Cost to build a random binary search tree

Imagine building a Binary Search Tree (BST) by inserting n elements into an empty BST in a random order. Each inserted item is compared to every one of its ancestors (from the root down to its parent) before it is added (as a leaf), so the total number of comparisons performed to build the BST is the total sum of all the node depths. Let $D(n)$ be the sum of the depths of all n nodes in the final BST averaged over all $n!$ insertion orders. The first element inserted becomes the root key of the final BST, so if it is the ith smallest then the root's left subtree is a BST with $i-1$ elements (inserted in random order) and the root's right subtree is a BST with $n-i$ elements (inserted in random order). The sum of the depths of the $i-1$ elements in the left subtree is $D(i-1)$ on average, while the right subtree has average depth sum $D(n-i)$. Notice that every element in both subtrees is one deeper in the final tree because the root is their parent, so the final tree's average depth sum, assuming the root is the ith smallest value, is $D(i-1)+D(n-i)+n-1$. It is equally likely that the first element inserted is the ith smallest for any value of i from 1 to n, so the final tree's average depth sum is:

$$D(n) = \frac{1}{n}\sum_{i=1}^{n}(D(i-1)+D(n-i)+n-1) = n-1+\frac{2}{n}\sum_{i=1}^{n}D(i-1)$$

and $D(1) = 0$. This is the same recurrence we saw for calculating the average case number of comparisons performed by Quicksort! Why? Because it's counting the same thing. The pivots Quicksort compares to a particular element are the ancestors of the element in the BST. So the depth of an element in the BST equals the number of comparisons it is part of in Quicksort. If we sum up the depths, we sum up the number of comparisons.

To find a closed form expression for $D(n)$, multiply by n and subtract the recurrence for $D(n-1)$:

$$nD(n)-(n-1)D(n-1) = 2(n-1)+2D(n-1) \qquad \text{for } n>1$$

Rearrange terms: $nD(n) = (n+1)D(n-1)+2(n-1) \qquad \text{for } n>1$.
Divide both sides by $n(n+1)$:

$$\frac{D(n)}{n+1} = \frac{D(n-1)}{n}+\frac{2(n-1)}{n(n+1)} \qquad \text{for } n>1.$$

Repeatedly substitute:

$$\frac{D(n)}{n+1} = \frac{D(1)}{2}+2\sum_{i=2}^{n}\frac{2(i-1)}{i(i+1)} \qquad \text{for } n>1.$$

Do partial fraction expansion:

$$\frac{2(i-1)}{i(i+1)} = \frac{a}{i}+\frac{b}{i+1} = \frac{a(i+1)+bi}{i(i+1)} = \frac{(a+b)i+a}{i(i+1)}$$

where $a+b=2$ and $a=-2$, so $b=4$. Now the summation splits into two pieces:

$$\sum_{i=2}^{n}\frac{2(i-1)}{i(i+1)} = \sum_{i=2}^{n}\frac{-2}{i}+\sum_{i=2}^{n}\frac{4}{i+1} = 4(1/3+1/4+\cdots+1/(n+1))-2(1/2+1/3+\cdots+1/n).$$

The sum $H(n) = 1+1/2+1/3+\cdots+1/n$ is called the nth Harmonic number and it is well-approximated by $\ln(n)$ for large n. Hence,

$$D(n) \approx 2(n+1)(4\ln n - 2\ln n) \in \Theta(n\log n).$$