GOPI
Page No.:
Date : / /

D.A.A

Assignment - 1

1) Define following terms:

1) Algorithm
An algorithm is step-by-step procedure or a set of well defined instructions designed to perform a specific task or solve particular problem

2) Relation
a relation is a set of ordered pairs that shows a relationship between elements of two sets

ex: If A & B are two sets, a relation from A to B is a subset of A x B

Let A = {1, 2}, B = {x, y} Relation = {(1, x), (2, y)}

3) Time complexity
It is a measure of the amount of time an algorithm takes to run as a function of the size of its input (n). It helps in evaluating the efficiency of an algorithm

4) Space complexity
It is a measure of amount of memory an algorithm uses with respect to the input size n. It includes both input data & auxiliary (extra) space used

GOPI
Page No.:
Date : / /

2] explain various properties of an algorithm

Ans :

1] Input
Algorithm may take 0 or more input arguments depending on problem input may be vector, array, tree, graph or some other data structure

2] Output
algorithm produces atleast one output

3] de-finiteness
All instructions in algorithm should be unambiguous Instruction should be simple to interprit. There should not be multiple ways to interprit instruction

4] finiteness
every algorithm must terminate after finite number of steps

5] effectiveness
The instruction which are used to execute task must be basic. So human can trace the instruction by using paper & pencil everywhere

GOPI
Page No.:
Date : / /

3) What is matrix ? explain the various operations which can be performed on the matrix

Ans:

→ A matrix is a rectangular arrangement of numbers, symbols or expressions organized in rows & columns

→ It is a two-dimensional data structure used in mathematics, computer science etc

If a matrix has m rows and n colums, it is called an m x n matrix

$$A = \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \cdots & \vdots \\ a_{m1} & a_{m2} & \cdots & a_{mn} \end{bmatrix}$$

a_{ij} represents the element in the i^{th} - row and j^{th} column

(1) Matrix addition

matrices can be added only if they have the same dimensions

$$A = \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}, \quad B = \begin{bmatrix} 5 & 6 \\ 7 & 8 \end{bmatrix}$$

$$A + B = \begin{bmatrix} 6 & 8 \\ 10 & 12 \end{bmatrix}$$

2] Matrix subtraction
similar to addition, subtract corresponding element

$$A - B = \begin{bmatrix} 1-5 & 2-6 \\ 3-7 & 4-8 \end{bmatrix} = \begin{bmatrix} -4 & -4 \\ -4 & -4 \end{bmatrix}$$

3] Scalar multiplication
Multiply each element of the matrix by a scalar

$$\text{if } K=3 \text{ then; } 3A = \begin{bmatrix} 3A_{11} & 3A_{12} \\ 3A_{13} & 3A_{14} \end{bmatrix} = \begin{bmatrix} 3 & 6 \\ 9 & 12 \end{bmatrix}$$

4] Matrix Multiplication
Multiply $m \times n$ matrix by a $n \times p$ matrix.

$$A = \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix} \quad B = \begin{bmatrix} 2 & 0 \\ 1 & 3 \end{bmatrix} \quad A \times B = \begin{bmatrix} 4 & 6 \\ 10 & 12 \end{bmatrix}$$

5] Transpose of a matrix
Flip the matrix over its diagonal. Rows become columns

$$A = \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}, \quad A^T = \begin{bmatrix} 1 & 3 \\ 2 & 4 \end{bmatrix}$$

6] Determinant (only for square matrices)
A scalar value that gives information about the matrix

$$\text{If } A = \begin{bmatrix} a & b \\ c & d \end{bmatrix}, \text{ then } \det(A) = ad - bc$$

GOPI
Page No.:
Date : / /

7] Inverse of a matrix
If A^{-1} is the inverse of A, then :

$A \cdot A^{-1} = I$

8] Identity matrix
A square matrix with 1s on the diagonal and 0s elsewhere

$$I = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

1] List types of algorithm

1] Brute force Algorithm
It tries all possible solutions until it finds the correct one

feature : Simple to implement
Inefficient for large inputs

e.g : Linear Search

2] Greedy Algorithm
It makes the best possible choice at each step, assuming it will lead to the optimal solution

features : - fast & simple
- may not always give best solution for complex problems

features e.x : Prim's & Kruskal's

(3) Divide & conquer
It divides problem into smaller sub-problems solve them independently and then combines their results

e.x : Merge sort, Quick sort, Binary Search

(4) Dynamic Programming
It is used to solve problems by breaking them into overlapping sub-problems, storing the results of sub-problems to avoid recomputation

features :- More efficient than recursion for overlapping sub-problems
- Uses memoization or tabulation

e.x : Fibonacci sequence, 0/1 Knapsack problem

(5) Backtracking algorithm
It is a recursive algorithm that tries to build a solution incrementally and abandons a path as soon as it determines that path will not lead to a valid solution

feature : Useful for constraint satisfaction problem

(6) Recursive algorithm
A recursive algorithm calls itself with a smaller input until a base condition is met

features : - easy to write but may lead to stack overflow.
- can be inefficient without memoization

e.g: factorial calculation

(7) Randomized Algorithm
This make random choices during execution to achieve a better average-case perform

feature : - may give diffrent results or performance on different runs

e.g : Quick sort

(8) Searching algorithm
Used to search for an element or value in a data structure.

e.g : Linear Search, Binary Search

(9) Sorting algorithm
Used to arrange data in a particular order
e.g : bubble, selection, merge, Quick

(10) Hashing algorithm
Used to map data to fixed-size values using hash functions, commonly used in hash table

GOPI
Page No.:
Date : / /

5] explain the various operations which can perform on set

Ans:

→ A set is a well-defined collection of distinct elements. Sets are widely used in mathematical operations.

(1) Union of sets

The union of two sets A & B is the set of all elements that are in A, in B or in both

$A \cup B = \{x \mid x \in A \text{ or } x \in B\}$

ex: $A = 1, 2, 3$, $B = 3, 4, 5$

$\therefore A \cup B = \{1, 2, 3, 4, 5\}$

(2) Intersection of sets

The intersaction of sets A & B is the set of all elements that are common to both A & B.

$A \cap B = \{x \mid x \in A \ \& \ x \in B\}$

$A = 1, 2, 3$, $B = 3, 4, 5$

$\therefore A \cap B = \{3\}$

(3) Difference of sets

The difference of set A & B (A−B) is the set of elements that are in A but not in B.

$A - B = \{x \mid x \in A \ \& \ x \notin B\}$

$A = 1, 2, 3$ & $B = 3, 4, 5$

$A - B = \{1, 2\}$; $B - A = \{4, 5\}$

GOPI
Page No.:
Date : / /

4] Symmetric difference

The symmetric difference of sets A & B is the set of elements which are in either of the sets, but not in both

$A \Delta B = (A - B) \cup (B - A)$

e.g: $A = \{1, 2, 3\}$ & $B = \{3, 4, 5\}$

$A \Delta B = \{1, 2, 4, 5\}$

5] Subset

A set A is a subset of B, if all elements of A are also in B

$A \subseteq B \Leftrightarrow \forall_x (x \in A \rightarrow x \in B)$

$A = \{1, 2\}$ & $B = \{1, 2, 3, 4\}$

$A \subseteq B$

6] Super Set

A set B is superset of A if it contains all elements of A.

$B \supseteq A$

$A = \{1, 2\}$ & $B = \{1, 2, 3, 4\}$

$B \supseteq A$

7] Complement of a set

The complement of set A is the set of all elements in the universal set U that are not in A

$A' = U - A$

e.g : $U = \{1, 2, 3, 4, 5\}$ & $A = \{1, 2\}$
$A' = \{3, 4, 5\}$

8] Cartesian Product

The cartesian product of two sets A & B is the set of all ordered pairs (a, b) where $a \in A$, $b \in B$

$A \times B = \{(a, b) \mid a \in A, b \in B\}$

e.g : $A = \{1, 2\}$ & $B = \{x, y\}$
$A \times B = \{(1, x), (1, y), (2, x), (2, y)\}$

6] Why do we use asymptotic notations in the study of algorithms ? Briefly describe the commonly used asymptotic notations.

Ans:

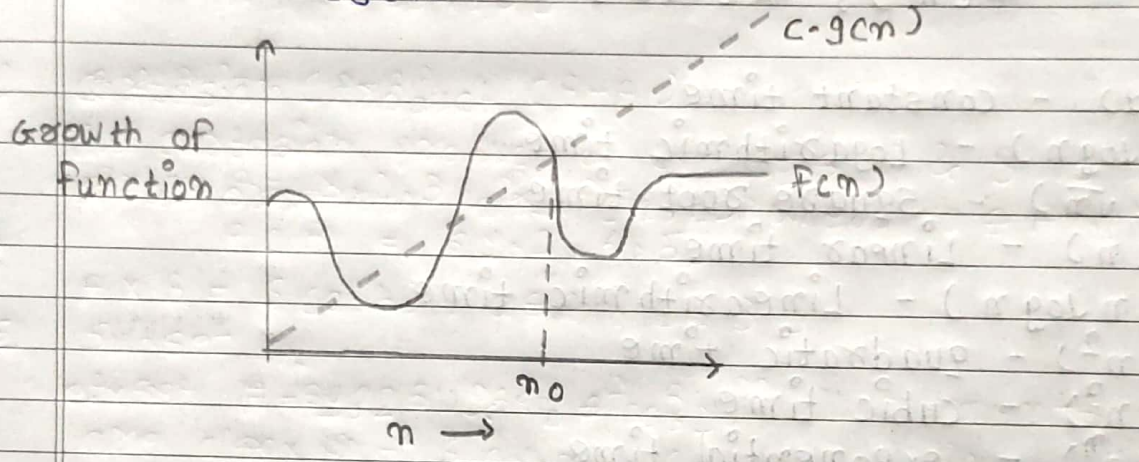
→ This are used to analyze the performance of algorithm especially when the input size becomes very large. They help us :

- measure time & space complexity
- compare algorithms
- predict scalability
- simplify analysis

(1) O - notation (upper band)

$0 \leq f(n) \leq c \cdot g(n)$ (for all $n \geq n_0$)

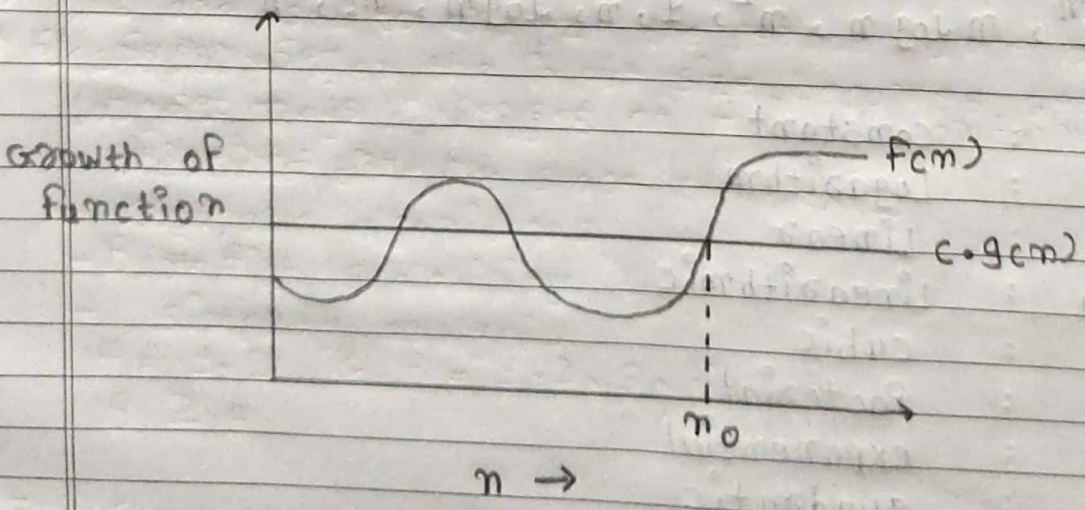
$f(n) = O(g(n))$



(2) Ω - notation

$0 \leq c \cdot g(n) \leq f(n)$ (for all $n \geq n_0$)

$f(n) = \Omega(g(n))$



(3) Θ - notation (Same order, tight bound)

$0 \leq c_1 \cdot g(n) \leq f(n) \leq c_2 \cdot g(n)$

$f(n) = \Theta(g(n))$

GOPI
Page No.:
Date : / /

7] Arrange following growth rates in increasing order :

(i) $O(n^{1/4})$, $O(n^{1.5})$, $O(n^3 \log n)$, $O(n^{1.02})$, $\Omega(n^6)$, $\Omega(n!)$, $O(\sqrt{n})$, $O(n^{6/2})$, $\Omega(2^n)$

$O(1)$ - constant time
$O(\log n)$ - Logarithmic time
$O(\sqrt{n})$ - Square root time
$O(n)$ - Linear time
$O(n \log n)$ - Linearithmic time
$O(n^2)$ - quadratic time
$O(n^3)$ - cubic time
$O(2^n)$ - exponential time
$O(n!)$ - Factorial time

Ans : $O(1) < O(\log n) < O(\sqrt{n}) < O(n) < O(n \log n) < O(n^2) < O(n^3) < O(2^n) < \Omega(2^n) < \Omega(n!)$

(ii) 2^n, $n \log n$, n^2, 1, n, $\log n$, $n!$, n^3

1 : constant
$\log n$: Logarithm
n : Linear
$n \log n$: Linearithmic
n^3 : cubic
$n!$: factorial
2^n : exponential
n^2 : quadratic

Ans : $1 < \log n < n < n \log n < n^3 < n! < 2^n < n^n$

GOPI
Page No.:
Date : / /

8] $5n^3 + 2n = O(n^3)$

L.H.S = $5n^3 + 2n$

$\therefore 0 \leq f(n) \leq c \cdot g(n)$ (for all $n \geq n_0$)

Let's assume $c \cdot g(n) = 5n^3 + 2n + n$
$= 5n^3 + 3n$

$\therefore 0 \leq f(n) \leq c \cdot g(n)$
$\therefore 0 \leq 5n^3 + 2n \leq 5n^3 + 3n$

here, $c = 1$, $g(n) = 5n^3 + 3n$
(ignore constant)
$= n^3$

$f(n) = O(g(n))$
$= O(n^3)$
$= R.H.S$

9] (i) Find out big-oh notation of the $f(n) = 3n^2 + 5n + 10$

big-oh notation : $0 \leq f(n) \leq c \cdot gn$

assume $c \cdot gn = 3n^2 + 5n \cdot n + 10$
$= 3n^2 + 5n^2 + 10$

$\therefore 0 \leq 3n^2 + 5n + 10 \leq 3n^2 + 5n^2 + 10$

here all condition is true.

$c \cdot g(n) = 8n^2 + 10$

here, $c = 1$; $g(n) = 8n^2 + 10$

$f(n) = O(g(n))$
$= O(8n^2 + 10)$
$= O(8n^2)$
$= O(n^2)$

(ii) Find omega (Ω) notation of function
$f(n) = 2n + 6n * \log n + 6n$

Ω notation : $0 \leq c \cdot g(n) \leq f(n)$

let's assume $c \cdot g(n) = 2n + 6n * \log n$

$\therefore 0 \leq 2n + 6n * \log n \leq 2n + 6n * \log n + 6n$

here all condition is satisfied

$f(n) = \Omega(g(n))$
$= \Omega(2n + 6n * \log n)$

(iii) find out θ-notation for function $f(n) = 27n^2 + 16n$

θ-notation : $0 \leq c_1 \cdot g(n) \leq f(n) \leq c_2 \cdot g(n)$

let's assume $c_1 \cdot g(n) = 20n^2 + 16n$
$c_2 \cdot g(n) = 30n^2 + 16n$

$0 \leq 20n^2 + 16n \leq 27n^2 + 16n \leq 30n^2 + 16n$

$c \cdot g(n) = n^2$

$\therefore f(n) = \Theta(g(n))$
$= \Theta(n^2)$

10] Write an algorithm for insertion sort. Sort given array $A = \{27, 16, 44, 95, 67, 32, 78\}$ using insertion sort algorithm. Also perform analysis

$A = 27, 16, 44, 95, 67, 32, 78$

Sorted part	Un-Sorted part
27	16, 44, 95, 67, 32, 78
	i
27, 16	44, 95, 67, 32, 78
	i
44, 27, 16	95, 67, 32, 78
	i
44, 27, 16, 95	67, 32, 78
	i
44, 27, 16, 67, 95	32, 78
	i
44, 27, 32, 16, 67, 95	78
	i
44, 27, 32, 16, 67, 78, 95	

$\therefore$ Sorted array : 44, 27, 32, 16, 67, 78, 95

GOPI
Page No.:
Date : / /

Algorithm :

Algorithm InsertionSort (A[0 ... n-1])

for i ← 1 to n-1
do
{
key ← A[i]
j ← i-1

while j ≥ 0 & A[j] > key
do
A[j+1] ← A[j]
j ← j-1
end while

A[j+1] ← key
end for

analysis :

T(n) = Outer for loop with variable i × Inner for loop with variable i × base operation

$$= \sum_{i=1}^{n-1} \cdot \sum_{j=i-1}^{0} 1$$

$$= \sum_{i=1}^{n-1} i$$

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

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

$= O(n^2)$

[11] Sort the letters of word "programming" in alphabetical order using bubble sort. Write an algo for bubble sort. Also do analysis

A = ['p', 'R', 'O', 'G', 'R', 'A', 'm', 'm', 'i', 'n', 'g']

(do swapping with sorter one)

A = ['p', 'R', 'O', 'G', 'R', 'A', 'm', 'm', 'i', 'n', 'g']

A = ['p', 'R', 'O', 'G', 'R', 'A', 'm', 'm', 'i', 'n', 'g']

A = ['p', 'O', 'R', 'G', 'R', 'A', 'm', 'm', 'i', 'n', 'g']

A = ['p', 'O', 'G', 'R', 'R', 'A', 'm', 'm', 'i', 'n', 'g']

A = ['p', 'O', 'G', 'R', 'A', 'R', 'm', 'm', 'i', 'n', 'g']

A = ['p', 'O', 'G', 'R', 'A' 'm', 'R', 'm', 'i', 'n', 'g']

A = ['p', 'O', 'G', 'R', 'A', 'm', 'm', 'R', 'i', 'n', 'g']

A = ['p', 'O', 'G', 'R', 'A', 'm', 'm', 'i', 'R', 'n', 'g']

A = ['p', 'O', 'G', 'R', 'A', 'm', 'm', 'i', 'n', 'R', 'g']

A = ['p', 'O', 'G', 'R', 'A', 'm', 'm', 'i', 'n', 'g', 'R']

GOPI
Page No.:
Date : / /

(do the same iteration till end)

final = 'A', 'G', 'G', 'I', 'm', 'm', 'N', 'O', 'P', 'R', 'R'

algorithm :

Algorithm BubbleSort (A [0 ... n-1])

for i ← 0 to n-2
do
  for j ← 0 to n-2-i
  do
    if [A][j] > A [j+1] then
      temp ← A [j]
      A [j] ← A [j+1]
      A [j+1] ← temp
    end if
  end for
end for

analysis

T(n) : Outer for loop x Inner for loop x base operation
with variable i with variable i

$$= \sum_{i=0}^{n-2} \cdot \sum_{j=0}^{n-2-i} \cdot 1$$

$$= \sum_{i=0}^{n-2} [\, n-2-i-0+1 \,]$$

GOPI
Page No.:
Date : / /

$$= \sum_{i=0}^{n-2} [n-i-1]$$

$$= \sum_{i=0}^{n-2} (n-1) - \sum_{i=0}^{n-2} i$$

$$= \sum_{i=0}^{n-2} (n-1) - \left[\frac{(n-2)(n-2+1)}{2}\right]$$

$$= \sum_{i=0}^{n-2} (n-1) - \left[\frac{(n-2)(n-1)}{2}\right]$$

$$= \sum_{i=0}^{n-2} (n-1) - \left(\frac{n^2-3n+2}{2}\right)$$

$$= (n-1) \sum_{i=0}^{n-2} \cdot 1 - \left(\frac{n^2-3n+2}{2}\right)$$

$$= (n-1)(n-2-0+1) - \left(\frac{n^2-3n+2}{2}\right)$$

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

$$= \frac{2n^2-4n+2-n^2+3n-2}{2}$$

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

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

GOPI
Page No.:
Date : / /

12] What is an amortized analysis ? explain accounting method and aggregate analysis with suitable example.

Ans:

→ Amortized analysis is a method to analyze the average time per operation over a sequence of operations, even if a single operation might be expensive.

→ It gives a better estimate of the overall performance than worst-case analysis when we perform a series of application

→ In many data structure some operations are cheap and a few are expensive. Amortized analysis ensures we won't (don't) overestimate the cost due to these rare expensive operations

3 types : (1) Aggregate method
(2) Accounting method
(3) Potential method

(1) Aggregate method

Total cost of all operations is computed first. Then average cost per operation is calculated by dividing total cost by number of operations

ex : Dynamic Array

GOPI
Page No.:
Date : / /

initial size = 1
when full, the array doubles in size
copying elements during resizing is costly

Most insertion take O(1) time
few insertion take O(n) time due to rezing

Total cost: 1, 2, 4, 8, ..., n → log n times
total cost = O(n)

Amortized cost: = Total cost / number of operation
$= \frac{O(n)}{(n)}$
$= O(1)$

(2) Accounting Method

Assign different 'charges' to operation. over-charge some cheap operations and use that extra cost to pay for expensive ones later

e.x: stack with push & multi-pop

operation: push(x): Pushes x to stack
multiPop(k): pops min (k, current size)

Real cost: push O(1)
multiPop: up to O(n)

GOPI
Page No.:
Date : / /

Assign Amortized cost :

Push : charge 2 unit
- 1 unit to push
- 1 unit 'saved' for future pop

MultiPop : 0 cost
- use saved credits from PUSH to pay for POP

Amortized cost :

Push (n times) : $2 \times n = 2n$
MULTIPOP : free