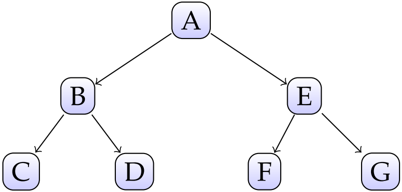
CSCI 2270: Data Structures

Lecture 17: Binary Search Trees

Ashutosh Trivedi



Department of Computer Science
UNIVERSITY OF COLORADO BOULDER

Recursive (Inductive) Definition

"In order to understand recursion you must first understand recursion."

Recursive (Inductive) Definition

Definition (Recursion Definition)

- Defining an object using recursive definition.
- Sometimes it is difficult to define a function or object explicitly; it is easier to define the function or object in terms of the function or object itself. This is called a recursive definition or recursion.

Recursive Definition

Recursive Definition

Recursive Definition: Examples

1. Set of natural numbers $\mathbb{N}$:
   - $0 \in \mathbb{N}$ (base case)
   - if $n \in \mathbb{N}$ then $(n+1) \in \mathbb{N}$. (general case)
2. Factorial $n! = n \cdot (n-1) \cdot \cdots \cdot 1$:

$$n! = \begin{cases} 1 & \text{if } n = 1 \\ n \cdot (n-1)! & \text{otherwise} \end{cases}$$

3. Fibonacci numbers $Fib(n)$:

$$Fib(n) = \begin{cases} 1 & \text{if } n = 1 \\ 2 & \text{if } n = 2 \\ Fib(n-1) + Fib(n-2) & \text{otherwise.} \end{cases}$$

Recursively Defined Structures

1. Strings.
   - ε is the empty string containing no symbols.
   - if w is a string and x is a character, then wx is a string.
2. Well-formed parenthetical expression over $\{(, \{, [,), \},]\}$
   - empty string is a well-formed parenthetical expression
   - If E is well-formed then so is (E) and $\{E\}$ and $[E]$.

Recursively Defined Structures

1. A Linked List L is:
   - either null,
   - or a node with a "key" and a next pointer pointing to a linked list L', i.e. $Node(key : a, next : L')$.
2. A Binary Search Tree T is
   - either null;
   - or a node with a "key" and a left and a right pointer pointing to a binary search trees T_ℓ and T_r, i.e., $Node(key : a, left : T_\ell, right : T_r)$.

Recursively-Definition and Induction

When a sequence is defined recursively, mathematical induction can be used to prove properties about it.

Theorem

For every $n \in \mathbb{N}$ the property $P(n)$ defined as

$$1 + 2 + 3 + \ldots + n = n(n+1)/2$$

holds.

Proof.

- (base case): Verify that $P(n)$ is true for $n = 0$
- (inductive step): Assuming that for an arbitrary n property $P(n)$ holds, show that it holds for $P(n+1)$.

From mathematical induction, $P(n)$ then holds for all $n \in \mathbb{N}$. □

Recursively-Definition and Structural Induction

When a structure is defined recursively, *structural induction* can be used to prove properties about it.

Theorem

For every linked-list L we have that

$$size(L) = \begin{cases} 0 & L = null \\ 1 + size(L') & L = Node(key : a, next : L'). \end{cases}$$

Proof.

- (base case): Verify that the property holds for the base case $L = null$.
- (inductive step): Assuming that property holds for all sub-structures used in the inductive definitions, show that the property will hold for the defined structure.

From structural induction, the property then holds for all structures. □

Recursively-Definition and Structural Induction

When a structure is defined recursively, *structural induction* can be used to prove properties about it.

Theorem

For every linked-list L, its reverse list is equal to:

$$reverse(L) = \begin{cases} null & L = null \\ reverse(L') : a & L = Node(key : a, next : L') \end{cases}$$

Proof.

- (base case): Verify that the property holds for the base case $L = null$.
- (inductive step): Assuming that property holds for all sub-structures used in the inductive definitions, show that the property will hold for the defined structure.

From structural induction, the property then holds for all structures. $\square$

Recursively-Definition and Structural Induction

When a structure is defined recursively, *structural induction* can be used to prove properties about it.

Theorem

For every binary search tree T we have that:

$$size(T) = \begin{cases} 0 & T = null \\ 1 + size(T_\ell) + size(T_r) & T = Node(key : a, left : T_\ell, right : T_r) \end{cases}$$

Proof.

- (base case): Verify that the property holds for the base case $L = null$.
- (inductive step): Assuming that property holds for all sub-structures used in the inductive definitions, show that the property will hold for the defined structure.

From structural induction, the property then holds for all structures. □

Recursively-Definition and Structural Induction

When a structure is defined recursively, *structural induction* can be used to prove properties about it.

Theorem

For every binary search tree T and a value k we have that $search(T, k)$ equals

$$\begin{cases} false & T = null \\ (k == a) || search(T_\ell, a) || search(T_r, a) & T = Node(key: a, left: T_\ell, right: T_r) \end{cases}$$

Proof.

- (base case): Verify that the property holds for the base case $L = null$.
- (inductive step): Assuming that property holds for all sub-structures used in the inductive definitions, show that the property will hold for the defined structure.

From structural induction, the property then holds for all structures. □

Exercise

Implement the following using recursion.

1. Search for an element in a linked list.
2. Compute the size of a linked list.
3. Search for an element in a binary search tree.
4. Compute the number of elements in a binary search tree.
5. Find the minimum element in a BST (recursive).
6. Find the maximum element in a BST (recursive).
7. Print all of the elements in the tree.
   - In-order
   - Pre-order
   - Post-order