ECE430.217 Data Structures

Balanced Trees

Weiss Book Chapter 4.4/4.5

Byoungyoung Lee
https://compsec.snu.ac.kr
byoungyoung@snu.ac.kr

Lecture slides were prepared based on Douglas W. Harder's notes (**dwharder@alumni.uwaterloo.ca).**

Outline

In this topic, we will:

- Introduce the idea of ***balance***
- We will introduce a few examples

Background

Run times depend on the height of the trees

As was noted in the previous section:

- The best case height is Q1
- The worst case height is Q2
- The average case is Q3
  - However, following random insertions and erases, the average height tends to increase to Q4
    - Check more in Weiss $4.3.6 or CLRS $12.4

CLSR $12.4

Unfortunately, little is known about the average height of a binary search tree when both insertion and deletion are used to create it. When the tree is created by insertion alone, the analysis becomes more tractable. Let us therefore define a ***randomly built binary search tree*** on n keys as one that arises from inserting the keys in random order into an initially empty tree, where each of the $n!$ permutations of the input keys is equally likely. (Exercise 12.4-3 asks you to show that this notion is different from assuming that every binary search tree on n keys is equally likely.) In this section, we shall prove the following theorem.

Theorem 12.4

The expected height of a randomly built binary search tree on n distinct keys is $O(\lg n)$.

Requirement for Balance

We want to ensure that the run times never fall into $\omega(\ln(n))$

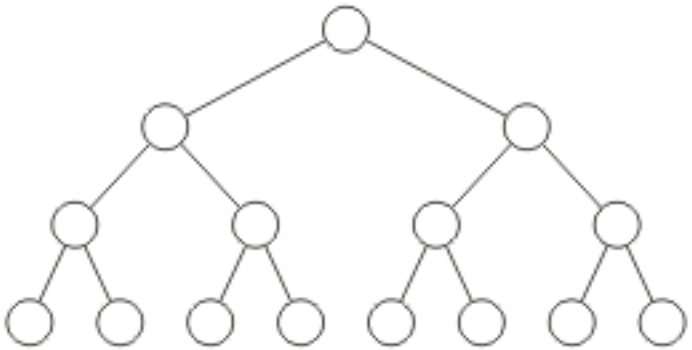
Requirement:

- We must maintain a height which is $\Theta(\ln(n))$

To do this, we will define an idea of balance

Examples

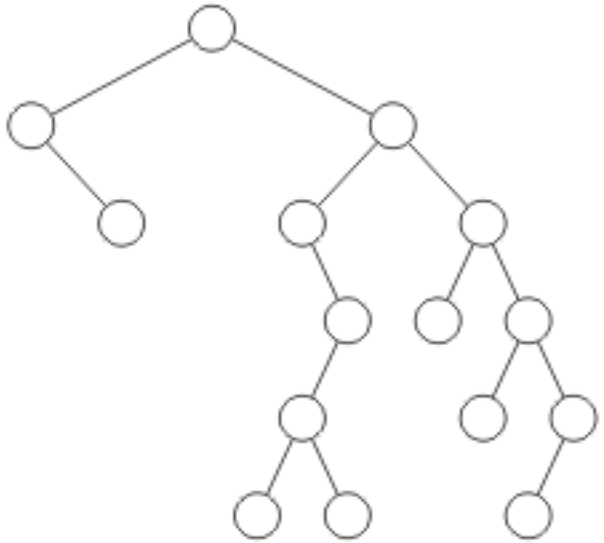
For a perfect tree, all nodes have the same number of descendants on each side



Perfect binary trees are balanced while linked lists are not

Examples

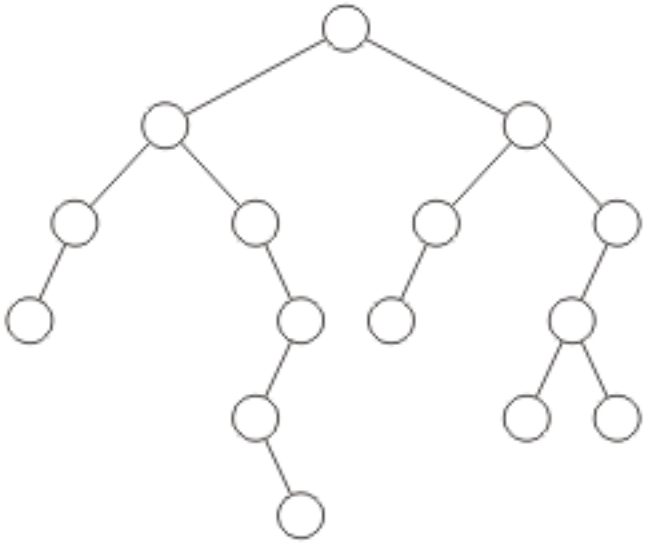
This binary tree would also probably not be considered to be “balanced” at the root node



Examples

How about this example?

- The root seems balanced, but what about the left sub-tree?



Definition for Balance

We need **a quantitative definition of *balance***

"Balanced" may be defined by:

- ***Height balancing***: comparing the heights of the two sub trees
- ***Null-path-length balancing***: comparing the null-path-length of each of the two sub-trees
- ***Weight balancing***: comparing the number of null sub-trees in each of the two sub trees

We will have to mathematically prove that if a tree satisfies the definition of balance, its height is $\Theta(\ln(n))$

Balanced Trees

Height balancing:

- AVL trees

Null-path-length balancing

- Red-Black Trees

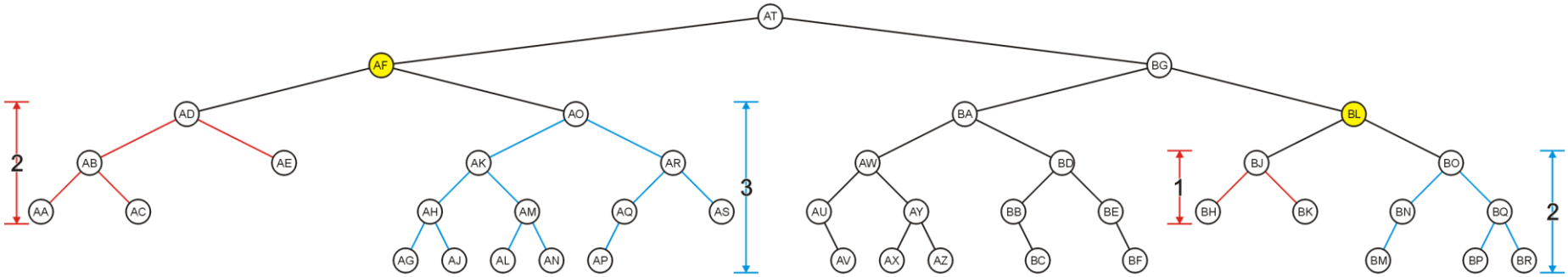
Weight-Balanced Trees

- BB Trees

AVL Trees

A node is AVL balanced

- if two sub-trees differ in height by at most one



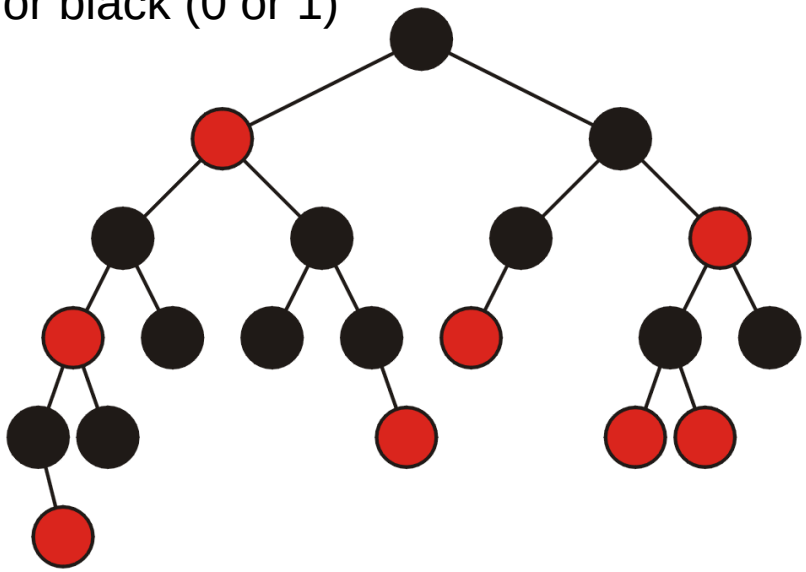
Red-Black Trees

Red-black trees maintain balance by

- All nodes are *colored* red or black (0 or 1)

Requirements:

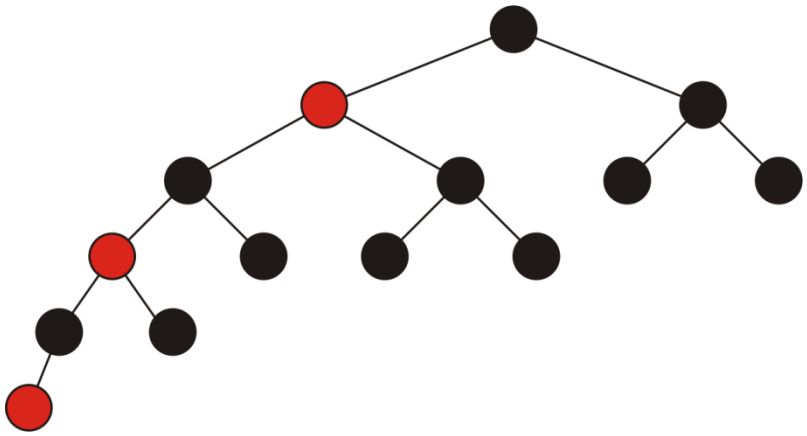
- The root must be black
- All children of a red node must be black
- Any path from the root to an empty node must have the same number of black nodes



Red-Black Trees

Red-black trees are **null-path-length balanced** in that the null-path length going through one sub-tree must not be greater than twice the null-path length going through the other

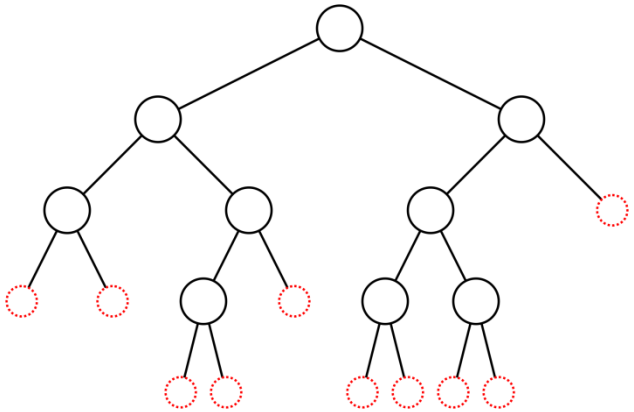
- For all path
  - # of black nodes >= # of red nodes



Weight-Balanced Trees

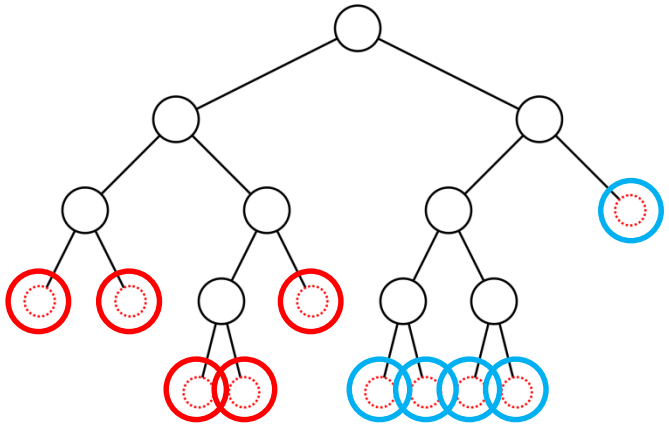
Recall: an empty node/null subtree is any position within a binary tree that could be filled with the next insertion:

- This tree has 9 nodes and 10 empty nodes:



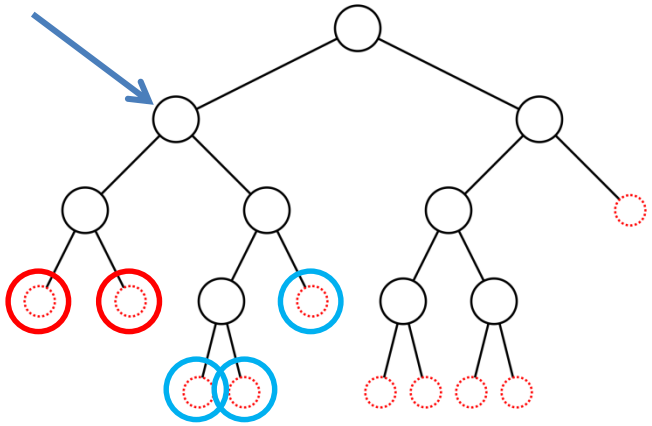
Weight-Balanced Trees

The ratios of the empty nodes at the root node are 5/10 and 5/10



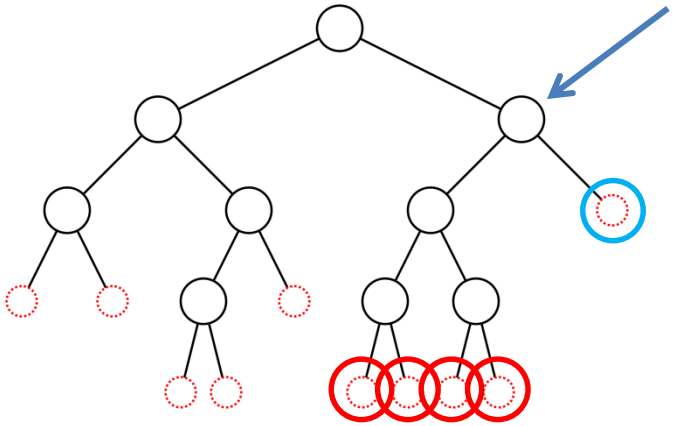
Weight-Balanced Trees

The ratios of the empty nodes at this node are 2/5 and 3/5



Weight-Balanced Trees

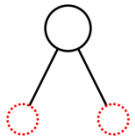
The ratios of the empty nodes at this node, however, are 4/5 and 1/5



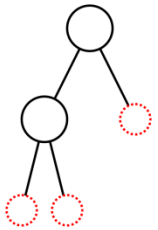
Weight-Balanced Trees

BB(α) trees ($0 < \alpha \leq 1/3$) maintain weight balance requiring that neither side has less than a α proportion of the empty nodes, *i.e.*, both proportions fall in $[\alpha, 1 - \alpha]$

- With one node, both are 0.5



- With two, the proportions are $1/3$ and $2/3$



Summary

In this talk, we introduced the idea of *balance*

- We require $\Theta(\ln(n))$ run times
- Balance will ensure the height is $\Theta(\ln(n))$

There are numerous definitions:

- AVL trees use height balancing
- Red-black trees use null-path-length balancing
- BB(a) trees use weight balancing

References

Blieberger, J., *Discrete Loops and Worst Case Performance*, Computer Languages, Vol. 20, No. 3, pp.193-212, 1994.