Algorithmics Problem Sheet 3

Jan Kurkofka, Will Turner

Summer 2024

This problem sheet is due on Monday 29th April. The tutorial will be 14:30-16:00 in LAM-2090. Please have a go at as many problems as you can. As well as going through the problems, the session on Monday can be used to answer any other questions you have about the course.

Problem 1

For the set $\{1, 4, 5, 10, 16, 17, 21\}$ of keys, draw BSTs of heights 2, 3, 4, 5, and 6.

Problem 2

Consider the following algorithm which finds the node with minimum key in a BST.

Require: BST T
1: initialise $x \leftarrow \text{root}(T)$
2: **while** node left(x) exists **do**
3: $\quad x \leftarrow \text{left}(x)$
4: **return** x

Write a recursive version of the above algorithm. Recall that a recursive function is defined in terms of itself. That is, somewhere in the pseudocode, the algorithm should run itself.

Problem 3

Consider using the following algorithm to find the number 363 in a BST.

Require: BST T, key k
1: initialise $x \leftarrow \text{root}(T)$
2: **while** True **do**
3: $\quad$ **if** $T[x] = k$ **then**
4: $\quad\quad$ **return** x
5: $\quad$ **if** $T[x] > k$ **then**
6: $\quad\quad x \leftarrow \text{left}(x)$
7: $\quad$ **if** $T[x] > k$ **then**
8: $\quad\quad x \leftarrow \text{right}(x)$

Which of the following sequences cannot be the sequence of nodes examined?

(a) 2, 252, 401, 398, 330, 344, 397, 363.

(b) 924, 220, 911, 244, 898, 258, 362, 363.

(c) 925, 202, 911, 240, 912, 245, 363.

(d) 2, 399, 387, 219, 266, 382, 381, 278, 363.

(e) 935, 278, 347, 621, 299, 392, 358, 363.

Problem 4

The following is the algorithm for inserting a new node into a BST:

```
Require: BST T, new key value k
initialise x ← 0, p ← 0
while node x exists do
    p ← x
    if k < T[x] then
        x ← left(x)
    if k > T[x] then
        x ← right(x)
if k < T[p] then
    create left child of p with key k
if k > T[p] then
    create right child of p with key k
```

Give a recursive version of the above algorithm.

Problem 5

Give a non-recursive algorithm that 'traverses' a BST in order. That is, it takes a BST as an input and outputs each key in order.

Problem 6

Given a BST T and a node x in T, we write $\text{succ}(x)$ to denote the node with the next-highest key value in T. Show that if a node in a BST has two children, then $\text{succ}(x)$ has no left child.

Problem 7

It is possible to sort a set of numbers by first building a BST containing them and then traversing the BST in order. To build a BST, we can repeatedly run our node insertion algorithm on the unsorted set. What are the worst-case and best-case running times for this sorting algorithm?

Problem 8

Consider the following algorithm. It takes in an array $A[1:2n]$ of even length $2n$ and swaps the key $A[i]$ with the key $A[2n+1-i]$ for each $i = 1, \ldots, n$.

State a loop invariant for this procedure. Use it to show that the algorithm reverses the order of the keys in the array A.

Require: array $A[1:2n]$

1: **for** $i = 1$ upto n **do**

2: swap the keys $A[i]$ and $A[2n+1-i]$

3: **return** $A[1:2n]$

HINT: You need to come up with a true/false statement that holds at each iteration of the loop and is dependent on i. Then you need to show that this is true at the first step. Then you need to show that, if it is true at the ith iteration, it's also true at the $(i+1)$th iteration. Finally, you must show that when the algorithm ends, that this loop invariant tells you that the algorithm has swapped all of the numbers in the array correctly.