

B.Tech.

Third Semester Examination.

Data Structures Using 'C' (CSE-201-F)

Note : Attempt any *FIVE* questions.

Q. 1. (a) What do you mean by the term Data Structure? Discuss the basic operations of data structures with examples.

Ans. Data structure is representation of the logical relationship existing between individual elements of data.

In other words, a data structure is a way of organising all data items that considers not only the elements stored but also their relationship to each other.

We also define it as a mathematical or logical model of particular organisational of data items.

Data structures mainly specifies the following four things :

- (i) Organisation of data
- (ii) Accessing methods.
- (iii) Degree of association
- (iv) Processing alternatives for information.

Data structures are building blocks of a program. And hence the selection of a particular data structure stresses on the following two things :

- (i) The data structures must be rich enough in structure to reflect the relationship existing between the data.
- (ii) And the structure should be simple so that we can process data effectively whenever required.

The identification of the increment data structure is vital in nature. And the structure of input and output data can be used to derive the structure of a program. Data structure affects the designs of both structural and functional aspects of a program.

Algorithm + Data structure = Program

Q. 1. (b) What is a two dimensional array? How it is stored in memory? Differentiate it with structure.

Ans. A two dimensional array is a list of a finite number, say $m \times n$ of homogeneous data element such that :

The element of the array are referenced by two index sets consisting of m and n consecutive integer numbers.

The element of the array are stored in consecutive memory location.

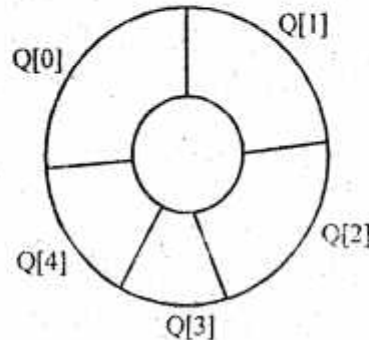
Sale					
	0	1	2	3	4
0	2000	1200	1500	1000	1005
1	1500	1450	2010	1550	1250
2	1000	1250	1400	2000	3000
3	1200	1275	1575	3500	1750
4	5200	4000	1000	1200	1575

Two Dimensional Array Sale for Storing Sales

Q. 2. (a) What is circular queue? Discuss its advantages over linear queue?

Ans. A circular queue is one in which the insertion of a new element is done at the very first location of the queue if the last location of the queue is full. In other words if we have a queue Q of say n elements then after inserting an element last (i.e., in the $n-1^{\text{th}}$) location of the array the next element will be inserted at the very first location (i.e., location with subscript 0) of the array. It is possible to insert new elements, if and only if those locations (slots) are empty. We can say that a circular queue is one in which the first element comes just after the last element. It can be viewed as a mesh or loop of wire, in which the two ends of wire are connected together.

The following figure shows a empty circular queue Q[5] which can accommodate five elements.



A circular queue overcomes the problem of initialized space in linear queues implemented as arrays. A circular queue also has a front and Rear to keep the track of the elements to be deleted and inserted and therefore to maintain the unique characteristics of the queue. The following assumptions are made.

- (i) Front will always be pointing to the first element (as in the linear queue).
- (ii) If front = Rear the queue will be empty.
- (iii) Each time a new element is inserted into the queue the Rear is incremented by one.

$$\text{Rear} = \text{Rear} + 1$$

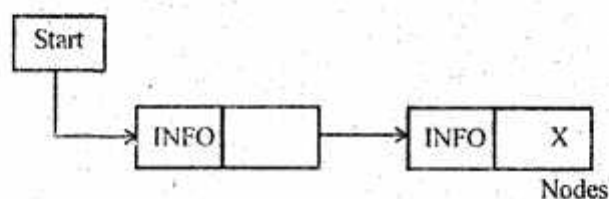
- 4. Each time an element is deleted from the queue the value of front is incremented by one.

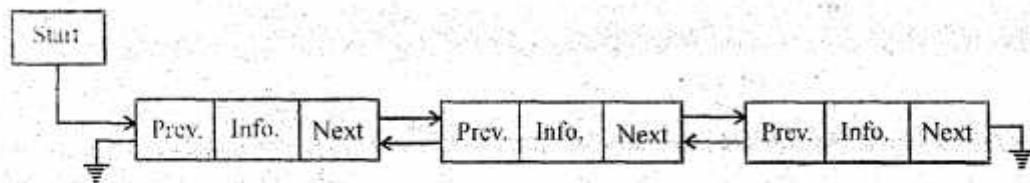
$$\text{Front} = \text{Front} + 1.$$

Q. 2. (b) What do you understand by Dynamic memory management? Create a circular link list using dynamic memory and show its advantages over linear linked list.

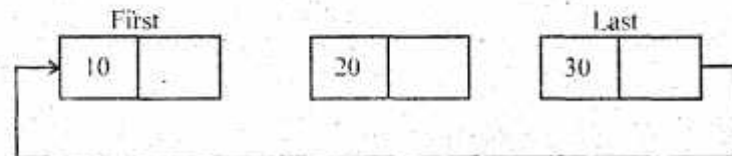
Ans. Linked lists are dynamic data structures. That is, they can grow or shrink, during the execution of a program.

The logical ordering is represented by having each element pointing to the next element. Each has two parts: INFO part which stores the information and POINTER which points to the next element.





A circular linked list is just a singly linked list in which the link field of the last node contains the address of the first node of the list that is the link field of the last node does not point to NULL rather it points back to the beginning of the linked list.



In circular linked list :

- (i) Nodes can be accessed easily.
- (ii) Deletion of node is easier.
- (iii) Concatenation and splitting of circular linked list are more efficient.

Q. 3. (a) What do you mean by Stack? Define with the help of algorithms for its operations e.g., PUSH and POP. Describe a method to convert an infix expression into a postfix expression with the help of example.

Ans. A stack is a non-primitive linear data structure. It is an ordered list in which addition of new data items and deletion of already existing data item is above from only one end, known as top of stack (TOS).

Operations On Stack : The basic operations that can be performed on stack are as follows :

(i) **PUSH :** The process of adding a new element to the top of stack is called PUSH operation. Pushing an element in the stack invoke adding of element, as the new element will be inserted at the top after every push operation the top is incremented by one. In case the array is full and no new element can be accommodated it is called STACK-FULL condition. This condition is called STACK OVERFLOW.

(ii) **POP :** The process of deleting an element from the to top of the stack is called POP operation. After every pop operation the stack is decremented by one. If there is no element on the stack and the pop is performed then this will result into stack under flow condition.

Algorithms for PUSH and POP :

- (i) Algorithm for Inverting an item into the stack (PUSH).
Let stack [Max size] is an array for implementing the stack.
 - (i) [Check for stack overflow?]
If $\text{top} = \text{max size} - 1$, then print : overflow and exit.
 - (ii) Set $\text{TOP} = \text{TOP} + 1$ [Increase TOP by 1]
 - (iii) Set $\text{stack}[\text{TOP}] = \text{ITEM}$ [Inserts Item in new TOP Position]
 - (iv) Exit.
- (ii) Algorithm for Deleting an Element from the Stack (POP) :
 - (i) [Check for the stack underflow]
If $\text{Top} < 0$, then
Print "stack underflow" and exit.
Else [Remove the Top element]

- Set item = Stack [Top]
(ii) Decrement the stack top.
Set Top = Top - 1
(iii) Return the deleted item from the stack.
(iv) Exit.

Converting Infix Expression to Postfix Form :

$A + B * C$	Infix form
$A + (B * C)$	Paranthesized expression
$A + (BC * C)$	Convert the multiplication
$A + (BC*) +$	Convert the addition
$ABC * +$	Postfix form

The rules to be remembered during infix to postfix conversion are :

- Paranthesize the expression starting from left to right.
- During paranthesizing the expression, the operands associated with operator having higher precedence are first paranthesized. For example in above expression $B * C$ is paranthesized first before $A+B$.
- The sub-expression (part of expression) which has been converted into postfix is to be treated as single operand.
- Once the expression is converted to postfix form remove the parenthesis.

Q. 3. (b) What do you mean by searching? Discuss and compare various search methods.

Ans. Searching is used to find the location where element is available or not. There are two types of searching techniques as given below :

- Linear or sequential searching
- Binary searching

Linear or Sequential Searching : In linear search, we access each element of an array one by one sequentially and see whether it is desired element or not. A search will be unsuccessful if all the elements are accessed and the desired element is not found. In the worst case, the number of average case we may have to scan is half of the size of the array ($n/2$). Therefore linear search can be defined as the technique which traverses the array sequentially to locate the given item.

Binary Searching : Binary search is an extremely efficient algorithm. This search technique searches the given item in minimum possible comparisons. To do this binary search, first we had to sort the array elements. The logic behind this technique is given below :

- First find the middle element of the array
- Compare the mid element with an item
- There are three cases :
 - If it is a desired element then search is successful.
 - If it is less than desired item then search only the first half of the array.
 - If it is greater than the desired element search in the second half of the array.

Q. 4. (a) Write an algorithm of complexity $O(n)$ to find the k^{th} smallest element in an array num [n], where n and k are given as input.

- Ans.**
- [Initialize the value of i] set $i = \text{len}$
 - Repeat for $i = \text{len}$ down to pos.
[Shift the element down by 1 position]
Set $a[i+1] = a[i]$
[End of loop]

- (iii) [Insert the element at required position]
Set a [pos] = num
- (iv) [Reset ln] set ln = ln + 1
- (v) Display the new list of arrays
- (vi) End.

Q. 4. (b) What is meant by height balance tree? Show insert and delete operation on it.

Ans. A tree is height balanced if for each node in the tree, the height of the left subtree differs from the height of the right subtree by no more than 1. The tree is height balanced but it is not completely balanced. On the other hand, the tree is completely balanced tree.

An almost height balanced tree is called an AVL tree after the Russian mathematician G. M. Adelson Velspic and E.M. Lendis, who first defined and studied this form of a tree.

AVL tree may or may not be perfectly balanced.

The number of nodes from level 1 through level $n-1$ will be

$$1 + 2 + 2^2 + 2^3 + \dots + 2^{n-2} = 2^{n-1} - 1$$

Q. 5. Propose a data structure that supports a special operation for finding smallest or largest element in $O(1)$ time. Prove for such data structure that the insert operation must take $O(\log_2 n)$ time.

Ans. Efficiency of Quick Sort : The time analysis of quick sort algorithm is given by the following recurrence relations :

$$T(n) = \begin{cases} a & n = 1 \\ Cn + T(n/2) + T(n/2) \end{cases}$$

Where, $T(n/2) \rightarrow$ Time required to partition the array.

$T(n/2) \rightarrow$ Time required to sort the right subarray.

$T(n/2) \rightarrow$ Time required to sort the left subarray.

Here time required to partition the array is $O(n)$.

Worst-Case Analysis: In this case, on every function call, the given array is partitioned into two subarray. One of them is an array one of them is an empty array, now the timing, analysis taken the form,

$$\begin{aligned} T(n) &= Cn + T(0) + T(n-1) \\ &= Cn + T(n-1) \\ &= Cn + C(n-1) + T(n-2) \\ &= Cn + C(n-1) + C(n-2) + T(n-3) \\ &\vdots \\ &= Cn + C(n-1) + C(n-2) \\ &\quad + \dots + C(1) + T(0) \\ &= C[n + (n-1) + (n-2) + \dots + 2 + 1] \\ &= C\left(\frac{n(n+1)}{2}\right) \end{aligned}$$

$$= \frac{Cn^2}{2} + \frac{Cn}{2}$$

$$= O(n^2)$$

Best Case Analysis : The best case time analysis is possible when the array is always partitioned in half. The timing analysis takes the following form :

$$T(n) = Cn + T(n/2) + T(n/2)$$

$$= Cn + 2T(n/2)$$

Let n is a power of 2 i.e.,

$$n = 2^k$$

Thus,

$$T(2^k) = 2T(2^{k/2}) + C(2^k)$$

$$= 2T(2^{k-1}) + C2^k$$

$$= 2[2T(2^{k-2}) + C2^{k-1} + C2^k]$$

$$= 2^2 T(2^{k-2}) + 2C2^k$$

$$= 2^2 [2T(2^{k-3}) + C2^{k-2}] + 2C2^k$$

$$= 2^3 T(2^{k-3}) + 2C2^k$$

$$\vdots$$

$$= 2^k T(2^{k-k}) + kc2^k$$

$$= 2^k T(1) + KCn$$

Q. 6. Explain any two of the following with their run time complexities and apply them to some the following list of 10 elements in increasing order :

[30, 25, 17, 56, 99, 80, 54, 14, 66, 94] Merge sort

Ans. Merge Sort : 30, 25, 17, 56, 99, 80, 54, 14, 66, 94.

25	30
----	----

17	56
----	----

80	99
----	----

14	54
----	----

66	94
----	----

17	25	30	56
----	----	----	----

14	54	80	99
----	----	----	----

66	94
----	----

17	25	30	56	14	54	66	80
----	----	----	----	----	----	----	----

94	99
----	----

14	17	25	30	54	56	66	80	94	99
----	----	----	----	----	----	----	----	----	----

Q. 7. (a) What is graph? How it can be stored in memory? Explain BFT and DFT with the help of suitable examples.

Ans. A graph G consists of a set V of vertices (nodes) and a set E of edges (arcs). We write $G = (V, E)$, V is a finite and non empty set of vertices. E is a set of pairs of vertices, their pairs are called edges. Therefore,

$V(G)$, read as V of G , is set of vertices.

And $E(G)$, read as E of G , is set of edges.

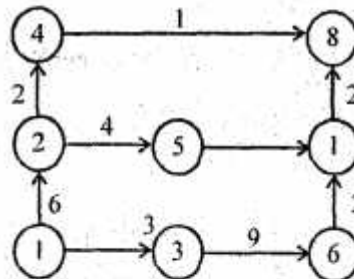
In memory, it can be stored as Adjacent matrix adjoining list representation and multi-list representation. There are two graph traversal methods :

- (i) Breadth First Search (BFS)
- (ii) Depth First Search (DFS)

In breadth first traversal of a graph are mode is selected as the start position. It is visited and marked, then all unvisited nodes adjacent to the next node are visited and marked in some sequential order. Finally, the universal nodes immediately adjacent to these nodes are visited and marked and so forth, until the entire graph has been transversed.

Breadth first traversal of the graph results in visiting the nodes in the following order : 1, 2, 3, 4, 5, 6, 7, 8.

The sequence 1, 3, 2, 6, 5, 4, 7, 8 is also valid, breadth first transversal visitation order.



Depth first traversal follows first a path from the starting node to an ending node, then another path from the start to the end, and so forth until all nodes have been visited.

Depth first traversal of the graph results in visiting the node in the following order 1, 2, 4, 8, 5, 7, 3, 6. A path is Perseid until no unvisited nodes remain reachable, then the algorithm backs up to the last node that was visited that has unvisited adjacent node. An equally valid sequence to results from depth-first traversal of the graph is 1, 3, 6, 7, 8, 2, 5, 4.

Q. 7. (b) What is hashing? What is the condition for collision? How collision can be resolved?

Ans. There is an approach in which we compute the location of the desired record in order to retrieve it in

a single access. This avoids the unnecessary comparisons. In this method, the location (or offset address) of the desired record present in the search table depends only on the given key but not on other keys.

A hash function h is simply a mathematical formula that manipulates the key in some form to compute the index for this key in the hash table.

A collision is a phenomenon that occurs when more than one key maps to same slot in the hash table. Though we can keep collisions to a certain minimum level. But we cannot eliminate them altogether. Therefore we need some mechanism to handle them.

By Chaining Method : In this scheme all the elements who keys hash to the service hash-table slot are put in as linked-list.

Open Addressing Method : In open addressing method scheme all the elements of the dynamic set are stored in the hash table itself.

In order to insert a key in a hash table under this scheme; if the slot to which key is hashed is free then the element is stored at that slot. In case the slot is filled, then other slots are examined systematically in the forward direction to free the free slot.

Q. 8. Write short notes on any two of the following :

(i) Threaded Binary Trees

(ii) Asymptotic notations (Big O, Θ etc.)

(iii) Priority Queues.

Ans. (i) Threaded Binary Trees : Our possible way to utilize the space is that we can store special pointer that point to nodes higher in the tree, i.e., ancestors. These special pointers are called threads and the binary tree having such pointers are called threaded binary tree.

(ii) Asymptotic notations (Big O, Θ etc.) : $F(n) = O(g(n))$ if there exists positive constants C and n_0 such that $f(x) < cg(x)$ for all $n, n > n_0$, where f and g are non-negative functions.

We write $O(1)$ to mean a computing time that is constant.

→ $O(n)$ is called linear.

→ $O(n^2)$ is called quadratic.

→ $O(n^n)$ is exponential.

(iii) Priority Queues : A priority queue is a kind of queue in which each element is assigned a priority and the order in which element are deleted and processed comes from the following rules :

(a) An element with highest priority is processed first before any element of lower priority.

(b) Two or more elements with the same priority are processed according to the order in which they are added to the queue.