Name : Boddy Sreeja Section : A Roll no : 17

Assignment 1.

1) Asymptotic notation :- It is a method/ language using which we can define the running time of the algorithm based on input size.

Types of Asymptotic notations.

i) Big O (O-notation) :- It represents the upper bound of the running time of an algorithm. It gives the worst-case complexity of an algorithm.

ii) Omega Notation (Ω) :- It represents the lower bound of the running time of an algo. It gives the best-case complexity of an algorithm.

iii) Theta Notation (Θ). It encloses the function from above and below. Since it represents the upper and the lower bound of the running time of an algo. It is used for analyzing the average-case complexity of an algorithm.

2) Time Complexity of for(i=1 to n) {i=i*2}

value of i

1

2 the series is in GP.

4 $a=1$

8 $r=2/1 = 2$

:

n

So $t_k = ar^{k-1}$

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

$2n = 2^k$.

$k = \log_2 2n$

So time complexity is $O(\log_2 n)$.

Scanned with CamScanner

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

if $n = n-1$

$T(n-1) = 3T(n-2)$ — (2) sub in (1)

$T(n) = 3(3T(n-2))$ — (3)

if $n = n-2$

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

So the general eq is

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

if $n-k = 1$

$n = k$

So $3^n T(0) = 3^n$ as $T(0) = 1$

So Time complexity $= O(3^n)$.

---

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

if $n = n-1$

$T(n-1) = 2T(n-2) - 1$ — (2)

~~Ed~~ Substituting in (1)

$T(n) = 4T(n-2) - 2$ — (3)

if $n = n-2$

$T(n-2) = 2T(n-3) - 1$ — (4) put in 3

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

So the general eq is

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

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

So $2^n T(0) - k$

$= 2^n - n$

$O(2^n)$

5. Time complexity

value of i	value of s
1	1
2	3
3	6
	⋮
	k

k → So sum of natural nos.

$$\frac{k*(k+1)}{2}$$

$$\frac{k^2+k}{2} \qquad k^2 \le n \Rightarrow \sqrt{n} = k$$

So Time complexity is $O(\sqrt{n})$

6. Time complexity

value of i	condition checked $(i*i) \le n$
1	1.
2	4.
3	9.
⋮	⋮
k	k^2

$k^2 \le n$

$k \le \sqrt{n}$

Time complexity $= O(\sqrt{n})$.

7. Time complexity of-

for k loop - $\log(n)$

for j loop - $\log(n)$

for i loop - $\frac{n}{2} \Rightarrow n$

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

8. Time complexity.

$T(n) = T(n-3) + n^2$ — (1)

put $n = n-3$ in eq (1)

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

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

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

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

So gen eq is

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

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

$T(1) = 0$

$n - 3k = 0$

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

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

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

9. for i loop it is $O(n)$

for j loop it is $\sum_{j=1,\ \text{step } i}^{n}$

$n = a + (l-1)$

$n = 1 + (k-1)i$

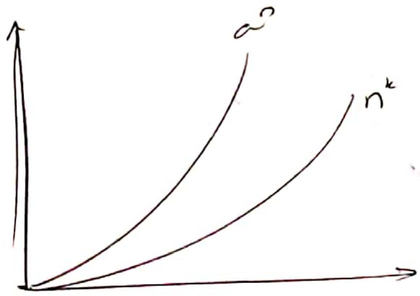
$\frac{n-1}{i} + 1 = k$

$\sum_{i=1}^{n} 1/i$

$\log n$

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

10.



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

$n^k \le a^n \cdot c \quad \forall\ c > 0\ \&\ n \ge n_0$

let $n = n_0$

$n_0^k \le c \cdot a^{n_0}$ if $k = a = 2$

$n_0^2 \le c \cdot 2^{n_0}$

$c \ge 1\ \&\ n_0 \ge 1.$

11.

i	j
0	1
1	2
3	3
6	4
10	5
. . .	

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

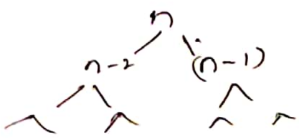
$K = \sqrt{n}$

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

12.

0, 1, 1, 2, 3 . . . n

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



$\Rightarrow 1 + 2 + 4 + \ldots 2^n$

$a = 1,\ r = 2$

$T = \frac{a(r^{\text{terms}} - 1)}{r-1}$

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

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

13.)

$n\log n$

```
for (int i=0 ; i<=n ; i++)
  for (int j=0 ; i<=j ; j=j*2)
    int x = x+j;
```

n^3

```
for (int i=0; i<=n ; i++)
  for (int j=0; j<=n ; j++)
    for (int k=0; k<=n ; k++)
      printf("Hello");
```

$\log(\log n)$

```
int x=1 , b=1
while (x<= n) {
  x = x*b;
  b *= 2;
}
```

14.

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

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

$k = \log n$

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

$= cn^2 (1)$

$= n^2$

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

15.

i	j
1	(1, 2, 3, .. n)
2	(1+

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

$\downarrow$ n $\quad \frac{n-1}{i}$ $\log n$

$\Rightarrow O(n\log n)$

Scanned with CamScanner

16) $i = 2 \cdot 2^k \cdot 2^{k^2} \; 2^{k^3}$

$$2^{k^x} = n$$

$$\log n = k^x$$

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

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

17. $T(n) = T\left(\frac{99}{100} n\right) + n/100 - (1)$

$T(1) = 0$.

if $n = \frac{99}{100} n$

$$T\left(\frac{99}{100} x\right) = T\left(\left(\frac{99}{100}\right)^2 n\right)$$

$$T(n) = T\left(\left(\frac{99}{100}\right)^2 n\right) + n/100$$

general eq $\quad T(n) = T\left(\left(\frac{99}{100}\right)^k n\right) + \frac{kn}{100}$.

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

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

$$k = \log_{\frac{100}{9}} n.$$

putting k in eq (1)

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

18.

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

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

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

19.
```
input a(N, key);
for (i=0 to n-1){
    if (a[i]= key)   return i;
}
return -1;
```

21.

Algorithms	Best case	Avg case	Worst case
• Bubble sort	$O(n^2)$	$O(n^2)$	$O(n^2)$
• Selection sort	$O(n^2)$	$O(n^2)$	$O(n^2)$
• Insertion sort	$O(n^2)$	$O(n^2)$	$O(n^2$
• Merge sort	$O(n\log n)$	$O(n\log n)$	$O(n\log n)$
• Quick sort	$O(n\log n)$	$O(n\log n)$	$O(n^2)$
• Heap sort	$O(n\log n)$	$O(n\log n)$	$O(n\log n)$.

22.

Algo	In-place	stable	online
• Bubble sort	✓	✓	✗
• Selection sort	✓	✗	✗
• Insertion sort	✓	✓	✓
• Merge sort	✗	✓	✗
• Quick sort	✗	✗	✗
• Heap sort.	✓	✗	✗

---

24. $T(n) = T(n/2) + 1$

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

Scanned with CamScanner