DAA TCS - SOS

Name :- Isthiqua Moin Tanjavur

Sec :- H

Roll no :- 2014682

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

Ans1 Asymptotic notation are the mathematical notation used to describe the running time of an algorithm when the input toil words a particular value or a limiting value. Asymptotic Notation is a way to compare function that ignores constant for and small input sizes.

Types of Asymptotic Notations →

① Big Theta (θ) → Tight bound, complexity represented is like average value or range within which the actual time of execution will be

② Big Oh (O) - This is used for upper bound of algorithm or worst case of an algorithm. It tells that a function w never exceed specified time for any value of input n.

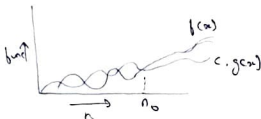③ Big Omega (Ω) → Used to define lower bound of any algorithm or the best case of a algorithm.
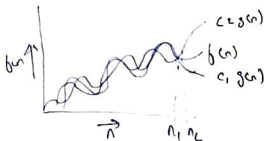
1 Big OH (o) →
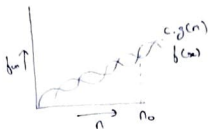
$$f(n) = O(g(n)) \qquad g(n) \text{ is tight upper bound}$$

2 Big omega (Ω)

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

3 Theta (θ) :- Gives both upper and lower bound

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

Q2 What should be complexity of for (i=1 ton) {i=i*2}

$$\sum_{i=1}^{n} step + L$$

$1, 2, 4, 8, \cdots \cdots n \quad (\text{where})$

$$\Rightarrow 2^0, 2^1, 2^2, 2^3 \cdots \cdots \cancel{2^k} = n$$

taking log

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

$$(k-1) = \log n$$

$$k = \log n + 1$$

$$= O(\log n)$$

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

$$T(n) = \begin{cases} 3T(n-1) & \text{if } n > 0 \\ 1 & \text{otherwise} \end{cases}$$

$$T(n) = 3T(n-1) \quad \forall \quad n > 0 \quad -①$$

put $n = (n-1)$

$$T(n-1) = 3T(n-2) \quad -②$$

putting ② in ①st

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

$$= 3^2 T(n-2) \quad -③$$

putting n→n-2 in eq ①

$T(n-1) = 3^1 (n-3)$ —④

$T(n) = 3^3 T(n-3)$ · ⑤

$T(n) = 3^k T(n-k)$ —⑥

Base case → $T(0) = 1$

$n-k = 0$

$n = k$

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

$= 3^n T(0)$

$= 3^n$

Q4 $T(n) = \{ 2T(n-1)-1 \ \text{if} \ n > 0 \ \text{otherwise} \ 1 \}$

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

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

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

$\Rightarrow 2^2 (T(n-2)) -2-1$

$\Rightarrow 2^2 (2T(n-3)-1)-2-1$

$\Rightarrow 2^3 T(n-3) - 2^2 - 2^1 - 2^0$

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

$\Rightarrow 2^n - \{2^n-1\}$

$\Rightarrow 2^n - 2^n + 1 = 1$

$T(n) = 1$

⑤

Q5 What should be time complexity - $int \ i = 1 \ - - - - - printf("\#");$

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

$1, 3, 6, 10 \ - - - n \ terms$

$S = 1+3+6+10+ \cdots \ k$

$0 = 1+2+3+4 \ \cdots \ k$

$$\frac{k(k-1)}{2}$$

$$n \approx k^2$$

$$k = \sqrt{n}$$

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

Q6 Time complexity of
```
void function (int n) {
  int i, count=0;
  for (i=1; i+i <=n; i++)
    count ++;
}
```

$1, (2)^2, (3)^2, (4)^2 \ - - - - n$

$1, 2, 3, 4 \ - - - \sqrt{n}$

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

Q7 Time complexity of-

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

$$\sum_{i=n/2}^{n} \sum_{J=1}^{n} \sum_{k=1}^{(J*2)} 1$$

$$\sum_{i=n/2}^{n} \sum_{J=1}^{n} \log(n)$$
$$\text{step} \to 2$$

$$\sum_{i=n/2}^{n} (\log n)^{2}$$

$$\Rightarrow \left(\frac{n}{2} + 1\right)(\log n)^{2} \Rightarrow T(n) = O(n \log n^{2})$$

Q8 Time complexity - - - - . (n-3).

$$\sum_{i=1}^{n} \sum_{J=1}^{n} 1 \Rightarrow \sum_{i=1}^{n} n = n^{2}$$

$$T(n) = T(n-3) + n^{2} - ①$$

putting $n = n-3$ in eq①

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

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

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

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

$T(1) = 0$

$n - 3k = 1$

$k = n - \dfrac{1}{3}$

$T(n) = n^2 + (n-3)^2 + \cdots \quad (n-k)^2$

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

Q9 Time complexity ----- y b

$\sum\limits_{i=1}^{n} \sum\limits_{\substack{v=1 \\ step = 1}}^{n} 1$ n^2

$\sum\limits_{i=1}^{n} \left(\dfrac{(n-1)}{1} + 1 \right) \Rightarrow (n-1) \sum\limits_{i=1}^{n} \dfrac{1}{i} + \sum\limits_{i=1}^{n} 1.$

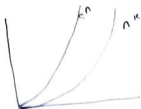$\Rightarrow (n-1) \sum\limits_{i=1}^{n} \dfrac{1}{i} + n$

$(n-1) \log n + n$

$T = O(n \lg n)$

Q10 For function n^x and a^n what is asymptotic relationship between then. Find out value of c and n_0 for which relationship holds.

Ans 10

$n^k = o(a^n)$

$n^k \leq a^n . c$ $\forall\ c > 0$ and $n \geq n_0$

let $n = n_0$

$n_0^k \leq c . a^{n_0}$

~~let n = n_0~~

$n_0^3 \leq c . 3^{n_0}$ $k = c = 3$ (say)

$\Rightarrow c \geq 1$ & $n_0 \geq 1$

Q11 what is time complexity -

```
void bun ( int n) {
  int j= 1 , i=0;
  while ( i<n) {
    i = i+j ;
    j++ ; } }
```

J	i
1	0
2	1
3	3
4	6
5	10
6	15
7	21

$S = 0, 1, 3, 6, 10, 15 \cdots$ ①

$S = 0, 1, 3, 6, 10 \cdots$ ─ ②

$0 = 0, 1, 2, 3, 4, 5 \cdots k-n$

$n = \dfrac{k(k-1)}{2}$

$n \simeq k^2$
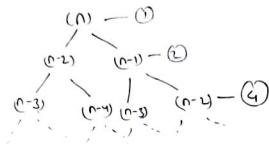
$k = \sqrt{n}$

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

Q12 Write recurrence relation for recursive function that prints fibonacciso.
Find time and space complexity?

$0,1,1,2,3 - \cdots - \cdots$ Tin

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

$(n) — ①$

$(n-2) \quad (n-1) — ②$

$(n-3) \quad (n-4) \; (n-5) \quad (n-4) — ④$

$T = 1 + 2 + 4 + 8 - \cdots - 2^n$

$a = 1$
$r = 2$

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

$= 2^{n+1} - 1$

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

Space complexity = $O(n)$
because max stack frame is
same as the longest node.

Q13 Write time with complexity - $n(\log n)$, n^{13}, $\log(\log n)$?

(i) for $(int \; i = 0; \; i \leq n; \; i++)$
$\{$
$\quad$ for $(int \; j = 1; \; j \leq n; \; j \ast = 2)$
$\quad\quad$ print $(``")$;

```
int i=1, p=hi
while ( a <= n) {
    a *= bi;
    b *= a;
}
```
$\sigma (\log_b \log_b n)$

(ii)
```
int func ( int n)
{
    if ( n <= 2 )
        return 1;
    else
        return func ( floor ( sqrt (n)) + n);
}
```
$\log (\log n)$

(iii)
```
for ( i = 0; i < n; i++)
    for ( int j = 0; j < n; j++)
        for ( int k = 0; k < n; k++)
            print (" *");
```
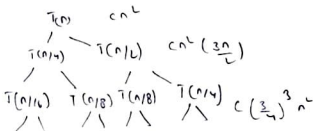n^3

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

$cn^2 \left(\dfrac{3n}{4} \right)$

$c \left(\dfrac{3}{4} \right)^3 n^2$

$cn^2 \left(\dfrac{3}{4} \right)^k n$

$\dfrac{n}{2^k} = 1$

$n = 2^k$

$k = \log n$

$$T(n) = cn^2 \left[1 + \frac{3}{4} + \left(\frac{3}{4}\right)^2 + \cdots + \left(\frac{3}{4}\right)^{lg\,n} \right]$$

$$cn^2 (1)$$
$$= n^2$$

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

Q15
```
int fun (int n) {
    for (int i=1; i <= n; i++)
        { for (int j=1; j < n; j += i)
```

$$\begin{array}{c} 3 \\ 3 \end{array}$$

$$T(n) = \sum_{i=1}^{n} \sum_{j=1}^{n-1} (1)$$

$$\Rightarrow \sum_{i=1}^{n} \frac{n-1}{i}$$

$$\Rightarrow (n-1) \sum_{i=1}^{n} \frac{1}{i} \quad \Rightarrow (n-1)\left(\frac{1}{1} + \frac{1}{2} + \frac{1}{3} + \frac{1}{4} \cdots n\right)$$

$$= (n-1) \, lg\, n$$

$$T(n) = (n\,lg\,n)$$

Q16
```
for (int i=2; i <= n; i = pow(i,k))
        { _j-o(1)
```

$$i = 2, \; 2^k \; 2^{k^2} \; 2^{k^3} \; \cdots \; 2^{k^k}$$

$$lg(n) = k^n \, lg\,2$$
$$lg_k(lg(n)) = k \, lg\,k$$
$$n \, lg(lg(n)))$$

Q17 Write a recurrence — — — analysis?

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

2 If we take longer branch i.e $\frac{99n}{100}$
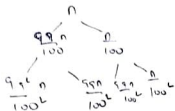
$$T \Rightarrow \log \frac{100}{99} \quad n \approx \log n$$

$$\left(\frac{99}{100}\right)^k n = 1$$

$$n = \left(\frac{100}{99}\right)^k \qquad k = \log \frac{100}{99} n$$

$$k = \log \frac{9 \, 100}{99} n$$

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

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

Q 18 Increasing order of growth →

(a) $100 < \log \log n < \log n < \sqrt{n} < n < n \log n < n^2 < n^3 < 2^n < n^n$
$\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad < n!$

(b) $1 < \log \log (n) < \sqrt{\log (n)} < \log (n) < 2n < 4n < n^2) < n \log (n)$
$\qquad < 2 \log (n) < n < n \log n = \log (n!) < N!$

(c) $a \, b < \log 8^n < n \log n \, \& \, e = n \log n < 5n < 8 n^2 < 7^n < 3^n$

Q19 Write linear search pseudocode to search an element in a sorted array with minimum comparision.

```
for (i=0 to k-1)
{
    if (arr[i] = key)
    {
        return i;
    }
    return -1;
}
```

Q20 Write - - - - - - - Insertion?

I_Iterative Insertion Sort

insertion Sort (arr, n)

loop from i=1 to i=n-1

Pick element arr[i] and insert it into sorted sequence

it arr[0---i-1]

Recursive Insertion Sort

insertion Sort (arr, n)

```
{
    if n <=1
        return
    recursively sort n-1 element
        insertion Sort (arr, n-1)
    Pick 1st element arr[i] and insert
    it into sorted arr[0---i-1]
}
```

Insertion sort considers one input element per iteration and produces a partial solution without considering future elements. It is also called online sorting Algorithm

Q 21 Complexity of all sorting algorithm?

	Algorithm	Best case	Average Case	Worst case
(1)	Bubble Sort	$O(n^2)$	$O(n^2)$	$O(n^2)$
(2)	Selection Sort	$O(n^2)$	$O(n^2)$	$O(n^2)$
(3)	Merge Sort	$O(n\log n)$	$O(n\log n)$	$O(n\log n)$
(4)	Insertion Sort :	$O(n)$	$O(n^2)$	$O(n^2)$
(5)	Quick Sort	$O(n\log n)$	$O(n\log n)$	$O(n^2)$
(6)	Heap Sort	$O(n\log n)$	$O(n\log n)$	$O(n\log n)$

Q 22 Divide all sorting algorithms inplace/stable / online sorting.

Algorithm	Inplace	Stable	Online Sorting
Bubble Sort.	✓	✓	✗
Selection Sort	✓	✗	✗
Insertion Sort	✓	✓	✓
Merge Sort	✗	✓	✗
Quick Sort	✗	✗	✗
Heap Sort	✓	✗	✗

Q 23 Write recursive . . . . . Binary search.

```
int binary search ( int ar [], int l , int n)
  {
    while ( l< r)
      {
        int m = ( l+r) / L;
        if ( ar [m] = x)
          return m;
        else if ( ar [m] < x)
          l= m+1;
        else
          r = m -1;
      }
      return -1;
  }
```

Recursive Binary Search

```
int Binary Search ( int arr [], int L, int r , int x)
  { if ( L > r)
      return -1
      int m= ( l+r) / L ;
      if ( arr [m]=n)
          return m;
      else if ( arr [m] < n)
      return Binary Search ( arr, m+1, r, x );
      else
          return Binary Search ( arr, l, m-1 , x)
```

Iterative Binary search

Time complexity => Best = O(1)   Avg = O(log n) , Worst = O(log n)
Space => O(1)

Recursive binary =>
Time complexity => Best = O(1)   Average = O(log n)   Worst = O(log n)
Space complexity => Best = O(1), Avg = O(log n)   Worst = O(log n)

Q 24 → T(n) = T (n/L)+1 = T (n) = O(log n)