B.TECH. DEGREE EXAMINATION, JAN 2023

Third Semester

Computer Science and Engineering

DATA STRUCTURES

(2013-14 Regulation)

PART-A

1. List out the steps in analyzing program.

1. Requirements
2. Design
3. Analysis
4. Refinement & coding
5. Verification

2. Define multidimensional Array.

If three or more subscript or index is required to refer the elements of an array then the array is called as three dimensional or multidimensional arrays.

DECLARATION OF AN ARRAY:

The declaration can be as follows:

```
data type array name[s1][s2][s3].....[sn];
```

3. Write the postfix form for the expression A+B*(C-D)/(P-R)

ABCD-*PR-

4. Mention the applications of Queue data structure.

1. Semaphores.
2. FCFS (first come first serve) scheduling, example: FIFO queue.
3. Spooling in printers.
4. Buffer for devices like keyboard.
5. CPU Scheduling.
6. Memory management.

5. Compare binary tree and Binary Search tree.

Binary tree	**Binary Search Tree**
A nonlinear data structure known as a Binary Tree is one in which each node can have a maximum of two child nodes.	A BST is a binary tree with nodes that has right and left subtrees that are also binary search trees.
Since Binary Trees are not ordered, the processes of inserting, deleting, and finding them take significantly more time.	Due to their ordered characteristics, insertion, deletion, and searching of an element is faster in a BST than in a Binary Tree.

6. What is the minimum degree of B tree?

Min-degree: Each node of the tree except the root must contain at least t − 1 keys (and hence must have at least t children if it is not a leaf).

7. State the advantage of minimum Spanning Tree.

A minimum spanning tree is used to **find easy paths in maps** and is also used for designing networks such as water supply networks or any electrical grid networks.

8 .Define transitive closure.

Transitive Closure it the reachability matrix to reach from vertex u to vertex v of a graph. One graph is given, we have to find a vertex v which is reachable from another vertex u, for all vertex pairs (u, v).

9. Differentiate Static and Dynamic tree table.

Static tree table	**Dynamic tree table**
1. Data type size is fixed	1. Memory allocation is done during program execution.
2. Memory allocation before program execution.	2. More efficient.
3. Stack is used.	3. Heap is used.

10. Give the example for External Sorting.

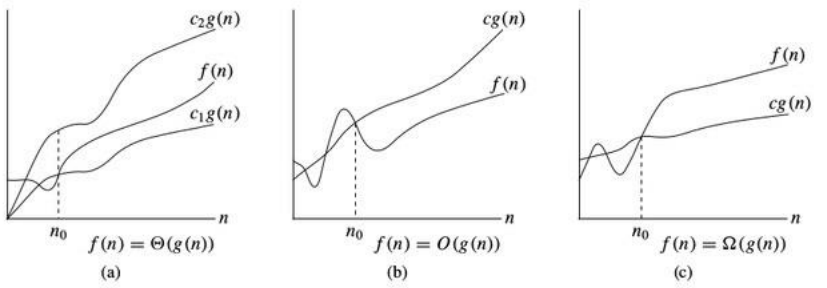
Tape drives and disk arrays are some of the example of external sorting.

PART-B

UNIT-1

11. Explain about the concept of algorithmic notation programming principles.

Big-O Notation
The Big-O notation describes the worst-case running time of a program. We compute the Big-O of an algorithm by counting how many iterations an algorithm will take in the worst-case scenario with an input of N. We typically consult the Big-O because we must always plan for the worst case. For example, O(log n) describes the Big-O of a binary search algorithm.
Big-Ω Notation
Big-Ω (Omega) describes the best running time of a program. We compute the big-Ω by counting how many iterations an algorithm will take in the best-case scenario based on an input of N. For example, a Bubble Sort algorithm has a running time of Ω(N) because in the best case scenario the list is already sorted, and the bubble sort will terminate after the first iteration.
Big-Θ Notation
We compute the big-Θ of an algorithm by counting the number of iterations the algorithm *always* takes with an input of n. For instance, the loop in the pseudo code below will always iterate N times for a list size of N. The runtime can be described as Θ(N).



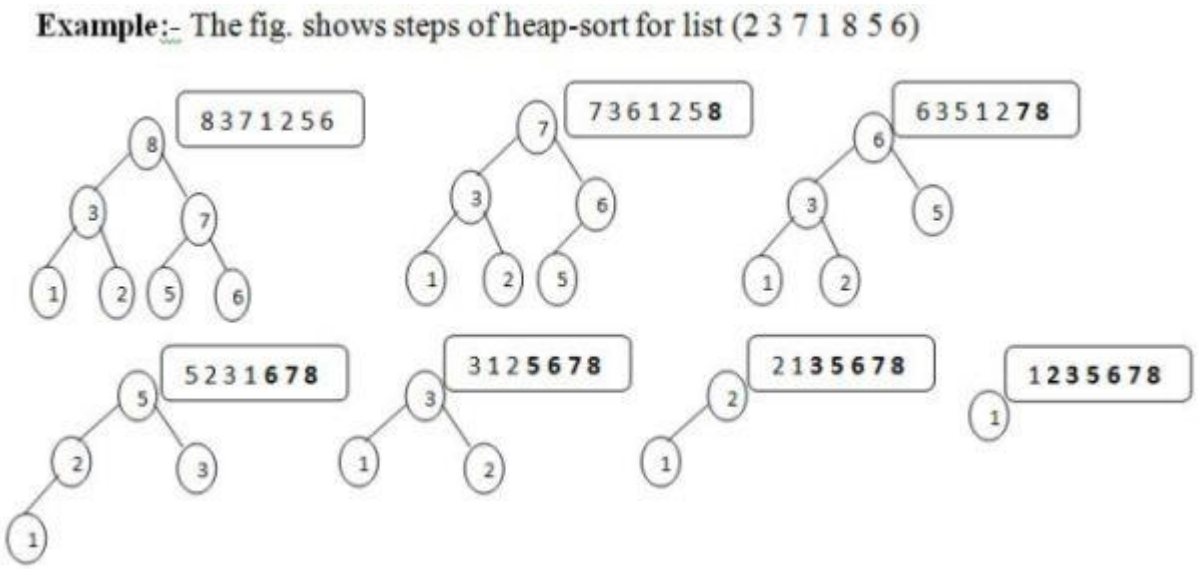
12. Describe the procedure to sort the elements using heap sort with an example.

A heap is a complete binary tree, and the binary tree is a tree in which the node can have the utmost two children. A complete binary tree is a binary tree in which all the levels except the last level, i.e., leaf node, should be completely filled, and all the nodes should be left-justified.

Heap sort is a popular and efficient sorting algorithm. The concept of heap sort is to eliminate the elements one by one from the heap part of the list, and then insert them into the sorted part of the list.

Heap sort is the in-place sorting algorithm.

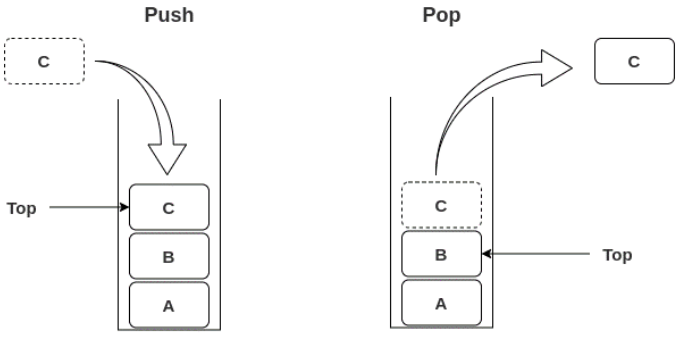
1. HeapSort(arr)
2. BuildMaxHeap(arr)
3. for i = length(arr) to 2
4. swap arr[1] with arr[i]
5. heap_size[arr] = heap_size[arr] ? 1
6. MaxHeapify(arr,1)
7. End



UNIT-II

13. Interpret about evaluation of arithmetic expression using stack with an example.

Stack is a linear data structure that follows a particular order in which the operations are performed. The order may be LIFO(Last In First Out) or FILO(First In Last Out). LIFO implies that the element that is inserted last, comes out first and FILO implies that the element that is inserted first, comes out last.



Stack Data Structure

There are many real-life examples of a stack. Consider an example of plates stacked over one another in the canteen. The plate which is at the top is the first one to be removed, i.e. the plate which has been placed at the bottommost position remains in the stack for the longest period of time. So, it can be simply seen to follow LIFO(Last In First Out)/FILO(First In Last Out) order.

Expression Evaluation Using Stack

Algorithm

Step 1: Create two stacks - the operand stack and the character stack.
Step 2: Push the character to the operand stack if it is an operand.
Step 3: If it is an operator, check if the operator stack is empty.
Step 4: If the operator stack is empty, push it to the operator stack.
Step 5: If the operator stack is not empty, compare the precedence of the operator and the top character in the stack.
If the character's precedence is greater than or equal to the precedence of the stack top of the operator stack, then push the character to the operator stack.
Otherwise, pop the elements from the stack until the character's precedence is less or the stac k is empty.
Step 6: If the character is "(", push it into the operator stack.
Step 7: If the character is ")", then pop until "(" is encountered in the operator stack.

Example

2 * (5 * (3 + 6)) / 5 - 2

Character	Action	Operand Stack	Operator Stack
2	Push to the operand stack	2	
*	Push to the operator stack	2	*
(	Push to the operator stack	2	(*
5	Push to the operand stack	5 2	(*
*	Push to the operator stack	5 2	* (*
(	Push to the operator stack	2 1	(* (*
3	Push to the operand stack	3 5 2	(* (*
+	Push to the operator stack	3 2 1	+ (* (*
6	Push to the operand stack	6 3 5 2	+ (* (*
)	Pop 6 and 3	5 2	+ (* (*
	Pop +	5 2	(* (*
	6 + 3 = 9, push to operand stack	9 5 2	(* (*

	Pop (	9 5 2	* (*
)	Pop 9 and 5	2	* (*
	Pop *	2	(*
	9 * 5 = 45, push to operand stack	45 2	(*
	Pop (	45 2	*
/	Push to the operator stack	45 2	/ *
5	Push to the operand stack	5 45 2	/ *
-	Pop 5 and 45	2	/ *
	Pop /	2	*
	45/5 = 9, push to the operand stack	9 2	*
	Pop 9 and 2		*
	Pop *		
	9 * 2 = 18, push to operand stack	18	
	Push - to the operator stack	18	-
2	Push to the operand stack	2 18	-
	Pop 2 and 18		-
	Pop -		
	18 - 2 = 16, push to operand stack	16	

The final answer (here 16) is stored in the stack and is popped at the end.

14. Explain the steps involved in Insertion and Deletion into a Circular Linked list.

Circular Linked List is a variation of Linked list in which the first element points to the last element and the last element points to the first element. Both Singly Linked List and Doubly Linked List can be made into a circular linked list.

Basic Operations

Following are the important operations supported by a circular list.

- **insert** − Inserts an element at the start of the list.
- **delete** − Deletes an element from the start of the list.
- **display** − Displays the list.

Insertion Operation

Following code demonstrates the insertion operation in a circular linked list based on single linked list.

Example

```
insertFirst(data):
Begin
   create a new node
   node -> data := data
   if the list is empty, then
      head := node
      next of node = head
   else
      temp := head
      while next of temp is not head, do
      temp := next of temp
      done
      next of node := head
      next of temp := node
      head := node
   end if
End
```

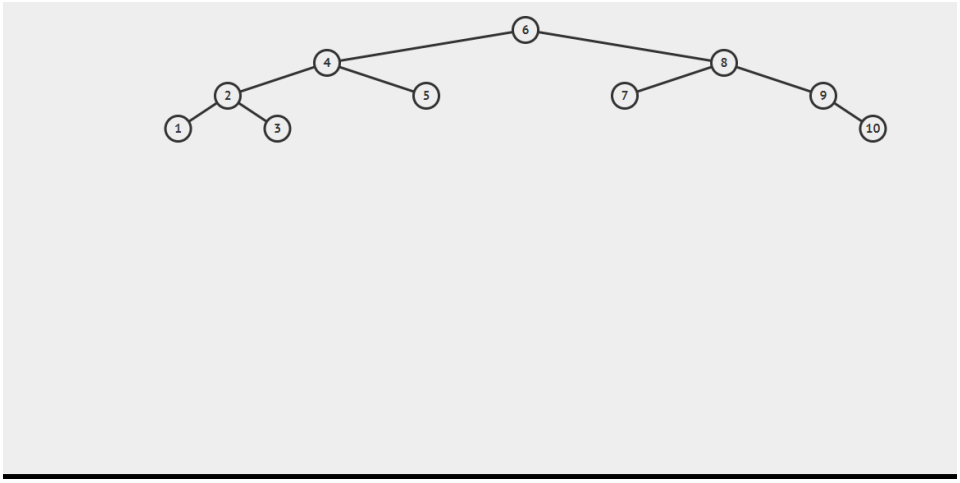
Deletion Operation

Following code demonstrates the deletion operation in a circular linked list based on single linked list.

```
deleteFirst():
Begin
   if head is null, then
      it is Underflow and return
   else if next of head = head, then
      head := null
      deallocate head
   else
      ptr := head
      while next of ptr is not head, do
         ptr := next of ptr
      next of ptr = next of head
      deallocate head
      head := next of ptr
   end if
End
```

UNIT –III

15. Insert the following elements in the empty AVL tree and how do you balance the tree after each element insertion. The Elements are 2,5,4,6,7,9,8,3,1,10.



16. Define B tree indexing with its advantages and disadvantages .Explain why B+ tree used for indexing?

1. B-tree indices are similar to B + -tree indices.
   - Difference is that B-tree eliminates the redundant storage of search key values.
   - In B + -tree of Figure 11.11, some search key values appear twice.
   - A corresponding B-tree of Figure 11.18 allows search key values to appear only once.
   - Thus we can store the index in less space.

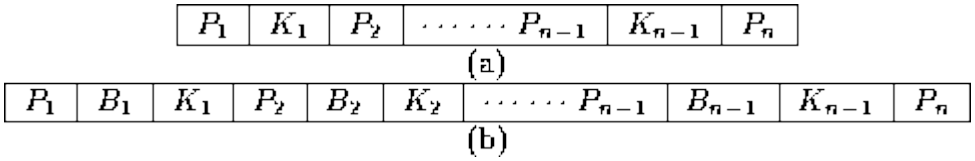


Figure 11.8: Leaf and nonleaf node of a B-tree.

2. **Advantages:**
   - Lack of redundant storage (but only marginally different).
   - Some searches are faster (key may be in non-leaf node).
3. **Disadvantages:**
   - Leaf and non-leaf nodes are of different size (complicates storage)
   - Deletion may occur in a non-leaf node (more complicated)

Generally, the structural simplicity of B + -tree is preferred.

UNIT –IV

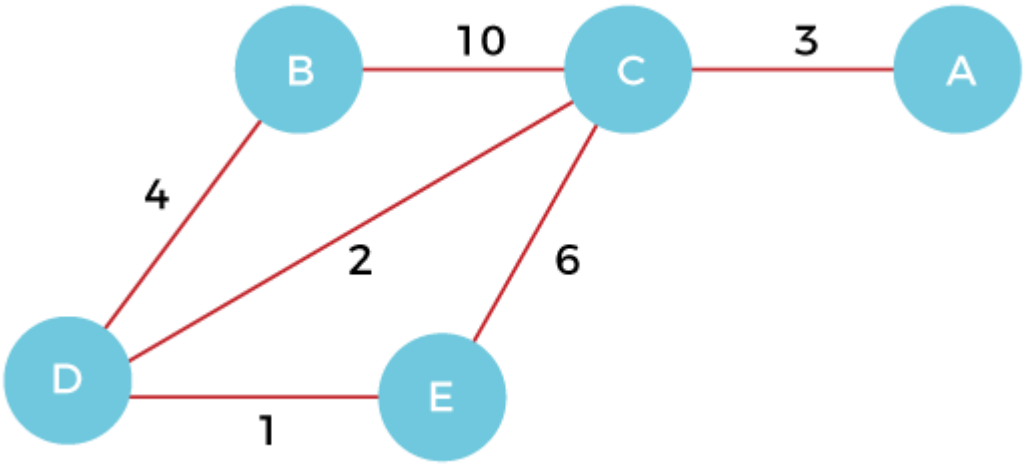
17. Illustrate the Prim's algorithm with an example.

Prim's Algorithm is a greedy algorithm that is used to find the minimum spanning tree from a graph. Prim's algorithm finds the subset of edges that includes every vertex of the graph such that the sum of the weights of the edges can be minimized.

Prim's algorithm starts with the single node and explores all the adjacent nodes with all the connecting edges at every step. The edges with the minimal weights causing no cycles in the graph got selected.

Now, let's see the working of prim's algorithm using an example. It will be easier to understand the prim's algorithm using an example.

Suppose, a weighted graph is -



Step 1 - First, we have to choose a vertex from the above graph. Let's choose B.

Step 2 - Now, we have to choose and add the shortest edge from vertex B. There are two edges from vertex B that are B to C with weight 10 and edge B to D with weight 4. Among the edges, the edge BD has the minimum weight. So, add it to the MST.

Step 3 - Now, again, choose the edge with the minimum weight among all the other edges. In this case, the edges DE and CD are such edges. Add them to MST and explore the adjacent of C, i.e., E and A. So, select the edge DE and add it to the MST.

Step 4 - Now, select the edge CD, and add it to the MST.

Step 5 - Now, choose the edge CA. Here, we cannot select the edge CE as it would create a cycle to the graph. So, choose the edge CA and add it to the MST.

So, the graph produced in step 5 is the minimum spanning tree of the given graph.

18. Elaborate the operations on Sets in the data structure with an example.

A **set** is a well-defined collection of objects. The objects may be numbers, alphabets, names of people, etc. Sets are represented using upper-case letters such as A, B, etc. For example:
A = {a, e, i, o, u}
A is a set of vowels in the English alphabet

Intersection and Union of Sets
Intersection
The intersection of two sets A and B is a set that contains all the elements that are common to both A and B.

Set Operations include **Set Union, Set Intersection, Set Difference, Complement of Set, and Cartesian product**

Subset and Proper Subset

Subset
For two set A and B, A is a subset of B if every element in A is also in B. A can be equal to B. This is formally written as

UNIT-V

19. Write short notes on:

(A).Rectangular tables:

Tables are very often in rectangular form with rows and columns. Most programming languages accommodate rectangular tables as 2-D arrays. Rectangular tables are also known as matrices. Almost all programming languages provide the implementation procedures for these tables as they are used in many applications. Fig. Rectangular tables in memory logically, a matrix appears as two-dimensional, but physically it is stored in a linear fashion. In order to map from the logical view to physical structure, an indexing formula is used. The compiler must be able to convert the index (i, j) of a rectangular table to the correct position in the sequential array in memory.

(b).Symbol Tables

A symbol table is a major data structure used in a compiler:

1. Associates attributes with identifiers used in a program
2. For instance, a type attribute is usually associated with each identifier
3. A symbol table is a necessary component

Use of identifiers may appear in many places of the program text

Identifiers and attributes are entered by the analysis phases

When processing a definition (declaration) of an identifier

In simple languages with only global variables and implicit declarations:

1. The scanner can enter an identifier into a symbol table if it is not already there
2. In block-structured languages with scopes and explicit declarations:
3. The parser and/or semantic analyzer enter identifiers and corresponding attributes

4. Symbol table information is used by the analysis and synthesis phases
5. To verify that used identifiers have been defined (declared)
6. To verify that expressions and assignments are semantically correct – type checking
7. To generate intermediate or target code

Symbol Table Interface

The basic operations defined on a symbol table include:

1. allocate – to allocate a new empty symbol table
2. free – to remove all entries and free the storage of a symbol table
3. insert – to insert a name in a symbol table and return a pointer to its entry
4. lookup – to search for a name and return a pointer to its entry
5. set_attribute – to associate an attribute with a given entry
6. get_attribute – to get an attribute associated with a given entry

Other operations can be added depending on requirement

For example, a delete operation removes a name previously inserted

1. Some identifiers become invisible (out of scope) after exiting a block
2. This interface provides an abstract view of a symbol table
3. Supports the simultaneous existence of multiple tables

20. Discuss about the concept of sorting with tapes and disks

External sorting is a term for a class of sorting algorithms that can handle massive amounts of data. External sorting is required when the data being sorted does not fit into the main memory of a computing device (usually RAM) and instead, must reside in the slower external memory (usually a hard drive).

External sorting typically uses a hybrid sort-merge strategy. In the sorting phase, chunks of data small enough to fit in the main memory are read, sorted, and written out to a temporary file. In the merge phase, the sorted sub-files are combined into a single larger file.

When We do External Sorting?

- When the unsorted data is too large to perform sorting in computer internal memory then we use external sorting.
- In external sorting we use the secondary device. in a secondary storage device, we use the tape disk array.
- when data is large like in merge sort and quick sort.
- Quick Sort: best average runtime.
- Merge Sort: Best Worse case time.
- To perform sort-merge, join operation on data.
- To perform order by the query.
- To select duplicate element.
- Where we need to take large input from the user.

Examples:

- Merge sort
- Tape sort
- Polyphase sort
- External radix
- External merge