PAGE NO.:
DATE / /

NAME :- AFNAN ATTAR PRN :- F19112003 CLASS :- BE COMP II SUBJECT :- DAA UNIT TEST :- 01

Q1) List different types of algorithm ? Explain any one with example.

Ans 1) Brute Force 2) Recursive 3) Randomised 4) Sorting 5) Searching 6) Hashing

- Brute Force Algorithm :-

1. It is an itutive, direct and straightforward technique of problem solving in which all the possible ways or all possible solutions to a problem are enumerated.
2. Brute force approach is guaranteed way to find correct solution.
3. Brute force approach is inefficient. For real-time problems algorithm analysis often goes beyond O(N!).
4. Many problems in our day-to-day life are solved using brute-force strategy for example exploring all paths to a market to find minimum shortest path.

Q2) Write a short note on order of growth.

Ans 1. An order of growth is a set of functions whose asymptotic growth behaviour is considered equivalent.
2. For example 2n, 100n and n+1 belong to the same order of growth which is O(n) (Big-Oh) and often called linear as the function grows linearily with n.
3. All functions with leading term n^2 belong to $O(n^2)$; called as quadratic.
4. Some order of growth are as follows :-
   a) O(1) : Constant (b) O(log n) : Logarithmic

(c) $O(n)$: Linear (d) $O(n \log n)$: n log n (e) $O(n^2)$: Quadratic
(f) $O(n^3)$: Cubic (g) $O(c^n)$: Exponential

Q3) Define Asymptotic Notation? Explain their significance in analysing algorithms.

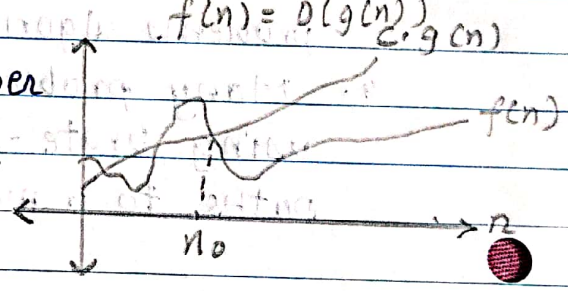
Ans 1. Asymptotic notations are the mathematical notations used to describe running time of an algorithm when the input tends towards a particular value or a limiting value.

2. Suppose that we are interested in the properties of a function $f(n)$ as n becomes very large.

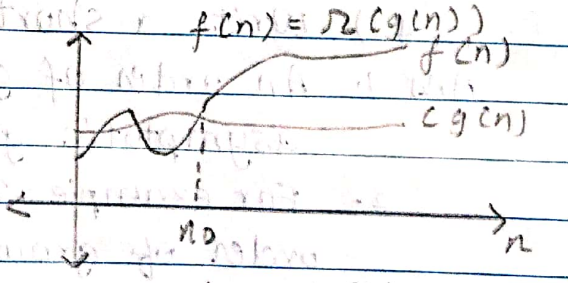
3. If $f(n) = n^2 + 3n$, as $n \to \infty$, "3n" becomes an insignificant term and we say $f(n)$ is asymptotically equivalent to n^2.

4. Lets go through each asymptotic notations:-

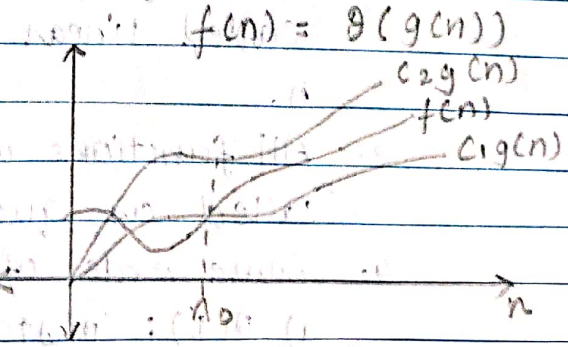
i) Big-O notation:-
Big-O notation represents upper bound of the running time of algorithm and gives its worst case complexity.



ii) Omega Notation (Ω):-
Omega notation represents the lower bound of the running time of algorithm and gives its best case complexity.



iii) Theta Notation (Θ):-
Theta notation encloses function from above and below and gives average running time of a algorithm.



PAGE NO.:
DATE / /

Q4) What is NP Completeness? Prove it for Clique Decision problem.

Ans 1. NP-complete problems are the hardest problems in the NP set.

2. A decision problem L is NP-complete if:

i) L is in NP. (ii) Every problem in NP is reducible to L in polynomial time.

* Clique decision problem :-

- To prove :- Clique is an NPC (NP Complete) problem.
- Proof :- we will try to prove the following :-
  - i) 3CNF $\leq_P$ Clique
  - ii) Clique $\leq_P$ 3CNF $\leq$ SAT
  - iii) Clique $\in$ NP

i) 3CNF $\leq_P$ Clique :

1. Draw the clause in the form of vertices and each vertex represents the literals of the clauses.
   a) They do no complement each other.
   b) They do not belong to same clause.
2. In the conversion, the size of the clique and size of 3CNF must be the same and you successfully converted 3CNF into clique within the polynomial time.

ii) Clique $\leq_P$ 3 CNF

1. For a function of K clause there must exist a clique of size k. It means P variables which are from different clauses can assign the same value. By using these values of all variables of the cliques, you can make the value of each clause in the function is equal to 1.

iii) Clique $\in$ NP

1. We can get clique through 3CNF and to convert the decision-based NP problem into 3 CNF we have to

first convert into SAT and SAT comes from NP.

2. Hence we conclude clique belongs to NP.

- Clique belongs to NP Complete:

1. Reduction achieved within polynomial time from 3CNF to Clique.
2. Verified the output after Reduction from clique to 3CNF above.

Hence as reduction and verification can be done in polynomial time, clique belongs to class NP complete.

Qs) Write short note on classification of time complexity.

Ans 1. According to the number of operations performed for a given input we classify time complexity.

2. We use Big-Oh notation to classify algorithms based on their running time or space (memory used) as the input grows.
3. The O() functions grow in function of size of n.
4. Classification of time complexity:-

Notation	Name	Example
1) O(1)	Constant	Odd or Even Number
2) O(logn)	Logarithmic	Binary search
3) O(n)	Linear	Max element in an array
4) O(nlogn)	Linearithmic	Sorting elements in array.
5) $O(n^2)$	Quadratic	Sorting array with bubble sort.
6) $O(n^3)$	Cubic	3 variable solution.
7) $O(2^n)$	Exponential	Power Set
8) O(n!)	Factorial	All permutation of a string.