Assignment-1

Ans 1] Asymptotic notation are matehematical tools to represent the time complexity of algorithms for asymptotic analysis.

The main idea of asymptotic analysis is to have a measure of the efficiency of algorithms that don't depends on machines. Specific constants and doesn't require algorithm 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 defines exact asymtotic behaviour.

2] Big O Notation :- It defines an upper bound of an algorithm, it bounds a function only from above

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

For eg consider Insertion sort

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

We can say that Insertion sort have

$O(n^2)$

$\Theta(n^2)$ for worst case

$\Theta(n)$ for best case

$\Omega(n)$

Ans 2] $\Theta(\log n)$

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

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

$3(3T(n-2)$

$3^2 T(n-2)$

$3^3 T(n-3)$

$\vdots$

$3^n T(n-n)$

$= 3^n$

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

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

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

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

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

$= 2^3T(n-3) - 2^2 - 2^1 - 2^0$

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

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

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

$T(n) = 1$

Ans5] $S_i = S_{i-1} + i$

If k is total number of iterations taken by the program, then while Loop terminates

$1+2+3 - - - - - k = [k(k+1)/2] > n$

$\therefore k = O(\sqrt{n})$

Ans 6] $O(\sqrt{n})$

Ans 7] j is Loop executing $\log n$ times

k " " " $\log n$ times

i " " $n/2$ times $n/2 \simeq n$

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

Ans8] $O(n^3)$

Ans 9/15] Inner Loop will execute $\left(n + \frac{n}{2} + \frac{n}{3} + \cdots \frac{n}{n}\right)$

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

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

Ans 10] · n^k a^n

$k >= 1$ $a > 1$

Taking $k = a = 2$

n^2 2^n

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

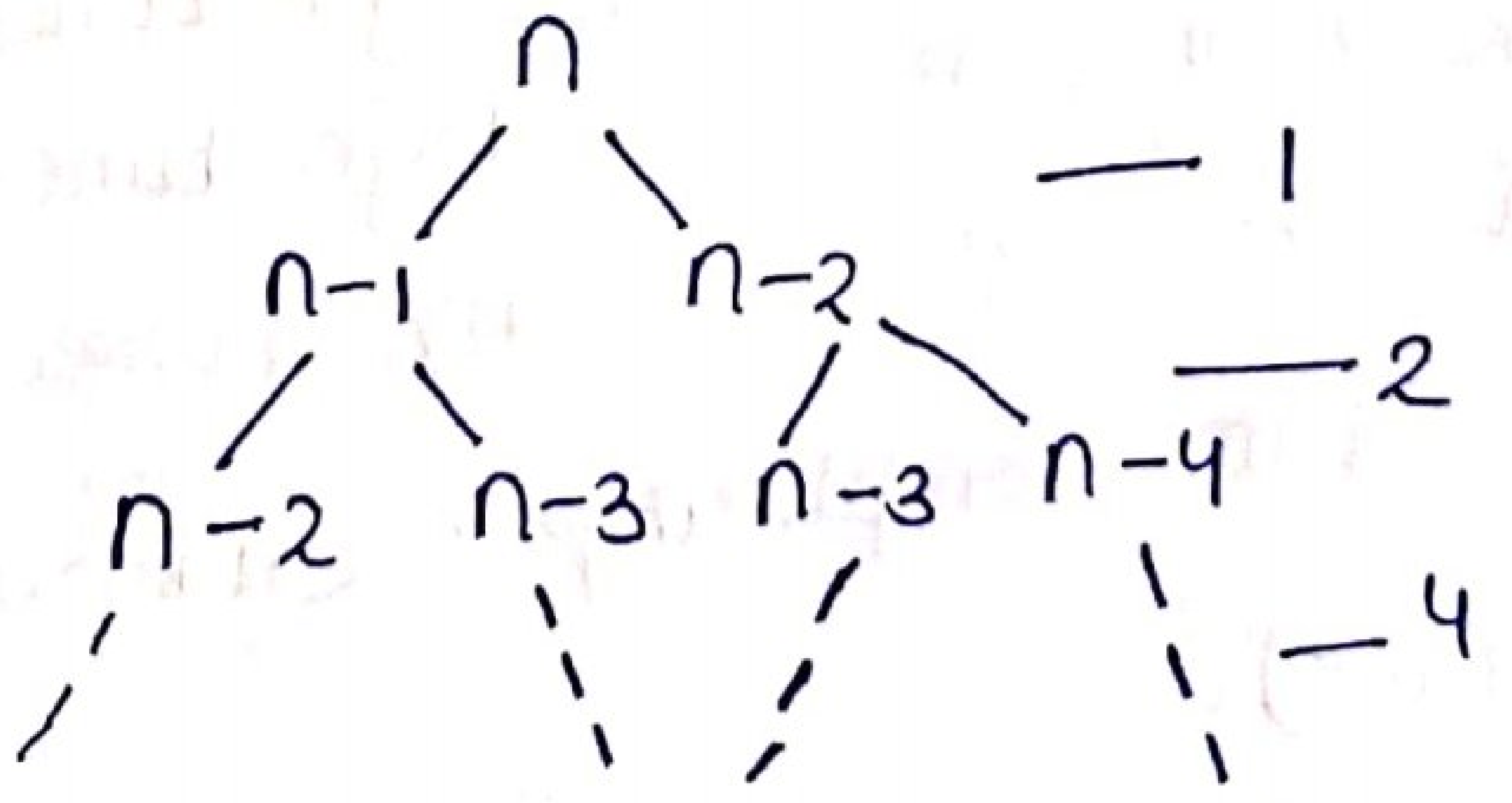
$n^k = O(a^n)$

Ans 11] $O(\sqrt{n})$ Same Logic given in Ques 5

Ans 12] Recurrence Relation

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

Making Recurrence Tree



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

$a = 1$, $r = 2$

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

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

Space complexity $= O(n)$

This is because maximum stack frame is ~~same~~ 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 $f(n-1)$

$\therefore$ It is equal to $O(n)$

Ans 13] n logn

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

n^3

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

Log log n

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

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 $\theta(n^2)$

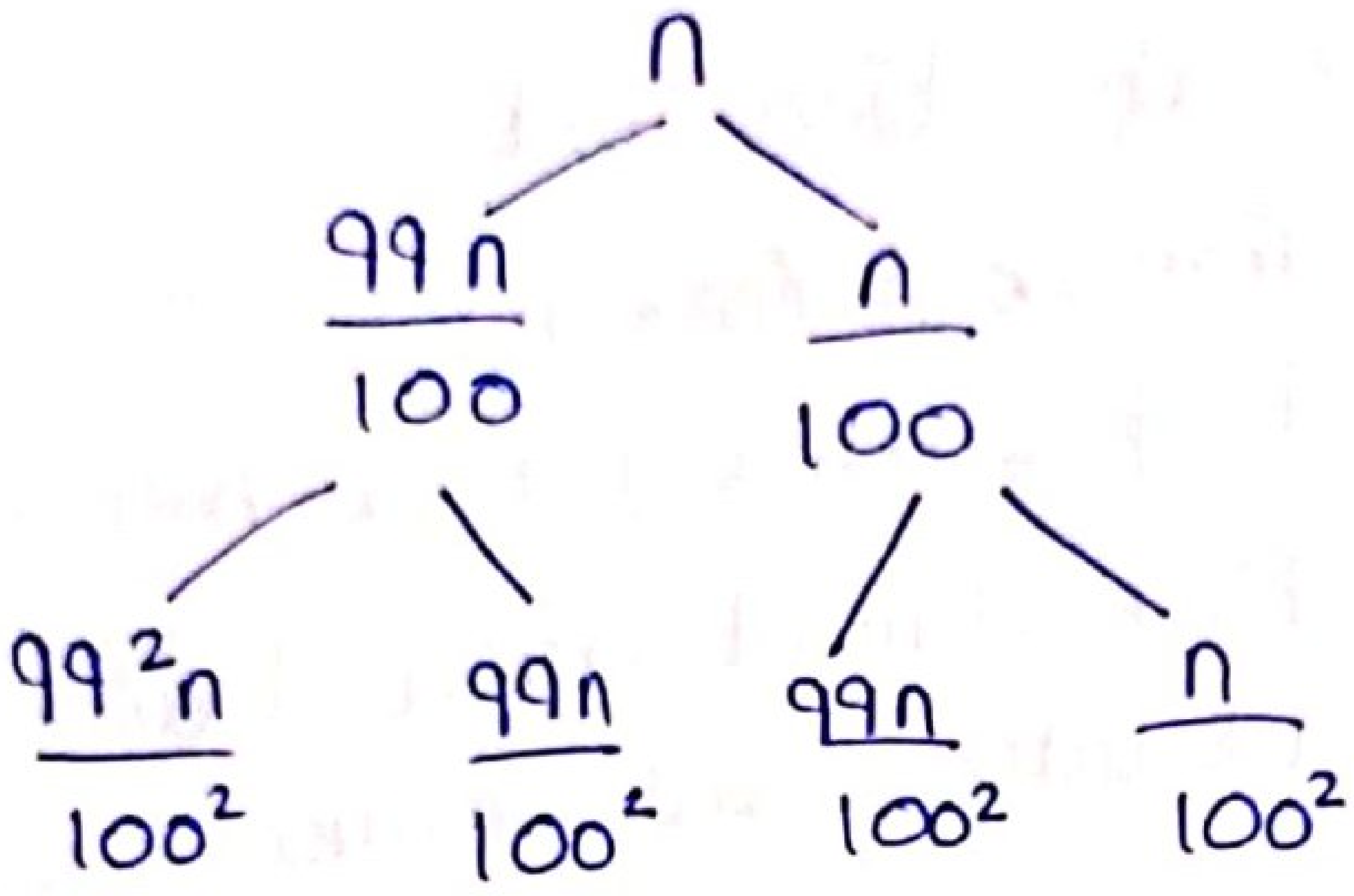
But as $T(n) <= \theta(n^2)$

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

Ans 16] if k is a constant greater than 1

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

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



If we take longer branch i.e $\frac{99n}{100}$

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

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

Ans 18] a) 100 $\log \log n$ $\sqrt{n}$ n $\log n!$ $n \log n$ n^2 2^n $2^{2n}/4^n$ $n!$

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

c] 96 $\log_8 n$ $5n$ $\log n!$ $n \log_6 n$ $n \log_2 n$ $8n^2$ $7n^3$ 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)

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 20/21/22]

Considering only 3 sorting Algo. till now as we get the Lectures of these 3 only.

Algo.	Best case	Avg. case	Worst case	S.C	Stable	Inplace	On
Bubble Sort	$O(n)$	$O(n^2)$	$O(n^2)$	$O(1)$	✓	✓	×
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

if upperbound < lower bound

EXIT : x does not exist

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

if A [mid point] < x

Lowerbound = midpoint + 1

if A [mid point] > x

Upperbound = midpoint − 1

if A [midpoint] = x

EXIT = x found at mid point

	Time Complexity	Space Complexity
Linear	$O(n)$	$O(1)$
Bineary Search (Recursive)	$O(\log n)$	$O(\log n)$
Bineary Search (Iterative)	$O(\log n)$	$O(1)$

Ans 24] $T(n) = T(n/2) + c$

Scanned with CamScanner