Dec 2014.

Roll No. | | | | | | | | | | | | |

Total No. of Pages: 02
Total No. of Questions: 09

B.TECH(3D ANIMATION & GRAPHICS,CSE, IT) (Sem.-3rd)

DATA STRUCTURE

Subject Code: BTCS-304
Paper ID: [A1126]

Time: 3 Hrs. **Max. Marks: 60**

INSTRUCTIONS TO CANDIDATES:

1. ***Section –A, is Compulsory.***
2. ***Attempt any four questions from Section-B.***
3. ***Attempt any two questions from Section-C.***

Section –A (10x2=20)

Q.1.

(a) What is meant by an abstract data type?
(b) Differentiate between Stack and Queue?
(c) What are the advantages in the array implementation of list?
(d) What is the usage of stack in recursive algorithm implementation?
(e) What is a circular queue and its use?
(f) What condition is checked to determine if pointer, P has moved past the end of the list?
(g) Evaluate: (a) + 1 24 3 + * 41 - (b) 25 7 * 14 – 6 +
(h) Define the term sparse matrix. How they are stored in memory.
(i) List out the different types of hashing functions?
(j) What is meant by strongly connected in a graph?

Section –B (4x5=20)

Q. 2. Define the terms: static and dynamic data structures. List some of the static and dynamic data structure in C.

Q. 3. What is traversing? Write an algorithm for traversing a link list?

Q. 4. What Criteria is used for evaluating the suitability of a particular data structure for a given application.

Q. 5. Make a binary search tree and a heap tree from the given data.
23 7 92 6 12 14 40 44 20 21

Q. 6. What is Graph. Describe in brief the various methods used to represent Graphs in memory.

Section –C (2x10=20)

Q. 7. What is the advantage and average efficiency of quick sort? Apply Quick sort on the following data and show the contents of the array every pass:
48 7 26 44 13 23 98 57 100 5 32

Q. 8. Write the algorithms for the following:

(a) Deleting an element from a doubly link list.

(b) Inserting an element in a priority queue.

(c) To reverse a string of characters using stack.

(d) To search an element in a sorted array.

Q. 9. Define AVL and B-trees and their applications? Explain various operations used for balancing a binary tree with the help of a suitable example?

----END----

Section- A —

2014.

What is meant by an abstract data type?

Ans Abstract data type (ADT) are a way of classifying data structures based on how they are used and the behaviours they provide. They do not specify how the data structure must be implemented but simply provide a minimal expected interface and set of behaviours.

(b) Differentiate between Stack and Queue.

Stack	Queue
* → Stack is an abstract data type represented by a physical stack of entities where insertion and deletion take place at the same end	* → Queue is also an abstract data type similar to stack expect it is open at both the end.
* → It is based on working principal of LIFO meaning Last In First Out.	* → It is based on the working principle of FIFO meaning Fist In First Out.
* → object can be added in the stack one at a time meaning the object can be removed which is added to list.	* → They object which is added first can only be removed from the Queue.
* → Push adds items to the stacks and pop removes recently added items from the stack.	* → Enqueue adds item to the rear and dequeue removes item from the front of the Queue.

(C) What are the advantages in the array implementation list?

Ans (i) Binary search can be done with array implementation as middle element can be accessed.

(ii) Elements can be randomly accessed and processed through array implementation.

(d) What is the usage of stack in recursive algorithm implementation?

Ans Stack is a LIFO (Last In First Out) data structure. and Recursion is a technique of solving any problem by calling same function again and again until some breaking condition where recursion stops and it starts calculating the solution from these on. For eg. Factorial of a given number.

(e) What is a circular Queue and its use?

Ans Circular Queue is a linear data structure in which the operation are performed based on FIFO (First in First Out) principle and the last position is connected back to the first position to make a circle.

→ Memory Management, Traffic System, CPU scheduling can be done by using Circular Queue.

List out the different type of hashing functions?

1) Division method
2) Mid Square method
3) Digit folding method.

(j) What is meant by strongly connected in a graph?

Ans Strongly connected in a graph means every element is connected with each other means every vertex is reachable from every other vertex.

(f) What Condition is checked to determine if po has moved past the end of the list?

Ans NULL condition is checked to determine that poi 'p' has moved past the end of the list.

→ if (P == NULL)

(g) Evaluate :

(a) * 1 2 2 4 3 + * 4 1 -

~~((1+24)+3)*~~

((1+24)*3)-41

→ 34

b) 2 5 7 * 14 - 6 +

((25 * 7) - 14) + 6

⇒ 167

(h) Define the term sprase matrix. How they are stored in memory.

Ans A matrix is a 2-dimensional data and a sprase matrix is that matrix whose most of the element of the matrix have 0 value. It is stored as a two-dimensional matrix in the memory.

2014 Paper

DATE ____
PAGE ____

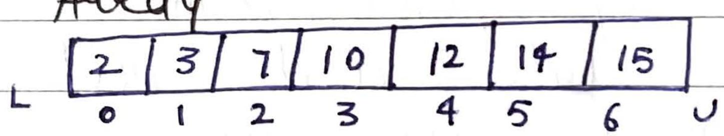
SECTION-B

Define the terms: Static and dynamic data structure. List some of the static and dynamic data structure in C.

Data structure is a way of storing and organising data efficiently such that the required operation on them can be performed be efficient with respect to time as well as memory

Static data structure: In static data structure the size of the structure is fixed. The content of the data structure can be modified but without changing the memory space allocated to it.

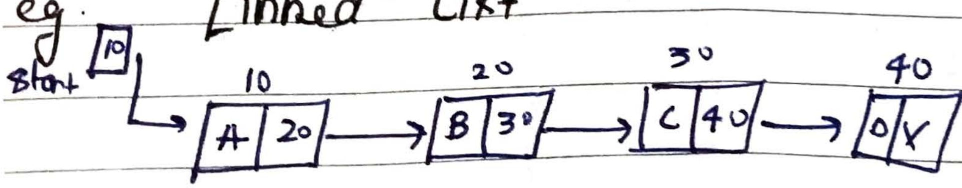
eg. Array



N = 7
L = 0
U = 6

Dynamic data structure: In dynamic data structure the size of the structure is not fixed and can be modified during the operations performed on it. Dynamic data structure are designed to facillitate change of data structures in the run time

eg. Linked List



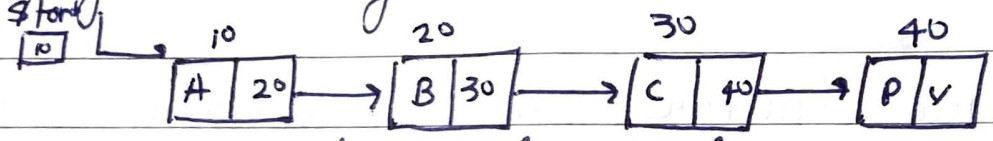
DATE
PAGE

Make the

What is traversing? Write an algorithm for traversing a link list?

A data structure contains elements, which contain data.

Traversing a data structure means: "visiting" or "touching" the elements of the structure, and doing something with the data.



Now, we have to traverse in this singly linked list.

ALGO: Traverse (start, ptr, data, next)
{ start is starting pointer, point to first node contain data and next pointer
ptr: is traverse all node }

Step 1. Start == NULL
Write "Underflow"
Exit

Step 2 ptr = start

Step 3 Repeat while ptr != NULL

Step 4 process ptr → data

Step 5 ptr = ptr → next

Step 6. Exit

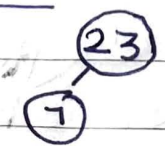
DATE
PAGE

Ques. Make a binary search tree and a heap tree from the given data

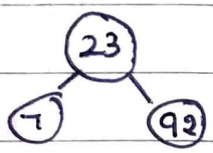
23, 7, 92, 6, 12, 14, 40, 44, 20, 21

Ans. Binary Search tree

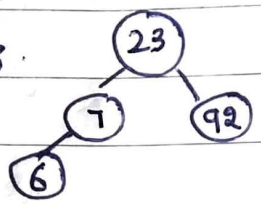
1.



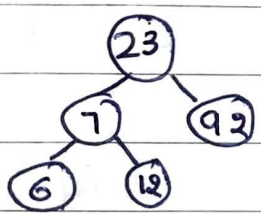
2.



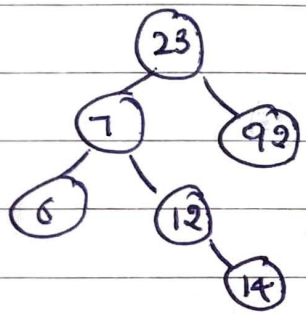
3.



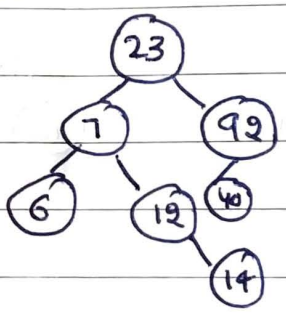
4.



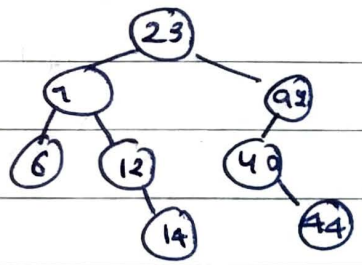
5.



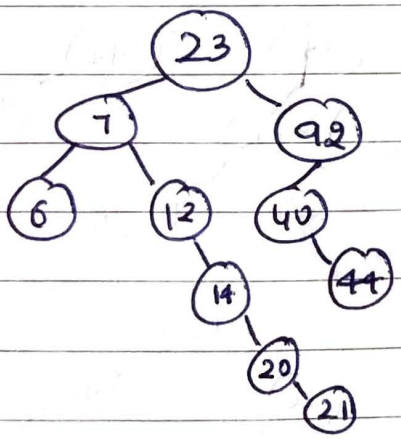
6.



7.



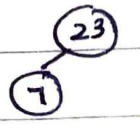
8.



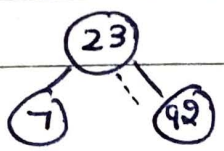
- Heap tree

1.

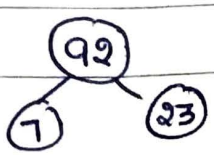
2.

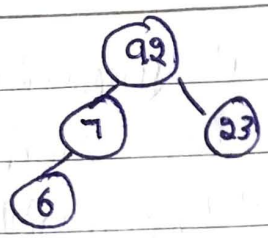


3.

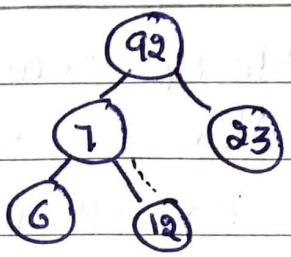


=>

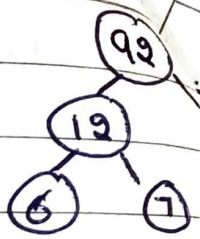


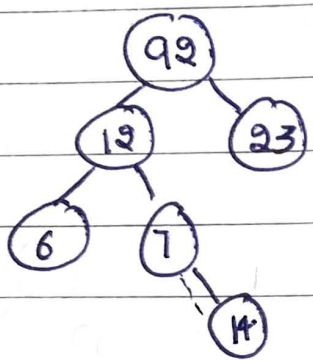

92
7
23
6

5.

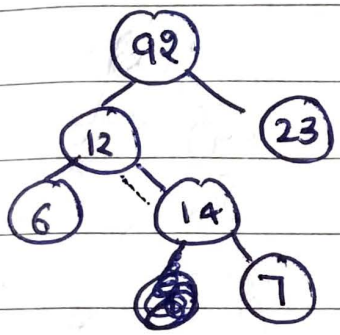

92
7
23
6
12

=>

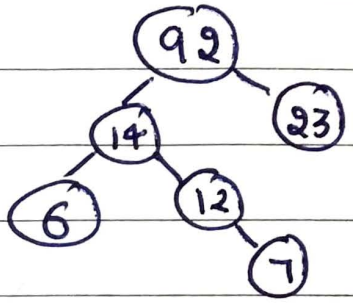

92
12
6
7

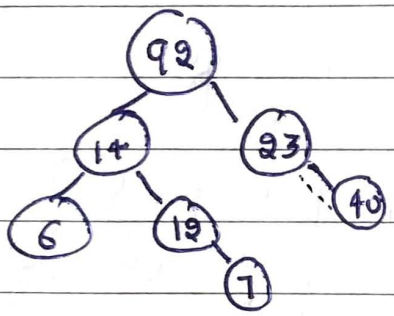

92
12
23
6
7
14

=>

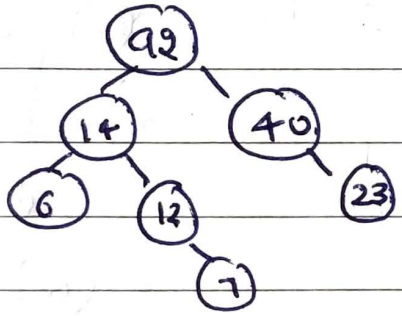

92
12
23
6
14
7

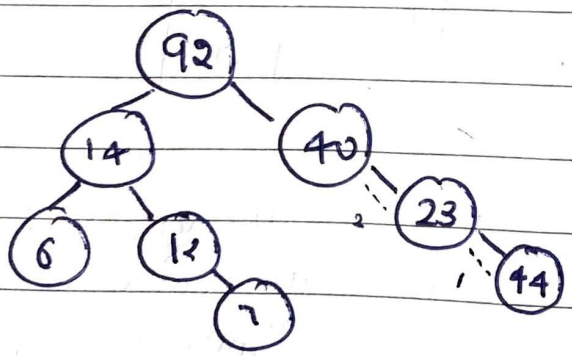
=>


92
14
23
6
12
7

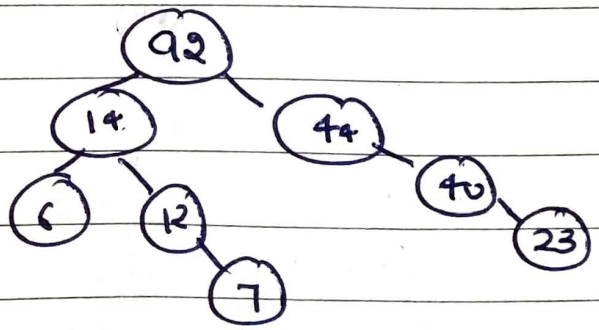

92
14
23
6
12
40
7

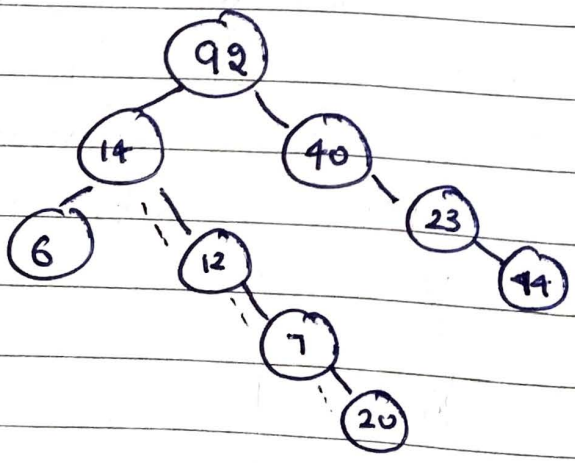
=>


92
14
40
6
12
23
7

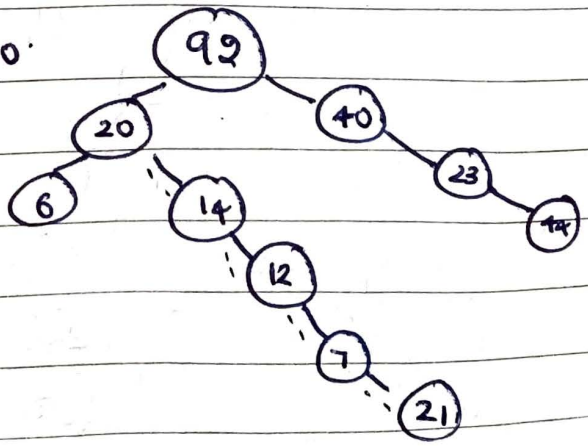

92
14
40
6
12
23
7
44

=>

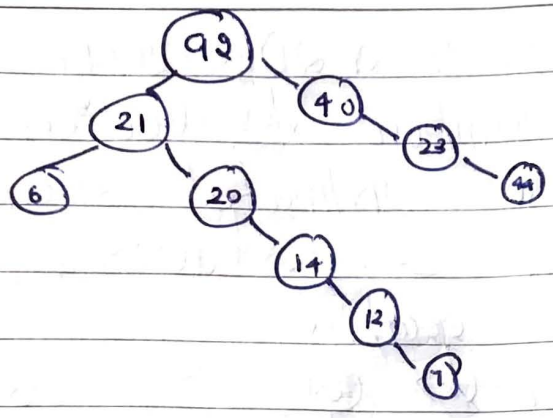

92
14
44
6
12
40
7
23


92
14
40
6
12
23
7
44
20

=> 10.


92
20
40
6
14
23
12
7
21

DATE
PAGE



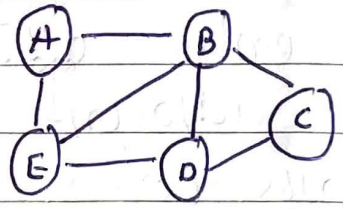
Que6. What is Graph. Describe in brief the various methods used to represent Graph in memory

Ans6. Graph is a data structure that consists of following two components:

1. A finite set of vertices also called as nodes
2. A finite set of ordered pair of the form called as edge.

It is non-linear collection of data structure that used to represent hierarchical relationship.

eg.



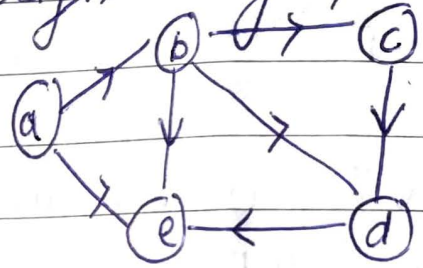
Following two are the most commonly used representations of a graph.

1. Adjacency Matrix
2. Adjacency list.

DATE
PAGE

Adjacency Matrix: It is a 2D array of size where V is the number of vertices in a graph. Adjacency Matrix for undirected graph is always symmetric. It is also used for represent weighted graphs.

eg.

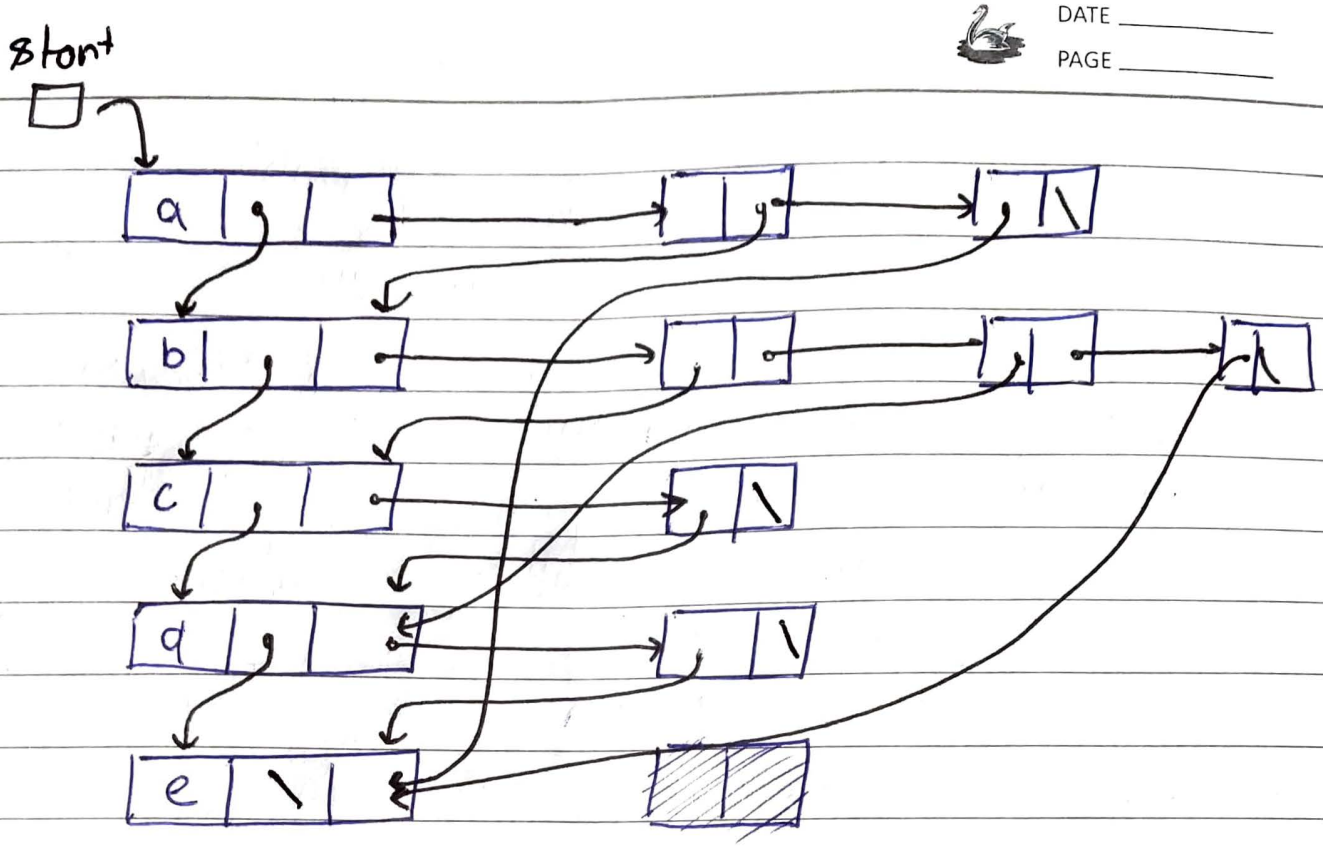


5 Vertices.

	a	b	c	d	e
a	0	1	0	0	1
b	0	0	1	1	1
c	0	0	0	1	0
d	0	0	0	0	1
e	0	0	0	0	0

Adjacency list: An array of list is used Size of array is equal to the no. of vertics Let the array be a[]. An entry a[] represents the list of vertices adjacent to the ith vertex. This representation can also be used to represent a weighted graph.

node list	edge list
a	b, e
b	c, d, e
c	d
d	e
e	x



This is linked representation of Graph.

Section-C (2014)

What is the advantage and average efficiency of Quick Sort? Apply Quick Sort on the following data and show the contents of the array every pass:

48 7 26 44 13 23 98 57 100 5 32

Ans: Quick Sort is a sorting algorithm. It is used on the principle of divide-and-conquer. It is a good general purpose sort and it consumes relatively fewer resources during execution.

Advantages of Quick Sort:

- Its cache performance is higher than other sorting algorithms. This is because of its in-place characteristic.
- It has an extremely short inner loop. Hence, if implemented well, quicksort will be around 2-3 times faster than mergeSort and Heap Sort.
- In quick sort, the array is parted into any ratio. There is no compulsion of dividing the array of elements into equal parts in quick sort.
- No additional storage is required. since, the quick sort algorithm is in-place.

Average Efficiency of Quick Sort:

The time required by the quicksort algorithm for sorting a total of n numbers is represented by the following equation:

$$T(n) = T(K) + T(n-K-1) + (n)$$

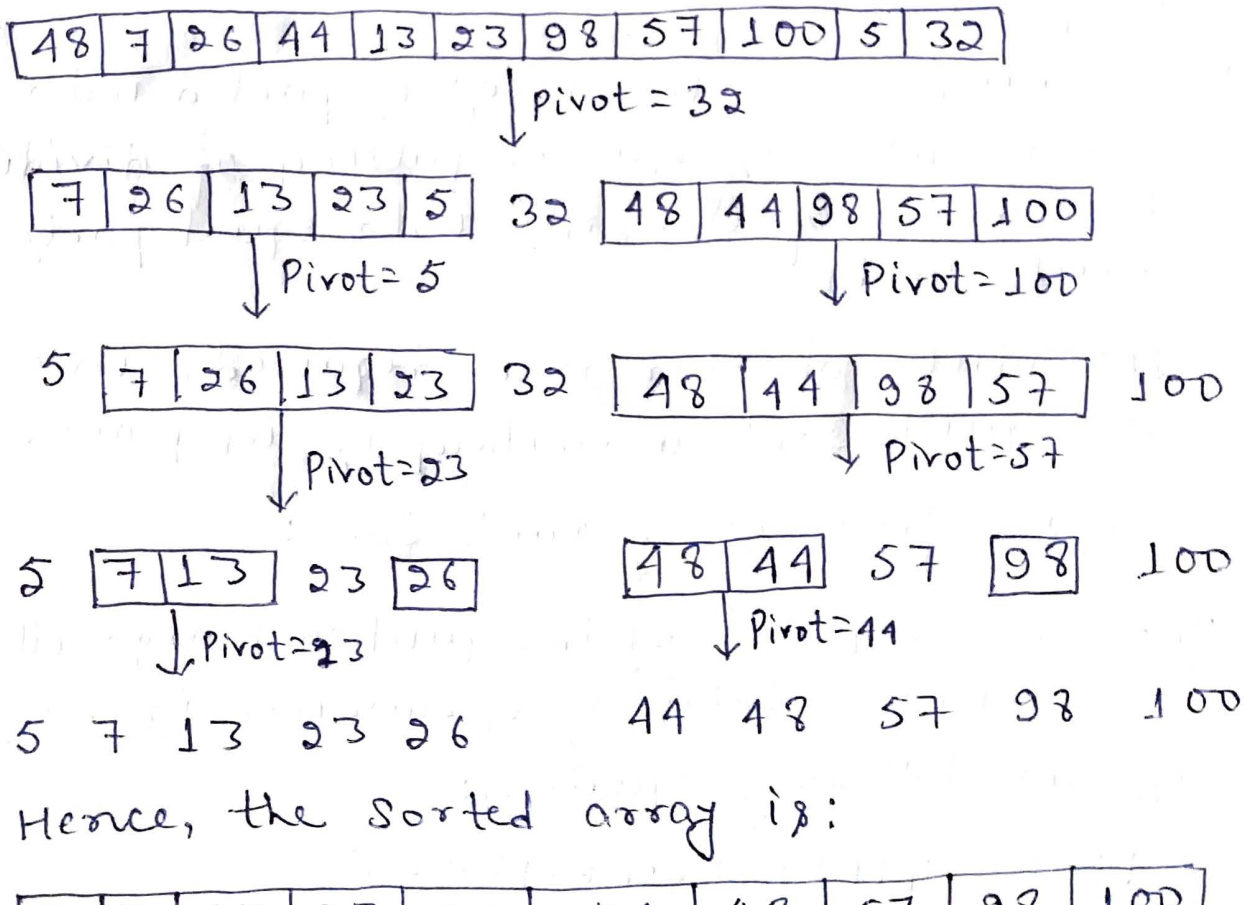
where, $T(k)$ and $T(n-k-1)$ represents the recursive calls in the quicksort algorithm $\theta(n)$ represents the partition process while k is representative of the total count of numbers present in the set that is smaller the the pivot.

Average Case : $T(n) = T(n/9) + T(9n/10) + \theta(n)$

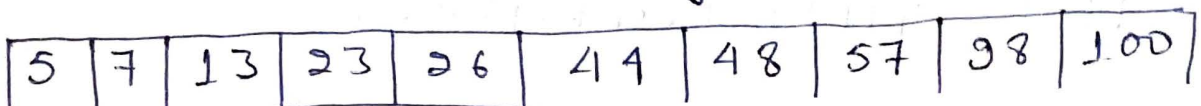
Solving the above mentioned recurrance relation will be $T(n) = (n \log n)$

Applying Quick Sort on the given array :

The given array of elements is partitioned into two arrays one of which holds values smaller than the specified value, say pivot, and the other holds greater values than the pivot. This partitioning calls itself recursively twice to sort the two resulting subarrays.



Hence, the sorted array is:



Write an algorithm for the following:

(a) Deleting an element from a doubly link list.

Following is the algorithm for deletion of an element from a doubly linked list:

DL_DELETE_SPOS( )

Step1: Declare P and TEMP as a pointer of type node, POS and a counter i.

Step2: If START = NULL
a> Print("Underflow")
b> return

Step3: input the POS to delete

Step4: Set P = START

Step5: Repeat for i = 1 to POS-1
Set P = P→NEXT
[End for loop]

Step6: Set TEMP = P→NEXT

Step7: P→NEXT = TEMP→NEXT

Step8: TEMP→NEXT→PREV = P

Step9: free(TEMP) & exit

(b) Inserting an element in a priority queue.

Algorithm:

PUSH(HEAD, DATA, PRIORITY)

Step1. create new node with DATA and PRIORITY

Step 2. Check if HEAD has lower priority

Step 5.

Step 3. NEW → NEXT = HEAD

Step 4. HEAD = NEW

Step 5. Set TEMP to head of list

Step 6. While TEMP → NEXT != NULL
and TEMP → NEXT → PRIORITY >
PRIORITY

Step 7. TEMP = TEMP → NEXT

[End of loop]

Step 8. NEW → NEXT = TEMP → NEXT

Step 9. TEMP → NEXT = NEW

Step 10. Exit

(c) To reverse a string of characters using Stack.

Step 1. Initialize a StringBuffer, this will be for output

Step 2. Initialize a Stack

Step 3. Traverse through string, One character at a time and keep PUSHing it to Stack.

Step 4. While Stack is not empty
POP out the characters from Stack and add it to the output StringBuffer.

Step 5. Print the Output & Exit.

Search an element in a Sorted array.

Binary Search is the most efficient Searching algorithm if the list is sorted. Hence, we will apply Binary Search Here.

Algorithm:

Binary Search (A, LB, UB, key, LOC)

'LB' is lower bound and 'UB' is Upper bound of an array 'A'. The algorithm Searches for 'key' element and return its location through 'loc'.

Step 1. Repeat Step2 while (LB <= UB)

Step 2. Set Mid = Int ((LB+UB) / 2)

Step 3. If A[mid] < key
Set LB = mid+1
else if A[mid] > key
Set UB = mid-1
else
loc = mid

Step 4. Exit

Q9. Define AVL and B-Trees and their applications? Explain various operations used for balancing a binary tree with the help of a Suitable example?

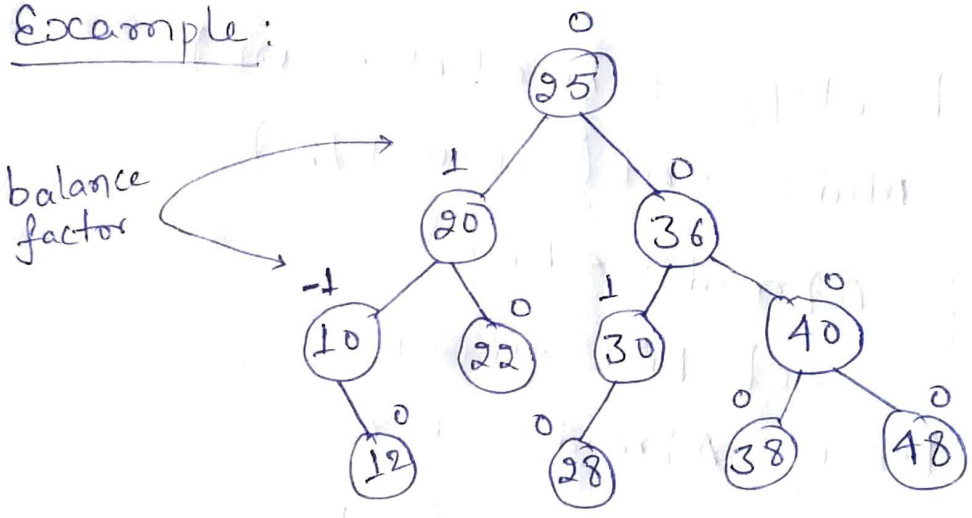
Ans: AVL Tree:

It is a height-balanced binary Search tree. That means, an AVL tree is also a

binary search tree but it is a balanced
A binary tree is said to be balanced if
difference between the heights of left
right subtrees of every node in the
is either −1, 0, or, +1.
Thus, in AVL Tree, every node maintains
an extra information known as balance
factor.

- Balance factor = Height Of left Subtree − Height Of Right Subtree

Example:



- Every AVL Tree is a binary search tree but every Binary search tree need not be AVL Tree.
- Applications of AVL Tree:

→ Directory structure of a file store.
→ Structure of an arithmetic expression.
→ Used in almost every 3D video game to determine what objects need to be rendered.
→ Used in almost every high-bandwidth router for storing router-tables.

B-tree :

B-tree is a tree data structure that keeps data sorted and allows searches, insertions, and deletions in logarithmic amortized time. Unlike self-balancing binary search trees, it is optimized for systems that read & write large block of data.

Important properties:

→ B-tree nodes have many more than two children.

→ A B-tree node may contain more than just a single element.

- **Applications of B-Tree:**

→ It is used to implement indexes

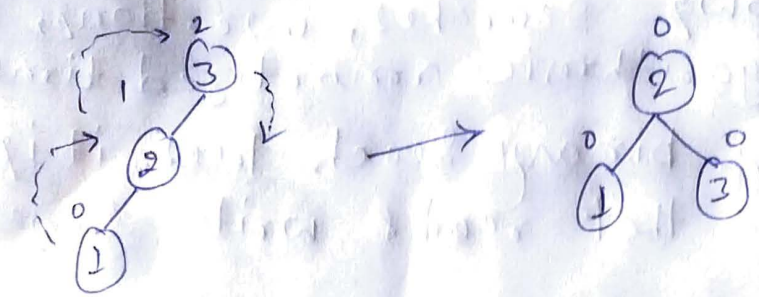
→ The data record pointers in the leaves corresponds to the data record pointers in sequential indexes.

- **Various operations used for Balancing a Binary tree:**

(i) **Single Rotation:**

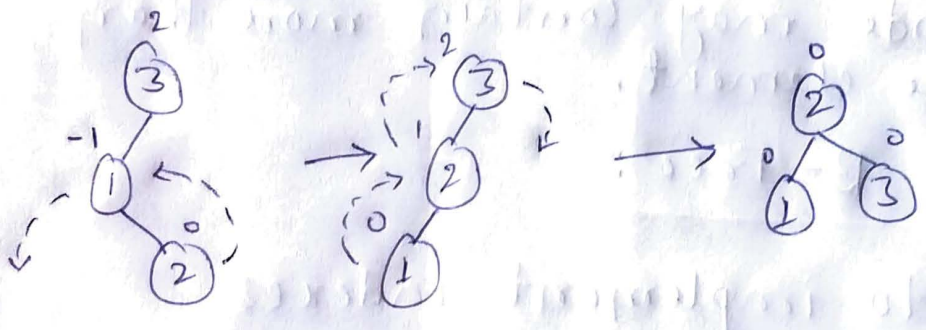
(a) **Left Rotation:**

(b) Right Rotation:

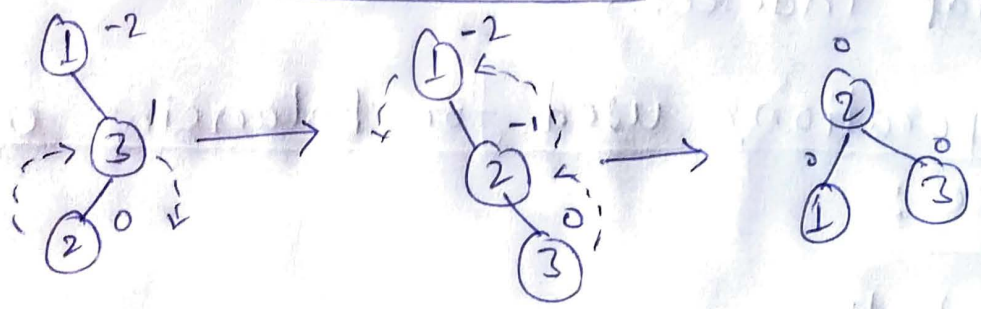


(ii) Double Rotation:

(c) Left - Right Rotation:



(d) Right - left Rotation:



May 2015.

Roll No. | | | | | | | | | | | |

Total No. of Pages :02

Total No. of Questions : 09

B.Tech.(3D Animation & Graphics) (2012 Onwards)

B.Tech.(CSE)/(IT) (2011 Onwards)

(Sem.–3)

DATA STRUCTURES

Subject Code :BTCS-304

Paper ID : [A1126]

Time : 3 Hrs.

Max. Marks : 60

INSTRUCTION TO CANDIDATES :

1. **SECTION-A is COMPULSORY consisting of TEN questions carrying TWO marks each.**
2. **SECTION-B contains FIVE questions carrying FIVE marks each and students has to attempt any FOUR questions.**
3. **SECTION-C contains THREE questions carrying TEN marks each and students has to attempt any TWO questions.**

SECTION A

1. **Write briefly :**

(a) Describe Big 'O' notation used in algorithms.

(b) Give the classification of data types.

(c) Differentiate between linear and non-linear data structure.

(d) Explain the terms Front and Rear for queue.

(e) State the principle of stack and give it's two applications.

(f) Explain why binary search cannot be performed on a linked list.

(g) State different ways of traversing binary tree.

(h) What is hash function? Write its significance.

(i) Describe complete binary tree.

(j) Write any two applications of graph.

SECTION B

2. What are Linear and Non linear data structures? Give one example of each.

3. Write an algorithm for deleting a specific element from an array.

4. Explain application of Stack in recursive functions with example.

5. Explain the concept of circular queue and priority Queue with suitable example.

6. Discuss Heap sort with suitable example.

SECTION C

7. a) Write an algorithm to insert new node at the middle of a Singly Linked List.

   b) Convert the given Infix expression to Postfix expression using Stack and show the details of Stack at each step of conversion.

   Expression: (a + b * c^ d) * (e + f / g).

   Note : ^ indicates exponent operator.

8. Discuss in brief the AVL tree and B-tree. What are its advantages?

9. Write short note on :

   a) Transversal of a graph

   b) Bubble sort

Section -A

2015

Big O notation is used to describe the performance or complexity of an algorithm. It specifically describes the worst case scenario, and can be used to describe the execution time required or the space used by an algorithm.

2. A data type is an attribute of data which tells the compiler or interpreter how the programmer intends to use the data. The most common data types of real, integer and boolean.

3.

Linear data structure	Non-linear data structure
i.) Linear data structure arrange data in a sequential manner	Non-linear arrange data in hierarchical manner.
ii.) Memory utilization is in-efficient.	Memory utilization is efficient
iii.) It is of single-level	It is of multi-level.
iv.) eg. array, linked list, queue, stack.	eg. tree, graph

4. Queue is a linear data structure where the first element is inserted from one end called REAR and deleted from the other end called as FRONT. In a queue one end is used to enqueue and other is used to dequeue.

5. A stack is a container of objects that are inserted and removed according to Last-In-First-Out principle (LIFO). In case of stack elements are inserted and deleted only at one end called top. The main applications of stack are:-
i.) It is used to evaluate prefix, postfix and infix

ii) Stack is used to reverse a string. We push the ch... of string one by one into stack and then pop chara... from stack.

6. Binary search is possible on linked list if the list is ordered and the count of element is known. But while sorting, we can access a single element at a time through a pointer to node. This increases the traversal steps in linked list just to find the middle element. This makes it slow and inefficient.

7. There are two ways of traversing a binary tree :- breadth first and depth first

Depth first → It includes preorder, post order and inorder traversal. Preorder – Data, Left child, right child
Inorder – Left child, data, right child
Postorder – Left child, right child, data

Breadth first → In this we take the root, then its direct children and then their direct children and so on.

8. A hash function is a data ~~structure~~ function which is applied on a key by which it produces an integer, which can be used as in address of hash table. It can also be used for accessing the data from hash table.

9. A complete binary tree is one in which every node except the (leaf ~~root~~ node has two children) possibly deepest, is completly filled and all nodes must be as far as possible.

10. The main application of graphs are :-
i) Graphs are used to represent the flow of computation.
ii) In google maps, graphs are used. Intersection of two roades are considered to be vertex and roads connecting two roads are considered as edges. So, navigation system is set up.

SECTION-B

2015 Questions -

Linear Data Structure :-

- In linear data structure, data elements are arranged in a linear order.
- In linear data structure, single level is involved
- Its implementation is easy in comparison to non-linear data structure.
- Examples :- stack, Queue etc.

Non-linear Data structure

- In a non-linear data structure, data element are attached in hierachically manner.
- In non-linear data structure, multiple level is involved
- In non-linear data structure, memory is utilized in an efficient way.
- Example :- tree and graphs.

Ans 3

2	8	9	12	16	20	28	30

Step1 Back = i => Back = 5

Step2 while (Back < M) repeat step 3 to 4.
=> 5 < 8 true

Step3 Reg [5] => Reg [6] => reg [5] = 20

Step4 Back = Back + 1

Now Back = 6

again step 3.

Reg[b] = Reg[6=1]

= 28

6

Step 4: Back = Back+1 => 7

again step 3 until Back<M

step 5. M = M-1

= 8-1 => 7

7. step 6. End.

Ans 3 :- Application of stack in recursive functions.

→ The high level programming languages such as Pascal, c etc. that provides support for recursion use stack for book keeping.

→ Remember in recursion call, there is need to save the current value of parameter, local variable and return address.

→ Since each functions run its own environment or context, it becomes possible for function to call itself.

→ Example → Recursion as an application of stack is keeping book inside the drawer and removing each bok recursive

Circular Queue

- Circular Queue is a linear data structure in which the operations are performed on the basis of FIFO (First In First Out) principles
- In this the last position is connected back to the first position to make a circle.
- It is also called Ring Buffer.
- In a Normal Queue, we can insert element until queue becomes full.

Priority Queue

A priority Queue is a collection of element such that each element has been assigned a priority and such that the order in which elements are deleted and processed comes from the following rules.

1) An element of higher priority is processed before any elements of lower priority.
2) Two element with the same priority are processed a/c to the order in which they were added to the queue.

Heap Sort

- Heap sort is a comparison based technique based on Binary Heap data structure.
- It is similar to selection sort where we first find the maximum element & place the maximum element at the end.

Example

```
void heapSort ( int arr[], int n )
for ( int i = n/2 - 1 ;  i >= 0 ; i-- )
heapify (arr, n, i),
for ( int i = n-1; i >= 0 ; i-- )
Swap ( arr[0], arr [i] ),
hapify (arr, i, 0),
}
```

Section-C 2MS May

Write an algorithm to insert new node at middle

Input: n position to insert data in the list.

```
Begin: createSinglyLinkedList (head)
       alloc (newNode)
       If (newNode == NULL) then
         write ("Unable to allocate.")
       End if
       Else then
         read(data)
         newNode.data ← data
           temp ← head
         for i ← 2 to n-1
           temp ← temp.next
           If (temp==NULL) then
             break
           End if
         End for
         If (temp != NULL) then
           newNode.next ← temp.next
           temp.next ← newNode
         End if
       End else
End.
```

b) ~~a+b*c^d)*(e+f/g~~ Infix to Postfix

(a+b*c^d)*(e+f/g)

Symbol	Stack	Expression
(	((	
(	((	a
a	((	a
+	((+	ab
b	((+	ab
*	((+*	abc
c	((+*	abc
^	((+*^	abcd
d	((+*^)	abcd^*+
)	((+*^)=(-)(	abcd^*+
*	(*	abcd^*+
(	(*(	abcd^*+e
e	(*(	abcd^*+ef
+	(*(+	abcd^*+ef
f	(*(+	abcd^*+efg
/	(*(+/	abcd^*+efg/
g	(*(+/	abcd^*+efg/+*
)	(*(+/))	

8. Discuss AVL and B-tree. What are its advantages?

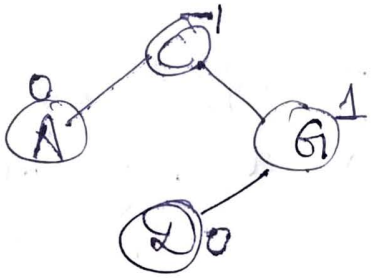
ns AVL Tree → A non empty tree is an AVL-tree if given T^L and T^R to be left and right subtrees of T and $h(T^L)$ and $h(T^R)$ to be heights of subtrees

and T^R are AVL trees and $|h(T^L) - h(T^R)| \leq 1$.

$h(T^L) - h(T^R)$ is known as balance factor and balance factor can be only 0, 1, −1

Also, AVL search tree is a binary search tree which is an AVL tree.

Representation



Insertion

Inserting leads to effecting the balance factor Rotations are used to restore the balance of search tree. The rebalancing rotations are LL, LR, RR, RL

LL rotation – Inserted node is in left subtree of left subtree of node A.

RR rotation – Inserted node is in the right subtree of right subtree of node A.

LR rotation – Inserted node is in the right subtree of left subtree of node A.

RL rotation – Inserted node is in the left subtree of right subtree of node A.

Deleting → requires rotations → classified as

- R_0, R_1, R_{-1} or L_0, L_1, L_{-1} depending upon deletion on left or right as well as value of Bf in tree

B tree → A B-tree of order m, if non empty is an m-way search tree in which

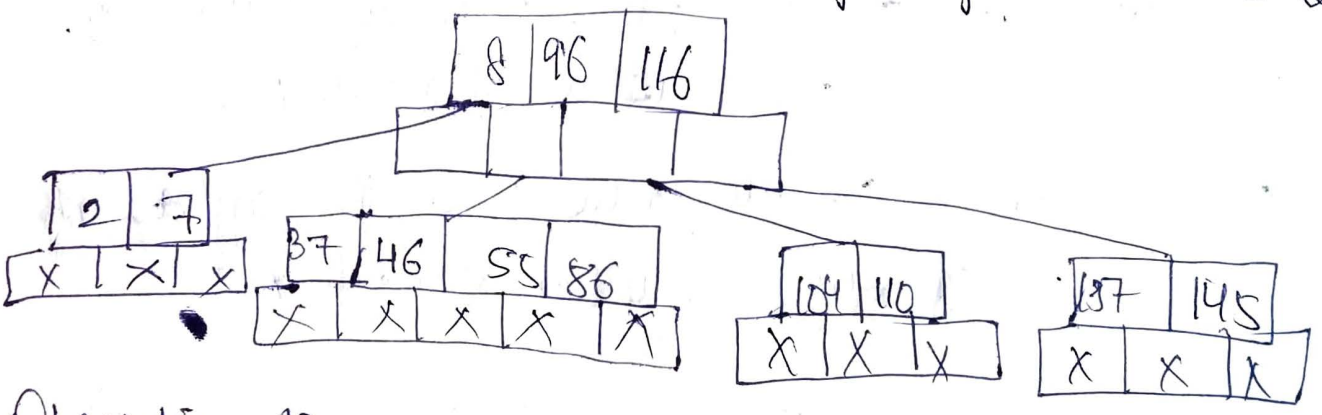
(i) the root has at least two child nodes and at most m child nodes.

(ii) the internal nodes except the root have at least $\lceil m/2 \rceil$ child nodes and at most m child nodes.

(iii) the number of nodes (keys) in each internal node is one less than the number of child nodes and these keys partition the keys in subtrees of node in a manner similar to that of m-way search trees.

(iv) all leaf nodes are on the same level.

A B tree of order 3 is referred to as 2-3 tree since the internal nodes are of degree 2 or 3 only.



Operation that can be performed

(i) Searching - This is similar to searching on m way tree

(ii) Insertion -

(iii) Deletion -

Advantages of B tree

keeps keys in sorted order for sequential traversing
uses a hierarchical index to minimize the number of disk reads
(3) uses partially full blocks to speed insertions and deletions
(4) keeps the index balanced with recursive algorithm

Advantages of AVL tree

(1) AVL trees are height balance trees. This make practically efficient.
(2) Operations like insertion and Deletion have low time complexity.

(4) Write a short note on:
(a) Transversal of a graph:
(a) There are 2 standard ways that to examine the nodes and edges of a graph. One is called breadth first search. Other is called depth first search.
(i) Breadth first search – This is accomplished by using a queue to hold nodes that are waiting to be processed.
(ii) Depth first search – This is accomplished by using a stack to hold nodes that are waiting to be processed.

Algorithm for Breadth first Search

1) Initialise all nodes to ready state.
2) Put the starting Node A in Queue and change status to waiting state
3) Repeat steps 4 & 5 until Queue is empty.
4) Remove the front Node N of QUEUE and Process N and change state of N to the processed state
5) Add to the rear of QUEUE all the neighbours of N that are in steady state and change their status to waiting state
   End of step 3 Loop
6) Exit

Algorithm for Depth first Search

1) Initialise all nodes to the ready state
2) Push the starting node A onto STACK and changes its status to the waiting state
3) Repeat steps 4 and 5 until Stack is empty
4) Pop the top node N of stack. Process N and changes its status to the processed state
5) Push onto stack all the neighbours of N that are still in the ready state and change their status to the waiting state.
   (End of step 3 Loop)
6) Exit.

ble Sort.

Bubble Sort is a simple algorithm which is used to sort a given set of n elements provided in form of an array with n number of elements. Sorting takes place by stepping through all the elements one by one and comparing it with adjacent element and swapping it if required.

Best complexity ⇒ n

Worst complexity ⇒ n^2.

Algorithm

1. Input array A of elements n.
2. for i = 0 to N-1 repeat Step 2.
3. for j = 0 to N-1 repeat
4. If A[J] > A[J+1]
5. Swap A[J] and A[J+1]
6. End of Inner Loop.
7. End of Outer Loop
8. Exit.

Dec 2015

Roll No.

Total No. of Pages : 02

Total No. of Questions : 09

B.Tech.(3D Animation & Graphics) (2012 Onwards)
B.Tech.(CSE/IT) (2011 Onwards)
(Sem.–3)

DATA STRUCTURES

Subject Code : BTCS-304
Paper ID : [A1126]

Time : 3 Hrs.

Max. Marks : 60

INSTRUCTIONS TO CANDIDATES :

1. SECTION-A is COMPULSORY consisting of TEN questions carrying TWO marks each.
2. SECTION-B contains FIVE questions carrying FIVE marks each and students have to attempt any FOUR questions.
3. SECTION-C contains THREE questions carrying TEN marks each and students have to attempt any TWO questions.

SECTION-A

1. Write briefly :

a. What is meant by an static and dynamic data structure?

b. Differentiate between AVL and B-tree.

c. What are the objectives of studying data structures?

d. What is recursion and its advantages and disadvantages?

e. What is a circular linked list and its use?

f. What condition is checked to check whether a tree is empty or full?

g. Convert into infix :

i) + A B C + * D –

ii) A B * C – D +

h. Define the term Heaps. How they are stored in memory?

i. What is meant by Hashing and rehashing?

j. What is the role of Priority queue in operating system design?

SECTION-B

2. Define the terms : Linear and non-linear data structures. List down any four applications of data structures.

3. What are the different ways of traversing a tree? Write a tree traversal algorithm which outputs sorted data.

4. What are the different ways to implement list? What are the advantages in the array implementation of list?

5. State the difference between arrays and linked lists. Discuss the advantages of representing stacks using linked lists than arrays.

6. What is Graph? Describe in brief the various methods used to represent Graphs in memory.

SECTION-C

7. What is the advantage and average efficiency of Insertion sort? Sort the following data using an insertion sort algorithm and Show the contents of the array after every pass :

   23 7 92 6 12 14 40 44 20 21

8. Why do we need Queues? Write the algorithms/programs for EmptyQ, FullQ, InsertQ and DeleteQ operations.

9. Write the algorithms for the following :

   a. Inserting an element from a doubly link list.

   b. Deleting an element in a priority.queue.

   c. To reverse a string of characters.

   d. To search an element in a sorted array.

Section - A

Write briefly:

a. What is meant by an static and dynamic data structure?

Ans) Static data structure: In static data structure the size of the structure is fixed. The content of the data structure can be modified but without changing the memory space allocated to it.

Example: Array

Dynamic data structure: In dynamic data structure the size of the structure is not fixed and can be modified during the operations performed on it.

Example: Linked list.

b. Differentiate between AVL and B tree.

Ans)

AVL tree	B tree
An AVL tree is a self balacing binary search tree.	if A B tree is a balanced tree, but it is not a binary tree.
It is used for in memory backed search trees.	if A B-tree is primarily used as a storage backed search tree.

What are the objectives of studying data structure.

Ans)
- To evaluate postfix expressions by using stacks.
- To comprehend the expression formats of prefix, infix and postfix.
- To determine the various types of abstract data such a queue, stack, lists.
- To evaluate the performance of linear data structure

e) What is recursion and its advantages and disadvantages

Ans) When a function calls itself from its body is called recursion.

Advantages: 1) Reduce unnecessary calling of function.
2) Through recursion one can solve problems in easy way while its iterative solution is very big and complex.

Disadvantages: 1) Recursive solution is always logical and it is very difficult to trace errors.
2) Recursion uses more processor time.

f) What is circular linked list and its use?

Ans) Circular linked list is a linked list where all nodes are connected to form a circle. There is no NULL at the end. A circular linked list can be a singly circular list or doubly circular linked list.

Use: Circular linked list are used where the entire list is accessed one by one in a loop.

Example: Operating Systems may use it to switch between various running applications in a circular loop.

f) What condition is checked to check whether a tree is empty or full?

Ans) If a binary node is NULL then it is a full tree or in which all nodes have either zero or two child nodes is a full tree.

g) Convert into infix:

i) *ABC+*D-

A

A, B

A, B, C

A, B, C, +

A, B+C, *

(A *(B+C)), D

(A *(B+C)), D, -

= (A* (B+C))-D

iii) AB * C - D +

A	A
B	A, B
*	A, B, *
C	A*B, C
-	A*B, C, -
D	(A*B)-C, D
+	(A*B)-C, D, +

= ((A*B)-C)+D

b) Define the term heaps. How they are stored in memory.

Ans) Heap is a memory used by programming languages to store global variables. It supports dynamic memory allocation.

Variables allocated on the heap have their memory allocated at run time and accessing this memory is slower. We can allocate a block at any time and free it at any time.

c) What is meant by hashing and rehashing?

Ans) A hash function converts a number in a large range into a smaller range. This process is called hashing.

Rehashing means hashing again. When load factor inc to more than its pre defined value the complexity increases To overcome this array size is increased & all values are

j) What is the role of priority queue in operating system design?

Ans) Role of priority queue in operating system is for load balancing, interrupt handling

Section-B

2) Define the terms linear and non-linear data structure.

Ans) Linear data structure: Data structure where data elements are arranged sequentially or linearly where the elements are attached to its previous and next adjacent in what is called linear data structure. Examples: Array, stack.

Non linear data structure: Data structure where data elements are not arranged sequentially or linearly are called non-linear data structure.
They are not easy to implement in comparison to linear data structure. Examples: Trees and graphs.

Applications:-

1) Queues are used for interprocess communications.
2) Stack is used when a function is called.
3) Binary Search trees are used for implementing maps.
4) Heaps are used to implement priority queue which is used for scheduling processes by operating system.

Otherwise
(a) EMPTY := false;
3. Return.

FULL(QUEUE, FRONT, REAR, MAX, FULL)
1. If (REAR = MAX) then
(a) FULL := true;
2. Otherwise
(b) FULL := false;
3. Return.

QINSERT(QUEUE, N, FRONT, REAR, ITEM)
1. If FRONT = 1 and REAR = N, or if FRONT = REAR + 1, then
Write: OVERFLOW, and Return
2. If FRONT := NULL, then:
Set FRONT := 1 and REAR := 1
Else if REAR = N then:
Set REAR := 1
Else:
Set REAR := REAR + 1
3. Set QUEUE[REAR] := ITEM
4. Return.

QDELETE(QUEUE, N, FRONT, REAR, ITEM)
1. If FRONT := NULL, then: Write UNDERFLOW and Return
2. Set ITEM := QUEUE[FRONT].
3. If FRONT = REAR, then:
Set FRONT := NULL and REAR := NULL.

6 7 12 14 23 40 44 92 20, 21

6 7 12 14 20 23 40 44 92 21.

6 7 12 14 20 21 23 40 44 92.

Sorted list is found.

Algorithm for Insertion Sort:

INSERTION (A, N)

1. Set A[0] := $-\infty$
2. Repeat step 3 to 5 for K = 2, 3, ... N
3. Set TEMP := A[K] and PTR := K-1
4. Repeat while TEMP < A[PTR]
   (a) Set A[PTR+1] := A[PTR]
   (b) Set PTR := PTR-1
5. Set [PTR+1] := TEMP
6. Return

Q-8. Queue

A queue is a linear structure which follows a particular order in which the operations are performed. Example ticket counter.

It follows FIFO (first in first out).

EMPTY (QUEUE, FRONT, REAR, EMPTY)

This algorithm is used to check whether a QUEUE is EMPTY or not.

1. If (FRONT = REAR+1) then
   a. Set EMPTY := true;

linked Representation:

1. A graph can be represented using linked list
2. It requires less amount of memory.
3. For each vertex, a list of adjacent vertices is maintained using linked list.
4. Adjacency list of graph is represented by an array of pointers and each pointer points to the respected linked list of vertex.

SECTION – C.

7. Advantage of Insertion sort.
   - Simple Implementation
   - Efficient for small data sets
   - Adaptive - efficient for data sets that are already substantially sorted: the time complexity is O(n+d) where d is number of inversions.
   - more efficient in practice than most other simple quadratic i.e. O(n2) algorithms such as selection sort or bubble sort; the best case is O(n)

23 7 92 6 12 14 40 44 20 21

7 23 92 6 12 14 40 44 20 21

7 23 92 6 12 14 40 44 20 21

6 7 23 92 12 14 40 44 20 21

6 7 12 23 92 14 40 44 20 21

6 7 12 14 23 92 40 44 20 21

6 7 12 14 23 40 92 44 20 21

Q5.

Array	Linked list
• It is a consistent set of a fixed number of data items.	• It is an ordered set comprising a variable number of data items.
• Element location is allocated during compile time	• Element position is assigned during run time.
• We can perform binary search and linear search.	• We can only perform linear search.
• Memory required is less	• Memory required is more.

Advantages of representing stacks using linked list than arrays are:

- It is not necessary to specify the number of elements to be stored in stack during its declaration, since memory is allocated dynamically at run time when an element is added to stack.
- Insertion and deletion can be handled easily.

Q6. A graph is a non-linear data structure consisting of nodes and edges. The nodes are sometimes also referred to as vertices and the edges are lines or arcs that connect any two nodes in the graph.

Graph can be represented in two forms:

(a) Sequential Representation

1. Graphs can be represented through matrix in system memory.
2. This is sequential in nature

```
struct list_element *prev;
struct list_element *next;
void * data;
};
```

Another method:

```
struct list_element{
    struct list_element *prev;
    struct list_element *next;
};
struct person{
    char name[20];
    unsigned int age;
    struct list_element list_entry;
};
```

Advantages in array implementation:

→ Using array implementation, you can have sequential and faster access to nodes of list.

→ Array implementation is helpful when you are dealing with fixed no. of elements.

3) What are the different ways of traversing a tree. Write a tree traversal algorithm which outputs sorted data.

Ans) Different ways of traversing a tree:

1) Inorder (Left, Root, Right)
2) Preorder (Root, Left, Right)
3) Postorder (Left, Right, Root)

Algorithm

INORD (INFO, LEFT, RIGHT, ROOT)

1) Set TOP := 1, STACK [1] = NULL and PTR := ROOT
2) Repeat while PTR != NULL
   Set TOP := TOP + 1 and STACK [TOP] := PTR
   Set PTR := LEFT [PTR]
3) Set PTR := STACK [TOP] and TOP := TOP - 1
4) Repeat steps 5 to 7 while PTR != NULL
5) apply PROCESS to INFO [PTR]
6) If RIGHT [PTR] != NULL then:
   Set PTR := RIGHT [PTR]
   Go to step 3
7) Set PTR := STACK [TOP] and TOP := TOP - 1.
8) Exit.

4) What are the different ways to implement list? What are the advantages in the array implementation of list.

Ans) One method is to build functions around a structure like

struct list_element {

FRONT = N then:
Set FRONT := 1
Else
Set FRONT := FRONT + 1
4. Return.

Q-9

(a) INSTIOL(INFO, FORW, BACK, START, AVAIL, LOCA, LOCB, ITEM)

1. If AVAIL = NULL, then: Write OVERFLOW and exit.
2. Set NEW := AVAIL, AVAIL := FORW[AVAIL],
   INFO[NEW] := ITEM.
3. Set FORW[LOCA] := NEW,
   FORW[NEW] := LOCB, BACK[LOCB] := NEW,
   BACK[NEW] := LOCA
4. Exit.

(b) 1. Set ITEM := INFO[START]
2. Delete first node from the list
3. Process ITEM
4. Exit.

(c) BINARY(DATA, LB, UB, ITEM, LOC)

1. Set BEG := LB, END := UB and MID = INT(BIG + END)/2
2. Repeat Step 3 and 4 while BEG ≤ END and DATA[MID] ≠ ITEM
3. If ITEM < DATA[MID], then:
   Set END := MID - 1
   Else:
   Set BEG := MID + 1
4. Set MID := INT(BEG + END)/2

5. If DATA[MID] = ITEM, then:
   Set LOC := MID
   Else:
   Set LOC := NULL.
6. Exit.

May 2016

Roll No. | | | | | | | | | | | |

Total No. of Pages :02

Total No. of Questions : 09

B.Tech.(3D Animation & Graphics) (2012 Onwards)

B.Tech.(CSE/IT) (2011 Onwards)

(Sem.–3)

DATA STRUCTURES

Subject Code :BTCS-304

Paper ID : [A1126]

Time : 3 Hrs. Max. Marks : 60

INSTRUCTIONS TO CANDIDATES :

1. **SECTION-A is COMPULSORY consisting of TEN questions carrying TWO marks each.**
2. **SECTION-B contains FIVE questions carrying FIVE marks each and students have to attempt any FOUR questions.**
3. **SECTION-C contains THREE questions carrying TEN marks each and students have to attempt any TWO questions.**

SECTION-A

1. **Write briefly :**

a) Memory leaks

b) Data structure versus data type

c) Sparse matrix

d) Stacks and recursive functions

e) Linked representation of queue

f) AVL Tree

g) Representation of graph in memory

h) Rehashing

i) Algorithm complexity

j) Dynamic memory allocation

SECTION-B

2. How multidimensional arrays are stored in memory? Explain row major representation of an array.

3. What is meant by postfix expressions? How postfix expressions are evaluated by using stacks?

4. Explain the linked representation of queue and operations to be performed on it with the help of suitable example.

5. Discuss the operations on heap with the help of suitable example.

6. What is a hash table? Discuss the concept of collision resolution in hash table with the help of suitable example.

SECTION-C

7. Consider the following numbers are stored in an array A : 32, 51, 27, 85, 66, 23, 13, 57
Apply Bubble sort algorithm to the array A and show each pass separately.

8. Write the algorithm for pre-order tree traversal. Also show the steps of this algorithm on an example set of numbers.

9. What is a doubly-linked list? Write an algorithm to create a doubly-linked list and also write a function to insert a node in doubly-linked list.

Section - A

Ques 1 Write briefly:

(a) Memory leaks: Memory leaks occurs when programmers create a memory in heap and forget to delete it. Memory leaks are particularly serious issues for programs like daemons and servers which by definition never terminate. To avoid memory leaks memory allocated on heap should always be freed when no longer needed.

(b) Data structure versus datatype.

Data Structure	Data Type
It is way of describing certain way to organize pieces of data so that operations and algorithms can be more easily applied. For example, tree type datastructure allow efficient searching algorithm.	It describes pieces of data that all share common property. For example Integer datatype describes every integer data computer can hold.

(c) Sparse matrix: A matrix is a two dimensional data objects made of m rows and n column, therefore having total $m \times n$ values. If most of elements of matrix have 0 value then it is called sparse matrix. Since there are less non zero elements so less memory is used

Eg $\begin{bmatrix} 0 & 0 & 3 & 0 & 4 \\ 0 & 3 & 1 & 0 & 0 \\ 3 & 1 & 1 & 0 & 0 \end{bmatrix}$ is an example of sparse matrix.

d) Stacks and recursive functions:

Stack is a linear data structure which follows particular order in which operations are performed on basis of principle Last in first out [LIFO].

The process in which function calls itself directly or indirectly is called recursive function. Using the recursive algorithm certain problem can be solved quite easily like Tower of Hanoi, Tree Traversals.

In recursive program solution to base case is provided and solution of bigger problem is expressed interms of smaller problem.

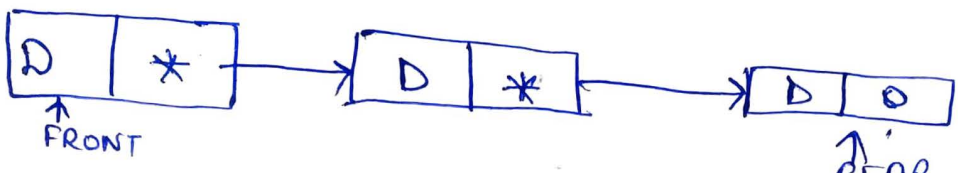
eg:

```
int fact (int n)
{
  if (n<=1)
      return 1;
  else
    return n * fact (n-1);
}
```

e) Linked representation of queue.

In linked queue each node of queue consist of two parts ie data part and link part. Each element of queue points to immediate next element in the memory. In linked queue there are two pointers maintained in memory, front and rear pointer. Front pointer contain the address of starting element of queue while rear pointer contains address of last element of queue.

eg



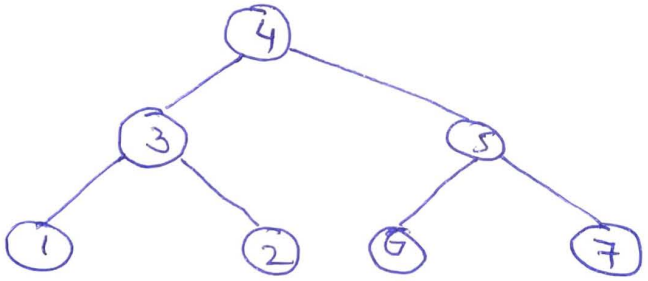
(f) AVL Tree : AVL Tree is a binary search tree in which difference of heights of left and right subtrees of any node is less than or equal to one. Technique of balancing heights of binary tree was developed by Adelson, Velskii, Landis and hence given name AVL tree. Tree is height balanced if

(a) T_L and T_R are height balanced

(b) $h_L - h_R <= 1$ where $h_L - h_R$ are heights of T_L, T_R.

The balance factor of node in binary tree can have values 1, 0, -1 depending on whether the left tree have height, greater than, equal to or less than height of Right subtree.

Eg



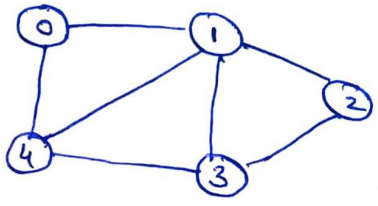
(g) Representation of graph in memory.

Graph is a data structure that consist of finite set of vertices called nodes and finite set of ordered pair called edges. The commonly used method of representation of graph in memory are :-

1 Adjacency Matrix (2) Adjacency list

Adjacency matrix is two dimensional (2D) array of size $m \times m$ where m is number of vertices Eg

For eg



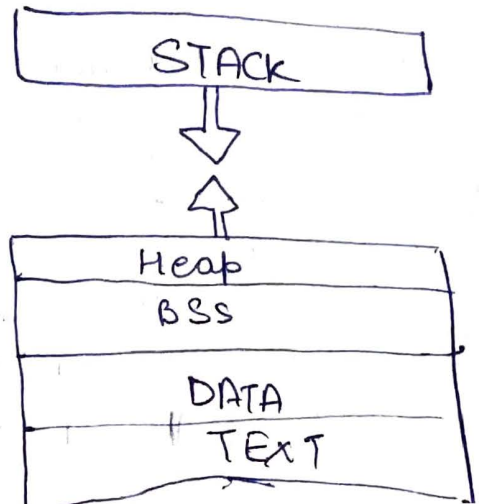
	0	1	2	3	4
0	0	1	0	0	1
1	1	0	1	1	1
2	0	1	0	1	0
3	0	1	1	0	1
4	1	1	0	1	0

$\begin{bmatrix} 2 & 3 \\ 4 & 5 \end{bmatrix}_{2\times 2}$

(h) Rehashing → Hashing refers to mapping given value to particular key for faster access of elements. Efficiency of function depends on efficiency of hash function used.
Rehashing is done because whenever key value pairs are inserted into map load factor increases which implies time complexity also increases. Hence rehashing must be done by increasing size of bucket Array so as to reduce load factor and time complexity.

(i) Algorithm complexity :- Algorithm complexity is a measure which evaluates order of count of operations performed by given algorithm as function of size of the input data. To put this, complexity is rough approximation of number of steps necessary to execute algorithm. It is generally represented by O(f) notation, (Big-O-notation)
Complexity can be constant, logarithmic, linear.

(j) Dynamic memory allocation:- Dynamic memory allocation is when an executing program request that operating system give it block of main memory. Program uses this memory for getting new memory for new objects
Thus Dynamic memory allocation refers to managing system memory at runtime. When program is loaded in system memory, memory region allocated to program is divided into three broad region: stack, heap, code.



How multidimensional arrays are stored in memory Explain row major representation of an array!

Ans: A multidimensional array is an array with more than two dimensions. In a matrix, the two dimensions are represented by rows and columns. There are two ways to store array elements in the memory: Row and Column major.

- Row major representation (row wise)

Under this representation the first row of the array occupies the first set of memory location reserved for the array, the second row occupies the next set and so forth.

Consider a Two Dimensional Array consist of N rows and m columns. It can be stored sequentially in memory row by row as shown below:

Row 0	A[0,0]	A[0,1]	-----	A[0,m-1]
Row 1	A[1,0]	A[1,1]	-----	A[1,m-1]
⋮	⋮			⋮
Row n-1	A[N-1,0]	----------		A[N-1, m-1]

Ques3: What is meant by postfix expressions? How postfix expressions are evaluated by using stack?

Ans: Postfix expression is a notation for writing arithmetic expressions in which the operands appear before their operators.

Evaluation rules of a Postfix Expression are:

1. While reading the expression from left to right, push the element in the stack if it is an operand.
2. Pop the two operands from the stack, if the element is an operator and then evaluated it.
3. Push back the result of the evaluation. Repeat it will till the end of the expression.

ALGORITHM:

Step 1: Add) to postfix expression.

Step 2: Read postfix expression Left to Right until) encount

Step 3: If operand is encountered, push it onto stack.

[Stack Is Full]

[End If]

Step 4: If operatove is encounted, Popo two elements

(i) A -> Top element

(ii) B -> Next to Top element

(iii) Evaluate B operator A

push B operator A onto Stack

Step 5: set reesult = pop

Step 6: END

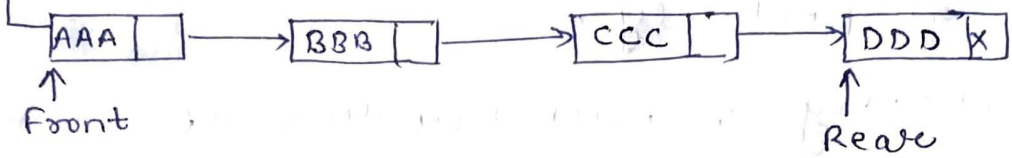
Ques4: Explain the linked rulpresentation of queue and operations to be peroformed on it with the help of example.

Ans: In a linked queue, each node of the queue consists of two parts, i.e. data part and the link part. Each element of the queue points to it's immediate next elemen

memory. In the linked queue, there are two pointer
tained in the memory i.e. front pointer and rear point

Representation:

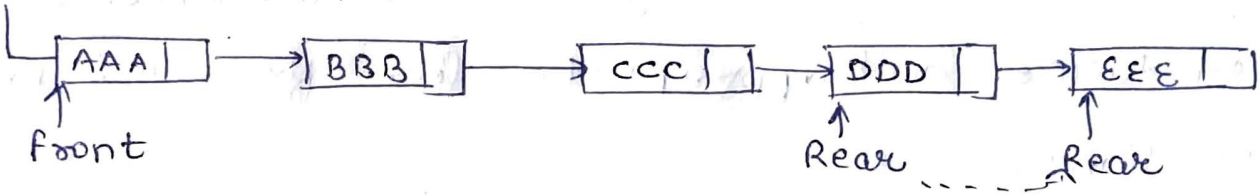
Queue Q:



OPERATIONS:

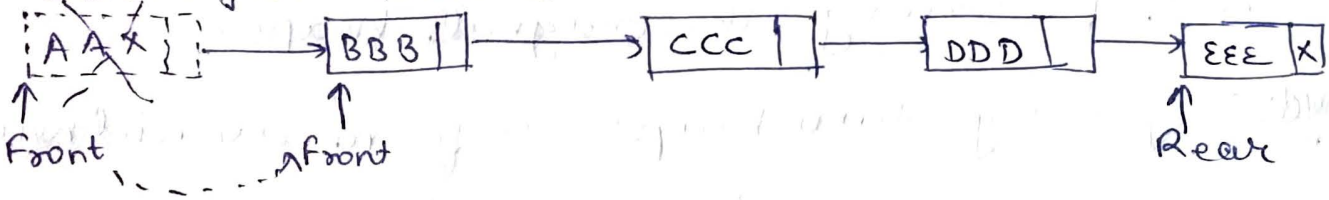
i) EnQueue: This operation add a new node after rear and moves rear to the next node.

Insert 'EEE' into queue Q:



ii) DeQueue: The operation removes the front node and moves front to the next node.

Delete from queue Q



Que 5: A heap is tree based data structure in which all the nodes of the tree are in specific order.

for eg:- if the parent node of, then the value of follows a specific order with respect to the value of and the same order will be follo-wed accoss the tree.

The comman operations involving heaps are:-

Basic

find-max (or find min): find a max. item of a max heap, or a minimum item of a min. heap, respectively.

insert: adding a new key to heap

delete-max: removing the root node of max heap respectively replace: pop root and push a new key. More efficient than pop followed by push, since only need to balance once, not twice and appropiate for fixed size heap

Creation:-

Create-heap:- Crestte an empty heap heapify: create a heap out of given array of elements of both preserving the orignal heaps.

merge: joining two heaps to form a valid new heap containg all the elements of both, preserving the orignal heaps.

Size: Return the number of items in the heap.

is empty: return true if the heap is empty false otherwise.

Internal

increase key or decrease key: updating a key with a max- or min heap, respectively

Q3 What is a hash table? Discuss the concept of collision resolution in hash table with example.

Ans A hash table is a data structure that implements an associative array abstract data type, a structure that can map keys to values. A hash table uses a hash function to compute an index, also called a hash code, into an array of buckets or slots, from which the desired value can be found.

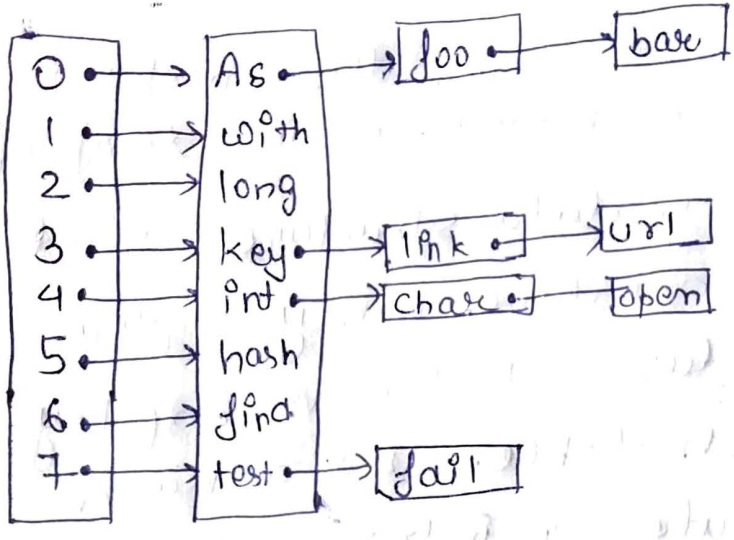
Concept of collision:

- Two keys mapping to the same location in the hash table is called collision.
- Collision can be reduced with a selection of good hash function.
- But it is not possible to avoid collisions altogether
  → Unless we can find a perfect hash function.
  → which is hard to do.

Collision Resolution Strategies

- Few Collision Resolution ideas
  → Separate chaining
  → Some Open addressing techniques like linear and Quadratic Probing

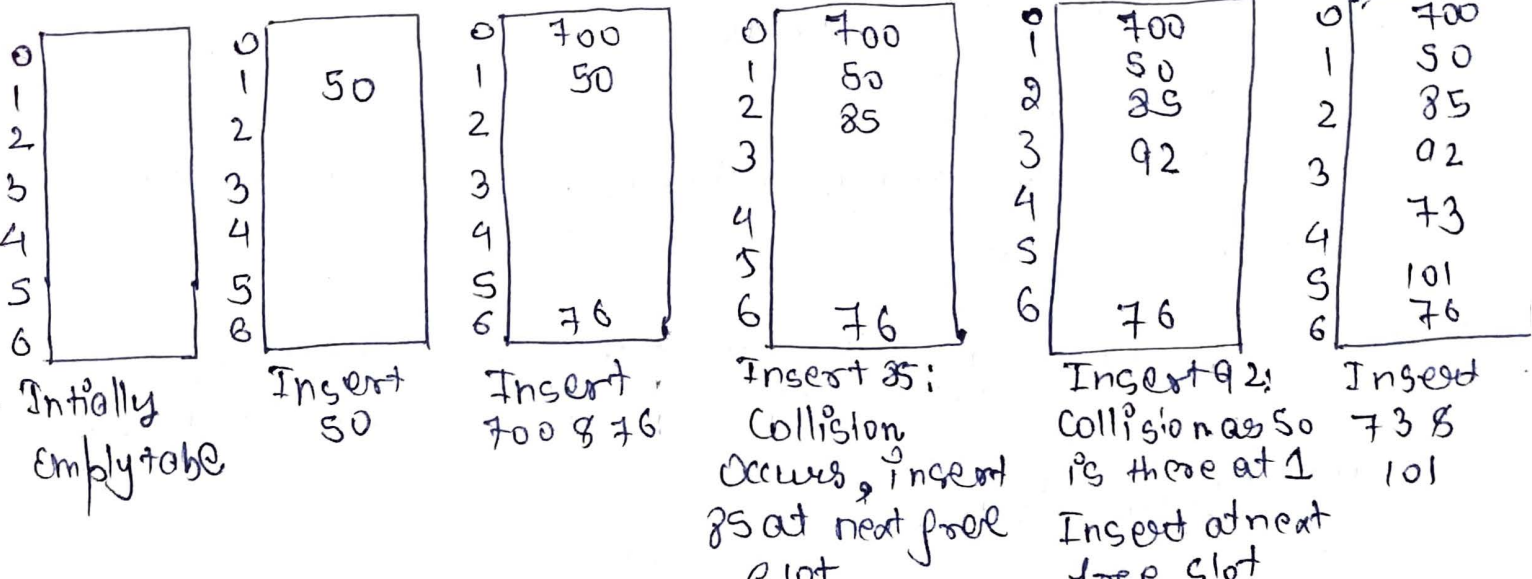
(a) Separate chaining: Collisions can be resolved by creating a list of keys that map to the same level.



(b) Addressing (linear Probing)

- The idea :
  - → Table remains a simple array of size N.
  - → On insert (x), compute f(x) mod N, if the cell is full, find another by sequentially searching for the next available slot
    go to f(x)+1, f(x)+2 etc---
  - → On find (x), compute f(x) mod N, if the cell doesn't match look elsewhere.
  - → Linear probing function can be given by
    - $h(x,i) = (f(x)+i) \bmod (N) \; (i = 1, 2 \text{---})$

Example: Let us consider hash function as "key mod 7" and sequence of keys as 50, 700, 76, 85, 92, 73, 101



Consider the following Numbers are sorted In an array A: 32, 51, 27, 85, 66, 23, 13, 57.
Apply bubble sort algorithm to the Array A and show each pass Respectively?

Ans-1.

Pass 1. (a) Compare A_1 and A_2. Since 32 < 51, the list Is Not altered.

(b) Compare A_2 and A_3. Since 51 > 27 Interchange 51 & 27 as follows: 32 (27) (51) 85 66 23 13 57

(c) Compare A_3 And A_4. Since 51 < 85 the list Is Not altered.

(d) Compare A_4 And A_5. Since 85 > 66 interchange 85 and 66 as follows - 32 27 51 (66) (85) 23 13 57

(e) Compare A_5 And A_6. Since 85 > 23, interchange 85 And 23 as follows -

32 27 51 66 (23) (85) 13 57

(f) Compare A_6 And A_7. Since 85 > 13, interchange 85 And 13 to yield -

32, 27, 51, 66, 23, (13), (85) 57

(g) Compare A_7 And A_8. Since 85 > 57, Interchange 85 And 51 to yield.

32, 27, 51, 66, 23, 13, (57) (85)

At the End of this first pass, the largest Number 85, Has Moved to the last position. However, the rest of the Numbers are Not sorted, even though some of them Have changed theire positions.

Pass 2- (27) (33) 51 66 23 13 57 85

27 33 51 (23) (66) 13 57 85

27 33 51 23 (13) (66) 57 85

27 33 51 23 13 (57) (66) 85

At the End of pass 2, the second largest Number, 66, Has Moved Its way down to the Next-to-last position.

Pass 3 - 27 33 (23) (51) 13 57 66 85

27 33 23 (13) (51) 57 66 85

Pass 4 - 27 (23) (33) 13 51 57 66 85

27 (23) (13) (33) 51 57 66 85

Pass 5 - (23) (27) 13 33 51 57 66 85

23 (13) (27) 33 51 57 66 85

Pass 6 - (13) (23) 27 33 51 57 66 85

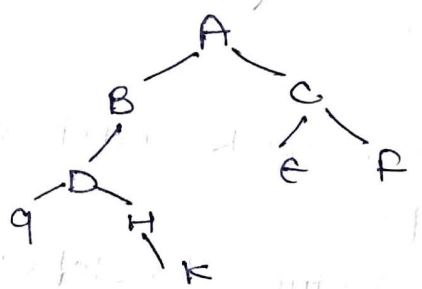
Pass 7 - Finally, A_1 is compared with A_2. Since $13 < 23$, NO Interchange takes place.

Write the Algorithm for pre-order Traversal. Also show the steps of this algorithm on an example set of Numbers?

Ans- A Binary Tree T is in Memory. The Algorithm does a preorder Traversal of Tree. Applying an operation process to each of its Nodes. An Array stack is used to Temporarily Hold the addresses of Node.

1. [Initially push Null onto stack] and Initialize PTR. Set TOP:=1, stack[1]:= Null And PTR:=ROOT
2. Repeat steps 3 to 5 While PTR ≠ Null:
3. Apply Process to INFO[PTR].
4. If Right[PTR] ≠ Null, then:
   Set TOP:= TOP+1 and stack[TOP] := Right[PTR].
   [End of If structure].
5. If left[PTR] ≠ Null, then:
   Set PTR := left[PTR].
   else: [Pop from stack]
   Set PTR := stack[TOP] and TOP := TOP-1
6. exit.

example



1. Initially push Null onto stack: stack: ϕ
   Then set PTR:= A, the Root of T.
2. Proceed down the left-Most path rooted at PTR=A
   (i) Process A and push its Right child C on stack
   stack: ϕ, C.
   (ii) Process B (Three is No Right child)
   (iii) Process D And push its Right child H onto stack:
   stack: ϕ, C, H.

(iv) process G.

No other Node Is processed.

3. [Backtracking] Pop the element H from stack, and set PTR := H. This leaves: stack: ϕ, C.

Since PTR ≠ Null return to step (a) of the algorithm.

4. Proceed down the left Most path rooted at PTR = H as follows-

(v) Process H and push Its Right child K onto stack:

stack: ϕ, C, K.

No other Node Is processed, since H No child

5. [Backtracking] Pop K from stack, and set PTR := K. This leaves-

stack: ϕ, C.

Since PTR ≠ Null, return to step (a) of the Algorithm.

6. Proceed down the left Most path Rooted at PTR = K as follows-

(vi) Process K. (There Is No Right child).

No other Node Is processed, since K Has No left child.

7. [Backtracking]. pop C from STACK and set PTR := C This leaves: stack: ϕ.

Since PTR ≠ Null, return to step (a) of the algorithm.

8. Proceed down the left Most path rooted at PTR = C as follows-

(vii) Process C And push Its Right child F onto stack.

stack: ϕ, F.

(viii) Process F.

9. [Backtracking]. Pop F from stack And set PTR := F

This leaves: stack: ϕ.

Set PTR ≠ Null, return to step (a) of algorithm.

10. Proceed down the left Most path rooted at PTR = F as follows:

(ix) Process F. [There Is No Right child].

No other Node Is processed, since F Has No left child.

11. [Backtracking]. Pop the Top elemet Null from stack and set PTR := Null.

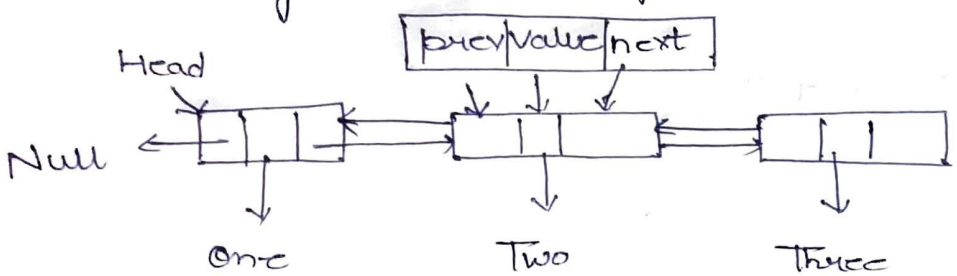
Since PTR = Null, the Algorithm Is Completed.

2. What Is a Doubly linked list. write An algorithm to create a doubly linked list And Also write a function to Insert a Node In a Doubly linked list?

Ans- A doubly linked list Is one In which all the Nodes are linked together by Multiply links which Helps In Accessing both successor And predecessor Node for Any Arbitary Node within the list.

(i) Every Node In a Doubly linked list Has Three fields left pointer, Right Pointer And Data.

(ii) The last Node Has a Next link with value, Null, Marking the End of the list, and the first Node Has a prev link with the value Null. The Start of the list Is Marked by the Head pointer.



Algorithm

Assume that Start is the first Element In the linked list And Tail Is the last Element of the linked list.

(i) Insert at Beginning

1. Start
2. Input the Data to be Inserted
3. Create a New Node
4. New Node → Data = Data Newnode → Upoint = Null
5. If Start Is Null New Node → Rpoint = Null.
6. Else New Node → Rpoint = Start
   Start → Upoint = New Node
7. Start = New Node
8. Stop.

(ii) Insertion at location-

1. Start

... And POS

3. Initialize Temp= Start ; i=0
4. Repeat the Step 4 If (i less than POS) And (Temp Is Not equal to Null)
5. Temp= Temp → Rpoint ; i= i+1
6. If (Temp Not Equal to Null) And (Equal to POS)

(a) Create a New Node.

(b) New Node → Data = Data

(c) New Node → Rpoint = Temp → Rpoint

(d) New Node → Lpoint = Temp

(e) (Temp → Rpoint) → Lpoint = New Node

1. (f) Temp → Rpoint = New Node

2. Else.

(g) Display "position NOT found".

1. Stop

(iii) Insert At End-

1. Start
2. Input Data to be Inserted
3. Create a New Node
4. New Node → Data = Data
5. New Node → Rpoint = Null
6. If (Start Equal to Null)

(a) Start = New Node

(b) New Node → Lpoint = Null

1. Else

(a) Temp = Start

(b) While (Temp → Next Not Equal to Null)

(i) Temp = Temp → Next

(c) Temp → Rpoint = New Node

(d) New Node → Lpoint = Temp

1. Stop

Forward Traversal-

1. Start
2. If (Start Is Equal to Null)
   (i) Temp = Temp → Next
   (ii) Display "The List Is Empty"
   (b) Stop.
1. Initialize Temp = Start
2. Repeat the steps 5 And 6 until (Temp == Null)
3. Display "Temp → Data".
4. Temp = Temp → Next
5. Stop

(V) Backward Traversal-

1. Start
2. If (Start Is Equal to Null)
3. Display "The List Is Empty"
4. Stop.
5. Initialize Temp = Tail
6. Repeat the step 5 & 6 until (Temp == Null)
7. Display "Temp → Data".
8. Temp = Temp → Prev
9. Stop.

Dec 2016

Roll No. | | | | | | | | | | |

Total No. of Pages : 02

Total No. of Questions : 09

B.Tech.(3D Animation & Graphics) (2012 Onwards)

B.Tech.(CSE)/(IT) (2011 Onwards)

(Sem.–3)

DATA STRUCTURES

Subject Code :BTCS-304

Paper ID : [A1126]

Time : 3 Hrs.

Max. Marks : 60

INSTRUCTION TO CANDIDATES :

1. SECTION-A is COMPULSORY consisting of TEN questions carrying TWO marks each.
2. SECTION-B contains FIVE questions carrying FIVE marks each and students have to attempt any FOUR questions.
3. SECTION-C contains THREE questions carrying TEN marks each and students have to attempt any TWO questions.

SECTION A

1. **Write briefly:**

(a) What are the dangling pointers?

(b) How the complexity of an algorithm can be measured?

(c) What are the sparse matrices and how these are represented and stored?

(d) What are the advantages and disadvantages of linked lists?

(e) What are the circular queues? How insertion operation occurs in it?

(f) What is the difference between a general and a binary tree?

(g) What are the B-trees?

(h) How graphs are different from trees?

(i) Compare the complexities of linear and binary search.

(j) What are the different applications of heaps?

SECTION-B

2. Write short note on arrays.

3. What are the different representations of stacks? How recursion functions can be implemented using stacks?

4. What are the threaded binary trees? Discuss different operations of node insertion and deletion in these trees.

5. Discuss Depth First Search traversing technique for graphs with the help of suitable example. Write program for the same.

6. What is the use of Hash Tables? Discuss hashing in brief.

SECTION-C

7. What are doubly linked lists and what are their advantages? Write a program to insert and delete a node in a doubly linked list.

8. What are the different types of queues? Discuss basic operations for each type.

9. Discuss insertion sort in detail.

N-A

1 a) What are the dangling pointers?

When a programmer fails to intialize the pointer with a valid address it is called a dangling pointer or wild pointer. This can end up pointing anywhere in the memory that may include the program code itself or to the code of operating system which will lead to considerable result.

b) How the Complexity of an algorithm can be measured?

The complexity of an algorithm is a function $f(n)$ which measures the time and space used by an algorithm in terms of input size n. The complexity of an algorithm is a way to classify few efficient an algorithm is, compared to alternative ones. The focus is on how execution time increases with the data set to be Processed.

C) What are the advantages and disadvantages of linked list.

Matrices with a relatively high proportion of zero entries are called sparse matrices.

There are of 2 types:

(i) Triangular Matrix: The matrix in which all entries above the main diagnol are zero or where non zero entries can only occur on or below the diagnol.

ii) Tridiagonal matrix: The matrix in which non zero entries can only occur on the diagnol or on immediately above & below the main diagnol.

(i)
$$\begin{bmatrix} 1 & & \\ 2 & 3 & \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix}$$

(ii)
$$\begin{bmatrix} 1 & 7 & & \\ 2 & 2 & 5 & \\ & 3 & 3 & 6 \\ & & 5 & 4 \end{bmatrix}$$

d) What are the advantages and disadvantages of li...

Advantages:- 1. Insertion and deletion of nodes can ... implemented.
2. There is no need to define an initial size for a link...
3. Items can be added or removed from the middle of ...
4. Backtracking is possible into two way linked list.

Disadvantages:- 1. They use more memory than arrays because of the storage used by the pointers.
2. There Nodes in a linked list must be read in order from the begining as linked lists are inherently sequential access.

e) What are the circular queues? How insertion operat-ion occurs in it?

Circular queue is a linked list linear data structure. It follows FIFO principle. In FIFO the last node is connected back to the first node to make a circle.

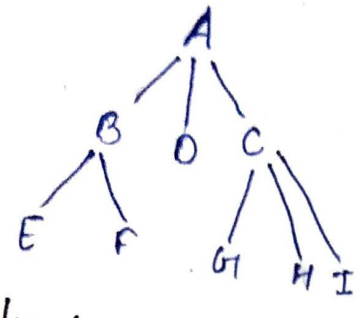
Steps for insertion in circular queue:

1. check whether queue is full. check ((rear == size-1 && front == 0) || (rear = front -))
2. If it is full then display queue is full. If queue is not full then, check if (rear == size-1 && front != 0) if it is true then set rear = 0 and insert element.

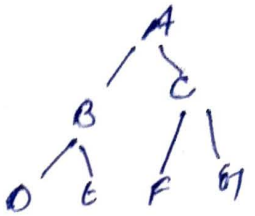
f) What is the difference between a general and a binary Tre...

General Tree	Binary Tree.
→ A general tree is a data structure in which each node can have infinite numbers of children.	→ It is a data structure in which each node can have at most two no... left and right.
→ A general tree can't be empty.	→ A binary tree can be empty.

...uctures of General trees are not ordered



subtrees of binary trees are ordered.



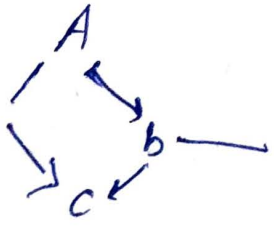
g) What are the B-Trees

B-Tree is a data structure that keeps data sorted and allows searches, insertions and deletion in logarithmic amortized for systems that read and write large blocks of DATA. It is most commonly used in database and file systems.

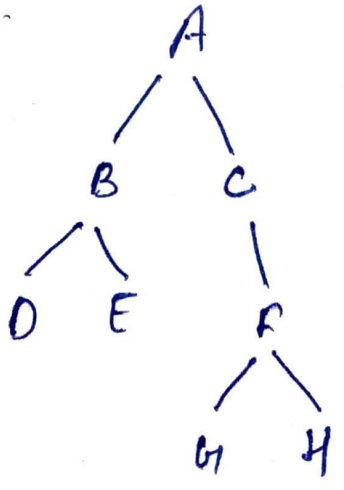
h) How graphs are different from Trees?

Graphs	Trees
→ In Graph there can be more than one path.	→ Tree is a special form of graph.
→ Graphs can have loops, circuits and self loop.	→ It does not have loops, circuits and self loop.
→ In Graphs there is no root concept of root node.	→ In tree there is exactly the root node and every child have only one parent

Eg:



Eg:



i) Compare the complexities of linear and binary sea

Linear search → O(n)

Binary search → O(log n)

Complexity of linear search is more than binary Search.

j) What are different applications of heap?

(i) Priority queues can be efficiently implemented using binary heap.

(ii) The data structure heap can be used efficiently to find the k^{th} smallest element in an array.

SECTION-B

2. Write a short note on Arrays.

An array in a data structure that contains a group of elements. Typically these elements are all of the same data type, such as integer or string. Arrays are commonly used in computer program to organize data so that a related set of value can be easily sorted or searched.

A Linear array is a list of finite no. of homogeneous data elements such that.

→ the elements of array are referenced respectively by an index set consisting of 'n' consective no's.

→ the elements of array are stored respectively in successive memory locations.

ARRAY:-

1	2	3	4	5	6

...ront array features are:

...ressing
...nsertion
...Deletion
(iv) Searching

3. What are the different representation of stacks? How recursion function can be implemented using stacks?

Stack is a list of element in which element may be inserted or deleted at only one end called TOP of the stack.

Stacks can be represented in 2 forms:

→ Array representation.
→ Link representation.

Implementation of recursive functions by stack:-

A Subprogram can contain both parameter and local variable. The parameter are variables which recieves values from objects in called program called arguements and which transmit values back to calling program.

Suppose our subprogram is reciever program.
Then each level of execution of subprogram may contain different values for parameters and local variables and return adderess.

Therefore, if ~~recieves~~ recursive program does call itself, then current values must be saved, since they will be again recalculated when the program is reactiv-ated.

4. What are the threaded binary trees? Discuss binary different operations of node insertion and deletion in these trees.

A binary tree is threaded by making all right child that will normally be null point to the inorder su... of the node and all left child pointers that would n... -lly to be null point to the inorder predecessor o... node.

Types of threaded binary tree:

- Single threaded:- Each node is threaded towards either the node in-order predecessor or successor (left or right) means all right null pointers will point to inorder predecessor.
- Double threaded:- Each node is threaded towards both the in-order predecessor and successor (left or right) means all right null pointers will point to inorder successor and all left null pointers will point to inorder predecessor.

- INSERTION:- Insertion in binary threaded tree is similar to insertion in binary tree but we will have to adjust the threads after insertion of each element. we start at the root and recursively go down the tree searching for the location in a binary tree to insert a new node.

DELETION:- In deletion, first the element to be deleted is searched and then there are cases for deleting the node in which element is found.

case 1: Node to be deleted has only 1 child.
After deleting the node, the Inorder successor of node and then copy information of this successor node are found out.

case 2:- Node to be deleted has two children
for this, we find inorder successor of node and then copy information of successor into node pointer.

leaf node to be deleted.
leaf node is to be deleted then its right child then after deletion, right pointer be should become thread pointing to its successor.

5. Discuss Depth first search traversing technique for graphs with the help of suitable example.

The DFS algorithm is a recursive algorithm that uses the idea of backtracking. It involves exhaustive searches for all the nodes by going ahead, if possible else by backtracking.

ALGORITHM:- This algorithm executes DFS on a graph G& begining at starting node A

(i) Initialize all nodes to the ready state STATUS = 1
(ii) PUSH the starting node A onto stack and change it's status to the waiting state, STATUS = 2.
(iii) Repeat steps 4&5 until stack is empty.
(iv) POP the node N of stack. Process N and change its status to the processed state, ie. STATUS = 3
(v) PUSH onto stack all the neighbour that are still in the ready state and change their status to waiting state.
(vi) Exit.

Example:-

STEP 1: Initially Push J onto stack.

STEP 2: Pop and print the top element J and then push the neighbours of J onto stack.
Print J Stack = D, K.

STEP 3: Pop and print the top element of K and push all the neighbours of K onto stack.
STACK = D, E, G. Print K

STEP 4: Pop and print the top element of G and push all the neighbours of G onto stack.
Stack: D, E, C Print G

STEP 5: Pop and Print the top element C and push onto Stack and neighbours of C onto stack.
Print C STACK = D, E, F

STEP 6: Pop and Print the top element E and Push the neighbours of E onto STACK.
Print E, stack = D

STEP 7: Pop and print the top element D and Push all the neighbours of E onto stack.
Print E, STACK = D

Pop and print the top element D and push all the neighbours are already printed - so, Stack will be already empty now. Print D, STACK = ϕ

TRAVERSAL ORDER = J, K, G, C, F, E, D

SECTION-C

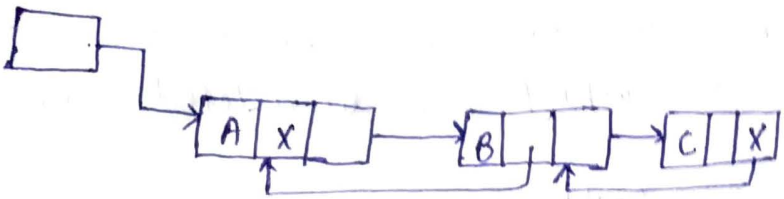
What are doubly linked lists and what are their advantages? WAP to insert and delete a node in a doubly linked list.

A double linked list is a linear collection of data elements called nodes where each node N is divided into 3 parts

(i) Information field:- INFO which contains the data element.

(ii) A pointer FORW (forward pointer) which contains the location of the next node in the LIST

(iii) A pointer field BACK which contains the location of the or preceeding node in LIST.



Advantages:-

(i) A doubly linked list can be transversed in both forward and backward direction.

(ii) The delete operation in doubly linked list is more efficient if pointer to the previous node to be deleted is given.

Operation on 2 way linked list.

a) INSERTION:- Suppose we are given the location LOCA & LOCB of adjacent nodes A and B in LIST and we want to insert a given ITEM of information b/w node A and B. NEW variable keeps track the location of new node N.

Algorithm to insert an element in doubly linked list.

INSWL (INFO, FORW, BACK, START, AVAIL, LOC)

1. [OVERFLOW?] if AVAIL = NULL then write OVERFLOW and exit.
2. [Remove node from AVAIL list and copy new data into node].
   Set NEW := AVAIL, AVAIL := FORW[AVAIL]
   INFO[NEW] := ITEM.
3. [Insert new node in list] set FORW[LOCA] = NEW
   FORW[NEW] = LOCB
   BACK[LOCB] = NEW
   BACK[NEW] = LOCA.
4. Exit.

b) Deletion :- Suppose we are given the location LOC of the node N in LIST which is to be deleted from CIRCULAR LIST.

DELTWL (INFO, FORW, BACK, START, AVAIL, LOC)

1. [Delete node]
   Set FORW[BACK[LOC]] := FORW[LOC]
   and
2. [Return deleted node to AVAIL list]
   Set FORW[LOC] := AVAIL and AVAIL := LOC
3. Exit.

8. What are the different types of queues? Discuss the operation for each type.

Queues are a linear list in which elements can be take place at only one end called the front and insertion can take place only at the other end called rear end.

...ent types of queues are:

Dequeues:- It is a linear list in which elements can be added or removed at either end but not in the middle. Dequeues are also called a double ended queue. Dequeue is maintained by circular array (DEQUEUE) with pointers LEFT & RIGHT which points to the 2 ends of DQUEUE.

Variations of Dqueue:

(i) Input restricted Dequeue:- It is the dequeue which allows insertion at only 1 end of the list but allows deletion at both ends.

(ii) O/p restricted Dequeue:- It is a dequeue which allows deletion at only 1 end of list but allows insertion at both ends of the list.

(11) Priority queue:- It is a collection of elements such that each element has been assigned a priority and such that the order in which elements are deleted & processed comes from the following rules:

a) An element of higher priority is processed before any element of lower priority.

b) 2 elements with the same Priority are processed according to the order in which they were added to the queue.

Operations on a Priority queue

Algorithm for deletion in Priority Queues.

This algorithm deletes and process the first element in a priority queue which appears in memory as a 1-way list.

1. Set ITEM = INFO[START]
2. Delete first node from the list.
3. Process ITEM.
4. exit

Algorithm for insertion into priority queue.

This algorithm inserts an ITEM with priority number N to a priority queue which is maintained in memory as a 1-way list.

1. Traverse the 1-way list until finds the node x whose priority no. exceeds N.
2. Insert ITEM in front of node x.
3. If no such node is found, insert ITEM as the element of the list
4. exit.

9) Explain insertion sort in detail.

Insertion sort is a simple sorting algorithm that builds the final sorted array one at a time. It is much less efficient on large lists than more advanced algorithms such as quicksort, heapsort, merge sort.

However, insertion sort provides several advantages;

* Simple implementation
* Efficient for small data sets
* Does not change the relative orders of elements with equal keys.

Set $A[0] = -\infty$

Repeat steps 3 to 5 for $k = 2$ to $k = N$

3. Set TEMP := A[k] and PTR := k-1

4. Repeat while TEMP < A[PTR]

   a) Set A[PTR+1] := A[PTR]

      [moves element forward]

   b) Set PTR := PTR - 1

      [End of Loop]

5. Set A[PTR+1] := TEMP

   [Inserts element at proper place]

   [End of Step 2 loop]

6. Exit.

Eg:- Sort the array A using insertion sort.

A = 77, 33, 44, 11, 88, 22, 66, 35

Sol:

Pass	A[0]	A[1]	A[2]	A[3]	A[4]	A[5]	A[6]	A[7]	A[8]
		77	33	44	11	88	22	66	35
k=1	$-\infty$	77	33	44	11	88	22	66	35
k=2	$-\infty$	33	77	44	11	88	22	66	35
k=3	$-\infty$	33	77	44	11	88	22	66	35
k=4	$-\infty$	33	44	77	11	88	22	66	35
k=5	$-\infty$	11	33	44	77	88	22	66	35
k=6	$-\infty$	11	33	44	77	88	22	66	35
k=7	$-\infty$	11	22	33	44	77	88	66	35
k=8	$-\infty$	11	22	33	44	66	77	88	35
k=9	$-\infty$	11	22	33	35	44	66	77	88

Sorted Array:- 11, 22, 33, 35, 44, 66, 77, 88

May 2017

Roll No. | | | | | | | | | | |

Total No. of Questions : 09

Total No. of Pages :02

B.Tech.(3D Animation & Graphics) (2012 Onwards)

B.Tech.(CSE)/(IT) (2011 Onwards)

(Sem.–3)

DATA STRUCTURES

Subject Code : BTCS-304

Paper ID : [A1126]

Time : 3 Hrs. **Max. Marks : 60**

INSTRUCTIONS TO CANDIDATES :

1. **SECTION-A is COMPULSORY consisting of TEN questions carrying TWO marks each.**
2. **SECTION-B contains FIVE questions carrying FIVE marks each and students have to attempt any FOUR questions.**
3. **SECTION-C contains THREE questions carrying TEN marks each and students have to attempt any TWO questions.**

SECTION-A

1. Write briefly :

(a) State the need of data structures.

(b) What are the advantages of Linked List over arrays?

(c) What is big 'O' notation?

(d) What is the difference between Linear and non-linear data structure?

(e) List the applications of Stack and Queue.

(f) Explain why binary search cannot be performed on a linked list.

(g) What is threaded binary tree?

(h) What is hashing?

(i) What is breadth-first traversal?

(j) Write **any two** applications of graph.

SECTION-B

2. Explain how stack is applied for evaluating an arithmetic expression.

3. Write an algorithm to insert an element at the specific position in an array.

4. Write an algorithm to find minimum and maximum element from a binary search tree.

5. How queues are represented in memory? Write their applications.

6. Write an algorithm for Binary search. What are its limitations?

SECTION-C

7. a) Write an algorithm to insert new node at the end of a Doubly Linked List.

   b) Convert the given Infix expression to Postfix expression using Stack and show the details of Stack at each step of conversion.

   Expression : (a – b ^ c * d) * (e – f / g).

   Note : ^ indicates exponent operator.

8. What are the tree traversal techniques? Explain each with an example.

9. Write short note on :

   a) Quick sort

   b) AVL Trees

Section - A May 2017

Ans briefly:

(a) Explain the need of data structure.

Data structure is a particular way of storing and organizing information in a computer so that it can be retrieved and used most productively. Different kinds of data structures are meant for different kinds of applications, and some are highly specialized to specific tasks.

(b) What are the advantages of linked list over arrays?

Ans Advantages of linked list over arrays are

1. Size of the list doesn't need to be mentioned at the beginning of the program, certainly dynamic memory allocation and deallocation
2. As the linked list doesn't have a size limit, we can go on adding new nodes (elements) and increasing the size of the list to any extent.

(c) What is big 'O' notation?

Ans Big 'O' notation is a mathematical notation that describes the limiting behaviour of a function when the argument tends towards a particular value or infinity... A description of a function in terms of big O notation usually only provides an upper bound on the growth rate of the function.

(d) What is the difference between linear and non-linear data structure?

Ans Linear data Structures

1. A type of data structure that arranges the data items in an orderly manner where the elements are attached adjacently.
2. Memory utilization is inefficient
3. Single-level.
4. Easier to implement
5. Ex: Array, linked list, queue, stack

Non-linear Data Structu

1. A type of data structure that arranges data in sorted order creating a relationship amon the data elements
2. Memory utilization is efficie
3. Multi-level.
4. Difficult to implement.
5. Ex: tree, graph.

(e) List the applications of stack and Queue.

Ans It can be used for sorting (eg:- bubble sort, insertion sort merge sort and etc.) QUEUE works on the principle of FIFO that is FIRST IN FIRST OUT. It is used to insert elements in a node (eg:- Circular queue, linear queue and etc). Linked list is used to insert elements in an array of unknown size

(f) Explain why binary search cannot be performed on a linked list.

Ans Yes, Binary search is possible on the linked list if the list is ordered and you know the count of elements in list. But while searching the list, you can access a single element at a time through a pointer to that node i.e either a previous node or next node

(g) What is threaded binary tree?

Ans A threaded binary tree defined as follows. "A binary tree is threaded by making all right child pointers that would normally be null point to the in-order successor of the node (if it exists), and all left child pointers that would normally be null pointers to the in-order predecessor of the node.

(h) What is hashing?

Ans A hash function is any function that can be used to map data of arbitrary size to fixed size values. The values returned by a hash function are called hash values, hash codes, digests, or simply hashes. The values are used to index a fixed-size table called a hash table.

(i) What is breadth-first traversal?

Ans Breadth-first search (BFS) is an algorithm for traversing or searching tree or graph data structures. It uses the opposite strategy as depth-first search, which instead explores the node branch as far as possible before being forced to backtrack and expand other nodes.

(j) Write any two applications of graph?

Ans Applications of graph

1. Computer Science:- In computer science, graph is used to represent network of communication, data organization, computational devices etc.

Social Science :- Graph theory is also widely used in sociology

Section B.

Explain how stack is applied for evaluating an arithmetic expression

Evaluation rule of a Postfix Expression states:

1. While reading the expression from left to right, push the element in the stack if it is an operand.
2. Pop the two operands from the stack, if the element is an operator and then evaluate it.
3. Push back the result of the evaluation. Repeat it till the end of the expression

Algorithm.

1. Add) to postfix expression.
2. Read postfix expression left to right until) encountered.
3. If operand is encountered, push it onto stack

   [End if]
4. If operator is encountered, Pop two elements.

   i.) A → Top element.

   ii.) B → Next to Top element.

   iii.) Evaluate B operator A

   Push B operator A onto stack
5. Set result = Pop
6. End.

Q3:- Write an algorithm to insert an element at the specific position in an array.

Ans Insert operation is to insert one or more data elements into an array. Based on the requirement, a new element can be added at the beginning, end, or any given index of array

Here, we see a practical implementation of insertion operation, where we add data at the end of the array.

Algorithm:

Let Array be a linear unordered array of MAX elements.
Let LA be a linear Array (unordered) with N elements and k
positive integer such that $k <= N$. Following is the algorithm
where ITEM is inserted into k^{th} position of LA.

1. Start
2. Set J = N
3. Set N = N+1
4. Repeat steps 5 and 6 while J >= k.
5. Set LA [J+1] = LA [J]
6. Set J = J-1
7. Set LA [k] = ITEM.
8. Stop

Q4:- Write an algorithm to find minimum and maximum element from a binary search tree.

Ans Algorithm for finding minimum or maximum element from a Binary search tree.

→ Approch for finding minimum element:

1. Traverse the node from root to left recursively until left is NULL
2. The node whose left is NULL is the node with minimum value.

→ Approch for finding maximum element:

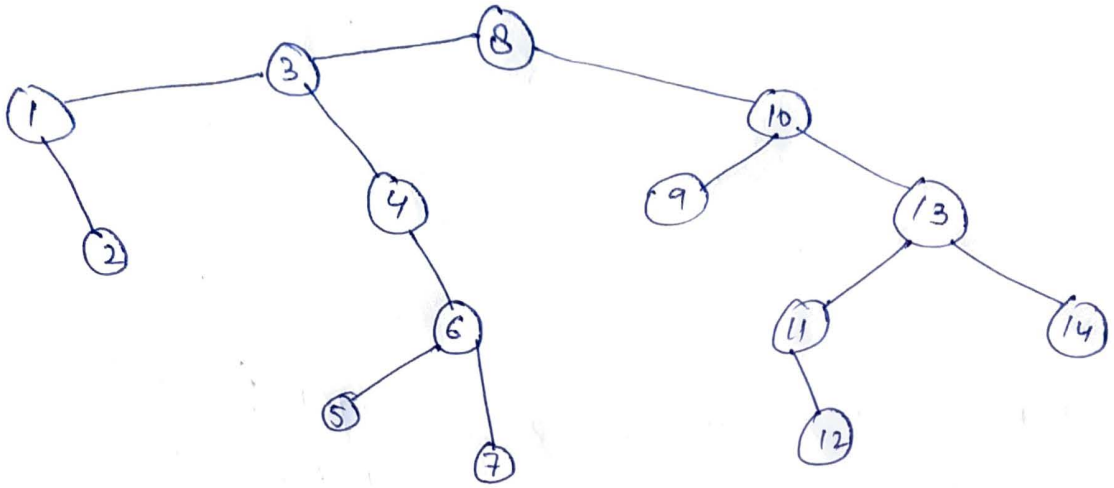
1. Traverse the node from root to right recursively until right is NULL.
2. The node whose right is NULL is the node with maximum value.

Explanation:

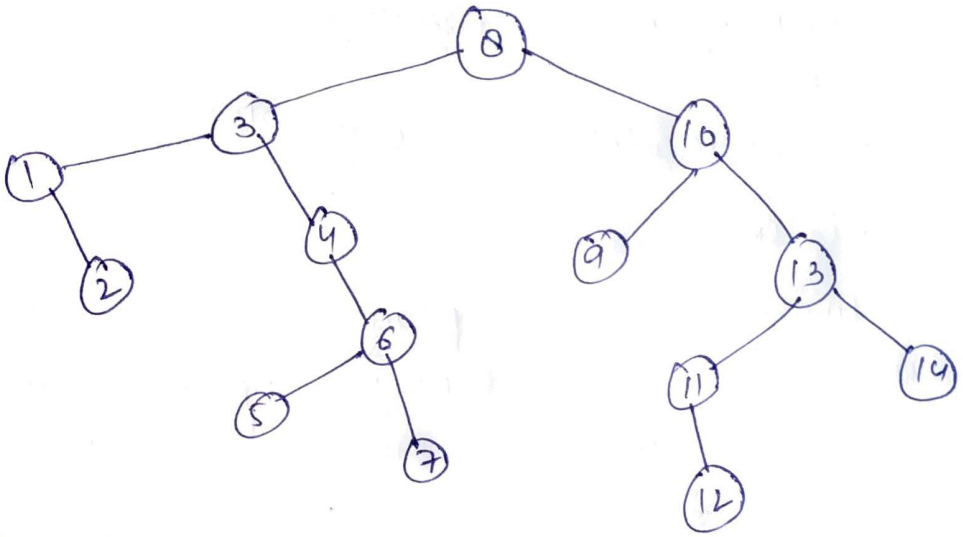
< finding minimum value in Binary search tree

May 2017

Start from root i.e 8.

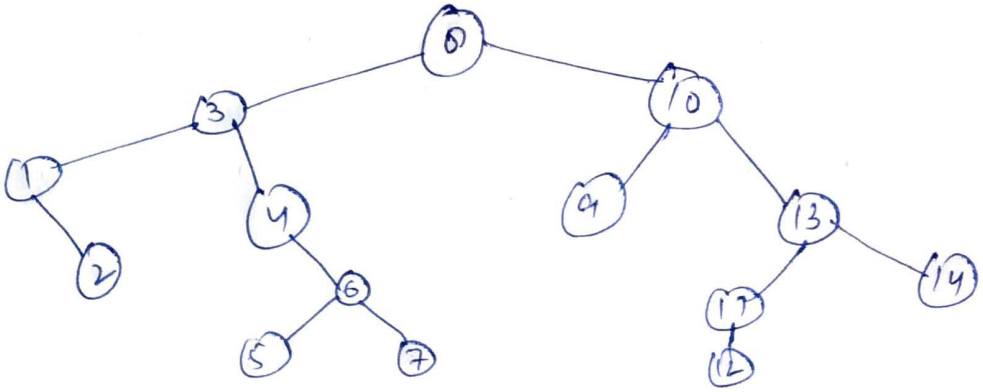


2. As left of root is not null go to left of root i.e 3



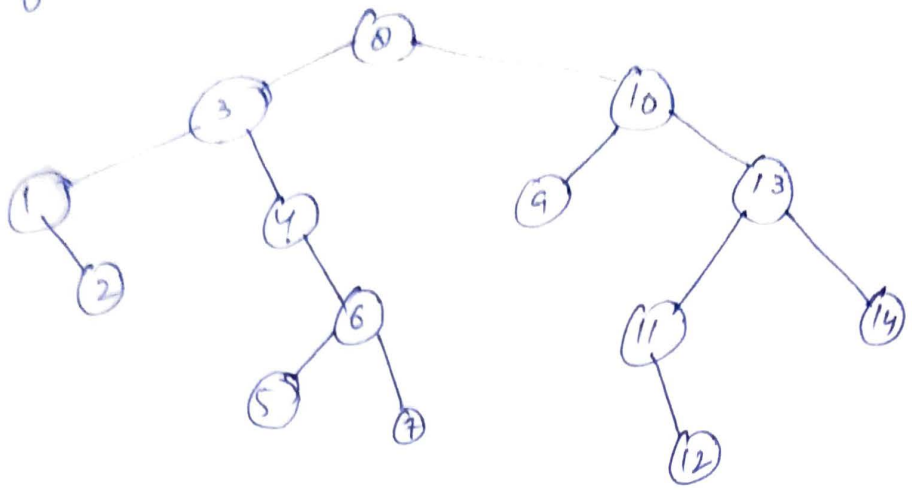
3. As left of 3 is not null go to left of 3 i.e. 1.

4. Now as the left of 1 is null thus for 1 is the minimum element



→ For finding Maximum value in Binary Search tree

1. Start from root i.e 8

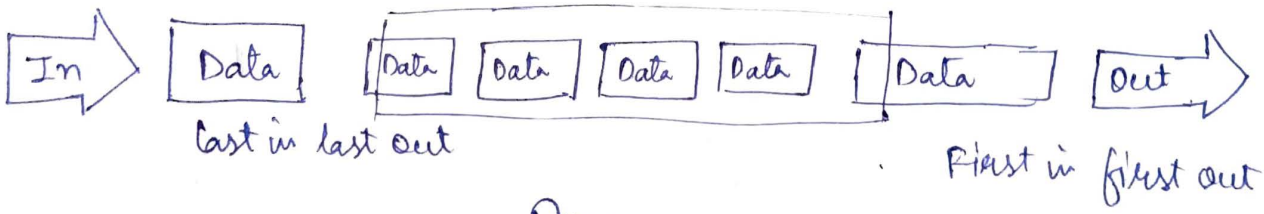


2. As right of root is not null go to right of root i.e 10.
3. As right of 10 is not null go to right of root i.e 13
4. As right of 13 is not null go to right of root i.e 14.
5. Now as the right of 14 is null therefore 14 is the maximum element.

DS: How queues are represented in memory? write their applications

Ans Queue Representation

As we now understand that in queue, we access both ends for different reasons. The following diagram given below tries to explain queue representation as data structure:—



Queue.

As in stack, a queue can also be implemented using Arrays, Linked-Lists, Pointers and structures. For the sake of simplicity, we shall implement queues using one-dimensional array.

Applications of Queue data structure:

1. Queue is useful in CPU scheduling, Disk scheduling, when multiple processes require CPU at the same time, various CPU scheduling algorithms are used which are implemented using Queue data structure.
2. When data is transferred asynchronously between two processes. Queue is used for synchronization. Example: IO Buffers, pipes, file IO, etc.
3. In print spooling, documents are loaded into a buffer and then the printer pulls them off the buffer at its own rate. Spooling also lets you place a number of print jobs on a queue instead of waiting for each one to finish.

Q6:- write an algorithm for Binary Search. What are its limitations?

Ans Algorithm for Binary Search.

BINARY_SEARCH (A, lower_bound, upper_bound, VAL)

Step 1: [INITIALIZE] SET BEG = lower_bound.

END = upper_bound, POS = -1

Step 2: Repeat Step 3 and 4 while BEG <= END

Step 3: SET MID = (BEG + END)/2

Step 4: If A[MID] = VAL.

SET POS = MID

PRINT POS

Go to step 6

ELSE IF A[MID] > VAL

SET END = MID - 1

ELSE

SET BEG = MID + 1

[END OF IF]
[END OF LOOP]

Step 5: If POS = -1
Print "Value is not present in the Array"
[END of If]
Step 6: EXIT

→ Limitations of Binary Search

1. Though faster than Sequential Search, binary search still requires an unacceptable number of accesses for data files with more than 1000 records.
2. Resorting the index after each record is inserted is not practical if the index cannot be kept in memory.

Section - C

Q7: (a) Write an algorithm to insert new node at the end of a Doubly linked list.

Ans Algorithm:

Step 1: IF PTR = NULL
write OVERFLOW
Go to step 11
[END OF IF]

Step 2: SET NEW_NODE = PTR.

Step 3: SET PTR = PTR → NEXT

Step 4: SET NEW_NODE → DATA = VAL

Step 5: SET NEW_NODE → NEXT = NULL

Step 6: SET TEMP = START

[END OF LOOP]

Step 9: SET TEMP -> NEXT = NEW_NODE

Step 10: SET NEW_NODE -> PREV = TEMP

Step 11: EXIT

Q8:- What are the tree traversal techniques? Explain each with an example.

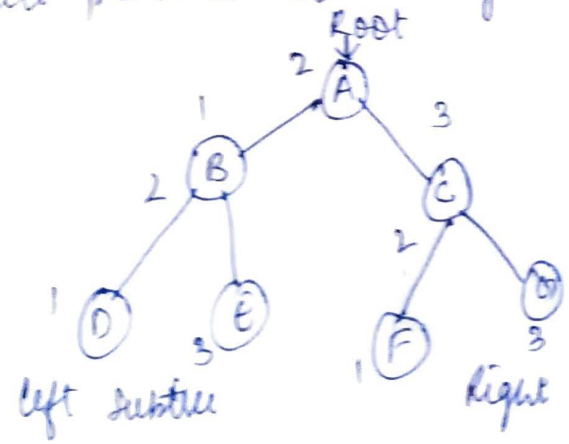
Ans: Traversal is a process to visit all the nodes of a tree and may print their values too. Because, all nodes are connected via edges (links) we always start from root (head) node. That is, we cannot randomly access a node in a tree. There are three ways which we use to traverse a tree.

1. In-order Traversal
2. Pre-order Traversal
3. Post-order Traversal

1. In order Traversal

In this traversal method, the left subtree is visited first, then the root and later the right sub-tree. We should always remember that every node may represent a subtree itself.

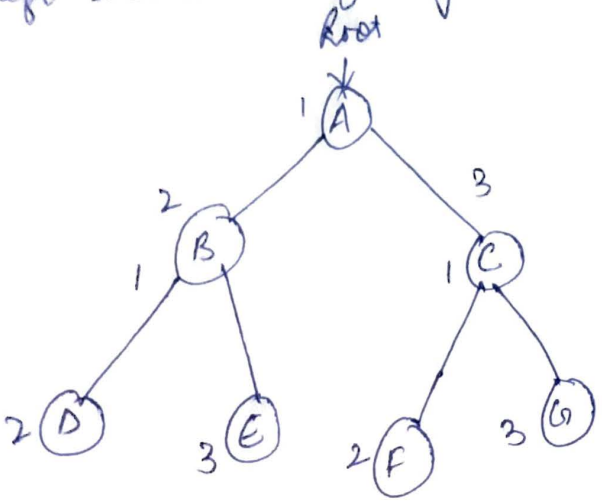
If a binary tree is traversal in-order, the output will produce sorted key values in an ascending order



Algorithm.
Inlili all nodes and traversal
Step 1: Recursively traverse left subtree
Step 2: Visit root node
Step 3: Recursively traverse right subtree

→ Pre-order Traversal

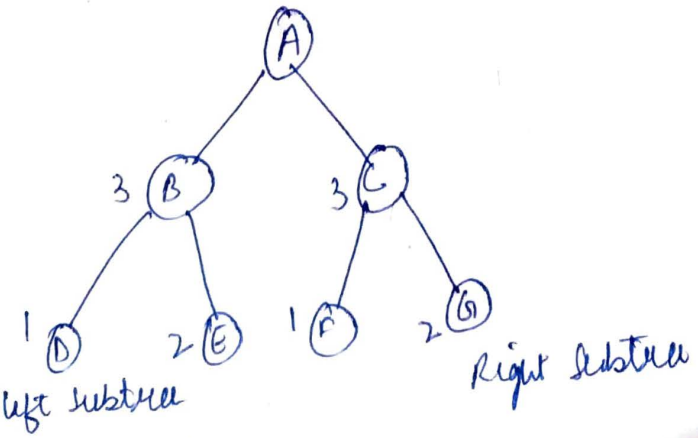
In this traversal method, the root node is visited first, then the left subtree and finally the right subtree



Algorithm
Step 1: Visit root node.
Step 2: Recursively traverse left subtree
Step 3: Recursively traverse right subtree

→ Post-order Traversal

In this traversal method, the root node is visited last, hence the name. First we traverse the left subtree, then the right subtree and finally the root node



algorithm:
Recursively traverse left subtree.
Recursively traverse right subtree
Visit Root node

Q4: Write short note on:

a) Quick Sort

Ans Quick Sort :- Quick sort is a divide and conquer algorithm. It creates two empty arrays to hold elements less than the pivot value, and elements greater than the pivot value, and then recursively sort the sub arrays. There are two basic operations in the algorithm, swapping items in place and partitioning a section of the array.

b) AVL Trees.

Ans AVL Trees :- AVL tree is a binary search tree in which the difference of heights of left and right subtrees of any node is less than or equal to one. The technique of balancing the height of binary trees was developed by Adelson, Velskii, and Landi and hence given the short form as AVL tree or Balanced Binary Tree.

Dec 2017

Roll No. | | | | | | | | | | |

Total No. of Questions: 09 **Total No. of Pages: 02**

B. Tech. 3D Animation & Graphics/CSE/IT (Sem. 3)
DATA STRUCTURES
Subject Code: BTCS-304
Paper ID: A1126

Time: 3 Hrs. **Max. Marks: 60**

INSTRUCTIONS TO CANDIDATES:

1. **Section A is COMPULSORY consisting of TEN Questions carrying TWO marks each.**
2. **Section B contains FIVE questions carrying FIVE marks each and students have to attempt any FOUR questions.**
3. **Section C contains THREE questions carrying TEN marks each and students have to attempt any TWO questions.**

SECTION A

1.
   a. What is usage of pointers?
   b. Relate data structures with data types?
   c. Discuss double Linked List?
   d. What are postfix expressions?
   e. Define debugging?
   f. Discuss B-trees?
   g. What is adjacency Matrix?
   h. Write a short note on Hash Tables?
   i. What are advantages of insertion sort?
   j. What are non recursive procedures?

SECTION B

2. Discuss some of the common operations that can be performed on data structures by taking suitable example?
3. Define recursion? Which data structure can be used to implement it?
4. Discuss various operations on queues?
5. Give the brief introduction to threaded Binary trees?
6. Illustrate the concept of breadth-first search traversing of graph?

SECTION C

7. Write an algorithm to implement the stacks using Link List?

8. How a linear array is represented in memory? Explain the program which reads two matrixes?

9. Write an algorithm to sort an array of integers in the descending

Section-A DEC 2017

a) what is usage of pointers?

ans-a) ① Pointers are memory variables used to store the memory addresses of other variables.

② values stored on the pointers are hexadecimal values

③ They save memory space.

④ used to help locating exact value at exact loc-tion.

⑤ structures can be handled very easily.

b) Relate data structure with data types?

ans-b) data structure is the collection of different kinds of data. where as data type is a kind or form of a variable which is being used throughout the program.

c) Double linked list & discuss its functions?

ans-c) A double linked list is a linked data structure that consists of a set of sequentially linked records called nodes. & in Double linked list, every node has a link to its previous node & next node.

d) what are postfix expression?

ans-d) postfix expression are used for representing algebric expressions. The expression written in postfix form are evaluated faster & compared to infix notation as parenthesis are not required in postfix.

e) define debugging?

ans-e) debugging is the process of identifying & removing errors from computer hardware & software.

f) Discuss B-trees?

ans f) B-tree is a self-balancing tree data str... tree that maintains stored data & allows se... sequential access, insertions, & deletions in ~~log~~ log ... time. The B-Tree has a node can only have more than two children.

g) what is adjacency matrix?

ans g) The adjacency matrix is also called as connection matrix, is a maxtrix with rows & columns labelled by graph vertices. with a 1 or 0 in position according to whether are adjacent or not.

h) write a short note on Hash Tables?

ans h) Hash table is a data structure which stores data in an associative manner. In a hash table, data is stored in an array format, where each data value has its own unique index value.

i) what are advantages of insertion sort?

ans (i) advantages →

1. Simple to code.
2. very good performance with small lists.
3. very good when the list is almost sorted.
4. very memory efficient.
5. Good with sequential data.

j) what are non-recursive procedures?

ans j) non-recursive means not involving a function calling itself, & thereby consuming constant stack space. A recursive function are procedures in a programming language, whose implementation does not refer itself.

Section B

Q-3 Define Recursion? what data structures can be used to implement it?

ans-3 The process in which a function calls itself directly or indirectly is called recursion & the corresponding function is called as recursive function.

data structure that can be used to implement it is → Stack has LIFO (last in first out) property; Therefore, it knows to whom it should return when the function has to return. on the other hand recursion make use of the system stack for storing the return address of the function call.

Every recursive function has its equivalent iterative (non-recursive) function.

Q-2 Discuss some of the common operations that can be performed on data structure & taking suitable examples.

ans-2 The operations performed on data structures are

1. Insertion
2. deletion
3. searching.
4. Traversing
5. sorting
6. merging.

(1) Insertion → Insertion means addition of a new data element in a data structure.

(2) deletion → deletion means removal of a data element from a data structure if it is found.

(3) searching → searching involves searching for a specified data element in a data structure.

(4) Traversing → accessing each records exactly once so that certain items in the record may be processed.

(5) Sorting → managing the data or records in some logical order.

(6) merging → combining the record in two different stored files into a single sorted file.

Q-4 Discuss various operations on queue?

ans-4 Queue → A queue is a container of objects that are inserted & removed according to the FIFO. In queue only two operations are allowed enqueue & dequeue.

Queue operations may ~~one~~ involve initializing or defining the queue, utilizing it & then erasing it from the memory.

(1) Enqueue ()

(2) Dequeue ()

(1) Enqueue () → add (store) an item to the Queue.

(2) dequeue() → it removes (access) an item from the given Queue.

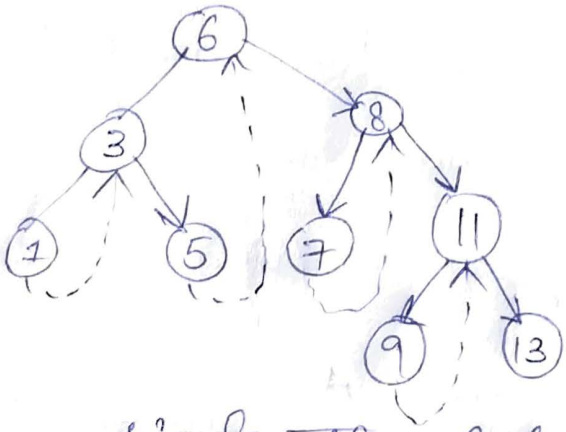
Q-5 Give the brief introduction to threaded Binary tree?

ans-5 The idea of threaded Binary tree is to make inorder traversal faster & do it without stack & without recursion. A binary tree is made threaded by making all right child pointers that would normally be NULL point to the inorder successor of the node.

There are two types of threaded binary trees →

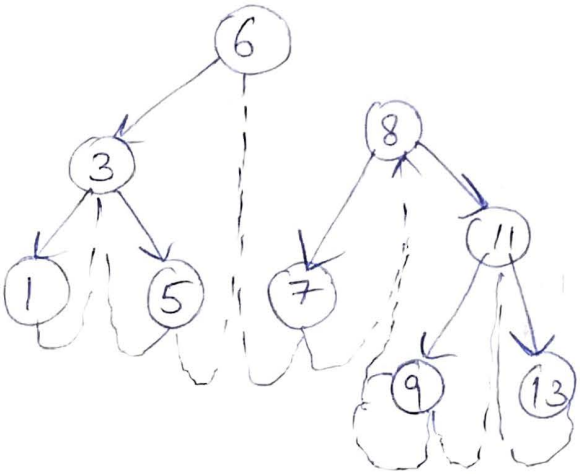
1. single threaded →
2. Double threaded →

(1) Single threaded → where the NULL right pointers i.e. made to point to inorder successor.



Single threaded

(2) double threaded → when both left & right NULL pointer are made to point inorder predecessor, inorder successor



Double - Threaded

Q-6 Illustrate the concept of breadth-first search traversing of graph?

ans-6 Breadth first search is a graph traversal algorithm that starts traversing the graph from root node & explores all the neighbouring nodes. Then, it selects the nearest nodes. The algorithm follows the same process for each of the nearest node until it finds the goal.

The algorithm is given below→

Step 1→ Set STATUS=1 (ready state) for each node in G.

Step 2→ Enqueue the starting node A & set its status = 2

Step 3→ Repeat steps 4 & 5 until Queue is empty.

Step 4→ Dequeue a node N. process it set its STATUS = 3 (processed state).

Step 5→ Enqueue all the neighbours of N that are in the ready state.

(whose STATUS=1) & set their STATUS = 2.

(waiting state).

(End of loop).

Step 6→ Exit.

Section-C

Q7 write an algorithm to ~~represent~~ implement the stack using link list?

Ans-7 Basic operations performed on stack are →

(1) PUSH → storing an element on the stack.

(2) POP → Removing an element on the stack.

(1) algorithm for PUSH →

```
procedure
begin push : stack, data
    if stack is full
    return null
    end if

    top ← top + 1
    stack [top] ← data.

    end procedure
```

(2) algorithm for POP →

```
begin procedure pop : stack.
    if stack is empty
    return null

    end if
    data ← stack [top]
    top ← top - 1
    return data.
end procedure.
```

Q-8 How is a linear array is ~~represented~~ represented in memory & explain the program which reads two matrixes?

ans-8 Representation of linear array in memory to implement array data structure, memory bytes must be reserved & the accessing functions must be coded.

```
#include <iostream.h>
int main()
{
  int a[10][10], b[10][10], result[10][10], r1,
      c1, r2, c2, i, j, k;
  int m, n, c, d, first[10][10], second[10][10],
  sum[10][10];
cout << " Enter the no of rows & column of
        matrix ";
cin>>m>>n;
cout<<" Enter the elements of first matrix";
for (c=0; c<m ; c++)
  for (d=0; d<n; d++)
    cin>> first[c][d];
cout<<" Enter the elements of second matrix";
for (c=0; c<m; c++)
for (d=0; d<n ; d++)
  cin>>second[c][d];
for (c=0; c<m ; c++)
```

```
for (d=0; d<m; d++)
Sum[c][d] = first[c][d] + second[c][d];
cout << "Sum of entered matrices";
for (c=0; c<m; c++)
{
for (d=0 ; d<n; d++)
cout << Sum[c][d] << "\t";
cout <<endl;
}
return 0;
}
```

Q-9 write an algorithm to sort an array of integers of the descending?

ans algorithm

step1→

May 2018

Total No. of Questions: 09

B.Tech. (3D Animation & Graphics) (2012 Onwards) / B.Tech. (CSE)/(IT) (2011 Onwards) (Sem. – 3)

DATA STRUCTURES

M Code: 56594
Subject Code: BTCS-304
Paper ID: [A1126]

Time: 3 Hrs. **Max. Marks: 60**

INSTRUCTIONS TO CANDIDATES:
1. **SECTION-A is COMPULSORY consisting of TEN questions carrying TWO marks each.**
2. **SECTION-B contains FIVE questions carrying FIVE marks each and students have to attempt any FOUR questions.**
3. **SECTION-C contains THREE questions carrying TEN marks each and students have to attempt any TWO questions.**

SECTION A

1. Write briefly:

a) What are the applications of Queue?

b) How Trees are represented in memory?

c) Define circular Queue.

d) What are B trees?

e) What do you mean by depth of tree?

f) What is the linked representation of stacks?

g) Define header nodes.

h) Write the drawbacks of DQUEUES.

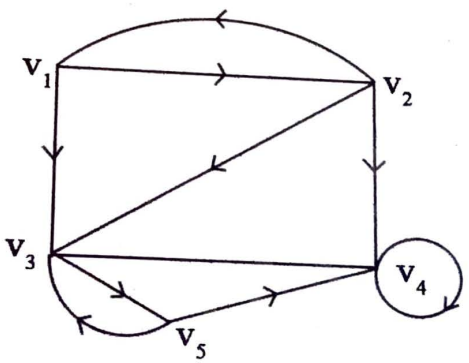
i) What are the applications of heaps?

j) What is the complexity of insertion sort?

2. Write an algorithm to implement Merge sort.
3. Make a binary search tree by considering the following eight numbers:
   50, 24, 38, 24, 67, 40, 60, 52.
4. Write an Algorithm to traverse a graph using Depth First Search.
5. Explain Radix sort.
6. Build a heap H from the following list of numbers:
   40, 65, 15, 48, 14, 50, 17, 22.

SECTION C

7. Write an algorithm to implement Quick sort. Write the steps to sort the following elements by quick sort method:
   17, 28, 6, 87, 46.
8. a) Suppose a binary tree T is in memory. Write a procedure to delete all the terminal nodes.
   b) Write an algorithm to insert a new node in linked list.
9. a) Consider the directed Graph G.



   i) Find indegree and outdegree of each node.
   ii) Find number of simple paths.
   iii) Is there any source or sink?

   b) Are B trees of order 2 are full binary trees? If yes, explain how.

DATA STRUCTURES

QUESTION PAPER

MAY 2018

Ques 1 → Write briefly:

a) What are the applications of Queue?

Ans → A queue is a (ordered) collection of items, where all insertions are made to the end of sequence and all deletions always are made from the beginning of the sequence. It is referred to as First-in-First-Out (FIFO). The various applications of queue are classified as:-

1. Queues in Real life situations
   i) Lines in daily life → The lines in real life are supposed to be queue.
2. Queues in data structures
   i) In Tree data structures → Various algorithms are solved by using queues like level-order traversing.
   ii) In graphs → The breadth first search in graphs is done using queues.
   iii) In Sorting → Priority Queues are used in heap sorting
3. Queues in Computer Hardware
   i) Printing
   ii) Computer Network
4. Queues in Operating System Simulation
   i) Job scheduling by operating system
   ii) Multiprogramming system

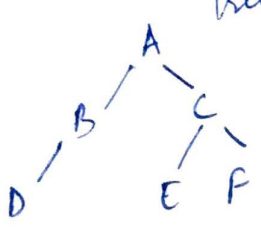
b) How trees are represented in memory?

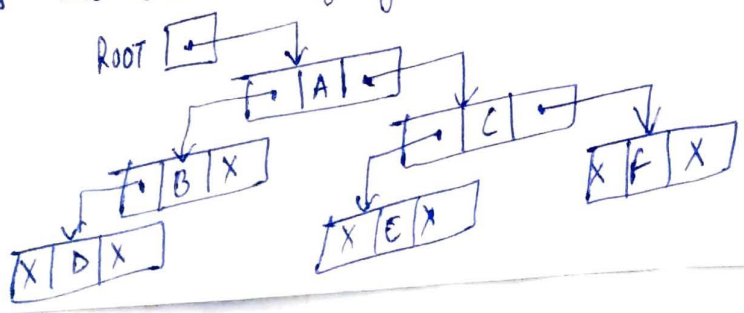
Ans → Trees can be represented in two different ways

i) Linked Representation of Binary Trees → Any binary tree T can be represented using three parallel arrays, INFO, LEFT and RIGHT, and a pointer variable ROOT. Each node N of T will correspond to a location K such that:-

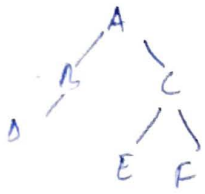
   i) INFO[K] contains the data at node N.
   ii) LEFT[K] contains the location of left child of N.
   iii) RIGHT[K] contains the location of right child of N

Tree





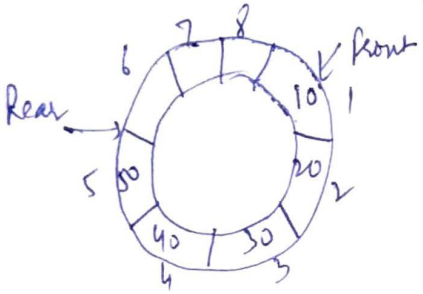
b) Sequential Representation of Binary Trees :
Only a single linear array TREE as follows:-
a) The root R of T is stored in TREE[1]
b) If a node N occupies TREE[K], then its left child is stored in TREE[2*K+1] and its right child is stored in TREE[2*K+2]



1	A
2	B
3	C
4	D
5	
6	E
7	F

c) Define circular Queue?

Ans Circular queue is a linear data structure in which the operations are performed based on FIFO (First In First Out) principle and the last position is connected back to the first position to make a circle. It is also called 'Ring Buffer'



In a normal queue, we can insert elements until queue becomes full. But in circular queue, once queue becomes full, we can insert the next element if there is a space in front of queue. This is not possible in case of normal queue.

Operations on Circular Queue

→ Front → get the front item from queue.
→ Rear → get the last item from queue.
→ enQueue(value) → for inserting an element into circular queue
→ DeQueue(value) → for deleting an element from circular queue.

d) What are B trees?

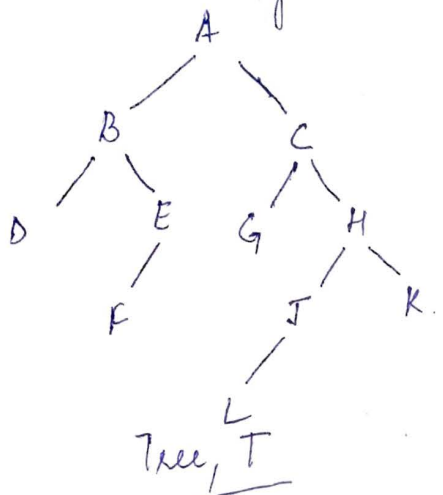
Ans → A B-tree of order m, if non empty
i) The root has at least two child nodes and at most m child nodes.
ii) the internal nodes except the root have at least $\lceil \frac{m}{2} \rceil$ child nodes and at most m child nodes.

number of keys in each internal node is one less than the ...ies of child nodes and these keys partition the keys in the ...arch trees of the node in a manner similar to that of m-way) all leaf nodes are on the same level.

m-way search trees have the advantage of minimizing file accesses due to their restricted height. However, it is essential that the height of the tree be kept as low as possible and therefore there arises the need to maintain balanced m-way search trees. such a balanced m-way search tree is B-tree

e) What do you mean by depth of the tree?

Ans→ The depth (or height) of a tree T is the maximum number of nodes in a branch of T. This turns out to be 1 more than the largest level number of T. for example,



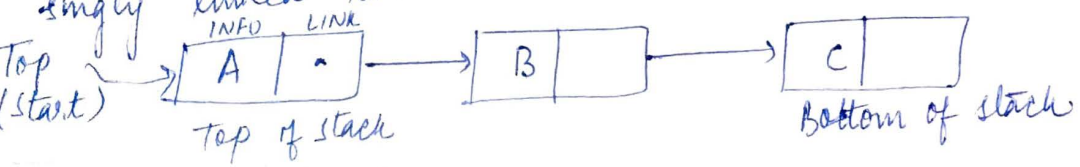
Tree, T

Level number of root is 0.
So, level of tree T is 4.
So, the ~~depth~~ depth of the tree is 4+1=5.

⇒ Also the maximum nodes in branch A→C→H→J→L is 5.

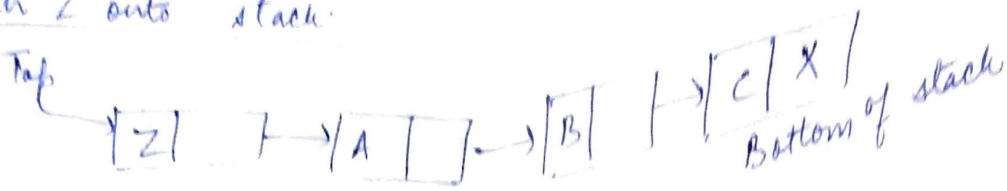
f) What is linked representation of stacks?

Ans→ The INFO fields of the nodes hold the elements of the stack and the LINK fields hold pointers to the neighboring element in the stack. The START pointer of the linked list behaves as the TOP pointer variable of the stack and the null pointer of the last node in the list signals the bottom of stack. The linked representation of a stack, commonly termed linked stack is a stack that is implemented using a singly linked list.

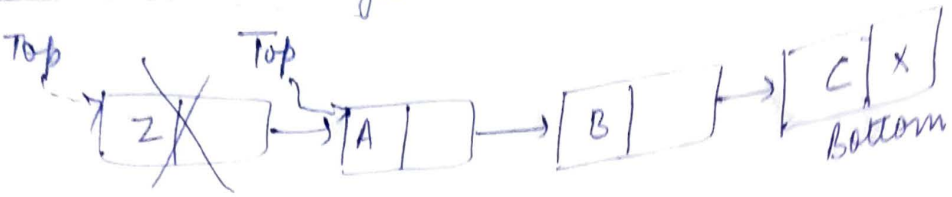


A push operation into STACK is accomplished by inserting into the front or start of the list and a pop operation is undertaken by deleting the node pointed to by the START.

Push Z onto stack.



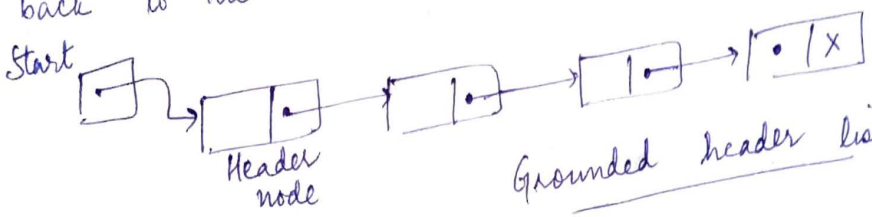
Pop Z onto stack given above

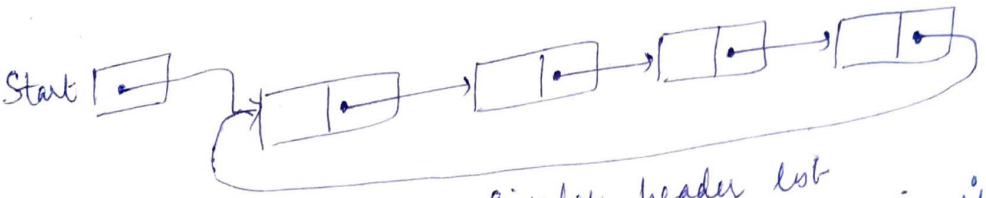


g) Define header nodes.

Ans A header linked list is a linked list which always contains a special node, called the header node, at the beginning of the list. The following are two kinds of widely used linked header lists:-

1. A grounded header list is a header list where the last node contains the null pointer.
2. A circular header list is a header list where the last node points back to the header node.

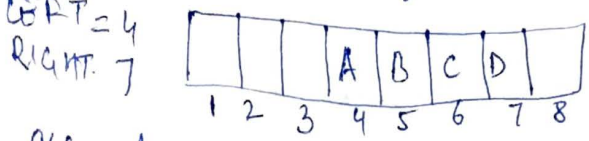




The utility of header node in a linked list is to simplify the functions of adding and deleting elements in the list. One other use of the header node could be to store in it data value the number of elements in the list.

The drawbacks of DQUEUES.

A deque is a linear list in which elements can be added or removed at either end but not in the middle. This is also known as double-ended queue.



There are two variations of deque given by input-restricted deque and output-restricted deque. The various drawbacks of deque includes the complexity while implementation of dequeue. Other includes the cases when there is overflow, that is, when an element is to be inserted into a deque which is already full and when there is underflow, that is, when an element is to be deleted from a deque, which is empty.

i) What are the applications of heap?

Ans The various applications of heap includes:-

1) Heapsort → one of the best sorting methods being in-place and with no quadratic worst-case scenarios.
2) Selection Algorithms → A heap allows access to the min or max element in constant time.
3) Graph algorithms → Run time will be reduced by polynomial order if heaps are used for traversal.
4) Priority Queues → A priority queue can be implemented with heap.
5) k-way merge → A heap data structure is useful to merge many already-sorted input streams into a single sorted output stream.
6) Order statistics → The heap data structure can be used efficiently find the kth smallest (or largest) element in an array.

j) What is the complexity of insertion sort?

Ans → The function $f(n)$ of comparisons in insertion sort algorithm can be easily computed. First of all, the worst case occurs when the array A is in reverse order and the inner loop must use the maximum number of $k-1$ of comparisons. Hence

$$f(n) = 1+2+\ldots+(n-1) = \frac{n(n-1)}{2} = O(n^2)$$

Furthermore, one can show that, on the average, there will approximately $(k-1)/2$ comparisons in the inner loop. According to the average case,

$$f(n) = \frac{1}{2} + \frac{2}{2} + \ldots + \frac{n-1}{2} = \frac{n(n-1)}{4} = O(n^2)$$

Algorithm	Worst case	Average case
Insertion Sort	$O(n^2)$	$O(n^2)$.

SECTION-B

Ques 2 → Write an algorithm to implement Merge sort.

Ans → Merge-sort algorithm for sorting an array A has the following property. After pass k, the array A will be partitioned into sorted subarrays where each subarray, except possibly the last, will contain $L = 2^k$ elements. Hence the algorithm requires atmost $\log n$ passes to sort an n-element array A. The merge sort algorithm divided into two parts MERGEPASS and MERGESORT. The MERGEPASS procedure applies to an n-element array A which consists of a sequence of sorted subarrays.

MERGEPASS (A, N, L, B) ⇒ The N-element array A is composed of sorted subarrays where each subarray has L elements. The procedure merges the pair of subarrays of A and assigns them to the array B.

1. Set $Q = INT(N/2*L))$, $S = 2*L*Q$ and $R = N - S$
2. Use Procedure to merge the Q pairs of subarrays

   Repeat for $J = 1, 2, \ldots Q$:

   a) Set $LB = 1 + (2*J - 2)*L$. [Finds lower bound]

   b) call MERGE(A, L, LB, A, L, LB+L, B, LB).

   End of Loop
3. If $R \leq L$, then

   Repeat for $J = 1, 2, \ldots, R$:

   Set $B(S+J) = A(S+J)$

Call MERGE (A, L, S+1, A, R, L+S+1, B, S+1).

Return.

The formal MERGESORT algorithm is :-

MERGESORT (A, N) => This algorithm sorts the N- element array A using an auxiliary array B.

1. Set L=1
2. Repeat steps 3 to 6 while L<N:
3. Call MERGEPASS (A, N, L, B).
4. Call MERGEPASS (B, N, 2*L, A).
5. Set L := 4* L
6. Exit

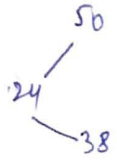
Ques3 → Make a binary search tree by considering the following eight numbers:

50, 24, 38, 24, 67, 40, 60, 52

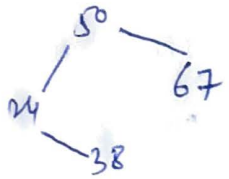
Ans→ As there are eight numbers, so the binary search tree will be constructed in 8 stages.

1) No. 50

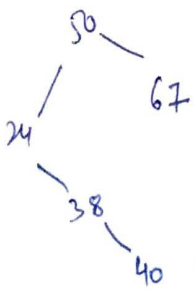
2) No. 24



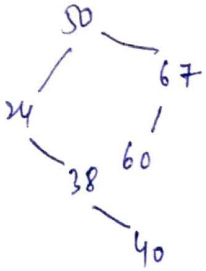
3) No. 38



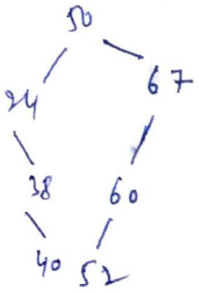
4) No. 67



5) No. 40.



6) No. 60



7) No. 52.

Since 24 is repeated, so it was completed in seven steps.

Ques 4 → Write an algorithm to traverse a graph using depth first search.

Ans → The general idea behind a depth-first search beginning at a sta. node A is as follows. First we examine the starting node A. Then we examine each node N along a path P which begins at A; that is, we process a neighbor of A, then a neighbor of a neighbor of A, and so on. After coming to a 'dead end', that is, to the end of the path P, we backtrack on P until we can continue along another path P'. And so on. In this we use queues for traversing. Again, a field STATUS is used to tell us the current status of a node. The algorithm is as follows:-

⇒ This algorithm executes a depth-first search on a graph G beginning at a starting node A.

1. Initialize all nodes to the ready status (STATUS = 1)
2. Push the starting node A onto STACK and change its status to the waiting state (STATUS = 2).
3. Repeat steps 4 and 5 until STACK is empty.
4. Pop the top node N of STACK. Process N and change its status to the processed state (STATUS = 3)
5. Push onto STACK all the neighbors of N that are still in the ready state (STATUS = 1), and change their status to the waiting state (STATUS = 2)

[End of step 3 loop]

6. Exit.

The above algorithm will process only those nodes which are reachable from the starting node A.

Explain Radix Sort

Radix sort is the method that many people intuitively use or begin to use when alphabetizing a large list of names. That is, suppose a card sorter is given a collection of cards where each card contains a 3-digit number punched in columns 1 to 3. The cards are first sorted according to the units digit. On the second pass, the cards are sorted according to the tens digit. On the third and last pass, the cards are sorted according to the hundreds digit. For example, the cards are

348, 143, 361, 423, 538, 128, 321, 543, 366

First pass,

Input	0	1	2	3	4	5	6	7	8	9
348				143			366		348	
143		361		423					538	
361		321		543					128	
423										
538										
128										
321										
543										
366										

Second pass

Input	0	1	2	3	4	5	6	7	8	9
361			321		143		361			
321			423		543		366			
143			128	538	348					
423										
543										
366										
348										
538										
128										

Third pass

Input	0	1	2	3	4	5	6	7	8
321				321					
423					423				
128		128							
538						538			
143		143							
543						543			
348			348						
361			361						
366			366						

So, the final numbers after sorting will be

128, 143, 321, 348, 361, 366, 423, 538, 543.

On the worst case, the complexity will be $O(n^2)$ and in the best case, the complexity will be $O(n \log n)$.

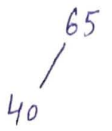
Ques 6 : Build a heap H from the following list of numbers.

40, 65, 15, 48, 14, 50, 17, 22.

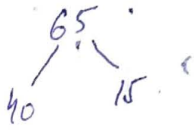
Ans

a) No. 40

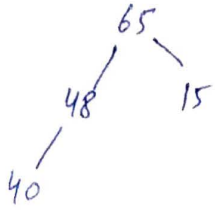
b) No. 65



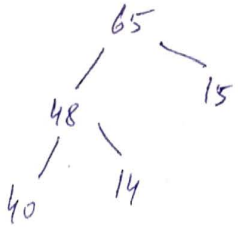
c) No. 15



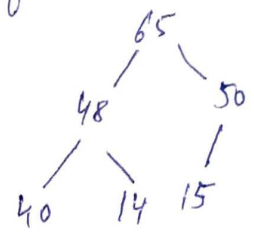
d) No. 48



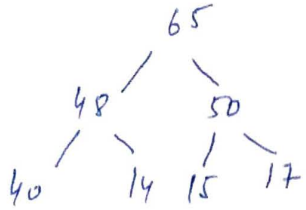
e) No. 14



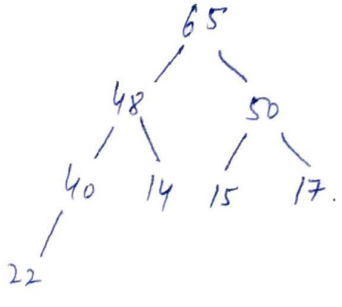
f) No. 50



g) 17



h) No. 22.



Heap.

Write an algorithm to implement Quick sort. Write the steps and the following elements by quick sort method.

17, 28, 6, 87, 46.

Ans → Quicksort is an algorithm of the divide-and-conquer type. That is, the problem of sorting a set is reduced to the problem of sorting two smaller sets. The Quicksort algorithm is given below. It uses algorithm QUICK (A, N, BEG, END, LOC) where A is an array with N elements. Parameters BEG and END contain the boundary values of the sublist of A to which this procedure applies. LOC keeps track of the position of the first element.

QUICKSORT → This algorithm sorts an array A with N elements.

1. [Initialize] TOP = NULL
2. If N > 1, then TOP = TOP + 1, LOWER [1] = 1, UPPER [1] = N.
3. Repeat steps 4 to 7 while TOP ≠ NULL.
4. Set BEG = LOWER [TOP], END = UPPER [TOP],
   TOP = TOP - 1
5. Call QUICK (A, N, BEG, END, LOC)
6. If BEG < LOC - 1, then:
   TOP = TOP + 1, LOWER [TOP] = BEG,
   UPPER [TOP] = LOC - 1
   End of If structure.
7. If LOC + 1 < END, then
   TOP = TOP + 1,
   LOWER [TOP] = LOC + 1,
   UPPER [TOP] = END
   End If
8. Exit.

Quick sorting on the elements given by

17, 28, 6, 87, 46

1st step - scan from right to left, found 6 < 17, Interchange

6 28 17 87 46

2nd step - scan from left to right, found 28 > 17, Interchange

After Pass1.

6 | 17 | 28 87 46

First | second

After Pass 2 ⇒ 6 17 | 28 | 87 46

Subset

Pass 3 6 17 28 | 87 46

Scan from Right to left, found 46 < 87, Interchange

6 17 28 46 87, final list after Quick sorting.

The average case complexity of Quick sorting is O(n log n)

But for the worst case is $O(n^2)$.

Ques 8 → a) Suppose a binary tree T is in memory. Write a procedure to delete all the terminal nodes.

Ans The following is the function to delete terminal nodes from binary trees :-

```
void deleteleaves (struct node* root, struct node *prev)
{
if (root)
  {
```

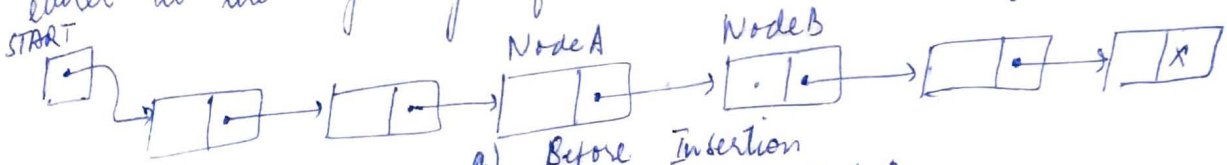
```
(root -> left == NULL && root -> right == NULL)
{
if (prev -> item) == root -> item)
    prev -> left = NULL;
else
    prev -> right = NULL;
printf ("\n\n % d is being deleted ...", root -> item);
free (root);
return;
}
delete leaves (root -> left, root);
deleteleaves (root -> right, root);
}
}.
```

the call to this function will be deleteleaves (root, root).

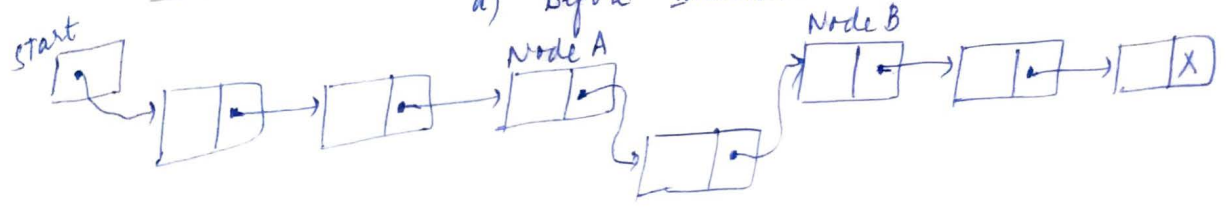
⇒ The nodes with no successors are called terminal nodes. In this function. The terminal nodes are found by checking its left and right child. If there is no children, then that node is terminal node and can be deleted.

b) Write an algorithm to insert a new node in linked list.

Ans→ Insertion of a new node in a given linked list can be done either at the beginning of the node or after a given node.



a) Before Insertion



A formal statement of the algorithm follows:-

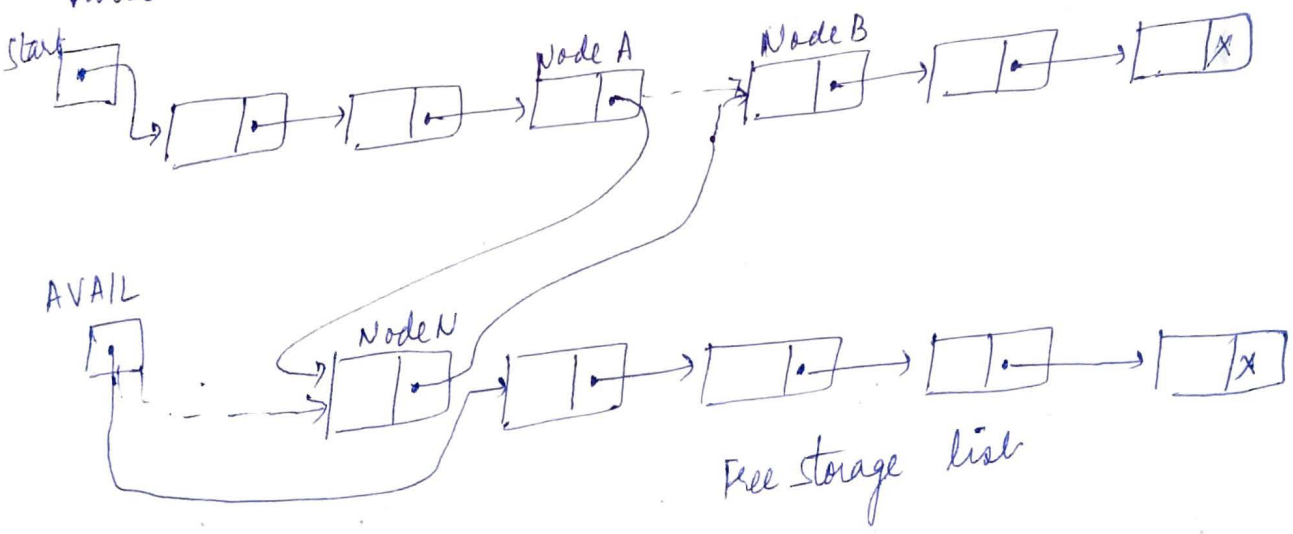
INSLOC (INFO, LINK, START, AVAIL, LOC, ITEM)

This algorithm inserts ITEM so that ITEM follows the node with location LOC or inserts ITEM as the first node when LOC = NULL.

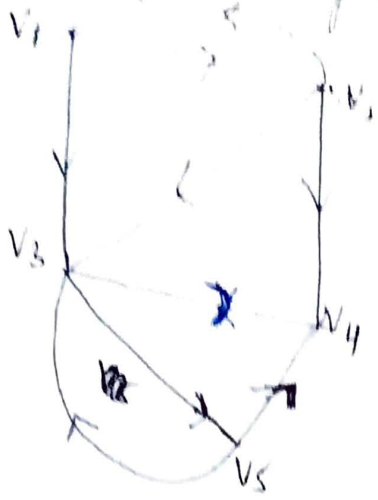
1. If AVAIL = NULL, then: Write = OVERFLOW, and Exit
2. Set NEW = AVAIL and AVAIL = LINK[AVAIL].
3. Set INFO[NEW] = ITEM
4. If LOC = NULL, then:
   Set LINK[NEW] = START and START = NEW.
   Else:
   Set LINK[NEW] = LINK[LOC] and LINK[LOC] = NEW
   End If
5. Exit

⇒ If LOC = NULL, then N is inserted as the first node in LIST.

⇒ The first node in the AVAIL list will be used for the new node N.



Consider the directed graph G



i) Find indegree and outdegree of each node.

Node v_1
Indegree – 1
Outdegree – 2

Node v_2
Indegree – 1
Outdegree – 3

Node v_3
Indegree – 4
Outdegree – 1

Node v_4
Indegree – 3
Outdegree – 0

Node v_5
Indegree – 1
Outdegree – 2

ii) Find number of simple paths?

In graph, a simple path is a path in a graph which does not have repeating vertices.

source – v_1, destination – v_5

1) $v_1 - v_3 - v_5$ – Number 1 of length 3

2) $v_1 - v_2 - v_3 - v_5$ – Number 2 of length 4

iii) Is there any source or sink?

Ans. Source is the node with positive outdegree and zero indegree.
Sink is the node with zero outdegree and positive indegree.
So, according to this graph => Node v_4 is a sink.

Ques 9 (b) Are B trees of order 2 are full binary trees? If yes, explain how.

Ans → A B-tree of order m is an m-way search tree if the B-tree is not empty, the corresponding extended tree satisfies the following properties:-

1. The root has at least two children.
2. All internal nodes other than the root have at least $\lceil m/2 \rceil$ children.
3. All external nodes are at the same level.

In a B-tree of order 2, no internal node has more than two children. Since an internal node must have at least two children, all internal nodes of a B-tree of order 2 have exactly two children. This observation, coupled with the requirement that all external nodes be on the same level, implies that B-trees of order 2 are full binary trees. As such, these trees exist only when the number of elements is $2^k - 1$ for some integer k.

———— x ————

Dec -18

Section-A

a) What is a NULL pointer in C?
b) What is dangling pointer and how to avoid it?
c) What is the definition of Big O Notation?
d) Give the syntax of conversation from infix to postfix?
e) How does priority queue work?
f) Difference b/w array and linked list?
g) Explain Binary tree?
h) Describe Adjency list?
i) What is hashing?
j) What is the complexity of Merge sort?

Section-B

a) Give the syntax of searching a specific element in an array.
b) Write an operation to delete n nodes after m nodes of a linked list.
c) How trees provide an efficient insertion and searching? show with example.
d) Give the brief introduction to concept of collision and its resolution using open addressing.
e) Illustrate the concept of breadth-first search traversing of graph by taking a suitable example.

Section-C

a) Explain various datatypes used in data structure with their syntax and application?
b) How a linear array is represented in memory? Explain the program which reads two matrices.
c) Write an algorithm to sort an array of integers in the desending order using insertion sort. How insertion sort algorithm works?

Chandigarh Engineering College, Landran

2nd Sessional Test (MCQ)
Subject: Software Engineering (BTCS-603)
Branch: CSE
Time: 20 Mins
Date: 9-04-2019 (M)

Quality Check by IQAC
Quality(5):- 4.75 / 5
R / NR:- " R " for Printing
Sign IQAC team:- ___(M1), Q(M2)
Hod Sign :- ___

Name: Diksha
Roll No: 1602277
Semester & Section: CSE-K2

Max. Marks: 24

1. **Alpha and Beta Testing are forms of _____.**

Acceptance testing
Integration testing
System Testing
Unit testing

CO2

2. **Burst force, backtracking, cause elimination are strategies used in art of debugging.**

Yes
No

CO3

3. **The interviews, which are held between two persons across the table is _____.**

Written
Non-structured
Group
One-to-one

CO1

4. **When elements of module are grouped because the output of one element serves as nput to another element and so on, it is called _____.**

unctional Cohesion
Communicational cohesion
Sequential cohesion
Procedural cohesion

CO1

5. **One of the fault base testing techniques is _____.**

nit Testing
Beta Testing
Stress Testing
Mutation Testing

CO3

6. **Activities and action taken on the data that are represented by Circle or Round-edged tectangles are called, _____.**

Process
Data storage
Data flow
Entities

CO2

7. **When multiple modules have read and write access to some global data, it is called, _____.**

ntent coupling
Stamp coupling
Data coupling

CO2

Common coupling
what type of coupling, the complete data structure is passed from one module to another?
Control Coupling
Stamp Coupling
External Coupling
Content Coupling

CO2

9. tasks must be executed in the same time-span, what type of cohesion is being exhibited?
stional Cohesion
mporal Cohesion
ctional Cohesion
idential Cohesion

CO3

10. **Which of the following techniques is not a White box technique?**

Statement Testing and coverage
Decision Testing and coverage
Condition Coverage
Boundary value analysis

CO2

11. **The desired level of coupling is** Data coupling
oupling
rol coupling
mon coupling
coupling

CO2

12. **Changes are made to the system to reduce the future system failure chances is called _____.**

ntive Maintenance
ative Maintenance
c. Corrective Maintenance
d. Perfective Maintenance

CO1

13.. **The feature of the object oriented paradigm which helps code reuse is _____.**

a. Object
b. Class
c. Inheritance
d. Aggregation.

CO1

14.. **Boundary value analysis belong to?**
a) White Box Testing
b) Black Box Testing

15. **Alpha testing is done at**
a) Developer's end
b) User's end
c) Both a and b

CO3

16. **Which of the following is non-functional testing?**
a) Black box testing
b) Performance testing
c) Unit testing
d) None of the mentioned

CO1

17. **Name an evaluation technique to assess the quality of test cases.**
a) Mutation analysis
b) Validation
c) Verification
d) Performance analysis

CO2

18. **Effective testing will reduce _____ cost.**
a) maintenance
b) design
c) coding
d) documentation

19. **What is normally considered as an adjunct to the coding step**
a) Integration testing
b) Unit testing
c) Completion of Testing
d) Regression Testing

CO1

20. **Knowledge of software program, design and structure is essential in _____.**

a. Black-box testing
b. White-box testing
c. Integration testing
d. None of the above

CO2

21. **The tools that support different stages of software development life cycle are called a _____**

a. CASE Tools
b. CAME tools
c. CAQE tools
d. CARE tools

CO3

22. **What is Cyclomatic complexity?**
a) Black box testing
b) White box testing
c) Yellow box testing
d) Green box testing

CO2

23. **Which of the following is/are White box technique?**
a) Statement Testing
b) Decision Testing
c) Condition Coverage
d) All of these

CO1

24. **Which of the following is the worst type of module cohesion?**
a) Logical Cohesion
b) Temporal Cohesion
c) Functional Cohesion
d) Coincidental Cohesion

CO3

SECTION - A :

a) What is a NULL pointer in C?

Ans At the very high level, we can think of NULL as null pointer which is used in C for various purposes. Some of the most use cases of NULL are :-

1. To initialize a pointer variable when that pointer isn't assigned any memory address yet.
2. To check for null pointer before accessing any pointer variable. By doing so, we cannot perform error handling in pointer related code. Eg. Dereference pointer variable only if it's not NULL.
3. To pass a null pointer to a function argument when we don't want to pass any valid memory address.

(b) What is dangling pointer and how to avoid it?

Ans Dangling pointers in computer programming are pointers that point to a memory location that has been deleted.

Dangling pointers arise during object destruction, when an object that has an incoming reference is deleted, or deallocated, without modifying the value of pointer, so that the pointer still points to the memory location of the deallocated memory.

The system may reallocate the previously freed memory, unpredictable behaviour may result as the memory may now contain completely different data.

(c) What is the definition of Big O Notation?

Ans Big O notation is a mathematical notation that describes the limiting behaviour of a function when the argument tends towards a particular value or infinity. It is a member of family of notations invented by Paul Bachmann, Edmund Landau and others, collectively called Bachmam-Landau notation or asymptotic notation.

In computer science, big O notation is used to classify algorithms, according to how their running time or space requirements grow as the input size grows.

In analytic number theory, big O notation is often used to express a bound on the difference between an arithmatical function and a better understood approximation; a famous example of such a difference is the remainder term in the prime number theorem.

(d) Give the syntax of conversion from infix to postfix?

Ans Let x - arithmetic expression.

Algorithm finds equivalent postfix expression y.

1st Push (onto stack. Add) to end of x.

2nd Scan x left to right. Repeat steps 3 to 6 while stack is not empty.

3rd If operand is encountered, add it to y.

4th If left parenthesis is encountered, push onto stack.

5th If an operator is encountered:-

1st Repeatdly pop from stack and add to Y, each operator.

2nd Add operator to stack.

[End of if]

6th If a right parenthesis is encountered:-

1st Repeatedly pop from stack and add to y each operator until a left parenthesis is encountered.

2nd Remove the operator to stack. Remove the left parenthesis.

[End of if]

7th Exit.

(e) How does priority queue work?

A priority queue could work by sorting after each insertion, but it would be slow; each insertion would take time proportional to N * lnN where N is the size of the queue.

(f) Difference b/w array and linked list?

A

ARRAY	LINKED LIST
(a) Consistent set of a fixed number of data items.	(a) It is an ordered set comprising a variable number of data items.
(b) Specific size.	(b) Dynamically allocated size.
(c) Location allocated at compile time.	(c) Location allocated at run time.
(d) Stored consecutively	(d) Stored randomly.
(e) Directly or randomly	(e) Sequentially accessed.
(f) Less memory required.	(f) More memory required
(g) Ineffective	(g) Efficient.

(g) Explain Binary tree?

Ans Binary tree does not stand for Binary tree or Binary search tree There are some common characteristics that can be summarized with words that begin with B, which is most likely the origin of the name.

(h) Describe Adjency list?

Ans Adjency list is a collection of unordered lists used to represent a finite graph. Each list describes the set of neighbours of vertex in the graph.
For use as a data structure, the main alternation is is adjency matrix. The compactness encourages locality of reference.

(i) What is hashing?

Ans It is a collision resolving technique in open addressed Hash tables. It uses the idea of applying a second hash function to key when a collision occurs.

(j) What is the complexity of Merge sort?

Ans $O(\log n)$

n-B1.

Give the syntax of searching a specific element in an array

```
#include <stdio.h>
int main()
{ int a[30], ele, num, i;
  printf("\n Enter no. of elements:");
   scanf("%d", &num);
   printf("\n Enter the values:");
    for (i=0; i<num; i++)
     { scanf("%d", &a[i]);
     }
     // Read the element to be searched
     printf("\n Enter the elements to be searched:");
     scanf("%d", &ele);
     // Search starts from the zeroth location
      i=0;
      while (i<num && ele != a[i])
       { i++;
       }
     // if i<num then Match found
     if (i<num)
     { printf("Number found at the location =%d", i+1);
     }
      else
      { printf("Number not found");
      }
     return 0;
}
```

OUTPUT:

```
Enter no. of elements : 5
11 22 33 44 55
Enter the elements to be searched : 44
Number found at the location = 4
```

Ques 3. Write an operation to delete n nodes after m nodes of a linked list.

Answer.

```
// C++ program to delete N nodes
// after M nodes of a linked list
#include <bits/stdc++.h>
using namespace std;

// A linked list node
class Node
{ public : int data;
           Node *next;
};
void push (Node ** head_ref , int new_data)
{
/* allocate node */
Node* new_node = new Node();
/* put in the data */
new_node -> data = new_data;
/* link the old list off the new node */
new_node -> next = (*head_ref);
/* move the head to point to the new node */
(*head_ref) = new_node;
}
/* Function to print linked list */
```

```
void printList (Node * head)
{
    Node * temp = head;
    while (temp != NULL)
    {
        cout << temp -> data << " ";
        temp = temp -> next;
    }
    cout << endl;
}
// Function to skip M nodes and then
// delete N nodes of the linked list.
void skipMdeleteN (Node * head, int M, int N)
{   Node * curr = head, *t;
    int count;
    // The main loop that traverses
    // through the whole list
    while (curr)
    {
        // Skip M nodes
        for (count = 1; count < M && curr != NULL; count++)
        {   curr = curr -> next;
        }
        // if we reached end of list, then return
        if (curr == NULL)
            return;
        // Start from next node and delete N nodes
        t = curr -> next;
        for (count = 1; count <= N && t != NULL; count++)
        {   Node * temp = t;
            t = t -> next;
```

```
        free(temp);
    }
    // Link the previous list with remaining nodes
    curr -> next = t;
    // Set current pointer for next iteration.
    curr = t;
  }
}

// Driver code
int main()
{ /* Create following linked list
   1->2->3->4->5->6->7->8->9->10 */
  Node* head = NULL;
  int M=2, N=3;
  push(& head, 10);
  push(& head, 9);
  push(& head, 8);
  push(& head, 7);
  push(& head, 6);
  push(& head, 5);
  push(& head, 4);
  push(& head, 3);
  push(& head, 2);
  push(& head, 1);

  cout << "M = " << M << "N = " << N << "\n Given linked list: \n";
  printList(head);
  skipMdeleteN(head, M, N);
  cout << "\n Linked list after deletion is: \n";
  printList(head);
  return 0;
}
```

OUTPUT:

M = 2 , N = 3

Given Linked List :

1 2 3 4 5 6 7 8 9 10

Linked list after deletion is :

1 2 6 7

Ques 4. How Trees provide an effecient insertion and searching? Show with example.

Answer. • Inserting a node -

A naïve algorithm for inserting a node into a BST is that, we start from the root node, if the node to insert is less than the root, we go to left child, and otherwise we go to the right child of the root. We continue this process (each node is a root for some sub tree) until we find a null pointer (or leaf node) where we cannot go any further. We then insert the node as a left or right child of the leaf node based on node is less or greater than the leaf node. We note that a new node is always inserted as a leaf node.

A recursive algorithm for inserting a node into a BST is as follows. Assume we insert a node N to tree T. if the tree is empty, then we return new node N as the tree. Otherwise, the problem of insertion is reduced to inserting the node N to left of right sub trees of T, depending on N is less or greater than T. A definition is as follows:

Insert (N, T) = N if T is empty

= insert (N, T.left) if N<T

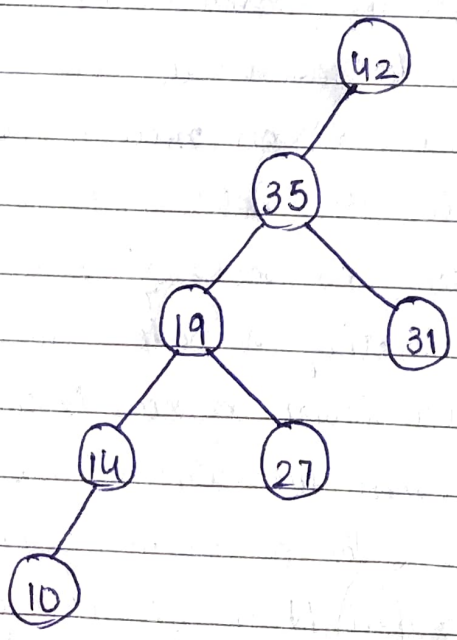
= insert (N, T, right) if N>T

- **Searching for a node -**

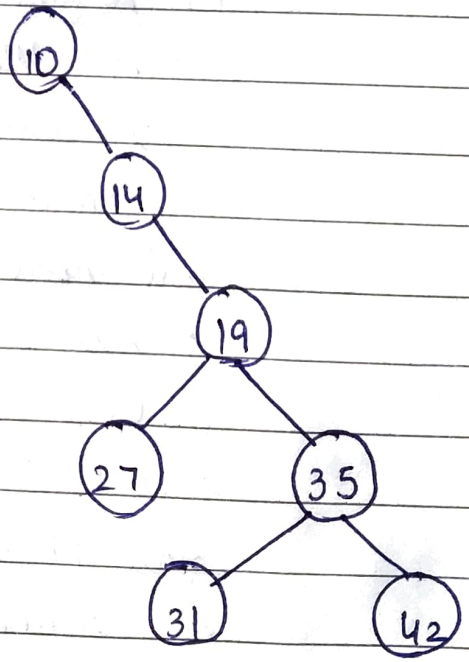
Searching for a node is similar to inserting a node. We start from root, and then go left or right until we find (or not find the node). A recursive definition of search is as follows. If the node is equal to root, then we return true. If the root is null, then we return false. Otherwise we recursively solve the problem for T.left or T.right, depending on N<T or N>T. A recursive definition is as follows.

Search should return a true or false, depending on the node is found or not.

search(N, T) = false if T is empty
= true if T=N
= search(N, T.left) if N<T
= search(N, T.right) if N>T



If input 'appears' non-increasing manner



If input 'appears' in non-decreasing manner

Ques 5. Give the brief introduction to concept of collision and its resolution using open addressing.

Answer. Like seperate chaining, open addressing is a method for handling collisions. In open addressing, all elements are sorted in the hash table itself. So at any point, size of the table must be greater than or equal to the total number of keys

Insert (k): Keep probing until an empty slot is found. Once an empty slot is found, insert k.

Search (k): Keep probing until slot's key doesn't become equal to k or an empty slot is reached.

Delete (k): If we simply delete a key, then search may fail. so slots of deleted slot, but the search doesn't stop at a deleted slot.

Open Addressing is done following ways:

(a) Linear Probing: In linear probing, we linearly probe for next slot. For example, typical gap between two probes is 1 as taken in below example also.

Let hash(x) be the slot index computed using hash function and S be the table size.

If slot hash(x) % S is full, then we try (hash(x)+1)%S

If (hash(x)+1) % S is also full, then we try (hash(x)+2) %S

If (hash(x)+2) %S is also full, then we try (hash(x)+3) %S

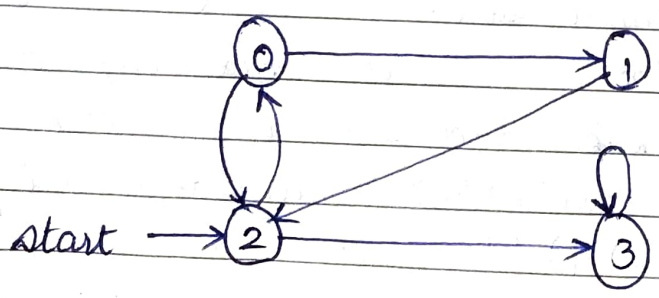
Clustering: The main problem with linear probing is clustering, many consecutive elements form groups and it starts taking time to find a free slot or to search an element.

(b) Quadratic Probing: We look for i^{2} th slot in i^{th} iteration
Let hash(x) be the slot index computed using hash function
If slot hash(x) % S is full, then we try (hash(x)+1*1) % S
If (hash(x)+1*1) % S is also full, then we try (hash(x)+2*2) % S
If (hash(x)+2*2) % S is also full, then we try (hash(x)+3*3) % S

(c) Double Hashing: We use another has function hash2(x) and look for i* hash2(x) slot in i^{th} rotation.
Let hash(x) be the slot index computed using hash function.
If slot hash(x) % S is full, then we try (hash(x)+1*hash2(x)) %

Ques 6. Illustrate the concept of breadth-first search traversing of graph by taking a suitable example.

Answer. Breadth First Search or BFS for a graph.
BFS for a graph is similar to Breadth First Traversal of a tree. The only catch here is, unlike trees, graphs may contain cycles, so we may come to same node again.



A BFS is another technique for traversing a finite graph. BFS visits the sibling vertices before visiting child vertices and a queue is used in search process.

Pseudocode

- Input - A graph G and a vertex v of G.
- Output - The closest vertex to v satisfying some conditions, or null if no such vertex exists.

```
1 procedure BFS (G, v)
2 create a queue Q
3 enqueue v onto Q
4 mark v
5 while Q is not empty do
```

Section - C

Ques → 7 Explain various data types used in data structure with their syntax and application?

Ans → 7 Data structure provides a means to manage large amounts of data efficiently for uses such as large data bases and internet indexes services.

Data structure is a particular way of organising and storing data in a computer so that it can be accessed and modified efficiently.

Different data structures are →

Array → Array is a linear data structure consisting of a collection of elements each identified by array index. Array can be used for sorting elements. can perform matrix operation and can be used in CPU Scheduling.

An Array with 5 elements. Also array indexing starts from 0.

Stack → Stack is a linear data structure consisting of a collection which follows a particular order in which operations are performed.

Insertion in a stack is called PUSH while deletion from stack is called POP.

Queue → Queue is a linear structure which follows a particular order in which operations are performed. Order is FIFO [First in first out]

Insertion in stack is called Enqueue while deletion from stack is called Dequeue.

Application → Queue is used when a resource is shared among multiple consumer like in CPU scheduling, Disk scheduling.

It is also used in Palindrome recognition.

Binary tree → is a tree data structure in which each node has atmost two children, which are referred to as the left child and right child. A binary tree of size 9 and height 3, with a root node whose value is 2.

Application → Binary search tree, heaps, Binary tries etc.

Binary search tree → is a binary tree where the value of each node is greater than or equal to the value of left subtree and the value of each node is less than or equal to the

value of right subtree.

A binary search tree of size 9 and depth 3, with 8 at the root.

Binary search tree of ~~size~~ is used to implement multi-level indexing in database. It is also used in Huffman coding algorithm and to implement searching algorithm.

Heap → is a specialised tree data structure that satisfies the following property: If P is a parent node of C, then the value of P is either greater than or equal to or less than or equal to the value of C.

Max heap with node keys being integers from 1 to 100.

Heap is used in heapsort, and in dynamic memory allocation in lisp.

Trie → Trie is an ordered tree data structure that is used to store a dynamic set or associative array where the keys are usually string.

A Trie for keys "A", "to", "tea", "ted", "ten", "i", "in", and "inn".

Trie is used as dictionary, such as one found on a mobile telephone for auto completion and spell-checking.

It is also used in Aho - Corasick algorithm and in KMP algorithm.

Hash table → is a data structure that is used to store keys / value pairs. It uses a hash function to compute an index into an array in which an element will be inserted or searched.

Hash table is used for fast data lookup - symbol table for compilers, database indexing, caches, unique data representation.

Graph → Graph is a data structure that consists a finite set of vertices called as Nodes and a finite set of ordered pair called as edge. Graph can be directed or undirected.

Graph are used to represent networks. Graphs are also used in social networks like linkedIn, facebook.

For example → In facebook, each person is represented with a vertex. Each node is a structure and contains information like person id, name, gender and locale. They are also used in Routing algorithms.

Program → #include <stdio.h>

```
int main()
{
    int r, c, a[100][100], b[100][100], sum[100][100], i, j;
    printf("Enter number of rows (b/w 1 and 100): ");
    scanf("%d", &r);

    printf("Enter no. of columns (b/w 1 and 100): ");
    scanf("%d", &c);

    printf("\n Enter elements of 1st matrix: \n");

    for (i=0; i<r; ++i)
        for (j=0; j<c; ++j)
        {
            printf("Enter element a %d %d: ", i+1, j+1);
            scanf("%d", &a[i][j]);
        }
    printf(" enter elements of 2nd matrix: \n");
    for (i=0; i<r; ++i)
        for (j=0; j<c; ++j)
        {
            printf("enter elements a %d %d: ", i+1, j+1);
            scanf("%d", &b[i][j]);
        }

// Adding two matrices
    for (i=0; i<r; ++i)
        for (j=0; j<c; ++j)
        {
            sum[i][j] = a[i][j] + b[i][j];
        }
```

```
// displaying the result
printf("\n Sum of two matrix is : \n\n");

for(i=0; i<r; ++i)
    for (j=0; j<c; ++j)
    {
        printf("%d", sum[i][j]);
        if (j==c-1)
        {
            printf("\n\n");
        }
    }
return 0;
}
```

Output →

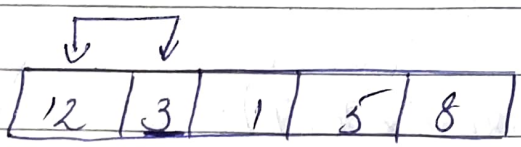
```
Enter no. of rows (b/w 1 and 100) : 2
Enter no. of columns (b/w 1 and 100) : 3
Enter elements of 1st matrix →
Enter element a11 : 2
Enter element a12 : 3
Enter element a13 : 4
Enter element a21 : 5
Enter element a22 : 2
Enter element a23 : 3

Enter elements of 2nd matrix →
Enter element a11 : -4
Enter element a12 : 5
Enter element a13 : 3
Enter element a21 : 5
Enter element a22 : 6
Enter element a23 : 3
Sum of two matrix is :
-2  8  7
10  8  6
```

Ques → 9 Write an algorithm to sort an array of integers in the descending order using insertion sort. How insertion sort algorithm works?

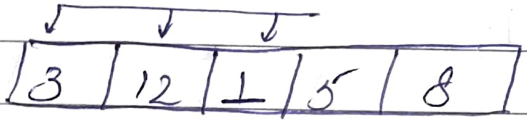
Ans → 9

Step → 1



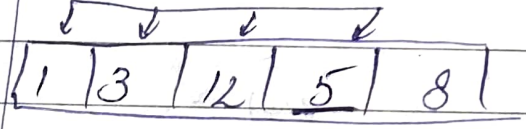
checking second element of array with element before it and inserting it in proper position. In this case 3, is inserted in position of 12.

Step → 2



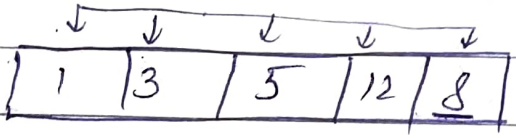
checking 3rd element of array with elements before it and inserting it in proper pos. In this case 1 is inserted in position of 3.

Step 3



checking 4th element of array with elements before it and inserting it in proper position. In this case, 5 is inserted in pos. of 12.

Step 4.



checking 5th element of array with elements before it and inserting in proper position. In this case, 8 is inserted in pos. of 12.

Step 5

Sorted array in ascending order.

Explanation →

1. Step →1 → The second element of an array is compared with the elements that appears before it (only first element in this case). If the second element is smaller than first element, second element is inserted in the position of first element. After first step, first two elements of an array will be sorted.

2. Step →2 → The third element of an array is compared with the elements that appears before it (first and second element). If third element is smaller than the first element, it is inserted in the position of first element. If third element is larger than first element but, smaller than second element, it is inserted in the position of second element. If third element is larger than both the elements, it is kept in the position as it is. After second step, first three elements of an array will be sorted.

3. step →3 → Similarly, the fourth element of an array is compared with the elements that appears before it (first, second and third element) and the same procedure is applied and that element is inserted in the proper position. After third step, first four elements of an array will be sorted.

May 2019

Total No. of Pages : 02

Roll No.

Total No. of Questions : 18

B.Tech.(3D Animation & Graphics) (2012 Onwards)

B.Tech.(CSE)/(IT) (2011 Onwards)

(Sem.–3)

DATA STRUCTURES

Subject Code : BTCS-304

M.Code : 56594

Max. Marks : 60

Time : 3 Hrs.

INSTRUCTION TO CANDIDATES :

1. SECTION-A is COMPULSORY consisting of TEN questions carrying TWO marks each.
2. SECTION-B contains FIVE questions carrying FIVE marks each and students has to attempt any FOUR questions.
3. SECTION-C contains THREE questions carrying TEN marks each and students has to attempt any TWO questions.

SECTION-A

Answer briefly :

1) How pointers are used to manage address of memory?

2) What is dangling pointer give example?

3) Give some applications of stack.

4) How time complexity of an algorithm is computed?

5) Discuss recursive procedures in trees.

6) Discuss AVL trees.

7) Write use of heap sort.

8) What is undirected graph?

9) Discuss rehashing in hash tables.

10) Give the syntax of selection sort.

SET-A

CHANDIGARH ENGINEERING COLLEGE, LANDRAN

1st Sessional Test (MCQ) Subject: CN II (BTCS-501)

Branch: CSE **Time: 20 Mins** **Date: 06-09-2019 (M)** **Max. Marks: 24**

Name: _____ **Roll No:** _____ **Semester & Section:** _____

1. IPv6 does not use _____ type of address CO1
   a) Broadcast b) Multicast
   c) Anycast d) None of the mentioned
2. These are the features present in IPv4 but not in IPv6 CO1
   a) Fragmentation b) Header checksum
   c) Options d) All of the mentioned
3. The _____ field determines the lifetime of IPv6 datagram CO1
   a) Hop limit b) TTL
   c) Next header d) None of the mentioned
4. Which mode of IPsec should you use to assure security and confidentiality of data within the same LAN? CO2
   a) AH transport mode b) ESP transport mode
   c) ESP tunnel mode d) AH tunnel mode
5. 3. Which two types of IPsec can be used to secure communications between two LANs? CO2
   a) AH tunnel mode b) ESP tunnel mode
   c) Both AH tunnel mode and ESP tunnel mode
   d) ESP transport mode
6. _____ provides authentication at the IP level. CO2
   a) AH b) ESP
   c) PGP d) SSL
7. IPsec defines two protocols: _____ and _____ CO2
   a) AH; SSL b) PGP; ESP
   c) AH; ESP d) All of the mentioned
8. Which is not a key security requirement? CO1
   a) Confidentiality b) Integrity
   c) Threat d) All of the above
9. _____is simply to exploit vulnerability. CO1
   a) Threat b) Attack
   c) Security d) None of the above
10. The header length of an IPv6 datagram is _____ CO1
    a) 10bytes b) 25bytes
    c) 30bytes d) 40bytes
11. In the IPv6 header, the traffic class field is similar to which field in the IPv4 header? CO1
    a) Fragmentation field b) Fast-switching
    c) ToS field d) Option field
12. ESP provides CO2
    a) source authentication b) data integrity
    c) privacy d) all of the mentioned
13. The encryption process where same keys are used for encrypting and decrypting the information. CO1
    a)Symmetric Key Encryption
    b) Asymmetric Key Encryption
    c) Both a & b d) None of the above
14. If plain text is "HELLO" & cipher text is "KHOOR", what type of cipher is it? CO2
    a) Monoalphabetic
    b) Polyalphabetic d) None of the above
    c) Both a & b
15. Use the shift cipher with key = 15 to encrypt the message "HELLO." CO2
    a) WTAAD b) WTABD
    c) WTABC d) WTBAC
16. In computer security… means that computer system assets can be modified only by authorized parities. CO2
    a) Confidentiality b) Integrity
    c) Availability d) Authenticity
17. In computer security… means that the information in a computer system only be accessible for reading by authorized parities. CO2
    a) Confidentiality b) Integrity
    c) Availability d) Authenticity
18. The type of authentication used in network security
    a) One Factor Authentication
    b) Two Factor Authentication
    c) Three Factor Authentication d) All of the above
19. What are the key security requirements? CO1
    a)Confidentiality b) Integrity
    c) Authentication d) All of the above
20. A _____ is an inherent weakness in the design, configuration, implementation or management of a network. CO1
    a) Threat b) Attack
    c) Vulnerability d) None of the above
21. An attack that attempts to alter the resources or affect their operation CO1
    a) Active attack b) Passive attack
    c) Single attack d) None of the above
22. Which is a passive attack? CO1
    a) Traffic Analysis b) Replay
    c) Denial of Service d) Masquerade
23. Which is an active attack? CO1
    a) Traffic Analysis b) Release of Message contents
    c) Denial of Service d) None of the above
24. Which is not key components of cryptosystem? CO2
    a) Cipher text
    b) Encryption & Decryption Algorithm
    c) Encryption & Decryption Key
    d) None of the above

SECTION-B

How queues are represented in memory? Write their applications.

What are the tree traversal techniques? Explain each with an example.

What is Stack? Why it is known as LIFO? Write an algorithm using PUSH and POP.

Give idea of hashing and its use as hashing function.

Explain Inorder, Preorder and Postorder Traversal operation on Binary tree with examples.

SECTION-C

a. Write the procedure to implement the adjacent matrix.

b. Define data structure graph. How they are represented in memory?

What do you mean by Link list? Write an algorithm to insert and delete a node in Singly Linked List.

How does a linear search algorithm work? Give the syntax by taking an example set. Compute the complexity of linear search algorithm.

NOTE : Disclosure of Identity by writing Mobile No. or Making of passing request on any page of Answer Sheet will lead to UMC against the Student.

SET-A

CHANDIGARH ENGINEERING COLLEGE, LANDRAN

Quality(5):- ___ / 5
R / NR:- " ___ " for Print
Sign IQAC team:-
Hod Sign :-

1st Sessional Test (MCQ) Subject: CN II (BTCS-501) **Date: 06-09-2019 (M)** **Max. Marks:**

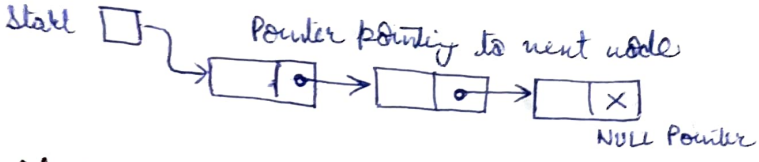
Branch: CSE **Time: 20 Mins** **Semester & Section:**

Name: _____ **Roll No:** _____

1. IPv6 does not use _____ type of address CO1
   a) Broadcast b) Multicast
   c) Anycast d) None of the mentioned
2. These are the features present in IPv4 but not in IPv6. CO1
   a) Fragmentation b) Header checksum
   c) Options d) All of the mentioned
3. The _____ field determines the lifetime of IPv6 datagram CO1
   a) Hop limit b) TTL
   c) Next header d) None of the mentioned
4. Which mode of IPsec should you use to assure security and confidentiality of data within the same LAN? CO2
   a) AH transport mode b) ESP transport mode
   c) ESP tunnel mode d) AH tunnel mode
5. 3. Which two types of IPsec can be used to secure communications between two LANs? CO2
   a) AH tunnel mode b) ESP tunnel mode
   c) Both AH tunnel mode and ESP tunnel mode
   d) ESP transport mode
6. _____ provides authentication at the IP level. CO2
   a) AH b) ESP
   c) PGP d) SSL
7. IPsec defines two protocols: _____ and _____ CO2
   a) AH; SSL b) PGP; ESP
   c) AH; ESP d) All of the mentioned
8. Which is not a key security requirement? CO1
   a) Confidentiality b) Integrity
   c) Threat d) All of the above
9. _____is simply to exploit vulnerability. CO1
   a) Threat b) Attack
   c) Security d) None of the above
10. The header length of an IPv6 datagram is _____ CO1
    a) 10bytes b) 25bytes
    c) 30bytes d) 40bytes
11. In the IPv6 header, the traffic class field is similar to which field in the IPv4 header? CO1
    a) Fragmentation field b) Fast-switching
    c) ToS field d) Option field
12. ESP provides CO2
    a) source authentication b) data integrity
    c) privacy d) all of the mentioned
13. The encryption process where same keys are used for encrypting and decrypting the information. CO1
    a)Symmetric Key Encryption
    b) Asymmetric Key Encryption
    c) Both a & b d) None of the above
14. If plain text is "HELLO" & cipher text is "KHOOR", what type of cipher is it? CO2
    a) Monoalphabetic
    b) Polyalphabetic
    c) Both a & b d) None of the above
15. Use the shift cipher with key = 15 to encrypt the message "HELLO." CO2
    a) WTAAD b) WTABD
    c) WTABC d) WTBAC
16. In computer security… means that computer system assets can be modified only by authorized parities. CO2
    a) Confidentiality b) Integrity
    c) Availability d) Authenticity
17. In computer security… means that the information in a computer system only be accessible for reading by authorized parities. CO2
    a) Confidentiality b) Integrity
    c) Availability d) Authenticity
18. The type of authentication used in network security
    a) One Factor Authentication
    b) Two Factor Authentication
    c) Three Factor Authentication d) All of the above
19. What are the key security requirements? CO1
    a)Confidentiality b) Integrity
    c) Authentication d) All of the above
20. A _____ is an inherent weakness in the design, configuration, implementation or management of a network. CO1
    a) Threat b) Attack
    c) Vulnerability d) None of the above
21. An attack that attempts to alter the resources or affect their operation CO1
    a) Active attack b) Passive attack
    c) Single attack d) None of the above
22. Which is a passive attack? CO1
    a) Traffic Analysis b) Replay
    c) Denial of Service d) Masquerade
23. Which is an active attack? CO1
    a) Traffic Analysis b) Release of Message contents
    c) Denial of Service d) None of the above
24. Which is not key components of cryptosystem? CO2
    a) Cipher text
    b) Encryption & Decryption Algorithm
    c) Encryption & Decryption Key
    d) None of the above

How are pointers used to manage address of memory?

Ans) A pointer is a variable that stores or points to the address of a variable. Pointers are used to allocate memory during run time. Pointer may point to any data type such as int, float, char, etc.
Pointers are mostly used in data structures like lists, queues and trees. In link lists, the linear order of nodes is given by means of pointers. In the link field of node, pointer is used to point to memory address of next node.



(Use of pointers in link list)

2) What is dangling pointer. Give example

Ans) Dangling pointers are type of pointers that point towards a memory location that has been deleted or freed. They arise during object destruction i.e when an object is destroyed but the value of pointer isn't changed.

```
eg) int main ()
    { char * * strPtr;
        { char * str = "Hello";
          strPtr = & str; }
      cout << * strPtr;
    }
```

As str is non visible in outer block, strptr is dangling pointer.

3) Give some applications of Stack.

Ans) A stack is a data structure which follows ~~first~~ last first out operation i.e element added last is deleted first. Stacks can be used to :-

i) Evaluate prefix, infix and postfix expressions.
ii) Convert one form of expression to another.
iii) Solve complex problems such as Tower of Hanoi, etc.
iv) String Reversal.

4) How is time complexity of an algorithm computed?

Ans) The time and space an algorithm uses are two main measures to find complexity of an algorithm.

An algorithm 'A' for probability 'P' is said to have time complexity of $T(n)$, the number of steps required to complete its run for an input of size n is always $\leq T(n)$.

Time complexity can be computed using ~~two~~ three cases.

1) Average Case : Average time to run the algorithm.
2) Worst Case : Worst time to run the program.
3) Best Case : Shortest time to run an algorithm.

Eg) in Linear search, best case is $C_n = 1$, average case = $\frac{n+1}{2}$ and worst case = n.

5) Discuss recursive procedures in trees.

Ans) Recursion / Recursion tree is another method for solving recurrence relations.

Procedure to solve Recurrence Relation using recursion tree :-

1) Draw a recursion tree based on recurrence relation.
2) Determine : a) Cost of each level
   b) Total no of levels in recursion tree
   c) Number of nodes in last level
   d) Cost of last level.
3) Add cost of all levels and simply expression obtained in terms of asymptotic notation.

Discuss AVL tree

Ans AVL tree is one of the balanced tree introduce in 1962 by Adelson - Velskii and Landis.

An empty binary tree is an AVL tree. A non-empty binary tree T is an AVL tree iff given T^L and T^R to be the left and right subtree of T and $h(T^L)$ and $h(T^R)$ to be the height of subtree T^L and T^R resp., T^L and T^R are AVL tree and $|h(T^L) - h(T^R)| \leq 1$

* $h(T^L) - h(T^R)$ is called balance factor (BF) and for an AVL tree the BF of a node can be either 0, -1, 1

* An AVL search tree is binary search tree which is an AVL tree.

* Since AVL tree are height balance trees operation like insertion and deletion have low time complexity.

Q7 Write use of heap sort.

Ans 1) Guaranteed O(n log n) performance :-

When you don't necessarily need ~~the~~ very fast performance but need O(n log n) performance (eg. in a game) because Quicksort is $O(n^2)$ can be very slow and merge sort take O(n) extra space.

2) To avoid Quicksort worst case :-

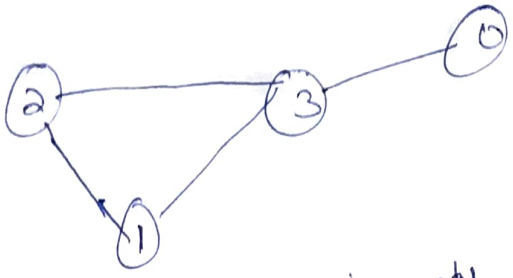
C++'s std::sort generally use variation of quicksort called Introsort, which use heapsort to sort the current partition if Quicksort recursion goes too deep indicating the worst case has occured.

3) Partially sorted array even if stopped abruptly :-

We get partialy sorted array if heapsort somehow stopped absuptly.

Q8 What is undirected graph?

Ans When a graph has an unordered pair of vertex. it is an undirected graph in other word there is no specific direction to represent the edges the vertices connected together by undirected arcs, which are edges without arrow. Of there is an edge b/w vertex A and vertex B it's possible to traverse from B to A or A to B. no specific direction



There is no direction in any of the edge.

Set of vertices (V) = {1, 2, 3, 0}

Set of edges (E) = {(1,2), (2,1) (2,3) (3,2), (1,3) (3,1), (3,0), (0,3)}

Q9 Discuss rehashing the hash table.

Ans If the hash table is close to full, the search time grow and may become equal to table size. When the load factor exceed a certain value (e.g. greater then .5) we do rehashing.

Build a second table twice as large as the original and rehash these all the keys ~~of all~~ ~~the~~ of the original table

Rehashing is expensive operation with running time O(N)

But Once it done new hash table will have good performance.

Give Syntax of selection sort.

MIN (A, K, N, LOC)

An array A is in memory this find smallest element among A[K], A[K+1] ... A[N]

1) set MIN = A[K] and LOC := K
2) Repeat for J = K+1, K+2, ... N
   If MIN [> A[J] then set MIN := A[J]
   and A LOC := J
   [end of loop]
3 return.

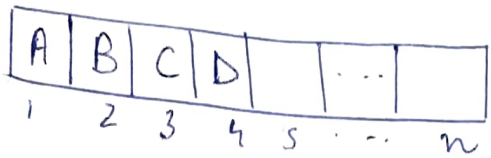
SELECTION (A, N)

1) Repeat step 2 and 3 for K = 1, 2, ... n-1.
2) call MIN (A, K, N, LOC)
3) set Temp := A[K], A[K] := A[LOC] and A[LOC] = Temp
   [end of step 1 loop]
4) exit.

How are queues represented in memory? Write their applications.

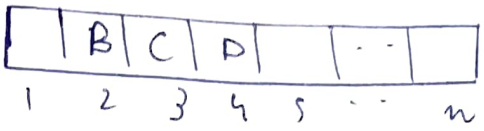
1) A queue is a data structure where insertion takes place at rear end and deletion takes place at front end. Queues are represented in memory by means of one way lists or linear arrays (Usually). In linear array, two pointer variables FRONT, containing the location of the front element of queue and REAR, containing location of REAR end.

eg)

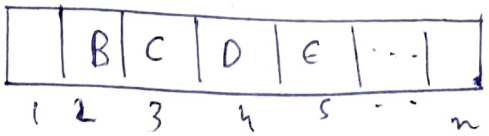


FRONT : 1
REAR : 4

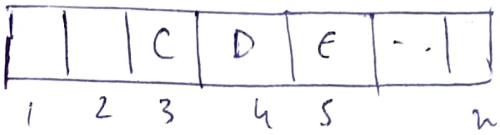
Queues



FRONT : 2
REAR : 4



FRONT : 2
REAR : 5



FRONT : 3
REAR : 5

Queues are used when data is to be processed in FIRST IN FIRST OUT operation

1) When a resource is shared among multiple consumers
2) When data has to be transferred asynchronously

12) What are the tree traversal techniques? Explain each with example.

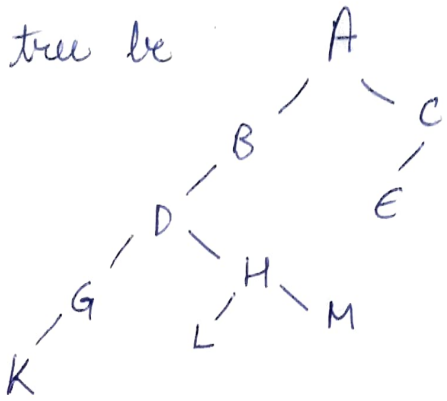
Ans 12) There are three standard ways to traverse a tree

1) Preorder : i) Process root R
ii) Traverse the left subtree of R in preorder
iii) Traverse the right subtree of R in preorder

2) Inorder : i) Traverse the left subtree of R in inorder
ii) Process the root R
iii) Traverse the right subtree of R in inorder

3) Postorder : i) Traverse left subtree of R in post order
ii) Traverse right subtree of R in post order
iii) Process the root R

Example) Let tree be



* Using STACK, elements of tree will be processed.

A) Preorder) 1) Initially push null onto STACK.
STACK: ϕ
⇒ Set PTR = A, root of tree
2) Process A and push right child C onto STACK
STACK: ϕ, C
3) Process B.
4) Process D and push rightt child H onto STACK
STACK: ϕ, C, H.
5) Process G
6) Process K
7) [Backtrack] Process H and push M onto stack.
STACK: ϕ, C, M
Process L.
8) Process M and backtrack to C
9) Process C

does a linear search also. Why?

process E.

⇒ Traversal: A B D G K H L M C E

3. ~~Postorder~~ Inorder
1) Using same tree before, initially push NULL onto STACK (STACK = ϕ)
Set PTR = A.
2) Push A, B, D, G, K onto Stack
3) Pop K, G and D.
4) Set PTR = H
5) Push H onto STACK and L onto STACK.
6) Pop L and H.
7) Set PTR = M and push M onto STACK.
8) Pop M
9) Pop B and A. Set PTR = C
10) Push C and E onto STACK
11) Pop E and C.

⇒ Traversal: K, G, D, L, H, M, B, A, E, C.

c) POSTORDER

1) Using same tree before, push NULL to STACK and set ptr = A
2) Push A, B, D, G, K onto stack. Push -C onto stack before B and -H before G.
Stack: ϕ, A, -C, B, D, -H, G, K
3) Pop and process K and G. Since -H is negative, pop -H.
Stack: ϕ, A, -C, B, D.
Set PTR = H and return to step 1.
4) Push H onto STACK, Push -H onto stack and Push L.
Stack: ϕ, A, -C, B, D, H, -H, L.
5) Pop and process L. ~~Set~~ POP -H and set PTR = H
6) Push H onto Stack. (Stack ⇒ ϕ, A, -C, B, D, H, H)
7) [Backtrack] Pop and process H, H, D and B but only pop -C.
Stack [ϕ, A]
Set PTR = C
8) Push C then E in stack [stack = ϕ, A, C, E]
9) Pop and process E, C and A.

* Traversed: K, G, L, M, H, D, B, E, C, A

13) What is Stack? Why is it known as LIFO? Write an algorithm using PUSH and POP.

Ans) Stack is a type of linear data structure in which element may be inserted or deleted at one end only. It is called LIFO i.e. last in, first out as last element pushed onto STACK is first element to get popped

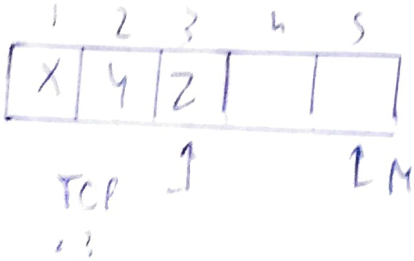
(A) PUSH (STACK, TOP, MAXSTK, ITEM)

1) [Stack full?]
   If TOP = MAXSTK, print overflow and exit
2) Set TOP = TOP + 1
3) Set Stack[TOP] = ITEM
4) Return

(A) POP (STACK, TOP, ITEM)

1) [Stack Empty?]
   If TOP = 0, print underflow and exit
2) Set ITEM = STACK[TOP]
3) Set TOP = TOP - 1
4) Return

Eg)



Linear Representation of Stack

Q14 Give idea of hashing and it use as hashing

hashing is a technique that is used to uniquely identify a specific object from a group of similar object.

ex In universities each student given unique roll no. that can be use to retrieve info. about them.

Assume that you have an object and you want to assign a key to it to make searching easy. To store key we can use array where key (integers) can used directly as index to store value. But in case of large key and not used directly as an index we use hashing.

In hashing large key is convert to small by using hashing function. The value are store in data structure called hash table.

The idea of hashing is to distribute entries (key) uniformly accross the array. By using that key we can access the element in O(1) time

hashing is implemented in 2 step:-

1) An element is converted into an integer by using hash function this element can be use as an index to store the original element which falls into the hash table.

2) The element is stored in hash table where it can be quickly retrieved using hash key.

```
hash = hash func(key)
index = hash % array_size
```

In this method the hash is independent of array size and it reduce to an index (a number b/w 0 and array_size - 1) by using modulus operator (%)

Hashing Function

The general idea of using the key to determine the address of a record is an excellent idea but it must be modified so that space is not wasted. This modification take the form of function H that transforms set K of keys into set L of memory address.

$$H: K \rightarrow L$$

is called hash or hashing function.

Unfortunately such a fn H may not yield distinct value it is possible that for 2 key K_1 and K_2 have same hash address.

It is called collision and to remove it some method is used

The 2 principal criteria used in selecting hash function $H: K \rightarrow L$ are :-

1) function H should be easy and quick to compute.
2) function H should, as far as possible, uniformly distribute the hash address throughout the set L so that there are min. no. of collision.

Some hash function

1) Division method :- choose no. m larger then n of key K

The hash function is defined by

$$H(K) = K \pmod m \quad \text{or} \quad H(K) = K \pmod m + 1$$

Here $K \pmod m$ denote the reminder when K is divided by m and $K \pmod m$ is use to show Hash address to range from 1 to m rather then from 0 to m-1.

2) Mid square method

The key K is squared and Hash function H is define as:

$$H(K) = L$$

where L is obtained by deleting digit from both end of K^2

Ans Folding method

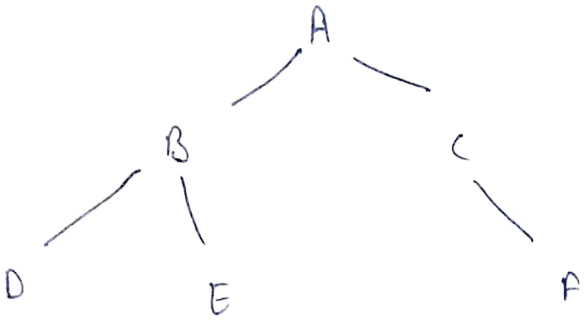
The key k is partitioned into no. of part $k_1 \ldots k_n$ where each part except possibly last has same no. of digit as required address. the part are added together ignoring the last carry

$$H(k) = k_1 + k_2 + k_3 \ldots k_r$$

where the leading digit carries if any, are ignored sometime bcz extra mildly the even no. part $k_2, k_4 \ldots$ are each reversed befor addition.

Q) Is Explain Inorder, Preorder and Postorder Traversal operation on Binary tree with example.

Ans



Consider a binary tree T. in it A is the root that it is left subtree L_1 consist of B, D and E and that it's right subtree R_1 consist of nodes C and F.

Traversing Binary tree

There are 3 standard way of traversing a binary tree T with root R. theses are :-

Pre order

1) Process the root.
2) traverse the left subtree of T in preorder
3) Traverse the right subtree of T in preorder

How does algo

1) Initially push NULL onto stack and then set PTR = root

Then current node

stack: ϕ

2) Proceed down the left most Path rooted at PTR: A as follow

i) Process A and Push it's right child C onto stack.

stack = ϕ C

ii) Process B and Push it's right child in stack.

stack: ϕ C E

3) Now Process D

No o there node to process

[Backtracking] Pop the top element E from stack and set PTR = H

stack = ϕ C

Since PTR ≠ Null return to step (A) of algo.

4) Since E has no child

[Back track] Pop C from stack and set PTR = C

Stack = ϕ

Since PTR ≠ NULL return to step (A) of algo.

5) Proceed down the left most Path rooted at PTR = C as follow

i) Process C and Push it right child F onto stack.

Stack = ϕ F

6) No other element left so [Back Tracking]

Pop F from stack and set PTR = F

Since PTR ≠ NULL return to step (A) of algo.

7) Since no child of F pop top element of the stack

that is NULL and set PTR = NULL

Since PTR = NULL then algo. completed.

Pre order = D B E A C F

Inorder Traversal

Traverse the left subtree of T in order

2) Process the root R.

3) Traverse the right subtree of R in order.

1) Initially Push NULL onto stack

Stack := ϕ

then set PTR := A

2) Proceed down the left most Path ~~rooted~~ rooted at PTR = A

Push A B and D on stack.

Stack = ϕ A B D

(no other node is push on to stack. as no child of D)

3) [Backtracking] The node D and B is pop and process

Stack = ϕ A

4) (we stop process at B as B has right child)

Set PTR = E right child of B

4) Push E in stack. no more child of E.

5) [Back Tracking] pop E

Stack = ϕ A

6 as no more node in left part. [Backtracking]

Pop A

Stack = ϕ

(no other element of stack is popped since A has right child] set PTR = C , Push the node C

Stack = ϕ C

7) no left child of C , Pop C and process.

Stack = ϕ

8) Since C has right child set PTR = F

Push F onto stack.

more element left [Backtracking] Pop F from stack.

The next element, NULL, is pop from stack.

The algo. finish since NULL is pop from stack.

Tree process in order D B E A C F

Post order Traversal

1) Traverse the left subtree of T in Post order
2) Traverse the right subtree of T in Post order
3) Process the root R.

Now

1) Initially push NULL onto stack set PTR = A
2) Process down left most Path and Push A B D, Since A ~~Stack~~ = ~~ϕ A B D~~ Has right child C Push - C onto stack after A but befor B, as B has right child push - E onto stack after B but Before D.

Stack = ϕ A - C B - D

3) Pop and Process D, since - E has negative sign only pop - E

Stack = ϕ A - C B

PTR = - E, reset PTR = E return to step 2

) Proceed down left most Path rooted PTR = H, first Push E onto Stack = ϕ A - C B E

) no more element to push Pop and Process E, and B and only pop - C, set PTR = - C

Reset PTR = C and Push on stack.

Stack = ϕ A C

Proceed down left most path rooted PTR = C
as C has only right child push -F onto stack.
Stack = Φ A C -F

6) Now only Push -f and set PTR = -f
reset PTR = F and return to step(a)

8) Push F onto stack B
Stack = Φ A C F

9) [Back Tracking] Pop and Process F, C, A when NULL is Popped the algo. is complete.

Post orders = D E B F C A

Ans:) Write the procedure to implement the adjacent matrix.

b) Define data structure graph how they are represented in memory.

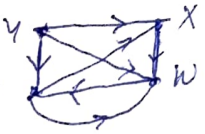
Ans 16) Let G be simple directed graph with m nodes and nodes of G are ordered and are called $v_1, v_2, \ldots v_m$. Then adjacency matrix $A = (a_{ij})$ of $m \times m$ matrix defined as follows:

$$a_{ij} = \begin{cases} 1 & \text{if } v_i \text{ is adjacent to } v_j \text{ i.e, if there's edge } (v_i, v_j) \\ 0 & \text{otherwise} \end{cases}$$

Such a matrix A which contains entries of 0 and 1 is called a bit matrix or Boolean matrix. The adjacency matrix A depends on nodes of G i.e different ordering of nodes result in different adjacency matrix

Suppose G is an undirected graph. Then adjacency matrix A of G will be a symmetric matrix i.e one in which $a_{ij} = a_{ji}$ for every i and j.

Example let us consider graph



Its nodes are stored in memory in a linear array DATA as follows

DATA: X, Y, Z, W.

Then we assume that ordering of the nodes in G is as follows: $v_1 = X$, $v_2 = Y$, $v_3 = Z$, $v_4 = W$.

The adjacency matrix A of G is as follows :-

$$A = \begin{pmatrix} 0 & 0 & 0 & 1 \\ 1 & 0 & 1 & 1 \\ 1 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix}$$

6) A graph is a data structure which consists of 2 things
1) A set V of elements called nodes (or points or vertices)
2) A set E of edges such that each edge e in E is identified with unique (unordered) pair [u, v] of nodes V, denoted by e.

A graph is said to be connected if there is a path between a two of its nodes.

Graph is represented is memory by 2 most common methods :-

1) (Adjacency matrix): It is a 2D array of size V × V where V is the number of vertices in a graph let 2D array be adj [][] a slot adj [i][j] = 1 indicates that there is an edge from vertex i to vertex j.

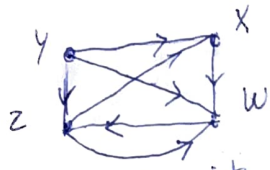
eg) Nodes of graph are stored in linear array DATA as follows :

DATA = X, Y, Z, W.

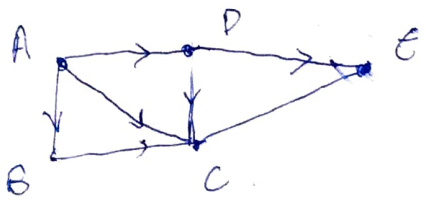
we assume ordering of the nodes in G is as follows : $v_1 = X$, $v_2 = Y$, $v_3 = Z$ and $v_4 = W$. The adjacency matrix A of G is as follows :

$$A = \begin{bmatrix} 0 & 0 & 1 & 1 \\ 1 & 0 & 0 & 1 \\ 1 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

, the number of 1's in A = number of edges in G



2) (Adjacency list) : Let G be a graph (directed) with m nodes. Each node of G is followed by its adjacency list, which is its list of adjacent nodes, also called its successors or neighbours. The linked representation contains 2 lists, a node list NODE and an edge list EDGE as follows :

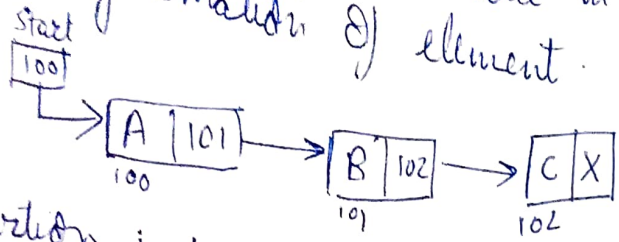


NODE	Adjacency list
A	B, C, D
B	C
C	
D	C, E
E	C

... do you mean by Link List? Write an algorithm to ...ert and delete a node in Singly Linked list.

A Link List is a linear collection of data elements called nodes where linear order is given by means of pointers. Each node is divided into 2 parts — LINK which contains address of next node in the list and INFO which contains information of element.

Eg)



Insertion: Insertion ~~at Beginning of list~~ after given Node

INSLOC (INFO, LINK, START, AVAIL, LOC, ITEM)

1) If AVAIL = NULL, then write overflow and exit.
2) Set NEW = AVAIL, AVAIL = LINK[AVAIL]
3) Set INFO[NEW] = ITEM.
4) If LOC = NULL, then
   Set LINK[NEW] = START and START = NEW
   Else
   Set LINK[NEW] = LINK[LOC] and LINK[LOC] = NEW
   [End of if structure]
5) Exit

Insertion into a Sorted list

- FIND A (INFO, LINK, START, ITEM, LOC)

1) If Start = NULL, set LOC = NULL and return
2) If ITEM < INFO[START], then: Set LOC = NULL and return
3) Set SAVE = START and PTR = LINK[START]
4) Repeat Step 5 and 6 while PTR ≠ NULL
5) If ITEM < INFO[PTR], then
   Set LOC = SAVE and return
   [End of if]
6) Set SAVE = PTR and PTR = LINK[PTR]

[End of step 4 loop]
7) Set LOC = SAVE
8) Return

INSERT (INFO, LINK, START, AVAIL, ITEM)
1) [Use FINDA to find location of node preceding ITEM.]
   Call FINDA [INFO, LINK, START, ITEM, LOC]
2) Call INSLOC [INFO, LINK, START, AVAIL, LOC, ITEM] and exit

Deletion : Deleting node following given node

DEL (INFO, LINK, START, AVAIL, LOC, LOCP)
1) IF LOCP = NULL, set START = LINK [START]
   else set LINK [LOCP] = LINK [LOC]
2) Set LINK [LOC] = AVAIL and AVAIL = LOC
3) Exit

Deleting node with given ITEM of information

FINDB (INFO, LINK, START, ITEM, LOC, LOCP)
1) If START = NULL, set LOC = NULL, LOCP = NULL and return
2) If INFO [START] = ITEM, then set LOC = START, LOCP = NULL & return
   [End of if]
3) Set SAVE = START and PTR = LINK [START]
4) Repeat step 5 and 6 while PTR ≠ NULL,
5) If INFO [PTR] = ITEM, then
   Set LOC = PTR and LOCP = SAVE and Return
   [End of if]
6) Set SAVE = PTR and PTR = LINK [PTR]
   [End of step 4 loop]
7) Set LOC = NULL and exit.

DELETE [INFO, LINK, START, AVAIL, ITEM)
1) Call FINDB [INFO, LINK, START, ITEM, LOC, LOCP)
2) If LOC = NULL, write item not in list and exit
3) If LOCP = NULL, set START = LINK [START]
   Else set LINK [LOCP] = LINK [LOC]
   [End of if]
4) Set LINK [LOC] = AVAIL and AVAIL = LOC
5) Exit

Q5 How does a linear search algo. work? Write complete algo. compute the complexity of linear search.

Ans

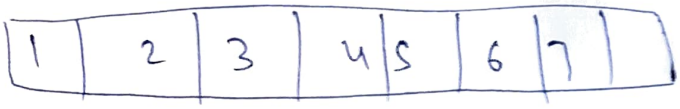
LINEAR (DATA, N, ITEM, LOC)

1) Set DATA[N+1] := ITEM
2) set LOC := 1
3) repeat while DATA(LOC) ≠ ITEM
   set LOC := LOC + 1
   [end of Loop]
4) [Successful?] if LOC = N+1 then set LOC := 0
5 EXIT

Ex

Suppose we wanto find 7 from above array.

* so accordind to our algo. 7 place at the end of array



By setting A[7] = 7

algo search the array. Since 7 appear in A[N+1]

i.e. 7 is not in original array.

generally in linear search the item we want to search is compare with each element of and if it is found it show the result if it is unsuccessful then exit.

Complexity of linear search

The complexity of linear search is measured by the number of comparison required to find item in Data where Data contain n element.

Case : discuss average and worst.

Clearly worst case happen when one must search thrugh entire data. ie. when item does not appear in DATA. In this case

$$f(n) = n + 1$$

Our time is proportional to n

complexity = $O(n)$

That is the case in our example use above.

Average case

In average case analysis we take all possible input and compute time for all the input. Sum of all computed value divided by total no. of input.

$$\text{Average case time} = \sum_{i=1}^{i=n+1} \frac{O(i)}{n+1}$$

$$= \frac{O\left((n+1) + \sum (n+2)/2\right)}{n+1}$$

$$= O(n)$$