Data Structure

KCS-301

Solution of ST-1

Section - A

Ques 1) How can you represent a sparse matrix in memory?

Sol:- Representing a sparse matrix by a 2D array leads to wastage of lots of memory as zeros in the matrix are of no use in most of the cases. So, instead of storing zeros with non-zero elements, we can only store non-zero elements. This means storing non-zero elements with triples - {Row, column, Value}.

Sparse Matrix Representation can be done in many ways:-

Method 1:- Using Array

- Row - Index of Row, where non-zero element is located.
- Column - Index of Column, where non-zero element is located.
- Value - Value of the non-zero element located at index - (row, column).

$$\begin{bmatrix} 0 & 0 & 3 & 0 & 4 \\ 0 & 0 & 5 & 7 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 2 & 6 & 0 & 0 \end{bmatrix} \Rightarrow$$

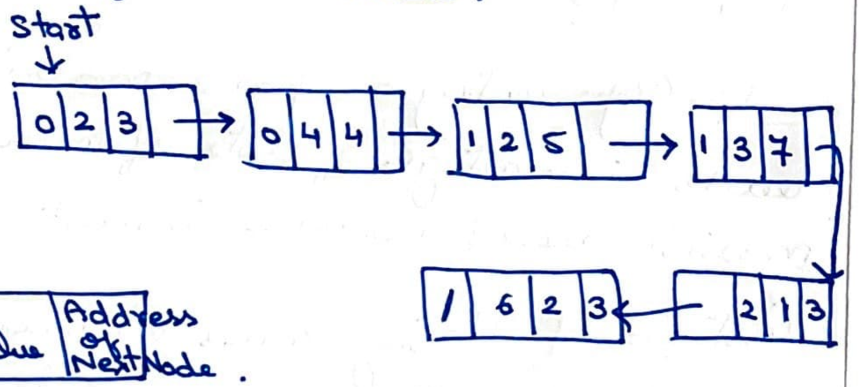
Row	0	0	1	1	3	3
Column	2	4	2	3	1	2
Value	3	4	5	7	2	6

Method 2: Using Linked Lists

In linked list, each node has four fields. These four fields are defined as :-

- Row: Index of Row, where non-zero element is located.
- Column: Index of column, where non-zero element is located.
- Value: Value of the non-zero element located at Index - (Row, column).
- Next Node :- Address of the next node.

$$\begin{bmatrix} 0 & 0 & 3 & 0 & 4 \\ 0 & 0 & 5 & 7 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 2 & 6 & 0 & 0 \end{bmatrix} \Rightarrow$$



Row	Column	Value	Address of Next Node

Ques2) Explain Abstract Data Types?

Solution:- Abstract Data Type (ADT) is a type (or class) for objects whose behavior is defined by a set of value & set of operations.

The definition of ADT only mentions what operations are to be perfomed but not how these operations will be implemented. It does not specify how data will be organised in memory & what algorithms will be used for implementing the operations. It is called 'abstract' because it gives an implementation-independent view. The process of providing only the essentials & hiding the details is known as abstraction.

the users of data Type doesn't need to know how that data Type is implemented.

For Eg:- Using primitive values like int, float, char data types.

① Stack ADT:- Having following operations like:-

push ( ) → Insert an element at one end called top.

pop ( ) → Remove & return the Element at the top of Stack, if not empty.

peek( ) → Return the Element at the top of stack without removing it, if the stack is not Empty

size ( ) → Return the no. of Elements

isEmpty ( ) → Return true if stack is Empty, otherwise false.

isfull ( ) → Return true if stack is full, otherwise return false.

etc.

Ques 3) Explain the use of Calloc ( ) and realloc ( ) functions with example?

Solution:- CALLOC ( ):-

"Calloc" or 'Contiguous allocation' method in C is used to dynamically allocate the specified number of blocks of memory of the specified type. It is very much similar to malloc( ) but has different points and these are:-

It initializes each block with a default Value '0'.

It has two parameters or arguments as compare to malloc( ).

ptr = (float*) Calloc (25, size of (float));

This statement allocates contiguous space in memory for 25 elements each with the size of the float.

If space is insufficient, allocation fails & returns a NULL pointer.

REALLOC()

"Realloc" or "Re-allocation" method in C is used to dynamically change the memory allocation of a previously allocated memory. In other words, if the memory previously allocated with the help of malloc or calloc is insufficient, realloc can be used to dynamically re-allocate memory. Re-allocation of memory maintains the already present value and new blocks will be initialized with the default garbage value.

Syntax:-

```
newptr = realloc (old pointer, new size);
```

Section-B

Ques 4) Discuss the representation of polynomial using Linked List. Write 'C' function to add two such polynomials represented by linked List.

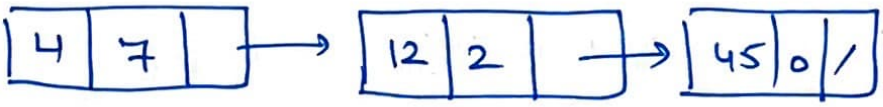
Solution:-

In the polynomial LL, the coefficients and exponents of the polynomial are defined as the data node of the list.

For adding two polynomials that are stored as a LL. We need to add the coefficients of variables with the same power. In a LL node contains 3 members, coefficient value link to the next node .

A LL that is used to store polynomial looks like :-

Polynomial :- $4x^7 + 12x^2 + 45$



Addition of polynomial using Singly LL :-

```
Struct node
    {
     int coeff;
     int exp;
    Struct node * next;
  }
Struct node *first, *second;
Struct node * third;
Addition_of_poly ( first, second, third, expo, coef)
  {
    While ((first != null) && (Second != Null))
     {
      If (first → exp == second == exp)
        {
         expo_t  = first_exp ;
         coeff_t = (first → coeff) + (Second →
                                            coeff);
         first = first → next;
          Second = Second → next;
       Add_node (third, expo_t , coef_t);
   }
```

```
Else if (first → expo > second → expo)
     {
       expo_t = first → expo;
       Coeff_t = first → coeff
        first = first → next ;
   Add_node (third , exp_t , coef_t);
    }
Else
     {
      expo_t = second → expo ;
      Coef_t = second → expo ;
      second = second_next ;
   Add_Node (third , expo_t , coef_t);
     }
}
 While (first ! = null)
   {
      expo_t = first → expo;
      Coeff_t = first → coef;
       first = first → next ;
      Add_node (third , expo_t , coef_t);
  }
  While ( Second ! = null )
     {
          expo_t = Second → expo;
        Coef-t = Second → coef ;
        Second = Second → next ;
      Add - node (third, expo-t , coef-t);
    }
  }
```

Ques5) What do you understand by time space trade off? Define the various asymptotic notations.

Solution:-

Time - Space Trade off:-

It is a way of solving problem or calculation in less time by using a more storage space (or memory), or by solving a problem in very little space by sending a long time.

It is a case where an algorithm or program trades increased space usage with decreased time. Here, space refers to the data storage consumed in performing a given task (RAM, HDD, etc) & time refers to the time consumed in performing a given task (computation time or response time).

The commonly used asymptotic notations used for calculating the running time complexity of an algorithm is given below :-

① Big oh Notation (O)

② Omega Notation (Ω)

③ Theta Notation (Θ)

Big oh Notation (O)

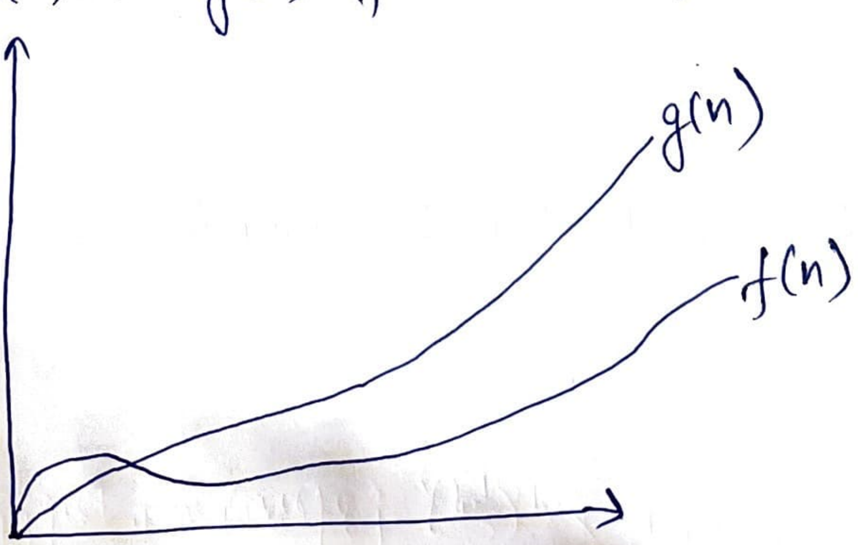
This notation provides an upper bound on a function which ensures that the function never grows faster than the upper bound.

It measures the worst case of time complexity or the algorithm's longest amount of time to complete its operation.

If $f(n)$ & $g(n)$ are the two functions defined for the integers.

then $f(n) = O(g(n))$ as $f(n)$ is big oh of $g(n)$ or $f(n)$ is on the order of $g(n)$ if there exists constant c & n_0 such that :-

$$f(n) \leq c \cdot g(n) \text{ for all } n \geq n_0 .$$

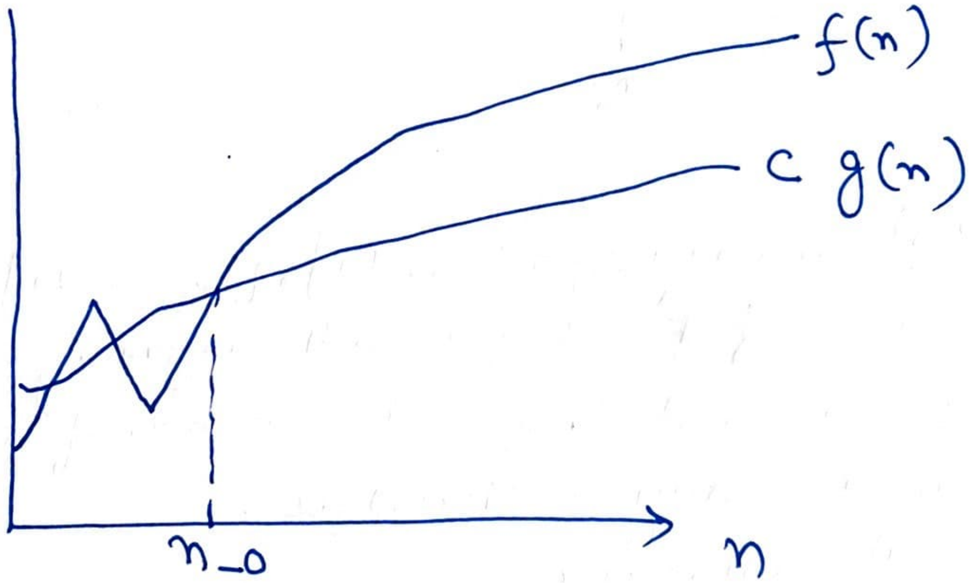


Omega Notation (Ω)

It is the formal way to represent the lower bound of an algorithm's running time. It determines what is the fastest time that an algorithm can run.

If $f(n)$ & $g(n)$ are the two functions defined for +ve integers,

then $f(n) = \Omega(g(n))$ as $f(n)$ is omega of $g(n)$ $f(n)$ is on the order of $g(n)$ if there exists constants & n_0 such that :-



Theta Notation (θ)

The theta Notation mainly describes the average case scenarios.

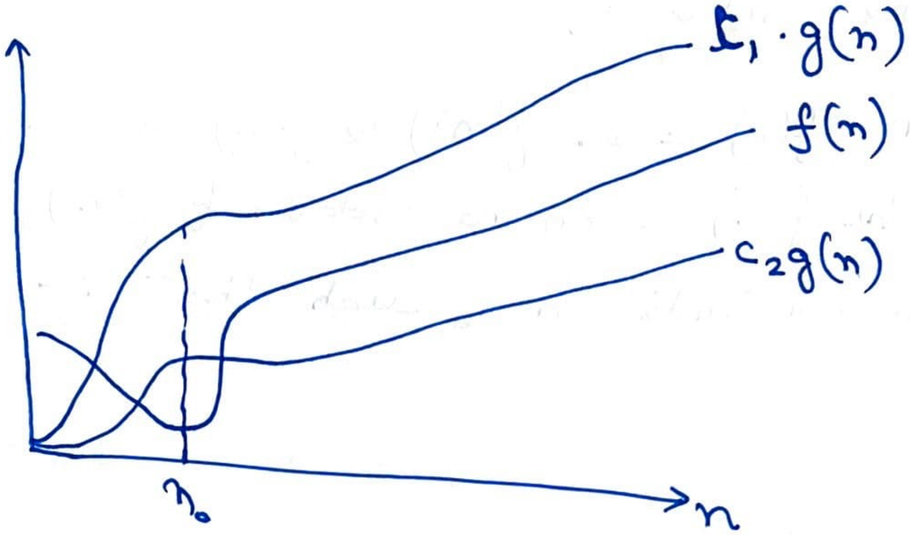
It represents the realistic time complexity of an algorithm.

Let $f(n)$ & $g(n)$ be the functions of n where n is the steps required to execute the program then:

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

The above condition is satisfied only if when

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



Ques 6) An array X[-15.... 10], [15 .... 40] required one byte of the storage & the beginning location is 1500. So determine the location of X[15][20], when the array is stored as.

① Row major

② Column major.

Solution:-

① Row major:-

$$A[i][j] = BA + w[(i-LB_1) \times C + (j-LB_2)]$$

Not Exist, because X[15] as a row not Exist.

② Column Major:-

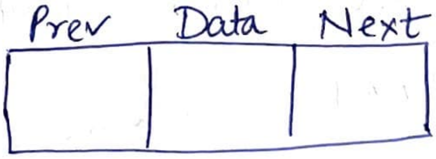
$$A[i][j] = BA + w[(j-LB_2) \times R + (i-LB_1)]$$

Not exist, because X[15] as a row not exist.

Q7- What is doubly linked list? What are its application? Write an algorithm how to insert an element at the end of doubly linked list.

Ans A doubly linked list is a 'Two-way linked list' in which a node contains a pointer to the previous as well as the next node in the sequence.

A node in doubly linked list consists of three parts :- node data, pointer to the next node, and pointer to the previous node.



NODE

The C implementation of a node in doubly linked list :

```
struct NODE
{
  int data;
  struct NODE *next;
  struct NODE *prev;
}
```

Application of doubly linked list:-

1) Implementation of Stacks and queues.

2) Redo and undo functionality

3) Use of the Back and forward button in a browser.

4) Most Recently Used also can be represented as a doubly linked list.

Algorithm: Insert the node at the end.

```
1.  If AVAIL = NULL then
        Print 'Overflow' and goto step 11.
2.  Set New_NODE = AVAIL
3.  Set AVAIL = AVAIL -> NEXT
4.  Set NEW_NODE -> DATA = VAL
5.  Set NEW_NODE -> Next = NULL
6.  Set ptr = START
7.  Repeat step 8 while ptr -> next != NULL
8.      Set ptr = ptr -> Next.
9.  Set ptr -> next = new_node
10. Set New_node => prev = ptr
11. Exit
```