

CSC 301: Analysis of Algorithms

Lecture 25: Binary Search Trees

William Rebelsky
[rebelskyw]

Grinnell College

Monday 1st April, 2024

Reminders

- Essay is due next Wednesday

Midterm Comments

- Problem 1: 8/10 were required to be correct for an S
- Problem 2: Needed to discuss the heap property in the loop invariant
- Problem 3: There were lots of issues with proving both that the greedy choice is valid and that the optimal substructure property held
- Problem 4: I was not strict on proofs as long as the proofs existed and you convincingly gave a faster than linear solution
- Problem 5: There appeared to be confusion on what operations you were allowed to take. The only comparison you could do was between one red and one blue jug. No blue to blue or red to red comparisons.

Midterm and Grade Updates

- I have released an assignment called Grade Matrix which is a quick way for you to see how you are doing in this course so far. If you have any questions about it, please let me know. (I have not yet added attendance).
- I have released midterm grades. An S required 90% (4/5 problems complete). If you did not get an S, I would like you to do the following:
 - 1 Review your grade
 - 2 One of the following:
 - 1 Write up an explanation of where you think you went wrong and an updated solution. Then send it to me.
 - 2 Schedule 20-30min to talk to me about where you went wrong and convince me that you understand the material related to that problem.

Course Schedule

- This Week (01 Apr): Trees
- Next Week (08 Apr): Graphs
- Week of 15 Apr: Flow
- Week of 22 Apr: Dynamic Programming
- Week of 29 Apr: Final Project
- Week of 06 Apr: TBD

Table of Contents

1 BST Basics

2 Practice

3 Insertion and Deletion

Binary Search Trees ADT

ADT operations:

- Minimum: Return the minimum value in the tree
- Maximum: Return the maximum value in the tree
- Predecessor: Return the previous value in the tree when printed in sorted order
- Successor: Return the next value in the tree when printed in sorted order
- Insert: Add an element to the tree
- Delete: Remove an element from the tree
- All operations take time proportional to height ($O(\lg n)$ in expectation)

Dictionaries and Priority queues

Binary Search Trees Implementation

- Key, Value/data, Left Child, Right Child, Parent
- Root has NIL Parent
- Leaves have NIL children
- BST Property: Left children are less than or equal to the value, Right children are greater than or equal to the value

Table of Contents

1 BST Basics

2 Practice

3 Insertion and Deletion

Practice 0

Given the set of keys $[1,4,5,10,16,17,21]$ draw binary search trees of heights:

- 2
- 3
- 4
- 5
- 6

Practice 1

How would you implement the following (sketch, not formal)

- Minimum
- Maximum
- Find/Search
- Printing in order

Practice 1 Answers

How would you implement the following (sketch, not formal)

- Minimum: Go left at each node
- Maximum: Go right at each node
- Find/Search: Binary search (if less, go left, if more go right)
- Printing in order: Recursive function. If value is not nil, recurse left, print val, recurse right

Practice 2

Given the answer to printing in order, how would you implement:

- Successor
- Predecessor

Practice 2 Answers

Given the answer to printing in order, how would you implement:

- Successor: If the right child is not NIL, return $\text{Tree-minimum}(x.\text{right})$. Otherwise, find the lowest ancestor of x whose left child is an ancestor of x
- Predecessor: If the left child is not Nil, return $\text{Tree-maximum}(x.\text{left})$. Otherwise return the parent of x .

Example [15,6,18,3,7,17,20,2,4,n,13,n,n,n,n,n,n,n,n,n,9]

Successor(13) is 15

Table of Contents

1 BST Basics

2 Practice

3 Insertion and Deletion

Insertion (If time)

How would you implement Insertion?

Insertion

- Similar to Tree-search, except extended to find a NIL value
- Updates the value and the parent's left or right child value
- Remember edge cases: empty tree

Deletion

Three basic cases:

- Z has no children: remove it by modifying parent to replace z with NIL as the child
- If Z has one child: modify parent to replace z with z's child
- If Z has two children: this is the complicated one. Who has ideas on how we could do this?

Case 3

- We know that the successor exists in the right subtree
- If Z's right child is the successor, this is easy, just replace Z with Z's right child
- Otherwise replace y with y's right child, then replace z with y.
- This may involve moving subtrees around, so we create a subroutine Transplant

Transplant

Algorithm 1: Transplant(T, u, v)

Input: Tree, nodes u and v **Output:** N/A

```
1 if  $u.p == NIL$  then  
2   |    $T.root = v$   
3 else if  $u == u.p.left$  then  
4   |    $u.p.left = v$   
5 else  
6   |    $u.p.right = v$   
7 if  $v \neq NIL$  then  
8   |    $v.p = u.p$ 
```

Tree Delete Pseudocode

Algorithm 2: Tree-delete(T, z)

Input: Tree, node to delete: z

```
1 if  $z.left == NIL$  then
2   | Transplant( $T, z, z.right$ )
3 else if  $z.right == NIL$  then
4   | Transplant( $T, z, z.left$ )
5 else
6   |  $y = \text{Tree-Minimum}(z.right)$ 
7   | if  $y \neq z.right$  then
8     | Transplant( $T, y, y.right$ )
9     |  $y.right = z.right$ 
10    |  $y.right.p = y$ 
11    Transplant( $T, z, y$ )
12     $y.left = z.left$ 
13     $y.left.p = y$ 
```

Last Question of the day

What is the difference between the BST property and Min Heap property? Can the min-heap property be used to print out the keys of an n node tree in sorted order in $O(n)$ time? Do one of the following: Show how and give an example, explain why not and give a counterexample

Last Question of the day Answer

No, min heap property is only parent vs child, does not give any information about left vs right subtree, so it is not possible.