6. Trees
Review

First half of class

- Introduced analysis of algorithms.
- Surveyed basic mathematics needed for scientific studies.
- Introduced analytic combinatorics.

<table>
<thead>
<tr>
<th></th>
<th>Analysis of Algorithms</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>Recurrences</td>
</tr>
<tr>
<td>3</td>
<td>Generating Functions</td>
</tr>
<tr>
<td>4</td>
<td>Asymptotics</td>
</tr>
<tr>
<td>5</td>
<td>Analytic Combinatorics</td>
</tr>
</tbody>
</table>

Note: Many applications beyond analysis of algorithms.
### Orientation

**Second half of class**
- Surveys fundamental combinatorial classes.
- Considers techniques from analytic combinatorics to study them.
- Includes applications to the analysis of algorithms.

<table>
<thead>
<tr>
<th>chapter</th>
<th>combinatorial classes</th>
<th>type of class</th>
<th>type of GF</th>
</tr>
</thead>
<tbody>
<tr>
<td>6</td>
<td>Trees</td>
<td>unlabeled</td>
<td>OGFs</td>
</tr>
<tr>
<td>7</td>
<td>Permutations</td>
<td>labeled</td>
<td>EGFs</td>
</tr>
<tr>
<td>8</td>
<td>Strings and Tries</td>
<td>unlabeled</td>
<td>OGFs</td>
</tr>
<tr>
<td>9</td>
<td>Words and Mappings</td>
<td>labeled</td>
<td>EGFs</td>
</tr>
</tbody>
</table>

Note: Many more examples in book than in lectures.
6. Trees

- Trees and forests
- Binary search trees
- Path length
- Other types of trees
Definition. A binary tree is an external node or an internal node and two binary trees.
Binary tree enumeration (quick review)

How many binary trees with $N$ nodes?

$T_1 = 1$

$T_2 = 2$

$T_3 = 5$

$T_4 = 14$
Symbolic method: binary trees

How many binary trees with $N$ nodes?

<table>
<thead>
<tr>
<th>Class</th>
<th>$T$, the class of all binary trees</th>
</tr>
</thead>
<tbody>
<tr>
<td>Size</td>
<td>$</td>
</tr>
<tr>
<td>OGF</td>
<td>$T(z) = \sum_{t \in T} z^{</td>
</tr>
</tbody>
</table>

*Atoms*

<table>
<thead>
<tr>
<th>type</th>
<th>class</th>
<th>size</th>
<th>GF</th>
</tr>
</thead>
<tbody>
<tr>
<td>external node</td>
<td>$Z_\square$</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>internal node</td>
<td>$Z_\bullet$</td>
<td>1</td>
<td>$z$</td>
</tr>
</tbody>
</table>

**Construction**

$T = Z_\square + T \times Z_\bullet \times T$

**OGF equation**

$T(z) = 1 + zT(z)^2$

$[z^N] T(z) = \frac{1}{N+1} \binom{2N}{N} \sim \frac{4^N}{\sqrt{\pi N^3}}$
Forest and trees

Each forest with $N$ nodes corresponds to a tree with $N+1$ nodes.

$[z^N]F(z) = [z^{N+1}]G(z)$

$zF(z) = G(z)$

GF that enumerates forests

GF that enumerates trees

A tree with $N+1$ nodes

add a root
**Anatomy of a (general) tree**

**Definition.** A *forest* is a sequence of disjoint trees.

**Definition.** A *tree* is a node (called the *root*) connected to the roots of trees in a forest.
Forest enumeration

How many forests with $N$ nodes?

$F_1 = 1$

$F_2 = 2$

$F_3 = 5$

$F_4 = 14$
Tree enumeration

How many trees with $N$ nodes?

$G_1 = 1$

$G_2 = 1$

$G_3 = 2$

$G_3 = 5$

$G_4 = 14$
Symbolic method: forests and trees

How many forests and trees with $N$ nodes?

<table>
<thead>
<tr>
<th>Class</th>
<th>$F$, the class of all forests</th>
</tr>
</thead>
<tbody>
<tr>
<td>Size</td>
<td>$</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Class</th>
<th>$G$, the class of all trees</th>
</tr>
</thead>
<tbody>
<tr>
<td>Size</td>
<td>$</td>
</tr>
</tbody>
</table>

**Atoms**

<table>
<thead>
<tr>
<th>type</th>
<th>class</th>
<th>size</th>
<th>GF</th>
</tr>
</thead>
<tbody>
<tr>
<td>node</td>
<td>$Z$</td>
<td>1</td>
<td>$z$</td>
</tr>
</tbody>
</table>

**Construction**

$F = SEQ(G)$ and $G = Z \times F$

**OGF equations**

$F(z) = \frac{1}{1 - G(z)}$ and $G(z) = zF(z)$

**Solution**

$F(z) - zF(z)^2 = 1$

**Extract coefficients**

$F_N = T_N = \frac{1}{N+1} \binom{2N}{N} \sim \frac{4^N}{\sqrt{\pi N^3}}$

$G_N = F_{N-1} \sim \frac{4^{N-1}}{\sqrt{\pi N^3}}$
Forest and binary trees

Each forest with $N$ nodes corresponds to

Connect each node to its
- left child
- right sibling

"rotation" correspondence

A binary tree with $N$ nodes
Aside: Drawing a binary tree

Approach 1:
- y-coordinate: height minus node depth
- x-coordinate: inorder node rank

Problem: distracting long edges

Design decision:
Reduce visual clutter by omitting external nodes
Aside: Drawing a binary tree

Approach 2:

• y-coordinate: height minus node depth
• x-coordinate: centered and evenly spaced by level

Drawing shows tree profile
Typical random binary tree shapes (400 nodes)

Challenge: characterize analytically
6. Trees

• Trees and forests
• **Binary search trees**
• Path length
• Other types of trees
**Binary search tree (BST)**

Fundamental data structure in computer science:
- Each node has a **key**, with comparable values.
- Keys are all distinct.
- Each node’s **left** subtree has **smaller** keys.
- Each node’s **right** subtree has **larger** keys.

Section 3.2
BST representation in Java

Java definition: A BST is a reference to a root Node.

A Node is comprised of four fields:
• A Key and a Value.
• A reference to the left and right subtree.

Notes:
• Key and Value are generic types.
• Key is Comparable.

```java
private class Node {
    private Key key;
    private Value val;
    private Node left, right;
    public Node(Key key, Value val) {
        this.key = key;
        this.val = val;
    }
}
```
BST implementation (search)

```java
class BST<Key extends Comparable<Key>, Value>
{
    private Node root;

    private class Node
    {
        /* see previous slide */
    }

    public Value get(Key key)
    {
        Node x = root;
        while (x != null)
        {
            int cmp = key.compareTo(x.key);
            if      (cmp  < 0) x = x.left;
            else if (cmp  > 0) x = x.right;
            else if (cmp == 0) return x.val;
        }
        return null;
    }

    public void put(Key key, Value val)
    {
        /* see next slide */
    }
}
```

Diagram:
- To search for M:
  - Go left
  - Then right
  - Successful!

- To search for Q:
  - Go left
  - Then right
  - Then right
  - Unsuccessful
BST implementation (insert)

```java
public void put(Key key, Value val)
{  root = put(root, key, val);  }

private Node put(Node x, Key key, Value val)
{  
   if (x == null) return new Node(key, val);
   int cmp = key.compareTo(x.key);
   if      (cmp  < 0) x.left  = put(x.left,  key, val);
   else if (cmp  > 0) x.right = put(x.right, key, val);
   else if (cmp == 0) x.val = val;
   return x;
}
```

concise, but tricky, recursive code

to insert Q

go left

then right

then right

then attach Q here
Key fact

The shape of a BST depends on the order of insertion of the keys.

Best case

Typical case

Worst case

search cost guaranteed $\sim \lg N$

Average search cost $\sim N/2$ (a problem)

Reasonable model: Analyze BST built from inserting keys in random order.
Typical random BSTs (80 nodes)

Challenge: characterize analytically (explain difference from random binary trees)
BST shape

is a property of permutations, not trees (!)

Note: Balanced shapes are more likely.
Mapping permutations to trees via BST insertion

Q. How many permutations map to this tree?

A. 2

Q. How many permutations map to this tree?

A. \( \binom{5}{2} \cdot 2 \cdot 1 = 20 \)

"result in this tree shape when inserted into an initially empty BST"

perms mapping to left subtree
perms mapping to right subtree

ways to mix left and right

root must be 4

1, 2, and 3 on the left
5 and 6 on the right

1 2 3 5 6
4 2 5 3 1 6
4 2 5 1 3 6
4 2 5 1 6 3
4 2 5 6 1 3
4 2 5 6 3 1
4 5 2 1 3 6
4 5 2 1 6 3
4 5 2 6 1 3
4 5 2 6 3 1
4 5 6 2 1 3
4 5 6 2 3 1
**Mapping permutations to trees via BST insertion**

**Q.** How many permutations map to a general binary tree $t$?

**A.** Let $P_t$ be the number of perms that map to $t$

\[
P_t = \left(\frac{|t_L| + |t_R|}{|t_L|}\right) \cdot P_{t_L} \cdot P_{t_R}
\]

much, much larger when $t_L \approx t_R$ than when $t_L \ll t_R$

(explains why balanced shapes are more likely)
Two binary tree models

that are fundamental (and fundamentally different)

**BST model**
- Balanced shapes much more likely.
- Probability root is of rank $k$: $1/N$.

**Catalan model**
- Each tree shape equally likely.
- Probability root is of rank $k$:
  $$\frac{1}{k} \binom{2k-2}{k} \frac{1}{N-k+1} \binom{2N-2k}{N-k} \frac{1}{N+1} \binom{2N}{N}$$
Catalan distribution

Probability that the root is of rank $k$ in a randomly-chosen binary tree with $N$ nodes.

Note: Small subtrees are extremely likely.

Ex. Probability that at least one of the two subtrees is empty: $\sim 1/2$
Aside: Generating random binary trees

```java
public class RandomBST {
    private Node root;
    private int h;
    private int w;

    private class Node {
        private Node left, right;
        private int N;
        private int rank, depth;
    }

    public RandomBST(int N) {
        root = generate(N, 0);
    }

    private Node generate(int N, int d) {
        // See code at right.
    }

    public static void main(String[] args) {
        int N = Integer.parseInt(args[0]);
        RandomBST t = new RandomBST(N);
        t.show();
    }
}
```

Note: “rank” field includes external nodes: x.rank = 2*k+1

```java
private Node generate(int N, int d) {
    Node x = new Node();
    x.N = N; x.depth = d;
    if (h < d) h = d;
    if (N == 0) x.rank = w++; else {
        int k = StdRandom.uniform(N)+1;
        x.left = generate(k-1, d+1);
        x.rank = w++;
        x.right = generate(N-k, d+1);
    }
    return x;
}
```

random BST: StdRandom.uniform(N)+1
random binary tree: StdRandom.discrete(cat[N]) + 1;

stay tuned
Aside: Drawing binary trees

```java
public void show()
{  show(root);  }

private double scaleX(Node t)
{  return 1.0*t.rank/(w+1);  }
private double scaleY(Node t)
{  return 3.0*(h - t.depth)/(w+1);  }

private void show(Node t)
{
  if (t.N == 0) return;
  show(t.left);
  show(t.right);
  double x = scaleX(t);
  double y = scaleY(t);
  double xl = scaleX(t.left);
  double yl = scaleY(t.left);
  double xr = scaleX(t.right);
  double yr = scaleY(t.right);
  StdDraw.filledCircle(x, y, .005);
  StdDraw.line(x, y, xl, yl);
  StdDraw.line(x, y, xr, yr);
}
```

Exercise: Implement "centered by level" approach.
6. Trees

- Trees and forests
- Binary search trees
- Path length
- Other types of trees
**Definition.** A *binary tree* is an external node or an internal node and two binary trees.

*internal path length:* \( ipl(t) = \sum_{k \geq 0} k \cdot \{\# \text{ internal nodes at depth } k\} \)

*external path length:* \( xpl(t) = \sum_{k \geq 0} k \cdot \{\# \text{ external nodes at depth } k\} \)
Path length in binary trees

<table>
<thead>
<tr>
<th>notation</th>
<th>definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>$t$</td>
<td>binary tree</td>
</tr>
<tr>
<td>$</td>
<td>t</td>
</tr>
<tr>
<td>$\mathbf{t}$</td>
<td># external nodes in $t$</td>
</tr>
<tr>
<td>$t_L$ and $t_R$</td>
<td>left and right subtrees of $t$</td>
</tr>
<tr>
<td>$ipl(t)$</td>
<td>internal path length of $t$</td>
</tr>
<tr>
<td>$xpl(t)$</td>
<td>external path length of $t$</td>
</tr>
</tbody>
</table>

**Recursive relationships**

\[
|t| = |t_L| + |t_R| + 1 \\
\mathbf{t} = t_L + t_R \\

ipl(t) = ipl(t_L) + ipl(t_R) + |t| - 1 \\
xpl(t) = xpl(t_L) + xpl(t_R) + |t|
\]

**Lemma 1.** \(\mathbf{t} = |t| + 1\)

*Proof.* Induction.

\[
\mathbf{t} = t_L + t_R = |t_L| + 1 + |t_R| + 1 = |t| + 1
\]

**Lemma 2.** \(xpl(t) = ipl(t) + 2|t|\)

*Proof.* Induction.

\[
xpl(t) = xpl(t_L) + xpl(t_R) + |t| = ipl(t_L) + 2|t_L| + ipl(t_R) + 2|t_R| + |t| + 1 = ipl(t) + 2|t|
\]
Problem 1: What is the expected path length of a random binary tree?

\[ Q_{Nk} = \# \text{ trees with } N \text{ nodes and ipl } k \]

\[ T_N = \# \text{ trees} \]

\[ Q_N = \text{cumulated cost (total ipl)} \]

\[ Q_{10} = 1 \]

\[ Q_{1} = 0 \]

\[ Q_{1}/T_1 = 0 \]

\[ T_1 = 1 \]

\[ Q_{21} = 2 \]

\[ Q_{2} = 2 \]

\[ Q_{2}/T_2 = 1 \]

\[ Q_{32} = 1 \]

\[ Q_{33} = 4 \]

\[ T_3 = 2 \]

\[ Q_3 = 1 \cdot 2 + 4 \cdot 3 = 14 \]

\[ Q_3/T_3 = 2.8 \]

\[ Q_{44} = 4 \]

\[ T_4 = 14 \]

\[ Q_{45} = 2 \]

\[ Q_4 = 4 \cdot 4 + 2 \cdot 5 + 8 \cdot 6 = 74 \]

\[ Q_{46} = 8 \]

\[ Q_4/T_4 \doteq 5.286 \]
Average path length in a random binary tree

$T$ is the set of all binary trees.
$|t|$ is the number of internal nodes in $t$.
$\text{ipl}(t)$ is the internal path length of $t$.
$T_N$ is the # of binary trees of size $N$ (Catalan).
$Q_N$ is the total ipl of all binary trees of size $N$.

Counting GF.  
\[ T(z) = \sum_{t \in \mathcal{T}} z^{|t|} = \sum_{N \geq 0} T_N z^N = \sum_{N \geq 0} \frac{1}{N + 1} \binom{2N}{N} z^N \sim \frac{4^N}{\sqrt{\pi N^3}} \]

Cumulative cost GF.  
\[ Q(z) = \sum_{t \in \mathcal{T}} \text{ipl}(t) z^{|t|} \]

Average ipl of a random $N$-node binary tree.  
\[ \left[ z^N \right] Q(z) = \left[ z^N \right] \frac{Q(z)}{T_N} \]

Next: Derive a functional equation for the CGF.
CGF functional equation for path length in binary trees

Counting GF.

\[ T(z) = \sum_{t \in T} z^{|t|} \]

CGF.

\[ Q(z) = \sum_{t \in T} ipl(t)z^{|t|} \]

Decompose from definition.

\[ Q(z) = 1 + \sum_{t_L \in T} \sum_{t_R \in T} (ipl(t_L) + ipl(t_R) + |t_L| + |t_R|)z^{|t_L|+|t_R|+1} \]

\[ = 1 + 2zQ(z)T(z) + 2z^2 T'(z)T(z) \]
Expected path length of a random binary tree: full derivation

\[ Q(z) = \sum_{t \in T} ipl(t)z^{|t|} \]

Decompose from definition.

\[ Q(z) = 1 + \sum_{t_L \in T} \sum_{t_R \in T} (ipl(t_L) + ipl(t_R) + |t_L| + |t_R|)z^{|t_L|+|t_R|+1} \]

\[ = 2zT(z)(Q(z) + zT'(z)) \]

Solve.

\[ Q(z) = \frac{2z^2T(z)T'(z)}{1 - 2zT(z)} \]

Do some algebra (omitted)

\[ zQ(z) = \frac{z}{1 - 4z} - \frac{1 - z}{\sqrt{1 - 4z}} + 1 \]

Expand.

\[ Q_N \equiv [z^N]Q(z) \sim 4^N \]

Compute average internal path length.

\[ \frac{Q_N}{T_N} \sim N\sqrt{\pi N} \]
Problem 2: What is the expected path length of a random BST?

\[ C_{Nk} = \text{# \textit{permutations} resulting in a BST with } N \text{ nodes and ipl } k \]
\[ N! = \text{# \textit{permutations}} \]
\[ C_N = \text{cumulated cost (total ipl)} \]

Recall: A property of \textit{permutations}.

\[ C_{10} = 1 \]
\[ C_1 = 0 \]
\[ C_1/1! = 0 \]
\[ C_2 = 2 \]
\[ C_2/2! = 1 \]
\[ C_3 = 2 \cdot 2 + 4 \cdot 3 = 16 \]
\[ C_3/3! \approx 2.667 \]
\[ C_4 = 12 \cdot 4 + 4 \cdot 5 + 8 \cdot 6 = 74 \]
\[ C_4/4! \approx 4.833 \]
Average path length in a BST built from a random permutation

\( P \) is the set of all permutations.
\(|p|\) is the length of \( p \).
\( \text{ipl}(p) \) is the ipl of the BST built from \( p \) by inserting into an initially empty tree.
\( P_N \) is the # of permutations of size \( N \) (\( N! \)).
\( C_N \) is the total ipl of BSTs built from all permutations.

Counting EGF.

\[
P(z) = \sum_{p \in P} \frac{z^{\mid p \mid}}{|p|!} = \sum_{N \geq 0} \frac{N^N}{N!} = \frac{1}{1 - z}
\]

Cumulative cost EGF.

\[
C(z) = \sum_{p \in P} \text{ipl}(p) \frac{z^{\mid p \mid}}{|p|!}
\]

Expected ipl of a BST built from a random permutation.

\[
\frac{N! [z^N] C(z)}{[z^N] P(z)} = \frac{N! [z^N] C(z)}{N!} = [z^N] C(z)
\]

Next: Derive a functional equation for the cumulated cost EGF.
CGF functional equation for path length in BSTs

Cumulative cost EGF.

\[ C(z) = \sum_{p \in P} \ipl(p) \frac{z^{|p|}}{|p|!} \]

Counting GF.

\[ P(z) = \sum_{p \in P} \frac{z^{|p|}}{|p|!} = \frac{1}{1 - z} \]

Decompose.

\[ C(z) = \sum_{p_l \in \mathcal{P}} \sum_{p_r \in \mathcal{P}} \left( \frac{|p_l| + |p_r|}{|p_l|} \right) \frac{z^{|p_l| + |p_r| + 1}}{(|p_l| + |p_r| + 1)!} (\ipl(p_l) + \ipl(p_r) + |p_l| + |p_r|) \]

Differentiate.

\[ C'(z) = \sum_{p_l \in \mathcal{P}} \sum_{p_r \in \mathcal{P}} \frac{z^{|p_l|} z^{|p_r|}}{|p_l|! |p_r|!} (\ipl(p_l) + \ipl(p_r) + |p_l| + |p_r|) \]

\[ = 2C(z)P(z) + 2zP'(z)P(z) = \frac{2C(z)}{1 - z} + \frac{2z}{(1 - z)^3} \]

\[ P(z) = \sum_{p \in P} \frac{z^{|p|}}{|p|!} = \frac{1}{1 - z} \]

\[ P'(z) = \sum_{p \in P} \frac{z^{|p|-1}}{|p| - 1)!} = \frac{1}{(1 - z)^2} \]
CGF functional equation for path length in BSTs

\[ C'(z) = \frac{2C(z)}{1-z} + \frac{2z}{(1-z)^3} \]

Look familiar?

**Solving the Quicksort recurrence with OGFs**

\[ C_N = N + 1 + \frac{2}{N} \sum_{1 \leq k \leq N} C_{k-1} \]

Multiply both sides by \( N \).

\[ NC_N = N(N + 1) + 2 \sum_{1 \leq k \leq N} C_{k-1} \]

Multiply by \( z^N \) and sum.

\[ \sum_{N \geq 1} NC_N z^N = \sum_{N \geq 1} N(N + 1)z^N + 2 \sum_{N \geq 1} \sum_{1 \leq k \leq N} C_{k-1} z^N \]

Evaluate sums to get an ordinary differential equation

\[ C'(z) = \frac{2}{(1-z)^3} + \frac{2C(z)}{1-z} \]

Solve the ODE.

\[ (1-z)^2 C'(z) = (1-z)^2 C'(z) - 2(1-z)C(z) \]

\[ (1-z)^2 \left( \frac{C'(z)}{1-z} - 2 \left( \frac{C(z)}{1-z} \right) \right) = \frac{2}{1-z} \]

Integrate.

\[ C(z) = \frac{2}{(1-z)^2} \ln \frac{1}{1-z} \]

Expand.

\[ C_N = [z^N] \left( \frac{2}{(1-z)^2} \ln \frac{1}{1-z} \right) = 2(N+1)(H_{N+1} - 1) \]
Expected path length in BST built from a random permutation: full derivation

CGF.

\[ C(z) = \sum_{p \in P} ipl(p) \frac{z^{|p|}}{|p|!} \]

Decompose.

\[ C(z) = \sum_{p_L \in P} \sum_{p_R \in P} \left( \frac{|p_L| + |p_R|}{|p_L|} \right) \frac{z^{|p_L|+|p_R|+1}}{(|p_L| + |p_R| + 1)!} (ipl(p_L) + ipl(p_R) + |p_L| + |p_R|) \]

Differentiate.

\[ C'(z) = \sum_{p_L \in P} \sum_{p_R \in P} \frac{z^{|p_L|}}{|p_L|!} \frac{z^{|p_R|}}{|p_R|!} \left( ipl(p_L) + ipl(p_R) + |p_L| + |p_R| \right) \]

Simplify.

\[ = 2C(z)P(z) + 2zP'(z)P(z) \]

\[ = \frac{2C(z)}{1-z} + \frac{2z}{(1-z)^3} \]

Solve the ODE (see GF lecture).

\[ C(z) = \frac{2}{(1-z)^2} \ln \frac{1}{1-z} - \frac{2z}{(1-z)^2} \]

Expand.

\[ C_N = 2(N + 1)(H_{N+1} - 1) - 2N \sim 2N \ln N \]
BST – quicksort bijection

Quicksort

- first entry in a permutation (partitioning element)

BST

- node corresponding to first entry in a permutation

Average # compares for quicksort

- = average external path length of BST *built from a random permutation*
- = average internal path length + 2N

model: random permutation
# compares: N+1 + # compares for subfiles

model: random permutation
xpl: N+1 + xpl of subtrees
Approach works for any “additive parameter” (see text). Height requires a different (much more intricate) approach (see text).

Summary:

<table>
<thead>
<tr>
<th>Typical shape</th>
<th>Average path length</th>
<th>Height</th>
</tr>
</thead>
<tbody>
<tr>
<td>random binary tree</td>
<td>$\sim \sqrt{\pi N}$</td>
<td>$\sim 2\sqrt{\pi N}$</td>
</tr>
<tr>
<td>BST built from random permutation</td>
<td>$\sim 2 \ln N$</td>
<td>$\sim c \ln N$</td>
</tr>
</tbody>
</table>

$c \approx 4.311$
6. Trees

- Trees and forests
- Binary search trees
- Path length
- Other types of trees
Other types of trees in combinatorics

Classic tree structures:
  • The free tree, an acyclic connected graph.
  • The rooted tree, a free tree with a distinguished root node.
  • The ordered tree, a rooted tree where the order of the subtrees is significant.

Ex. 5-node trees:

Enumeration? Path length? Stay tuned for *Analytic Combinatorics*
Other types of trees in algorithmics

Variations on binary trees:

• The **t-ary tree**, where each node has exactly \( t \) children.
• The **t-restricted tree**, where each node has at most \( t \) children.
• The **2-3 tree**, the method of choice in symbol-table implementations.

Enumeration? Path length? Stay tuned for *Analytic Combinatorics*
An unsolved problem

*Balanced trees* are the method of choice for symbol tables
- Same search code as BSTs.
- Slight overhead for insertion.
- Guaranteed height $< 2\lg N$.
- Most algorithms use 2-3 or 2-3-4 tree representations.

Ex. LLRB (left-leaning red-black) trees.

Q. Path length of balanced tree built from a random permutation?
Balanced tree distribution

Probability that the root is of rank $k$ in a randomly-chosen AVL tree.

Random binary tree

BST built from a random permutation
Q. Path length of balanced tree built from a random permutation?
6. Trees

- Trees and forests
- Binary search trees
- Path length
- Other types of trees
- Exercises
Exercise 6.6

Tree enumeration via the symbolic method.

Exercise 6.6 What proportion of the forests with $N$ nodes have no trees consisting of a single node? For $N = 1, 2, 3,$ and $4$, the answer is $0, 1/2, 2/5,$ and $3/7$, respectively.
Exercise 6.27

Compute the probability that a BST is perfectly balanced.

Exercise 6.27 For \( N = 2^n - 1 \), what is the probability that a perfectly balanced tree structure (all \( 2^n \) external nodes on level \( n \)) will be built, if all \( N! \) key insertion sequences are equally likely?
Exercises 6.43

Parameters for BSTs built from a random permutation.

Answer these questions for BSTs built from a random permutation.

**Exercise 5.15** Find the average number of internal nodes in a binary tree of size $n$ with both children internal.

**Exercise 5.16** Find the average number of internal nodes in a binary tree of size $n$ with one child internal and one child external.
Assignments for next lecture

1. Read pages 257-344 in text.

2. Run experiments to validate mathematical results.

   **Experiment 1.** Generate 1000 random permutations for $N = 100$, 1000, and 10,000 and compare the average path length and height of the generated trees with the values predicted by analysis.

   **Experiment 2.** Extra credit. Do the same for random binary trees.

3. Write up solutions to Exercises 6.6, 6.27, and 6.43.
6. Trees