DAA - ASSIGNMENT

NAME- RAHUL GUPTA
Uni. Rollno- 1961736
COURSE- B.TECH Branch- C.S
Sec- 'C' Sem- 5th.
Subject- Design and analysis algorithm

ASSIGNMENT - 1

1-) what do you understand by asymptotic notations Define different Asymptotic notations with example

Ans Asymptotic notation are the mathematical notations used to describe the running times of an algo when the i/p tends toward a particular value or a limiting value

There are mainly 3 asymptotic notations:-

(i) O notation

↳ It represents the upper bound of running time of an algorithm

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

⇒ $O(g(n)) = \{f(n)\}$: there exist positive constant c and n_0 such that $0 \leq f(n) \leq cg(n)$ for all $n \geq n_0$, where $c > 0$, & $n \geq n_0$

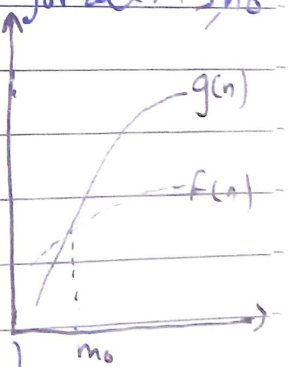
e.g. $f(n) = 3\log n + 100$

$g(n) = \log n$

$3\log n + 100 \leq c * \log(n)$

$c = 1 < 0$ & $n > \alpha$

(undefined at n_0)



(ii) 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 positive constant c and n_0 such that $0 \le c(g(n)) \le f(n)$ $\forall n, n > n_0\}$

⇒ eg $f(n) = 3n+2$

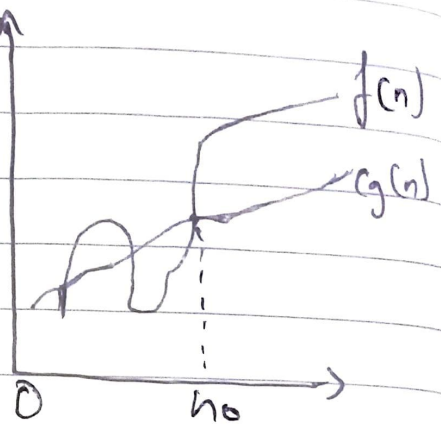
$c \cdot g(n) \le f(n)$

[c : constant, $g(n) = n$]

$cn \le 3n+2$

$cn - 3n \le 2$

$n(c-3) \le 2 \quad \therefore n \le \frac{2}{c-3}$



n : no. of i/p.

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

$[c = 4, n_0 = 2]$

(iii) Theta notation (θ)

⇒ It encloses the function from above & below. Since, it represent the upper & lower bound of running time of an algo

⇒ This is known as tight bound of an algo, an average case of algo.

⇒ $\theta(g(n)) = \{f(n)$: there exist +ve constant c_1 & c_2 & n_0 such that $0 \le c_1 * g(n) \le f(n) \le c_2 * g(n)$ $\forall n > n_0$

⇒ e.g $f(n) = 5n^3 + 16n^2 + 3n + 8$

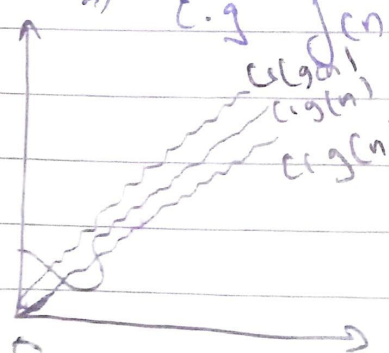
$c_1 g(n)$, $c_2 g(n)$

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

$5n^3 \le f(n) \le 32n^2$

$c_1 = 8, \quad c_2 = 32, \quad n_0 = 1$

$f(n) \longleftrightarrow \theta(n^3)$



Date

2 what should be the time complexity for:
for(i=1 to n){i=i*2}

Sol
i = 2, 4, 8, 16 ---- R^{th} term ---- n

$a_n = ar^{n-1}$

$a_n = 1(2)^{R-1}$

$n = 2^{R-1}$

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

$R = \log_2 n + 1$

$O(n) = \log n$

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

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

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

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

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

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

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

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

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

:

General form:-

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

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

$n-i = 0$

$n = i$

Putting $n=i$ in eq-(1)

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

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

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

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

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

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

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

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

⋮

general form:-

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

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

with $n-i=0$

$n=i$

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

$[T(0)=1]$

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

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

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

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

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

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

5. what should be the TC of :-

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

No. of steps (K)	S	P
0	0	1
1	1	2
2	3	3
3	6	4
4	10	5
5	15	6
6	21	7
⋮	⋮	⋮
K steps	n	⋮

$T(n) = O(K)$

$S: 0, 1, 3, 6, 10 \cdots n.$

$Sn: 1+3+6+10+15+\cdots+n$

$Sn = 1+3+6+10+15+\cdots+(n-1)+n$

$0 = 1+2+3+4+\cdots n.$

$n = 1+2+3+4+\cdots K \text{ steps}$

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

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

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

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

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

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

$$T(n) = T(K)$$

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

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

7. Time complexity of

```
void Junctn (int n)
{  for (i=n/2; 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 $O(n \log n \log n)$

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

8.) Time Complexity of Junctn (int n)

```
{ if (n==1) return 1;
   for (i=1 ton)
    for (j=1 ton)
     printf ("*");
      Junction (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 form:

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

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

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

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

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

Date

$n = i+1 = n-1 = i$

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

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

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

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

9.) T(of

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

Sol. $O(n\sqrt{n})$

10.) For the function n^2 & a^n, what is the asymptotic relation b/w these function? Assume that $k \geq -1$ & $a > 1$ are constants. Find out the value of c and n_0 for what relation holds.

Ans If $c > 1$, then the exponential c^n for asymptotically any term; so that answer is:

n^2 is $O(c^n)$

12.) Write the recurrence relation that prints Fibonacci series. Some recurrence to get of TC of program. what will the space complexity of this program & why?

Ans2 $T(n) = T(n-1) + T(n-2) + c$

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

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

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

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

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

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

$\vdots$

General form :-

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

$n - i = 0$

$n = i$

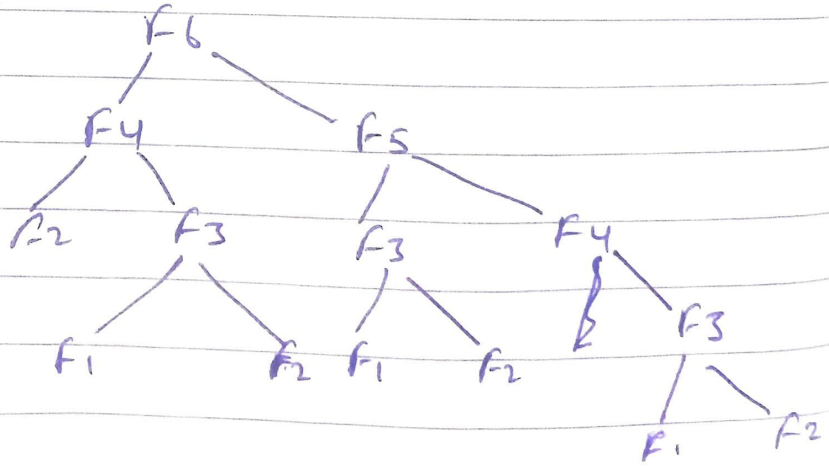
$T(n) = 2^n T(0) + (2^0 + 2^1 + \cdots 2^{n-1})c$

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

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

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

Fib (6).



The Max depth is proportional to the N, hence the space of Fibonacci recursive is $O(n)$.

Date

3.) Write programs which have T.C -

(i) nlogn

=)
```
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;j<=n;i++)
  {
  for (j=0;j<=n;j++)
  {
  printf("# ");}}}
```

(iii) log (logn)

=)
```
Void Seive of Erathosthenes (int n)
{ bool prime [n+1];
  memset (prime, true sizeof (prime));
  for(int p=2; p*p<=n; p++)
  {
    prime[i]=false;
  }}
```

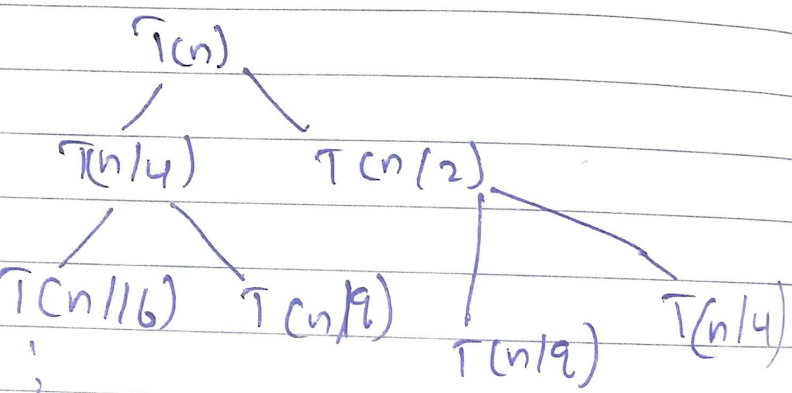
```
for(int p = 2; p<=n; p++)
    if (prime(p))
        Cout << p << endl;
    }
```

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

Ans $T(n) = T(n/4) + T(n/2) + cn^2$ $T(1) = c$ $n = n/2$

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

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



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

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

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

Q.15 What is the time complexity of the following

```
Function fun()
    int fun(int n)
    {
```

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

Ans: for i=1, the inner loop is executed n times
for i=2, " " " " " n/2 times
for i=3, " " " " " n/3 times
⋮
for i=n, the inner loop is executed n/n times

Total times $= n + n/2 + n/3 + \cdots + n/n$

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

$= n \log n$

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

(6) what should be the time complexity of

```
for(int i=2; i<=n; i=pow(i,K))
  {
    // Some O(1) expression or statement
  }
```

where K is a Constant.

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

Date

(a.) Arrange the following in increasing order of rate of growth

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

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

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

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

(c.) $96, \log_2(n), n\log_1(n), n\log_2(n), \log(n!), n!, \log(n^2), 96, 9n^2, 7n^3, 5n$.

Ans $96, \log_0 n, \log_2 n, \log(n!), 5n, n\log_6 n, n\log_2 n, 9n^2, 7n^3, 9^{2n}, n!$

19.) Write linear search pseudocode to search an element in a sorted array with minimum comparison.

```
Ans  Linear Search (A0, key)
       comp ← 0, f ← 0
       for(i = 0 to A0.length)
           comp ← (comp + 1)
```

Date / /

```
if A0[i] == key
    print "element founded"
        j = 1
    if j == 0
        print "element not found"
        print comp
```

20.) Write pseudo code for iterative & recursive insertion sort. Insertion sorting is called online why? what about other sorting algo that has been discussed in lecture?

Ans Iterative method of insertion sort

```
INSERTION_SORT (A0)
    for j = 2 to A0.length
        key = A0[j]
        i = j - 1
    while i > 0 and A0[i] > key
        A0[i+1] = A[i]
        i = i - 1
        A0[i+1] = key
```

Recursive Method of Insertion sort.

```
INSERTION_SORT (A0, n)
    if (n ≤ 1)
        return
    INSERTION_SORT (A0, n-1)
```

Key = [n-1]

j = n-2;

while j ≥ 0 and A₀[j] > Key

A₀[j+1] = A₀[j]

j = j-1

A₀[j+1] = Key

Insertion Sort consider one i/p element per iteration & procedures a partial Solution without considering future elements. That's why it is called Online Sorting.

other Sorting algorithm that means been discussed in lectures are.

(i) Bubble Sorting
(ii) Selection Sorting
(iii) Quick Sort
(iv) Merge Sort
(v) Heap Sort
(vi) Counting Sort.

2. Complexity of all Sorting algorithm that has been discussed in lectures.

	Best Case	Average Case	Worst Case
Bubble Sort	$\Omega(N)$	$\Theta(N^2)$	$O(N^2)$
Selection Sort	$\Omega(N^2)$	$\Theta(N^2)$	$O(N^2)$
Insertion Sort	$\Omega(N)$	$\Theta(N^2)$	$O(N^2)$
Merge Sort	$\Omega(N \log N)$	$\Theta(N \log N)$	$O(N \log N)$
Heap Sort.	$\Omega(N \log N)$	$\Theta(N \log N)$	$O(N \log N)$

Quick Sort	$\Omega(N \log N)$	$\Theta(N \log N)$	$O(N^2)$
Counting Sort	$\Omega(N+K)$	$\Theta(N+K)$	$O(N+K)$

22) Divide all sorting algorithms into in place stable Online Sorting

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

23) Write recursive/iterative pseudo code for binary search. what is the time & space complexity of linear & binary (Recursive & Iterative)

Ans Linear Search

```
Linear_Search (A, Key)
    found ← 0
    for i = 1 to N
        if A(i) = Key
            found ← 1
            print "Element found"
            break.
```

Date /

If found == 0
print "Element not found"
Time complexity - O(n)
Space complexity - O(1)

BINARY Search (Iterative)

BINARY_SEARCH (A, beg, end, Key)
while beg ≤ end
mid = beg + (end - beg)/2
If mid == key
return mid
If A(mid) == item
return BINARY_SEARCH (A, mid+1, end, key
Else
return BINARY_SEARCH (A, beg, mid-1, end)
return 1
Time complexity - O(log n)
space complexity - O(1)

24.) Write recurrence relation for binary recursive search.

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