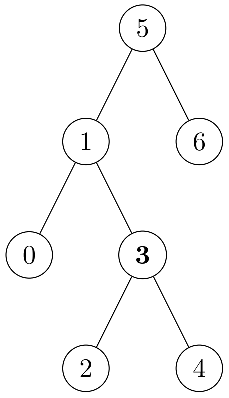
CS 161 Fall 2017: Section 3 Solutions

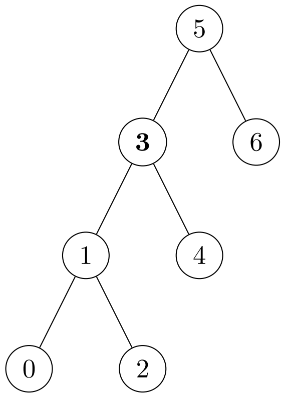
Tree Rotations

Modify the following trees via a sequence of rotations so that 3 is at the root of each tree. Draw out the intermediate stages (result after each rotation step). There could be an arbitrary number of descendants from the leaves of the trees shown. Your rotations should not depend on these descendants.

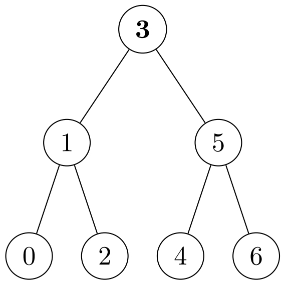
(a)



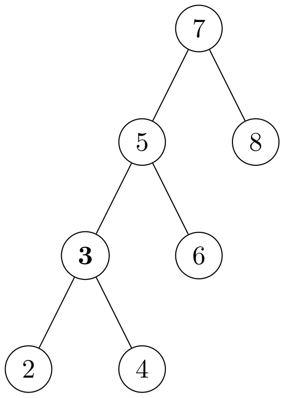
First, we need to rotate the 3 up left, where 1 currently is. This leaves the tree as



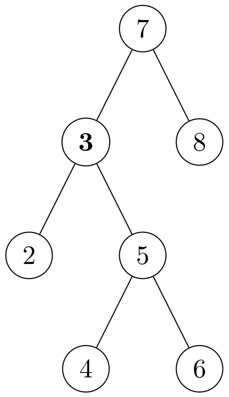
Next, we rotate the 3 up to the root, leaving

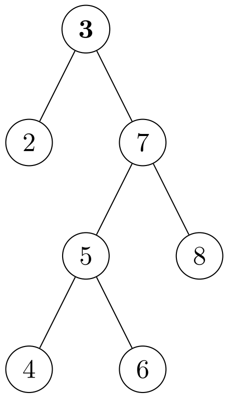


(b)



We need to successive rightward rotations to get 3 to the root.





Randomly Built BSTs

In this problem, we prove that the average depth of a node in a randomly built binary search tree with n nodes is $O(\log n)$. A *randomly built binary search tree* with n nodes is one that arises from inserting the n keys in random order into an initially empty tree, where each of the $n!$ permutations of the input keys is equally likely.

Let $d(x, T)$ be the depth of node x in a binary tree T (the depth of the root is 0). Then, the average depth of a node in a binary tree T with n nodes is

$$\frac{1}{n}\sum_{x\in T} d(x,T) \ .$$

(a) Let the *total path length* $P(T)$ of a binary tree T be defined as the sum of the depths of all nodes in T, so the average depth of a node in T with n nodes is equal to $\frac{1}{n}P(T)$. Show that $P(T) = P(T_L) + P(T_R) + n - 1$, where T_L and T_R are the left and right subtrees of T, respectively.

Let $r(T)$ denote the root of tree T. Note the depth of node x in T is equal to the length of the path from $r(T)$ to x. Hence, $P(T) = \sum_{x\in T} d(x,T)$.

For each node x in T_L, the path from $r(T)$ to x consists of the edge $(r(T), r(T_L))$ and the path from $r(T_L)$ to x. The same reasoning applies for nodes x in T_R. Equivalently, we have

$$d(x,T) = \begin{cases} 0, & \text{if } x = r(T) \\ 1 + d(x,T_L), & \text{if } x \in T_L \\ 1 + d(x,T_R), & \text{if } x \in T_R \end{cases} \ .$$

Then,

$$\begin{aligned} \sum_{x\in T} d(x,T) &= d(r(T),T) + \sum_{x\in T_L} d(x,T) + \sum_{x\in T_R} d(x,T) \\ &= 0 + \sum_{x\in T_L}[1 + d(x,T_L)] + \sum_{x\in T_R}[1 + d(x,T_R)] \\ &= |T_L| + |T_R| + \sum_{x\in T_L} d(x,T_L) + \sum_{x\in T_R} d(x,T_R) \\ &= n - 1 + P(T_L) + P(T_R) \ . \end{aligned}$$

It follows that $P(T) = P(T_L) + P(T_R) + n - 1$.

(b) Let $P(n)$ be the expected total path length of a randomly built binary search tree with n nodes. Show that $P(n) = \dfrac{1}{n}\displaystyle\sum_{i=0}^{n-1}(P(i) + P(n-i-1) + n - 1)$.

Let T be a randomly built binary search tree with n nodes. Without loss of generality, we assume the n keys are $\{1, \dots, n\}$.

By definition, $P(n) = \mathbf{E}_T[P(T)]$. Then, $P(n) = \mathbf{E}_T[P(T_L) + P(T_R) + n - 1] = n - 1 + \mathbf{E}_T[P(T_L)] + \mathbf{E}_T[P(T_R)]$, where T_L and T_R are the left and right subtrees of T, respectively. Note

$$\mathbf{E}_T[P(T_L)] = \sum_{i=1}^{n} \mathbf{E}_T[P(T_L) | r(T) = i] \cdot \Pr(r(T) = i) \ .$$

Since each element is equally likely to be the root of T, $\Pr(r(T) = i) = \frac{1}{n}$ for all i. Conditioned on the event that element i is the root, T_L is a randomly built binary search tree on the first $i-1$ elements. To see this, assume we picked element i to be the root. From the point of view of the left subtree, the elements $1, \dots, i-1$ are inserted into the subtree in a random order, since these elements are inserted into T in a random order and subsequently go into T_L in the same relative order. Hence, $\mathbf{E}_T[P(T_L)|r(T) = i] = P(i-1)$. Putting these together, we get

$$\mathbf{E}_T[P(T_L)] = \sum_{i=1}^{n} \frac{1}{n} P(i-1) \ .$$

Similarly, we get $\mathbf{E}_T[P(T_R)] = \sum_{i=1}^{n} \frac{1}{n} P(n-i)$. Then,

$$\begin{aligned} P(n) &= n - 1 + \mathbf{E}_T[P(T_L)] + \mathbf{E}_T[P(T_R)] \\ &= n - 1 + \frac{1}{n} \sum_{i=1}^{n} [P(i-1) + P(n-i)] \\ &= n - 1 + \frac{1}{n} \sum_{i=0}^{n-1} [P(i) + P(n-i-1)] \ , \end{aligned}$$

where we changed the indexing of the summation in the last equality.

(c) Show that $P(n) = O(n \log n)$. You may cite a result previously proven in the context of other topics covered in class.

This is the same recurrence that appears in the analysis of Quicksort.

(d) Design a sorting algorithm based on randomly building a binary search tree. Show that its (expected) running time is $O(n \log n)$. Assume that a random permutation of n keys can be generated in time $O(n)$

The algorithm is 1) construct a randomly built binary search tree T by inserting given elements in a random order; and 2) do the inorder traversal on T to get a sorted list. Note step 2 can be done in $O(n)$ time. We argue that step 1 takes $O(n \log n)$ time in expectation. We observe that given the final state of tree T, we can compute the amount of work spent to construct T. To insert a node x at depth d, we traversed exactly the path from the root to the parent of x, at depth $d-1$, to insert it. Hence, we can upper bound the total work done to construct T by $O(P(T))$. From part (c), we know that $P(T) = O(n \log n)$ in expectation. It follows that Step 1 takes $O(n \log n)$ time in expectation. Overall, the algorithm runs in $O(n \log n)$ in expectation.