28/8/19

1 Introduction To Data Structure

* Data : It is collection of information but it is in raw form.

or Unorganized fact that required to be processed to make it meaningful.

or Data can be a number, Symbol, Character or any kind of information.

* Information: Processed data is known as information.

or logical & meaningful form of data.

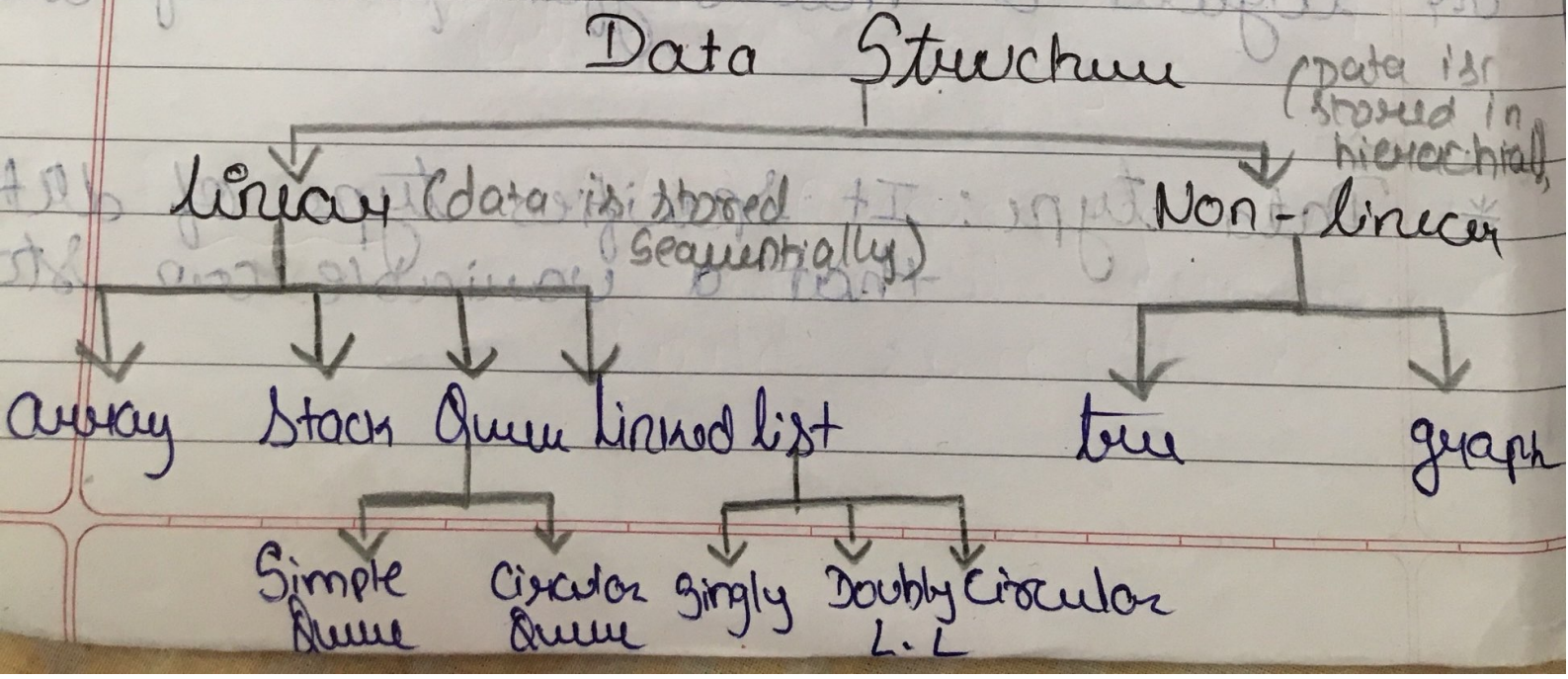
* Data type: It specifies type of data that a variable can store.

Data Type

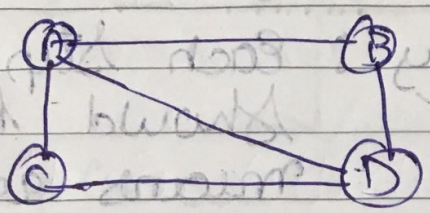
- **Primitive** (Built-in / Predefined data type)
  - Ex: int, float, char, double, bool etc
- **Non-Primitive** (Derive data type)
  - Ex: Structure, union

* Data Structure

→ Data Structure is a systematic way to store & organize data in specific format. So, it can be used efficiently.



* Graph - collection of nodes & Edges



$G = (V, E)$

Vertices (nodes) — Edges

* Operation On Data Structure

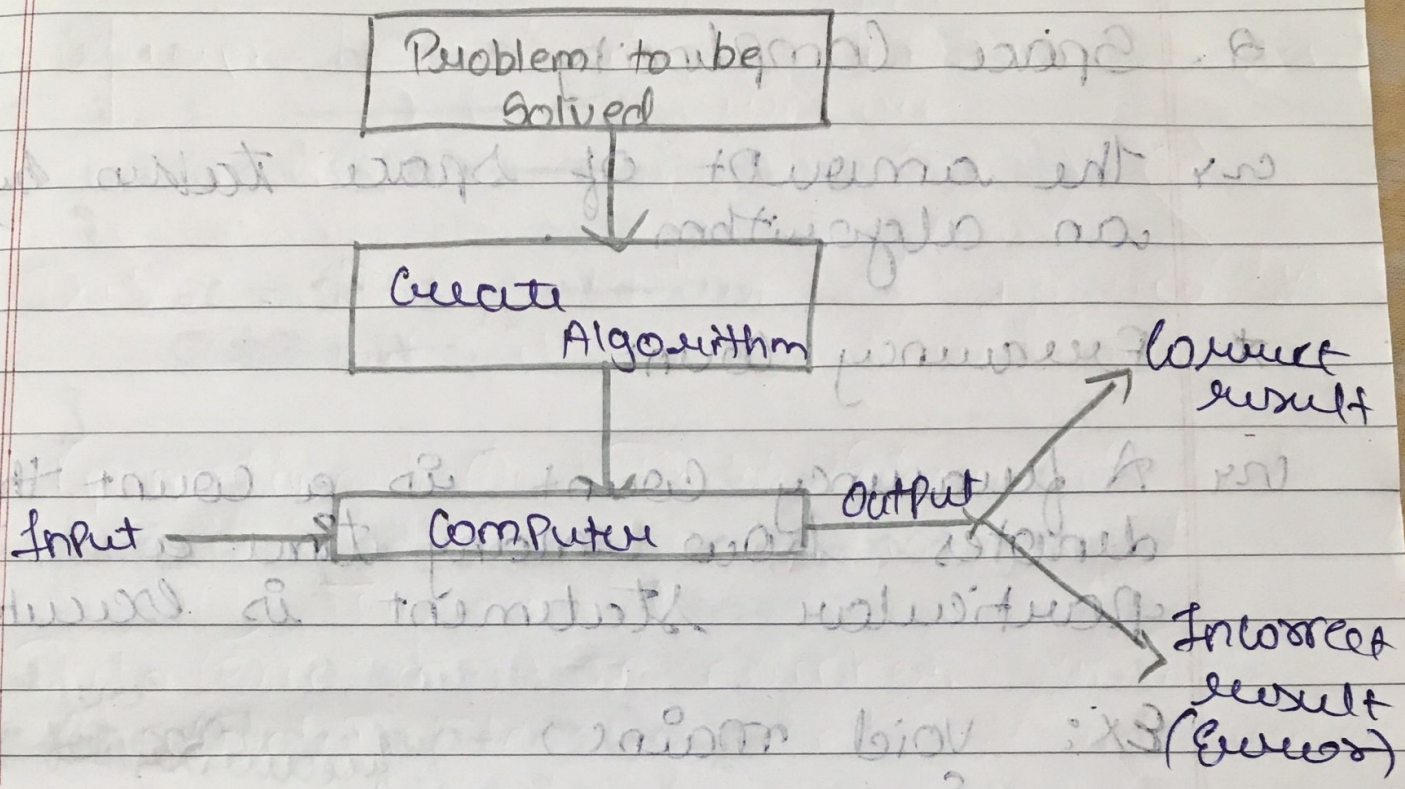
1. Traversal
2. Insertion
3. Delete
4. Search
5. Sorting
6. merging

Algorithm

Algorithm is collection of unambiguous instructions occuring in some specific sequence such an algorithm should produce output for given set of input in finite amount of time.

* Properties Of Algorithm

- Non-ambiguity : Each step in algorithm should be unambiguise means each instruction should be clear & precise
- Range Of Input : Range Of Input should be specified otherwise algo can go in infinite states.
- multiplicity : The same algorithm can be represented in several different way.
- Speed : The algorithm should be efficient and should produce result speedily.
- finiteness : After performing required operation the program should terminate.



* Analysis Of Algorithm

① Time Complexity

② Space Complexity

1. Time Complexity

⇒ The amount of time taken by an algorithm to run is called time complexity.

Q. Space Complexity

=> The amount of space taken by an algorithm.

* Freaency Count

=> A freaency count is a count that denotes how many time a particular statement is executed.

Ex:
```
void main()
{
    int a;
    a = 5;          ------> 1
    printf("%d", a); ------> 1
}
```

} freaency count of this are neglected

freq. Count = 2

Ex:
```
int i, n
i = 1 ---------- 1
while (i <= n) ---------- n+1
{
    x = x + 1; ---------- n
    i++; ---------- n
}
```

frequency count = 1 + n+1 + n + n
= (3n+2)

Complexity = O(n)

Ex:
```
void main()
{
    int a;
    a = 0; ---------- 1
    for (i=1; i<=n; i++) ---------- n+1
    {
        a = a + i; ---------- n
    }
    printf("%d", a); ---------- 1
}
```

freq. count = 1+1+n+1+n+n+1
= 3n+4 Complexity = O(n)

Ex:

```
for(i=1;  i<=n;  i++)        // 1, n+1, n
{
    for(j=1; j<=n; j++)      // n+1, n(n+1), n^2
    {
        x = x+10;            // n^2
    }
}
```

freq. count = $1 + n+1 + n + n + n^2 + n + n^2 + n^2$

$= 3n^2 + 4n + 2$

Complexity = $O(n^2)$

Ex:

```
for(i=1; i<=n; i++)          // 1, n+1, n
{
    for(j=1; j<=n; j++)      // 1, n+1, n(n+1)
    {
        c[i][j] = 0;         // n^2
        for(x=1; x<=n; x++)  // n^2, n^2+1, n^3+n^2
        {
            c[i][j] = c[i][j] + a[i][x] *
                      b[x][j];
        }
    }
}
```

freq. count = $1 + n+1 + n + n + n^2 + n + n^2 + n^2 + n^2 + n^3 + n^2 + n^3 + n^3$

$= 3n^3 + 5n^2 + 4n + 2$

Complexity = $O(n^3)$

30/8/19

* Performance Analysis & measurement of Algorithm

① Best Case time Complexity

Or If algo. takes minimum amount of time to run to compute specific set of input is called best case time complexity.

$C_{Best} = 1$

8	14	16	20	23

key = 8

---

(2) worst Case Time Complexity

=> Algo. takes max. amount of time to run to complete specific set of i/p is called worst case time complexity

$$C_{worst} = n$$

(3) Average Case time Complexity

=> Amount Of Sum Computational time used by algo. average over all possible i/p

$$C_{average} = \frac{\text{all Possible cases time}}{\text{no. of cases}}$$

$$= \frac{n+1}{2}$$

* Asymptotic Notation (O, Ω, Θ)

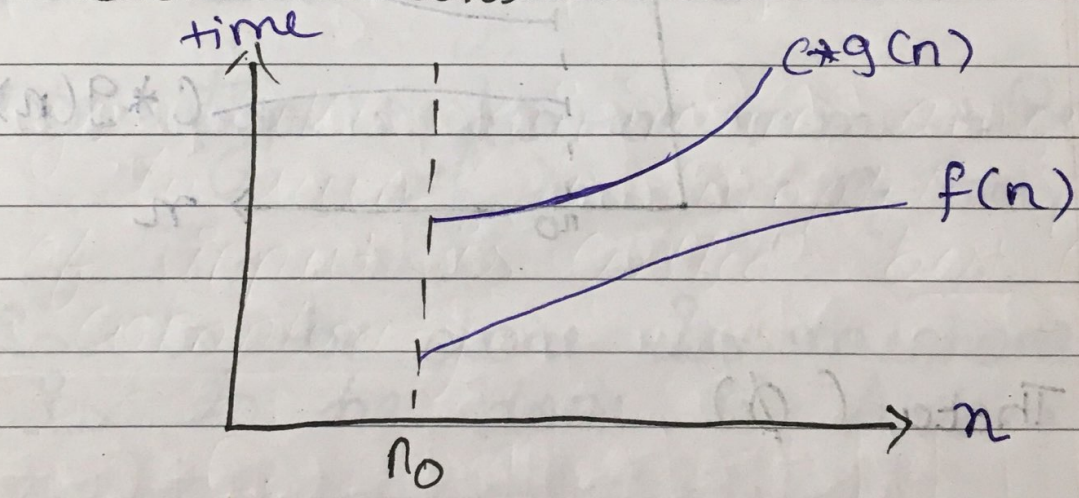
① Big o (O)

→ It is method of representing the upper bound of algorithm's running time.

→ denoted by "O".

$$f(n) \leq C * g(n)$$

(f(n): function; C: constant; g(n): function)

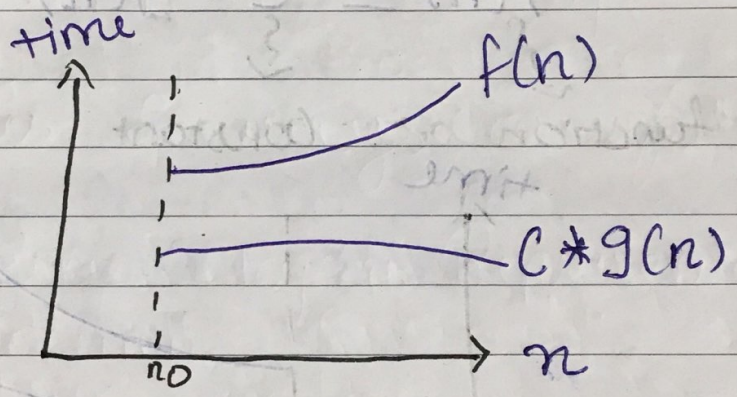


2. Omega (Ω)

↳ This notation is used to represent the lower bound of algorithm's running time.

↳ denoted by Ω

$$f(n) \geq c * g(n)$$

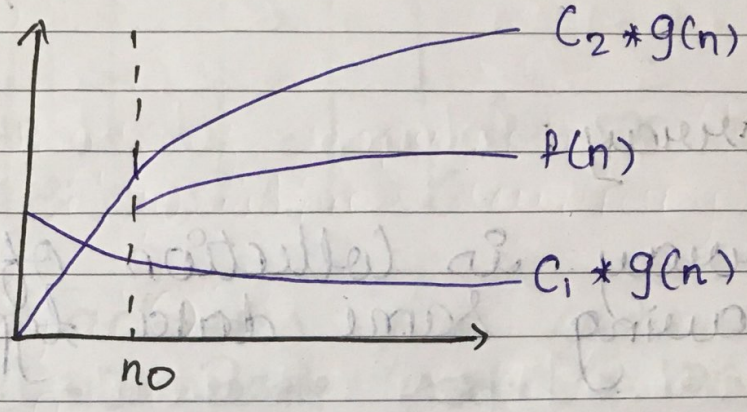


3. Theta (ϕ)

↳ Using ϕ notation the running time is between upper bound & lower bound

↳ denoted by ϕ

$C_1 * g(n) \leq f(n) \leq C_2 * g(n)$



31/8/19

Abstract Data Type (ADT)

ADT = type + function name + Behaviour of Each function.

→ What is to be done is mentioned but how is to be done is hidden.