ESc 101: Fundamentals of Computing

Lecture 38

Apr 15, 2010

Outline

Storing Elements in Sorted Order

- In many applications, a large number of elements are stored.
- In many of the operations performed on these elements, it is useful to have the elements in a sorted order.
- Example:
  - Details of students at IITK: name, roll number, department etc.
  - For each student, we can have a `struct` to store all the information.
  - If these elements are stored sorted by roll number, we can easily find the number of students in a batch for example.

Storing Elements in Sorted Order

- In many applications, a large number of elements are stored.
- In many of the operations performed on these elements, it is useful to have the elements in a sorted order.
- Example:
  - Details of students at IITK: name, roll number, department etc.
  - For each student, we can have a `struct` to store all the information.
  - If these elements are stored sorted by roll number, we can easily find the number of students in a batch for example.

Storing Elements in Sorted Order

- In many applications, a large number of elements are stored.
- In many of the operations performed on these elements, it is useful to have the elements in a sorted order.
- Example:
  - Details of students at IITK: name, roll number, department etc.
  - For each student, we can have a `struct` to store all the information.
  - If these elements are stored sorted by roll number, we can easily find the number of students in a batch for example.

Storing Elements in Sorted Order

- In many applications, a large number of elements are stored.
- In many of the operations performed on these elements, it is useful to have the elements in a sorted order.
- Example:
  - Details of students at IITK: name, roll number, department etc.
  - For each student, we can have a `struct` to store all the information.
  - If these elements are stored sorted by roll number, we can easily find the number of students in a batch for example.

Storing Elements in Sorted Order

- In many applications, a large number of elements are stored.
- In many of the operations performed on these elements, it is useful to have the elements in a sorted order.
- Example:
  - Details of students at IITK: name, roll number, department etc.
  - For each student, we can have a `struct` to store all the information.
  - If these elements are stored sorted by roll number, we can easily find the number of students in a batch for example.

Linked Lists Work But Are Inefficient

- We can store all the elements in a linked list ordered by roll number.
- Then insertion will need to change: we need to insert the element at its right position in the list, not at the start.
- This is not very efficient.
- In fact, search and delete are also rather inefficient in linked lists.
- So we need a better data structure!

Linked Lists Work But Are Inefficient

- We can store all the elements in a linked list ordered by roll number.
- Then insertion will need to change: we need to insert the element at its right position in the list, not at the start.
- This is not very efficient.
- In fact, search and delete are also rather inefficient in linked lists.
- So we need a better data structure!

Linked Lists Work But Are Inefficient

- We can store all the elements in a linked list ordered by roll number.
- Then insertion will need to change: we need to insert the element at its right position in the list, not at the start.
- This is not very efficient.
- In fact, search and delete are also rather inefficient in linked lists.
- So we need a better data structure!

Linked Lists Work But Are Inefficient

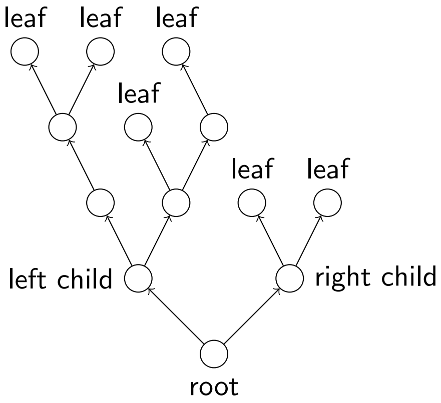
- We can store all the elements in a linked list ordered by roll number.
- Then insertion will need to change: we need to insert the element at its right position in the list, not at the start.
- This is not very efficient.
- In fact, search and delete are also rather inefficient in linked lists.
- So we need a better data structure!

Linked Lists Work But Are Inefficient

- We can store all the elements in a linked list ordered by roll number.
- Then insertion will need to change: we need to insert the element at its right position in the list, not at the start.
- This is not very efficient.
- In fact, search and delete are also rather inefficient in linked lists.
- So we need a better data structure!

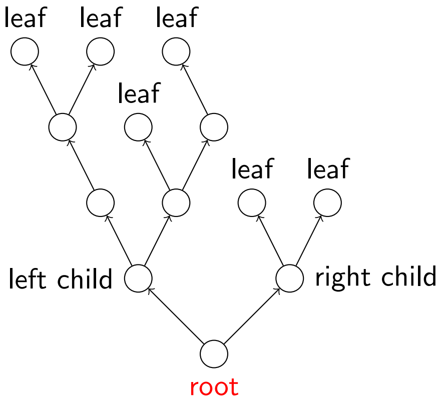
Binary Trees

A binary tree is a structure of the following kind:



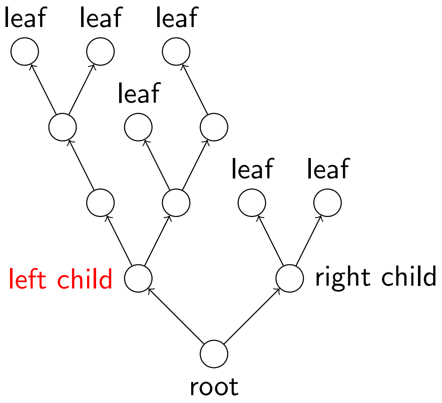
Binary Trees

A binary tree is a structure of the following kind:



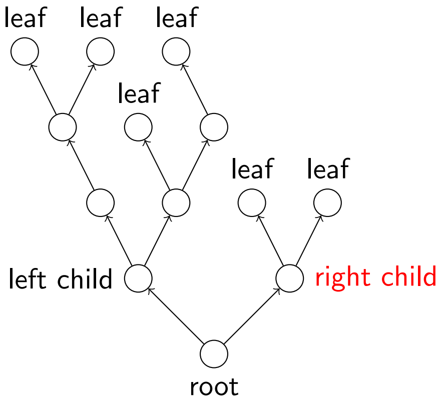
Binary Trees

A binary tree is a structure of the following kind:



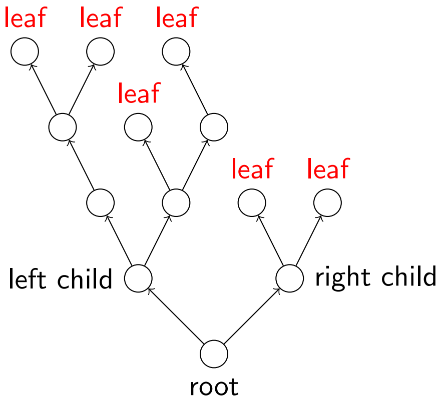
Binary Trees

A binary tree is a structure of the following kind:



Binary Trees

A binary tree is a structure of the following kind:



Binary Search Trees

These are binary trees with the following properties:

- Each node stores an element
- At any node, the element stored in greater than or equal to all elements in the left subtree at the node and less than or equal to all elements in the right subtree at the node.
- These trees help in speeding up operations.

Binary Search Trees

These are binary trees with the following properties:

- Each node stores an element
- At any node, the element stored in greater than or equal to all elements in the left subtree at the node and less than or equal to all elements in the right subtree at the node.
- These trees help in speeding up operations.

Binary Search Trees

These are binary trees with the following properties:

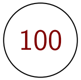
- Each node stores an element
- At any node, the element stored in greater than or equal to all elements in the left subtree at the node and less than or equal to all elements in the right subtree at the node.
- These trees help in speeding up operations.

Binary Search Trees

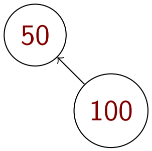
These are binary trees with the following properties:

- Each node stores an element
- At any node, the element stored in greater than or equal to all elements in the left subtree at the node and less than or equal to all elements in the right subtree at the node.
- These trees help in speeding up operations.

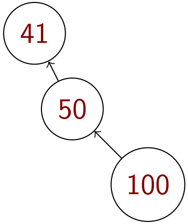
Inserting in a Binary Search Tree



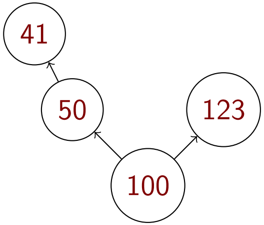
Inserting in a Binary Search Tree



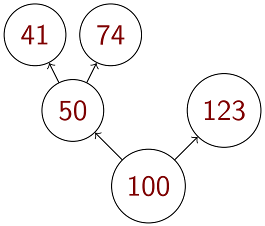
Inserting in a Binary Search Tree



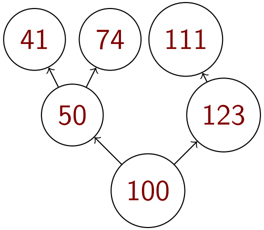
Inserting in a Binary Search Tree



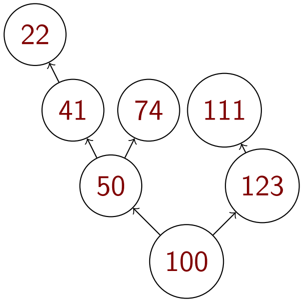
Inserting in a Binary Search Tree



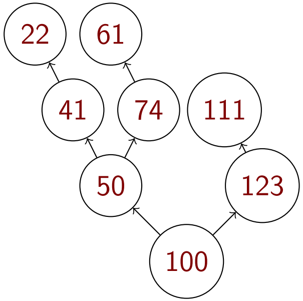
Inserting in a Binary Search Tree



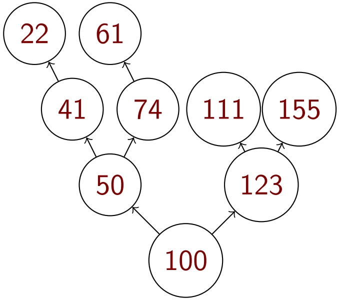
Inserting in a Binary Search Tree



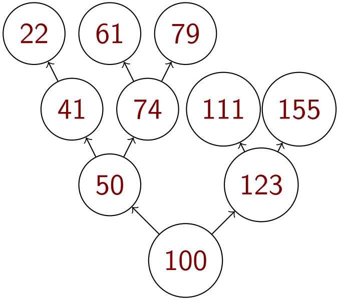
Inserting in a Binary Search Tree



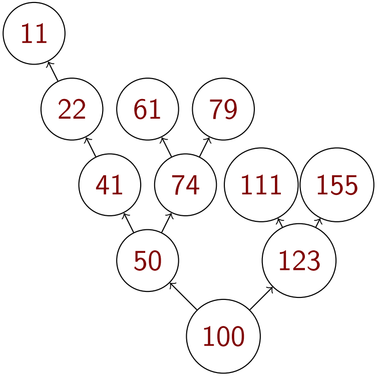
Inserting in a Binary Search Tree



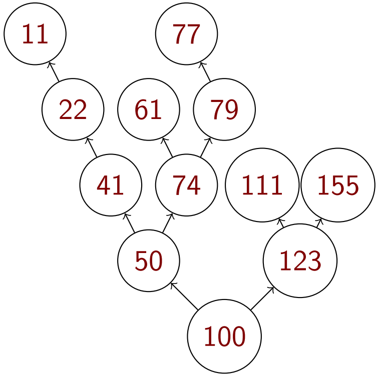
Inserting in a Binary Search Tree



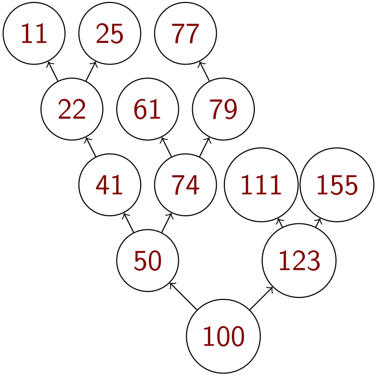
Inserting in a Binary Search Tree



Inserting in a Binary Search Tree



Inserting in a Binary Search Tree


11
25
77
22
61
79
41
74
111
155
50
123
100