Design and Analysis of Algorithms

Lab Report

Autumn 2024

Internal Assessment 4

Name: Ahmad Shadan Taiyabi

Roll no. : R2142230899

Sap_ID : 500121810

Program, Semester, Batch : B. Tech. CSE, Sem III, Batch 27

Date of experiment: 04/11/2024

School of Computer Science,
University of Petroleum and Energy Studies,
Dehradun

Questions from 1 to 28:

Q1) Write an algorithm for the addition of two numbers. Calculate the time complexity as well.

Q2) Find a pair of two numbers (X, Y) in an array A of N numbers, whose sum is equal to Z. Write the algorithm for the same and find the time complexity as well.

Q3) Find the time complexity:

```
for(i = 0; i < n; i++)
{
 for(j = 1; j < n; j*=2)
{
 statement;
}}
```

Q4) Find the time complexity:

```
for(i = 0; i*i < n; i++)
{
 statement;
 }
```

Q5) Write down the analysis of Strassen's multiplication method.

Q6) Write down the analysis for quick sort.

Q7) Fill in algo.

```
int pow(int x, int n)
{      ----------;
   If(n==0)
    ----------;
   ----------;
If(n%2==0)
    ----------;
else
   ----------;
}
```

Q8) Given n = 5 objects and a knapsack capacity W = 100, find the profits using 3 greedy approaches.

Q9) What is the activity selection problem?

Q10) Provide the main idea behind the greedy approach to solving the activity selection problem.

Q11) Write down the complexities of the activity selection problem in steps.

Q12) Message: DESIGNANDANALYSISOFALGORITM.

Write down the Huffman code for this message.

Q13) Consider a knapsack of size 6. Given w = {2, 3, 4} and p = {1, 2, 5}, calculate the maximum profit that can be collected using the 0/1 Knapsack problem.

Q14) What is memoization?

Q15) What is the bottom-up approach? How is it different from the top-down approach?

Q16) What is relaxation with reference to Dijkstra's algorithm?

Q17) Write 8 properties of asymptotic notations?

Q18) Write pseudocode for maximum subarray problem

Q19) Write pseudocode for interval partitioning problem

Q 20) Solve the recurrence relation below using forward substitution.

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

Q21) Write 5 differences between divide and conquer and greedy algorithmic approach.

Q22) Write 5 differences between greedy and dynamic programming algorithmic approach.

Q23) Describe the Floyd Warshall Algorithm and its purpose.

Q24) What algorithmic strategy is used in Floyd Warshall Algorithm?

Q25) Demonstrate the application of the Floyd Warshall algo using an example. Clearly state i/p and o/p.

Q26) What are the complexities of Floyd Warshall Algorithm?

Q27) What are the different classes of problems?

Q28) Describe the following with examples: P, NP, NP hard, NP complete.

Date / /

→ Do not write outside margins.

→ Complete Q15 - Q17

Date / /

Assignment - 1

Q.1 Write a pseudo code for addition of two numbers Compute the complexity of the algorithm

Pseudo code -

1. Input two numbers a and b
2. Add a and b and store it in c → a+b
3. Display c

Complexity : O(1)

Q.2 Find a pair of two numbers (X, Y) in an Array of N numbers, whose sum is equal to Z. Find the time complexity

1. Input size of array N and Z as sum
2. Declare an array of size N
3. Input the numbers inside the array (arr)
4. Run a for loop for the first element of the array (i) for (i=0; i<N-1; i++)

Date / /

5. Run another for loop for the adjacent element of i. for(j = (i+1), j<N; j++)
6. Check if arr[i] + arr[j] == Z
7. If step 6 is true print X and Y
8. If step 6 is false repeat the loop incrementing the values of i and j according to the loop,
9. If the sum (Z) is not found print "not found".

Time Complexity : $O(n^2)$

Q.3 Find the Time complexity of the following code snippets.

```
for (i=0 ; i<n ; i++)
  {
    for (j = 1 ; j <n ; j = j*2)
      {
        Statement ;   (n log n)
      }
```

Date / /

O(n) i	O(log n) j	Iterations
0	1, 2, 4	3
1	1, 2, 4	3
2	1, 2, 4	3
3	1, 2, 4	3
4	1, 2, 4	3
5	1, 2, 4	3
6	1, 2, 4	3
7	1, 2, 4	3

Time Complexity Combined : $O(n) \times O(\log n) \times O(n \log n)$

$= O(n^2 \cdot \log^2 n)$ (Ans)

Q.4
```
for(i=0; i*i<n; i++) {
    Stmt;
}
```

i	0 i*i<n	
0	$0^2 < n$	T
1	$1^2 < n$	T
2	$2^2 < n$	T
3	$3^2 < n$	T
$\sqrt{n} - 1$	$((\sqrt{n})-1)^2 < n$	T
$\sqrt{n}$	$((\sqrt{n})^2) < n$	F

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

Date / /

Q.5 Write the analysis of Strassen's matrix multiplication method.

Strassen's matrix multiplication method is an algorithm that improves the computional efficiency of multiplying two square matrices.
Strassen's method optimizes matrix multiplication by reducing the number of required multiplications

→ Dividing each matrix into four smaller submatrices of size $(n/2) \times (n/2)$

→ Recursively calculating seven products. (instead of the eight required in the traditional method).

→ Combining these products using matrix addition and subtraction to form the resultant matrix.

$$A = \begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix}, \quad B = \begin{bmatrix} B_{11} & B_{12} \\ B_{21} & B_{22} \end{bmatrix}$$

Date / /

$M_1 = (A_{11} + A_{22}) \cdot (B_{11} + B_{22})$
$M_2 = (A_{21} + A_{22}) \cdot B_{11}$
$M_3 = A_{11} \cdot (B_{21} - B_{22})$
$M_4 = A_{22} \cdot (B_{21} - B_{11})$
$M_5 = (A_{11} + A_{12}) \cdot B_{22}$
$M_6 = (A_{21} - A_{11}) \cdot (B_{11} + B_{12})$
$M_7 = (A_{12} - A_{22}) \cdot (B_{21} + B_{22})$

It reduces the number of multiplication from 8 to 7.

Using these intermediate results, the submatrices of C are computed as:

$C_{11} = M_{11} + M_4 - M_5 + M_7$
$C_{12} = M_3 + M_5$
$C_{21} = M_2 + M_4$
$C_{22} = M_1 - M_2 + M_3 + M_6$

Recursive Relation : $T(n) = O(n^{\log_6(a)})$
$= O(n^{\log_2(7)})$

Master Theorem.

$\log_2(7) \approx 2.81$
$T(n) = O(n^{2.81})$

Date / /

Advantages:

→ Reduced number of multiplications, leading to lower time complexity.
→ More efficient for large matrices.

Disadvantages

→ Increased number of additions and subtractions, which can introduce overhead.
→ May not be efficient for small matrices due to the constant factors involved.

Q.6 Write the analysis of quick sort algorithm.

Ans. 6 Quick Sort is a widely used sorting algorithm based on the divide-and-conquer approach. It works by selecting a "pivot" element from the array and dividing the other elements into two sub-arrays according to whether they are less than or greater than the pivot.

Date / /

Time complexity :-

* Best Case: The best-case time complexity of quick sort occurs when the pivot element divides the array into two nearly equal halves.

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

Solving this recurrence relation gives:

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

* Space Complexity: $O(\log n)$

Recurrence Relation:

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

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

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

Date / /

Q.7 Write the ~~algorithm~~ code for powering of a number

Soln

```
int power (int n, int N)
    if (N == 0)
        return 1;
else
    return divide (n, N)

int divide (n, N)
    if (n == 1)
        return n
else if (n % 2 == 0)
    y = divide (n, N/2)
    return y x y
else y = divide (n, (N-1)/2)
    return (y * y x n)
```

Date / /

Q6 Given n = 5 objects and a knapsack capacity W = 100, find the profit using 3 diff + greedy choices

w = 10, 20, 30, 40, 50
V = 20, 30, 66, 40, 60
v/w = 2 3/2 2.2 1.5

Sol^n
max vi = { 0 0 1 20/40 1 } 146
min wi = { 1 1 1 1 0 } 156
max vi/wi = { 1 1 1 0 40/50 } 164

(1) Maximum Profit

1	2	3	4	5
0	0	30	20	50
→ 0	0	1	20/40	1

profit

1	2	3	4	5
0	0	66	20	60

= 146

(2) Min weight

1	2	3	4	5
1	1	1	1	0

profit

1	2	3	4	5
20	30	66	40	0

= 156

(3) max vi/wi

weight	1	2	3	4	5
	1	1	1	0	40/50

Profit

1	2	3	4	5	&
20	30	66	0	40/50 × 66	
				48	

= 116 + 48

Max profit from max vi/wi = 164

Date / /

Q.9 Providing the main idea behind the greedy approach to solving the activity selection problem. (marks)

Ans The main idea behind the greedy approach to solving the activity selection problem is to select the max no. of activities that don't overlap in time.

Pseudocode:

1) Activity Selection (S, F, N)
   Take input from user.
2) Sort activities based on their finish times in ascending order.

```
for i = 1 to n-1;
    if F(i) < F(i+1)
        swap (F[i], F[i+1])
        swap (S[i], S[i+1])
```

3) Select the first activity & print it's index

```
i = 0
print activity
```

4)

```
for j = 1 to n-1;
    if S[j] >= F[i]
        print activity
        i = j
```

Date / /

5) END

Q.10. What is the activity selection problem? (marks)

Ans. The activity selection problem is a classic optimization problem that involves selecting the maximum number of non-overlapping activities from a given list, where each activity has a start time, and a finish time. The goal is to find the largest subset of activities that can be completed without any overlap in their scheduled times.

Q.11 Write the complexity of the greedy algorithm to solve the activity selection program. (marks)

Ans. Time Complexity:

* Sorting: The activity must be first sorted by their finish times, which takes O(n log n), where n is the no. of activities.

* Selection process: after sorting, the algorithm iterates through the list of activities to select compatible ones which takes O(n)

Date / /

* Overall time complexity : O(n log n)
* Space Complexity : O(n)

Q.12 Write eight properties of asymtotic notation.

Ans. Asymtotic notation is used in computer science and mathematics to describle the behaviour of algorithm as the input size ignoring the constant factors & lower order terms.

1) Focus on growth rates: an algorithm time & space complexity grows with input size.
2) Upper bound (Big O) : The upper bound of an algo's growth rate providing worst case scenario.
3) Lower bound (Omega): the lower bound of an algo's growth rate providing best case scenario.
4) Tight Bound (Theta) : provides both upper & lower bound, giving a tight description of the algo's growth rate.

Date / /

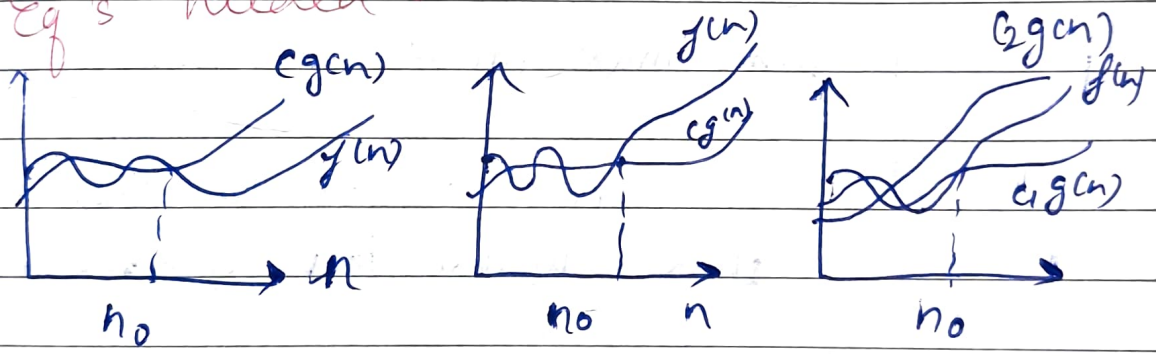
5) Simplification: Simplifies the analysis by disregarding constants and non-dominant terms, focusing only on the most significant factors.

6) Relative Comparision: It allows comparision of algorithm by comparing their growth rate, especially for large input sizes.

7) Platform Independence: Ignores machine specific constants.

8) Input size sensitivity: Asymptotic notations is sensitive to size of the input meaning that its used to analyse how performance changes.

Eq^n's needed.



Big Oh	Big Omega	Theta
Eqn $f(n) = O(g(n))$	$f(n) = \Omega(g(n))$	$f(n) = \Theta(g(n))$
$f(n) \leq c * g(n)$	$f(n) \geq c * g(n)$	$c_1 * g(n) \leq f(n) \leq c_2 * g(n)$

Date / /

Q.18 Write the pseudocode for maximum subarray problem. (10 marks)

1. Initialize
   Set max_current to the first element of the array arr [0]

   Set max_global to the first element of the array arr [0]

2. For each element from second element to the last element.
3. Update max_current to the maximum value of arr [i]
4. Perform max_current + arr [i];
5. If max_current is greater than max_global then update max_global to max_current.

Date / /

6. return max_global which will contain maximum subarray.

Q.19 Write the pseudocode for interval partitioning problem. (10 marks)

1. Let R be the set of all request
2. Let d = 0 be the number of resources.
3. while R is not empty
4. choose a request where i belongs to R that has the earliest start time
5. If 'i' can be assigned to some resources k <= d
6. then assign i to request k, else
7. allocate a new resource d+1
8. assign request i to resource d+1
9. d = d+1
10. then return d.

Date / /

Q. 20 Solve the recurrence relation below using forward substitution (15 marks)

$T(n) = T(n+1) + n, \quad n > 1$

$T(n) = 1$ when $n = 1$

$T(n) = T(n-1) + n$ —— (1)

$T(n-1) = T(n-2) + n-1$ —— (2)

Substitute (2) in (1)

$T(n) = T(n-2) + n-1 + n$ —— (3)

$T(n-2) = T(n-3) + n-2$ —— (4)

Substitute (4) in (3)

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

$T(n) = T(n-k) + n-k+1 + n-k+2 + n-k+3 + n-k+4 - \cdots$

$\cdots - n$

Put $n-k = 1$

$n = k$

Date / /

$$T(n) = T(1) + 1 + 2 + 3 + \cdots\cdots n$$

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

$$= \frac{n^2+n}{2}$$

$$= n^2$$

$$T(n) = O(n^2) \quad \text{(Ans)}$$

Q.13 Consider the knapsack of size 6. Given $W = \{2, 3, 4\}$ and $P = \{1, 2, 5\}$. Find the maximum profit that you can collect following the 0/1 knapsack

Capacity	0	1	2	3	4	5	6
No. of item	0	0	0	0	0	0	0
Item (W=2, P=1)	0	0	1	1	1	1	1
W=3, P=2	0	0	1	2	2	3	3
	0	0	1	2	5	5	6

∴ Max Profit = 6

Date / /

Q.14 What is memoization.

→ Memoization is a technique used to optimize algorithms by storing the results of expensive function calls and reusing them when the same inputs occurs again. It is particularly useful in recursive algorithms where the same subproblems are solved multiple times.

Q.15 What is bottom-up technique? How it is different from a top-down approach

Ans.15 Bottom up technique involves

→ Involves solving subproblems iteratively from smallest to largest (bottom to up)

→ Uses an array to store results of subproblem

Date / /

Different from top-down as

→ Top-down uses the main problem & works down the base case

→ Suitable for cases where not subbranches are needed

→ Uses recursion to break the main problem smaller subproblem

Q.16 What is relaxation with respect to Dijkstra's Algorithm)

Ans: Relaxation is the process of updating the shortest known distance to a vertex when a shorter path is found.

→ For each edge (u, v) if the shortest path distance to v can be reduced by taking path through u, the updated the distance to v

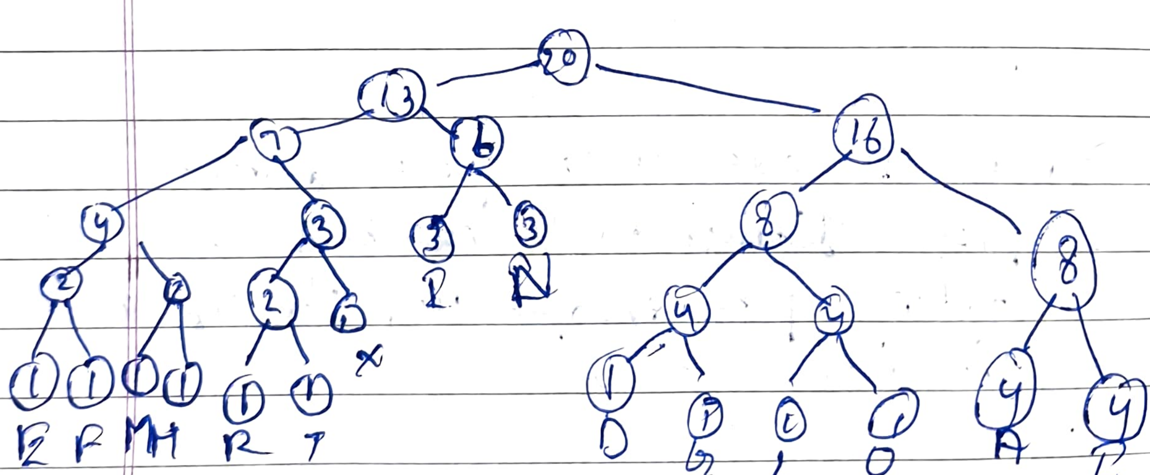
→ Ensures shortest path from each source to each vertex is found.

Date / /

Q12 DESIGN AND ANALYSIS OF ALGORITHMS

Char	Count	Code	Bitcount	Modified Bit
A	4	110	4x3=12	20
D	2	1000	8	10
E	1	0000	5	5
F	1	0000	5	5
G	2	00010	8	10
H	1	010	5	5
I	3	1010	9	15
L	2	00011	8	10
M	1	011	5	10
N	3	1011	9	15
O	2	00100	8	10
R	1	1111	5	5
S	4	00101	12	20
T	1	00100	5	5
X	1	0011	5	5

Total bitcount = 106 Total new bitcount = 106



Date / /

Q.21 Give 5 difference between divide and conquer & greedy approach ? (5marks)

Ans)

	Divide & Conquer	Greedy Approach
1.	Divides the problem into smaller subproblem.	Makes a series of choice selecting best of each step
2.	Requires optimal soln for each subproblem	Assume locally optimal choice lead to a global solution
3.	Combine solution of subproblems to build main solution.	Each choice is made independent without combining results.
→	Used for problems with overlapping subproblems.	Used for problems with a greedy choice problem.
→	Eg. Merge sort, Quick sort, Binary Search	Eg. Prims Algorithm, Kruskal Algo, Dijkstra Algorithm

Date / /

Q.22 Write 5 differences b/w greedy and dynamic programming approach (5 marks) ?

	Greedy	Dynamic
Ans.22	Makes a locally optimal choice without backtracking.	Breaks down the problem into overlapping.
2.	Assumes that locally optimal choices lead to a global optimum.	Uses optimal solution of subproblems that build the main solution.
3.	Makes decision one and doesn't recons them	Makes decision at every stage.
4.	Less efficient	More efficient
5.	eg. Fractional knapsack	eg. 0/1 knapsack problem

Date / /

Q.23 Describe the Floyd - Warshall algorithm and its purpose (5 marks) — 200 marks max.

Ans.23 The Floyd-Warshall algorithm is a fundamental dynamic programming algorithm used for finding the shortest paths between all pairs of vertices in a weighted graph. Unlike other shortest-path algorithms like Dijkstra's which only computes paths from a single source to all destinations, Floyd-Warshall computes the shortest path between every pair of nodes. This makes it suitable for dense graphs and widely used in network routing, optimization, and graph analysis applications.

The algorithm operates by iteratively improving estimates of the shortest path between two nodes through an intermediate node, progressively considering all possible intermidiate nodes. For each pair of nodes (i, j), if there is a path from i to j through another node k,

Date / /

The algorithm checks if the distance $i \to j$ can be reduced by by first traveling from $i \to k$ and then from $k \to j$.
The core of the algorithm is a triple nested loop, each iterating over all nodes, giving it a time complexity of $O(n^3)$

One significant advantage of Floyd-Warshall is that it can handle graphs with negative weights, provided there are no negative cycles. The algorithm provides a comprehensive solution for computing shortest paths, making it valuable in network optimization and routing contexts.

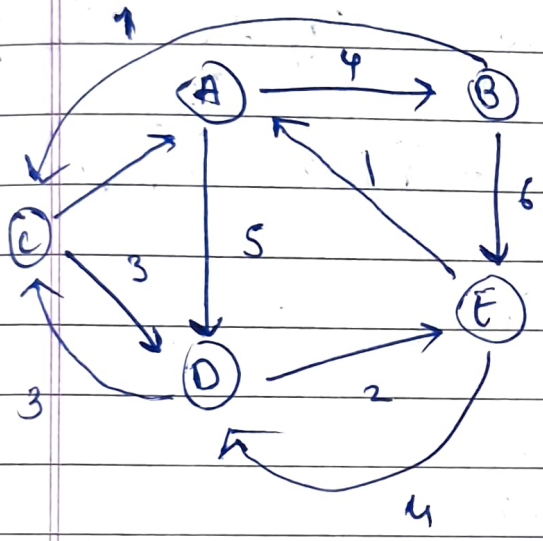
Q.24 What algorithm strategy is used in the Floyd Warshall algorithm (1 mark)

The Floyd-Warshall algorithm uses a dynamic programming strategy, building up shortest paths iteratively considering all

Date / /

intermediate nodes to optimize distances between all pairs of nodes.

Q.25 Demonstrate the application of the Floyd Warshall algorithm using an example at the end clearly state the i/p, o/p (10 marks).



Step 1: Initialize the distance [i][j] matrix using the i/p graph such that the distance [i][j] = wt. of edge from i to j, also distance [i][j] = ∞ if there is no edge from i to j.

	A	B	C	D	E
A	0	4	∞	5	∞
B	∞	0	1	∞	6
C	2	∞	0	3	∞
D	∞	∞	1	0	2
E	1	∞	∞	4	0

Date / /

Step 2: Treat node D as an intermediate node and calculate the distance [i][j] for every i, j node pair using the formula : Distance [i][j] = minimum

Distance [i][j] = (Distance from i to j) + Distance from A to j))

Distance [i][j] = minimum [Distance [i][j], Distance [i][A] + Distance [A][j]]

	A	B	C	D	E
A	0	4	∞	5	∞
B	∞	0	1	∞	6
C	2	6	0	3	12
D	∞	∞	1	0	2
E	1	5	∞	4	0

Date / /

Step 3: Treat node B as an intermediate node & calculate the dist[][] for every {i, j} node pair using the formula:

Distance [i][j] = minimum Distance[i][j]
Distance from (i to B)
+ Distance from B to j)

∴ Distance[i][j] = minimum (
(Distance[i][j],
Distance[i][B] +
Distance[B][j])

	A	B	C	D	E
A	0	4	5	5	16
B	∞	0	1	∞	∞
C	2	6	0	3	12
D	∞	∞	1	0	2
E	1	5	6	4	0

Date / /

Step 4: Treat node i as an intermediate node of calculate the distance for every {i, j} node pair using formula.

Distance [i][j], minimum ([Distance [i][j], Distance from i to i) + Distance from (i to j))

Distance [i][j] = min (dist [i][j], dist [i][i] + dist [i][j])

	A	B	C	D	E
A	0	4	5	5	10
B	3	0	1	4	6
C	2	6	0	3	12
D	3	7	1	0	2
E	1	5	6	4	0

Date / /

Step 5: Treate node E as an intermediate node.

Distance[i][j] = min(distance[i][j], distance[i][E] + distance[E][j])

	A	B	C	D	E
A	0	4	5	5	7
B	3	0	1	4	6
C	2	6	0	3	5
D	3	7	1	0	2
E	1	5	5	4	0

Step 6: Since all the nodes have been visited as intermediate nodes, We can now return updated dist[][] matrix as answer.

	A	B	C	D	E
A	0	4	5	5	7
B	3	0	1	4	6
C	2	6	0	3	5
D	3	7	1	0	2
E	1	5	5	4	0

Date / /

Q.26 What are the complexities of Floyd Warshall Algo

Time Complexity is $O(n^3)$ and
Space Complexity is $O(n^2)$

Q.27 What are the different classes of problem (1 mark)

There are four types of problem classes.

1. P
2. NP
3. NP complete (Non deterministic polynomial Time)
4. NP Hard
5. Co-NP
6. PSPACE
7. EXPTIME (Exponential Time)

Date / /

Q.28 Describe the different classes of the following problem:

(a) P Class
(b) NP
(c) NP hard
(d) NP complete

Ans: Pclass (Polynomical Time) → A problem is in P iff it can be solved in polynomical time, meaning the algorithm's running time increases at a polynomial rate of input. Ex → sorting algo like merge sort.

NP (Non Deterministic Polynomical Time):
→ A problem is in NP if given are condition. Problems that can be solved by any non-deterministic machine in polynomical time.

NP Hard: Solving it is atleast as hard as any problem in NP but it might not be in NP. Eg. Travelling Salesman Problem.

Date / /

NP complete : If a problem is both NP & NP-hard then these are the hardest problems in NP, and if one is solved in polynomical time, all NP problem can be solved in polynomial time.