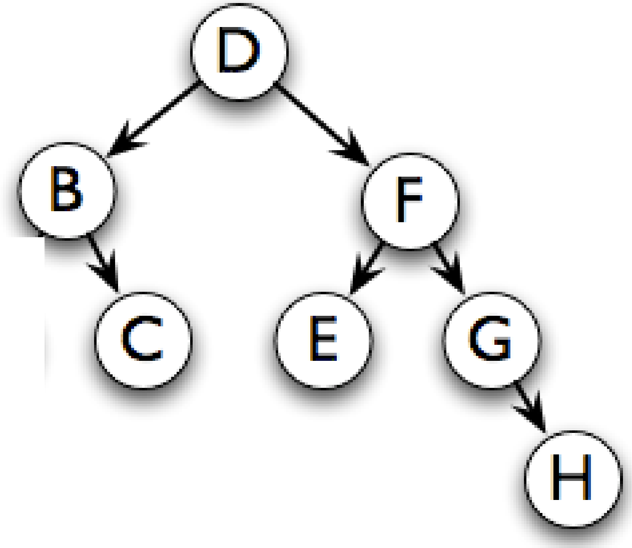


CSSE 230 Day 12

Height-Balanced Trees

After today, you should be able to…
…give the minimum number of nodes in a height-balanced tree
…explain why the height of a height-balanced trees is O(log n)
…help write an induction proof

Today's Agenda

- Announcements
  - Can voice preferences for partners for the term project (groups of 3, starting next Tuesday)
    - EditorTrees partner preference survey on Moodle
      - Balanced with experience level
    - If you don't reply by Monday afternoon, no problem; I'll assign you.

- Another induction example
- Recap: The need for balanced trees
- Analysis of worst case for height-balanced (AVL) trees

Another induction example (we'll use this result) Q1

- Recall our definition of the Fibonacci numbers:
  - $F_0 = 0, F_1 = 1, F_{n+2} = F_{n+1} + F_n$
- An exercise from the textbook

7.8 Prove by induction the formula

$$F_N = \frac{1}{\sqrt{5}}\left(\left(\frac{(1+\sqrt{5})}{2}\right)^N - \left(\frac{1-\sqrt{5}}{2}\right)^N\right)$$

Recall: How to show that property P(n) is true for all $n \geq n_0$:

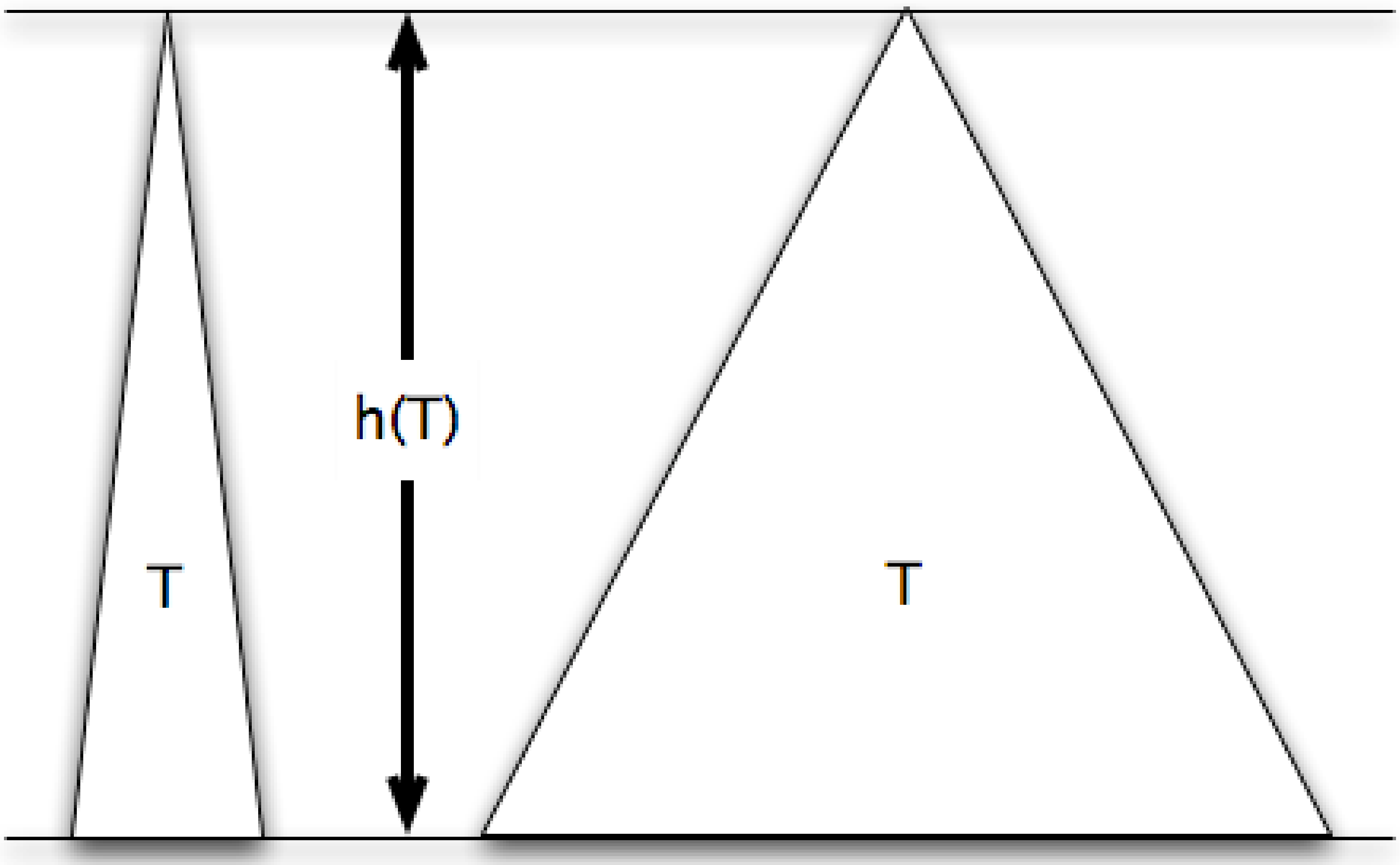
(1) Show the base case(s) directly

(2) Show that if P(j) is true for all j with $n_0 \leq j < k$, then P(k) is true also

Details of step 2:

a. Write down the induction assumption for this specific problem
b. Write down what you need to show
c. Show it, using the induction assumption

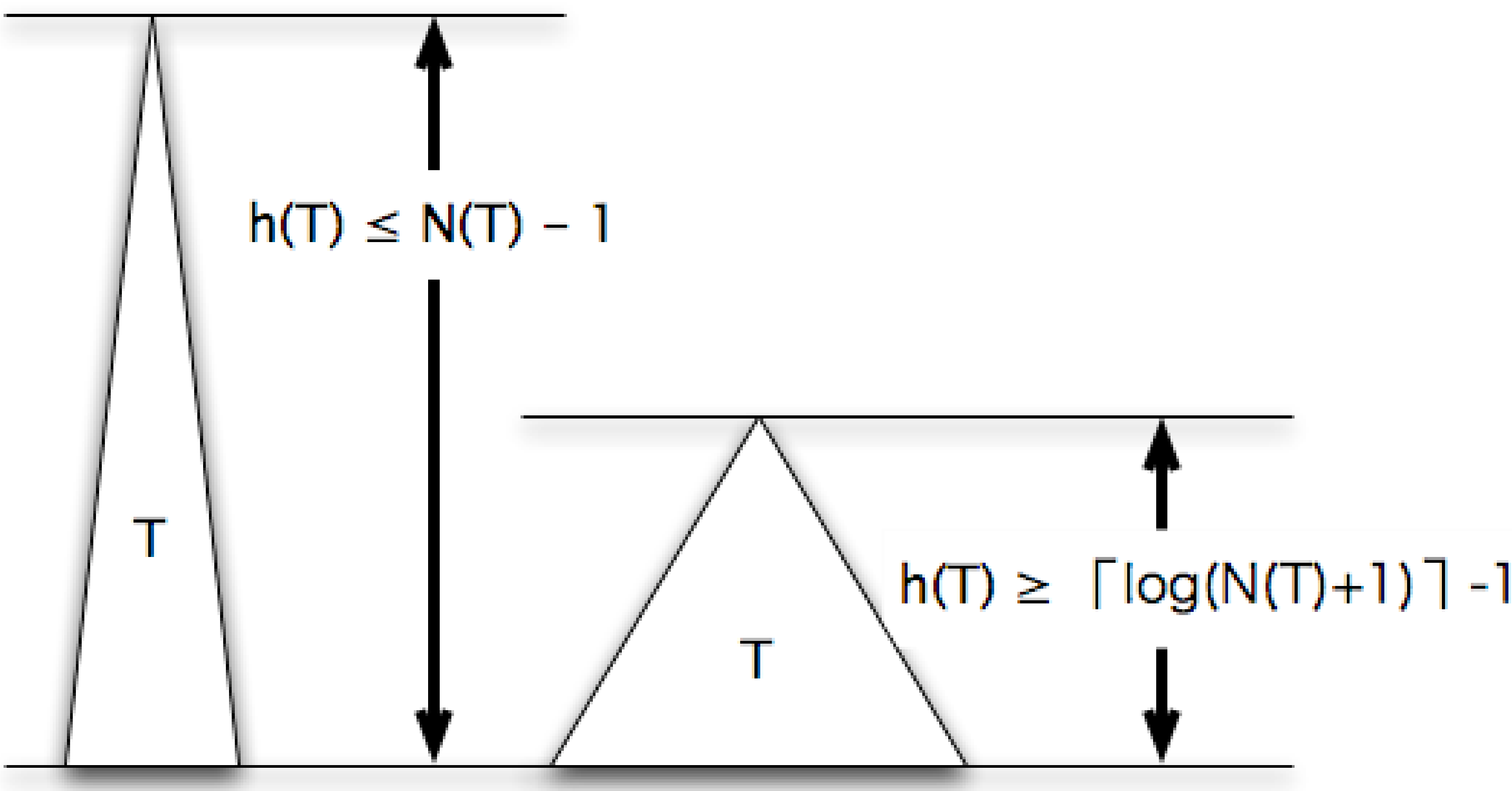
Review: The number of nodes in a tree with height h(T) is bounded



$N(T) \geq h(T) + 1$

$N(T) \leq 2^{h(T)+1} - 1$

Review: Therefore the height of a tree with N(T) nodes is also bounded



We want to keep trees balanced so that the run run time of BST algorithms is minimized

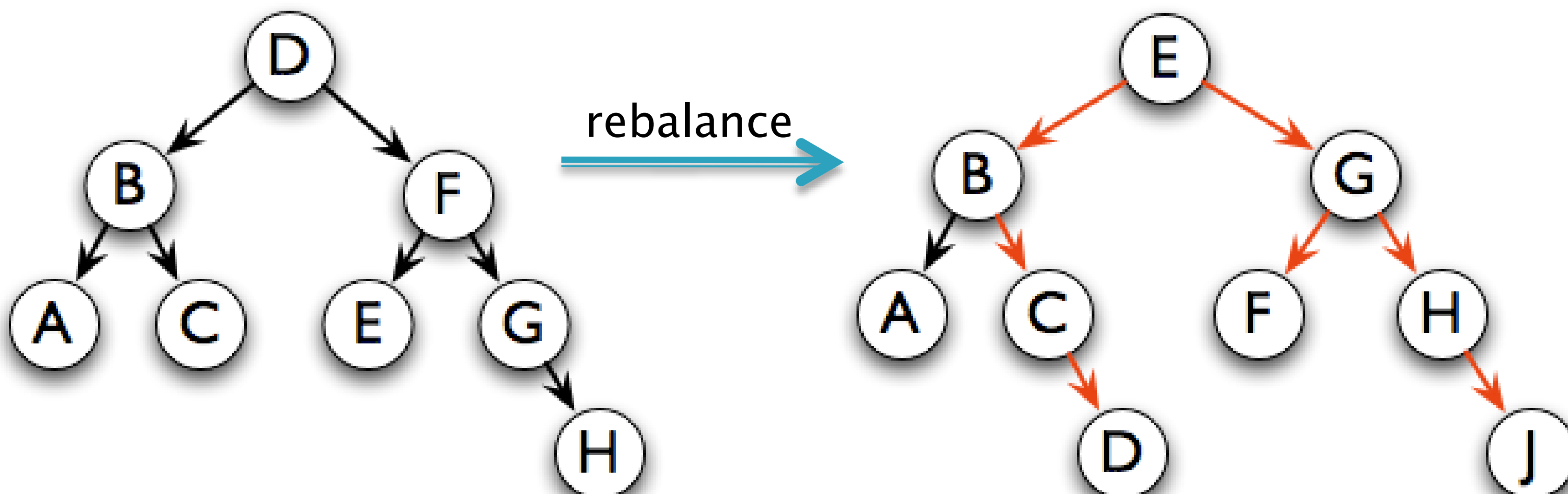
Q2

- BST algorithms are $O(h(T))$
- Minimum value of $h(T)$ is $\lceil \log(N(T)+1) \rceil - 1$
- Can we rearrange the tree after an insertion to guarantee that $h(T)$ is always minimized?

But keeping complete balance is too expensive!

Q3

- Height of the tree can vary from log N to N
- Where would J go in this tree?
- What if we keep the tree perfectly balanced?
  - so height is always proportional to log N
- What does it take to balance that tree?
- Keeping completely balanced is too expensive:
  - O(N) to rebalance after insertion or deletion

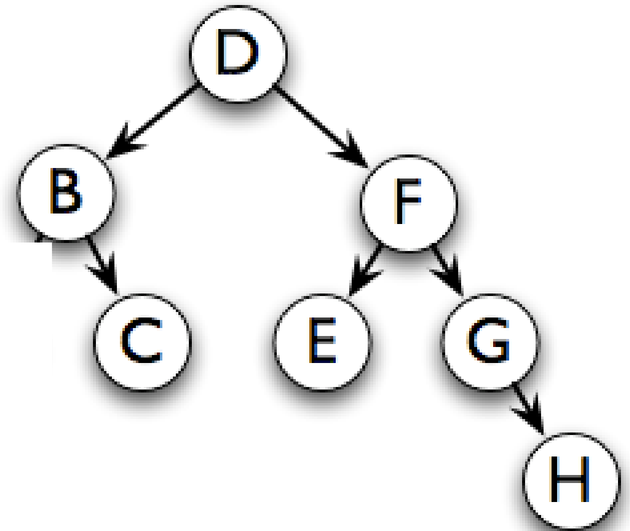


Solution: Height Balanced Trees (less is more)

Height-Balanced Trees have subtrees whose heights differ by at most 1

Q4

Still height-balanced?



More precisely , a binary tree **T** is height balanced if

T is empty, or if

| **height**(T_L) – **height**(T_R) | ≤ 1, and

T_L and T_R are both height balanced.

What is the tallest height-balanced tree with N nodes?

- Is it taller than a completely balanced tree?
  - Consider the dual concept: find the minimum number of nodes for height h.

A binary search tree **T** is height balanced if

T is empty, or if

| **height(** T_L **)** – **height(** T_R **)** | ≤ **1**, and

T_L and T_R are both height balanced.

An AVL tree is a height-balanced BST that maintains balance using "rotations"

- Named for authors of original paper, **A**delson-**V**elskii and **L**andis (1962).

- Max. height of an AVL tree with **N** nodes is:
$H < 1.44 \log (N+2) - 1.328 = O(\log N)$

Our goal is to rebalance an AVL tree after insert/delete in O(log n) time

Q8

- Why?
- Worst cases for BST operations are **O(h(T))**
  - find, insert, and delete
- **h(T)** can vary from **O(log N)** to **O(N)**
- Height of a height-balanced tree is **O(log N)**
- So if we can rebalance after insert or delete in **O(log N)**, then **all** operations are **O(log N)**