Data Structure & Algorithms (DSA)

Saathi

Date / DSA - (MC1T001)

Unit 1

Complexity Analysis and Hashing :-

* Complexity Analisys :- Time and space Complexity Asymptotic Notations (Big O, Theta, Omega.

* Effecient Algorithms :- Importance & performance measurements.

* Hashing :- Implementation of Dictionaries, hash functions, handling colisions, Open addressing and analysys of search operation.

* Data Structure :-

(i) Data structure is created with the help of two words i.e. Data & Structure that means we have to store the data iin the form of structure in the main memory.

(ii) A data structure is a mathematical or a logic way of organizing data in a memory where not only data items stored inside them but also relationship among each data is also considered

DS = Organized data + Allowed ~~dat~~ operations

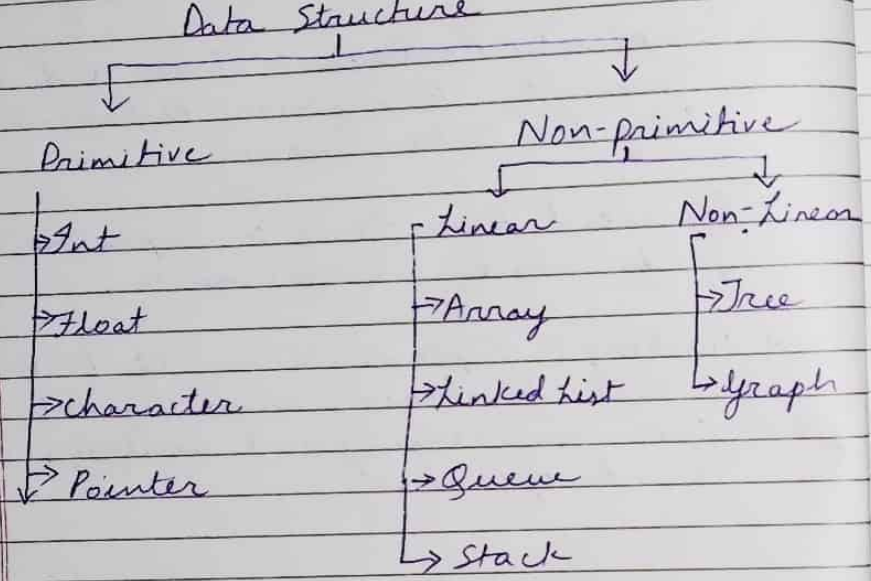
No.

Page No.

Saathi

Date __/__/____

Classification of Data Structure



Saathi

Date __/__/____

* Algorithm Complexity:-

Algorithm is a finite sequence of steps or instruction which can be carried out to solve a particular problem in-order to obtain the desired result.

* Time Complexity:-

The time complexity of a program or algorithm is the amount of time required by the program statement to execute.

The most commonly used expressed in Big O notation.

The most commonly used Big O Notations are:-

① O(1) :- It describe algorithms that will always execute in the same time regardless of the input size.

② O(log N) :- It takes a fixed additional amount of time each time. Each time the input size doubles and it is called as logarithmic time algorithms.

③ O(N) :- Describes an algorithm whose performance will grow linearly and indirect proportion to the input size.

④ $O(N^2)$:- It describes an algorithm whose performance is directly proportional to square of the input size. and it is called as polynomial time algorithm.

⑤ $O(2^N)$:- It denotes an algorithm whose growth doubles with each addition to the input size.

Input Size	$O(1)$	$O(\log N)$	$O(N)$	$O(N^2)$	$O(2^N)$
1	1	1	1	1	2
2	1	1	2	4	4
4	1	3	4	16	16
8	1	4	8	64	256
16	1	5	16	256	65536

Ex1:-

```
int i=1          → 1
loop(i<=n)       → n+1
print i          → n
i = i+1          → n
```

total $3n+2$ (Ignore +2)

∴ Time complexity of above example is
→ $O(N)$

Ex2:-

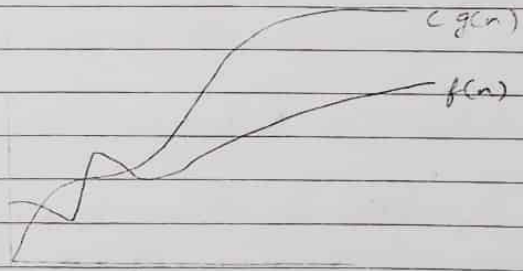
```
int sum(int a[], int n) → 1
{
  int total = 0;                → 1
  for(int i=0; i<n; i++)        → 1, n+1, n   (2n+2)
  {
    total = total + a[i]        → n
  }
  return(total);                → 1
}
```

$3n+5$

→ $O(N)$

* Asymptotic Notation for Analysis of Algorithm

① Big O Notation :- Big-O notation represents the upper bound of the running time of an algorithm. Therefore, it gives the worst-case complexity of an algorithm.



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

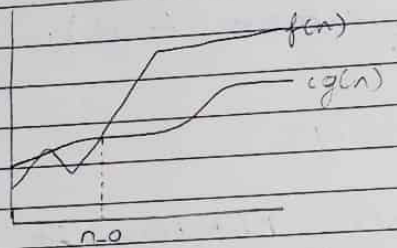
$O(g(n)) = \{f(n)$: there exist positive constant c and n_0 such that $0 \le f(n) \le cg(n)$ for all $n \ge n_0$

Saathi

Date __/__/____

② Omega Notation (Ω Notation):-

Omega notation represents the lower bound of the running time of an algorithm. Thus it provides the best case complexity of an algorithm is time complexity.



f(n) = Omega (g(n))

Mathematical Representation of Omega notation:-

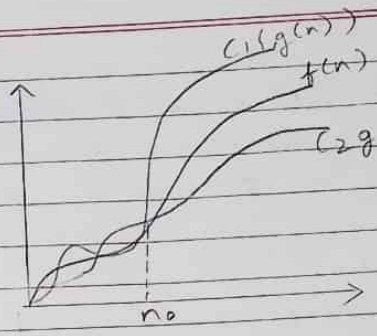
Ω(g(n)) = {f(n): there exist positive constant c and no such that 0 ≤ cg(n) ≤ f(n) for all n ≥ n0}

③ Theta Notation (Θ Notation):-

Theta Notation encloses the function from ~~lower~~ above and below. Since it represents the upper and lower of the running time of an algorithm, it is used to analyze the average-case complexity of an algorithm.

Saathi

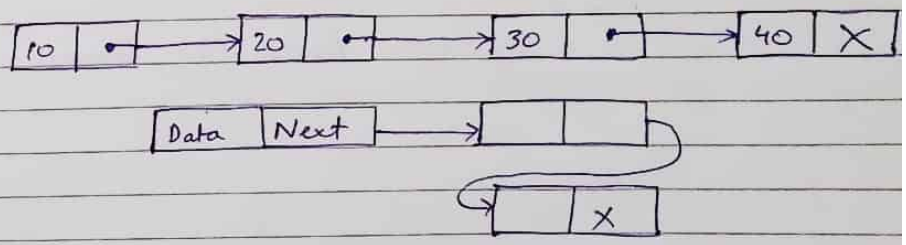
Date __/__/____



Mathematical Representation of Theta Notation

Θ(g(n)) = {f(n): there exist positive constants c1, c2 and n0 such that 0 ≤ c1 * g(n) ≤ f(n) ≤ c2 * g(n) for all n ≥ n0}

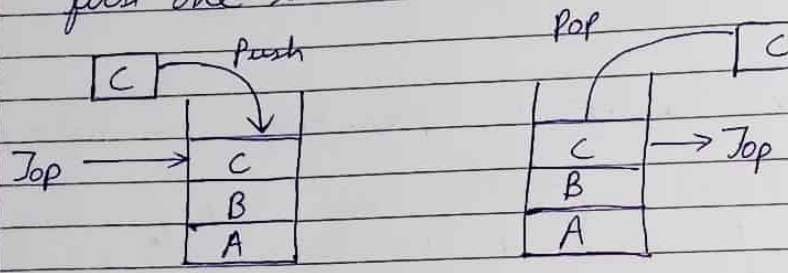
* Linked List :- A linked list is a linear data structure which looks like a chain of nodes, where each node contains a data field and reference (link) to the next node in the list. Unlike Arrays, linked list elements are not stored in continuous location.



Saathi

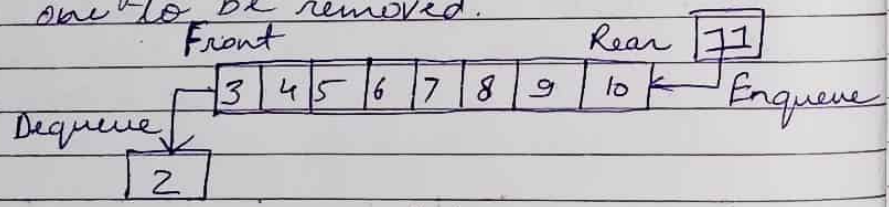
Date __/__/____

* Stack :- A stack is a linear data structure that follows the last-In-First-Out (LIFO) principle, meaning that the element added to the stack is the first one to be removed.



Stack Data Structure.

* Queue :- A queue is a linear data structure that follows the First-In-First-Out (FIFO) principle. In a queue, the first element added is the first one to be removed.



Queue Data Structure

Page No.

Saathi

Date __/__/____

* Linear Search :- In this method elements to be search is checked in the entire data structure in a sequential way from starting to end is called as Linear search.

Linear search is an easiest and straight most searching technique as search element is compared to all the elements in an array.

Algorithm :-

① Repeat i=0 to N-1
② If Array[i] == item
③ Print index and return from loop.
④ if i==N
⑤ Print "Element not found".

Ex :- Item = 18

Index :-	0	1	2	3	4
	12	13	14	18	20

∵ 18 is item
∴ i = 0 to N = 1 (∵ N = Size of array i.e. 5
∴ N = 5

① ∴ i=0 to 4
∴ 12 ≠ 18

② ∴ i=1 to 4
13 ≠ 18

③ i=2 to 4
14 ≠ 18

④ i=3 to 4
18 == 18

Page No.