Name :- Akash sharma

University Roll No :- 2015488

Class Roll :- 08.

Section :- I.T

Akash Sharma

Assignment :- 1

Ans: 1 Asymptotic notation are Mathematical tools to represent the time complexity of algorithm for asymptotic analysis.

The main idea of asymptotic analysis is to have a Measure of the efficiency of algorithm that don't depends on machine specific constants and doesn't requires to be implemented and time taken by the program to be compared.

Following are the asymptotic notations that are mostly used.

1. Θ Notation :- The theta notation bounds a function from above and below, so it define exact asymptotic behaviour

2. Big O Notation :- It define an upper bounds of an algorithm. it bound a function only from above.

3. Ω Notation :- Ω Notation provides an asymptotic Lower bound

For Example consider Insertion Sort.

Akash Sharma

It takes linear time in best case and quadratic time in worst case.

We can say that Insertion sort have.

$O(n^2)$.

$O(n^2)$ for worst case.

$\Theta(n)$ for best case

$\Omega(n)$

Ans 2 $\Theta(\log n)$

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

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

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

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

$= 3^3 T(n-3$

$\vdots$

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

$\Rightarrow 3^n$

Akanksha

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

$$T(n) = 2T(n-1) - 1$$
$$= 2(2T(n-2) - 1) - 1$$
$$= 2^2(T(n-2)) - 2 - 1$$
$$= 2^2(2T(n-3) - 1) - 2 - 1$$
$$= 2^3 T(n-3) - 2^2 - 2^1 - 2^0$$
$$= 2^n T(n-n) - 2^{n-1} - 2^{n-2} - 2^{n-3} \ldots - n^2 - 2^1 - 2^0.$$
$$= 2^n - (2^n - 1)$$
$$= 2^n - 2^n = 1$$

$$T(n) = 1$$

Ans:5 $s_i = s_{i-1} + i$

If k is total number of iterations taken by the program then while loop terminates.

$$1+2+3 \ldots\ldots k = [k(k+1)/2] \geq n$$
$$\therefore k = O(\sqrt{n})$$

Ans:-6 $O(\sqrt{n})$

Akash Chauhan

Ans:-7 j is loop executing log n times
k is loop executing log n times
i is loop executing n/2 times $n/2 \simeq n$

Time complexity = $O(n\log^2 n)$

Ans:-8 $O(n^3)$

Ans:-9 Inner loop will execute $(n + \frac{n}{2} + \frac{n}{3} + \cdots \frac{n}{n}]$

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

It is equal to $\Theta(n\log n)$

Ans:-10. n^k a^n

$k \geq 1$ $a \geq 1$

Taking $k = a = 2$

n^2 2^n

we can say $n^2 = O(2^k)$

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

Ans:-11 $O(\sqrt{n})$ same logic given in Ques 5

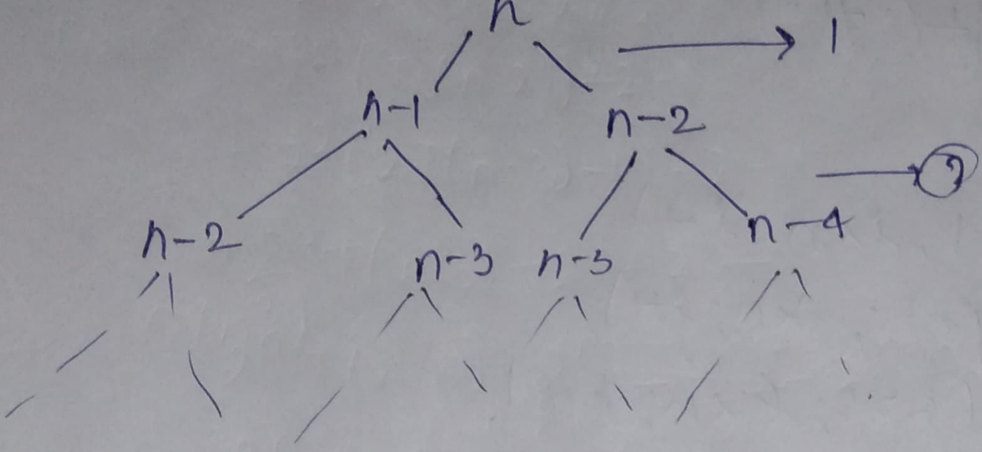
Akash Sharma

5

Ans)-12 Reccurrence Relation

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

Making Reccurrence Tree.



$$T.C = 1 + 2 + 4 + \cdots + 2^n$$

$$a = 1$$

$$r = 2$$

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

$$O(2^{n+1}) = O(2 + 2^n) = O(2^n)$$

space complexity = $O(n)$

This is because maximum stack frame is equal to n only as function is called like this.

$$f(n-1) + f(n-2)$$

$f(n-2)$ is called when we get the return value from $f(n-1)$

∴ It is equal to $O(n)$

Akash Shaw

Ans! 13

n log n

```
for(i=1; i<n; i++).
    for(j=1; j<=n; j=j*2)
        print("#");
```

n^3

```
for(i=1; i<n; i++)
    for(j=1; j<n; j++)
        for(k=1; k<n; k++)
            print("#");
```

log log n

```
int fun(int n)
{
    if (n<=2)
        return 1;
    else
        return (fun(floor(sqrt(n))) + n);
}
```

Akashshaurey

Ans:-14 $T_n = T(n/4) + T(n/2) + Cn^2$

we can assume,

$T(n/2) >= T(n/4)$

$T(n) = 2T(n/2) + Cn^2$

Applying Masters Method,

$a = 2$, $b = 2$

$k = \log_b a = \log_2 2 = 1$

$n^k = n$

$f(n) = n^2$

It is $\Theta(n^2)$

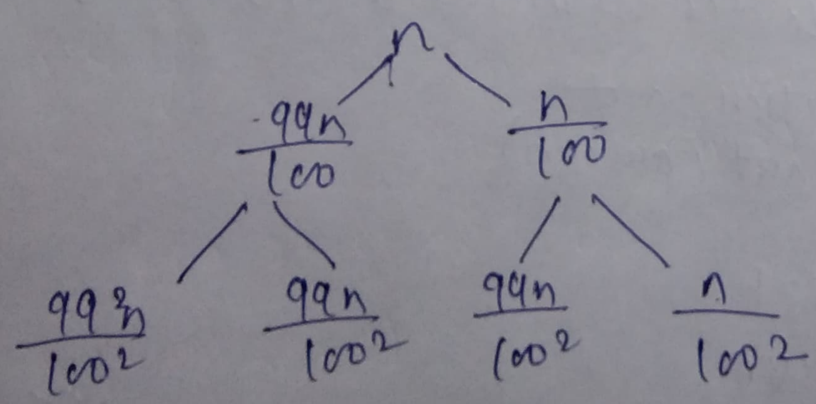
But as $T(n)\ \alpha = \Theta(n^2)$

$T(n) = \Theta(n^2)$

Ans:-16 If k is a constant greater than 1

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

Ans:-17 $T(n) = T\left(\frac{99n}{100}\right) + T\left(\frac{n}{100}\right)$



$T.C = \log_{100/99} \simeq \log n.$

we can say that the base of log does not matter as it only a matter of constant.

Ans:- 18

a) 100 log logn $\sqrt{n}$ nlogn! nlogn n^2 2^n $2^m/4^n$ n!

b) 1 log logn $\sqrt{\log n}$ logn 2logn log2n n 2n 4n logn! nlogn n^2 $2(2^n)$ n!

c) 96 $\log_8 n$ 5n logn! $n\log_6 n$ $n\log_2 n$ $8n^2$ $7n^3$ 8^{2n} n!

Ans:- 19 Linear search (array, key)
for i in array
if value == key
return i;

Ans:- 20 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 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)

Akash Sharma

Pick last element arr[i] and insert it into sorted sequence arr [0 — i-1]

}.

Insertion sort considers one input element per iteration and produces a partial solution without considering future elements.

It is called online sorting Algorithm.

Ans

considering only 3 sorting Algo. till now as we get the lecture get the lecture of these 3 only.

Algo	Best Case	Avg. case	worst case	SC	stable	Inplace	a
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	$O(n)$	$O(n^2)$	$O(n^2)$	$O(1)$	✓	✓	✓

Ans :- 23. Binary search.

A ← sorted Array.

n ← size of Array

x ← value to be sorted

while x not found

Akash Sharma

if upperbound < Lowerbound.
EXIT : X does not exist

Set mid point = lower bound + (upperbound − Lowerbound) / 2

if A[mid point] < X.
Lower bound = mid point + 1

if A[mid point] > X
upperbound = midpoint − 1

if A[mid point] = X
Exit = X found at mid point.

	Time complexity	space complexity
Linear	$O(n)$	$O(1)$
Binary Search (Recursive)	$O(\log n)$	$O(\log n)$
Binary Search (Iterative)	$O(\log n)$	$O(1)$

Ans-24 $T(n) = T(n/2) + C$

Akash Kumar