Computer Science 331

Average Case Analysis: Binary Search Trees

Mike Jacobson

Department of Computer Science
University of Calgary

Lecture #15

Outline

Cost of Binary Search Tree Operations

Operations on a Binary Search Tree T ...

- Require a walk down (part of) a path from the root to a leaf of the tree
- Constant time is required for each node that is visited

Thus, the worst-case time of each operation is:

- linear in the *height* of T

Bounds on Height: Worst- and Average-Case

If a binary search tree T has size n and height h then

$$n \leq 2^{h+1} - 1, \quad \text{so that} \quad h \geq \log_2(n+1) - 1$$

and

$$n \geq h + 1, \quad \text{so that} \quad h \leq n - 1 \ .$$

Worst Case: These bounds cannot be improved.
In particular, $h = n - 1$ in some cases.

Average Case: It *seems* that $h \in \Theta(\log n)$ most of the time.

Objective, and Difficulty

Objective:

- Prove that the height of a binary search tree really *is* logarithmic in its size, "most of the time."

Difficulty:

- This — or any other "average case analysis" — requires an *assumption* about how frequently each binary search tree (of a given size) occurs.
- **If our assumption is inaccurate then so is our analysis!**

Concepts from Probability Theory

These will also be useful for the analysis of operations on *hash tables* and the `QuickSort` algorithm, later in the course.

- **Sample Space:** Set S of *events* that we are interested in. We will be interested in situations where S is a *finite* set.
- **Probability Distribution:** Function $\Pr : S \to \mathbb{R}$ such that

$$0 \leq \Pr(s) \leq 1 \text{ for all } s \in S \quad \text{and} \quad \sum_{s \in S} \Pr(s) = 1.$$

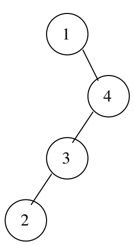
- **Random Variable:** A real valued function of S. That is, a function $X : S \to \mathbb{R}$.
- **Expected Value of a Random Variable:** The *expected value* of a random variable X is

$$\mathrm{E}[X] = \sum_{s \in S} \Pr(s) \cdot X(s).$$

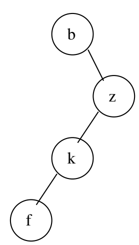
Useful Property of Shape

Problem: There are infinitely many binary search trees of a given size!

Consider the following binary search trees, each obtained by inserting four elements into an empty tree.



Insertion Order: 1, 4, 3, 2



Insertion Order: b, z, k, f

Useful Property of Shape (cont.)

If

- T_1 is generated by inserting a sequence of values $x_1, x_2, \ldots, x_n$ into an initially empty tree, and
- T_2 is generated by inserting a sequence of values $y_1, y_2, \ldots, y_n$ into an initially empty tree, and
- **for all** i, j such that $1 \leq i, j \leq n$,

$$x_i \leq x_j \quad \text{if and only if} \quad y_i \leq y_j$$

then T_1 and T_2 have the same **shape** — and the same *height*.

Assumption for Analysis

Conclusion: It is sufficient to consider the *relative order* of the inserted keys when considering the height of a binary search tree.

Condition and Assumption for Analysis:

- **Condition:** We will consider binary search trees of size n, produced by inserting $1, 2, \ldots, n$ into an empty tree *in some order*
- **Fact:** There are $1 \times 2 \times \cdots \times n = n!$ possible relative orders of these values
- **Assumption:** We will *assume* that each of these relative orders is equally likely.

Ideas from Probability Theory, Applied

Making This Formal:

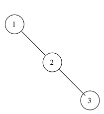
- When considering binary search trees of size n we will use a *sample space* S_n of size $n!$ — whose elements correspond to the $n!$ relative orderings of the inserted keys
- According to the assumptions that have been stated we will be using the *uniform distribution* in our analysis:

$$\Pr(s) = \frac{1}{|S_n|} = \frac{1}{n!} \qquad \text{for all } s \in S_n$$

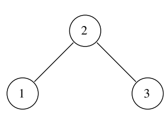
Possible Relative Orders and Trees When $n = 3$

Insertion order appears above each tree.

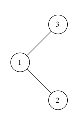
T_1: 1, 2, 3



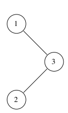
T_3: 2, 1, 3



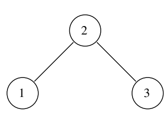
T_5: 3, 1, 2



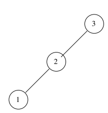
T_2: 1, 3, 2



T_4: 2, 3, 1



T_6: 3, 2, 1



Note: Tree **shapes** do not all occur with the same probability (under our assumption).

Exponential-Height

If a binary search tree has height h, its *exponential-height* is 2^h.

Heights and Exponential Heights of Previous Trees

i	1	2	3	4	5	6
height(T_i)	2	2	1	1	2	2
exp-height(T_i)	4	4	2	2	4	4

Average Exponential Height if $n = 3$ (Written as Y_n):

$$\text{E(exp-height)} = Y_3 = \frac{1}{6}(4+4+2+2+4+4) = \frac{10}{3}$$

Goal: determine an upper bound on Y_n, derive bound on avg. height

Trees with Root i

Suppose i is an integer between 1 and n.

One Way To Choose a Relative Ordering Starting with i**:**

- Begin with i as the first thing to list
- Pick one of the $(n-1)!$ relative orderings uniformly and independently. Use this to determine the ordering for the other values that should be listed after i

Trees with Root i

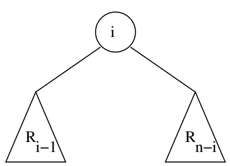
Another Way To Choose a Relative Ordering Starting with i**:**

- Begin with i as the first thing to list
- Choose one of the $\binom{n-1}{i-1}$ subsets of the remaining positions of size $i-1$, from the $n-1$ positions that are left after this — the integers between 1 and $i-1$ will be placed here
- Choose one of the $(i-1)!$ relative orderings for the integers less than i. Insert the values $1, 2, \ldots, i-1$ *in the positions chosen in the previous step* using this ordering
- Choose one of the $(n-i)!$ relative orderings for the integers between $i-1$ and n. Insert the values $i+1, i+2, \ldots, n$ *in the positions that are left* using this ordering.

Trees with Root i

Crucial Observation: Each of these methods produces exactly the same set of relative orderings, and every ordering that starts with i is listed exactly once, in each case.

The corresponding trees are as follows:



R_{i-1} : BST with $i-1$ nodes $1, 2, \ldots, i-1$

- all $(i-1)!$ relative orders equally likely

R_{n-i} : BST with $n-i$ nodes $i+1, i+2, \ldots, n$

- all $(n-i)!$ relative orders equally likely

Exponential Height with Root i

Bounds on height and exponential height:

- If a tree T has a left subtree with height h_L and a right subtree with height h_R, then height of T is $1 + \max(h_L, h_R)$
- If a tree T has a left subtree with exp-height H_L and a right subtree with exp-height H_R, then the exp-height of T is

$$2 \cdot \max(H_L, H_R) \leq 2 \cdot (H_L + H_R) \ .$$

Consequence: The average exponential-height of a binary search tree with n nodes $(1, 2, \ldots, n)$ *and root* i is

$$Y_{n,i} = 2 \cdot \max(Y_{i-1}, Y_{n-i}) \leq 2 \cdot (Y_{i-1} + Y_{n-i})$$

Relationship holds for $i = 1$ and $i = n$ if we "define" Y_0 to be 0.

Recurrence for Y_n

Since every binary search tree with size one has height zero,

$$Y_1 = 2^0 = 1 \ .$$

A binary search tree with n nodes $1, 2, \ldots, n$ has root i with likelihood $1/n$ (under our assumption). Thus

$$\begin{aligned} Y_n &= \frac{1}{n}\sum_{i=1}^{n} Y_{n,i} \\ &\le \frac{2}{n}\sum_{i=1}^{n}(Y_{n-i} + Y_{i-1}) \\ &= \frac{4}{n}\sum_{i=0}^{n-1} Y_i. \end{aligned}$$

Bounding Y_n Using the Recurrence

It is possible to use *mathematical induction* to show that

$$\frac{4}{n}\sum_{i=0}^{n-1}\binom{i+3}{3} = \frac{4}{n}\binom{n+3}{4} = \binom{n+3}{3}$$

where the binomial coefficient $\binom{n}{k} = \frac{n!}{k!(n-k)!}$.

It is also easily checked that

$$Y_1 = 1 = \frac{1}{4}\binom{1+3}{3} \ .$$

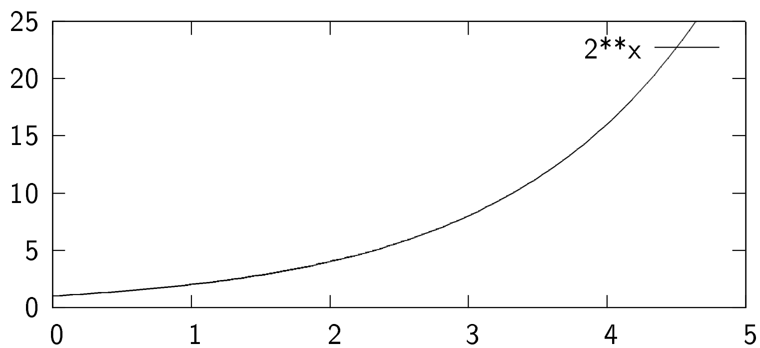
These can be used with the previous inequality to prove that

$$Y_n \le \frac{1}{4}\binom{n+3}{3} = \frac{(n+3)(n+2)(n+1)}{24}$$

for *every* integer $n \ge 1$.

Useful Property of $f(x) = 2^x$

Consider the function $f(x) = 2^x$:



This function is **convex**: If $\alpha \ge 0$, $\beta \ge 0$, and $\alpha + \beta = 1$ then

$$f(\alpha x_1 + \beta x_2) \le \alpha f(x_1) + \beta f(x_2) \ .$$

Useful Property of $f(x) = 2^x$ (cont.)

Theorem 1 (Jensen's Inequality)

For every integer $m \ge 1$ and positive values $x_1, x_2, \ldots, x_m$,

$$f\left(\tfrac{1}{m}(x_1 + x_2 + \cdots + x_m)\right) \le \frac{1}{m}\left(f(x_1) + f(x_2) + \cdots + f(x_m)\right)$$

if the function f is convex.

Can be proved by induction on m.

Because 2^x is convex, Jensen's Inequality is applicable

Application of Property

Let X_n be the average height of a binary search tree with size n (under our assumption). Then

$$X_n = \frac{1}{m}(h_1 + h_2 + \cdots + h_m)$$

where $m = n!$ and $h_i = \text{height}(T_i)$.

Consequence of Previous Inequality:

$$2^{X_n} \leq \frac{1}{m}\left(2^{h_1} + 2^{h_2} + \cdots + 2^{h_m}\right) = Y_n \ .$$

Note that this implies

$$X_n \leq \log_2 Y_n \ .$$

Simplification of Bound

Corollaries: *Under Our Assumption about Construction of Trees*

1. Average height of a binary search tree of size n is

$$X_n \leq \log_2 Y_n \leq \log_2 \left(\frac{1}{4}\binom{n+3}{3}\right),$$

   so that $X_n \leq \log_2 n^3 = 3\log_2 n$ for sufficiently large n.

2. If c is a positive integer, n is sufficiently large, and T is a randomly constructed BST with size n, then the probability that

$$\text{height}(T) \geq 3c\log_2 n$$

   is less than $\frac{1}{c}$.