

With  $0 < k < n$  and  $k, n \in \mathbb{Z}$ , we prove the following lemmas

### Lemma 1

$$\prod_{i=k}^n i = \frac{n!}{(k-1)!}$$

Proof:

$$\prod_{i=k}^n i = \prod_{i=k}^n i \frac{(k-1)!}{(k-1)!} = \prod_{i=k}^n i \frac{\prod_{i=1}^{k-1} i}{(k-1)!} = \frac{n!}{(k-1)!}$$

### Lemma 2

$$\prod_{i=0}^k (n-i) = \frac{n!}{(n-k-1)!}$$

Proof:

$$\prod_{i=0}^k n-i = (n)(n-1) \cdots (n-k) = \prod_{i=(n-k)}^n i$$

And by using lemma 1 we get

$$\prod_{i=(n-k)}^n i = \frac{n!}{(n-k-1)!}$$

## Problem Statement

We are given a set of  $n$  nodes in a valid binary search tree  $B$ , with unique keys from a (finite) set of contiguous integers  $S$ .

We then pick another key  $x \in S$ , and begin searching for the key using a linear BST algorithm.<sup>1</sup> How does  $p(x \in B)$  change with each node traversed?

## Calculating $p(x \in B)_0$ using products

Before beginning the search, ie at time  $t = 0$ , we calculate  $p(x \in B)_0$  as the probability that  $x$  is chosen in  $n$  tries from a set of  $|S|$  items, by first calculating its inverse.

The probability of not picking  $x$  on the 1st try is  $\frac{|S|-1}{|S|}$ . On the next try, it will be  $\frac{|S|-2}{|S|-1}$ , accounting for the missing element, and so on until we have the final probability as  $\frac{|S|-n}{|S|-(n-1)}$ .

Thus the probability that we never picked  $x$  for any node is:

$$p(x \notin B)_0 = \prod_{i=0}^{n-1} \frac{|S| - (i+1)}{|S| - i} = \frac{\prod_{i=0}^{n-1} (|S| - 1 - i)}{\prod_{i=0}^{n-1} |S| - i}$$

---

<sup>1</sup>Is 'a linear BST' algorithm guaranteed to traverse nodes in a certain order? Can we prove that the linear BST algorithm is unique given certain constraints? (eg "nodes are traversed in same order")

We can now apply lemma 2 on both numerator and denominator to get

$$\begin{aligned} & \left( \frac{(|S| - 1)!}{((|S| - 1) - (n - 1) - 1)!} \right) \left( \frac{|S|!}{(|S| - (n - 1) - 1)!} \right)^{-1} \\ &= \frac{|S| - 1!}{(|S| - n - 1)!} \frac{(|S| - n)!}{|S|!} = \frac{|S| - n}{|S|} \end{aligned}$$

Which finally gives us

$$p(x \in B)_0 = (1 - p(x \notin B)_0) = \frac{n}{|S|}$$

## Calculating $p(x \in B)_0$ using binomial coefficients

There are  $\binom{|S|-1}{n-1}$  ways to pick  $x$ , and then  $n - 1$  more elements. There are, in total,  $\binom{|S|}{n}$  ways to pick  $n$  elements. Thus,

$$\begin{aligned} p(x \in B)_0 &= \frac{\binom{|S|-1}{n-1}}{\binom{|S|}{n}} = \frac{(|S| - 1)!}{(n - 1)!((|S| - 1) - (n - 1))!} \div \frac{|S|!}{n!(|S| - n)!} \\ &= \frac{(|S| - 1)!}{(n - 1)!(|S| - 1 - n + 1)!} \times \frac{n!(|S| - n)!}{|S|!} = \frac{(|S| - 1)!}{(|S| - n)!} \times \frac{(|S| - n)!}{|S|!} \\ &= \frac{(|S| - 1)n!}{|S|!(n - 1)!} = \frac{n}{|S|} \end{aligned}$$

## Some notation

For a given node  $B_i$  in the tree  $B$ ,  
 $b_i$  is the value at that node,  
 $|B_i|$  is the size of the sub-tree rooted at  $B_i$ .

## Calculating $p(x \in B)_1$

After checking the root node, which we will call  $B_1$ , we will have either found  $x$ , or we will not have. If we found  $x$ , we now know  $p(x \in B)_1 = 1$ , so we will consider the other case.

We must now calculate  $p(x \in B)_1 = p(x \in B | b_1 \neq x)$ .

Because the set  $B$  is a binary tree, the search algorithm will disregard one of  $B_1$ 's two children, reducing the size of the domain. We'll call  $B_1$ 's children  $B_L$  and  $B_R$ . Since all the values here are arbitrary, we can say without loss of generality that the algorithm will pick  $B_L$  as the next node to examine, which we will now label  $B_2$ , and further, we can claim that the algorithm has found  $x > b_1$ , since it would be the same either way.<sup>2</sup>

We have now ruled out every element in the set  $\{b_i | b_i \leq b_1\}$ , and we are ready

---

<sup>2</sup>If such a bold claim makes you uncomfortable, we can write  $x \oplus b_2$  where  $\oplus \in \{<, >\}$

for our calculation. Let's call the new domain  $S_1 = S - \{b_i | b_i \leq b_1\}$ . We can use the previous result to obtain

$$p(x \in B)_1 = \frac{|B_2|}{|S_1|}$$

That is, the size of the sub-tree  $B_L$ , divided by the size of the reduced domain  $S_1$ .

**In general,**  $p(x \in B)_i$

We now have a general pattern. The algorithm will examine a sequence of  $k$  nodes  $\{B_1, B_2, B_3 \dots B_k\}$ , as it does this, it will generate subsets of  $S$  according to the following:

$$S_n = S_{n-1} - \{b_i | b_i \leq b_1\}$$

.

## GARBAGE

The probability of not picking  $X$  is thus

$$\begin{aligned} p(x \notin B)_0 &= \prod_{i=0}^{n-1} \left[ \frac{|S| - (i+1)}{|S| - i} \right] = \frac{\prod_{i=0}^{n-1} |S| - (i+1)}{\prod_{i=0}^{n-1} |S| - i} = \frac{\prod_{i=0}^{n-1} (|S| - 1) - i}{\prod_{i=0}^{n-1} |S| - i} \\ &= \frac{(|S| - 1)!}{((|S| - 1) - n + 1)!} \frac{(|S| - n + 1)!}{|S|!} \\ &= \frac{(|S| - 1)! (|S| - n + 1)!}{(|S| - n + 1)! |S|!} \end{aligned}$$

—

We can calculate the probability of picking  $x$  by,

$$\begin{aligned} p(x \in B)_0 &= \prod_{i=0}^{n-1} \frac{1}{|S| - i} = \frac{1}{\prod_{i=0}^{n-1} |S| - i} = \frac{(|S| - (n-1) - 1)!}{|S|!} \\ &= \frac{(|S| - n)!}{|S|!} = \frac{1}{|S|! / (n-1)!} = \frac{(n-1)!}{|S|!} \end{aligned}$$

and through similar reasoning we can determine