



<http://algs4.cs.princeton.edu>

3.2 BINARY SEARCH TREES

- ▶ *BSTs*
- ▶ *ordered operations*
- ▶ *deletion*



3.2 BINARY SEARCH TREES

- ▶ *BSTs*
- ▶ *ordered operations*
- ▶ *deletion*

Binary search trees

Definition. A BST is a **binary tree** in **symmetric order**.

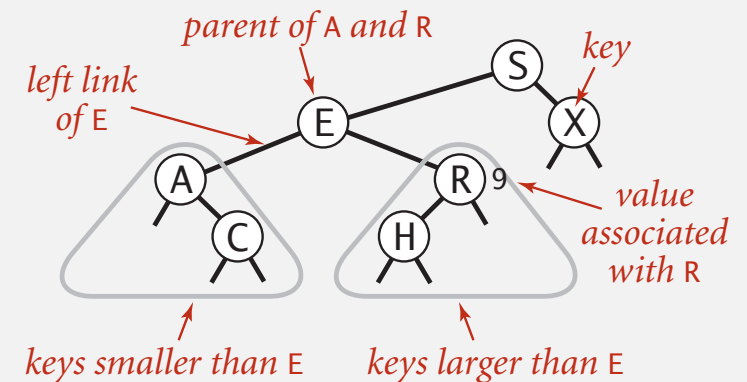
A binary tree is either:

- Empty.
- Two disjoint binary trees (left and right).



Symmetric order. Each node has a key, and every node's key is:

- Larger than all keys in its left subtree.
- Smaller than all keys in its right subtree.



BST representation in Java

Java definition. A BST is a reference to a root Node.

A Node is comprised of four fields:

- A Key and a Value.
- A reference to the left and right subtree.

↑ smaller keys ↑ larger keys

```
private class Node
{
    private Key key;
    private Value val;
    private Node left, right;
    public Node(Key key, Value val)
    {
        this.key = key;
        this.val = val;
    }
}
```

Key and Value are generic types; Key is Comparable



BST implementation (skeleton)

```
public class BST<Key extends Comparable<Key>, Value>  
{
```

```
    private Node root;
```

← root of BST

```
    private class Node  
    { /* see previous slide */ }
```

```
    public void put(Key key, Value val)  
    { /* see next slides */ }
```

```
    public Value get(Key key)  
    { /* see next slides */ }
```

```
    public void delete(Key key)  
    { /* see next slides */ }
```

```
    public Iterable<Key> iterator()  
    { /* see next slides */ }
```

```
}
```

Binary search tree demo

Search. If less, go left; if greater, go right; if equal, search hit.

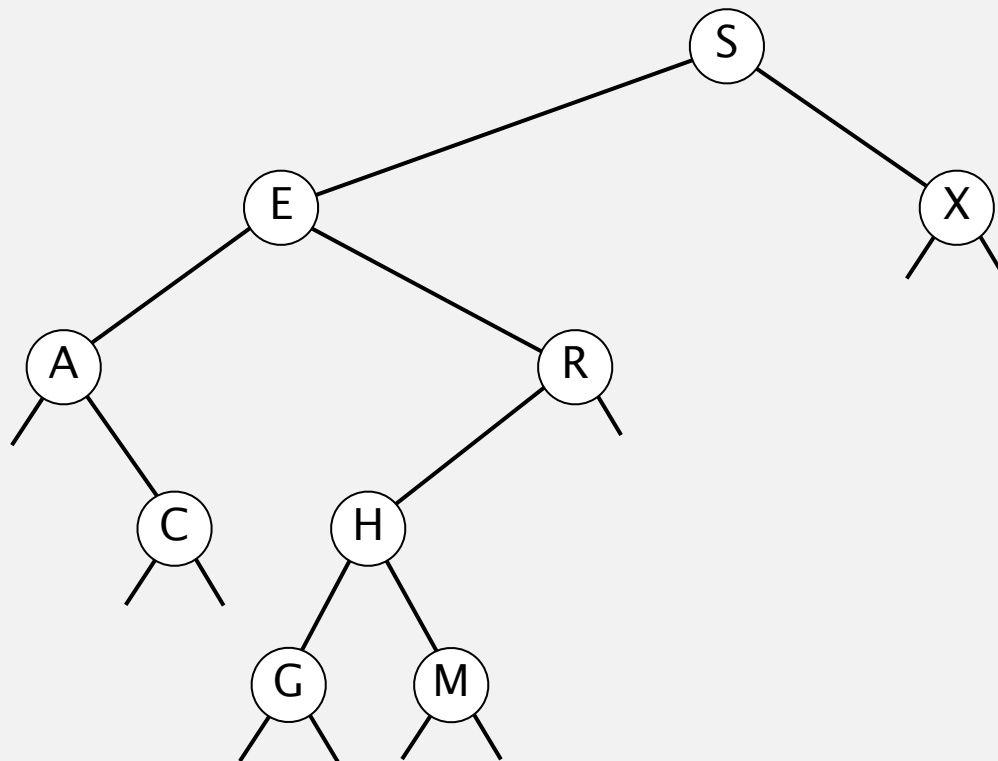
successful search for H



Binary search tree demo

Insert. If less, go left; if greater, go right; if null, insert.

insert G



BST search: Java implementation

Get. Return value corresponding to given key, or null if no such key.

```
public Value get(Key key)
{
    Node x = root;
    while (x != null)
    {
        int cmp = key.compareTo(x.key);
        if      (cmp < 0) x = x.left;
        else if (cmp > 0) x = x.right;
        else if (cmp == 0) return x.val;
    }
    return null;
}
```

Cost. Number of compares is equal to 1 + depth of node.