5n^2 + 16n^2 + 3n + 8$



$5n^3 \leq 5n^3 + 16n^2 + 3n + 8 \leq (5+16+3+8)n^3$

$5n^3 \leq f(n) \leq 32n^3$

$c_1 = 5, \quad c_2 = 32, \quad n_0 = 1$

$f(n) \hookrightarrow \Theta(n^3)$

Q2. What should be the time Complexity
for (i=1 to n) { i = i * 2}

i = 2, 4, 8, 16, ... k^{th} term ... n

$Gn = ar^{n-1}$

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

$n = 2^{k-1}$

$\log_2 n = (k-1)\log_2 2$

$k = \log_2 n + 1$

$O(n) = \log n$

Page:
Date:

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

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

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

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

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

$T(n) = 3 \times 3 \times 3T(n-3)$

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

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

$T(n) = 3^3 \times 3T(n-4)$

$T(n) = 3^4 \times T(n-4)$

|

General form:-

$T(n) = 3T(n-i)$ ---- (1) $[T(0) = 1]$

$T(n-i) = T(0)$

$n - i = 0$

$\boxed{n = i}$

Putting $n = i$ — in (1)

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

$= 3^n T(0)$ $\because T(0) = 1$

$T(n) = 3^n$

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

Page:
Date:

4. $T(n) = \begin{cases} 2T(n-1)-1, & \text{if } n>0 \\ \text{otherwise } 1 \end{cases}$

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

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

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

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

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

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

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

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

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

⋮

General form:

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

$T(n-i) = T(0)$

$n - i = 0$

$\boxed{n = i}$

$T(n) = 2^n T(0) - (1 + 2 + 2^2 + 2^3 + \dots 2^{n-1})$

$[T(0) = 1]$

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

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

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

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

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

$$\boxed{T(n) = O(2^n)}$$

Q5. What should be the T.C of :-

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

Ans

No. of steps (K)	S	i
0	0	1
1	1	2
2	3	3
3	6	4
4	10	5
5	15	6
6	21	7
!	!	!
Ks tep	n	

$T(n) = O(K)$

$S = 0, 1, 3, 6, 10, \ldots n$

$Sn = 1 + 3 + 6 + 10 + 15 + \ldots + n$

$Sn = 1 + 3 + 6 + 10 + \ldots + (n-1)th$

$0 = 1 + 2 + 3 + 4 + 5 + \ldots - n$

$n = 1 + 2 + 3 + 4 + \ldots$ K step

$n = \frac{K}{2}\left[2(1) + (K-1)1\right]$

$2n = K[2 + K - 1]$

$2n = K^2 + K \Rightarrow 2n = \left(K + \frac{1}{2}\right)^2 - \left(\frac{1}{2}\right)^2$

$2n + \left(\frac{1}{2}\right)^2 = \left(K + 1/2\right)^2$

$K + 1/2 = \sqrt{2n + (1/2)^2}$

$K = \sqrt{2n + (1/2)^2} - 1/2)$

$T(n) = T(K)$

$$T(n) = T\left(\sqrt{2n + (1/2)^2} - 1/2\right)$$

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

Q7. T.C of

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

  function (n-1); }
```

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

$$\left(T(n-1) = T(n-2) + (n-1)^2\right)$$

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

$$\left[T(n-2) = T(n-3) + (n-2)^2\right]$$

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

⋮

General term -

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

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

$$n = i+1 \Rightarrow \boxed{n-1 = i}$$

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

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

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

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

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

Q8. T.C of

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

```
for (K=1; K<=n; K=K*2)
    Count ++;
}
```

Ans $O(n \log n \log n)$

T.C = $\boxed{O(n(\log n)^2)}$

Q9. T.C of

```
Void function (int n)
{
    for (i=0 to n) {
     for (j=1; j<=n; j=j+i)
      printf ("#");
}}
```

Ans $O(n\sqrt{n})$

Q10. For the function n^k and a^n, what is the asymptotic relation b/w these function? Assume that $k >= 1$ and $a \neq 1$ are constant. find out the value of c & n_0 for what relation hold.

Page:
Date:

Ans If $c > 1$, then the exponential c^n for outgrows any term, so that ans is

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

Q12. Write recurrence relation for the recurrence function that prints fibonacci Series. Solve the recurrence relation to get time Complexity of the program. What will be the Space Complexity of this program and why?

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

$T(n-2) \simeq T(n-1)$

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

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

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

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

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

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

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

⋮

General Term :-

$$T(n) = 2^i T(n-i) + (2^0 + 2^1 + 2^2 + \dots 2^{i-1})c$$

$$n - i = 0$$

$$\boxed{n = i}$$

$$T(n) = 2^n T(0) + (2^0 + 2^1 + 2^2 + \dots + 2^{n-1})c$$

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

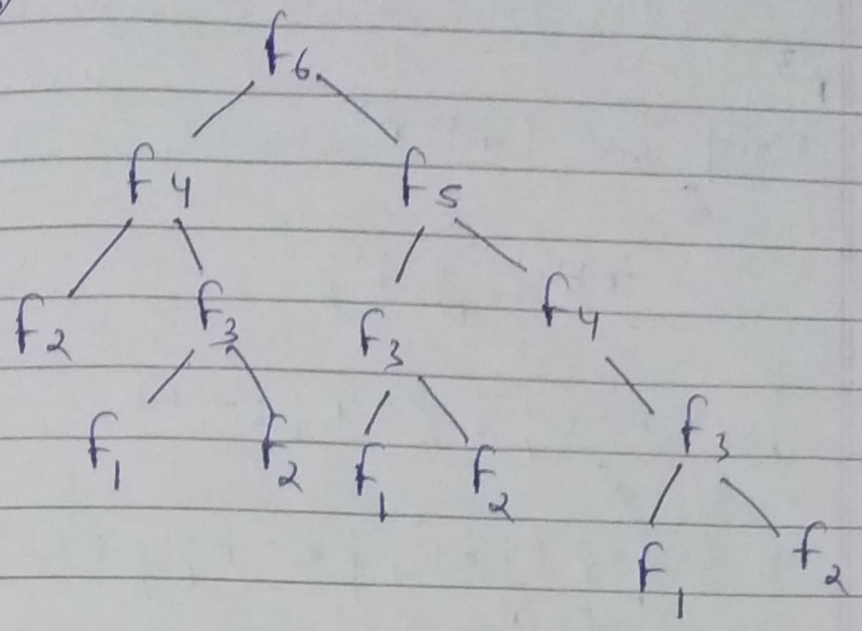
$$T(n) = 2^n(1) + \frac{2^0(2^n-1)}{2-1}c$$

$$T(n) = 2^n(1) + 2^0\frac{(2^{n-1}-1)}{(2-1)}c$$

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

$$\boxed{T(n) = O(2^n)}$$

fib. (6)



The max. depth is proportional to N, hence the space com. of fibonacci recursive is O(n).

Q13. Write programs which have Tc

(i) n log n

⇒

```
Void fun ()
{
  int i, j;
for( i = 1; i <= n; i++)
{
for(j = 0; j <= n; j = j * 2)

    Printf ("#");
Printf ("\n"); }}
```

(ii) n^3

```
Void fun (int n)
{
    int i, j, k;

for (i=0; i<=n; i++)
{
    for (j=0; j<=n; j++)
    {

        for (k=0; k<=n; k++)

            printf ("#"); } } }
```

(iii) log (log n)

Ans

```
Void Sieve of Eathosthenes (int n)
{
    bool prime [n+1];
memset (prime, true, Size of (prime));

for (int p=2, p*p<=n, p++)
{
    if (prime [p] = = true)
    {
        for (int i = p*p ; i<=n, i+=p)
            prime [i] = false;
    }
}
```

```
for (int p=2; p<=n; p++)

    if (prime [p]
        Cout << p << endl;

}
```

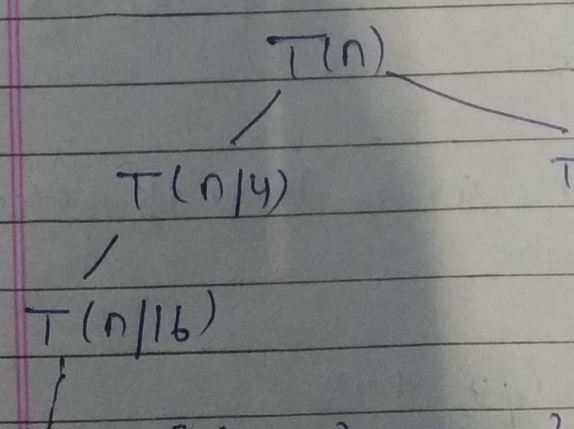
Q14. Solve the following recurrence relation $T(n) = T(n/4) + T(n/2) + cn^2$

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

$T(1) = c$

$T(n/2) = T(n/8) + T(n/4) + c(n^2/4)$

$T(n) = T(n/4) + 2T(n/16) + c(n^2/16) + n^2/4 + n^2)$



$$T(n) = c\left(n^2 + \frac{5n^2}{16} + \frac{25n^2}{256} + \cdots\right)$$

$$T(n) = n^2 c \left[1 + \frac{5}{16} + \frac{25}{16^2} + \cdots \right]$$

$$\boxed{T(n) = O(n^2)}$$

Q15. What is the T.C of following function fun()?

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

Ans For i = 1, the inner loop is executed n times

For i = 2, the inner loop is executed n/2 time

For i = 3, n/4 time

i = n, n times

Page:
Date:

Total time = $n + n/2 + n/3 + \ldots + n/n$

$= n(1 + 1/2 + 1/3 + \ldots 1/n)$

$= n \log n$

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

Q16. What should be the T.C of

```
for (int i=2; i<=n; i = pow(i,k))
{
    // Some O(1) expressions or
       statements
}
```

where k is a constant

Ans $O(\log(\log n))$

Q18. Arrange the following in increasing order of rate of growth

(a) $n, n!, \log n, \log\log n, \sqrt{n}, \log(n!), n \log n, 2^n, 2^{2n}, 4^n, n^2, 100$

Ans $100, \log\log n, \log n, \sqrt{n}, n, n \log n, n^2, 2^n, 2^{2n}, 4^n, n!$

b) $2(2^n)$, $4n$, $2n$, 1, $\log(n)$, $\log(\log(n))$, $\sqrt{\log(n)}$, $\log 2n$, $2\log(n)$, n, $\log(n)$, $n!$, n^2, $n\log(n)$

Ans 1, $\log(\log(n))$, $\sqrt{\log n}$, $\log n$, $\log(2n)$, $\log(n!)$, $2\log(n)$, n, $2n$, $4n$, $n\log(n)$, n^2, $2(2^n)$, $n!$

c) 8^{2n}, $\log_2(n)$, $n\log_6(n)$, $n\log_2(n)$, $\log(n!)$, $n!$, $\log_8(n)$, 96, $8n^2$, $7n^2$, $5n$

Ans 96, $\log_8 n$, $\log_2 n$, $\log(n!)$, $5n$, $n\log_6 n$, $n\log_2 n$, $8n^2$, $7n^3$, 8^{2n}, $n!$

Q19. Write linear Search pseudocode to Search an element in a sorted array with minimum Comparison.

```
linear_Search ( a, key)
        Comp ← 0, f ← 0
for i = 1 to a.length
        Comp ← Comp + 1
```

```
if a[i] == key
    print "Element founded"
      f = 1
if    f == 0
    print "Element not founded"

Print Comp.
```

Q20. Iterative Method of insertion Sort

```
Insertion_Sort (a)
for j = 2 to a.length
      key = a[j]
  i = j - 1
  while i > 0 and a[i] = key
      a[i+1] = a[i]
          i = i - 1
  a[i+1] = key
```

Recursive Method of insertion Sort

```
Insertion - Sort (a, n)
    if n ≤ 1
        return
Insertion - Sort (A, n-1)
    key = a[n-1]
        j = n-2
  while j ≥ 0 and a[j] > key
```

$a[j+1] = a[j]$

$j = j - 1$

$a[j+1] = key$

Insertion Sort Considers one input element per iteration and produces a partial Sol^n without Considering future elements. thats why it is called online Sorting

Other Sorting algorithm that have been discussed in features are -

1. Bubble Sorting
2. Selection Sorting
3. Quick Sort
4. Merge Sort
5. Heap Sort
6. Count Sort

Q21. Complexity of all Sorting algorithms that has been discussed in features.

	Best Case	Avg Case	Worst Case
Bubble Sort	$\Omega(N)$	$\Theta(N^2)$	$O(N^2)$
Selection	$\Omega(N^2)$	$\Theta(N^2)$	$O(N^2)$
Insertion	$\Omega(N)$	$\Theta(N^2)$	$O(N^2)$
Merge	$\Omega(N \log N)$	$\Theta(N \log N)$	$O(N \log N)$
Heap	$\Omega(N \log N)$	$\Theta(N \log N)$	$O(N \log N)$
Count	$\Omega(N \log(N))$	$\Theta(N+K)$	$O(N^2)$
Quick	$\Omega(N \log(N))$	$\Theta(N \log N)$	$O(N+K)$

Q22. Divide all the sorting alg into in place / stable / online Sorting

	In Place	Stable	Online
Bubble	Yes	Yes	Yes
Insertion	Yes	Yes	Yes
Selection	Yes	No	Yes
Merge	No	Yes	Yes
Quick	Yes	No	Yes
Heap	Yes	No	Yes
Count	No	Yes	Yes

Q23. Linear Search

```
Linear_Search ( a, key)
    found ← 0
for i = 1 to N
    if a(i) == key
```

found ← 1
print "Element found"
break
if found == 0
print "Element not found"

T.C - O(n)
S.C - O(1)

Binary Search (Iterative)

Binary_Search (a, Key, end, beg)

while beg ≥ end
mid = beg + (end - beg)/2

if mid == key
return mid

if a[mid] < key
beg = mid + 1

if a[mid] > key
end = mid - 1
return -1,

T.C - O(log n)
S.C : O(1)

Binary Search (Recursion)

```
Binary-Search (a, beg, end, key)
  if end ≥ beg
     mid = (beg + end)/2
  if a[mid] == item
        return mid + 1
  else if a[mid] < item
  return Binary-Search (a, mid+1,
                        end, key)
  else
     return Binary-Search (A, beg, mid-1,
                           end)
  return -1
```

T.C = O(log n)

S.C = O(1)

Q24. Write recurrence relation for binary recursive Search.

Ans $T(n) = T(n/2) + c$