Ch 11. Trees

Trees

Sungwon Kang

Acknowledgement
- [Rosen 19] Kenneth H. Rosen, for Discrete Mathematics & Its Applications (8th Edition), Lecture slides
- [Hunter 11] David J. Hunter, Essentials of Discrete Mathematics, 2nd Edition, Jones & Bartlett Publishers, 2011, Lecture Slides

Trees

1. Lists
2. Trees
3. Efficiency

1. Lists

Recursively defined **data structure**

Definition

Let X be a set. A *list* of elements of X is

B. x where $x \in X$.

R. L, x where $x \in X$ and L is a list of elements of X.

Constructing lists, bottom-up:

$$
\begin{array}{lllll}
L_1 & = & \texttt{cubs} & & \text{by part } \mathbf{B} \\
L_2 & = & L_1, \texttt{bears} & = \texttt{cubs}, \texttt{bears} & \text{by part } \mathbf{R} \\
L_3 & = & L_2, \texttt{bulls} & = \texttt{cubs}, \texttt{bears}, \texttt{bulls} & \text{by part } \mathbf{R} \\
L_4 & = & L_3, \texttt{cubs} & = \texttt{cubs}, \texttt{bears}, \texttt{bulls}, \texttt{cubs} & \text{by part } \mathbf{R}
\end{array}
$$

What are the differences between Array and List?

Another list definition: A sorted list

Another example of recursively defined data structure

Definition

An *SList* is

B. x where $x \in \mathbf{R}$, the real numbers.

R. (X, Y) where X and Y are SLists having the same number of elements, and the last number in X is less than the first number in Y.

For example, $(((1,3),(8,9)),((12,16),(25,30)))$ is an SList of depth 3.

What are the first number and the last number of ((a,b),(c,d))?

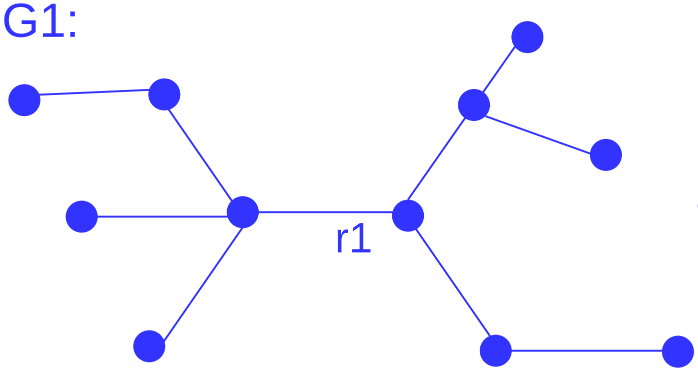
2. Trees

A tree is a connected graph with no simple cycles that has a node designated as the root.

Definition

A *tree* is a graph T with a specified vertex r, called the *root*, with the property that, for any vertex v in T $(v \neq r)$, there is a unique simple path from r to v.

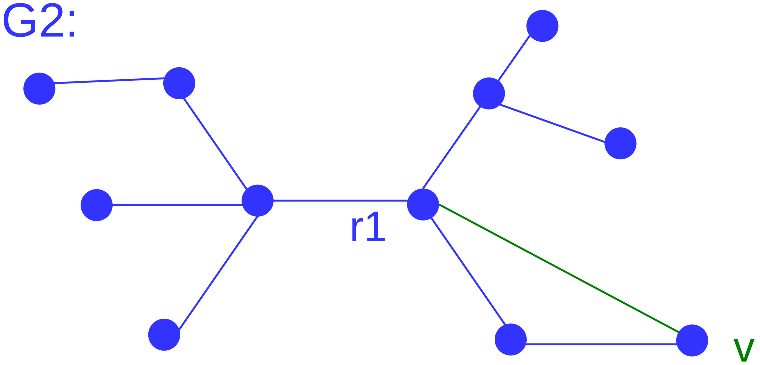
G1:



Yes

trees?

G2:



No

Three important tree theorems

Theorem

Let G be an undirected graph, and let $r \in G$. Then G is a tree with root r if and only if G is connected and has no simple circuits.

Corollary

In an undirected tree, there is a unique simple path between any two vertices in the tree.

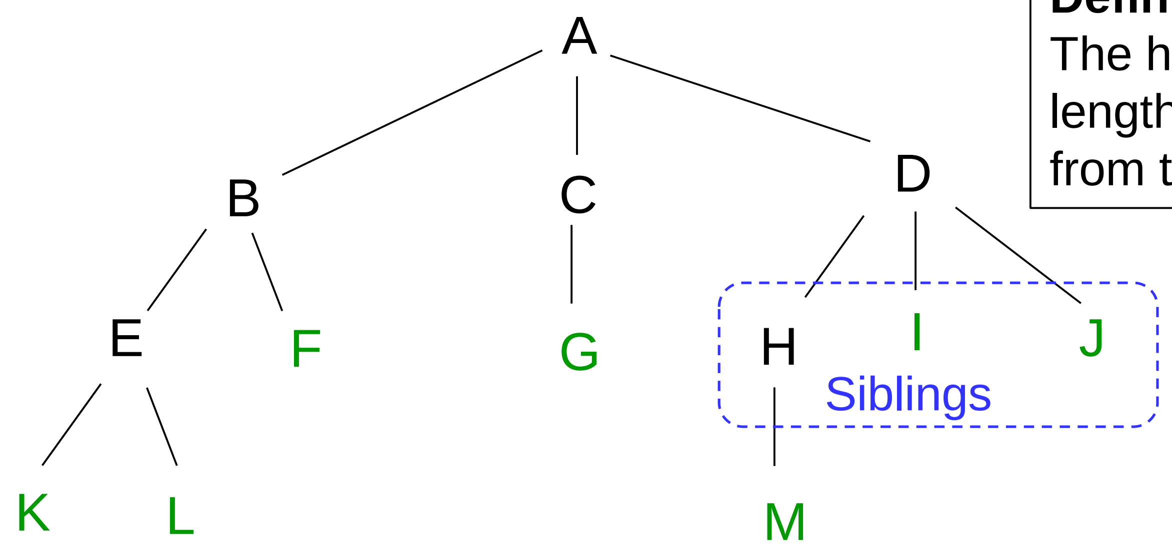
Theorem

Let T be a tree with n vertices. Then T has $n - 1$ edges.

For proofs, see the text.

Tree – Alternative Definition

- A collection of elements called nodes,
  one of which is distinguished as a root, along with
  a relation ("parenthood") that places a hierarchical structure on the nodes.



Definition (Height of a tree)
The height of a tree is the length of the longest path from the root to a leaf node.

Terminology: parent, child, sibling, ancestor, leaf node, degree of a tree

See [Horowitz 08] Section 5.1 for more.

Binary Trees

B₁. The empty tree is a binary tree.

Tree: *empty tree* or *null tree*

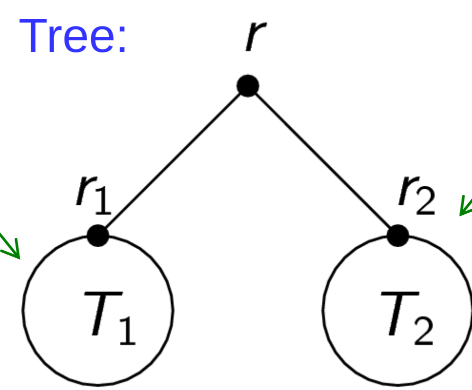
B₂. A single vertex is a binary tree. In this case, the vertex is the root of the tree.

Tree: ● r

R. If T_1 and T_2 are binary trees with roots r_1 and r_2 respectively, then the tree

Tree:

left child of r

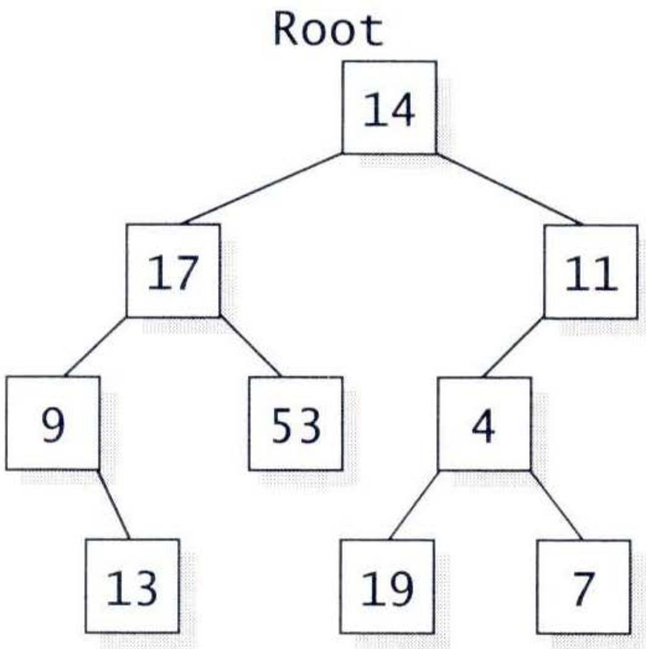


right child of r

vertex = node
edge = link

is a binary tree with root r. Here the circles represent the binary trees T_1 and T_2. If either of these trees T_i $(i = 1, 2)$ is the empty tree, then there is no edge from r to T_i.

A Binary Tree with Integer Nodes

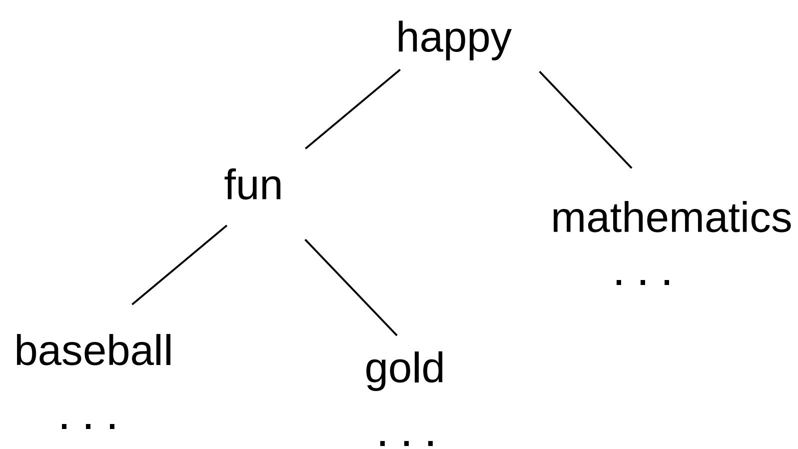


Binary Search Tree

Definition

A **binary search tree** is a binary tree in which, for any node,
its left child, if any, comes alphabetically (or numerically) before it and
its right child, if any, comes alphabetically (or numerically) after it.

Example

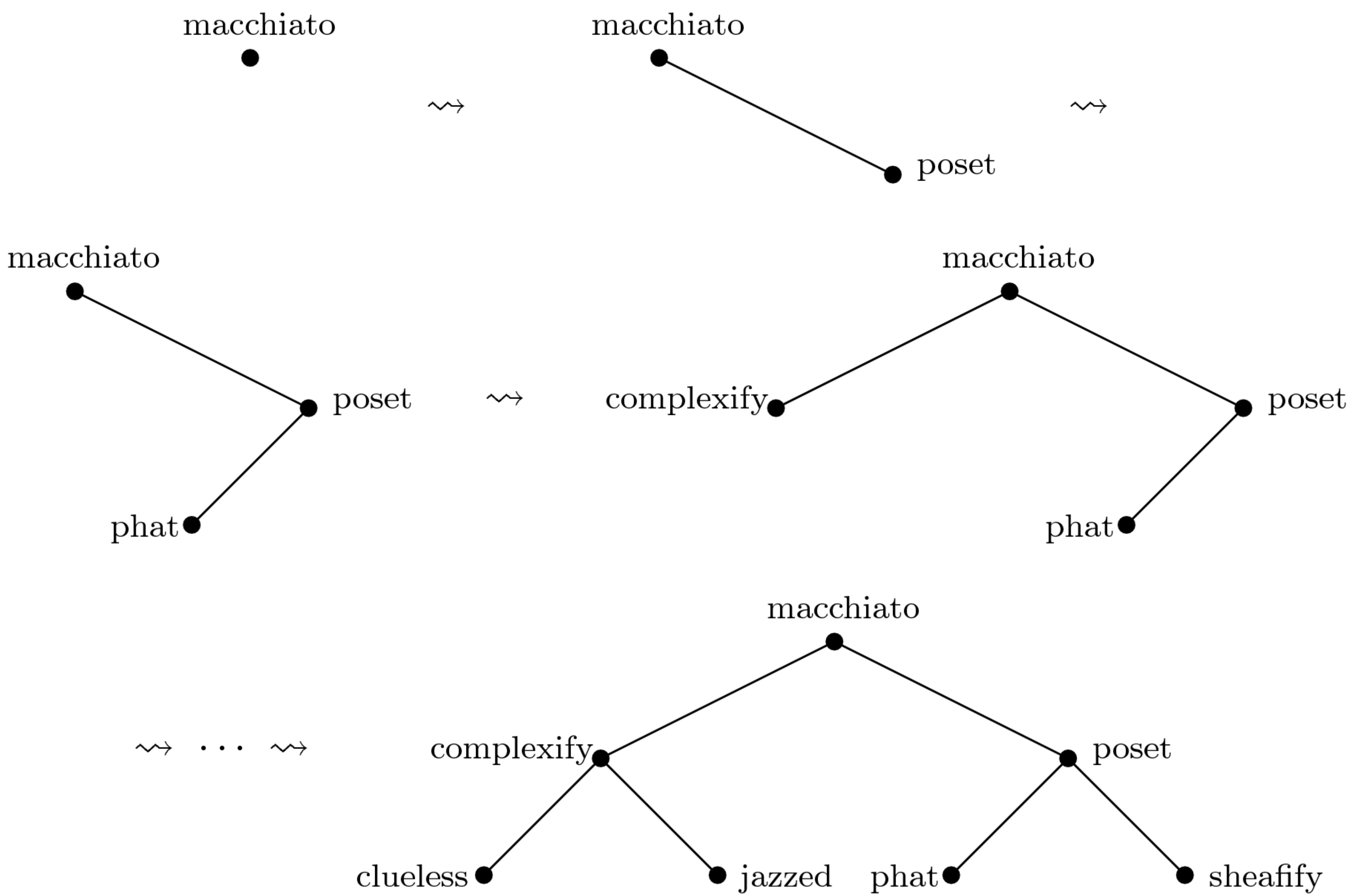


A spell checker needs to organize data efficiently so that:

- It is easy to find data.
- It is easy to add data.

Organize the following data:

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


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

A function to search an SList

Recursively defined function on recursively defined data structure

Definition

Define a true/false function Search(t, L), where t is a number (the "target") and L is an SList, as follows.

[redacted]

Recursively defined function on recursively defined data structure

Definition

Define a true/false function Search(t, L), where t is a number (the "target") and L is an SList, as follows.

B. Suppose $L = x$, a list of depth 0. Then

$$\text{Search}(t, L) = \begin{cases} \text{true} & \text{if } t = x. \\ \text{false} & \text{if } t \neq x. \end{cases}$$

A recursively defined function on a recursively defined data structure

Definition

Define a true/false function Search(t, L), where t is a number (the "target") and L is an SList, as follows.

B. Suppose $L = x$, a list of depth 0. Then

$$\text{Search}(t, L) = \begin{cases} \text{true} & \text{if } t = x. \\ \text{false} & \text{if } t \neq x. \end{cases}$$

R. Suppose the depth of L is greater than 0, so $L = (X, Y)$. Then

$$\text{Search}(t, L) = \text{Search}(t, X) \vee \text{Search}(t, Y).$$

.

Evaluating Search, top-down

Let $L = (((1,3),(8,9)),((12,16),(25,30)))$.

$$\text{Search}[8, L]$$

[redacted]

Let $L = (((1,3),(8,9)),((12,16),(25,30)))$.

$$
\begin{aligned}
& \mathrm{Search}[8, L] \\
= \quad & \mathrm{Search}[8, ((1,3),(8,9))] \vee \mathrm{Search}[8, ((12,16),(25,30))]
\end{aligned}
$$

Let $L = (((1,3),(8,9)),((12,16),(25,30)))$.

$$
\begin{aligned}
& \text{Search}[8, L] \\
= \quad & \text{Search}[8, ((1,3),(8,9))] \vee \text{Search}[8, ((12,16),(25,30))] \\
= \quad & \text{Search}[8, (1,3)] \vee \text{Search}[8, (8,9)] \\
& \vee \text{Search}[8, (12,16)] \vee \text{Search}[8, (25,30)]
\end{aligned}
$$

[redacted]

Let $L = (((1,3),(8,9)),((12,16),(25,30)))$.

$$
\begin{aligned}
& \text{Search}[8, L] \\
= \ & \text{Search}[8, ((1,3),(8,9))] \vee \text{Search}[8, ((12,16),(25,30))] \\
= \ & \text{Search}[8, (1,3)] \vee \text{Search}[8, (8,9)] \\
& \vee \text{Search}[8, (12,16)] \vee \text{Search}[8, (25,30)] \\
= \ & \text{Search}[8, 1] \vee \text{Search}[8, 3] \vee \text{Search}[8, 8] \vee \text{Search}[8, 9] \\
& \vee \text{Search}[8, 12] \vee \text{Search}[8, 16] \vee \text{Search}[8, 25] \vee \text{Search}[8, 30] \\
= \ & \text{false} \vee \text{false} \vee \underline{\text{true}} \vee \text{false} \vee \text{false} \vee \text{false} \vee \text{false} \vee \text{false} \\
= \ & \underline{\text{true}}
\end{aligned}
$$

Is this an efficient algorithm?

Another search function: BSearch

Definition

Define a true/false function BSearch(t, L), where t is a number and L is an SList, as follows.

i.e. a target number

Definition

Define a true/false function BSearch(t, L), where t is a number and L is an SList, as follows.

B. Suppose $L = x$, a list of depth 0. Then

$$\text{BSearch}(t, L) = \begin{cases} \text{true} & \text{if } t = x. \\ \text{false} & \text{if } t \neq x. \end{cases}$$

Definition

Define a true/false function BSearch(t, L), where t is a number and L is an SList, as follows.

B. Suppose $L = x$, a list of depth 0. Then

$$\text{BSearch}(t, L) = \begin{cases} \text{true} & \text{if } t = x. \\ \text{false} & \text{if } t \neq x. \end{cases}$$

R. Suppose L has depth $p > 0$, so $L = (X, Y)$. Let r be the last element of X. Then

$$\text{BSearch}(t, L) = \begin{cases} \text{BSearch}(t, Y) & \text{if } t > r. \\ \text{BSearch}(t, X) & \text{if } t \not> r. \end{cases}$$

Evaluating BSearch: top-down

Let $L = (((1,3),(8,9)),((12,16),(25,30)))$.

$$\text{BSearch}[8, L] \quad =$$

Let $L = (((1,3),(8,9)),((12,16),(25,30)))$.

$$\text{BSearch}[8, L] \quad = \quad \text{BSearch}[8, ((1,3),(8,9))] \quad \text{since } 8 \not> 9$$

[redacted]

Let $L = (((1,3),(8,9)),((12,16),(25,30)))$.

$$\begin{aligned}
\text{BSearch}[8, L] &= \text{BSearch}[8, ((1,3),(8,9))] && \text{since } 8 \not> 9 \\
&= \text{BSearch}[8, (8,9)] && \text{since } 8 > 3 \\
&= \text{BSearch}[8, 8] && \text{since } 8 \not> 8 \\
&= \text{true} && \text{since } 8 = 8
\end{aligned}$$

$$
\begin{aligned}
& \mathrm{Search}[8, L] \\
= \; & \mathrm{Search}[8, ((1,3),(8,9))] \vee \mathrm{Search}[8, ((12,16),(25,30))] \\
= \; & \mathrm{Search}[8, (1,3)] \vee \mathrm{Search}[8, (8,9)] \\
& \vee \mathrm{Search}[8, (12,16)] \vee \mathrm{Search}[8, (25,30)] \\
= \; & \mathrm{Search}[8, 1] \vee \mathrm{Search}[8, 3] \vee \mathrm{Search}[8, 8] \vee \mathrm{Search}[8, 9] \\
& \vee \mathrm{Search}[8, 12] \vee \mathrm{Search}[8, 16] \vee \mathrm{Search}[8, 25] \vee \mathrm{Search}[8, 30] \\
= \; & \mathrm{false} \vee \mathrm{false} \vee \underline{\mathrm{true}} \vee \mathrm{false} \vee \mathrm{false} \vee \mathrm{false} \vee \mathrm{false} \vee \mathrm{false} \\
= \; & \underline{\mathrm{true}}
\end{aligned}
$$

Let $L = (((1,3),(8,9)),((12,16),(25,30)))$.

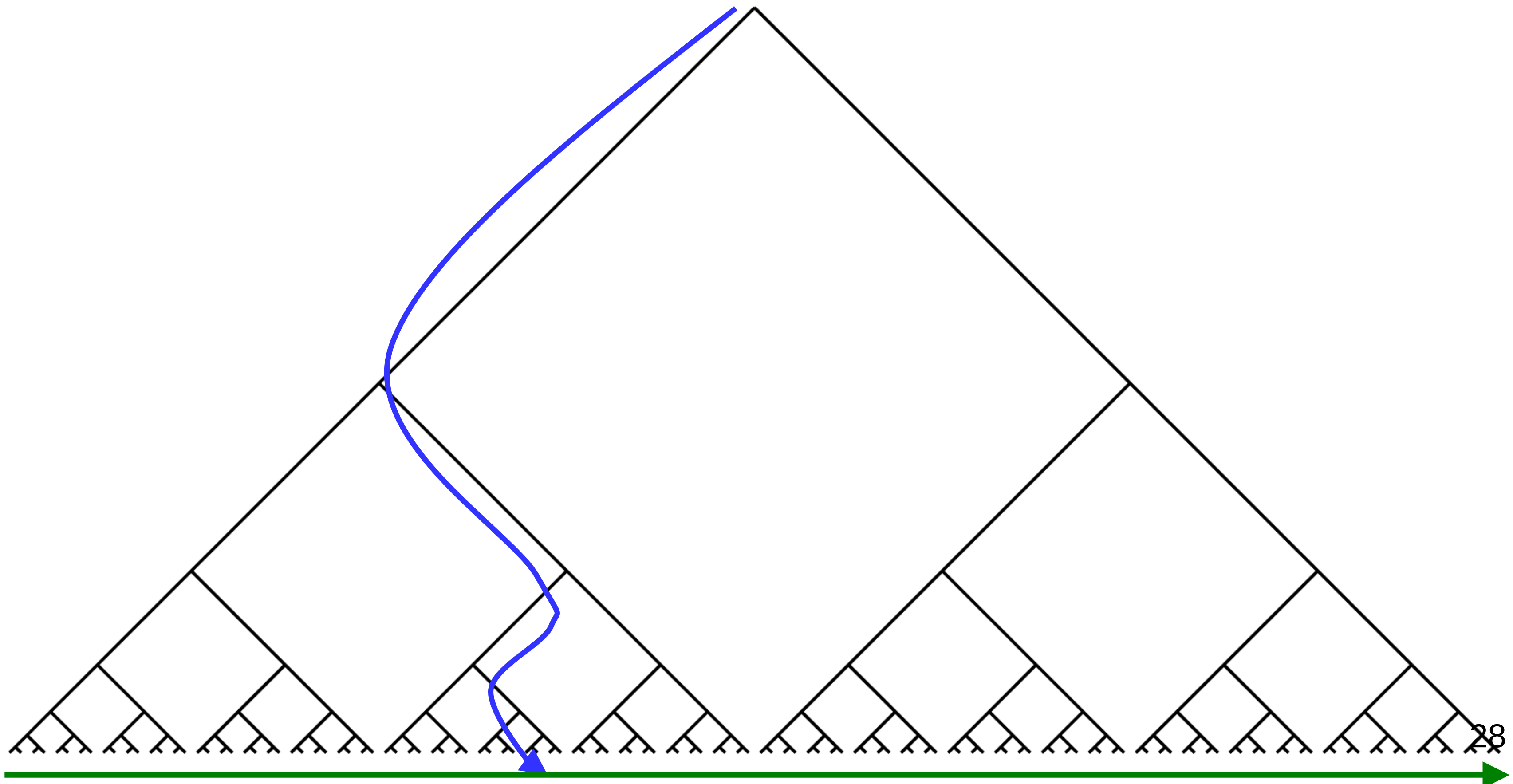
$$
\begin{aligned}
\mathrm{BSearch}[8, L] &= \mathrm{BSearch}[8, ((1,3),(8,9))] && \text{since } 8 \not> 9 \\
&= \mathrm{BSearch}[8, (8,9)] && \text{since } 8 > 3 \\
&= \mathrm{BSearch}[8, 8] && \text{since } 8 \not> 8 \\
&= \mathrm{true} && \text{since } 8 = 8
\end{aligned}
$$

How does this compare with the old Search function?

What would you like to compare?

Efficiency

How many comparisons need to be made to search through a balanced binary search tree with 255 nodes?



[redacted]

The number of comparisons made by Search on a list of depth p is:

$$C(p) = \begin{cases} 1 & \text{if } p = 0 \\ 2C(p-1) & \text{if } p > 0 \end{cases}$$

Closed form solution: $C(p) = 2^p$.

The number of comparisons made by Search on a list of depth p is:

$$C(p) = \begin{cases} 1 & \text{if } p = 0 \\ 2C(p-1) & \text{if } p > 0 \end{cases}$$

Closed form solution: $C(p) = 2^p$.

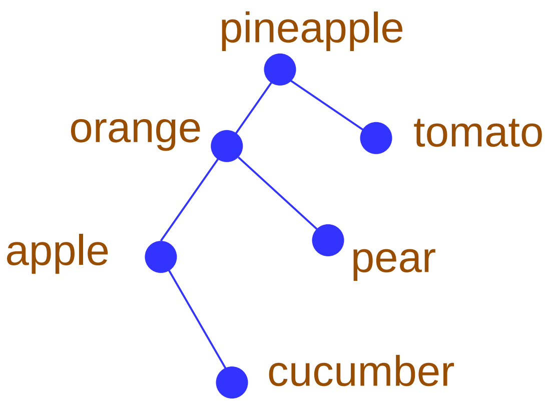
The number of comparisons made by BSearch is:

$$D(p) = \begin{cases} 1 & \text{if } p = 0 \\ 1 + D(p-1) & \text{if } p > 0 \end{cases}$$

Closed form solution: $D(p) = p + 1$.

Quiz 25-1

What sequence of input words below has triggered the construction of the following binary search tree ?



(a) pineapple, orange, tomato, apple, pear, cucumber

(b) apple, cucumber, orange, pear, pineapple, tomato

(c) pineapple, tomato, orange, pear, apple, cucumber

(d) pineapple, tomato, orange, pear, cucumber, apple

(e) cucumber, apple, pear, orange, tomato, pineapple

(f) cucumber, pear, tomato, apple, orange, pineapple