CS301 Data Structures Lecture Wise Questions and Answers For Final Term Exam Preparation By Virtualians Social Network

Lec 23:

Double rotation is simply making a single rotation twice, once left then right or vice versa. While inserting a node in an AVL tree, if we have to balance the tree, then we need to do only one single right, left or one double rotation to balance the tree.

In deletion from an AVL tree, we delete node as we delete it in a BST.

After deleting the node, we traverse up the tree from the deleted node checking the balance of each node at each level up to the root.

Deletion cases in AVL:

Case1a: The parent of the deleted node had a balance of 0 and the node was deleted in the parent's left sub tree.

Action→ Change the balance of the parent node and stop.

Case1b: the parent of the deleted node had a balance of 0 and the node was deleted in the parent's right sub tree.

Action→ change the balance of the parent node and stop.

Case2a: Parent of the deleted node had a balance of 1 and the node was deleted in the parent's left sub tree.

Action→ Change the balance of the parent node. This deletion of node may have caused imbalance in higher nodes so it is advised to continue up to the root of the tree.

> Q.when we use " using namespace std" what does it mean? often time i hear the word **only namespace,** what is meant by namespace alone?
>
> Ans:
>
> It is not mandatory to use **using namespace std;** in the beginning of your program, however it is better to use it. ***It is a combination of some objects, class and functions*** etc.

Q. Why Hoffman tree is not an AVL Tree?

Ans

For a tree to be AVL, there are two conditions:
1) Tree must be Binary Search Tree
2) For every node, height of left and right subtree may differ by at most 1

For Huffman encoding, we also need a Binary tree to get the codes of characters. And we have different method to build that Binary tree.
Very Important: Binary tree used in Huffman encoding is not even Binary Search Tree.

Q. Abstract data type (ADT)

Ans

Abstract data type (ADT): A data type whose properties (domain and operations) are specified independently of any particular implementation.

List, for example, is an ADT. The functionality of an ADT can be implemented in many ways. The programs that use these ADTs will not need to know which implementation was used as long as it performs the required functionality.

The List ADT can be implemented through Arrays (elementary data structure that exists as built-in in most programming languages) or linked-structure (list implemented through linked-structure is called Linked-list).

Programming languages provide some built-in data types, also called primitive data types. Like C++ language provides int, float, char etc as built-in data types.

Lec : 24:

Case 2b→ the parent of the deleted node had a balance of -1 and the node was deleted in the parent's right subtree.

Action→ Change the balance of the parent tree and in case the imbalance occurs in the higher nodes, continue up the tree.

Case 3a→ The parent had balance of -1 and the node was deleted in the parent's left subtree while right subtree was un-balanced.

Action→ Perform single rotaion, adjust balance. Balance of the higher nodes will not be affected so that's it.

Case 4a→ Parent had balance of -1 and the node was deleted in the parent's left subtree, right subtree was unbalanced.

Action→ Double rotation to bring the root(B) upwards.

Case 5a→ the parent had balance of -1 and the node was deleted in the parent's left subtree, right subtree was unbalanced.

Acction→ Single rotation at B. In case of unbalance, continue up the tree.

Expression trees→ Parse trees and abstract syntax trees , which are significant components of compilers.

The parse trees are used in query processing. The queries are questions to the database to see a particular kind of data. The language used to query is called SQL(Structured Query Language).

Optimizers use the expressions trees and convert them into graphs for efficiency purposes.

Q. Binary Tree ?

Ans

Binary Tree is a tree data structure in which each node has at most two children. Typically the child nodes are called left and right.

Q. Binary Search Tree ?

Ans

Binary Search Tree (BST) is a binary tree data structure in which
->the left subtree of a node contains only values less than the node's value;
->the right subtree of a node contains only values greater than or equal to the node's value.

Q. AVL tree ?

Ans

AVL Tree is a special sub type of BST called self-balancing binary search tree. In an AVL tree the heights of the two child sub trees of any node differ by at most one, therefore it is also called height-balanced BST. Additions and deletions may require the tree to be rebalanced by one or more tree rotations to ensure this balance of differ at most by one only.

Lec 25:

Huffman encoding → basically relates to the compression of data. To make the computer networks faster, one can adopt one of the two choices i.e. increase the data rate of transmission or send the less data. Here sending less data doesn't mean to send in-complete information/data but to compress the data.

Huffman Algorithm's working :

Liste all the letters, spaces, frequency in which they occur in a message.

Consider each of these character/ frequency pairs to be node.

Pick the two nodes with the lowest frequency. In case of a tie, pick randomly.

Make a new node out of these two, and turn two nodes into its children.

That new node will be assigned the sum of the frequencies of its children.

Continue the combining process of two nodes of the lowest frequency till only one node which is the root, is left.

Q. Inorder Traversal, Topic of parse tree in compiler is given at page no 275 of handouts. In this topic term, factor and id are used in tree. Please explain these words in parse tree. 2. Inorder traversal tree routine given at page no. 281 is different from the one shown in lecture presentation. Is it a second solution of Inorder traversal with parenthesis?

Ans

1. These are different terminologies which have used for identifier (operand) in an expression.
2. Yes, this is the second version of inorder traversal function for the infix expression having parenthesis.

Q. What are ***Basic steps of recursive programs ?***

Ans: All recursive programs follows the same basic sequence of steps:
1: Initialize the algorithm. Recursive programs often need a seed value to start with.
2: Check to see whether the current value(s) being processed match the base case. If so, process and return the value.
3: Redefine the answer in terms of a smaller or simpler sub-problem or sub-problems.
4: Run the algorithm on the sub-problem.
5: Combine the results in the formulation of the answer.
6: Return the results

Lec 26:

Generating Huffman code :

1. Start at the root, assigning 0's to left branch and subtrees and1.s to the right branch and subtrees.

2. Traverse the tree from the root to the character leaf node reading off the 0's and 1's along the path.

Note that in Huffman encoding, code of letter is of variable length.

There will be need of 5, 4 or 3 bits to represent a character whereas in ASCII code, 8 bits are required for each character. So Huffman encoding is more appropriate in case of compressing the data.

Q. how to find tree balance

Ans

In order to check that whether a tree is AVL or not, we need to check the balance factor of each node, which should be 0, 1 or -1.

The balance factor of a node can be calculated as,

Number of levels in the left sub tree minus the number of levels in the right sub tree.

For example, if the number of levels in the left sub tree of a node is 3 and the number of levels in its right sub tree is 4, then the balance factor is -1 (i.e. 3-4 = -1).

Q. **"Threaded binary tree "?**

Ans

It is in fact a binary tree with property that all right child pointers that would normally be null point to the inorder successor of the node, and all left child pointers that would normally be null point to the inorder predecessor of the node.

Lec 28:

An extra node (dummy) in the binary tree and would be inserted. The left pointer of this node would point to the root node of the tree while the right pointer will be pointing to itself. Here's the routine :

```
/* This routine will traverse the binary search tree */
void fastInorder(TreeNode* p)
{
while((p=nexInorder(p)) != dummy) cout << p->getInfo();
}
```

A **Complete binary tree** is a tree that is completely filled with the possible exception of the bottom level.

Here possible exception means that the tree might or might not be filled/complete/having both or one child at the bottom level.

The bottom level is filled from left to right.

For any **Array element** at position i, the left child would be at 2i the right would be at (2i+1) and the parent would be at floor(i/2)

Regards from Persian !

Q. Dummy node ?

Ans.

An extra node (dummy) in the binary tree and would be inserted. The left pointer of this node would point to the root node of the tree while the right pointer will be pointing to itself.
Here's the routine :

```
/* This routine will traverse the binary search tree */
void fastInorder(TreeNode* p)
{
while((p=nexInorder(p)) != dummy) cout p->getInfo();
}
```

Q. A complete binary tree ?

Ans

A Complete binary tree is a tree that is completely filled with the possible exception of the bottom level.
Here possible exception means that the tree might or might not be filled/complete/having both or one child at the bottom level.
The bottom level is filled from left to right.

Q. Algorithm ?

Ans

For any Array element at position i, the left child would be at 2i the right would be at (2i+1) and the parent would be at floor(i/2)

Lec 29:

Two things to be kept in mind while constructing a data structure are memory and time.

One of the problems with arrays is that the memory becomes useless inc ase of too many empty positions in the array.

Heap : it is basically used in priority queue which does not follow the FIFO rule.

in a (min) heap for every node X, the key in the parent is smaller than (or equal to) the key in X. In other words, the parent node has key smaller than or equal to both of its children nodes. This means that the value in the parent node is less than the value on its children nodes. This situation is reversed in case of a (max) heap.

Regards from Persian !

Q. constructing a data structure ?

Ans

Two things to be kept in mind while constructing a data structure are memory and time.
One of the problems with arrays is that the memory becomes useless inc ase of too many empty positions in the array.
Heap : it is basically used in priority queue which does not follow the FIFO rule.
in a (min) heap for every node X, the key in the parent is smaller than (or equal to) the key in X. In other words, the parent node has key smaller than or equal to both of its children nodes. This means that the value in the parent node is less than the value on its children nodes.
This situation is reversed in case of a (max) heap.

Q. **the dynamic memory allocation is done from the Heap memory; i.e. once I use new keyword, then memory is allocated from Heap to the specific variable type. Can we say that the HEAP memory for all applications are managed in the form of complete binary tree, as discussed in lecture No 29?**

Ans

The heap memory for all applications will be of course managed/implemented thourgh some data structure, however it depend on the operating system that which data strutue it use for heap memory, it may be heap or some other data structure.

Q. what is the basic purpose of tree? and why we use these in data structure ?

Ans

Tree is a useful data strucutr and each data structure is in fact organizing and storing of Data in Computer, in order to make the use of data efficient. We use tree in data structrue because it is a useful data structure.

Q. what is the difference b/w heap & min heap? as you told in lecture 30?

Ans

Heap is a complete binary tree which satisfies heap order (heap property), it is a data structure while **Min Heap** is a type of heap in which each node has smaller value than its Childs.

Lec 30:

Figure 30.4 : This is not a binary search tree. Here we are only fulfilling one condition that the parent value should be less than that of its two-children.

Deleting from a Min-Heap (deleteMin) :

We want to write a function deleteMin(), which will find and delete the minimum number from the tree.

Finding the minimum is easy; it is at the top of the heap.
deleteMin(): heap size is reduced by 1

Following are some of the important facts about building a heap.
− Suppose we are given as input N keys (or items) to build a heap of the keys.
− Obviously, this can be done with N successive inserts.
− Each call to insert will either take unit time (leaf node) or log2N (if new key percolates all the way up to the root).
− The worst time for building a heap of N keys could be Nlog2N.
− It turns out that we can build a heap in linear time.

The following algorithm will build a heap out of N keys.
for(i = N/2; i > 0; i--)
percolateDown(i);

Regards from Persian !

Q. where these trees are used?

Answer:

There are a lot of tree uses (application), for example,

1. Searching Data (Use BST)
2. Reducing data size, using Huffman encoding.
3. File system on computers also use the tree data structures.
4. Tree data structures are also use in routers and networking.
5. Trees are used to store data having hierarchal in nature.

Q. insertion in Heap process i cannot understand the concept of 2i and 2i+1 please explain it ?

Answer:

Heap data structure is implemented like complete binary tree through array, and we know that in complete binary tree when an element (tree node) is stored at **i th** position (where i is an array index) then its left child will store at **2i** position and its right child will store at **2i +1** position.

For example, if a heap (complete binary tree) node is store at position 2 (**i th**) then its left child will store in the array at position 4 (2i) and its right child will store in the array at position 5 (2i +1).

Q. Priority queue ?

Priority queue is like a regular queue or stack data structure, but additionally, each element is associated with a "priority".There are different techniques to implement it.
The most simple and inefficient technique is, to keep all the elements in an unsorted list. Whenever the highest-priority element is requested, search through all elements for the one with the highest priority.

Secondly to get better performance, priority queues typically use a heap as their backbone, giving O(log n) performance for inserts and removals, and O(n) to build initially.

Q. in order successor?

Ans

In Binary Tree, Inorder successor of a node is the next node in Inorder traversal of the Binary Tree.
Inorder Successor is NULL for the last node in Inoorder traversal.

Lec 31:

BuildHeap :

• The general algorithm is to place the N keys in an array and consider it to be an unordered binary tree.
• The following algorithm will build a heap out of N keys.

```
for ( i = N/2; i > 0; i-- )
percolateDown(i);
```

why we have started the 'for loop' with *i = N/2*. We can start *i* from N. As *i* moves from level to level in the tree. If we start *i* from N, it will start from the right most leaf node. As long as *i* is l less than N/2 it will remain on leaf nodes level. Is the leaf node alone is *min-heap* or not? Yes it is. As it does not have any left or right child to compare that it is smaller or not. All the leaf nodes satisfy the *min-heap* definition. Therefore we do not need to start with the leaf nodes.

Other Heap Methods :

decreaseKey(p, delta) takes a pointer to the node that may be the array position as we are implementing it as an array internally. The user wants to decrease the value of this node by delta.

increaseKey(p, delta) will increase the value of the element by delta. These methods are useful while implementing the priority queues using heap.

remove(p) removes the node at position p from the heap.
Q. Huffman coding ?
Ans
Huffman coding is a statistical technique which attempts to reduce the amount of bits required to represent a string of symbols. The algorithm accomplishes its goals by allowing symbols to vary

in length. Shorter codes are assigned to the most frequently used symbols, and longer codes to the symbols which appear less frequently in the string (that's where the statistical part comes in).

Q. A Threaded Binary ?
Ans
A Threaded Binary Tree is a binary tree in which every node that does not have a right child has a thread (in actual sense, a link) to its INORDER successor, and a binary tree in which every node that does not have a left child has a thread (in actual sense, a link) to its INORDER predecessor.
Q. **Heap and Min Heap ?**
Ans

Heap is a complete binary tree which satisfies heap order (heap property), it is a data structure while **Min Heap** is a type of heap in which each node has smaller value than its Childs.

Lec 32 :
deleteMin method : The following two code lines perform this task.
minItem = array[1];
array[1] = array[currentSize--];
perculateDown method : It takes the array index as an argument and starts its functionality from that index. In the code, we give the name *hole* to this array index.
The getMin method gives us the minimum value in the heap, which is not deleted from the heap.

```
template <class eType>
const eType& Heap<eType>::getMin( )
{
if( !isEmpty( ) )
return array[ 1 ];
}
```

Theorem : "For a perfect binary tree of height *h* containing $2^{h+1} - 1$ nodes, the sum of the heights of nodes is $2^{h+1} - 1 - (h+1)$, or *N-h-1".*
If there are N nodes in a tree, than it will have N-1 links.
points that prove the theorem :

• For any node in the tree that has some height *h*, darken h tree edges
–Go down the tree by traversing left edge then only right edges.
• There are $N - 1$ tree edges, and *h* edges on right path, so number of darkened edges is $N - 1 - h$, which proves the theorem.

If there are N nodes in a tree, than it will have N-1 links

Q. where these trees are used ?

Ans

There are a lot of tree uses (application), for example,

1. Searching Data (Use BST)
2. Reducing data size, using Huffman encoding.
3. File system on computers also use the tree data structures.
4. Tree data structures are also use in routers and networking.
5. Trees are used to store data having hierarchal in nature.

Q. Heap data structure

Ans

Heap data structure is implemented like complete binary tree through array, and we know that in complete binary tree when an element (tree node) is stored at **i th** position (where i is an array index) then its left child will store at **2i** position and its right child will store at **2i +1** position.

For example, if a heap (complete binary tree) node is store at position 2 (**i th**) then its left child will store in the array at position 4 (2i) and its right child will store in the array at position 5 (2i +1)

Lec 33 :

The Selection Problem :

Suppose, we have list of N names (names of students or names of motor vehicles or a list of numbers of motor vehicles or list of roll numbers of students or id card numbers of the students, whatever). However, we are confronting the problem of finding out the kth smallest element. Suppose we have a list of 1000 numbers and want to find the 10th smallest number in it.

The sorting is applied to make the elements ordered. After sorting out list of numbers, it will be very easy to find out any desired smallest number.
− One way is to put these *N* elements in an array and sort it. The *k*th smallest of these is at the *k*th position.

A faster way is to put the *N* elements into an array and apply the *buildHeap* algorithm on this array.
Finally, we perform *k deleteMin* operations. The last element extracted from the heap is our answer.
The interesting case is *k* = À*N*/2Â as it is also is known as the *median*.
Heap Sort : If *k* = *N*, and we record the *deleteMin* elements as they come off the heap. We will have essentially sorted the *N* elements.
Disjoint Set ADT - Equivalence Relations :
A binary relation R over a set S is called an *equivalence relation* if it has following propertiesReflexivity: for all element x ∈ S, x R x
2. Symmetry: for all elements x and y, x R y if and only if y R x
3. Transitivity: for all elements x, y and z, if x R y and y R z then x R z
− The relation "is related to" is an equivalence relation over the set of people.
You are advised to read about equivalence relations yourself from your text books or from the internet.

Q. a min heap that is shown in lec 29 is not an complete binary tree coz all of its non leaf nodes are not at depth d,But while deivering lec u said it is complete binary tree and also it is mentioned in handouts but it is not fulling the condition of complete binary tree then how can it be complete binary tree ?? guide
Ans

The tree given in the lecture no. 29 is a complete binary tree, the definition of complete binary tree is given below,

Complete Binary Tree: A binary tree in which every level except possibly the deepest (Last level) is completely filled, is called complete binary tree.

On the basis of this definition the tree given in the lecture no. 29 of handouts is complete binary tree because in this tree only the deepest level is not completely filled, so the tree is complete binary tree.

Q. when inserting or deleting a node from a heap then we will have to show the array implementation of it step by step with it or not???

Ans

You can observe that the insertion and deletion from the heap has performed step by step in the handouts, the complete code of these functions is available in the handouts.
Q. can we take the double quotes like "" as a character that has a frequency as 1. am true or not??

Ans
Double quote (“ ”) is not part of the message, you do not need to consider it as character.

Q. In build heap what will be procedure for inserting a left or right sub trees mean the value of nodes will be considered or not?
Ans
In the **buildHeap** procedure we do not insert the left or right subtree, in this procedure we create the heap (Complete binary tree which satisfy heap order) from an already filled array.

Lec 34:
Dynamic Equivalence Problem :
If the relation is stored as a two-dimensional array of booleans, this can be done in constant time. The problem is that the relation is usually not explicitly defined, but shown in implicit terms. Suppose we are given 1000 names of persons and relations among them. Here we are not talking about the friendship as it is not a family relation.

We are only talking about the family relations like son-father, brother-brother, cousincousin, brother-sister, etc. Can we deduce some more relations from these given relations? Make a two dimensional array, write the names of persons as the columns and rows headings. Suppose Haris is the name of first column and first row.

Saad is the name of 2nd col and 2nd row. Similarly Ahmed, Omar, Asim and Qasim are in the3rd, 4th, 5th, and 6th columns and rows respectively. The names are arranged in the same way in columns and rows.

algorithm building process : Initially we have $Si = \{i\}$ for $i = 1$ through n. We can give numbers to persons like person1, person2 etc
Secondly, the name of the set returned by *find* is fairly arbitrary. All that really matters is that $find(x) = find(y)$ if and only if x and y are in the same set.

Q. In the Huffman encoding tree every one uses his own sequence to build the tree and the compressed code is sent to save data transmitting time. In order to give understading to receiving side we have to send the tree strucrure also, this is another wastage of time for the transmitting of message. sir, if we have also to send tree structure then this will consume more time. then what is the benifit of huffman encoding in term of reducing transmittin time as total transmitting time of a phrase is equal to = time to trasmitt encoded text + time to transmitt huffman encoded tree ,

which may be greater then the origional time spent by phrase without using huffman encoding?????

Ans

Our focus here is on the concept of Huffman encoding, means Huffman encoding is a technique (Data structure /algorithm) which can reduce the size of data.

This will not consume extra time, for example, as routers communicate its routing table data after every few seconds, this type of information is sending in the headers of actual data (packet),

Q. is every tree rooted three, if not please explain.

Ans
Every tree has a root node, there is no tree without root node, so we can say every tree is rooted tree.

Q. Is left rotation anti -clockwise and right rotation clockwise?
Ans

Do not confuse yourself by including clockwise and anti-clockwise in AVL tree rotations.

Left-to-Right rotation means when we rotate node from left to right, and Right-to-Left rotation means when we rotate node from right to left.

Q. help me in understanding procedure of recursive functions as they are so much confusing. The code with recursive calls make it impossible to understand and take a plenty of time. Also, please elaborate the code for expression trees where inorder function is called recursively, i mean functionality. Please help me in the same manner as you did for the understanding of building AVL tree(if you remember).

Ans

About recursive function, you need to know two things,

1. Recursive function, is a function which calls itself,
2. There must be some recursive condition (means a condition which terminate recursion).

When we say recursion is a function which calls itself, so it comes in mind that it will be like an infinite loop. And in fact this is the case. So we know that an infinite loop can be terminated by putting a valid condition in the loop body. And the same is done with recursive function.

In short, Recursion is a function which calls itself and having some valid condition.

Lec 35:

it is not necessary that the caller should send the roots to union. It is necessary that the union function should be such that if a caller sends elements of two sets, it should find the sets of those elements, merge them and return the name of new set.

we have to keep (and find) pointer to the parent of a node unlike a binary tree in which we keep pointer to child nodes. In the array, we will set the parent of root to –1. We can write it as
Parent[i] = -1 // if i is the root

Parent Array
Initialization :

```
for ( i = 0; i < n ; i ++)
Parent [i] = -1 ;
for(j=i; parent[j] >= 0; j=parent[j])
;
return j;
```

i is an argument passed to the find function. The execution of the loop starts from the value, passed to the find function. We assign this value to a variable *j* andcheck whether its parent is greater than zero. It means that it is not –1. If it is greater than zero, its parent exists, Thus this loop continues till we find parent of *j* (parent [*j*]) less than zero (i.e. -1).

Union (i, j)

Now let's see the code for the function of union. Here we pass two elements to the function union. The union finds the roots of i and j. If *i* and *j* are disjoint sets, it will merge them.

```
root_i = find(i); root_j = find(j);
if (root_i != root_j) parent[root_j] = root_i;
```

union is clearly a constant time operation
Running time of *find*(*i*) is proportional to the height of the tree containing node *i*.
This can be proportional to n in the worst case (but not always)
Goal: Modify *union* to ensure that heights stay small

Q. it is stated that after studying lecture No. 32 till now, one point could yet not be cleared i.e:-

when you declare a Node or Tree Class
{
Here how you can define Node* p? While this class is under construction.
}
I mean how a pointer to that class is created which is currently under processed.
a very technical point in my view so please elaborate it

Ans
This is programming technique to define a pointer or an object in the same class. The compiler allows it.

Q. ust want to confirm if the ASCII Values are decimal based so that we could remember?

Ans
ASCII codes are decimal based.

> Up to encoding Huffman Scheme, we reduce the size of the data, but to transmit that data we have to transmit the tree also. As the receiver cannot understand our code unless we do not tell him. i.e. to detect "e" from the string "Traversing Threaded Binary Tree" where the "e" occurs. So sir to tell the receiver where the "e" is don't we send the "e" 2 times?.
> Regards
> Ans
> If the receiver is using the same data structure, then the stream of binary digits sent to the receiver will be encoded in the right sequence. You just have to send the tree information once to the receiver and then the receiver can easily decode the sent stream of bits.

Q. The worst case ?
Ans
The worst case running time of build heap method is **Nlog(N)** for N item (keys). So when Nlog(N) is the worst case then this method will be better than Nlog(N) on average basis.

Lec 36:
Running Time Analysis :
Goal: Modify *union* to ensure that heights stay small
union is clearly a constant-time operation.
Union by Size
Following are the salient characteristics of this method:

− Maintain sizes (number of nodes) of all trees, and during *union.*
− Make smaller tree, the subtree of the larger one.
− Implementation: for each root node i, instead of setting parent[i] to -1, set it to -k if tree rooted at i has k nodes.
− This is also called *union-by-weight*.

− If unions are done by weight (size), the depth of any element is never greater than log2*n*.
Intuitive Proof:
− Initially, every element is at depth zero.
− When its depth increases as a result of a union operation (it's in the smaller tree), it is placed in a tree that becomes at least twice as large as before (union of two equal size trees).
− How often can each union be carried out? -- log2*n* times, because after at most log2*n* unions, the tree will contain all *n* elements.
Union by Height
− Alternative to *union-by-size* strategy: maintain heights,
− During *union*, make a tree with smaller height a subtree of the other.
− Details are left as an exercise.

Q. What is the difference b/w Inorder Traversal and Inorder Successor?

Ans

In **inorder traversal** visits the left node first, then root node and then right node i.e left, root and right.
Inorder successor of a node is the next node in Inorder traversal of the Binary Tree.

Note: Only lecture relevant queries will be entertained, please update your self according to the vu lecture schedule.

Q. Explain the difference b/w Null Pointer and Null Nodes?

Ans

An empty node in a binary tree is called null node and each node in a binary having two pointers left and right if a node has empty node(s) then their pointer(s) will be called null pointer. i.e each leaf node has two null pointers.

Q. Plz explain Threaded Binary Tree?

Ans

Threaded Binary Tree is a binary tree in which every node that does not have a right child has a THREAD (in actual sense, a link) to its INORDER successor, and a binary tree in which every node that does not have a left child has a thread (in actual sense, a link) to its INORDER predecessor.

Q. What is difference among Binary Tree, Complete Binary Tree and Binary Search Tree?

Ans

A **binary tree** is a tree data structure in which each node has at most two child nodes, usually distinguished as "left" and "right". Nodes with children are parent nodes, and child nodes may contain references to their parents.

Complete Binary Tree: A complete binary is a binary tree in which all non-leaf nodes have two children and all leaf nodes are found at tree depth level. **OR**

A binary tree which is completely filled except possibly for bottom level, which filled from left to right.

A **Binary search tree** is a special type of binary tree with added property that for each node, the left child is less than parent is less than or equal to right child.

Lec 37:

Image Segmentation : In image segmentation, we will divide the image into different parts. An image may be segmented with regard to the intensity of the pixels. We may have groups of pixels having high intensity and those with pixels of low intensity. These pixels are divided on the basis of their threshold value.

How can we generate maze? The algorithm is as follows:
• Randomly remove walls until the entrance and exit cells are in the same set.
• Removal of the wall is the same as doing a union operation.
• Do not remove a randomly chosen wall if the cells it separates are already in the same set.

Pseudo Code of the Maze Generation :

```
MakeMaze(int size) {
```

```
entrance = 0; exit = size-1;
while (find(entrance) != find(exit)) {
cell1 = randomly chosen cell
cell2 = randomly chosen adjacent cell
if (find(cell1) != find(cell2) {
knock down wall between cells
union(cell1, cell2)
}
}
}
```

Q. In this para (computer revolutionized the world($)it is doller sign or s?

Ans

It is a new line character.

Q. What is mean by octet? how they are converted?

Ans

An **octet** is a unit of digital information in computing and telecommunications that consists of eight bits.

Q. differences between binary tree, binary search tree, AVL tree?

Ans

A **binary tree** is a tree data structure in which each node has at most two child nodes, usually distinguished as "left" and "right". Nodes with children are parent nodes, and child nodes may contain references to their parents.

A **Binary search tree** is a special type of binary tree with added property that for each node, the left child is less than parent is less than or equal to right child.

An **AVL tree** is a self-balancing binary search tree. In an AVL tree, the heights of the two child sub trees of any node differ by at most one.

Lec 38:

Tables and Dictionaries : a collection of rows and columns of information.

From rows and columns, we can understand that this is like a two dimensional array.
But it is not always a two dimensional array of same data type.

A table consists of several columns, known as fields. The row of a table is called a record.
Dictionary : for a programmer, the dictionary is a data type. This
data is in the form of sets of fields. These fields comprise data items.
To find an *entry* in the table, we only need to know the contents of one of the fields
(not all of them). This field is called *key*.

Implementation of Table

- How often entries are inserted, found and removed?
- How many of the possible key values are likely to be used?
- What is the likely pattern of searching for keys? Will most of the accesses be to just one or two key values?
- Is the table small enough to fit into the memory?
- How long will the table exist?

Q. what is disjoint ADT and what is the purpose of ADT?

Ans

ADT:
In programming, a data set defined by the programmer in terms of the information it can contain and the operations that can be performed with it. An Abstract Data Type (ADT) is more generalized than a data type constrained by the properties of the objects it contains. The standard example used in illustrating an abstract data type is the stack, a small portion of memory used to store information, generally on a temporary basis. As an abstract data type, the stack is simply a structure onto which values can be pushed (added) and from which they can be popped (removed). The type of value, such as integer, is irrelevant to the definition. The way in which the program performs operations on abstract data types is encapsulated, or hidden, from the rest of the program. Encapsulation enables the programmer to change the definition of the data type or its operations without introducing errors to the existing code that uses the abstract data type. Abstract data types represent an intermediate step between traditional programming and object-oriented programming.

Q. in the huffman encoding example given in the lecture handouts that "traversing threaded binary trees ". when we completed the tree with root node 33.

either we perform this final tree in any order.I mean we joined the nodes with frequencies 2 and get node 4 and then made node 9 by joining nodes of frequency 5 and node 4. this joining of nodes may occure in any order.

Ans

In huffman encoding, select the nodes with minimum frequency and then join them. So, in the discussed example, first of all the nodes with frequency 1 will be joined to get node 2, then nodes with frequency 2 will be joined and so on.

Yes, you can join the nodes in any order. For example, if there are four nodes with frequency 1, then you can join them in any order.

Q. difffrentiate the equilence and disjoint stets

Ans

Equivalence is a relation instead of set.

Equivalence relation:

An E**quivalence relation** is a relation that partitions a set so that every element of the set is a member of only one cell (part) of the partition and two element of the set are said to be equivalent with respect to the equivalence relation if they are the elements of the same cell (part).

Disjoint sets:

Two sets are said to be disjoint sets if they have no element in common, or in other word if the intersection of two sets is an empty set then they are called disjoint sets.

Q. WHAT WE MEAN STRING DATA TYPE OR WHAT IS ACTUAL CONCEPT OF STRING. WHAT IS ADVANTAGE OF DATA STRUCTURE.

Ans

A data structure is a way of grouping fundamental types (like integers, floating point numbers, and arrays) into a bundle that represents some identifiable thing.

For example, a matrix may be thought of as the bundle of the number of rows and columns, and the array of values of the elements of the matrix. This information must be known in order to manipulate the matrix.
C introduced the *struct* for declaring and manipulating data structures. C++ extended the *struct* to a *class*.

Q. string ?
Ans
Strings are lists of characters; they represent text in a program. Strings are used extensively in all kinds of programs. They label output, they are read from a file and sent to an output stream, or they can be the data that a program processes. Many languages have strings as a built-in data type. The C++ standard library provides a string class, whose declarations are available in the header file <string>. The operations given in the string class include concatenation of two strings using the + operator, searching a string for a substring, determining the number of characters in a string, and some input/output operations.

Q. What is Dymnamic Equivalance Problem?

Ans

Given an equivalence relation ~, the natural problem is to decide, for any a and b, if a ~ b. If the relation is stored as a two-dimensional array of booleans, then, of course, this can be done in constant time. The problem is that the relation is usually not explicitly, but rather implicitly, defined.

Lec 39:
Searching an Array: Binary Search (Sorted):
− Binary search is like looking up a phone number or a word in the dictionary
− Start in middle of book
− If the name you're looking for, comes before names on the page, search in the first half
− Otherwise, look into the second half
if (value == middle element)
value is found
else if (value < middle element)
search left half of list with the same method
else
search right half of list with the same method
Binary Search – Efficiency :
consider when we divide the

array of N items into two halves first time.
After 1 bisection N/2 items
After 2 bisections N/4 = N/22 items
. . .
After i bisections N/2i =1 item

_____

i = log2 N

First half contains N/2 items while the second half also contains around N/2 items.
After one of the halves is divided further, each half contains around N/4 elements.

Skip List :

Can do better than n comparisons to find
element in chain of length *n.*
The *head* and *tail* are special nodes at start and end of the list respectively.
− *Skip list contains a hierarchy of chains*

> *Q.* in handout page 393 the line states "....for any sequence of at most m finds and up to n-1 unions will require time proportional to (m + n). ..." Sir, please explain what is (n) and (m) in the above line. Good Day!!!!!!!
>
> Ans
>
> This is defined as the Dynamic equivalence problem and its algorithm building process. All elements are numbered sequentially from 1 to n. m is that particular element.

Q. it is necessary that when we delete a min no from min heap, the array wil be free at the end.

forexample if we have array of 15 and we delete a min no then the free memory is at 15th .

is it possible that the array is freed at 12 or any other possition instead of 15.

thanksssssss

Ans

Array position will be free from where the element is deleted.

> *Q.* I cannot completely understand the concept of fractional knapsack . please give me some explanation

Ans

The knapsack problem or rucksack problem is a problem in combinatorial optimization: Given a set of items, each with a weight and a value, determine the number of each item to include in a collection so that the total weight is less than a given limit and the total value is as large as possible. It derives its name from the problem faced by someone who is constrained by a fixed-size knapsack and must fill it with the most useful items.

The problem often arises in resource allocation with financial constraints. A similar problem also appears in combinatorics, complexity theory, cryptography and applied mathematics.

Q. what does mean by Dynamic Equivalence?

Ans

A problem in which an equivalence relation is deinfed over every pair of elements in a set S.

Q. Please describe the proper definition of disjointSets?

Ans

In computer science, two sets are said to be disjoint if they have no element in common. For example, {1, 2, 3} and {4, 5, 6} are disjoint sets.

Lec 40 :

Skip List – Formally : − A skip list for a set ***S*** of distinct (key, element) items is a series of lists ***S***0, ***S***1 , …
, ***Sh*** such that
o Each list ***Si*** contains the special keys +∞ and −∞
o List ***S***0 contains the keys of ***S*** in non-decreasing order Each list is a subsequence of the previous one, i.e.,
S0 ⊇ ***S***1 ⊇ … ⊇ ***Sh***
o List ***Sh*** contains only the two special keys

Skip List Search :

We search for a key *x* in the following fashion:

• We start at the first position of the top list
• At the current position *p*, we compare *x* with $y \leftarrow key(after(p))$
• $x = y$: we return $element(after(p))$
$x > y$: we "scan forward"
• $x < y$: we "drop down"
• If we try to drop down past the bottom list, we return *NO_SUCH_KEY*

Insertion in Skip List :

The first step of the algorithm is that
• We repeatedly toss a coin until we get tails, and we denote with ***i*** the number of times the coin came up heads.
• If ***i*** > ***h***, we add to the skip list new lists ***Sh***+1, … , ***Si*** +1, each containing only the two special keys
• We search for ***x*** in the skip list and find the positions ***p***0, ***p***1 , …, ***pi*** of the items with largest key less than ***x*** in each list ***S***0, ***S***1, … , ***Si***
• For ***j*** ← 0, …, ***i***, we insert item (***x***, ***o***) into list ***Sj*** after position ***pj***

Deletion from Skip List : To remove an item with key ***x*** from a skip list, we proceed as follows: We search for ***x*** in the skip list and find the positions ***p***0, ***p***1 , …, ***pi*** of the items with key ***x***, where position ***pj*** is in list ***Sj*.** This means that we look for the links of the item to be removed . We know that an item in the list has necessarily link in the list S0. Moreover it may have links in other lists up to Si or say Sj. After this, we remove positions ***p***0, ***p***1 , …, ***pi*** from the lists ***S***0, ***S***1, … , ***Si*.** We remove all the lists except the list containing only the two special keys.

Q. Can there two elements equal in heap??? For example we have two nodes of value 14, then is it possible to have both in heap???

Ans

It is up to you, if you want to allow then two nodes with same values can occur in the heap.

The definition of the heap also allows us, for example, "Heap (Minimum heap) is a data structure in which each node is **greater than or equal to** its parent"

This "greater than or equal to" tells us that two nodes with same values can occur in the heap.

Q. How many leaf and non-leaf nodes are present in a complete binary tree if its depth is 7 ?

Ans

The total number of nodes in a complete binary tree having dept "D" can be calculated by the formula $\mathbf{2^{D+1} - 1}$ (When level number start form 0) or $\mathbf{2^{D} - 1}$ (When level number start form 1).

Further, the complete binary tree contains **2N -1** nodes, where the leaf nodes are **N** while none leaf nodes are **N – 1**.

Q. is there no null character at the end of the phrase. because we have learned that a null should be at the end of the prase.

Ans

In the given phrase, there is no NULL character.
The total number of characters are mentioned in the assignment question.

Q. What are Union Operations?

Ans

n Dynamic Equivalence problem, there are two permissible operations on sets. The first is find, which returns the name of the set (that is, the equivalence class) containing a given element. The second operation adds relations. If we want to add the relation a ~ b, then we first see if a and b are already related. This is done by performing finds on both a and b and checking whether they are in the same equivalence class. If they are not, then we apply union. This operation merges the two equivalence classes containing a and b into a new equivalence class.

Lec 41: A quad-node stores:
• item
• link to the node before
• link to the node after
• link to the node below
• link to the node above

Hashing
The hashing is an algorithmic procedure and a methodology. It is not a new data structure. It is a way to use the existing data structure. The methods- find, insert and remove of table will get of constant time. You will see that we will be able to do this in a single step.

Examples of Hashing
Let's see some examples of hashing and hash functions. With the help of these examples you will easily understand the working of find, insert and remove methods.
Suppose we want to store some data. We have a list of some fruits. The names of fruits are in string. The key is the name of the fruit. We will pass it to the hash function to get the hash key.
Suppose our hash function gave us the following values:
HashCode ("apple") = 5
hashCode ("watermelon") = 3
hashCode ("grapes") = 8
hashCode ("cantaloupe") = 7
hashCode ("kiwi") = 0
hashCode ("strawberry") = 9
hashCode ("mango") = 6
hashCode ("banana") = 2

Q. explain the getmin method and build heap in linear time? thank you

Ans

1- getmin method gives us the minimum value in the heap, which is not deleted from heap.

```
if (! isEmpty())
return array[1];
```

2- buidHeap is a linear time algorithm and is batter than N log2N algorithm which is not of linear nature. We ca n draw it in straight line with some slope.

Q. What is the difference between pointer and array where we use pointer and where we use array. For complete binary tree can we use pointers for storing data thanks
Ans
An array is a single, preallocated chunk of contiguous elements (all of the same type), fixed in size and location. A pointer is a reference to any data element (of a particular type) anywhere. A pointer must be assigned to point to space allocated elsewhere, but it can be reassigned at any time. A pointer can point to an array, and can simulate a dynamically allocated array. You can use array or pointer depending on the situation

Q. What is the difference between pointer and array where we use pointer and where we use array. For complete binary tree can we use pointers for storing data

Ans
An array is a single, preallocated chunk of contiguous elements (all of the same type), fixed in size and location. A pointer is a reference to any data element (of a particular type) anywhere. A pointer must be assigned to point to space allocated elsewhere, but it can be reassigned at any time. A pointer can point to an array, and can simulate a dynamically allocated array. You can use array or pointer depending on the situation.
Q. What is difference betweeen root and root node of the tree?

Ans
The top node of the tree is clled the root node. Root node is also called as Root.

Lec 42:

Collision
Collision takes place when two or more keys (data items) produce the same index.
Let's see the previous example of storing the names of fruits.

hash("watermelon") = 3
hash("grapes") = 8
hash("cantaloupe") = 7
hash("kiwi") = 0
hash("strawberry") = 9
hash("mango") = 6
hash("banana") = 2
We store these data items in the respective index in the array. In the above example, the index given by the hash function does not collide with any other entry. Suppose we want to add another fruit "honeydew" in it. When "honeydew" is passed to the hash function, we get the value 6 i.e.
hash("honeydew") = 6

Q. Up to encoding Huffman Scheme, we reduce the size of the data, but to transmit that data we have to transmit the tree also. As the receiver cannot understand our code unless we do not tell him. i.e. to detect "e" from the string "Traversing Threaded Binary Tree" where the "e" occurs. So sir to tell the receiver where the "e" is don't we send the "e" 2 times?.

Ans
If the receiver is using the same data structure, then the stream of binary digits sent to the receiver will be encoded in the right sequence. You just have to send the tree information once to the receiver and then the receiver can easily decode the sent stream of bits.

Q. talking about queue data structure, do we have simple queue data structure i.e. we do not impose any priority stander on the data.

Ans
talking about queue data structure, do we have simple queue data structure i.e. we do not impose any priority stander on the data.

Q. I could not understand to count the new line characher in lecture no. 26 .Please describe it in detail

Ans
New line character is occured at the end of the character stream (sentence). If the character string consists of a single line sentence, then the number of new line characters will be 1. If character string consists of more than one sentence, then the new line characters will be according to number of lines.

Lec 43: Applications of Hashing

Compilers use hash tables to keep track of declared variables (symbol table).

important part of compilation process. Compiler puts variables inside symbol table during this process. Compiler has to keep track of different attributes of variables. The name of the variable, its type, scope and function name where it is declared etc is put into the symbol table. If you consider the operations on symbol table, those can be insertion of variable information, search of a variable information or deletion of a variable. Two of these insert and find are mostly used operations. You might have understood already that a variable name will be parameter (or the key) to these operations. But there is one slight problem that if you named a variable as x outside of a code block and inside that code block, you declared another variable of the similar type and name then only name cannot be the key and scope is the only differentiating factor. Supposing that all the variables inside the program have unique names, variable name can be used as the key. Compiler will insert the variable information by calling the insert function and by passing in the variable information. It retrieves the variable value by passing in the variable name. Well, this exercise is related to your Compiler Construction course where you will construct you own language compiler.

Q. plz explain the question i am not understand the question

(b) Below is given Huffman encoded form of a message according to the above Huffman encoding tree. Decode it into original Text. [Marks 8]

1000100001101011010011111010100001110101001001000100010011010011110

0000100001011101010100110011110000110011100001000100011111110101100

1100101110001011011110100100001011101010100111001101O

sir hum logo ne is binary code ko oper deya gaye tree se macth karna ha yah k jo bnaya ha os se karna ha

Ans
First of all you will assign binary digits to all the left and right nodes of the tree given in the assignment. Then you will start decoding the given binary code into the text by following the tree with binary digits.

Q. In Fig 27.3, before starting insertion of node 16, there are only one thread i.e. predecessor for all nodes 14, 15, 18 and 20. But after insertion of node 16, there are both threads for it i.e. predecessor and successor. Similar case in fig 27.12 for nodes 5, 16 etc. I could not understand it kindly explain.

Ans

In the Fig 27.3 the nodes 14,15, 18 are not leaf node, therefore they have only thread to inorder predecessor and have links towards their inorder successors.

While the new inserted node 16 is a leaf node, therefore it has threads towards predecessor and successor.

Note: node 20 is leaf node but it has only thread towards inorder predecessor, because it is the right most node, means it has no inorder successor (the same is true for left most node 3 in Fig 27.10).

> **Q.** At page 400, there is a representation of the array, which holds the tree. We have initialize **all the** array index values to -1, so that we should know this will be a parent in the future. Beneath The blocks (-1's) we have 1 to 8 numbers. Do these numbers indicate some Node values or these are the Index addresses marked as Numbers i.e some thing like 1 for 0x1234.
> If NO then do we store 2 arrays job in one FAT array?
>
> ans

The number from 1 to 8 under this array are the indexes of the array, however, the values (nodes) that are to be stored in this array, by chance, are also 1 to 8 mentioned above this array.

Q. in last topic of lecture No. 38

binary search,
there is written that log 1000000 will be 20, how it comes 20?
it comes 6 not 20.

ans

The log2 (1000000) is 19.93 which is about 20.

You can calculate it by yourself through calculator.

Q. In swap we have to do the operations at least 3 times. Which i can understand easily, what is the idea of doing the same thing in percolateDown method. I mean the hole (i or n) moves down and the same thing (swap) is achieved. Please guide me a little what really we are doing in percolateDown method. Or at least tell me what is the idea behind percolateDown().

ans

The swap function or operation may useful only, if the value that is needed to become up is below only one level which is not always the case.

You are required to look the code of **percolateDown()** method in handouts, this method can bring up a values up even if it more than one levels down and this is the main idea behind the percolateDown method.

Lec 44:
Selection Sort
Suppose we have an array with different numbers. For sorting it in an ascending order, selection sorting will be applied on it in the following manner. We find the smallest number in the array and bring it to the position 1. We do the same process with the remaining part of the array and bring the smallest number among the remaining array to the next position. This process continues till the time all the positions of the array are filled with the elements. Thus the main idea of selection sort is that
• find the smallest element

• put it in the first position
• find the next smallest element in the remaining elements
• put it in the second position
• …
• And so on, until we get to the end of the array

Q. if there is any possibility to start array from zero in min heap, if yes than where the left child will reside, as the right child could be located at 1 ?

ans

If we start from array zero location, then the formulas for left child, right child or parent of any node i.e 2i , 2i +1 and i/2 are locating elements in the array incorrectly. So we cannot start locating heap elements from zero location in the array.

Q. the location of root is 1, But 1/2 = 0 by floor, naheen?

ans

The formula floor (i/2) locates the parent of a node in the heap, while the root node has no parent.

Further, floor (1 / 2) = 0.

Q. Consider a min heap, represented by the following array: 11,22,33,44,55.

After inserting a node with value 66.What is the updated min heap. sir i can not understand min heap. Please sir tell me about this with a example
ans

After inserting the node 66 in this min heap, the value inserted in the array will be at location 6, by using the following formulas for heap.

2i // locate the left child of node i

2i + 1 // locate the right child of node i

Floor (i/2) // locate the parent of node i

Lec 45: Divide and Conquer

In the previous lecture, we had started discussing three new sorting algorithms; merge sort, quick sort and heap sort. All of these three algorithms take time proportional to nlog2n. Our elementary three sorting algorithms were taking n2 time; therefore, these new algorithms with nlog2n time are faster. In search operation, we were trying to reduce the time from n to log2n.

Let's discuss these sorting algorithms; merge sort, quick sort and heap sort in detail. We had started our discussion from divide and conquer rule where we also saw an example. Instead of sorting a whole array, we will divide it in two parts, each part is sorted separately and then they are merged into a single array.

10 12 8 4 2 11 7 5

S p lit th e lis t in to tw o p a rts

4 8 10 12 2 5 7 11

Q. whta is the main diff bet binary ans complete binary tree?

ans

A **binary tree** is a data structure in which each node has zero, one or two child nodes while **Complete Binary Tree** is a binary tree in which every level except possibly the deepest (Last level) is completely filled (means each node has two child nodes).

Q. clear about the hashing?

ans

Hashing (or hash function) is a procedure which usually transforms large size string (characters) data of variable length into smaller data of fixed length.

Q. Define Linear Probing briefly?

ans

Linear Probing is a rehashing technique, where the rehashing is done by looking for the next empty space that it can occupy to solve the collision problem. This method finding the second or third or so on location for the information, there for it is called linear