Question 1 : How to find middle element of linked list in one pass?
One of the most popular question from data structures and algorithm, mostly asked on telephonic interview. Since many programmer know that, in order to find length of linked list we need to first traverse through linkedlist till we find last node, which is pointing to null, and then in second pass we can find middle element by traversing only half of length. They get confused when interviewer ask him to do same job in one pass. In order to find middle element of linked list in one pass you need to maintain two pointer, one increment at each node while other increments after two nodes at a time, by having this arrangement, when first pointer reaches end, second pointer will point to middle element of linked list. See this trick to find middle element of linked list in single pass for more details.

Question 2 : How to find if linked list has loop ?
This question has bit of similarity with earlier algorithm and data structure interview question. I mean we can use two pointer approach to solve this problem. If we maintain two pointers, and we increment one pointer after processing two nodes and other after processing every node, we are likely to find a situation where both the pointers will be pointing to same node. This will only happen if linked list has loop.

Question 3 : How to find 3rd element from end in a linked list in one pass?
This is another frequently asked linked list interview question. This question is exactly similar to finding middle element of linked list in single pass. If we apply same trick of maintaining two pointers and increment other pointer, when first has moved upto 3rd element, than when first pointer reaches to the end of linked list, second pointer will be pointing to the 3rd element from last in a linked list.

Question 4 : In an integer array, there is 1 to 100 number, out of one is duplicate, how to find ?
This is a rather simple data structures question, especially for this kind of. In this case you can simply add all numbers stored in array, and total sum should be equal to n(n+1)/2. Now just subtract actual sum to expected sum, and that is your duplicate number. Of course there is a brute force way of checking each number against all other numbers, but that will result in performance of O(n^2) which is not good. By the way this trick will not work if array have multiple duplicates or its not numbers forming arithmetic progression. Here is example of one way to find duplicate number in array.

Question 6 : How to reverse String in Java ?
This is one of my favorite question. Since String is one of the most important type in programming, you expect lot of question related to String any data structure interview. There are many ways to reverse Sting in Java or any other programming language, and interviewer will force you to solve this problem by using without API i.e. without using reverse() method of StringBuffer. In follow-up he may ask to reverse String using recursion as well. See 3 ways to reverse String in Java to learn reversing String using both loops and recursion in Java.

Question 7 : Write a Java program to sort a array using Bubble Sort algorithm?
I have always send couple of questions from searching and sorting in data structure interviews. Bubble sort is one of the simplest sorting algorithm but if you ask anyone to implement on the spot it gives you an opportunity to gauge programming skills of a candidate. See How to sort array using Bubble Sort in Java for complete solution of this datastrucutre interview question.

Question 8 : What is difference between Stack and Queue data structure ?
One of the classical datastrucutre interview question. I guess every one know, No? Any way main difference is that Stack is LIFO(Last In First Out) data structure while Queue is a FIFO(First In First Out) data structure.

Question 9 : How do you find duplicates in array if there is more than one duplicate?
Sometime this is asked as follow-up question of earlier datastrucutre interview question, related to finding duplicates in Array. One way of solving this problem is using a Hashtable or HashMap data structure. You can traverse through array, and store each number as key and number of occurrence as value. At the end of traversal you can find all duplicate numbers, for which occurrence is more than one. In Java if a number already exists in HashMap then calling get(index) will return number otherwise it return null. this property can be used to insert or update numbers in HashMap.

Question 10 : What is difference between Singly Linked List and Doubly Linked List data structure?
This is another classical interview question on data structure, mostly asked on telephonic rounds. Main difference between singly linked list and doubly linked list is ability to traverse. In a single linked list, node only points towards next node, and there is no pointer to previous node, which means you can not traverse back on a singly linked list. On the other hand doubly linked list maintains two pointers, towards next and previous node, which allows you to navigate in both direction in any linked list.

Question 11 : Write Java program to print Fibonacci series ?
This is not a data structures question, but a programming one, which many times appear during data structure interview. Fibonacci series is a mathematical series, where each number is sum of previous two numbers e.g. 1,1, 2, 3, 5, 8, 13, 21. Interviewer is often interested in two things, a function which returns nth number in Fibonacci series and solving this problem using recursion in Java. Though, its easy question, recursion part often confuses beginners. See this link to find nth Fibonacci number in Java.

Question 12 : Write Java program to check if a number is palindrome or not?
This is similar to previous question, not directly related to data structures, but quite popular along with other questions. A number is called palindrome, if reverse of

number is equal to number itself. Interviewer ask to solve this problem without taking help from Java API or any open source library. Any way it's simple question, you can use division operator (/) and remainder operator (%) to solve this question. Just remember, division operator can be used to get rid of last digit e.g. 1234/10 will give you 123, and modulus operator can give you last digit e.g. 1234%10 will return 4. By the way, here is a Java program check if number is palindrome or not.

Question 13 : What is binary search tree?
This is a data structure question from Tree data structures. Binary Search Tree has some special properties e.g. left nodes contains items whose value is less than root , right sub tree contains keys with higher node value than root, and there should not be any duplicates in the tree. Apart from definition, interview can ask you to implement binary search tree in Java and questions on tree traversal e.g. IN order, preorder, and post order traversals are quite popular data structure question.

Question 14 : How to reverse linked list using recursion and iteration?
This is another good question on data structures. There are many algorithm to reverse linked list and you can search of them using google. I am thinking of writing another blog post to explain linked list reversal and will share with you later.

Question 15 : Write a Java program to implement Stack in Java?
You can implement Stack by using array or linked list. This question expect you to implement standard method provided by stack data structure e.g. push() and pop(). Both push() and pop() should be happen at top of stack, which you need to keep track. It's also good if you can implement utility methods like contains(), isEmpty() etc. By the way JDK has java.util.Stack class and you can check it's code to get an idea. You can also check Effective Java book, where Josh Bloch has explains how an incorrect implementation of stack can cause memory leak in Java.

Find nth to last node in a linked list using iteration

we just use 2 pointers – call them pntr1 and pntr2. Let's say that we advance pntr2 by N-1 nodes. After that, every time we iterate through the linked list, **both** pntr1 and pntr2 will be advanced to point to the very next node in the linked list. This means that there will always be a distance of exactly n-1 nodes between pntr1 and pntr2. So, when pntr2 is equal to null, this means that pntr2 has reached the very end of the list. And, most importantly, it means that pntr1 will be pointing at the nth to last node! So, we will have our answer!

What is hashing technique? Describe in brief.

In general, in all searching techniques, search time is dependent on the number of items. Sequential search, binary search and all the search trees are totally dependent on number of items and many

key comparisons are involved.

Hashing is a technique where search time is independent of the number of items or elements. In this technique a hash function is used to generate an address from a key. The hash function takes a key as input and returns the hash value of that key which is used as an address index in the array.

We can write hash function as follows

h(k)=a;

Where h is hash function, k is the key, a is the hash value of the key.

While choosing a hash function we should consider some important points.

- It should be easy to compute
- It should generate address with minimum collision.

What are different techniques for making hash function? Explain with example.

Techniques for making hash function.

- Truncation Method
- Midsquare Method
- Folding Method
- Division Method

Truncation Method

This is the simplest method for computing address from a key.In this method we take only a part of the key as address.

Example:

Let us take some 8 digit keys and find addresses for them. Let the table size is 100 and we have to take 2 rightmost digits for getting the hash table address. Suppose the keys are. 62394572,87135565,93457271,45393225.

So the address of above keys will be 72,65,71 and 25 respectively.

This method is easy to compute but chances of collision are more because last two digits can be same in more than one keys.

Midsquare Method

In this method the key is squared and some digits from the middle of this square are taken as address.

Example:

Suppose that table size is 1000 and keys are as follows

Key	1123	2273	3139	3045
Square of key	1261129	5166529	9853321	9272025
Address	612	665	533	720

Folding Method

In this technique the key is divided into different part where the length of each part is same as that of the required address, except possibly the last part.

Example:

Let key is 123945234 and the table size is 1000 then we will broke this key as follows

123945234 ----> 123 945 234

Now we will add these broken parts. 123+945+234=1302. The sum is 1302, we will ignore the final carry 1, so the address for the key 123945234 is 302.

Division Method (Modulo-Division)

In Modulo-Division method the key is divided by the table size and the remainder is taken as the address of the hash table.

Let the table size is n then

H (k) =k mod n

Example

Let the keys are 123, 945,234 and table size is 11 then the address of these keys will be.

123 % 11=2
945%11=10
235%11=4

So the hash address of above keys will be 2,10,4.

Note: - Collisions can be minimized if the table size is taken to be a prime number.

What are different methods of collision resolution in hashing.

A collision occurs whenever a key is mapped to an address that is already occupied. Collision Resolution technique provides an alternate place in hash table where this key can be placed. Collision Resolution technique can be classified as:

1) Open Addressing (Closed Hashing)

a) Linear Probing
b) Quadratic Probing
c) Double Hashing

2) Separate Chaining (Open Hashing)

Describe Linear Probing with an example.

In this method if address given by hash function is already occupied, then the key will be inserted in the next empty position in hash table. Let the table size is 7 and hash function returns address 4 for a key then we will search the empty location in this sequence.

4, 5, 6, 7, 0, 1, 2, 3

Example:

Let the keys are 28, 47, 20, 36, 43, 23, 25, 54 and table size is 11 then

h(28)=28%11=6

h(47)=47%11=3

h(20)=20%11=9

h(36)=36%11=3

h(43)=43%11=10

h(23)=23%11=1

h(25)=25%11=3

h(54)=54%11=10

0	
1	
2	
3	47
4	
5	
6	28
7	
8	
9	20
10	

insert 28,47,20

0	
1	
2	
3	47
4	36
5	
6	28
7	
8	
9	20
10	

insert 36

0	
1	23
2	
3	47
4	36
5	
6	28
7	
8	
9	20
10	43

insert 43,23

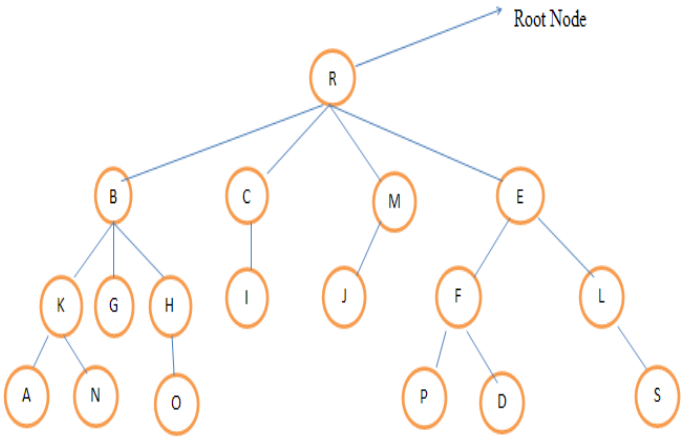
0	54
1	23
2	
3	47
4	36
5	25
6	28
7	
8	
9	20
10	43

insert 25,54

Describe the following term in a tree.

a) Level b) Height c) Degree.

Let's take an example to define the above term.



Level:

Level of any node is defined as the distance of that node from the root. The level of root node is always zero. Node B, C, M, E are at level 1.Nodes K,G,H,I,J,F,L are at level 2.Nodes A,N,O,P,D,S are at level 3.

Height:

Height is also known as depth of the tree. The height of root node is one. Height of a tree is equal to one more than the largest level number of tree. The height of the above tree is 4.

Degree:

The number of children of a node is called its degree. The degree of node R is 4, the degree of node B is 3.The degree of node S is 0.The degree of a tree is the maximum degree of the node of the tree. The degree of the above given tree is 4.

Describe binary tree and its property.

In a binary tree a node can have maximum two children, or in other words we can say a node can have 0,1, or 2 children.

Properties of binary tree.

1) The maximum number of nodes on any level i is 2^i where i>=0.

2) The maximum number of nodes possible in a binary tree of height h is 2^h-1.

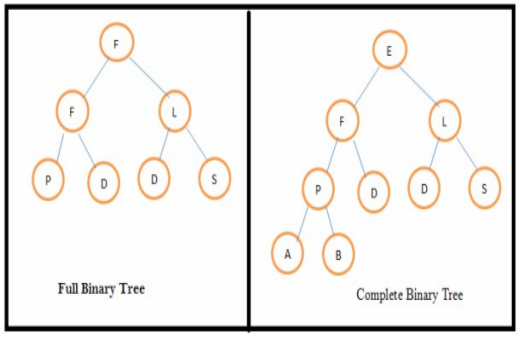
3) The minimum number of nodes possible in a binary tree of height h is equal to h.

4) If a binary tree contains n nodes then its maximum possible height is n and minimum height possible is $\log_2 (n+1)$.

5) If n is the total no of nodes and e is the total no of edges then e=n-1.The tree must be non-empty binary tree.

6) If n_0 is the number of nodes with no child and n_2 is the number of nodes with 2 children, then $n_0=n_2+1$.

Describe full binary tree and complete binary tree.



Full binary tree: A binary tree is full binary tree if all the levels have maximum number of nodes.

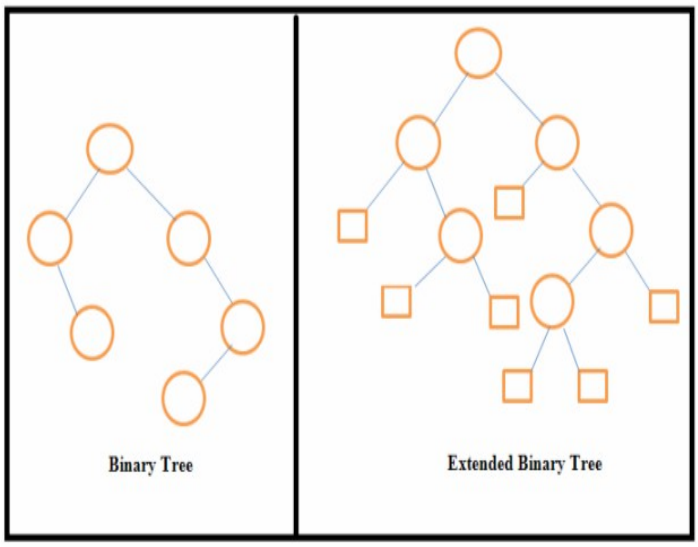
A full binary tree of height h has $(2^h – 1)$ nodes.

Complete binary tree:In a complete binary tree all the levels have maximum number of nodes except possibly the last level.

The minimum no of nodes in a complete binary tree is $2^{h-1}$and the maximum number of nodes possible is $(2^h – 1)$.Where h is the height of the tree.

Explain Extended Binary tree.

A binarytree can be converted to an extended binary tree by adding special nodes to leaf nodes and nodes that have only one child.Extended binary tree is also called 2-tree.



In the above figure external nodes are shown by squares and internal nodes by circles. The extended binary tree is a strictly binary tree means each node has either 0 or 2 children. The path length of any node is the number of edges traversed from that node to the root node. Internal path length of a binary tree is the sum of path lengths of all internal nodes and external path length of a binary tree is the sum of path lengths of all external nodes.

What are different dynamic memory allocation technique in C

•

The process of allocating memory at run time is called dynamic memory allocation. The allocation and release of this memory space can be done with the help of some predefined function. These functions allocate memory from a memory area called heap and free this memory whenever not required. The functions that are used to allocate memory at runtime are as follows:

- malloc()
- calloc()
- realloc()

1. malloc()

This function allocates memory dynamically. It is generally used as:

ptr= (datatype *) malloc(specified size);

Here ptr is a pointer of type datatype (can be int, float, double….) and specified size is the size in bytes. The expression (datatype *) is used to typecast the pointer returned by malloc().

Example:

```
int *ptr;
ptr=(int *)malloc(4*sizeof(int));
```

It allocates the memory space to hold four integer values and the address of first byte is stored in the pointer variable ptr. The allocated memory contains garbage value.

2. calloc()

The calloc() function is used to allocate multiple blocks of memory.

Example:

```
int *ptr;
ptr=(int *)calloc(4, sizeof(int));
```

It allocates 4 blocks of memory and each block contains 2 bytes.

3. realloc()

We can increase or decrease the memory allocated by malloc() or calloc() function. The realloc() function is used to change the size of the memory block. It changes the memory block without destroying the old data.

Example:

```
int *ptr;
ptr=(int *)malloc(4*sizeof(int));
ptr=(int *)realloc(ptr,newsize);
```

This function takes two argument, first is a pointer to the block of memory that was previously allocated by malloc() or calloc() and second argument is the new size of memory block.

```
ptr=(int *)realloc(ptr, 4*sizeof(int)); // newsize
```

Q.2 What are the difference between malloc() and calloc()?

Following are the main difference between malloc() and calloc().

- calloc() function takes two parameters but malloc() function takes only one parameter.

- Memory allocated by calloc() is initialized to zero while memory allocated by malloc() contains garbage value.

Q.3 How will you free the memory that is allocated at run time?

Memory is one of the most important resources and it is limited. The dynamically allocated memory is not automatically released; it will exist till the end of program. So it is programmer's responsibility to free the memory after completion. The free() function is used to release the memory that is allocated at run time.

Example:

```
free(ptr);
```

Here ptr is a pointer variable that contains the base address of a memory block created by malloc() or calloc() function.

Q.4 What are different application of stack.

Some of the applications of stack are as follows:

- Function calls.
- Reversal of a string.
- Checking validity of an expression containing nested parenthesis.
- Conversion of infix expression to postfix.

Q.5 How will you check the validity of an expression containing nested parentheses?

One of the applications of stack is checking validity of an expression containing nested parenthesis. An expression will be valid if it satisfies the two conditions.

- The total number of left parenthesis should be equal to the total number of right parenthesis in the

expression.

- For every right parenthesis there should be a left parenthesis of the same time.

The procedure for checking validity of an expression containing nested parenthesis:

1. First take an empty stack
2. Scan the symbols of expression from left to right.
3. If the symbol is a left parenthesis then push it on the stack.
4. If the symbol is right parenthesis then
    If the stack is empty then the expression is invalid because Right parentheses are more than left parenthesis.
    else
    Pop an element from stack.
    If popped parenthesis does not match the parenthesis being scanned then it is invalid because of mismatched parenthesis.
5. After scanning all the symbols of expression, if stack is empty then expression is valid else it is invalid because left parenthesis is more than right parenthesis.

Q.6 Give the example of validating the parenthesis of expression using stack.

Q.7 Describe AVL tree or height balanced binary search tree.

An AVL tree is binary search tree (BST) where the difference in the height of left and right subtrees of any node can be at most one. The technique for balancing the binary search tree was introduced by Russian Mathematicians G. M. Adelson and E. M. Landis in 1962. The height balanced binary search tree is called AVL tree in their honor.

Example:

For the leaf node 12, 40, 65 and 98 left and right subtrees are empty so difference of heights of their subtrees is zero.

For node 20 height of left subtree is 2 and height of right subtree is 3 so difference is 1.

For node 15 height of left subtree is 1 and height of right subtree is 0 so difference is 1.

For node 57 height of left subtree is 1 and height of right subtree is 2 so difference is 1.

For node 78 height of left subtree is 1 and height of right subtree is 1 so difference is 0.

Each node of an AVL tree has a balance factor, which is defined as the difference between the heights of left subtree and right subtree of a node.

Balance factor of a node=Height of its left subtree - Height of its right subtree.

In AVL tree possible values for the balance factor of any node are -1, 0, 1.

Q.8 Describe Tree Rotation in AVL tree.

After insertion or deletion operation the balance factor of the nodes in AVL tree can be changed and the tree may not be balanced. We can balance this tree by performing tree rotations. We know that AVL tree is binary search tree. Rotation of the tree should be in such a way that the new converted

tree maintain the binary search tree property with inorder traversal same as that of the original tree. There are two types of tree rotation

- Right Rotation
- Left Rotation

Fig: Right Rotation

Q.9 Give one example of Right Rotation.

Right Rotation about node 25.

Data structure interview questions - posted on June 27, 2013 at 15:55 PM by Kshipra Singh

1. What is Data Structure?

- Data structure is a group of data elements grouped together under one name.
- These data elements are called members. They can have different types and different lengths.
- Some of them store the data of same type while others store different types of data.

2. Which data structure is used to perform recursion?

- The data structure used for recursion is Stack.
- Its LIFO property helps it remembers its 'caller'. This helps it know the data which is to be returned when the function has to return.
- System stack is used for storing the return addresses of the function calls.

3. Does the Minimal Spanning tree of a graph give the shortest distance between any 2 specified nodes?

- No, it doesn’t.
- It assures that the total weight of the tree is kept to minimum.
- It doesn't imply that the distance between any two nodes involved in the minimum-spanning tree is minimum.

4. If you are using C language to implement the heterogeneous linked list, what pointer type will you use?

- A heterogeneous linked list contains different data types in its nodes. We can not use ordinary pointer to connect them.
- The pointer that we use in such a case is void pointer as it is a generic pointer type and capable of storing pointer to any type.

5. Differentiate between PUSH and POP?

- Pushing and popping refers to the way data is stored into and retrieved from a stack.
- PUSH – Data being pushed/ added to the stack.
- POP - Data being retrieved from the stack, particularly the topmost data.

6. When is a binary search algorithm best applied?

- It is best applied to search a list when the elements are already in order or sorted.
- The list here is searched starting in the middle. If that middle value is not the correct one, the lower or the upper half is searched in the similar way.

7. How do you reference all the elements in a one-dimension array?

- This is done using an indexed loop.
- The counter runs from 0 to the array size minus one.
- Using the loop counter as the array subscript helps in referencing all the elements in one-dimensional array.

8. What is Huffman's algorithm?

- It is used in creating extended binary trees that have minimum weighted path lengths from the given weights.
- It makes use of a table that contains frequency of occurrence for each data element.

9. What is Fibonacci search?

- It is a search algorithm that applies to a sorted array.
- It uses divide-and-conquer approach that reduces the time needed to reach the target element.

10. Which data structure is applied when dealing with a recursive function?

- A recursive function is a function that calls itself based on a terminating condition.
- It uses stack.
- Using LIFO, a call to a recursive function saves the return address. This tells the return address to the calling function after the call terminates.

11. How does dynamic memory allocation help in managing data?

- Dynamic memory allocation helps to store simple structured data types.
- It can combine separately allocated structured blocks to form composite structures that expand and contract as required.

12. What is a bubble sort and how do you perform it?

- Bubble sort is a sorting technique which can be applied to data structures like arrays.
- Here, the adjacent values are compared and their positions are exchanged if they are out of order.
- The smaller value bubbles up to the top of the list, while the larger value sinks to the bottom.

13. How does variable declaration affect memory allocation?

- The amount of memory to be allocated depends on the data type of the variable.

- An integer type variable is needs 32 bits of memory storage to be reserved.

14. You want to insert a new item in a binary search tree. How would you do it?

- Let us assume that the you want to insert is unique.
- First of all, check if the tree is empty.
- If it is empty, you can insert the new item in the root node.
- If it is not empty, refer to the new item's key.
- If the data to be entered is smaller than the root's key, insert it into the root's left subtree.
- Otherwise, insert it into the root's right subtree.

15. Why is the isEmpty() member method called?

- The isEmpty() member method is called during the dequeue process. It helps in ascertaining if there exists any item in the queue which needs to be removed.
- This method is called by the dequeue() method before returning the front element.

16. What is a queue ?

- A Queue refers to a sequential organization of data.
- It is a FIFO type data structure in which an element is always inserted at the last position and any element is always removed from the first position.

17. What is adequeue?

- A dequeue is a double-ended queue.
- The elements here can be inserted or removed from either end.

18. What is a postfix expression?

- It is an expression in which each operator follows its operands.
- Here, there is no need to group sub-expressions in parentheses or to consider operator precedence..