Round 1

Which of the following is false about a binary search tree?

1. The left child is always lesser than its parent
2. The right child is always greater than its parent
3. The left and right sub-trees should also be binary search trees
4. In order sequence gives decreasing order of elements

In order sequence gives decreasing order of elements

What differentiates a circular linked list from a normal linked list?

1. You cannot have the 'next' pointer point to null in a circular linked list
2. It is faster to traverse the circular linked list
3. You may or may not have the 'next' pointer point to null in a circular linked list
4. Head node is known in circular linked list

You may or may not have the 'next' pointer point to null in a circular linked list

What is the speciality about the inorder traversal of a binary search tree?

1. It traverses in a non increasing order
2. It traverses in an increasing order
3. It traverses in a random fashion
4. It traverses based on priority of the node

5. It traverses in an increasing order
1.
1.
1. Preorder traversal
2. Inorder traversal
3. Postorder traversal
4. Level order traversal

Postorder traversal

What does the following piece of code do?

function Tree(root)
{
console.log(root.data());
Tree(root.left());
Tree(root.right());
}

1. Preorder traversal
2. Inorder traversal
3. Postorder traversal
4. Level order traversal

Preorder traversal

What does the following piece of code do?

```
function Tree( root)
{
Tree(root.left());
Tree(root.right());
console.log(root.data());
}
```

What is the functionality of the following piece of code? Select the most appropriate.

```
function linked (data)
{
int flag = 0;
if( head != null)
{
temp = head.getNext();
while((temp != head) && (!(temp.getItem() == data)))
{
temp = temp.getNext();
flag = 1;
break;
}
}
if(flag)
console.log("success");
else
console.log("fail");
}
```

1. Print success if a particular element is not found
2. Print fail if a particular element is not found
3. Print success if a particular element is equal to 1
4. Print fail if the list is empty

Print fail if a particular element is not found

What is the output of following function for start pointing to first node of following linked list?

1->2->3->4->5->6

```
function fun( start)
{
if(start == null)
return;
console.log(start.data);
if(start.next != null )
fun(start.next.next);
```

console.log(start.data);
}

1. 1 4 6 6 4 1
2. 1 3 5 1 3 5
3. 1 2 3 5
4. 1 3 5 5 3 1

1 3 5 5 3 1

What are the worst case and average case complexities of a binary search tree?

1. O(n), O(n)
2. O(logn), O(logn)
3. O(logn), O(n)
4. O(n), O(logn)

O(n), O(logn)

Linked list is considered as an example of ___________ type of memory allocation.

1. Dynamic
2. Static
3. Compile time
4. Heap

Dynamic

What are the conditions for an optimal binary search tree and what is its advantage?

1. The tree should not be modified and you should know how often the keys are accessed, it improves the lookup cost
2. You should know the frequency of access of the keys, improves the lookup time
3. The tree can be modified and you should know the number of elements in the tree before hand, it improves the deletion time
4. The tree should be just modified and improves the lookup time

The tree should not be modified and you should know how often the keys are accessed, it improves the lookup cost

Given, arr = {1,3,5,6,7,9,14,15,17,19} key = 17 and delta = {5,3,1,0}
How many key comparisons are made?(exclude the comparison used to decide the left or right sub array)

1. 3
2. 4
3. 5
4. 6

3

What will be the height of a balanced full binary tree with 8 leaves?

1. 8
2. 5
3. 6
4. 4

4

The balance factor of a node in a binary tree is defined as ______

1. addition of heights of left and right subtrees
2. height of right subtree minus height of left subtree
3. height of left subtree minus height of right subtree
4. height of right subtree minus one

height of left subtree minus height of right subtree

Figure below is a balanced binary tree. If a node inserted as child of the node R, how many nodes will become unbalanced?

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

1

A binary tree is balanced if the difference between left and right subtree of every node is not more than _____

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

1

Which of the following tree data structures is not a balanced binary tree?

1. AVL tree
2. Red-black tree
3. Splay tree
4. B-tree

B-tree

Given an array arr = {45,77,89,90,94,99,100} and key = 99; what are the mid values(corresponding array elements) in the first and second levels of recursion?

1. 90 and 99
2. 90 and 94
3. 89 and 99
4. 89 and 94

90 and 99

Balanced binary tree with n items allows the lookup of an item in _____ worst-case time.

1. O(log n)
2. O(nlog 2)
3. O(n)
4. O(1)

O(log n)

Which of the following data structures can be efficiently implemented using height balanced binary search tree?

1. sets
2. priority queue
3. All the above
4. None of the above

All the above

Two balanced binary trees are given with m and n elements respectively. They can be merged into a balanced binary search tree in _____ time.

1. O(m+n)
2. O(mn)
3. O(m)
4. O(mlog n)

O(m+n)

Which of the following is an advantage of balanced binary search tree, like AVL tree, compared to binary heap?

1. insertion takes less time
2. deletion takes less time
3. searching takes less time
4. construction of the tree takes less time than binary heap

insertion takes less time

AVL trees are more balanced than Red-black trees.

1. TRUE
2. FALSE

TRUE

The figure shown below is a balanced binary tree. If node P is deleted, which of the following nodes will get unbalanced?

1. U
2. M
3. H
4. A

U

Which of the following is not the self balancing binary search tree?

1. AVL Tree
2. 2-3-4 Tree
3. Red – Black Tree
4. Splay Tree

2-3-4 Tree

The binary tree sort implemented using a self – balancing binary search tree takes _______ time is worst case.

1. O(n log n)
2. O(nlog 2)
3. O(n)
4. O(1)

O(n log n)

An AVL tree is a self – balancing binary search tree, in which the heights of the two child sub trees of any node differ by __________

1. At least one
2. At most one
3. Two
4. At most two

At most one

Associative arrays can be implemented using ___________

1. B-tree
2. A doubly linked list
3. A single linked list
4. A self balancing binary search tree

A self balancing binary search tree

Self – balancing binary search trees have a much better average-case time complexity than hash tables.

1. TRUE
2. FALSE

FALSE

Which of the following is a self – balancing binary search tree?

1. 2-3 tree
2. Threaded binary tree
3. AA tree
4. Treap

AA tree

What is the probability of selecting a tree uniformly at random?

1. Equal to Catalan Number
2. Less Than Catalan Number
3. Greater than Catalan Number
4. Reciprocal of Catalan Number

Reciprocal of Catalan Number

What is the time complexity to count the number of elements in the linked list?

1. O(1)
2. O(n)
3. O(logn)
4. O(n^2)

O(n)

What is the space complexity for deleting a linked list?

1. O(1)
2. O(n)
3. O(logn)
4. O(n^2)

O(1)

Which of these is not an application of a linked list?

1. To implement file systems
2. For separate chaining in hash-tables
3. To implement non-binary trees
4. Random Access of elements

Random Access of elements

Which of the following is false about a doubly linked list?

1. We can navigate in both the directions
2. It requires more space than a singly linked list
3. The insertion and deletion of a node take a bit longer
4. Implementing a doubly linked list is easier than singly linked list

Implementing a doubly linked list is easier than singly linked list

What is a memory efficient double linked list?

1. Each node has only one pointer to traverse the list back and forth
2. The list has breakpoints for faster traversal
3. An auxiliary singly linked list acts as a helper list to traverse through the doubly linked list
4. A doubly linked list that uses bitwise AND operator for storing addresses

Each node has only one pointer to traverse the list back and forth

How do you calculate the pointer difference in a memory efficient double linked list?

1. head xor tail
2. pointer to previous node xor pointer to next node
3. pointer to previous node – pointer to next node
4. pointer to next node – pointer to previous node

pointer to previous node xor pointer to next node

What is the worst case time complexity of inserting a node in a doubly linked list?

1. O(nlogn)
2. O(logn)
3. O(n)
4. O(1)

O(n)

What differentiates a circular linked list from a normal linked list?

1. You cannot have the ‘next’ pointer point to null in a circular linked list
2. It is faster to traverse the circular linked list
3. You may or may not have the ‘next’ pointer point to null in a circular linked list
4. Head node is known in circular linked list

You may or may not have the ‘next’ pointer point to null in a circular linked list

What is the time complexity of searching for an element in a circular linked list?

1. O(1)
2. O(n)
3. O(logn)
4. O(n^2)

O(n)

Which of the following is false about a circular linked list?

1. Every node has a successor
2. Time complexity of inserting a new node at the head of the list is O(1)
3. Time complexity for deleting the last node is O(n)
4. We can traverse the whole circular linked list by starting from any point

Time complexity of inserting a new node at the head of the list is O(1)