



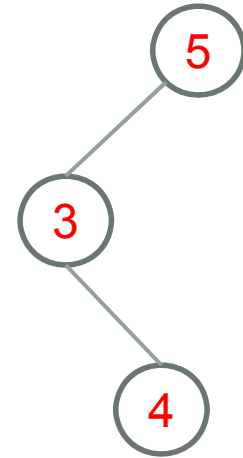
CSE 100: RUNTIME ANALYSIS: BST FIND

Average case analysis

- Warning! There will be math 😊
- Why is it important that we do this?
 - So you have a hope of doing it yourself on a new data structure (perhaps one you invent?)
 - Mathematical analysis can be insightful!

Given a BST having:

- N nodes $x_1, \dots, x_N$ such that $\text{key}(x_i) = k_i$
- Probability of searching for key k_i is p_i



What is the expected number of comparisons to find a key?

A. $\sum_{i=1}^N p_i (\text{No. of comparisons to find } k_i)$ =

B. $\sum_{i=1}^N p_i \cdot x_i$

C. $(\sum_{i=1}^N \text{No. of comparisons to find } k_i) / N$

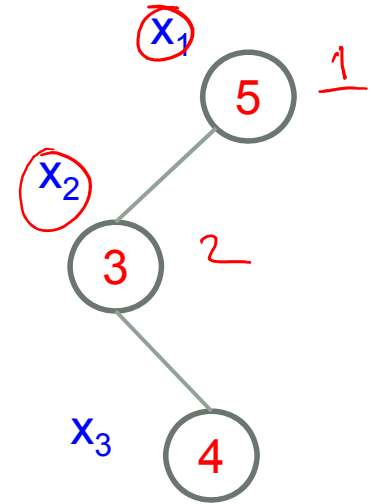
~~(1 - ...)~~

Number of compares to find key k_i is related to the **Depth** of x_i in the BST

- **Depth** of node x_i : No. of nodes on the path from the root to x_i inclusive
- Notation for depth of x_i :

$$d(x_i)$$

$$\# \text{ of compares} = d(x_i)$$



Probabilistic Assumption #1

- **Probabilistic Assumption #1:**

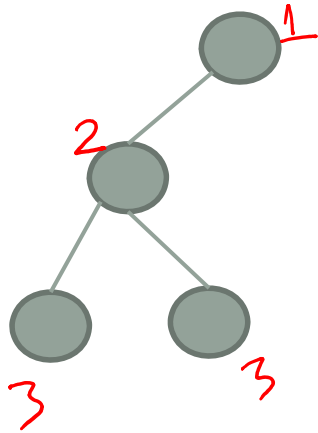
All keys are equally likely to be searched (how realistic is this)?

- Thus $p_1 = \dots = p_N = 1/N$ and the average number of comparisons in a successful find is:

$$\underline{D_{avg}(N)} = \sum_{i=1}^N \underline{p_i d(x_i)} = \sum_{i=1}^N \frac{1}{N} d(x_i) = \frac{1}{N} \left(\sum_{i=1}^N d(x_i) \right)$$

$$\sum_{i=1}^N d(x_i) = \underline{\text{total node depth}}$$

Calculating total node depth

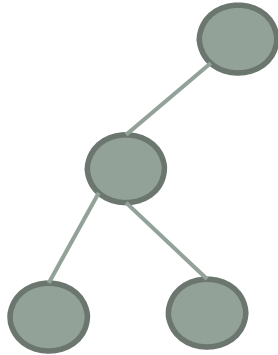


What is the total node depth of this tree?

- A. 3
- B. 5
- C. 6
- ☒ D. 9
- E. None of these

$$3 + 3 + 2 + 1 = 9$$

Calculating total node depth



- In a complete analysis of the average cases, we need to look at all possible BSTs that can be constructed with same set of N keys
- What does the structure of the tree relate to?

How many possible ways can we insert three elements into a BST?

- Suppose $N=3$ and the keys are (1, 3, 5)

How many possible ways can we insert three elements into a BST?

- Suppose $N=3$:

(1,3,5); (1,5,3); (3,1,5); (3,5,1); (5,1,3); (5,3,1)

6 possible trees

$$(N \times (N-1) \times \dots \times 1)$$

$$N!$$

$$\underline{3 \times 2 \times 1}$$

$$6$$

What is the total number of possibilities for an N-node BSTs?

A. N^N

☒ B. $N!$

C. e^N

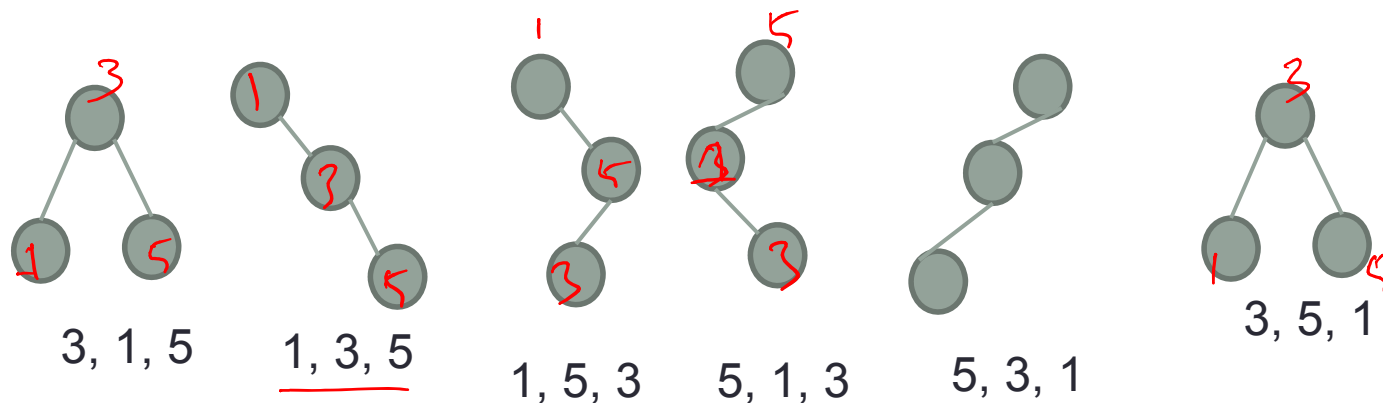
D. $N*N$

E. None of these

Relationship between order of insertion and structure of the BST

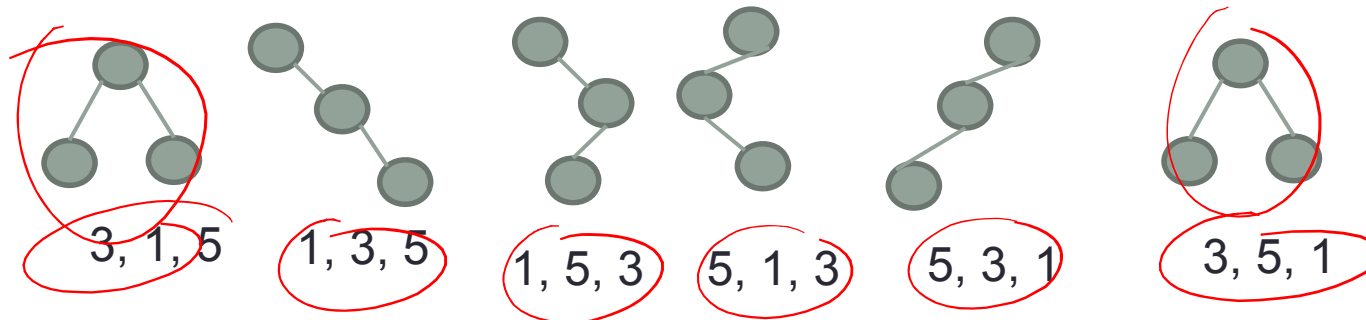
- Given a set of N keys: The structure of a BST constructed from those keys is determined by the order the keys are inserted
- Example: $N=3$. There are $N! = 3! = 6$ different orders of insertion of the 3 keys. Here are resulting trees:

($3!$)

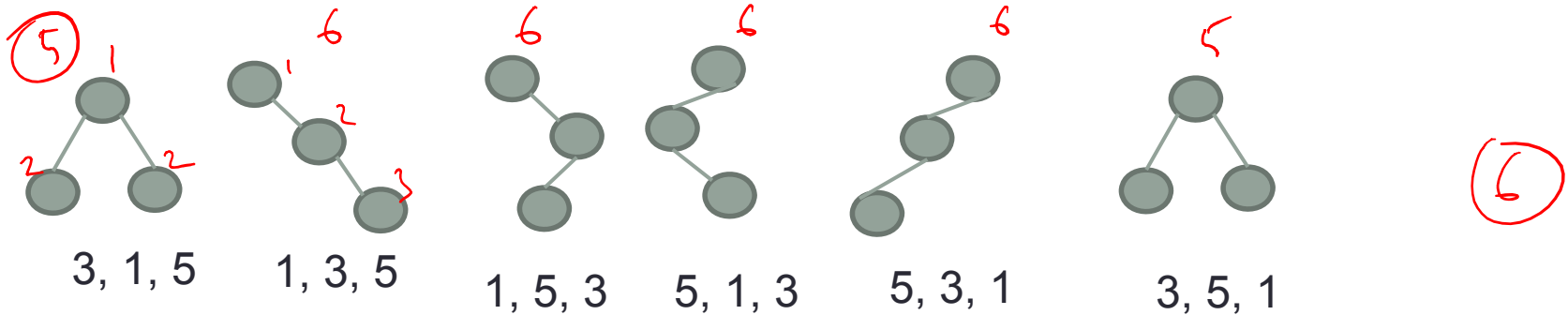


Probabilistic assumption #2

- We may assume that each key is equally likely to be the first key inserted; each remaining key is equally likely to be the next one inserted; etc.
- This leads to **Probabilistic Assumption #2**
Any insertion order (i.e. any permutation) of the keys is equally likely when building the BST
- This means with 3 keys, each of the following trees can occur with probability $1/6$



Average Case for successful Find: Brute Force Method



$$\sum p_i \cdot d(x_i)$$

$$= \frac{1}{N} \times \sum d(x_i)$$

$$= \frac{1}{6} \times (5 + 6 + 6 + 6 + 6 + 5)$$

$$=$$

Average # of comparisons in a single tree

- Let $D(N)$ be the expected total depth of BSTs with N nodes, over all the $N!$ possible BSTs, assuming that Probabilistic Assumption #2 holds

$$\begin{aligned} D(N) &= \sum_{\text{all BSTs } T_j \text{ with } N \text{ nodes}} (\text{probability of } T_j) (\text{Total Depth}(T_j)) \\ &= \sum_{\text{all BSTs with } N \text{ nodes}} \left(\frac{1}{N!} \right) \left(\sum_{i=1}^N d(x_i) \right) \end{aligned}$$

$N! = 10^6$

- If Assumption #1 also holds, the average # comparisons in a successful find is

$$\underline{D_{\text{avg}}(N)} = \underline{D(N)} / \underline{N}$$

The computationally intensive part is constructing $N!$ trees to compute $N!$ total depth values: This is a brute force method!

How do we compute $D(N)$?

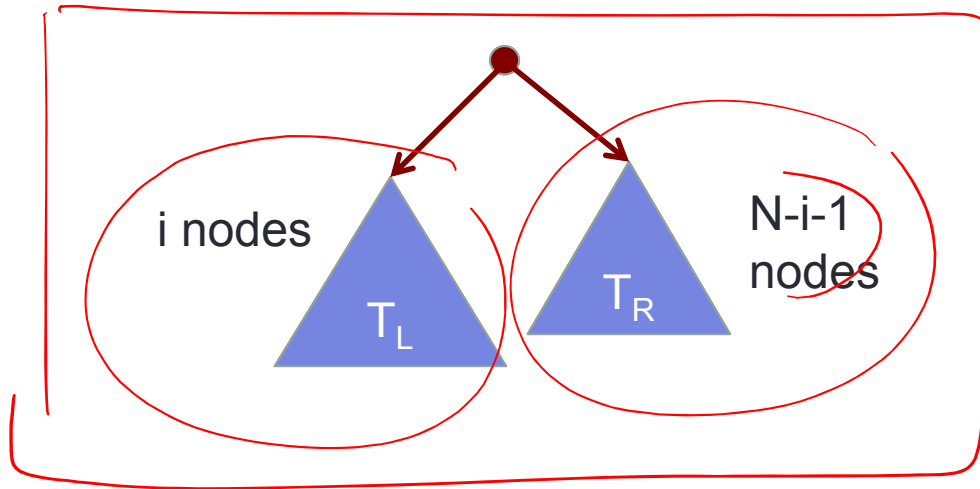
$$D(N) = \sum_{\text{all BSTs with } N \text{ nodes}} \left(\frac{1}{N!} \right) \left(\sum_{i=1}^N d(x_i) \right)$$

We need an equation for that does not involve computing $N!$ total depth values (in a brute force fashion)

Key Idea: We will build a recurrence relation for $D(N)$ in terms of $D(N-1)$
And then solve that recurrence relation to give us a sum over N
(instead of $N!$)

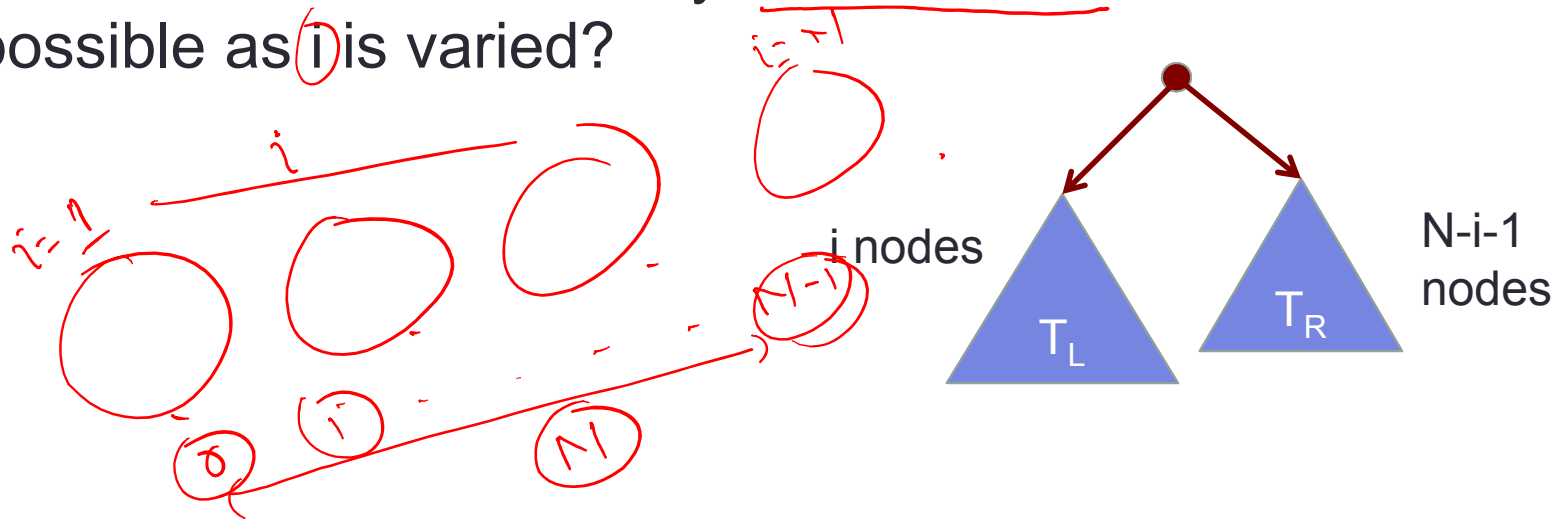
Towards a recurrence relation for average BST total depth

- Define $D(N|i)$ as expected total depth of a BST with N nodes, assuming that T_L has i nodes (and T_R has $N-i-1$)



Average case analysis of find in BST

- Given N nodes, how many such subsets of trees are possible as i is varied?



- ☒ A. N
- B. $N!$
- C. $\log_2 N$
- D. $(N-1)!$

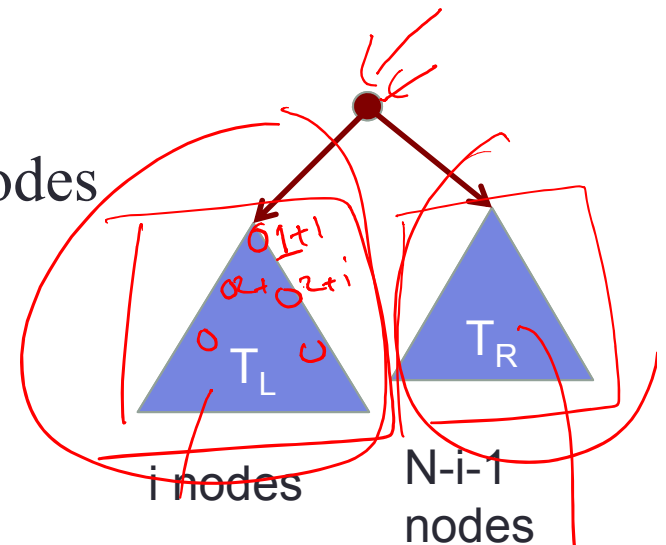
Probability of subtree sizes

- Let $P_N(i)$ = the probability that T_L has i nodes
- It follows that $D(N)$ is given by the following equation

$$D(N) = \sum_{i=0}^{N-1} P_N(i) D(N|i)$$

$$= \left(\frac{1}{N!} \right)$$

$$D(N|i)$$

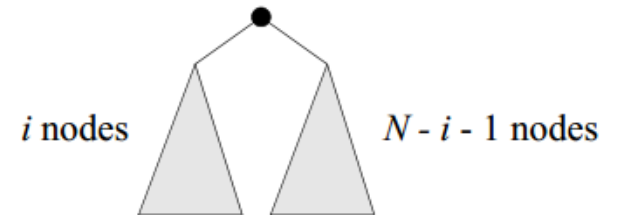


$$\frac{D(i) D(N-i-1)}{D(N)}$$

$$D(i) + D(N-i-1) + 1$$

Average total depth of a BST with N nodes

$$D(N) = \sum_{i=0}^{N-1} P_N(i) D(N|i)$$



$$D(N) = \sum_{i=0}^{N-1} \frac{1}{N} [D(i) + D(N-i-1) + N]$$

$$\sum_{i=0}^{N-1} \frac{N}{N} = N$$

$$= \frac{1}{N} \sum_{i=0}^{N-1} D(i) + \frac{1}{N} \sum_{i=0}^{N-1} D(N-i-1) + N$$

$$= \frac{1}{N} \sum_{j=N-1}^0 D(j) = \left[\frac{1}{N} \sum_{j=0}^{N-1} D(j) \right]$$

$j = (N-i-1)$

True or false: The term in the blue box is equal to the term in the red box

- A. True
- B. False

- Note that those two summations just add the same terms in different order; so

$$D(N) = \left(\frac{2}{N} \right) \sum_{i=0}^{N-1} D(i) + N$$

- ... and multiplying by N ,

$$ND(N) = 2 \sum_{i=0}^{N-1} D(i) + N^2$$

Handwritten expansion: $= 2(D(0) + \dots + D(N-1)) + (N^2)$

- Now substituting $N-1$ for N ,

$$(N-1)D(N-1) = 2 \sum_{i=0}^{N-2} D(i) + (N-1)^2$$

Handwritten expansion: $= 2(D(0) + \dots + D(N-2)) + (N-1)^2$

- Subtracting that equation from the one before it gives

$$ND(N) - (N-1)D(N-1) = 2D(N-1) + (N^2 - (N-1)^2)$$

- ... and collecting terms finally gives this recurrence relation on $D(N)$:

$$ND(N) = (N+1)D(N-1) + 2N-1$$

$$N * D(N) = (N+1) * D(N-1) + 2N - 1$$

How does this help us, again?

- A. We can solve it to yield a formula for $D(N)$ that does not involve $N!$
- B. We can use it to compute $D(N)$ directly
- C. I have no idea, I'm totally lost

$$ND(N) = (N+1)D(N-1) + 2N - 1$$

Through unwinding and some not-so-complicated algebra
(which you can find in your reading, a.k.a. Paul's slides)
we arrive at:

$$D(N) = 2(N+1) \sum_{i=1}^N \frac{1}{i} - 3N$$

No $N!$ to be seen! Yay!

And with a little more algebra, we can even show an approximation:

$$D_{avg}(N) \approx 1.386 \log_2 N$$

Conclusion: The average time to find an element in a BST with no restrictions on shape is $\Theta(\log N)$.



The importance of being balanced

- A binary search tree has average-case time cost for Find = $\Theta(\log N)$:

What does this analysis tell us:

- On an average things are not so bad provided assumptions 1 and 2 hold
- But the probabilistic assumptions we made often don't hold in practice
 - Assumption #1 may not hold: we may search some keys many more times than others
 - Assumption #2 may not hold: approximately sorted input is actually quite likely, leading to unbalanced trees with worst-case cost closer to $O(N)$ when N is large
- We would like our search trees to be balanced

The importance of being balanced

- We would like our search trees to be balanced
- Two kinds of approaches
 - Deterministic methods guarantee balance, but operations are somewhat complicated to implement (AVL trees, red black trees)
 - Randomized methods (treaps, skip lists) (insight from our result) – deliberate randomness in constructing the tree helps!!
 - Operations are simpler to implement
 - Balance not absolutely guaranteed, but achieved with high probability
 - We will return to this topic later in the course