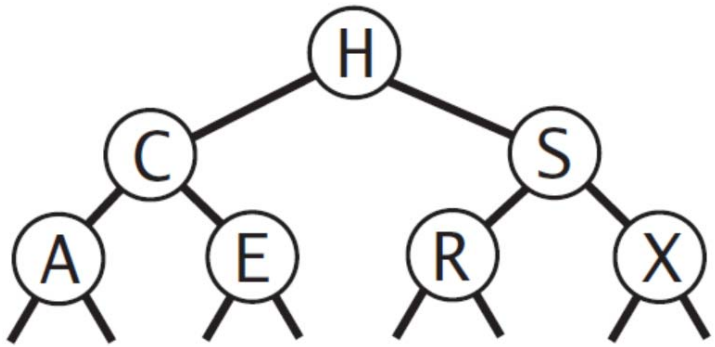
CSE214 Data Structures

Binary Search Tree

YoungMin Kwon

Binary Search Tree

- A Binary Search Tree is an ordered tree with each node has at most two children
  - Nodes in the left subtree of a node have smaller keys than the node's key
  - Nodes in the right subtree of a node have keys larger than or equal to the node's key

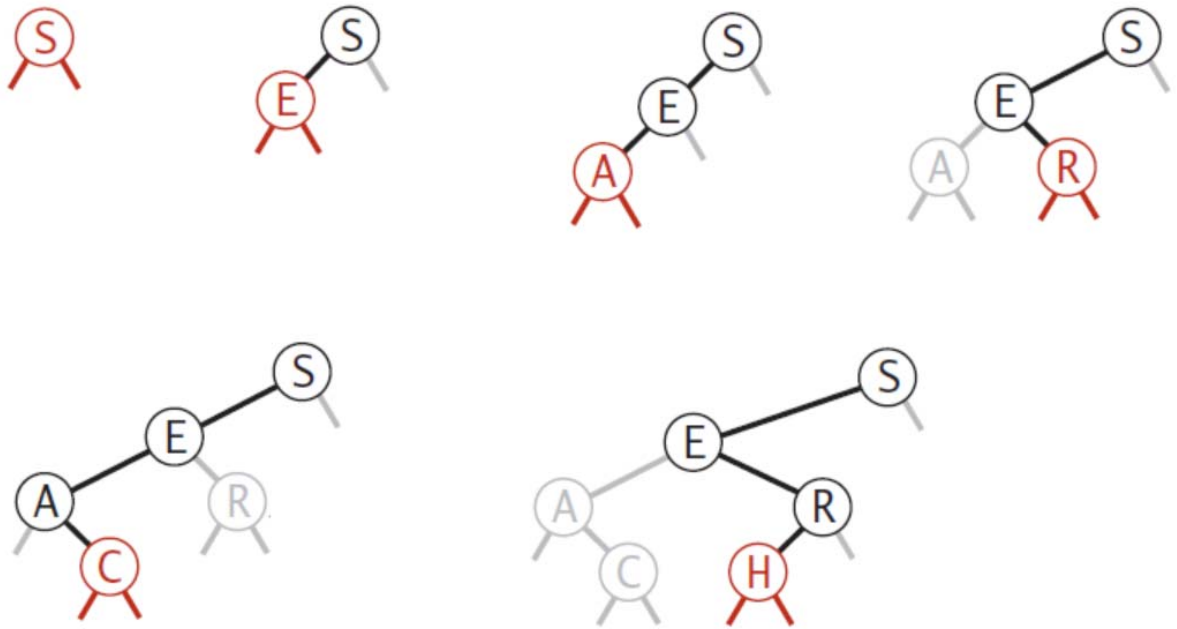


Binary Search Tree

- Adding a key to a tree
  - If the key is smaller than the root's key of the tree, add it to its left subtree recursively
  - Otherwise, add it to its right subtree recursively
  - If the left or the right link is null, create a node with the key and add it to the link

Binary Search Tree

- Example
  - Adding S E A R C H to a binary search tree



```java
import java.util.ArrayList;

public class BinarySearchTree<E extends Comparable<E>> {
    private static class Node<E extends Comparable<E>> {
        public E e;
        public Node<E> left, right;
        public Node(E e, Node<E> left, Node<E> right) {
            this.e = e; this.left = left; this.right = right;
        }
    }

    private Node<E> root;

    public BinarySearchTree() {}

    //TODO: implement find method
    public Node<E> find(E e) {
    }
    private Node<E> find(Node<E> node, E e) {
    }
```

```
//TODO: implement add method
public void add(E e) {
}
private void add(Node<E> node, E e) {
}

//TODO: implement visitInorder method
public Iterable<E> visitInorder() {
    java.util.List<E> snapshot = new ArrayList<E>(5);
}
private void visitInorder(Node<E> node, java.util.List<E> snapshot) {
}

public static void find(Integer[] arr) {
    BinarySearchTree<Integer> tree = new BinarySearchTree<>();
    for(Integer e : arr)
        tree.add(e);
    for(int i = 0; i < 10; i++)
        System.out.print((tree.find(i) != null) + ", ");
    System.out.println("");
}
```

```
    //TODO: implement sort by BinarySearchTree
    public static void sort(Integer[] arr) {
        BinarySearchTree<Integer> tree = new BinarySearchTree<>();

        //TODO: 1. add elements of arr to tree

        //TODO: 2. print the inorder traversal result

        System.out.println("");
    }

    public static void main(String[] args) {
        Integer[] arr = {5, 3, 6, 2, 8, 1, 9, 4, 7};
        find(arr);
        sort(arr);
    }
}
```