Advanced Data Structures

Computer Science Department

Random Binary Search Trees: An empirical analysis

Author:
Alex Herrero

Professor:
Amalia Duch

March 3, 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 can assume that, if u does not exist in T, our search terminates in any empty subtree with identical probability.

Let I_n be the expected cost of the insertion of a key x in a random BST of size n. Let, also, be $I_{n,q}$ be the expected cost of the insertion of a key x in a random BST which root is the $q-$th smallest element. Then, the expected cost of I_n is

$$\begin{aligned}
I_n &= \frac{1}{n}\sum_{q=1}^{n} I_{n,q} \\
&= 1 + \frac{1}{n}\sum_{k=1}^{n}(\frac{1}{n}0 + \frac{k-1}{n}I_{k-1} + \frac{n-k}{n}I_{n-k}) \\
&= 1 + \frac{1}{n}\sum_{k=0}^{n-1}(\frac{k}{n}I_k + \frac{n-k-1}{n}I_{n-k-1}) \\
&= 1 + \frac{1}{n^2}\sum_{k=0}^{n-1}(kI_k + (n-k-1)I_{n-k-1}) \\
&= 1 + \frac{2}{n^2}\sum_{k=0}^{n-1} kI_k
\end{aligned}$$

We can solve this recurrence using the continuous master theorem. The continuous master theorem solves recurrences of the form

$$F_n = t_n + \sum_{0 \le j < n} w_{n,j}F_j$$

with $t_n = \Theta(n^a(\log n)^b)$. We proceed as follows:

- Determine the values of a and b: Since $t_n = \Theta(1)$, it is straightforward to see that $a = b = 0$.
- Provide a shape function for the weights $w_{n,j}$: We use the following trick to determine the shape function:

$$w(z) = \lim_{n\to\infty} n \cdot w_{n,z\cdot n} = n \cdot \frac{2zn}{n^2} = 2z.$$

- Determine the value of

$$\mathcal{H} = 1 - \int_0^1 w(z) z^a dz.$$

Substituting the values, we obtain:

$$\mathcal{H} = 1 - \int_0^1 2z dz = 1 - (1 - 0) = 0.$$

- Since $\mathcal{H} = 0$, we need to compute

$$\mathcal{H}' = -(b+1) \int_0^1 w(z) z^a \ln z \, dz.$$

Substituting the known values,

$$\mathcal{H}' = -1 \int_0^1 2z \ln z \, dz.$$

This integral can be solved using integration by parts. For the purpose of applying the theorem, we skip the detailed calculation, giving the result:

$$\mathcal{H}' = -(x^2 \ln x - \frac{x^2}{2})\Big|_0^1 = \frac{1}{2}.$$

Since $\mathcal{H} = 0$ and $\mathcal{H}' \neq 0$, we use the result

$$F_n = \frac{t_n}{\mathcal{H}'} \ln n + o(t_n \log n).$$

Substituting the values, we obtain

$$I_n = 2 \ln n + o(\log n).$$

Thus, the expected cost of an insertion into a random binary search tree is bounded by $O(\log n)$.