Introduction to Algorithms

CELEN086

Seminar 8
(w/c 02/12/2024)

Outline

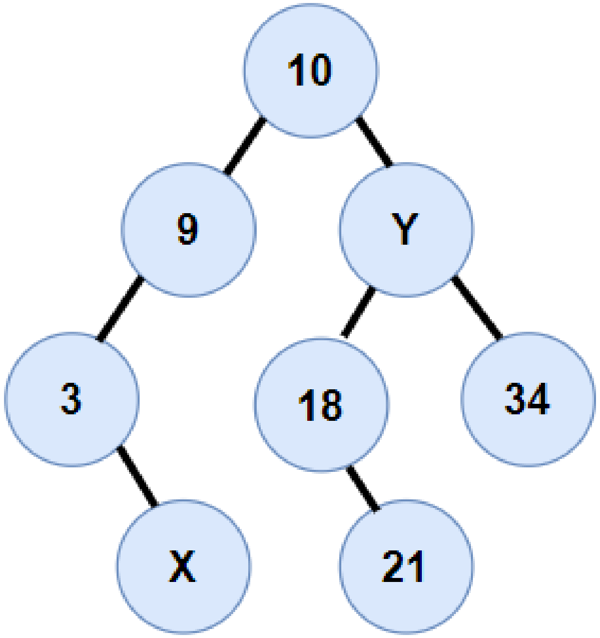
In this seminar, we will study and review on following topics:

- Binary search tree (BST)
- Designing recursive algorithms on BST
- Building BST from a list
- Traversal scheme and algorithm design

You will also learn useful Math/CS concepts and vocabularies.

Binary search tree

Determine the range of values X and Y that can be stored in the binary search tree:



$X>3$ and $X<9$

Range of X: $3<X<9$

$Y>10$ and $Y<34$

and $Y>18$ and $Y>21$

Range of Y: $21<Y<34$

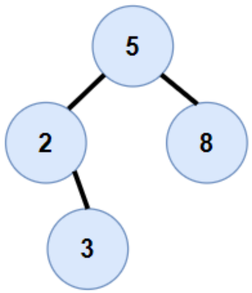
Algorithm Example: isBST (with issue)

Write a recursive algorithm isBST(T) to determine if a binary tree is also a binary search tree or not.

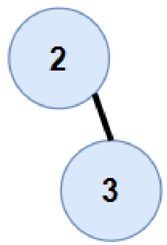
```
Algorithm: isBST(T)
Requires: a non-empty binary tree T
Returns: True if it is a BST; False otherwise

if  isLeaf(left(T)) && isLeaf(right(T))
        return True
elseif  isLeaf(left(T))
        return  root(T)<root(right(T)) && isBST(right(T))
elseif  isLeaf(right(T))
        return  root(T)>root(left(T)) && isBST(left(T))
else
        return  isBST(left(T)) && isBST(right(T))
endif
```

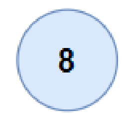
Trace (good case)

```
1. if isLeaf(left(T)) && isLeaf(right(T))
2.        return True
3. elseif isLeaf(left(T))
4.        return root(T)<root(right(T)) && isBST(right(T))
5. elseif isLeaf(right(T))
6.        return root(T)>root(left(T)) && isBST(left(T))
7. else
8.        return isBST(left(T)) && isBST(right(T))
9. endif
```

isBST(

)

Line 8: return isBST(

) && isBST(

) = True && True = True

Line 1: return True

Line 4: return

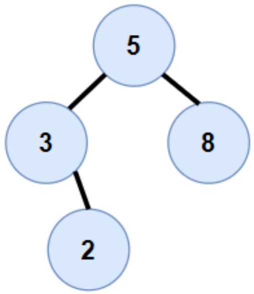
2<3 && isBST(

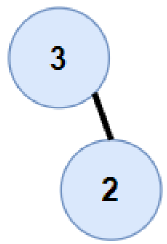
) = True && True = True

Line 1: return True

Trace (good case)

```
1.  if isLeaf(left(T)) && isLeaf(right(T))
2.          return True
3.  elseif isLeaf(left(T))
4.          return root(T)<root(right(T)) && isBST(right(T))
5.  elseif isLeaf(right(T))
6.          return root(T)>root(left(T)) && isBST(left(T))
7.  else
8.          return isBST(left(T)) && isBST(right(T))
9.  endif
```

isBST()

Line 8: return isBST() && isBST(8) = False && True = False

Line 1: return True

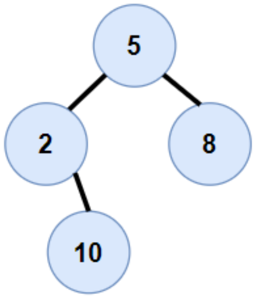
Line 4: return

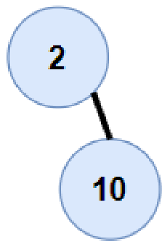
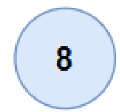
3<2 && isBST(2) = False && True = False

Line 1: return True

Trace (wrong case)

```
1. if isLeaf(left(T)) && isLeaf(right(T))
2.       return True
3. elseif isLeaf(left(T))
4.       return root(T)<root(right(T)) && isBST(right(T))
5. elseif isLeaf(right(T))
6.       return root(T)>root(left(T)) && isBST(left(T))
7. else
8.       return isBST(left(T)) && isBST(right(T))
9. endif
```

isBST()

Line 8: return isBST() && isBST() = True && True = True

Line 1: return True

Line 4: return

2<10 && isBST() = True && True = True

Line 1: return True

Clearly the answer obtained here is wrong.

Algorithm: isBST (correct version)

```
Algorithm: isBST(T)
Requires: a binary tree T
Returns: True if it is a BST; False otherwise
-----------------------------------------------------------
1.  if  isLeaf(left(T)) && isLeaf(right(T))
2.          return True
3.  elseif  isLeaf(left(T))
4.          return  root(T)<minBT(right(T))  &&  isBST(right(T))
5.  elseif  isLeaf(right(T))
6.          return  root(T)>maxBT(left(T))  &&  isBST(left(T))
7.  else
8.          return  isBST(left(T)) && root(T)>maxBT(left(T))
                        && isBST(right(T)) && root(T)<minBT(right(T))
9.  endif
```

Note:
Two sub-algorithms maxBT() and minBT() are used here, for computing the maximum and minimum value in a binary tree.

You should design them yourself as another practice.

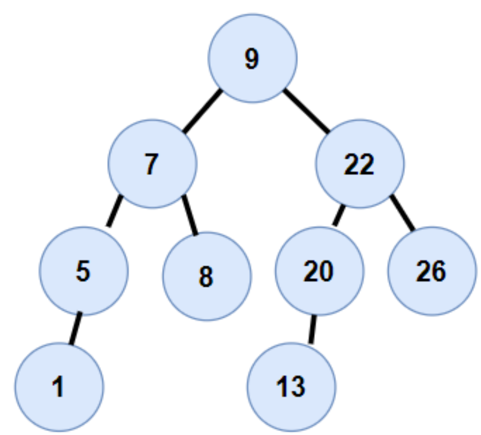
From List to BST

Create a binary search tree with minimal height (or depth) to store elements in the list [5, 13, 7, 26, 20, 1, 9, 22, 8].

Sort the list: $[\cancel{1}, \cancel{5}, \cancel{7}, \cancel{8}, \cancel{9}, \cancel{13}, \cancel{20}, \cancel{22}, \cancel{26}]$

$\boxed{\text{Height} = \lfloor \log_2 9 \rfloor = 3}$

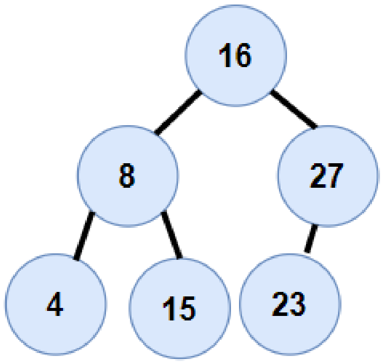
Levels:	Middle elements (N/2+1):
Level 0	9
Level 1	7 22
Level 2	5 8 20 26
Level 3	1 13



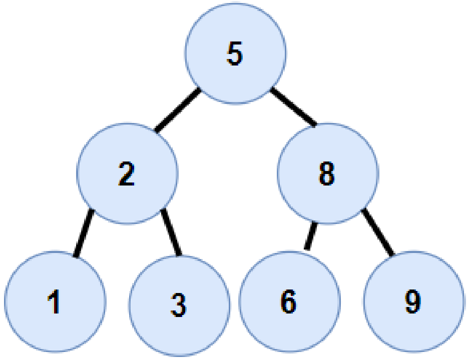
Practice

Create a binary search tree with minimal height (or depth) to store elements in the following lists:

L1=[15, 23, 8, 16, 27, 4]

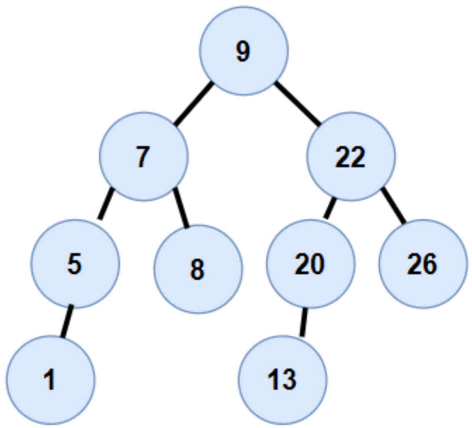


L2=[6, 8, 9, 3, 2, 1, 5]



Traversal schemes (Binary tree to List)

Find the lists obtained by traversing the binary tree with:



i. Breadth first scheme

[9,7,22,5,8,20,26,1,13]

ii. Depth first, preorder scheme (NLR)

[9,7,5,1,8,22,20,13,26]

iii. Depth first, inorder scheme (LNR)

[1,5,7,8,9,13,20,22,26]

iv. Depth first, postorder scheme (LRN)

[1,5,8,7,13,20,26,22,9]

Practice

You may pause the video for a few minutes.
Practice first before seeing the answers.

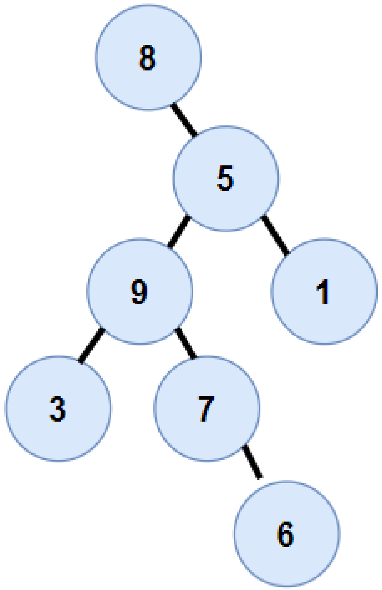
Find the lists obtained by traversing the binary tree with following schemes:

i. Breadth first

ii. Depth first, preorder

iii. Depth first, inorder

iv. Depth first, postorder



i. [8,5,9,1,3,7,6]

ii. [8,5,9,3,7,6,1]

iii. [8,3,9,7,6,5,1]

iv. [3,6,7,9,1,5,8]

We won't get a sorted list, if tree is not a binary search tree.

Practice

You may pause the video for a few minutes.
Practice first before seeing the answers.

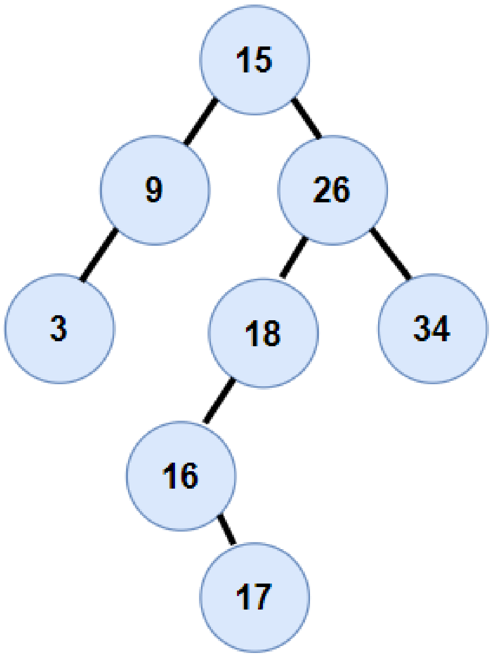
Find the lists obtained by traversing the binary search tree with following schemes:

i. Breadth first

ii. Depth first, preorder

iii. Depth first, inorder

iv. Depth first, postorder



i. [15,9,26,3,18,34,16,17]

ii. [15,9,3,26,18,16,17,34]

iii. [3,9,15,16,17,18,26,34]

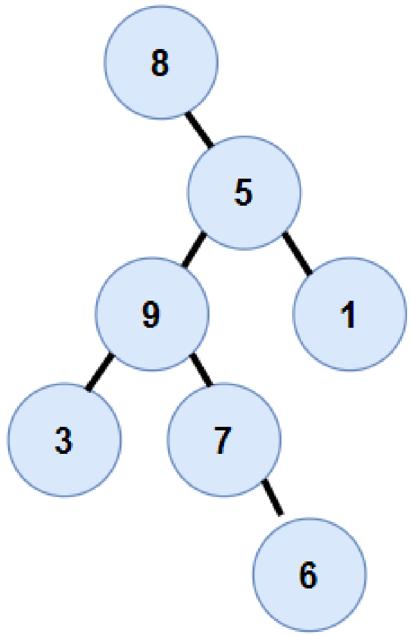
iv. [3,9,17,16,18,34,26,15]

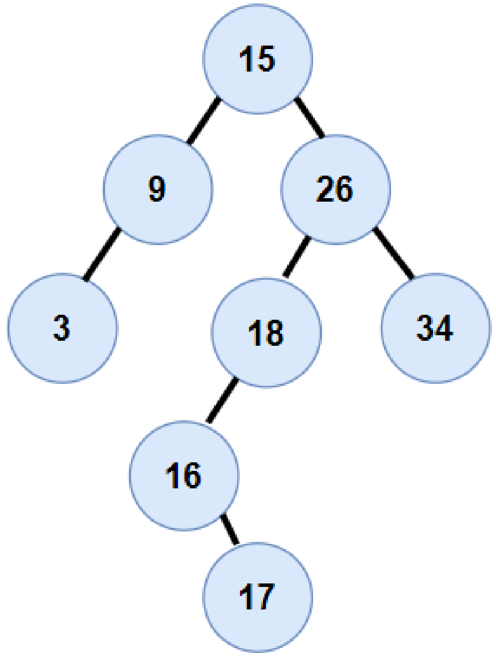
We will get a sorted list, if tree is a binary search tree.

Homework

Build binary search trees with minimal depth from the given trees.

You should demonstrate by showing necessary steps and draw your final binary search trees.





Algorithm for traversal schemes

Write a recursive algorithm preorder(BST) to return a list by traversing a binary search tree with preorder scheme.

Idea of designing this algorithm:

- Obtain three components
  - a (single-element) list for current node (or just the value itself)
  - a list for left sub-tree (recursively)
  - a list for right sub-tree (recursively)
- Combine them into a list with desired order (NLR)

concat(L1,L2) cons(x, L)

(Problem Sheet 5)

Algorithm: preorder(BST)

Algorithm: preorder(T)
Requires: a binary search tree T
Returns: a list for preorder scheme (NLR scheme)

```
1. if isLeaf(T)
2.         return nil
3. else
4.         let L1 = cons(root(T),nil) // N
5.         let L2 = preorder(left(T)) // L
6.         let L3 = preorder(right(T)) //R
7.         return concat(L1, concat(L2,L3)) // NLR
8. endif
```

Note 1:

The structure is very similar to the mergeSort algorithm in Seminar 6.

Trace it yourself with BST examples used in this session.

Note 2: review the algorithm concat(), for concatenating two lists.

Note 3: Line 4 and Line 7 can be replaced by following:

```
4.    let x=root(T)
7.    return cons(x, concat(L2,L3))
```

Practice

Write a recursive algorithm postorder(BST) to return a list by traversing a binary search tree with postorder scheme.

```
Algorithm: postorder(T)
Requires: a binary search tree T
Returns: a list for postorder scheme (LRN scheme)

if  isLeaf(T)
        return nil
else
        let L1 = cons(root(T),nil) // N
        let L2 = postorder(left(T)) // L
        let L3 = postorder(right(T)) //R
        return  concat(L2, concat(L3,L1)) // LRN
endif
```

Complete the following sentences using suitable word from the word list given below.

leftmost, minimum, ordering , smaller, greater, predecessor, successor, n-1, left, right, two, parent, leaf, node, sorted , in-order, preorder, maximum, rightmost, height, level, current.

1. In a binary tree, each node can have a maximum of ______ children.
2. A binary search tree is a binary tree where the key in the left child is ______ than the key in the parent node, and the key in the right child is ______ than the key in the parent node.
3. The ______ node in a binary search tree contains the smallest key value.
4. In a binary search tree, the ______ subtree of a node contains keys smaller than the node's key.
5. The height of a binary tree with n nodes is at most ______.
6. The process of inserting a new node into a binary search tree involves comparing the key of the new node with the key of the ______ and moving to the left or right child accordingly.
7. In a binary tree, a ______ node is a node that does not have any children.
8. The ______ traversal of a binary tree visits the nodes in the order: left child, parent, right child.
9. In a binary search tree, the ______ node contains the largest key value.
10. The ______ of a binary search tree represents the maximum number of edges in the path from the root to a leaf node.
11. The process of deleting a node from a binary search tree involves replacing it with its ______ or ______ and deleting the duplicate node from its original position.
12. In a binary search tree, the ______ property allows for efficient search, insertion, and deletion operations.

1. In a binary tree, each node can have a maximum of ______ children. Answer: two
2. A binary search tree is a binary tree where the key in the left child is ______ than the key in the parent node, and the key in the right child is ______ than the key in the parent node. Answer: smaller, greater
3. The ______ node in a binary search tree contains the smallest key value. Answer: leftmost or minimum
4. In a binary search tree, the ______ subtree of a node contains keys smaller than the node's key. Answer: left
5. The height of a binary tree with n nodes is at most ______. Answer: n - 1
6. The process of inserting a new node into a binary search tree involves comparing the key of the new node with the key of the ______ and moving to the left or right child accordingly. Answer: current or parent
7. In a binary tree, a ______ node is a node that does not have any children. Answer: leaf
8. The ______ traversal of a binary tree visits the nodes in the order: left child, parent, right child. Answer: in-order
9. In a binary search tree, the ______ node contains the largest key value. Answer: rightmost or maximum
10. The ______ of a binary search tree represents the maximum number of edges in the path from the root to a leaf node. Answer: height
11. The process of deleting a node from a binary search tree involves replacing it with its ______ or ______ and deleting the duplicate node from its original position. Answer: predecessor, successor
12. In a binary search tree, the ______ property allows for efficient search, insertion, and deletion operations. Answer: ordering or sorted