CSE 373 Midterm Review Lecture Discussion Session

Where you'll want to go, What you'll need to know

rkedziorski: kedzior@uw.edu

University of Washington

April 30, 2013

What this review lecture discussion session will cover

- Logistics
- Math
- Trees
- Lists, Stacks, and Queues
- Hashing
- Complexity

Logistics

you: But where is this midterm at? When?

Logistics

you: But where is this midterm at? When?

— Right Here, EEB 105; Friday May 3rd 2:30-3:20p

Logistics

you: But where is this midterm at? When?

— Right Here, EEB 105; Friday May 3rd 2:30-3:20p

you Wait, there will be essays?

Logistics

you: But where is this midterm at? When?

— Right Here, EEB 105; Friday May 3rd 2:30-3:20p

you Wait, there will be essays?

— No. The questions will involve short answers, drawing trees, and proving correctness

Logistics

you: But where is this midterm at? When?

— Right Here, EEB 105; Friday May 3rd 2:30-3:20p

you Wait, there will be essays?

— No. The questions will involve short answers, drawing trees, and proving correctness

you: Is the test Open Book?

Logistics

you: But where is this midterm at? When?

— Right Here, EEB 105; Friday May 3rd 2:30-3:20p

you Wait, there will be essays?

— No. The questions will involve short answers, drawing trees, and proving correctness

you: Is the test Open Book?

— You may bring **one 8.5"x11" page (double sided)** full of notes

Logistics

you: But where is this midterm at? When?

— Right Here, EEB 105; Friday May 3rd 2:30-3:20p

you Wait, there will be essays?

— No. The questions will involve short answers, drawing trees, and proving correctness

you: Is the test Open Book?

— You may bring **one 8.5"x11" page (double sided)** full of notes

you: That's super generous!

Logistics

you: But where is this midterm at? When?

— Right Here, EEB 105; Friday May 3rd 2:30-3:20p

you Wait, there will be essays?

— No. The questions will involve short answers, drawing trees, and proving correctness

you: Is the test Open Book?

— You may bring **one 8.5"x11" page (double sided)** full of notes

you: That's super generous!

— Professor Shapiro is very generous. I would make you do essay questions.

Mathematical Foundations

Proofs by Induction

Induction proofs are **correctness** proofs.

Basic Strategy: There are three parts to an induction proof

The Base Case: When some variable in question (call it n) is 0 or null, what happens?

The Inductive Hypothesis: Often only a single sentence of the form "Assume the code returns the correct result for all $n \leq k$"

The Inductive Step: Show that the result holds for $n = k + 1$ by explaining how the $k + 1$ case reduces to case(s) where $n \leq k$.

Mathematical Foundations

Proofs by Induction: Example

Prove by induction that the following procedure to determine the height of a binary tree returns the correct height.

```
height(T: treeptr): integer {
    if T == null return -1;
    else return 1 + max(height(T.left),height(T.right));
}
```

Mathematical Foundations

Proofs by Induction: Example

The Base Case:

Mathematical Foundations

Proofs by Induction: Example

The Base Case: When the tree is empty, T == null and the code correctly returns -1

The Inductive Hypothesis:

Mathematical Foundations

Proofs by Induction: Example

The Base Case: When the tree is empty, T == null and the code correctly returns -1

The Inductive Hypothesis: Assume the code returns the correct result for trees of height $\leq k$

The Inductive Step:

Mathematical Foundations

Proofs by Induction: Example

The Base Case: When the tree is empty, T == null and the code correctly returns -1

The Inductive Hypothesis: Assume the code returns the correct result for trees of height $\leq k$

The Inductive Step: Consider a tree of height $k+1$. The height function will add one to the height of the largest subtree and return. Because this tree is of height $k+1$, the subtrees must be of heights $\leq k$, and so by the inductive hypothesis their heights are correctly returned. Since the larger must be of height k (in order for this tree to have height $k+1$), the function correctly returns $k+1$.

Mathematical Foundations

Questions?

Trees

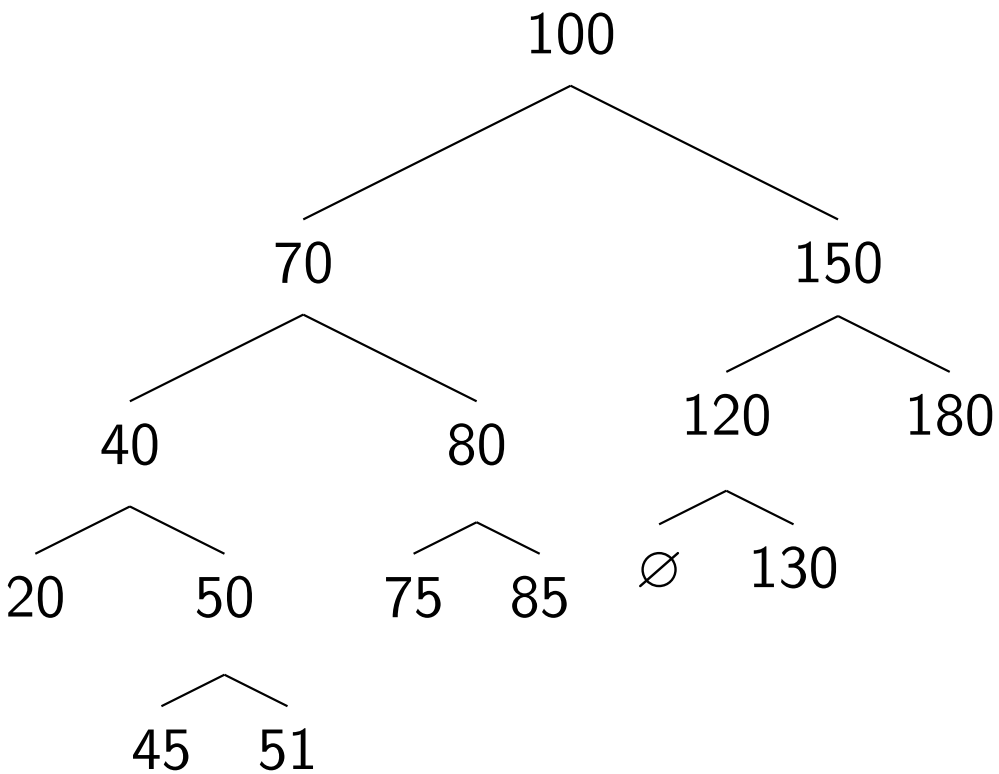
What to know:

- Operations on a Tree (Insert, Find, Delete, etc...); Recursive and Iterative versions
- Compute Balance Factors on nodes of a binary search tree
- Insert on an AVL tree: the 4 cases
- Splaying
- Insert and Find on a B+ Tree
- The time complexity of any algorithm above

Trees

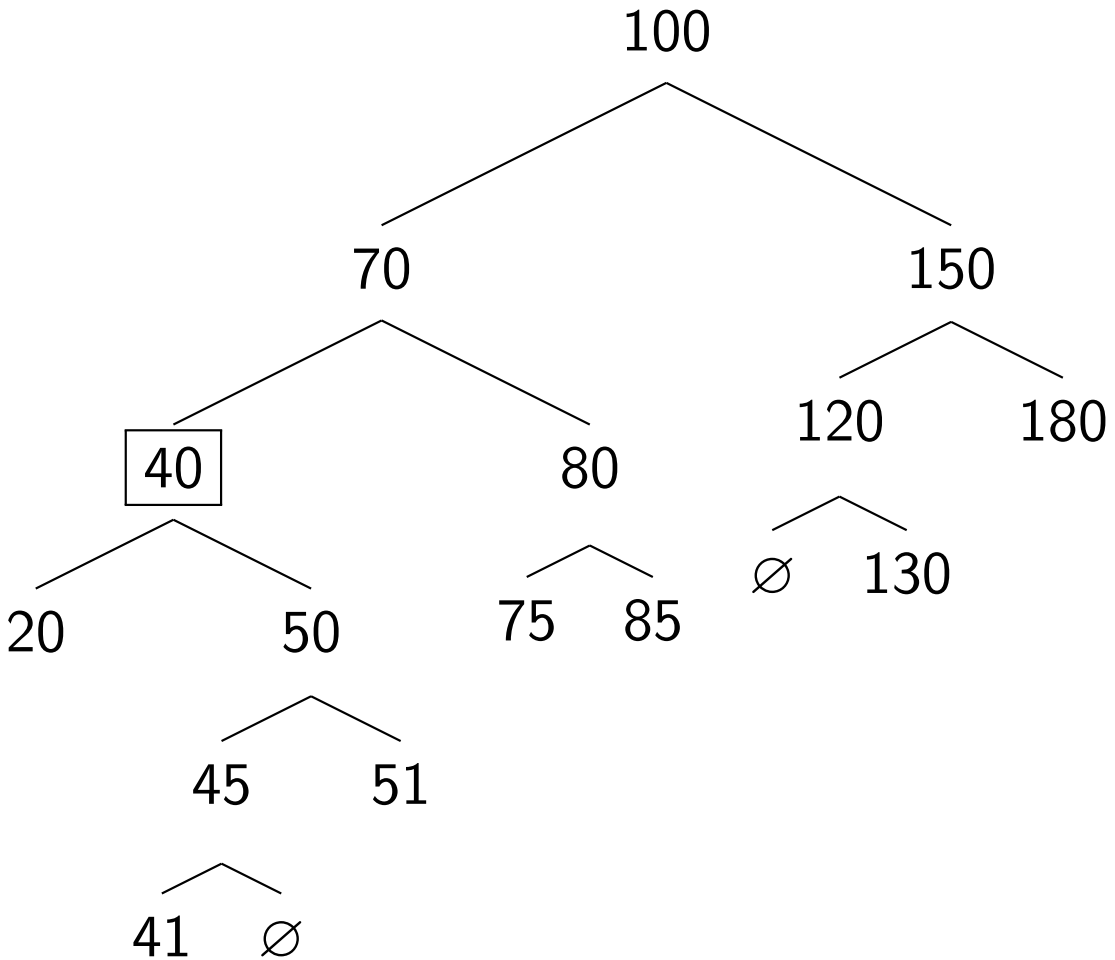
AVL Tree

Want to insert 41:



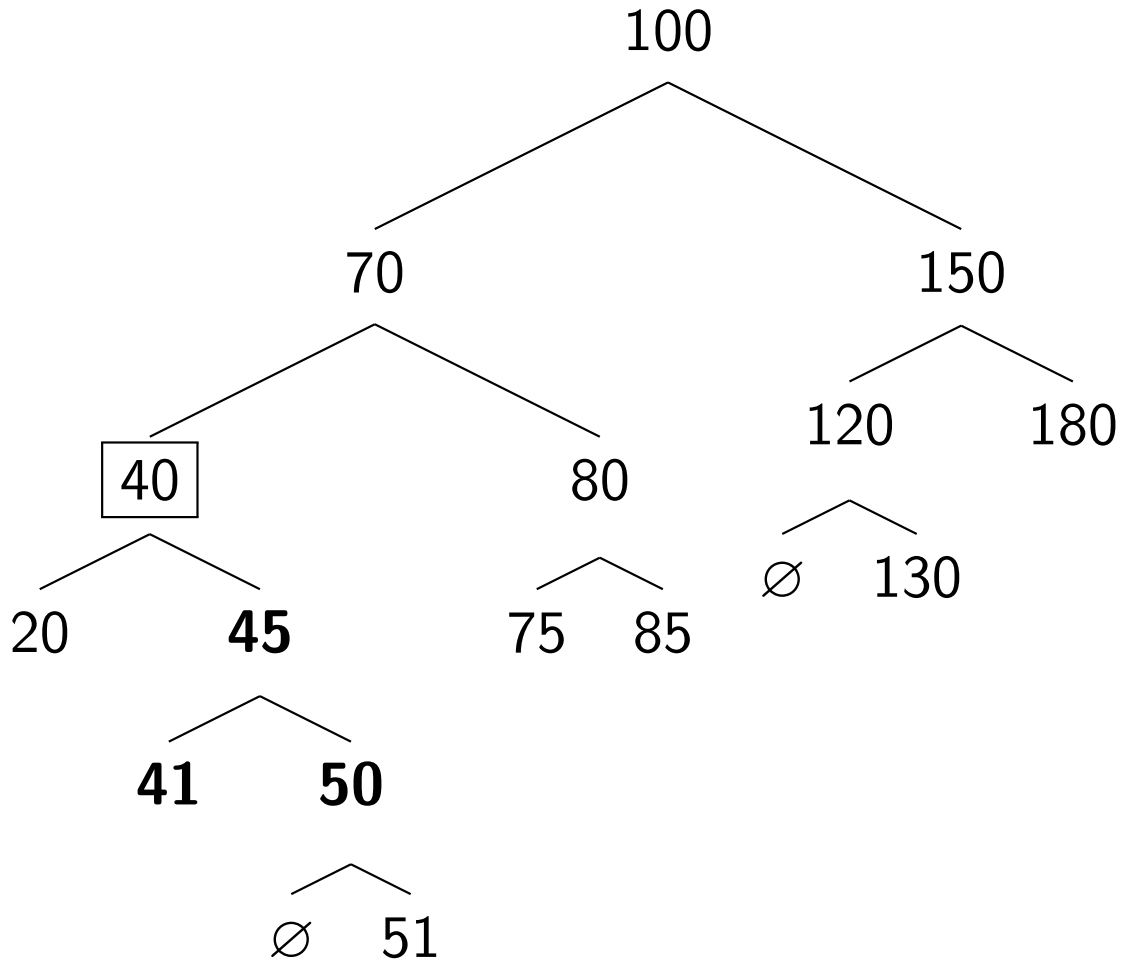
Trees

AVL Tree



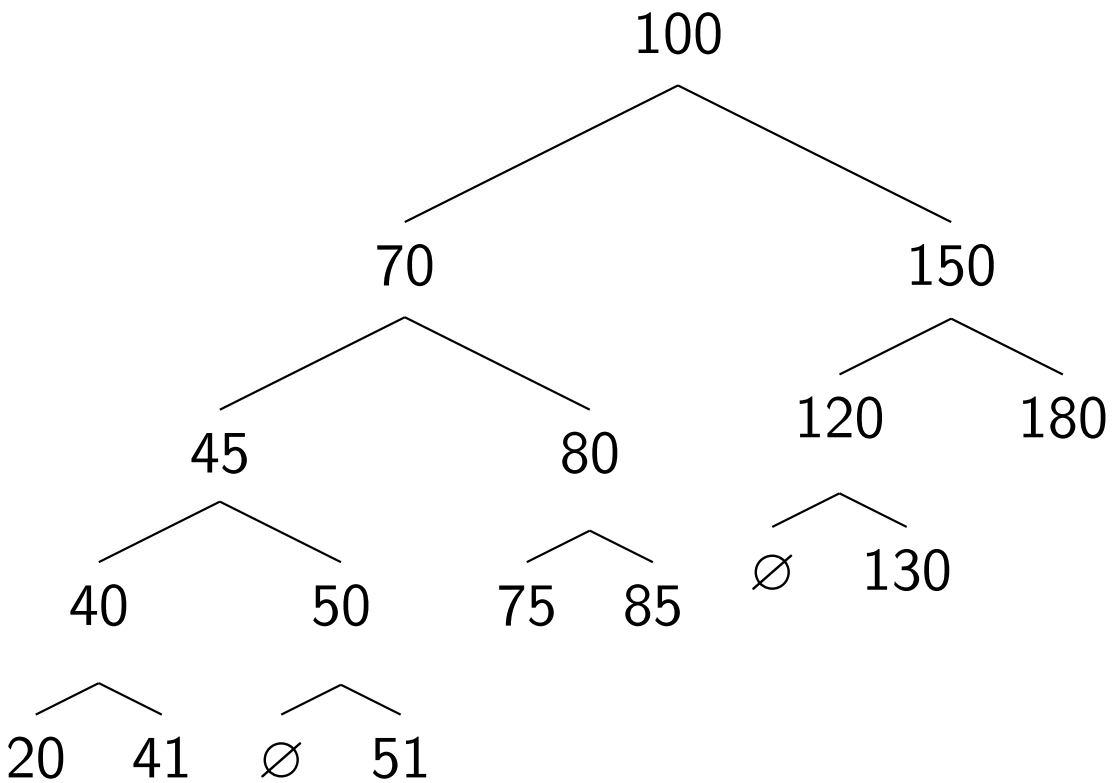
Trees

AVL Tree



Trees

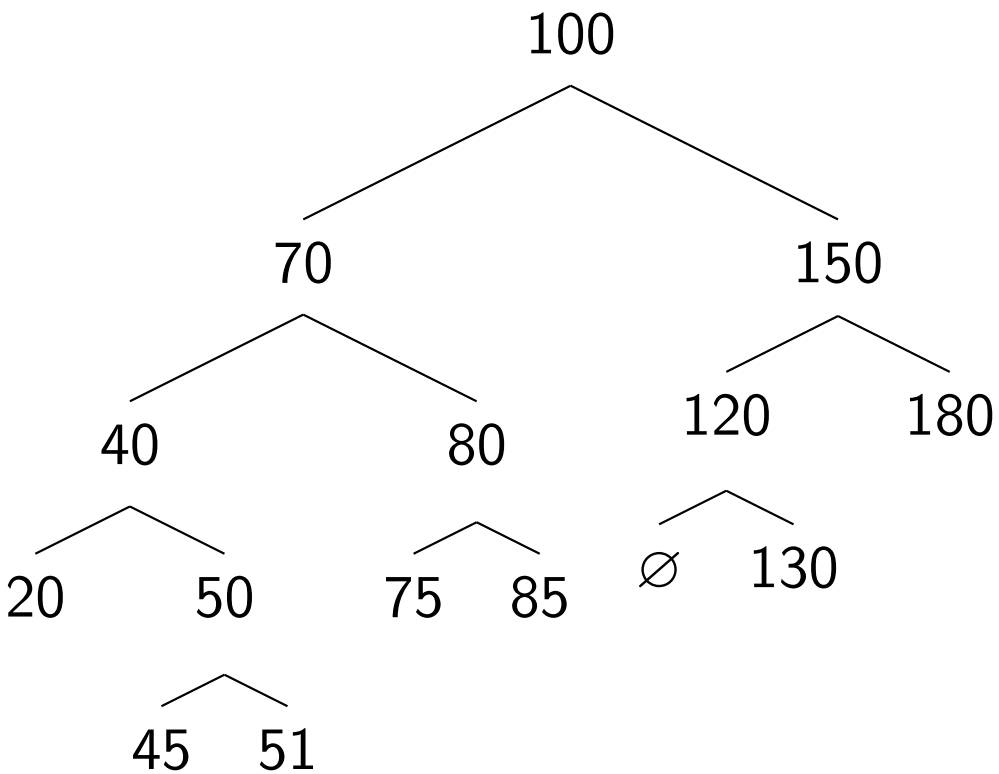
AVL Tree



Trees

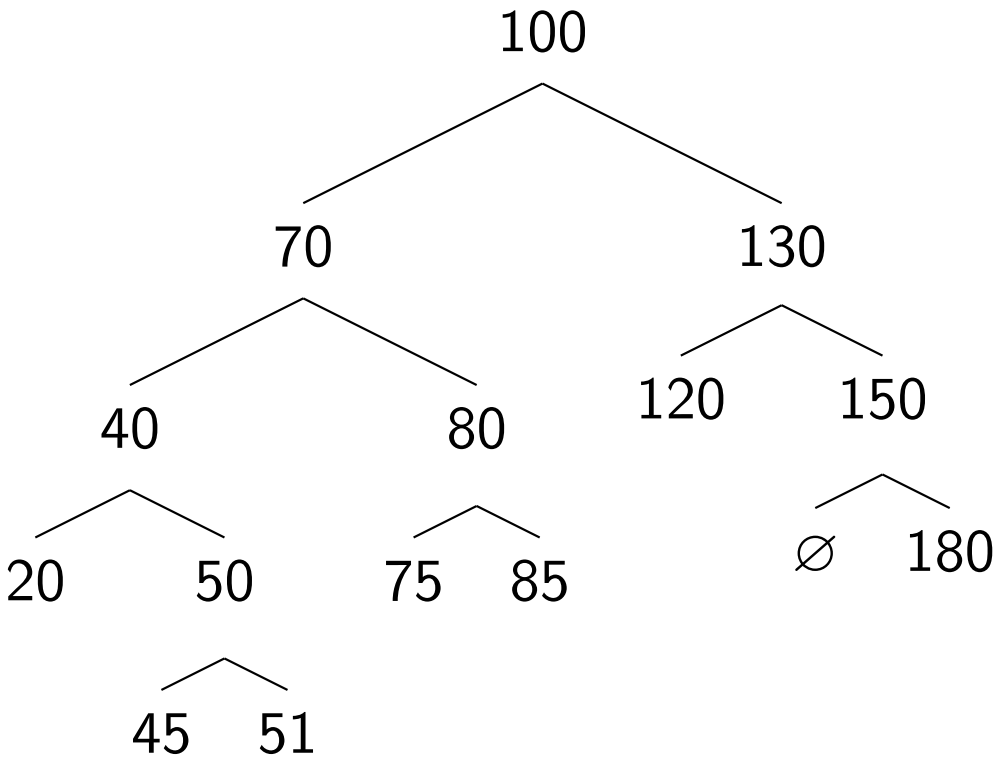
Splay Tree

Want to find 130:



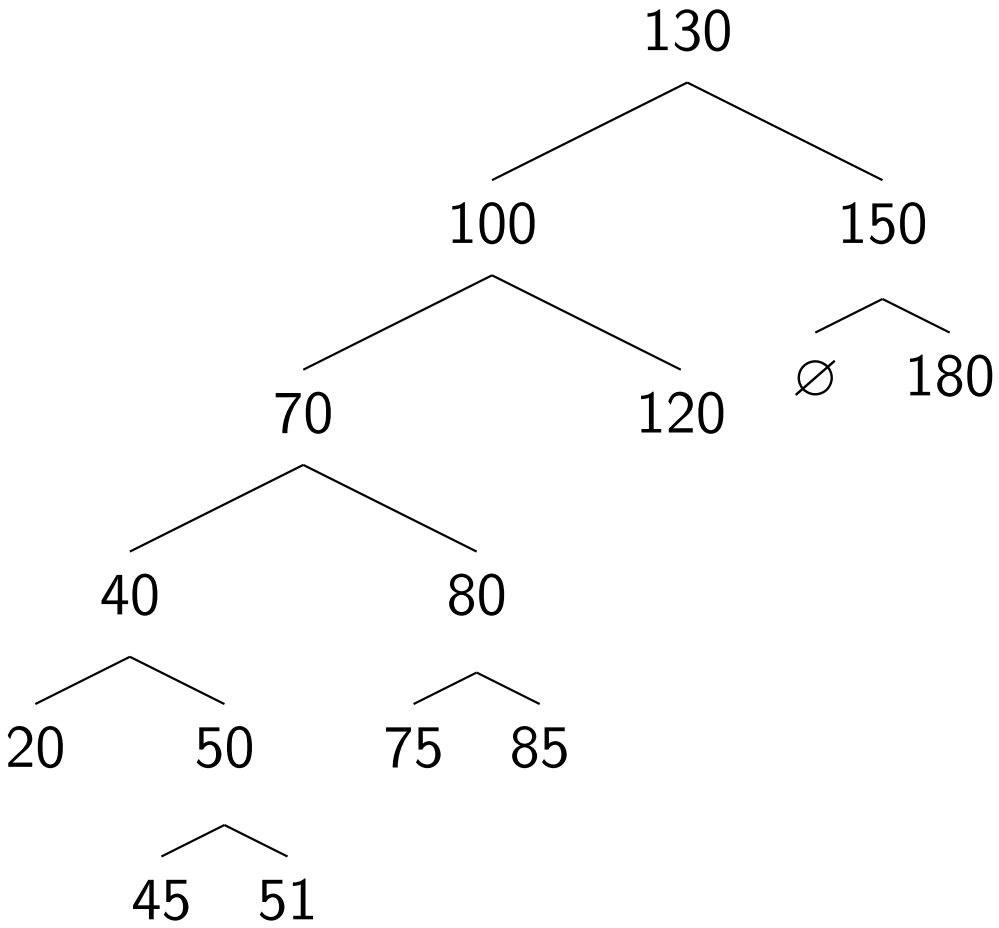
Trees

Splay Tree



Trees

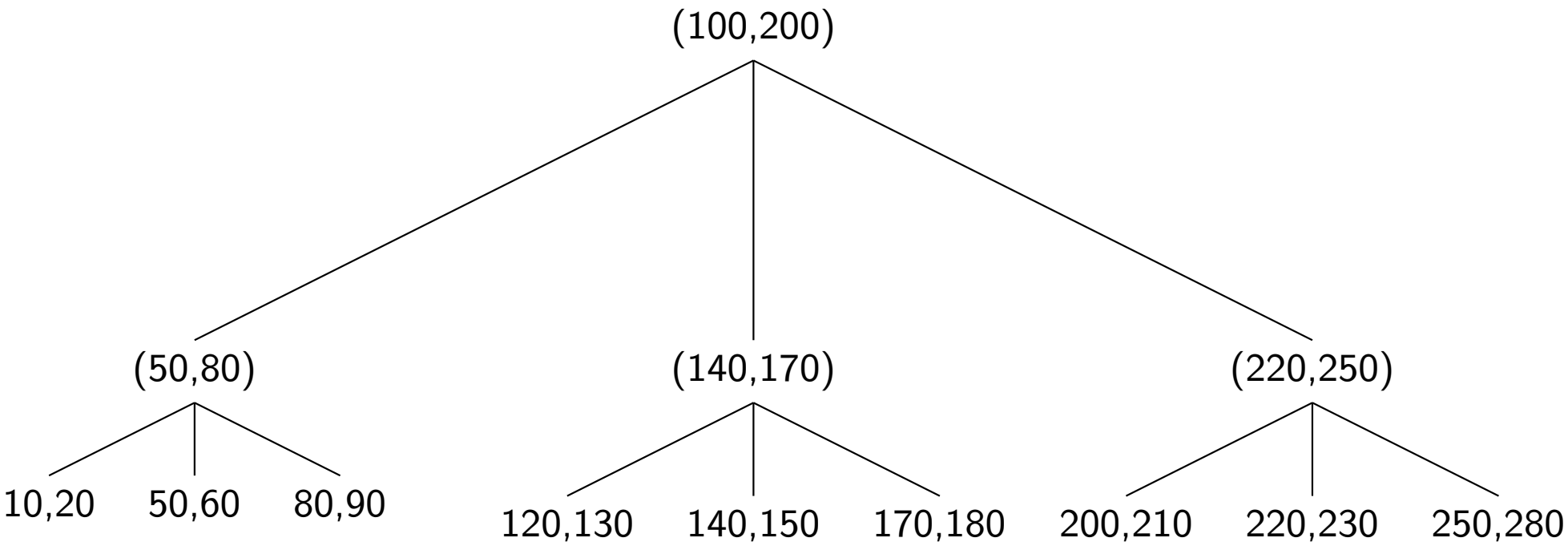
Splay Tree



Trees

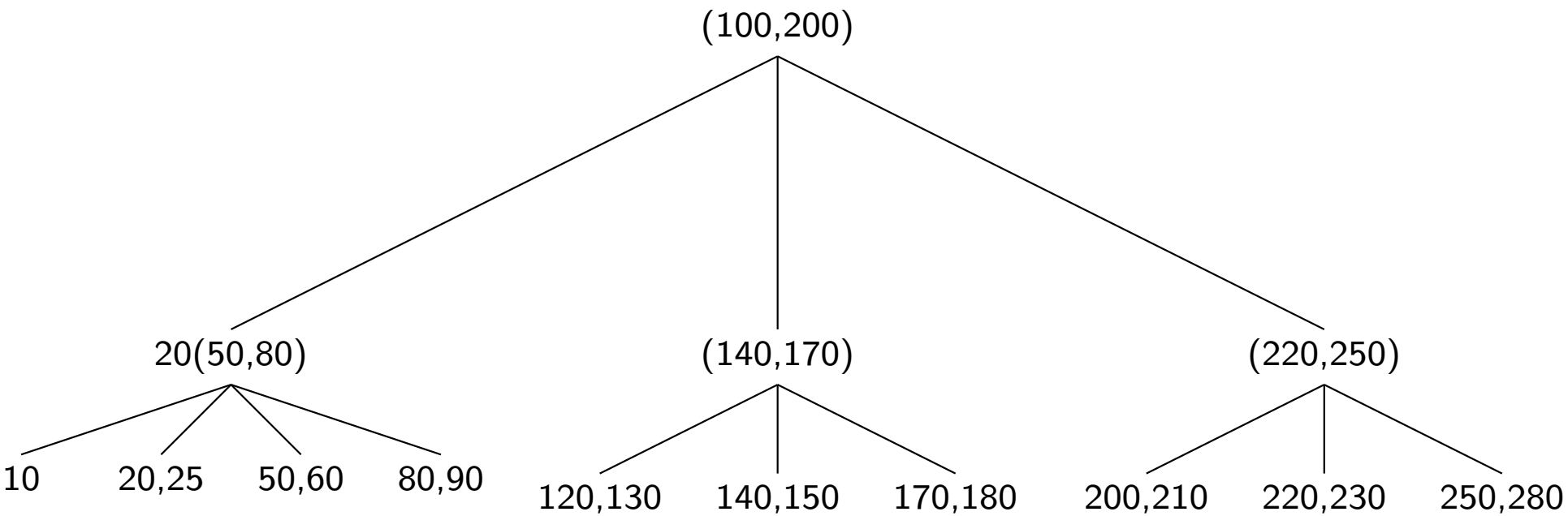
B+ Tree

Want to insert 25



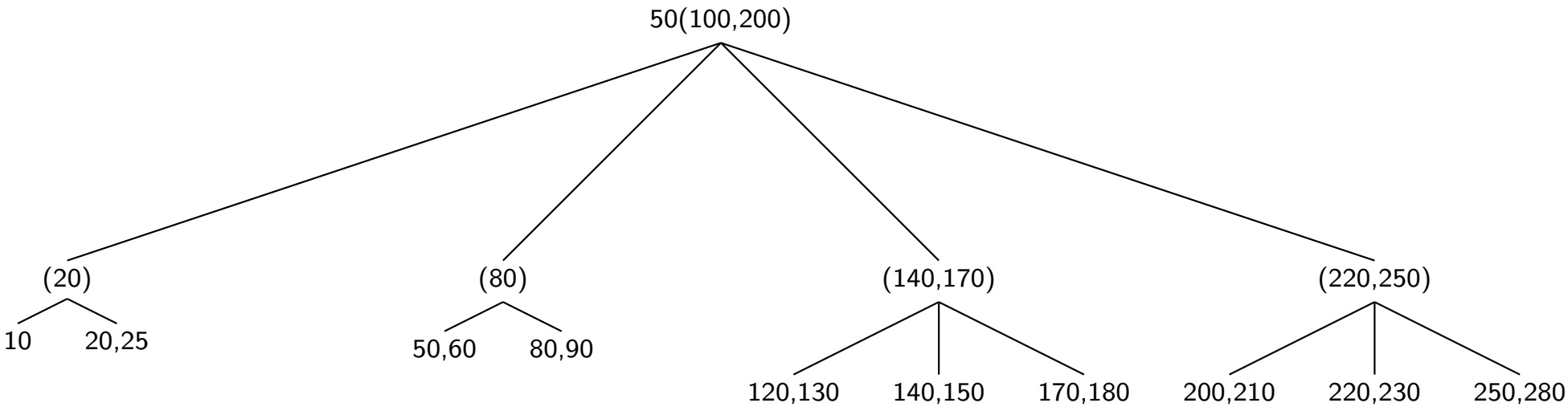
Trees

B+ Tree


(100,200)
20(50,80)
(140,170)
(220,250)
10
20,25
50,60
80,90
120,130
140,150
170,180
200,210
220,230
250,280

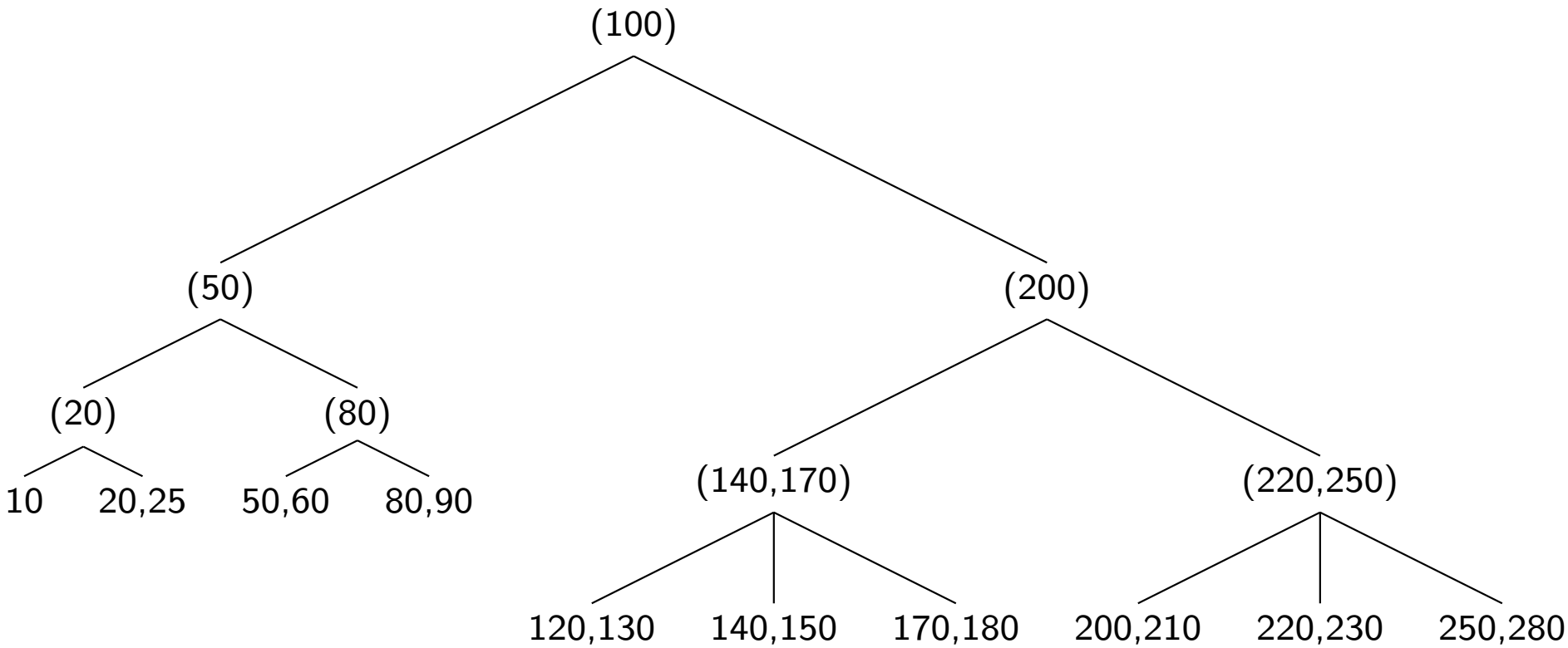
Trees

B+ Tree


50(100,200)
(20)
(80)
(140,170)
(220,250)
10
20,25
50,60
80,90
120,130
140,150
170,180
200,210
220,230
250,280

Trees

B+ Tree


(100)
(50)
(200)
(20)
(80)
(140,170)
(220,250)
10
20,25
50,60
80,90
120,130
140,150
170,180
200,210
220,230
250,280

Lists, Stacks, and Queues

What to know

- Be familiar with the basic operations for lists, stacks, and queues.
- Be able to compare algorithms for these operations with respect to sequential and linked implementations.
- Analyze the complexity of recursive and non-recursive procedures dealing with linear structures.

Lists, Stacks, and Queues

Implementing a Stack using a Linked List

```
Node{
    data, next = null;
    Node(d, n){
        this.data = d
        this.next = n
        }
    }
```

Lists, Stacks, and Queues

What to know

```
stack{
    root = null;
    push(n){
        root = new Node(n, root)
    }
    pop(){
        if root != null {
            val = root.data
            root = root.next
            return val
        }
    }
}
```

Lists, Stacks, and Queues

Questions?

Hashing

What to know:

- Be able to answer questions about hashing concepts.
- Be able to answer complexity questions about hashing.

Hashing

Concepts

What is a Hash Table?

Hashing

Concepts

What is a Hash Table? A hash table is an array of a fixed size and a function which maps keys to array indices in constant time.

Hashing

Concepts

What is a Hash Table? A hash table is an array of a fixed size and a function which maps keys to array indices in constant time.

What is Separate Chaining?

Hashing

Concepts

What is a Hash Table? A hash table is an array of a fixed size and a function which maps keys to array indices in constant time.

What is Separate Chaining? A collision resolution technique where items which hash to the same value are stored in a list

Hashing

Concepts

What is a Hash Table? A hash table is an array of a fixed size and a function which maps keys to array indices in constant time.

What is Separate Chaining? A collision resolution technique where items which hash to the same value are stored in a list

What is Open Addressing?

Hashing

Concepts

What is a Hash Table? A hash table is an array of a fixed size and a function which maps keys to array indices in constant time.

What is Separate Chaining? A collision resolution technique where items which hash to the same value are stored in a list

What is Open Addressing? Open Addressing is any of a number collision resolution methods for hash tables. We have seen Linear and Quadratic probing and Double Hashing.

Hashing

Complexity

What is the worst-case complexity of finding a key in a hash table that uses Separate Chaining for collision resolution?

Hashing

Complexity

What is the worst-case complexity of finding a key in a hash table that uses Separate Chaining for collision resolution?

$O(M)$ for M =the number of items stored in the table.

Hashing

Complexity

What is the worst-case complexity of finding a key in a hash table that uses Separate Chaining for collision resolution?

$O(M)$ for M =the number of items stored in the table.

What is the worst-case complexity of inserting a key in a hash table that uses open addressing for collision resolution?

Hashing

Complexity

What is the worst-case complexity of finding a key in a hash table that uses Separate Chaining for collision resolution?

$O(M)$ for M =the number of items stored in the table.

What is the worst-case complexity of inserting a key in a hash table that uses open addressing for collision resolution?

$O(\mathit{Tablesize})$

Hashing

Questions?

Complexity

What to know:

- Analyze an algorithm: Determine the number of statements executed for a given n
- Big O Notation: Convert your analysis to Big O notation
- Recursive or iterative
- Compare the complexities of various standard algorithms

Complexity

Big O: What is it? How do we use it?

FORMAL DEFINITION $T(N) = O(f(N))$ if there are positive constants c and n_0 such that $T(N) \leq cf(N)$ when $N \geq n_0$

What kind of f(N) are of interest?

- $\log \log N$
- $\log N$
- N
- N^2
- N^k
- 2^N

Complexity

Big O

To prove that a given function has a particular Big O, you must find the constants c and n_0 such that $T(N) \leq cf(N)$ when $N \geq n_0$

As an example, consider the function $g(x) = 4x^4 + 1000x^3 + 150$.
First: what is the Big O of $g(x)$?

Complexity

Big O

To prove that a given function has a particular Big O, you must find the constants c and n_0 such that $T(N) \leq cf(N)$ when $N \geq n_0$

As an example, consider the function $g(x) = 4x^4 + 1000x^3 + 150$.
First: what is the Big O of $g(x)$?

$$g(x) = O(x^4)$$

Complexity

Big O

$\boxed{g(x) = O(x^4)}$

Ok, so what can we use as c and n_0 to prove this?

$$\begin{aligned} g(x) &= 4x^4 + 1000x^3 + 150 \\ &\leq 4x^4 + 1000x^4 + 150x^4 \\ &= 1154x^4 \end{aligned}$$

Therefore, if we set $c = 1154$ and $n_0 = 1$, we have shown that $g(x) \leq c \cdot x^4$ for all $N \geq n_0$, and hence we have proved that $g(x) = O(x^4)$.

Complexity

Example

What is the worst-case complexity of deleting a node from a binary search tree with N nodes?

Complexity

Example

What is the worst-case complexity of deleting a node from a binary search tree with N nodes?

The deletion operation depends on the height of the tree.

In the worst case, the tree is completely unbalanced, and we aim to delete the bottom node.

The deletion operation is $O(N)$.

Complexity

Questions?