

CSE 100: RUNTIME ANALYSIS: BST FIND

Average case analysis: BST

- 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!

Average case analysis of a “successful” find

The big question: Given a set of N distinct keys, what is the average run time of finding an element using a BST?

Main insights:

- Run time directly relates to number of compares needed to find any key
- Number of compares to find a key depends on
 - Location (specifically depth) of the key in a BST
 - Structure of the BST

3

4

5

Average case analysis of a “successful” find

The big question: Given a set of N distinct keys, what is the average run time of finding an element using a BST?

Approach:

- Given a BST, derive the expected number of compares to find any element
 - **Main result:** expected number of compares depends on the “**total depth**” of the tree (defined later), IF
“all keys are equally likely to be searched for” (Probability Assumption 1)
- To answer our original question we need to find the average number of compares over all BSTs that can be constructed with N keys.
 - Compute the **expected total depth** of all BSTs with N keys (Brute force is too expensive, so we will look for a recursive solution) We can find a nice recursive formula IF
“all orders of key insertions are equally likely”(Probability Assumption 2)

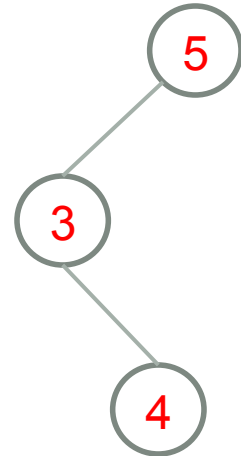
3

4

5

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

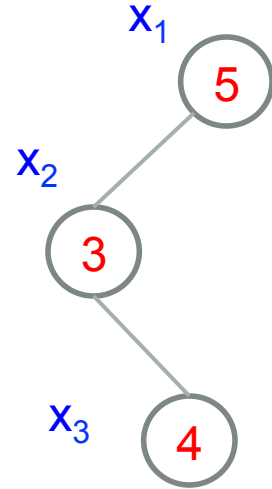
B. $\sum_{i=1}^N p_i \cdot k_i$ (Expected value of keys)

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

Expected no. of comparisons
if all keys are searched
for with equal probability

Number of compares to find key k_i is equal 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 : $d(x_i)$



$$D_{avg}(N) \Big|_{\text{Given BST}} = \sum_{i=1}^N p_i \cdot (\text{No. of comparisons to find node } x_i)$$

$$= \sum_{i=1}^N p_i d(x_i)$$

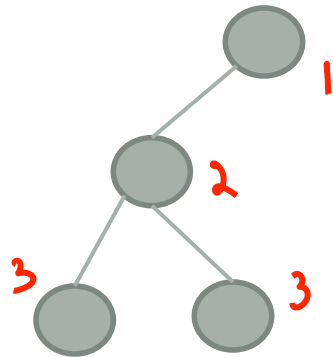
Probabilistic Assumption #1

- **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:

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

$$\sum_{i=1}^N d(x_i) = \text{total depth of the tree}$$

Calculating total depth



What is the total depth of this tree?

A. 3

B. 5

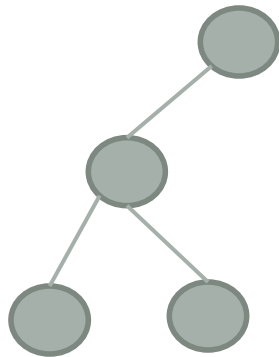
C. 6

☒ D. 9 (1+2+3+3)

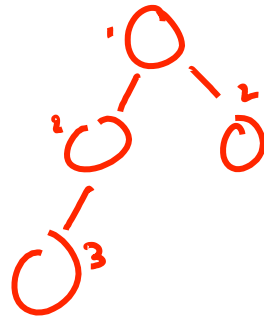
E. None of these

Calculating total depth

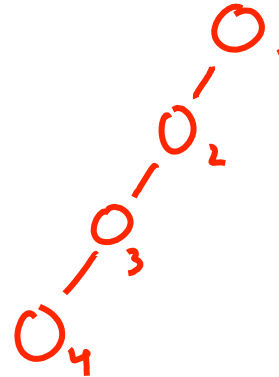
Tot depth = 9



Tot depth = 8



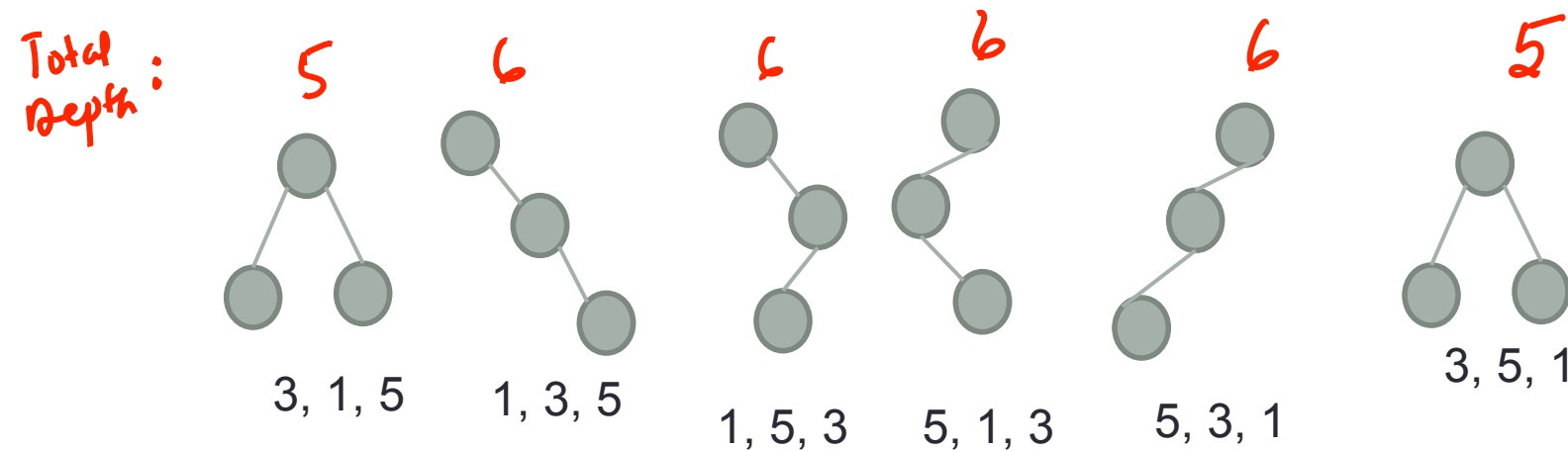
Tot depth = 10



- 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
- The total depth of the tree changes with the structure of the tree.
- What affects the structure of the tree?

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:

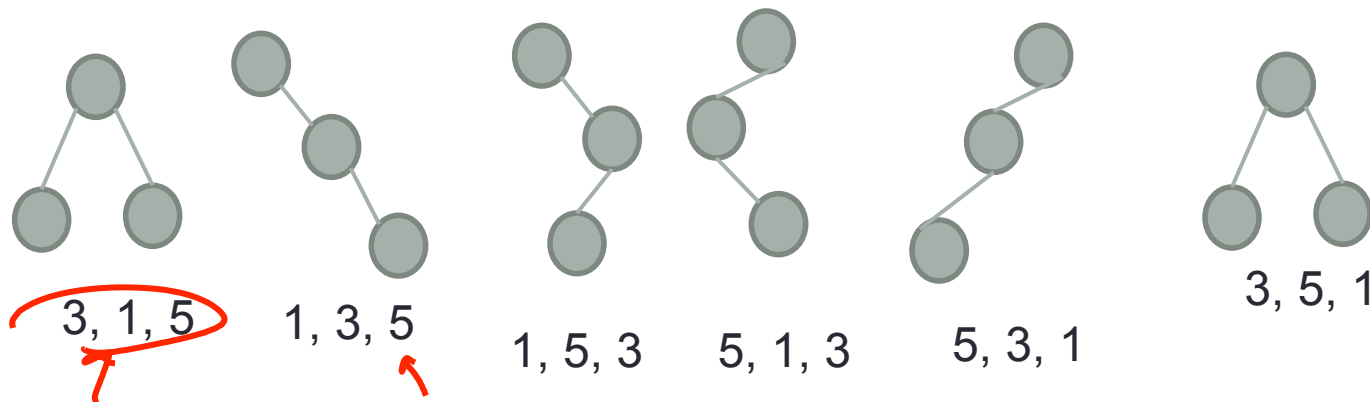


Need to find expected total depth

Relationship between order of insertion and structure of the BST

Q: What information is required to find the expected total depth of all BSTs with 3 keys

- A. The probability of each insertion order
- B. The probability of searching for any specific key
- C. No assumptions required



Need to find expected total depth

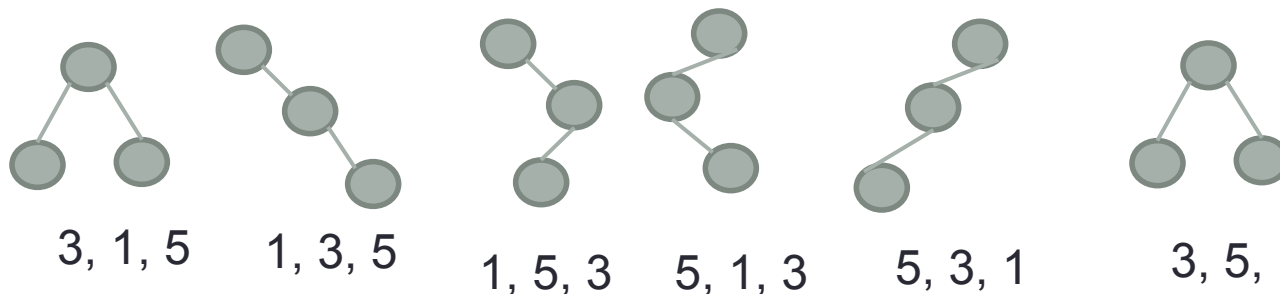
Probabilistic assumption #2

- Probabilistic Assumption #2**

Any insertion order (i.e. any permutation) of the keys is equally likely when building the BST

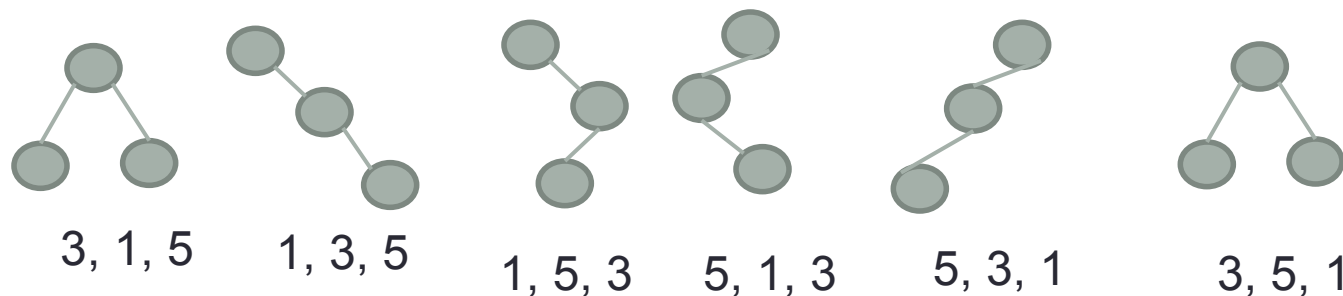
- Another way to put this is that each key is equally likely to be the first key inserted; each remaining key is equally likely to be the next one inserted; etc.
- This means with 3 keys, each of the following trees can occur with probability $1/6$

If probability Assumption 2 holds, probability of each order of insertion = $\frac{1}{N!}$



N key $\rightarrow N!$ orders of insertion $\Rightarrow N!$ possible BSTs (not all unique)

Average Case for successful Find: Brute Force Method



$D_{avg}(N)$: Average no of compares in a given BST = $\frac{\text{total depth}}{N}$

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

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

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

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

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

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

[Since we are calculating $D_{\text{avg}}(N)$ over all possible BSTs, we have replaced Total Depth by Expected Total Depth in the original formula.]

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

How do we compute $D(N)$?

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

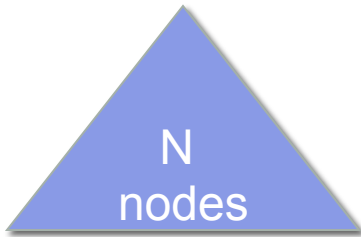
We need an equation for that does not involve computing $N!$ total depth values (in a brute force fashion)

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

- We are interested in finding:
- Assume we have solved the smaller versions of the problem

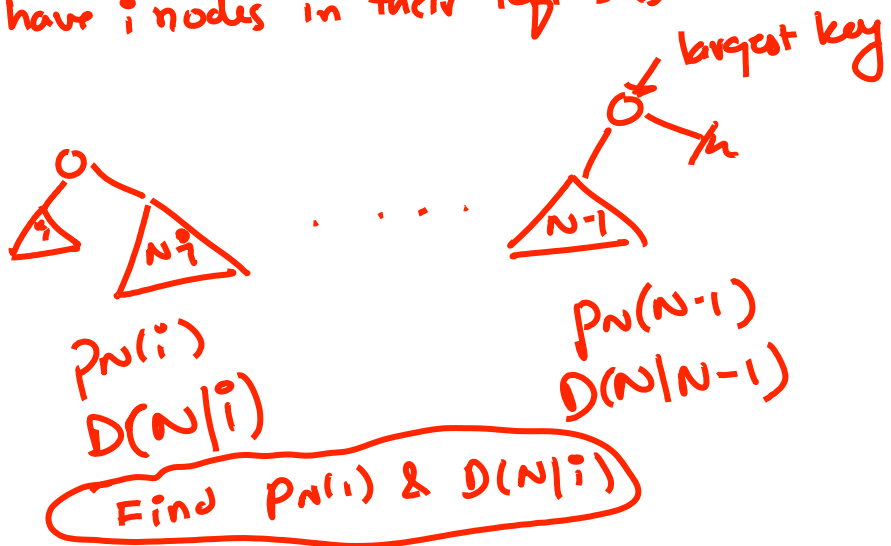
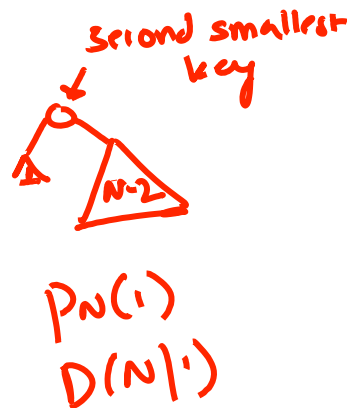
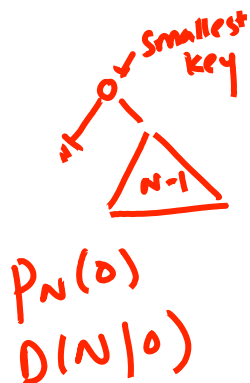
So, we know $D(i) \forall i < N$



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

$P_n(i)$: probability of all BSTs with n keys that have i nodes in their left subtree
 $D(n|i)$: Expected total depth of all BSTs with n nodes that have i nodes in their left subtree

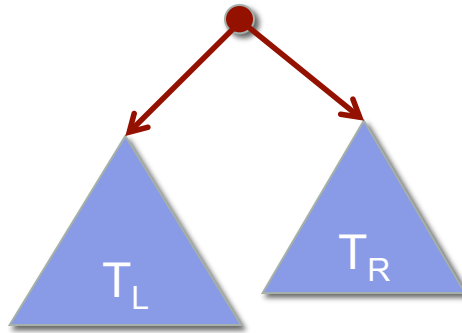
Break down into N smaller sub problems



Towards a recurrence relation for average BST total depth

Define the following sub-problem:

Find the expected depth of all trees where the root node is the i^{th} smallest key and the rest of the keys can be organized in any fashion in the left and right subtrees of the root

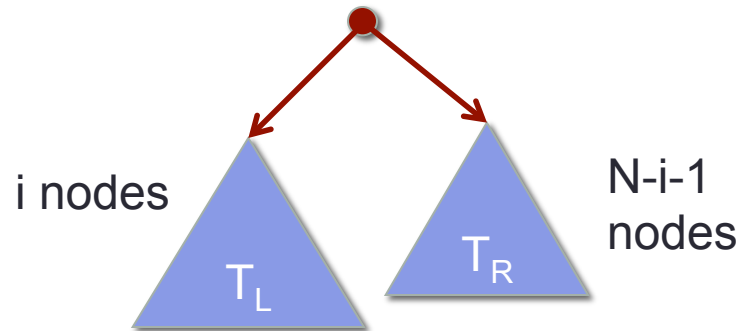


Which of the following best describes the consequence of fixing the root to be the i^{th} smallest key :

- A. We have described our original problem in terms of a smaller version of the problem (almost there)
- ☒ B. We are restricted to trees with a fixed number of nodes in the left and right subtrees of the root
- C. We can describe our original problem as $(N-1)!$ such sub problems

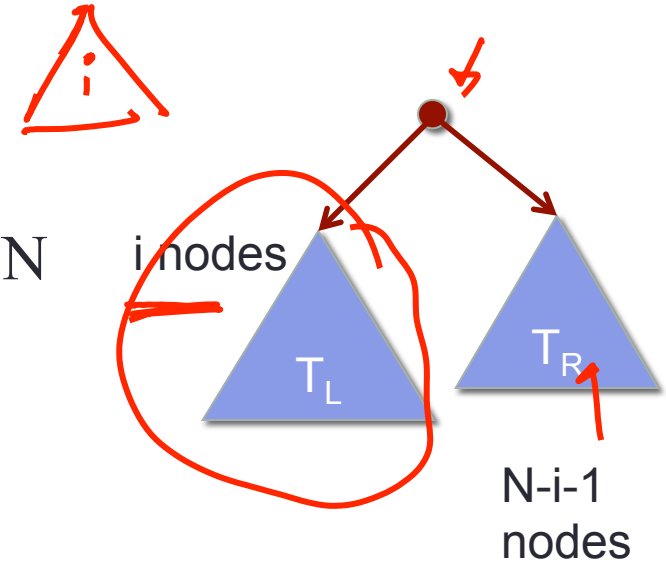
Towards a recurrence relation for average BST total depth

- 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 $N-i-1$)



Towards a recurrence relation for average BST total depth

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



What is $D(N|i)$ in terms of $D(i)$ & $D(N-i-1)$?

Hint: all nodes in each subtree are 1 deeper in tree T

- A. $D(i) + D(N-i-1)$
- B. $D(i) + D(N-i-1) + 1$
- ☒ C. $D(i) + D(N-i-1) + N$

$$\triangle_i \rightarrow D(i)$$

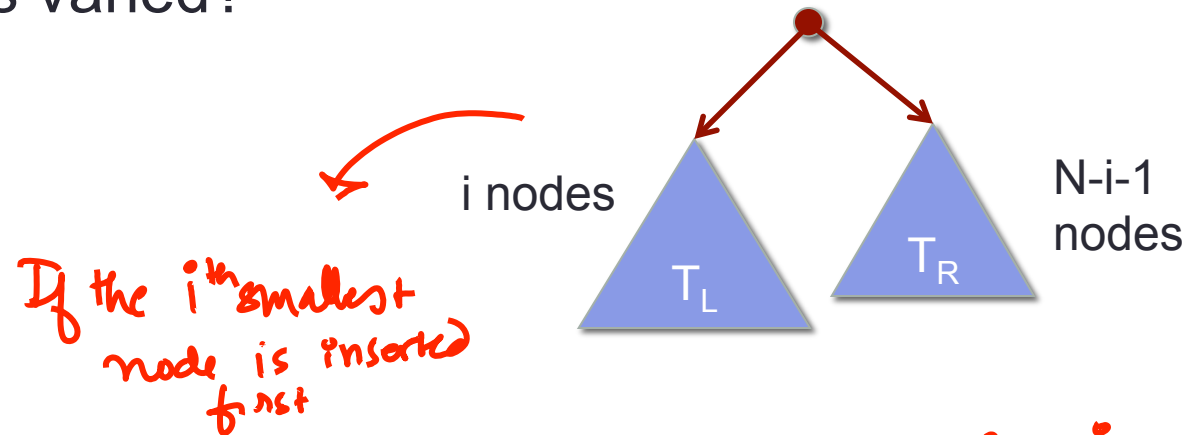
$$\triangle_i \rightarrow D(i) + i$$

$$D(N|i) = \text{Expected Depth}(T_L) + 1 + \text{Expected Depth}(T_R)$$

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

Average case analysis of find in BST

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



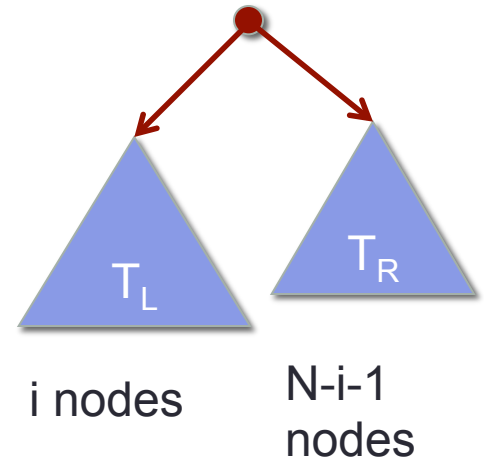
- ☒ A. N
- ☐ B. $N!$
- ☐ C. $\log_2 N$
- ☐ D. $(N-1)!$

Among N nodes any one can be the first insert.
So, N such subclasses of trees.
And probability of i^{th} smallest node being inserted first is $\frac{1}{N}$

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

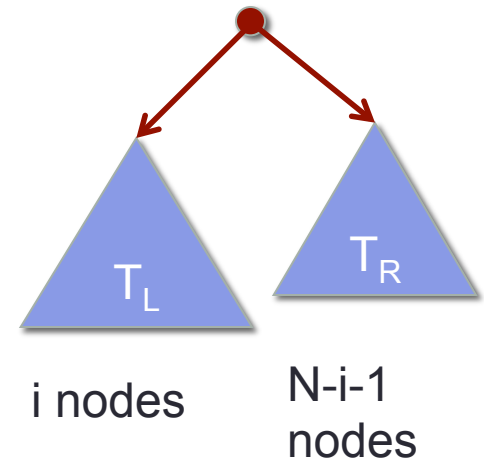
$$D(N) = \sum_{i=0}^{N-1} P_N(i) D(N-i-1)$$



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

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



What is the value of $P_N(i)$?

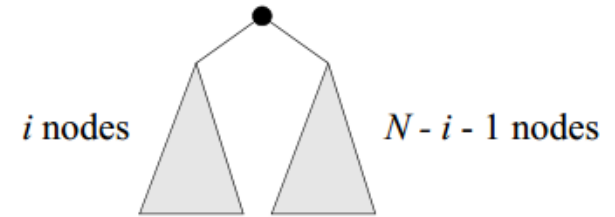
Hint: use assumption #2, any of the N keys are equally likely to be inserted first

☒ A. $1/N$

☐ B. It depends on i and N

Average total depth of a BST with N nodes

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



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

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

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

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

- ... and multiplying by N ,

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

- Now substituting $N-1$ for N ,

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

- 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