Lecture 9. Binary Search Trees (Section 11.2.2)

Example 9.1. The custom dictionary in a spell-checker keeps track of words that you don't want flagged as misspellings but aren't in the standard dictionary. These words get added one at a time, in no particular order, but it is necessary to keep them organized so that searching the list is easy. Suppose your custom dictionary contains the following words:

macchiato, poset, phat, complexify, jazzed, sheafify, clueless

What is an efficient way to organize this data?

Example 9.1. The custom dictionary in a spell-checker keeps track of words that you don't want flagged as misspellings but aren't in the standard dictionary. These words get added one at a time, in no particular order, but it is necessary to keep them organized so that searching the list is easy. Suppose your custom dictionary contains the following words:

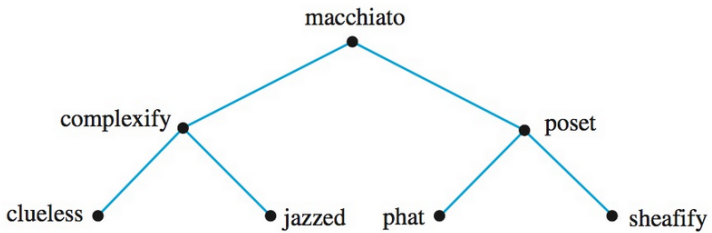
macchiato, poset, phat, complexify, jazzed, sheafify, clueless

What is an efficient way to organize this data?

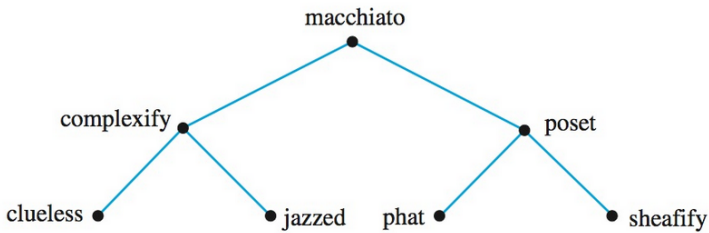
One possible organizational structure is a graph model called a binary search tree.

- Start with an item (chosen arbitrarily) at the top of the tree (root node).
- The root has (at most) two edges touching it, one going down to the right, and one going down to the left.
- The only condition is that the right child must come after its parent in alphabetical (or numerical) order, and the left child must come before its parent.

- The binary tree for the question is



- The binary tree for the question is



- For large data sets, the searching process in a binary tree structure goes very quickly since you don't need to look at every element. For example, to find "poset" in the tree, we only have to look at two words. To establish that "iPod" is not in the tree, we only need to check three words.

Each comparison moves the search one level down the tree, and each level containes twice the number of elements as the level before. For example, a balanced binary search tree within

$$255 = 1 + 2 + 4 + 8 + 16 + 32 + 64 + 128$$

elements requires, at most eight comparisons to search it completely.

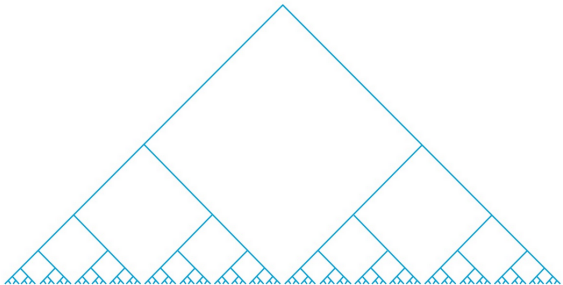


Figure 2.7 A balanced binary search tree with 255 nodes requires, at most, eight comparisons to search it completely.

Practice 9.2. Consider the following list of numbers.

$$123, 684, 121, 511, 602, 50, 43, 10, 320$$

(a) Place the list of numbers, in the order given, into a binary search tree. (The organizing condition is that the right child must come after its parent in numerical order, and the left child must come before its parent.)

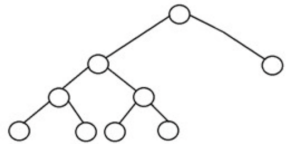
(b) What is the height of the tree in part (a)?

(c) Reorder the numbers[1] so that when they are put into a binary search tree, the height of the resulting tree is less than the height of the tree in part (a). Give both your new list and the search tree it produces.

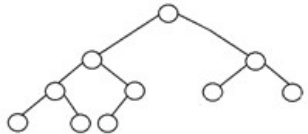
[1] That is, choose a different number as a root.

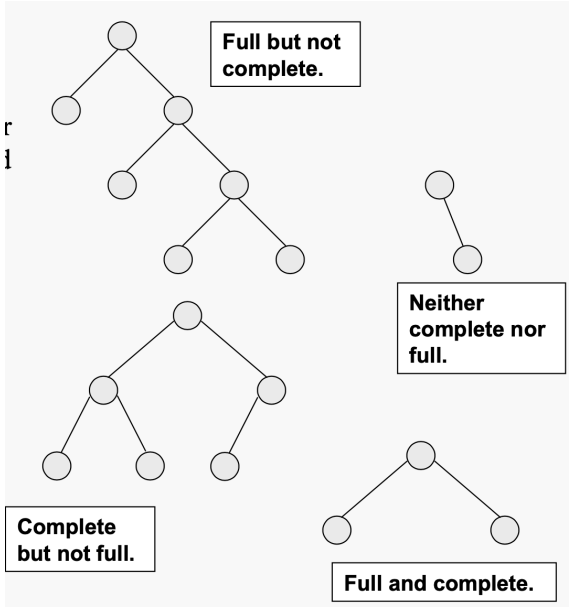
Full and Complete Binary Trees

Definition 9.3. A full binary tree is a tree in which every internal vertex has exactly two children.



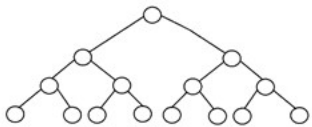
Definition 9.4. A complete binary tree is a binary tree in which every level, except possibly the last, is completely filled, and the last level has all its vertices to the left side.




Full but not complete.
Neither complete nor full.
Complete but not full.
Full and complete.

Perfect Binary Trees

Definition 9.5. A perfect binary tree is a binary tree in which all the internal vertices have strictly two children and every leaf is at the same level or same depth within a tree. (i.e., a perfect tree = a complete full tree)



Theorem 9.6. The number of vertices, $n(T)$, of a perfect binary tree having depth (or height) h is $2^{h+1} - 1$:

$$n(T) = 2^{h+1} - 1$$

(Hint: $1, 2, 4, ..., 2^h$ is a geometric sequence. The sum of finite geometric series is $S_n = \frac{a_0(r^n-1)}{r-1}$ with $|r| > 1$, where n is the number of terms, a_0 is the first term and $r \neq 1$ is the common ratio.)

How many items can a binary tree hold?

How many items can a binary tree hold? Since a perfect binary tree of depth h has $2^{h+1} - 1$ nodes,

Theorem 9.7 The number of items that a binary tree (complete or not) can store is at most $2^{h+1} - 1$. That is,

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

where $n(T) =$ the number of vertices on a binary tree T and $h(T) =$ the height of T.]pause

Exercise 9.8. A binary tree T_1 has 100 vertices. Find the minimum and maximum possible heights of this tree.

Exercise 9.9. A second binary tree T_2 has height $h(T_2) = 7$ and contains exactly 90 vertices. Determine whether T_2 can be a complete binary tree. Justify your answer.