5

1 Define algorithm. (1)

A → An algorithm is a set of step by step instructions that are designed to solve a specific problem / performs a specific task.

→ It is a well defined computational procedure that takes a set of inputs & produces an output.

2 Explain analysis of algorithms.

→ Algorithms analysis is the process of evaluating & comparing the efficiency & performance of different algorithms with respect to their time & space complexity.

→ It involves analyzing how an algorithm uses computational resources like time & memory as input size grows.

→ Aditional factors such as knowing the best case, worst case & average case analysis helps in understanding the algorithm's performance.

3 Find time complexity of matrix multiplication algorithm.

A.

PTO

```
Matrix_multiplication (a, b)
{
  for i = 1 to n  ──→ n times
  {
    for j = 1 to n  ──→ n × n times
    {  c[i][j] = 0  ──→ n^2
       for k = 1 to n  ──→ n^3 times
           c[i][j] = c[i][j] + a[i][k] * b[k][j]  ──→ n^3
    }
  }
}
```

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

$\therefore T(n) = n^3 \Rightarrow O(n^3)$. $\because$ considering the higher degree terms.

4 What is omega notation?

A → It is denoted by Ω, it is used to represent the lower bound / best case time complexity of an algorithm.

→ It provides ways to express the minimum of running time of an algo as input size approaches infinity.

→ If an algorithm has time complexity of $\Omega(g(n))$ it means there exists a constant (c) & a lower limit function $g(n)$ such that the algorithm takes atleast $c * g(n)$ steps to execute a process.

5 Write algorithm using iterative & recursive functions to find sum of n numbers. (2)

```
Iterative (n)
{
  sum=0 ;
  for i=1 to n do
        sum = sum+ i;
  return sum;
}

Recursive (n)
{
  if (n==1) then
      return 1;
  else
      return n+ Recursive(n-1);
}
```

6 Explain divide & conquer control abstraction.

```
DAndC(P)
{
  if small(P) then return S(P);
  else
  { divide p into smaller instances P1, P2, ..., Pk  k≥1;
     Apply D-AndC to each of subproblems;
     return Combine(D-AndC(P1), DAndC(P2), ..... DAndC(Pk));
  }
}
```

→ Small(P) is a boolean valued functⁿ that determines whether the input size is small that the answer can be computed without splitting.

→ Otherwise the problem P is divided into small subproblems. $(P_1, P_2, \ldots, P_k)$

→ These subproblems are solved by recursive application of DAndC.

→ Combine is a function that determines the solution to P by using solutions to k subproblems.

→ The computing time of DAndC is described by the recurrance relation.

$$T(n) = \begin{cases} g(n) & n \text{ is small.} \\ T(n_1) + T(n_2) + \ldots + T(n_k) & \text{otherwise} \end{cases}$$

7 What is recurrance relation for max-min algorithm. (3)

The recurrance relation for max-min algorithm is.

$$T(n) = \begin{cases} 2T(n/2) + 1 & n > 2 \\ 1 & n = 2 \\ 1 & n = 1 \end{cases}$$

8 When is sorting method stable.

→ A sorting algorithm is considered stable if it maintains the relative order of equal elements in sorted output.

→ In other words, if 2 elements have same value the one that appears first in input sequence, will also be first in output sequence.

Eg:-

→ If we sort the array [3, 1, 4, 1, 5] using merge sort or insertion sort the output's 1's will retain their input & output order.

→ If we sort the same array using quick sort then the 1's do not retain their input & output order.

9 What is graph? What do you mean by traversal in a graph?

→ A graph is a non-linear data structure that consists of a collection of vertices & edges.

→ In a graph, each vertex represents an entity while each edge represents a connection or a relationship between the entities.

Graph traversal:—

→ It refers to the process of visiting all edges & vertices of a graph in a systematic way.

→ Traversal algorithms are used to explore & analyze a graph & are an important part of many graph related applications, such as search algorithm, shortest path algorithm & network analysis.

10 Define optimal & feasible solution.

Optimal solution:—

→ In terms of algorithms, an optimal solution refers to best possible solution to a problem.

→ It is a solution that satisfies all constraints of problem & has the highest/lowest objective value, depending on whether the problem is a maximization/minimization problem.

Feasible solution:-

→ A feasible solution, on other hand refers to a solution that satisfies all the constraints of problem but may not necessarily be the best possible solution.

→ These are important because they provide a starting point for finding optimal solution.

" Write an algorithm for solving towers of Hanoi problem & compute the time complexity of algorithm.

```
Towers of Hanoi (n, x, y, z)
{ if (n≥1) then
  { Towers of Hanoi (n-1, x, z, y);
    write ("Move top disk from Tower ", x, " to top of tower ", y);
    Towers of Hanoi (n-1, z, y, x);
  }
}
```

→ Moving n-1 disks from source to auxillary peg takes (n-1) steps.

→ Moving nth disk from source to destination peg requires 1 step.

→ Moving n-1 disks from auxiliary to destination peg requires n-1 steps.

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

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

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

$= 2^2[2T(n-3)+1] + 2+1$

$= 2^3[2T(n-4)+1] + 2^2+2+1$

$\vdots$ $(k+1)$ times.

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

if $n-k=0$
$n=k$

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

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

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

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

12 Explain detecting a counter-feit coin from a set of $n(n>1)$ coins using divide & conquer methodology. How many weight comparisions are done?

→ Divide & conquer is a well known algorithmic strategy that can be used to detect a counterfiet coin from a set of n coins.

→ The basic idea is to split the set of coins into smaller subsets, & then recursively apply the same process to each subset untill the counterfiet coin is found.

Algorithm:-

→ Divide the set of n coins into two equal sized subsets.

→ Weigh 2 subsets on a scale, if they balance then counterfeit is must be in remaining set of coins.

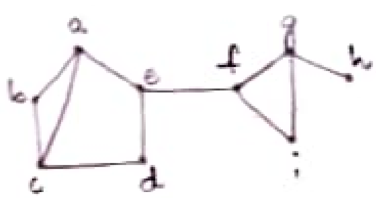
→ If they don't balance then the counterfeit coin must be in one of two weighted subsets.

→ Repeat the above sets recursively on subset that contains the counter-feit coin untill the counterfeit coin is found.

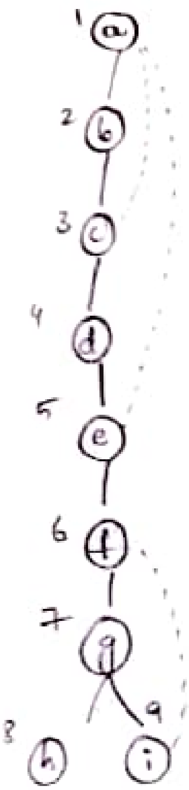
→ The no of weight comparisions required to find counterfeit coin using this algorithm is $\log_2 n$, as at each step, the size of set of coins being weighed is reduced by a factor of 2.

→ Therefore, the no. of weight comparisions required is proportional to the logarithm.

13 Find articulation point for the graph using the algorithm dfn().



→ Construct the depth first spanning tree of graph & identify the depth first number.



vertex	a	b	c	d	e	f	g	h	i
dfn	1	2	3	4	5	6	7	8	9

$L(u) = \min\{ dfn[u], \min\{ L[w] / w \text{ is a child of } u \},$
$\min\{ dfn[w] / (u, w) \text{ is a back edge} \}\}.$

$L(h) = \min\{ dfn[h] \} = \min\{8\} = 8.$

$L[i] = \min\{ dfn[i], \min\{ dfn[f] \} \}$

$= \min\{ 9, \min\{6\} \}$

$= \min\{9, 6\}$

$L[i] = 6.$

$L[g] = \min\{ dfn[g], \min\{ L[h], L[i] \} \}$

$= \min\{ 7, \min\{8, 6\} \}$

$L(g) = 6$

$L(f) = \min\{ dfn[f], \min\{ L[g] \}, \min\{ dfn(i) \} \}$

$= \min\{ 6, 6, 9 \}$

$L(f) = 6.$

$L(e) = \min\{ dfn[e], \min\{L[f]\}, \min\{dfn[a]\}\}$

$= \min\{5, 6, 1\}$

$L(e) = 1$

$L(d) = \min\{ dfn[d], \min\{L[e]\}, \min\{dfn[a]\}\}$

$= \min\{4, 1, 1\}$

$L(d) = 1$

$L(c) = \min\{ dfn[c], \min\{L[d]\}, \min\{dfn[a]\}\}$

$= \min\{3, 1, 1\}$

$L(c) = 1$

$L(b) = \min\{ dfn[b], \min\{L[c]\}, \min\{dfn[a]\}\}$

$= \min\{2, 1, 1\}$

$L(b) = 1$

$L(a) = \min\{ dfn[a], \min\{L[b]\}, \min\{dfn[c]\}\}$

$= \min\{1, 1, 3\}$

$L(a) = 1.$

→ vertex 'e' is an articulation point as child 'f' has $L[f] = 6$ & $dfn[e] = 5$.

→ vertex 'f' is an articulation point as child 'g' has $L[g] = 6$ & $dfn[f] = 6$

→ vertex 'g' is an articulation point as child 'i' has $L[i] = 8$ & $dfn[g] = 7$

∴ To make a graph biconnected construct edges.

(a, f), (a, g) & (i, h)

14 Explain how knapsack problem is solved using greedy method.

→ The knapsack problem is a well known optimization problem in computer science & mathematics that asks how to pack items of different weights & values into it, of a given capacity to maximize the total value of items.

→ The basic idea of greedy approach to solve knapsack problem is to sort items in descending order of their value to weight ratio & then iteratively add the items to knapsack.

Algorithm:-

→ Sort the items in decreasing order of their value-to-weight ratio.

→ Initialize the knapsack with zero weight & zero value.

→ For each item in sorted list:

a] If adding the item to the knapsack does not exceed it's weight limit, add the item to the knapsack & update the total weight & value of knapsack.

b] Else skip the item & move on to the next one.

→ Return the total value of the knapsack.

* The intuition behind this approach is that by selecting items with the highest value-to-weight ratio first, we are maximizing the value per unit of weight, which is the most efficient use of the knapsack capacity.

15 Show that the solution of

a) $T(n) = T(n-1) + n$ is $O(n^2)$

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

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

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

$\vdots$

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

$n = k$. [Let].

$= T(0) + n-(n-1) + n-(n-2) + n-(n-3) + \dots + n-(n-n)$

$= T(0) + \cancel{n} - \cancel{n}+1 + \cancel{n} - \cancel{n}+2 + \cancel{n} - \cancel{n}+3 + \dots + \cancel{n} - \cancel{n}+n.$

$= T(0) + (1 + 2 + 3 + \dots + n)$

$= T(0) + \left(\frac{n(n+1)}{2}\right)$

$= 0 + n^2 + n$

$\therefore T(n) = O(n^2)$ [$\because$ considering the higher degree term]