Page No.

Assignment no: 1

Date 28/02/24

Ques) What are the different algorithm design principle?

→ There are various algorithmic design principle -

1) Divide and Conquer

→ It is a top-down approach. It divides the original problem into a set of sub problems. Solve every sub-problem individually, recursively. Combine the solution of the sub problems into a solution of the whole original problem.

2) Greedy Technique

→ It is used to solve the optimization problem. An optimization problem is one in which we are given a set of input values, which are required either to be maximized or minimized i.e. some constraints or conditions.

3) Dynamic programming

→ It is a bottom up approach we solve all possible small problems and then

Page No.
Date / /

Combine them to obtain soln for bigger problems.

1) Randomized algorithm

→ A Randomized algorithm is defined as an algorithm that is allowed to access a source of independent, unbiased random bits, and it is then allowed to use these random bits to influence its computation.

5) Back tracking

→ It tries each possibility until they find the right one. It is the depth first search of the set of possible solution. During the search, if an alternative does not work, then backtrack to the choice point, the place which presented different alternatives and tries the next alternative.

Page No.

Date / /

6) Branch and Bound

→ In this, a given sub problem which cannot be bounded has to be divided into atleast two new restricted sub problems.

Branch and Bound algorithm are methods for global optimization in non-convex problems.

Page No.

Date / /

(Ques) Give the Analysis framework of algorithm with example?

→ Analysis of the algorithm is a way of finding the resources required by the algorithm. This helps to decide the best algorithm among the set of Candidate algorithm. Resource may be time, space, power consumption, communication bandwidth, computer hardware, etc.

The component of the efficiency analysis are shown below:-

Space Complexity
Time Complexity
Measuring Input size
Measuring Running Time
order of growth
Cases of Complexity (Best, worst, Average)

1) Space Complexity

→ Space complexity is a very important Notation of efficiency analysis.

Page No.

Date / /

The space complexity of the algorithm is controlled by 2 components -

a) Fixed - Sized components : It includes the programming part whose memory requirement does not alter program execution. for ex)
- Instructions
- Variables
- Constants
- Arrays.

b) Variable Size components : It includes the part of the program which whose size depends on the problem being solved. for ex)
- Size of loop
- Stack required to handle a recursive call
- Dynamic data structures like a linked list.

To indicate the space complexity of the issue with input size n, we use the notation S(n).

q The following examples demonstrate the concept of space complexity.

Example 1) Sum of elements of an array

Page No.

Date / /

Algorithm

SUM_ARRAY_ELEMENTS (A)

// Description : Add all element of array A

// Input : Array A of size n

// Output : Variable sum which holds the addition of array elements.

```
Sum ← 0
for i ← 1 to n do
    sum ← Sum + A[i]
end
return Sum
```

→ The addition of all array elements requires only one extra variable denoted as sum. This is independent of array size. Thus the space complexity of the algorithm is constant and $S(n) = O(1)$.

2] Time Complexity

→ The time complexity of an algorithm often determines its quality. The most essential component of the efficiency analytical framework

Page No.

Date / /

is time complexity.

Example perform addition of two arrays.

Algorithm ADD-ARRAY (A, B)
// Description: perform a element-wise arithmatic add^{n} of two arrays.
// Input: Two number arrays A & B of length n
// output: Array C holding the elements-wise sum of array A and B.

for i ← 1 to n do
// one initialization, n increment, n comparison
C[i] ← A[i] + B[i]
// n addition and n assignment

end

return C

→ T(n) = 1 (initialization) + n (comparison + increment addition + assignment)

= 1 + 4n

→ All multiplication & divide constant should be eliminated. So a result for a given algo-

T(n) = O(n)

Page No.

Date / /

3] Measuring Input Size

→ The algorithm execution time is mostly determined by the size of the input. For longer input, the algorithm runs longer. As a result, the algorithm's complexity is always assessed in terms of input size.

→ The complexity is independent of hardware architecture, available memory CPU speed and so forth.

4] Measuring Running Time

→ The running time of an algorithm is primarily determined by the amount of the inputs and as a result, the number of the frequent operations executed by the algorithm.

→ The exp. time is not measured in terms of real time consumed by this algorithm in seconds or minutes, but rather as a function of a number of primitive |

Page No.

Date / /

frequent operation.

5] Order of growth

→ The relationship b/w input size and performance of the algorithm is called order of growth.

The order of increase reflects how rapidly the time required by the algorithm rises in relation to the size of the input.

→ The algorithm may run a number of steps in the sequence of $\log n$, n_0, n_2, n_3 or anything else for input size n.

6] Cases of Complexity

→ Let $T(n)$ denotes the time necessary to solve a problem of size n.

Eg) Algorithm LINEAR_SEARCH (A, Key)

// Description : search element Key in array A.

// Input : Array of A of size n, and the element to be searched. i.e. Key.

// output : Status : success / failure

Page No.

Date / /

```
flag ← 1
for i+1 to n do
    if A[i] == Key then
        flag ← 1
        break
    end
end
if flag == Key then
    print " Key is found"
end
```

The Best case running time for removing the first node from the list is constant,

$T(n) = O(1)$

→ The number of comparisons is proportional to the size of the problem. As a result, the worst-case running time for this $T(n) = O(n)$ when an element is either at the end or not there at all.

→ Average case analysis is more than just taking the best & worst case and averaging them. The average case is determined by the data distribution.

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

Page No.

Date / /

(Q3) Solve the following using Substitution method.

$$T(n) = 9T\left(\frac{n}{3}\right) + n^2 \qquad n > 1$$

$$= 1 \qquad n = 1$$

→ $T(n) = 9T(n/3) + n^2$ —— (I)

To find $T(n/3)$ put $n = n/3$ in (I)

$$T\left(\frac{n}{3}\right) = 9T\left(\frac{n}{9}\right) + \frac{n^2}{9}$$

put the value of $T(n/3)$ in equⁿ (I)

$$T(n) = 9\left[9T\left(\frac{n}{9}\right) + \frac{n^2}{9}\right] + n^2$$

$$= 9^2 T\left(\frac{n}{9}\right) + n^2 + n^2$$

$$= 9^2 T\left(\frac{n}{3^2}\right) + 2n^2 \quad \text{——} \quad \text{(II)}$$

To find $T(n/9)$ put $n = n/9$

$$T\left(\frac{n}{9}\right) = 9T\left(\frac{n}{27}\right) + \frac{n^2}{81}$$

put this in equⁿ (II)

Page No.

Date / /

$T(n) = 9^2 T(\frac{n}{3}) + n^2$

$T(n) = 9^2 \left[9T\left(\frac{n}{27}\right) + \frac{n^2}{81} \right] + n^2$ $2n^2$

$= 9^3 T\left(\frac{n}{3^3}\right) + \frac{n^2}{9} + 2n^2$

$= 9^3 T\left(\frac{n}{3^3}\right) + 3n^2$ — (III)

from I, II and III

The generalized form is -

$$T(n) = 9^k T\left(\frac{n}{3^k}\right) + kn^2$$

The Terminating condⁿ are -

$\frac{n}{3^k} = 1$, or $n = 3^k$ or $k = \log_3 n$

$T(n) = 9^{\log_3 n} T(1) + \log_3 n \cdot n^2$

$= n^{\log_3 9} \cdot 1 + n^2 \log_3 n$

$= n^2 + n^2 \log_3 n$

Page No.

Date / /

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

(Que 4) Solve by using Substitution method -

$$T(n) = 2T\left(\frac{n}{2}\right) + n(\lg n)^2 \qquad n > 1$$

$$= 1 \qquad n = 1$$

$$\rightarrow T(n) = 2T\left(\frac{n}{2}\right) + n(\lg n)^2 \quad \text{(I)}$$

To find $T(n/2)$ put $n = n/2$

$$T\left(\frac{n}{2}\right) = 2T\left(\frac{n}{4}\right) + \frac{n}{2}\left(\lg \frac{n}{2}\right)^2$$

put in (I)

$$T(n) = 2\left[2T\left(\frac{n}{4}\right) + \frac{n}{2}\left(\lg \frac{n}{2}\right)^2\right] + n(\lg n)^2$$

$$= 2^2 T\left(\frac{n}{4}\right) + n\left(\lg \frac{n}{2}\right)^2 + n(\lg n)^2 \quad \text{(II)}$$

To find $T\left(\frac{n}{4}\right)$ put $n = n/4$ in (I)

$$T(n/4) = 2T\left(\frac{n}{8}\right) + \frac{n}{4}\left(\lg \frac{n}{4}\right)^2$$

put in equn (II)

Page No.

Date / /

$$T(n) = 2^2\left[2T\left(\frac{n}{8}\right) + \frac{n}{4}\left(\lg\frac{n}{4}\right)^2\right] + n\left(\lg\frac{n}{2}\right)^2 + n(\lg n)^2$$

$$T(n) = 2^3 T\left(\frac{n}{2^3}\right) + n\left(\lg\frac{n}{4}\right)^2 + n\left(\lg\frac{n}{2}\right)^2 + n(\lg n).$$

III

From I, II and III, the generalized equn is -

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

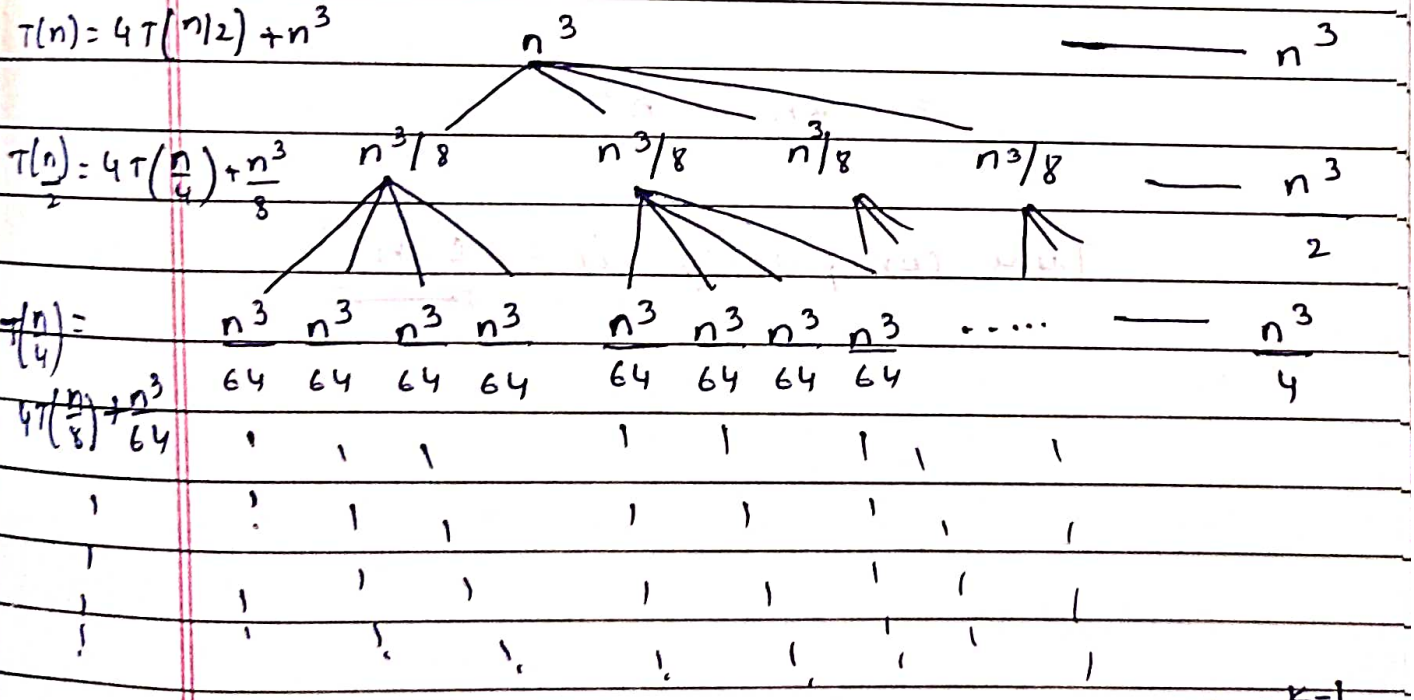
Page No. _____

Date __/__/__

Ques) Solve by using Recursion tree method.

$$T(n) = 4T(n/2) + n^3 \qquad n > 1$$

$$= 1 \qquad n = 1$$



Terminating cond^n are.

$$\frac{n}{2^k} = 1 \quad \text{or} \quad n = 2^k \quad \text{or} \quad k = \log_2 n$$

Page No.

Date / /

Total cost = L.C + I.C

$= 4^k + \left(\frac{1}{2}\right)^{k-1} \cdot n^3$

$= 4^{\log_2 n} + \left[\left(\frac{1}{2}\right)^0 + \left(\frac{1}{2}\right)^1 \cdots\right]$

$= n^{\log_2 4} + n^3 \left[\frac{1}{1-\frac{1}{2}}\right]$

$= n^2 + n^3 [2]$

$= 2n^3 + n^2$

Time complexity is = $O(n^3)$