Weekplan: Search Trees

Philip Bille Inge Li Gørtz

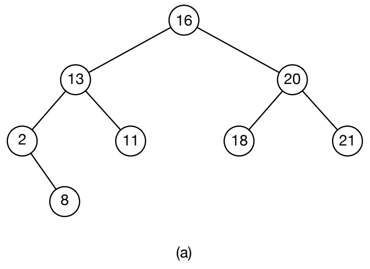
Reading

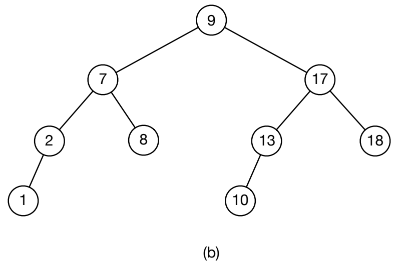
Introduction to Algorithms, 4th edition, Cormen, Rivest, Leisersons, and Stein (CLRS): Chapter 12 + *Algorithms*, 4th edition, Sedgewick and Wayne: Chapter 3.3.

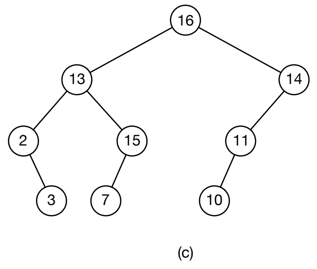
Exercises

1 Basics of Binary Search Trees

1.1 [w] Which of the following trees are binary search trees?







1.2 [w] Where are the elements with the smallest and largest key located in a binary search tree?

1.3 [w] Consider the set of keys $\{1,4,5,10,16,17,21\}$. Draw binary search trees of height 2, 3, 4, 5, and 6 containing these keys.

1.4 [w] Specify the preorder, inorder, or postorder sequence of keys for the tree in (b).

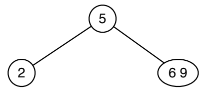
1.5 Compare the heap property and the search tree properties.

1.6 [w] Write pseudo-code for computing the inorder traversal of a binary search tree.

1.7 Show that if a node v in a binary search has 2 children, then the node with the smallest key $>$ than $v.key$ has no left child and the node with the largest key $<$ than $v.key$ has no right child. Assume that all keys are distinct for simplicity. *Hint:* prove it by contradiction.

2 Basics of 2-3 trees

2.1 [w] Consider the following 2-3 tree. Insert the sequence of keys $4,10,1$. Show the resulting tree after each step.



2.2 Consider a 2-3 tree T with n items. What is the maximum height and the minimum height of T? Conclude that the height of T is always $\Theta(\log n)$.

3 Queries on Balanced Search Trees Consider the following queries on 2-3 trees T.

- PREDECESSOR(k): return the element with the largest key in T that is $\leq k$.
- RANGEREPORT(k_1, k_2): return the set of elements with keys in the range $[k_1, k_2]$.
- RANGECOUNT(k_1, k_2): return the *number* of elements with keys in the range $[k_1, k_2]$.

For instance, on the small tree in Exercise **2.1**, we have that PREDECESSOR(3) returns the element with key 2, PREDECESSOR(7) returns the element with key 6, and PREDECESSOR(6) returns the element with key 6. Similarly, RANGEREPORT(3, 8) return the set of elements with keys $\{5, 6\}$ and RANGECOUNT(3, 8) returns 2.

3.1 Give an algorithm that supports PREDECESSOR in $O(\log n)$ time.

3.2 Give an algorithm that supports RANGEREPORT in $O(\log n + \text{occ})$ time, where occ is the number of elements in the output.

3.3 Give an algorithm that supports RANGECOUNT in $O(\log n)$ time. Here you will need to store and maintain some additional information in the data structure.

4 Inventory Management Suppose you are running a business that sells recycled single socks. The socks arrive from your suppliers (recycling facilities where people drop off their old socks), and you need to keep track of your current inventory and some basic statistics. Specifically, each sock is identified by a *size* and a *color*. We assume that two integers represent size and color. You want to maintain a database D of socks that support the following operations:

- NEW(s, c): Add a new sock to D of size s and color c.
- SELL(s, c): Remove a sock of size s and color c. Return "unavailable" if there is no such sock in D.
- UNIQUE: Return the total number of *distinct* socks in D, i.e., the number of different pairs (s, c) of sock size and color that appear in D.
- MAXFREQUENT: Return the most frequent sock that appears in D, i.e., the most frequent pair (s, c) that appears in D. If there is more than one such pair, return any of them.

Solve the following exercises.

4.1 Give an efficient data structure that supports NEW, SELL, and UNIQUE.

4.2 [$*$] Give an efficient data structure that supports all operations.