Advanced Data Structures

Computer Science Department

Random Binary Search Trees: An empirical analysis

Author:
Alex Herrero

Professor:
Amalia Duch

February 24, 2025

Contents

List of Figures

List of Tables

1 Introduction

Let T be a binary tree with subtrees T_l and T_r. We say that T is a *binary search tree* (BST) if it is either an empty binary tree or it contains at least one element x as its root such that

- T_l and T_r are also BSTs.
- $\forall y \in T_l, y < x$ and $\forall z \in T_r, z > x$.

Although it is well known that, in the worst case, a BST behaves like a linked list (with the height of the tree being $\Theta(n)$), in this report, we focus on *random BSTs* of size n.

By *random BSTs*, we mean the following: Given a universe of keys U with $|U| = n$, we construct the BST by inserting each element of U exactly once, choosing the insertion order uniformly at random.

2 Analysis of the Average Cost of Insertions

Let us first analyze the expected cost of inserting an element $u \in U$ into our BST T. For that, we will consider this cost as the cost of searching for u in our BST, which is valid since we assume that our search terminates in any empty subtree with identical probability.

Let $R(T)$ be a recurrence that, given a BST T with subtrees T_l and T_r, returns the expected number of nodes that the BST will traverse during the search. Then, $R(T)$ can be calculated as:

$$\begin{cases} R(T) = 0, & \text{if } |T| = 0, \\ R(T) = 1 + \frac{|T_l|}{n} R(T_l) + \frac{|T_r|}{n} R(T_r), & \text{otherwise.} \end{cases}$$

To explain $R(T)$, it suffices to note that, since each element is drawn from a uniformly random distribution over n different values, each node has probability $\frac{1}{n}$ of being u. Thus, we have:

$$\mathbb{P}[u \in T_l] = \mathbb{P}[\exists l_i \in T_l : l_i = u] = \sum_{l_i \in T_l} \mathbb{P}[l_i = u] = \sum_{i=1}^{|T_l|} \frac{1}{n} = \frac{|T_l|}{n}.$$

Using the same reasoning, we find that $\mathbb{P}[u \in T_r] = \frac{|T_r|}{n}$, which allows us to compute the expected number of nodes we need to traverse in order to find u.

Although this recurrence is useful, we must express it in terms of the size n to obtain a general result. Let I_n be the expected cost of searching for a key u in a random BST of size n, measured as the number of nodes visited until reaching u. Then, I_n can be expressed as:

$$\begin{cases} I_n = 0, & \text{if } n = 0, \\ I_n = 1 + \frac{1}{n} \sum_{k=0}^{n-1} \left(\frac{k}{n} I_k + \frac{(n-1-k)}{n} I_{n-1-k} \right), & \text{otherwise.} \end{cases}$$

The key difference from the previous recurrence is that we must now consider every possible partitioning of the tree sizes as being equally likely.