10/10/21

★ Assignment-1

1. Define is data structure. Explain different types of data structure. And also explain types of data-types.

Ans • Data Structure is a storage that is used to store & organize data.
- It is a way of arranging data on a computer so that it can be accessed & updated efficiently.
- Data structure & data types are slightly different.
- Data Structure is collection of data types arranged in a specific order.

* Types of Data Structure:-

1. Linear data structure
2. Non-linear data structure

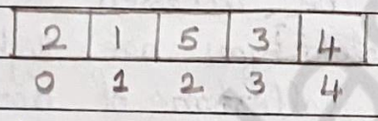
(1) Linear Data Structure:-

- In this, elements are arranged in sequence one after the other. Since elements are arranged in particular order, and easy to implement
- When complexity increases, it might be not the best choice.

- Popular linear data structures are:-

(i) Array:-

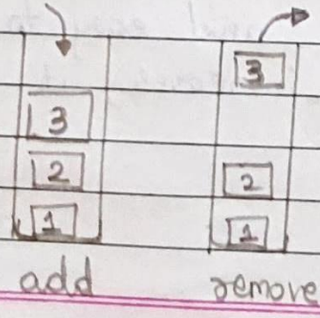
- In an array, elements in memory are arranged in continuous memory.
- All elements of array are of same type.



Array with each element represented by an index

(ii) Stack:-

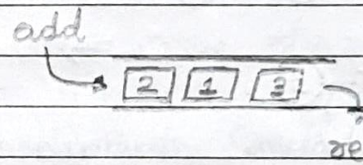
- In stack, elements are stored in LIFO principle. That is, the last element stored in a stack will be removed first.
- It works just like pile of plates where last plate kept on pile will be removed first.



In stack, operations can be performed only from 1 end (top here).

(iii) Queue:-

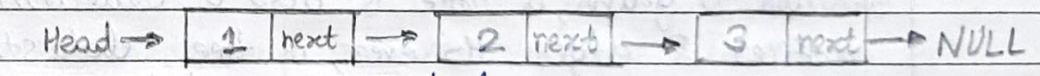
- Unlike stack, queue works in FIFO principle where first element stored in queue is removed first.
- It works just like a queue of people in ticket counters where first person on queue will get ticket first.



In queue, add & remove are performed from separate ends.

(iv) Linked list:-

- In this, data elements are connected through a series of nodes. And, each node contains the data items & address to next node.



A linked list

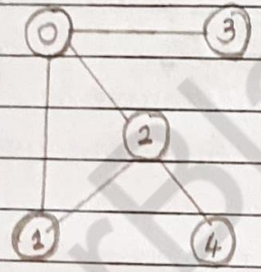
(2) Non-linear Data structure:-

- In this, elements in non-linear data structures are not in any sequence. Instead they are arranged in a hierarchical manner where one element will be connected to one or more elements.

- They are divided into :-

(i) Graph :-

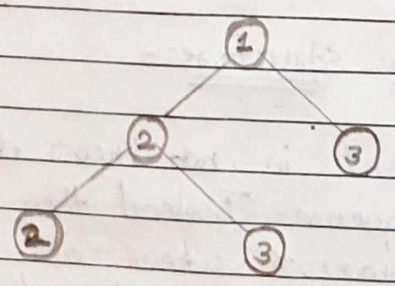
- In graph, each node is called vertex & each vertex is connected to other vertices through edges.



Graph data structure eg.

(ii) Trees :-

- Similar to graph, a tree is also a collection of vertices & edges. However, in tree data structure, there can only be one edge between 2 vertices.



Tree data-structure e.g.,

2. What is a sparse matrix? Explain Sparse matrix with example.

Ans - It is a matrix that contains a few non-zero elements.
- Almost all placed are filled with 0.
- And if non-zero elements in matrix are more than zero elements in matrix then it is called sparse matrix.
- And in case size of matrix is big, a lot of space is wasted to represent such a small no. of non-zero elements. Similarly for scanning same non-zero will take more time.
- Thus to limit processing time & space usage instead of storing a less no. of non-zero elements in a matrix, we use :-

* Array Representation :-

- The 2D array is converted to 1D array with 3 columns representing.

a. Row → Row index of non-zero element
b. Column → Column index of non-zero element
c. Value → Value at same row, column index in 2D.

- For eg,

	0	1	2	3	4	5	6	7
0	0	9	0	0	0	4	0	0
1	0	0	6	0	0	0	1	0
2	0	0	0	5	0	0	1	0
3	0	0	0	0	0	0	3	0
4	0	0	6	0	0	0	0	0

Total elements => 40 => 5 rows x 8 columns

Step 1:- list all non-zero elements in matrix with their rows & column index.

Step 2:- Create new array with following dimensions.
- Rows = no. of non-zero elements
- Columns = 3 (Row, Column & value)

Step 3:- Fill array with non-zero elements.

Row	Column	Value
0	1	9
0	5	4
1	2	6
1	6	1
2	3	5
2	6	1
3	6	3
4	2	6

- Total = 80 elements

Total Space occupied = 80 x 2 = 160 bytes

- Total no. of non-zero elements = $N_{(nz)}$ => 8
- Formula for non-zero rows => $(N_{(nz)} + 1)$

$(N_{(nz)} + 1) \times 3 \Rightarrow 9 \times 3 \Rightarrow 27$

- Non-zero elements space occupied => 27 x 2 = 54 bytes.
- Free Space => 160 − 54 = 106 bytes

3. Explain asymptotic notations in brief.

Ans:- To choose the best algorithm, we need to check efficiency of each algorithm.
- The efficiency can be measured by computing the time complexity of each algorithm.
- Using asymptotic notations, we can given time complexity as "Fastest possible", "Slowest possible" or "Average possible".
- It is a shorthand way to represent the time complexity.
- Various notations such as Ω, θ, O used are called

① Big-Oh Notation (O):-

- Denoted by O. It represents upper bound to algorithm's running time.
- It gives max. amount of time taken by an algorithm to complete.

* Def^n:- Let f(n) & g(n) be 2 non-negative functions.

- Let n_0 & constant c are 2 integers such that $n \geq n_0$.
- Similarly c is some constant such that $c > 0$.

$$F(n) \leq c * g(n)$$

- then F(n) is big Oh of g(n).
- Also denoted by $F(n) \in O(g(n))$.

② Omega Notation (Ω):-

- Denoted by 'Ω'. It represents lower bound to algorithm's running time.
- It gives minimum amount of time taken by an algorithm to complete.

* Def^n:- A function F(n) is said to be in $\Omega(g(n))$ if f(n) is bounded below by some positive constant multiple of g(n) such that

$$F(n) \geq c * g(n) \quad \text{for all } n \geq n_0$$

- It is denoted as $f(n) \in \Omega(g(n))$.

③ Θ Notation:-

- It is denoted by Θ. It By this method the running time is between upper bound & lower bound.

* Def^n:- Let f(n) & g(n) be 2 non-negative func. There are 2 +ve constants namely c_1 & c_2 such that

$$c_1 * g(n) \leq f(n) \leq c_2 * g(n)$$

- Thus, we can say that,

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

4. Differentiate linear & non-linear data structure.

Ans

Linear Data Structure	Non-Linear Data Structure
(i) The data items are arranged in sequential orders, one after another.	(i) The data items are arranged in non-sequential orders. (Hierarchical manner).
(ii) All the items are present on the single layers.	(ii) The data items are present at diff. layers.
(iii) It can be transversed on a single run. That is, if we start from the first element, we can transverse all elements sequentially in a single pass.	(iii) It requires multiple runs. That is, if we start from the first element it might not be possible to transverse all the elements in a single pass.
(iv) The memory utilization is not efficient.	(iv) Diff. structures utilize memory in diff. efficient ways depending on the need.
(v) The time complexity increase with the data size.	(v) Time complexity remains the same.
(vi) E.g., Arrays, Stacks, Queue	(vii) E.g., Tree, Graph.

5 Write an algorithm to count the total no of even elements in the list & find its time complexity.

Ans Step 1:- [Initialize] Set $i=0$, n, even=0, odd=0

Step 2:- Repeat Step 3 & 4 while $i<n$.

Step 3:- If (a[i] % 2 == 0)

then even ++

Else

odd ++

Step 4:- i++

[END OF LOOP]

Step 5:- Print even no of elements

Step 6:- Exit

* Time complexity

Statement	Freq. Count
i = 0	0
i<n	n+1
n[i]%2	n
i++	n
Total	3n+1

∴ The highest orders degree is considered.

∴ Time complexity = O(n).

6. What is the need of data structure?

Ans As applications are becoming more complex & the amount of data is increasing day by day, which may cause problems with processing speed searching data, handling multiple request etc. It provides a way of organizing, managing & storing data efficiently. With the help of it, data items can be transversed easily. It provides efficiency, reusability & abstraction. It plays an imp role in enhancing performance of a program because main function of program is to store & retrive the users data as fast as possible