Graph Theory

Definition: A **graph** $G = (V, E)$ is made up of a set of **vertices** V and a set of **edges** E.

Graph Theory

Definition: A **graph** $G = (V, E)$ is made up of a set of **vertices** V and a set of **edges** E.

Example. $G = (V, E)$ where $V = \{v, w, x, y\}$, $E = \{vw, vx, vy, wx\}$.

Graph Theory

Definition: A **graph** $G = (V, E)$ is made up of a set of **vertices** V and a set of **edges** E.

Think of a vertex v as a dot and an edge $e = vw$ as a curve connecting v and w.

Example. $G = (V, E)$ where $V = \{v, w, x, y\}$, $E = \{vw, vx, vy, wx\}$.

Graph Theory

Definition: A **graph** $G = (V, E)$ is made up of a set of **vertices** V and a set of **edges** E.

Think of a vertex v as a dot and an edge $e = vw$ as a curve connecting v and w.

A graph is **connected** if for every two vertices v and w, there is a path from v to w.

Example. $G = (V, E)$ where $V = \{v, w, x, y\}$, $E = \{vw, vx, vy, wx\}$.

Graph Theory

Definition: A **graph** $G = (V, E)$ is made up of a set of **vertices** V and a set of **edges** E.

Think of a vertex v as a dot and an edge $e = vw$ as a curve connecting v and w.

A graph is **connected** if for every two vertices v and w, there is a path from v to w.

Example. $G = (V, E)$ where $V = \{v, w, x, y\}$, $E = \{vw, vx, vy, wx\}$.

T/F: G is connected.

Graph Theory

Definition: A **graph** $G = (V, E)$ is made up of a set of **vertices** V and a set of **edges** E.

Think of a vertex v as a dot and an edge $e = vw$ as a curve connecting v and w.

A graph is **connected** if for every two vertices v and w, there is a path from v to w.

The **degree** of a vertex v is the number of edges connected to v.

A **leaf** is a vertex with degree 1.

Example. $G = (V, E)$ where $V = \{v, w, x, y\}$, $E = \{vw, vx, vy, wx\}$.

T/F: G is connected.

Graph Theory

Definition: A **graph** $G = (V, E)$ is made up of a set of **vertices** V and a set of **edges** E.

Think of a vertex v as a dot and an edge $e = vw$ as a curve connecting v and w.

A graph is **connected** if for every two vertices v and w, there is a path from v to w.

The **degree** of a vertex v is the number of edges connected to v.

A **leaf** is a vertex with degree 1.

Example. $G = (V, E)$ where $V = \{v, w, x, y\}$, $E = \{vw, vx, vy, wx\}$.

$\deg v =$

$\deg y =$

T/F: G is connected.

Graph Theory

Definition: A **graph** $G = (V, E)$ is made up of a set of **vertices** V and a set of **edges** E.

Think of a vertex v as a dot and an edge $e = vw$ as a curve connecting v and w.

A graph is **connected** if for every two vertices v and w, there is a path from v to w.

The **degree** of a vertex v is the number of edges connected to v.

A **leaf** is a vertex with degree 1.

Example. $G = (V, E)$ where $V = \{v, w, x, y\}$, $E = \{vw, vx, vy, wx\}$.

deg $v =$

deg $y =$

T/F: G is connected.

A **path** is a set of edges “in a line”: $\{v_1 v_2, v_2 v_3, \ldots, v_{k-1} v_k\}$

A **cycle** is a set of edges “in a circle”: $\{v_1 v_2, v_2 v_3, \ldots, v_{k-1} v_k, v_k v_1\}$

Graph Theory

Definition: A **graph** $G = (V, E)$ is made up of a set of **vertices** V and a set of **edges** E.

Think of a vertex v as a dot and an edge $e = vw$ as a curve connecting v and w.

A graph is **connected** if for every two vertices v and w, there is a path from v to w.

The **degree** of a vertex v is the number of edges connected to v.

A **leaf** is a vertex with degree 1.

Example. $G = (V, E)$ where $V = \{v, w, x, y\}$, $E = \{vw, vx, vy, wx\}$.

deg $v =$
deg $y =$

T/F: G is connected.
T/F: G has a cycle.

A **path** is a set of edges "in a line": $\{v_1 v_2, v_2 v_3, \ldots, v_{k-1} v_k\}$
A **cycle** is a set of edges "in a circle": $\{v_1 v_2, v_2 v_3, \ldots, v_{k-1} v_k, v_k v_1\}$

Graph Theory

Definition: A **graph** $G = (V, E)$ is made up of a set of **vertices** V and a set of **edges** E.

Think of a vertex v as a dot and an edge $e = vw$ as a curve connecting v and w.

A graph is **connected** if for every two vertices v and w, there is a path from v to w.

The **degree** of a vertex v is the number of edges connected to v.

A **leaf** is a vertex with degree 1.

Example. $G = (V, E)$ where $V = \{v, w, x, y\}$, $E = \{vw, vx, vy, wx\}$.

deg $v =$

deg $y =$

T/F: G is connected.

T/F: G has a cycle.

A **path** is a set of edges "in a line": $\{v_1v_2, v_2v_3, \ldots, v_{k-1}v_k\}$

A **cycle** is a set of edges "in a circle": $\{v_1v_2, v_2v_3, \ldots, v_{k-1}v_k, v_kv_1\}$

A **tree** is a connected graph containing no cycles.

A **forest** is a graph containing no cycles. (may not be connected)

Graph Theory

Definition: A **graph** $G = (V, E)$ is made up of a set of **vertices** V and a set of **edges** E.

Think of a vertex v as a dot and an edge $e = vw$ as a curve connecting v and w.

A graph is **connected** if for every two vertices v and w, there is a path from v to w.

The **degree** of a vertex v is the number of edges connected to v.

A **leaf** is a vertex with degree 1.

Example. $G = (V, E)$ where $V = \{v, w, x, y\}$, $E = \{vw, vx, vy, wx\}$.

deg $v =$
deg $y =$

T/F: G is connected.
T/F: G has a cycle.
T/F: G is a tree.

A **path** is a set of edges "in a line": $\{v_1v_2, v_2v_3, \dots, v_{k-1}v_k\}$
A **cycle** is a set of edges "in a circle": $\{v_1v_2, v_2v_3, \dots, v_{k-1}v_k, v_kv_1\}$

A **tree** is a connected graph containing no cycles.
A **forest** is a graph containing no cycles. (may not be connected)

Counting Trees

Question: How many trees are there?

Counting Trees

Question: How many trees are there?

Answer: It depends.

Counting Trees

Question: How many trees are there?

Answer: It depends.

- Are there restrictions?

- Are the vertices labeled?

Counting Trees

Question: How many trees are there?

Answer: It depends.

- Are there restrictions?
  - We'll end by counting binary trees.
- Are the vertices labeled?

Counting Trees

Question: How many trees are there?

Answer: It depends.

- Are there restrictions?
  - We'll end by counting binary trees.
- Are the vertices labeled?
  - Does this matter? (Oh, yes!)

Counting Trees

Question: How many trees are there?

Answer: It depends.

- Are there restrictions?
  - We'll end by counting binary trees.
- Are the vertices labeled?
  - Does this matter? (Oh, yes!)
  - Unlabeled vs. Labeled: (A000055 vs. A000272)

Counting Trees

Question: How many trees are there?

Answer: It depends.

- Are there restrictions?
  - We'll end by counting binary trees.
- Are the vertices labeled?
  - Does this matter? (Oh, yes!)
  - Unlabeled vs. Labeled: (A000055 vs. A000272)

★ We can label every unlabeled tree in *some number* of ways. ★

Counting Trees

Question: How many trees are there?

Answer: It depends.

- Are there restrictions?
  - We'll end by counting binary trees.
- Are the vertices labeled?
  - Does this matter? (Oh, yes!)
  - Unlabeled vs. Labeled: (A000055 vs. A000272)

★ We can label every unlabeled tree in *some number* of ways. ★

- There is no nice formula for the number of unlabeled trees.

Counting Trees

Question: How many trees are there?

Answer: It depends.

- Are there restrictions?
  - We'll end by counting binary trees.
- Are the vertices labeled?
  - Does this matter? (Oh, yes!)
  - Unlabeled vs. Labeled: (A000055 vs. A000272)

★ We can label every unlabeled tree in *some number* of ways. ★

- There is no nice formula for the number of unlabeled trees.
- There is an amazingly nice formula for the number of labeled trees.

Counting Labeled Trees

Thm 6.2.5 The number of labeled trees is . (drumroll....)

Counting Labeled Trees

Thm 6.2.5 The number of labeled trees is n^{n-2}. (drumroll....)

Counting Labeled Trees

Thm 6.2.5 The number of labeled trees is n^{n-2}. (drumroll....)

Proof. We will construct a bijection between:

$$f : \left\{ \begin{array}{c} \text{labeled trees } T \\ \text{with } n \text{ vertices} \end{array} \right\} \to \left\{ \begin{array}{c} \text{lists } L \text{ of length } (n-2) \\ \text{taken from } \{1, \dots, n\} \end{array} \right\}.$$

Counting Labeled Trees

Thm 6.2.5 The number of labeled trees is n^{n-2}. (drumroll....)

Proof. We will construct a bijection between:

$$f : \left\{ \begin{array}{c} \text{labeled trees } T \\ \text{with } n \text{ vertices} \end{array} \right\} \to \left\{ \begin{array}{c} \text{lists } L \text{ of length } (n-2) \\ \text{taken from } \{1, \ldots, n\} \end{array} \right\}.$$

Given a tree T, create a list L called its **Prüfer sequence**:

Counting Labeled Trees

Thm 6.2.5 The number of labeled trees is n^{n-2}. (drumroll....)

Proof. We will construct a bijection between:

$$f : \left\{ \begin{array}{c} \text{labeled trees } T \\ \text{with } n \text{ vertices} \end{array} \right\} \to \left\{ \begin{array}{c} \text{lists } L \text{ of length } (n-2) \\ \text{taken from } \{1, \ldots, n\} \end{array} \right\}.$$

Given a tree T, create a list L called its **Prüfer sequence**:

- Start with the empty list $L = ()$.
- Repeat the following steps until the tree has only two vertices:

Counting Labeled Trees

Thm 6.2.5 The number of labeled trees is n^{n-2}. (drumroll....)

Proof. We will construct a bijection between:

$$f : \left\{ \begin{array}{c} \text{labeled trees } T \\ \text{with } n \text{ vertices} \end{array} \right\} \to \left\{ \begin{array}{c} \text{lists } L \text{ of length } (n-2) \\ \text{taken from } \{1, \ldots, n\} \end{array} \right\}.$$

Given a tree T, create a list L called its **Prüfer sequence**:

- Start with the empty list $L = ()$.
- Repeat the following steps until the tree has only two vertices:
  - **Find** the leaf v with the smallest label.

Counting Labeled Trees

Thm 6.2.5 The number of labeled trees is n^{n-2}. (drumroll....)

Proof. We will construct a bijection between:

$$f : \left\{ \begin{array}{c} \text{labeled trees } T \\ \text{with } n \text{ vertices} \end{array} \right\} \to \left\{ \begin{array}{c} \text{lists } L \text{ of length } (n-2) \\ \text{taken from } \{1, \ldots, n\} \end{array} \right\}.$$

Given a tree T, create a list L called its **Prüfer sequence**:

- Start with the empty list $L = ()$.
- Repeat the following steps until the tree has only two vertices:
  - **Find** the leaf v with the smallest label.
  - **Append** the label of v's neighbor to L.

Counting Labeled Trees

Thm 6.2.5 The number of labeled trees is n^{n-2}. (drumroll....)

Proof. We will construct a bijection between:

$$f : \left\{ \begin{array}{c} \text{labeled trees } T \\ \text{with } n \text{ vertices} \end{array} \right\} \to \left\{ \begin{array}{c} \text{lists } L \text{ of length } (n-2) \\ \text{taken from } \{1, \dots, n\} \end{array} \right\}.$$

Given a tree T, create a list L called its **Prüfer sequence**:

- Start with the empty list $L = ()$.
- Repeat the following steps until the tree has only two vertices:
  - **Find** the leaf v with the smallest label.
  - **Append** the label of v's neighbor to L.
  - **Remove** v from the tree.

Counting Labeled Trees

Thm 6.2.5 The number of labeled trees is n^{n-2}. (drumroll....)

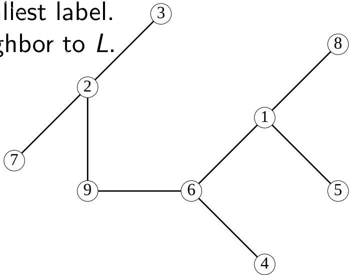
Proof. We will construct a bijection between:

$$f : \left\{ \begin{array}{c} \text{labeled trees } T \\ \text{with } n \text{ vertices} \end{array} \right\} \to \left\{ \begin{array}{c} \text{lists } L \text{ of length } (n-2) \\ \text{taken from } \{1, \ldots, n\} \end{array} \right\}.$$

Given a tree T, create a list L called its **Prüfer sequence**:

- Start with the empty list $L = ()$.
- Repeat the following steps until the tree has only two vertices:
  - **Find** the leaf v with the smallest label.
  - **Append** the label of v's neighbor to L.
  - **Remove** v from the tree.

Example. The Prüfer sequence of this tree is $(2, 6, 1, 2, 9, 1, 6)$.



Counting Labeled Trees

Thm 6.2.5 The number of labeled trees is n^{n-2}. (drumroll....)

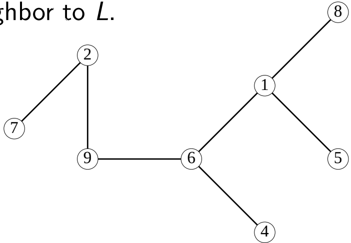
Proof. We will construct a bijection between:

$$f : \left\{ \begin{array}{c} \text{labeled trees } T \\ \text{with } n \text{ vertices} \end{array} \right\} \to \left\{ \begin{array}{c} \text{lists } L \text{ of length } (n-2) \\ \text{taken from } \{1, \ldots, n\} \end{array} \right\}.$$

Given a tree T, create a list L called its **Prüfer sequence**:

- Start with the empty list $L = ()$.
- Repeat the following steps until the tree has only two vertices:
  - **Find** the leaf v with the smallest label.
  - **Append** the label of v's neighbor to L.
  - **Remove** v from the tree.

Example. The Prüfer sequence of this tree is $(2, 6, 1, 2, 9, 1, 6)$.



Counting Labeled Trees

Thm 6.2.5 The number of labeled trees is n^{n-2}. (drumroll....)

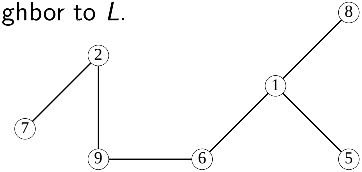
Proof. We will construct a bijection between:

$$f : \left\{ \begin{array}{c} \text{labeled trees } T \\ \text{with } n \text{ vertices} \end{array} \right\} \to \left\{ \begin{array}{c} \text{lists } L \text{ of length } (n-2) \\ \text{taken from } \{1, \ldots, n\} \end{array} \right\}.$$

Given a tree T, create a list L called its **Prüfer sequence**:

- Start with the empty list $L = ()$.
- Repeat the following steps until the tree has only two vertices:
  - **Find** the leaf v with the smallest label.
  - **Append** the label of v's neighbor to L.
  - **Remove** v from the tree.

Example. The Prüfer sequence of this tree is $(2, 6, 1, 2, 9, 1, 6)$.



Counting Labeled Trees

Thm 6.2.5 The number of labeled trees is n^{n-2}. (drumroll....)

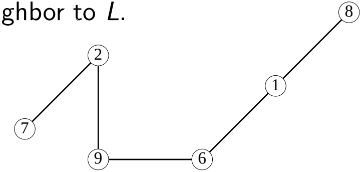
Proof. We will construct a bijection between:

$$f : \left\{ \begin{array}{c} \text{labeled trees } T \\ \text{with } n \text{ vertices} \end{array} \right\} \to \left\{ \begin{array}{c} \text{lists } L \text{ of length } (n-2) \\ \text{taken from } \{1, \ldots, n\} \end{array} \right\}.$$

Given a tree T, create a list L called its **Prüfer sequence**:

- Start with the empty list $L = ()$.
- Repeat the following steps until the tree has only two vertices:
  - **Find** the leaf v with the smallest label.
  - **Append** the label of v's neighbor to L.
  - **Remove** v from the tree.

Example. The Prüfer sequence of this tree is $(2, 6, 1, 2, 9, 1, 6)$.



Counting Labeled Trees

Thm 6.2.5 The number of labeled trees is n^{n-2}. (drumroll....)

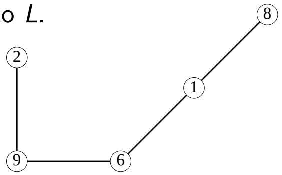
Proof. We will construct a bijection between:

$$f : \left\{ \begin{array}{c} \text{labeled trees } T \\ \text{with } n \text{ vertices} \end{array} \right\} \to \left\{ \begin{array}{c} \text{lists } L \text{ of length } (n-2) \\ \text{taken from } \{1, \ldots, n\} \end{array} \right\}.$$

Given a tree T, create a list L called its **Prüfer sequence**:

- Start with the empty list $L = ()$.
- Repeat the following steps until the tree has only two vertices:
  - **Find** the leaf v with the smallest label.
  - **Append** the label of v's neighbor to L.
  - **Remove** v from the tree.

Example. The Prüfer sequence of this tree is $(2, 6, 1, 2, 9, 1, 6)$.



Counting Labeled Trees

Thm 6.2.5 The number of labeled trees is n^{n-2}. (drumroll....)

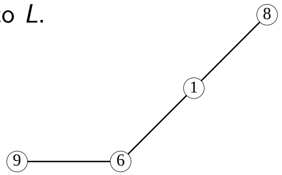
Proof. We will construct a bijection between:

$$f : \left\{ \begin{array}{c} \text{labeled trees } T \\ \text{with } n \text{ vertices} \end{array} \right\} \to \left\{ \begin{array}{c} \text{lists } L \text{ of length } (n-2) \\ \text{taken from } \{1, \ldots, n\} \end{array} \right\}.$$

Given a tree T, create a list L called its **Prüfer sequence**:

- Start with the empty list $L = ()$.
- Repeat the following steps until the tree has only two vertices:
  - **Find** the leaf v with the smallest label.
  - **Append** the label of v's neighbor to L.
  - **Remove** v from the tree.

Example. The Prüfer sequence of this tree is $(2, 6, 1, 2, 9, 1, 6)$.



Counting Labeled Trees

Thm 6.2.5 The number of labeled trees is n^{n-2}. (drumroll....)

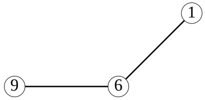
Proof. We will construct a bijection between:

$$f : \left\{ \begin{array}{c} \text{labeled trees } T \\ \text{with } n \text{ vertices} \end{array} \right\} \to \left\{ \begin{array}{c} \text{lists } L \text{ of length } (n-2) \\ \text{taken from } \{1, \ldots, n\} \end{array} \right\}.$$

Given a tree T, create a list L called its **Prüfer sequence**:

- Start with the empty list $L = ()$.
- Repeat the following steps until the tree has only two vertices:
  - **Find** the leaf v with the smallest label.
  - **Append** the label of v's neighbor to L.
  - **Remove** v from the tree.

Example. The Prüfer sequence of this tree is $(2, 6, 1, 2, 9, 1, 6)$.



Counting Labeled Trees

Thm 6.2.5 The number of labeled trees is n^{n-2}. (drumroll....)

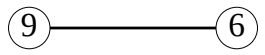
Proof. We will construct a bijection between:

$$f : \left\{ \begin{array}{c} \text{labeled trees } T \\ \text{with } n \text{ vertices} \end{array} \right\} \to \left\{ \begin{array}{c} \text{lists } L \text{ of length } (n-2) \\ \text{taken from } \{1, \ldots, n\} \end{array} \right\}.$$

Given a tree T, create a list L called its **Prüfer sequence**:

- Start with the empty list $L = ()$.
- Repeat the following steps until the tree has only two vertices:
  - **Find** the leaf v with the smallest label.
  - **Append** the label of v's neighbor to L.
  - **Remove** v from the tree.

Example. The Prüfer sequence of this tree is $(2, 6, 1, 2, 9, 1, 6)$.



Counting Labeled Trees

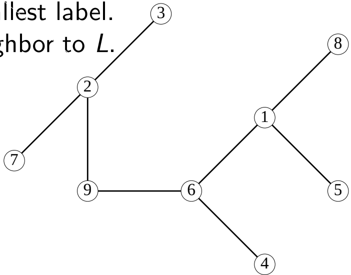
Thm 6.2.5 The number of labeled trees is n^{n-2}. (drumroll....)

Proof. We will construct a bijection between:

$$f : \left\{ \begin{array}{c} \text{labeled trees } T \\ \text{with } n \text{ vertices} \end{array} \right\} \to \left\{ \begin{array}{c} \text{lists } L \text{ of length } (n-2) \\ \text{taken from } \{1, \dots, n\} \end{array} \right\}.$$

Given a tree T, create a list L called its **Prüfer sequence**:

- Start with the empty list $L = ()$.
- Repeat the following steps until the tree has only two vertices:
  - **Find** the leaf v with the smallest label.
  - **Append** the label of v's neighbor to L.
  - **Remove** v from the tree.



Example. The Prüfer sequence of this tree is $(2, 6, 1, 2, 9, 1, 6)$.

★ This rule is well defined. ★

Counting Labeled Trees

There is an inverse rule

$$g : \left\{ \begin{array}{c} \text{lists } L \text{ of length } (n-2) \\ \text{taken from } \{1, \ldots, n\} \end{array} \right\} \to \left\{ \begin{array}{c} \text{labeled trees } T \\ \text{with } n \text{ vertices} \end{array} \right\}.$$

Counting Labeled Trees

There is an inverse rule

$$g : \left\{ \begin{array}{c} \text{lists } L \text{ of length } (n-2) \\ \text{taken from } \{1, \ldots, n\} \end{array} \right\} \to \left\{ \begin{array}{c} \text{labeled trees } T \\ \text{with } n \text{ vertices} \end{array} \right\}.$$

Given a list L, create a new list U (used vertices) and tree T:

Counting Labeled Trees

There is an inverse rule

$$g : \left\{ \begin{array}{c} \text{lists } L \text{ of length } (n-2) \\ \text{taken from } \{1, \ldots, n\} \end{array} \right\} \to \left\{ \begin{array}{c} \text{labeled trees } T \\ \text{with } n \text{ vertices} \end{array} \right\}.$$

Given a list L, create a new list U (used vertices) and tree T:

- Start with the empty list $U = ()$.
- Repeat the following steps until L is empty:

Counting Labeled Trees

There is an inverse rule

$$g : \left\{ \begin{array}{c} \text{lists } L \text{ of length } (n-2) \\ \text{taken from } \{1, \ldots, n\} \end{array} \right\} \to \left\{ \begin{array}{c} \text{labeled trees } T \\ \text{with } n \text{ vertices} \end{array} \right\}.$$

Given a list L, create a new list U (used vertices) and tree T:

- Start with the empty list $U = ()$.
- Repeat the following steps until L is empty:
    - **Find** the least vertex u on neither L nor U

Counting Labeled Trees

There is an inverse rule

$$g : \left\{ \begin{array}{c} \text{lists } L \text{ of length } (n-2) \\ \text{taken from } \{1, \ldots, n\} \end{array} \right\} \to \left\{ \begin{array}{c} \text{labeled trees } T \\ \text{with } n \text{ vertices} \end{array} \right\}.$$

Given a list L, create a new list U (used vertices) and tree T:

- Start with the empty list $U = ()$.
- Repeat the following steps until L is empty:
  - **Find** the least vertex u on neither L nor U –and–
    Find the first vertex l on list L.

Counting Labeled Trees

There is an inverse rule

$$g : \left\{ \begin{array}{c} \text{lists } L \text{ of length } (n-2) \\ \text{taken from } \{1, \ldots, n\} \end{array} \right\} \to \left\{ \begin{array}{c} \text{labeled trees } T \\ \text{with } n \text{ vertices} \end{array} \right\}.$$

Given a list L, create a new list U (used vertices) and tree T:

- Start with the empty list $U = ()$.
- Repeat the following steps until L is empty:
  - **Find** the least vertex u on neither L nor U –and–
    Find the first vertex l on list L.
  - **Add** the edge (u, l) to T.

Counting Labeled Trees

There is an inverse rule

$$g : \left\{ \begin{array}{c} \text{lists } L \text{ of length } (n-2) \\ \text{taken from } \{1, \ldots, n\} \end{array} \right\} \rightarrow \left\{ \begin{array}{c} \text{labeled trees } T \\ \text{with } n \text{ vertices} \end{array} \right\}.$$

Given a list L, create a new list U (used vertices) and tree T:

- Start with the empty list $U = ()$.
- Repeat the following steps until L is empty:
  - **Find** the least vertex u on neither L nor U –and–
    Find the first vertex l on list L.
  - **Add** the edge (u, l) to T.
  - **Add** u to the list U –and–
    Remove l from L.

Counting Labeled Trees

There is an inverse rule

$$g : \left\{ \begin{array}{c} \text{lists } L \text{ of length } (n-2) \\ \text{taken from } \{1, \ldots, n\} \end{array} \right\} \to \left\{ \begin{array}{c} \text{labeled trees } T \\ \text{with } n \text{ vertices} \end{array} \right\}.$$

Given a list L, create a new list U (used vertices) and tree T:

- Start with the empty list $U = ()$.
- Repeat the following steps until L is empty:
  - **Find** the least vertex u on neither L nor U –and–
    Find the first vertex l on list L.
  - **Add** the edge (u, l) to T.
  - **Add** u to the list U –and–
    Remove l from L.
- Add edge between vtx not in U.

Counting Labeled Trees

There is an inverse rule

$$g : \left\{ \begin{array}{c} \text{lists } L \text{ of length } (n-2) \\ \text{taken from } \{1, \ldots, n\} \end{array} \right\} \to \left\{ \begin{array}{c} \text{labeled trees } T \\ \text{with } n \text{ vertices} \end{array} \right\}.$$

Given a list L, create a new list U (used vertices) and tree T:

- Start with the empty list $U = ()$.
- Repeat the following steps until L is empty:
  - **Find** the least vertex u on neither L nor U –and–
    Find the first vertex l on list L.
  - **Add** the edge (u, l) to T.
  - **Add** u to the list U –and–
    Remove l from L.
- Add edge between vtx not in U.

Example. This method takes the Prüfer sequence $(2, 6, 1, 2, 9, 1, 6)$ and returns our original T.

Counting Labeled Trees

There is an inverse rule

$$g : \left\{ \begin{array}{c} \text{lists } L \text{ of length } (n-2) \\ \text{taken from } \{1, \ldots, n\} \end{array} \right\} \to \left\{ \begin{array}{c} \text{labeled trees } T \\ \text{with } n \text{ vertices} \end{array} \right\}.$$

Given a list L, create a new list U (used vertices) and tree T:

- Start with the empty list $U = ()$.
- Repeat the following steps until L is empty:
  - **Find** the least vertex u on neither L nor U –and–
    Find the first vertex l on list L.
  - **Add** the edge (u, l) to T.
  - **Add** u to the list U –and–
    Remove l from L.
- Add edge between vtx not in U.

(2,6,1,2,9,1,6)	()	(3,2)

Example. This method takes the Prüfer sequence $(2, 6, 1, 2, 9, 1, 6)$ and returns our original T.

Counting Labeled Trees

There is an inverse rule

$$g : \left\{ \begin{array}{c} \text{lists } L \text{ of length } (n-2) \\ \text{taken from } \{1, \ldots, n\} \end{array} \right\} \to \left\{ \begin{array}{c} \text{labeled trees } T \\ \text{with } n \text{ vertices} \end{array} \right\}.$$

Given a list L, create a new list U (used vertices) and tree T:

- Start with the empty list $U = ()$.
- Repeat the following steps until L is empty:
  - **Find** the least vertex u on neither L nor U –and– **Find** the first vertex l on list L.
  - **Add** the edge (u, l) to T.
  - **Add** u to the list U –and– **Remove** l from L.
- Add edge between vtx not in U.

(2,6,1,2,9,1,6)	()	(3,2)
(6,1,2,9,1,6)	(3)	(4,6)

Example. This method takes the Prüfer sequence $(2, 6, 1, 2, 9, 1, 6)$ and returns our original T.

Counting Labeled Trees

There is an inverse rule

$$g : \left\{ \begin{array}{c} \text{lists } L \text{ of length } (n-2) \\ \text{taken from } \{1,\ldots,n\} \end{array} \right\} \to \left\{ \begin{array}{c} \text{labeled trees } T \\ \text{with } n \text{ vertices} \end{array} \right\}.$$

Given a list L, create a new list U (used vertices) and tree T:

- Start with the empty list $U = ()$.
- Repeat the following steps until L is empty:
  - **Find** the least vertex u on neither L nor U –and– **Find** the first vertex l on list L.
  - **Add** the edge (u, l) to T.
  - **Add** u to the list U –and– **Remove** l from L.
- Add edge between vtx not in U.

(2,6,1,2,9,1,6)	()	(3,2)
(6,1,2,9,1,6)	(3)	(4,6)
(1,2,9,1,6)	(3,4)	(5,1)

Example. This method takes the Prüfer sequence $(2, 6, 1, 2, 9, 1, 6)$ and returns our original T.

Counting Labeled Trees

There is an inverse rule

$$g : \left\{ \begin{array}{c} \text{lists } L \text{ of length } (n-2) \\ \text{taken from } \{1, \ldots, n\} \end{array} \right\} \to \left\{ \begin{array}{c} \text{labeled trees } T \\ \text{with } n \text{ vertices} \end{array} \right\}.$$

Given a list L, create a new list U (used vertices) and tree T:

- Start with the empty list $U = ()$.
- Repeat the following steps until L is empty:
  - **Find** the least vertex u on neither L nor U –and– **Find** the first vertex l on list L.
  - **Add** the edge (u, l) to T.
  - **Add** u to the list U –and– **Remove** l from L.
- Add edge between vtx not in U.

(2,6,1,2,9,1,6)	()	(3,2)
(6,1,2,9,1,6)	(3)	(4,6)
(1,2,9,1,6)	(3,4)	(5,1)
(2,9,1,6)	(3,4,5)	(7,2)

Example. This method takes the Prüfer sequence $(2, 6, 1, 2, 9, 1, 6)$ and returns our original T.

Counting Labeled Trees

There is an inverse rule

$$g : \left\{ \begin{array}{c} \text{lists } L \text{ of length } (n-2) \\ \text{taken from } \{1, \ldots, n\} \end{array} \right\} \to \left\{ \begin{array}{c} \text{labeled trees } T \\ \text{with } n \text{ vertices} \end{array} \right\}.$$

Given a list L, create a new list U (used vertices) and tree T:

- Start with the empty list $U = ()$.
- Repeat the following steps until L is empty:
  - **Find** the least vertex u on neither L nor U –and– **Find** the first vertex l on list L.
  - **Add** the edge (u, l) to T.
  - **Add** u to the list U –and– **Remove** l from L.
- Add edge between vtx not in U.

(2,6,1,2,9,1,6)	()	(3,2)
(6,1,2,9,1,6)	(3)	(4,6)
(1,2,9,1,6)	(3,4)	(5,1)
(2,9,1,6)	(3,4,5)	(7,2)
(9,1,6)	(3,4,5,7)	(2,9)

Example. This method takes the Prüfer sequence $(2, 6, 1, 2, 9, 1, 6)$ and returns our original T.

Counting Labeled Trees

There is an inverse rule

$$g : \left\{ \begin{array}{c} \text{lists } L \text{ of length } (n-2) \\ \text{taken from } \{1, \ldots, n\} \end{array} \right\} \to \left\{ \begin{array}{c} \text{labeled trees } T \\ \text{with } n \text{ vertices} \end{array} \right\}.$$

Given a list L, create a new list U (used vertices) and tree T:

- Start with the empty list $U = ()$.
- Repeat the following steps until L is empty:
  - **Find** the least vertex u on neither L nor U –and– **Find** the first vertex l on list L.
  - **Add** the edge (u, l) to T.
  - **Add** u to the list U –and– **Remove** l from L.
- Add edge between vtx not in U.

(2,6,1,2,9,1,6)	()	(3,2)
(6,1,2,9,1,6)	(3)	(4,6)
(1,2,9,1,6)	(3,4)	(5,1)
(2,9,1,6)	(3,4,5)	(7,2)
(9,1,6)	(3,4,5,7)	(2,9)
(1,6)	(2,3,4,5,7)	(8,1)

Example. This method takes the Prüfer sequence $(2, 6, 1, 2, 9, 1, 6)$ and returns our original T.

Counting Labeled Trees

There is an inverse rule

$$g : \left\{ \begin{array}{c} \text{lists } L \text{ of length } (n-2) \\ \text{taken from } \{1, \ldots, n\} \end{array} \right\} \to \left\{ \begin{array}{c} \text{labeled trees } T \\ \text{with } n \text{ vertices} \end{array} \right\}.$$

Given a list L, create a new list U (used vertices) and tree T:

- Start with the empty list $U = ()$.
- Repeat the following steps until L is empty:
  - **Find** the least vertex u on neither L nor U –and– **Find** the first vertex l on list L.
  - **Add** the edge (u, l) to T.
  - **Add** u to the list U –and– **Remove** l from L.
- Add edge between vtx not in U.

(2,6,1,2,9,1,6)	()	(3,2)
(6,1,2,9,1,6)	(3)	(4,6)
(1,2,9,1,6)	(3,4)	(5,1)
(2,9,1,6)	(3,4,5)	(7,2)
(9,1,6)	(3,4,5,7)	(2,9)
(1,6)	(2,3,4,5,7)	(8,1)
(6)	(2,3,4,5,7,8)	(1,6)

Example. This method takes the Prüfer sequence $(2, 6, 1, 2, 9, 1, 6)$ and returns our original T.

Counting Labeled Trees

There is an inverse rule

$$g : \left\{ \begin{array}{c} \text{lists } L \text{ of length } (n-2) \\ \text{taken from } \{1, \ldots, n\} \end{array} \right\} \to \left\{ \begin{array}{c} \text{labeled trees } T \\ \text{with } n \text{ vertices} \end{array} \right\}.$$

Given a list L, create a new list U (used vertices) and tree T:

- Start with the empty list $U = ()$.
- Repeat the following steps until L is empty:
  - **Find** the least vertex u on neither L nor U –and–
    Find the first vertex l on list L.
  - **Add** the edge (u, l) to T.
  - **Add** u to the list U –and–
    Remove l from L.
- Add edge between vtx not in U.

Example. This method takes the Prüfer sequence $(2, 6, 1, 2, 9, 1, 6)$ and returns our original T.

(2,6,1,2,9,1,6)	()	(3,2)
(6,1,2,9,1,6)	(3)	(4,6)
(1,2,9,1,6)	(3,4)	(5,1)
(2,9,1,6)	(3,4,5)	(7,2)
(9,1,6)	(3,4,5,7)	(2,9)
(1,6)	(2,3,4,5,7)	(8,1)
(6)	(2,3,4,5,7,8)	(1,6)
()	(1,2,3,4,5,7,8)	(6,9)

Counting Labeled Trees

There is an inverse rule

$$g : \left\{ \begin{array}{c} \text{lists } L \text{ of length } (n-2) \\ \text{taken from } \{1, \ldots, n\} \end{array} \right\} \to \left\{ \begin{array}{c} \text{labeled trees } T \\ \text{with } n \text{ vertices} \end{array} \right\}.$$

Given a list L, create a new list U (used vertices) and tree T:

- Start with the empty list $U = ()$.
- Repeat the following steps until L is empty:
  - **Find** the least vertex u on neither L nor U –and– **Find** the first vertex l on list L.
  - **Add** the edge (u, l) to T.
  - **Add** u to the list U –and– **Remove** l from L.
- Add edge between vtx not in U.

Example. This method takes the Prüfer sequence $(2, 6, 1, 2, 9, 1, 6)$ and returns our original T.

(2,6,1,2,9,1,6)	()	(3,2)
(6,1,2,9,1,6)	(3)	(4,6)
(1,2,9,1,6)	(3,4)	(5,1)
(2,9,1,6)	(3,4,5)	(7,2)
(9,1,6)	(3,4,5,7)	(2,9)
(1,6)	(2,3,4,5,7)	(8,1)
(6)	(2,3,4,5,7,8)	(1,6)
()	(1,2,3,4,5,7,8)	(6,9)

Counting Binary Trees

Definition: A **binary tree** has a special vertex called its **root**. From this vertex at the top, the rest of the tree is drawn downward. Each vertex may have a **left child** and/or a **right child**.

Counting Binary Trees

Definition: A **binary tree** has a special vertex called its **root**. From this vertex at the top, the rest of the tree is drawn downward. Each vertex may have a **left child** and/or a **right child**.

Example. The number of binary trees with $1, 2, 3$ vertices is:

Counting Binary Trees

Definition: A **binary tree** has a special vertex called its **root**. From this vertex at the top, the rest of the tree is drawn downward. Each vertex may have a **left child** and/or a **right child**.

Example. The number of binary trees with $1, 2, 3$ vertices is:

Example. The number of binary trees with 4 vertices is:

Counting Binary Trees

Definition: A **binary tree** has a special vertex called its **root**. From this vertex at the top, the rest of the tree is drawn downward. Each vertex may have a **left child** and/or a **right child**.

Example. The number of binary trees with $1, 2, 3$ vertices is:

Example. The number of binary trees with 4 vertices is:

Conjecture: The number of binary trees on n vertices is ______.

Counting Binary Trees

Proof: Every binary tree either:

- Has no vertices (x^0) –or–
- Breaks down as one root vertex (x)
  along with two binary trees beneath ($B(x)^2$).

Counting Binary Trees

Proof: Every binary tree either:

- Has no vertices (x^0) –or–
- Breaks down as one root vertex (x)
  along with two binary trees beneath ($B(x)^2$).

Therefore, the generating function for binary trees satisfies $B(x) = 1 + xB(x)^2$.

Counting Binary Trees

Proof: Every binary tree either:

- Has no vertices (x^0) –or–
- Breaks down as one root vertex (x) along with two binary trees beneath ($B(x)^2$).

Therefore, the generating function for binary trees satisfies $B(x) = 1 + xB(x)^2$. We conclude $b_n = \frac{1}{n+1}\binom{2n}{n}$.

Counting Binary Trees

Proof: Every binary tree either:

- Has no vertices (x^0) –or–
- Breaks down as one root vertex (x) along with two binary trees beneath ($B(x)^2$).

Therefore, the generating function for binary trees satisfies $B(x) = 1 + xB(x)^2$. We conclude $b_n = \frac{1}{n+1}\binom{2n}{n}$. ☺

Counting Binary Trees

Proof: Every binary tree either:

- Has no vertices (x^0) –or–
- Breaks down as one root vertex (x) along with two binary trees beneath ($B(x)^2$).

Therefore, the generating function for binary trees satisfies $B(x) = 1 + xB(x)^2$. We conclude $b_n = \frac{1}{n+1}\binom{2n}{n}$. ☺

Another way: Find a recurrence for b_n.

Counting Binary Trees

Proof: Every binary tree either:

- Has no vertices (x^0) –or–
- Breaks down as one root vertex (x) along with two binary trees beneath ($B(x)^2$).

Therefore, the generating function for binary trees satisfies $B(x) = 1 + xB(x)^2$. We conclude $b_n = \frac{1}{n+1}\binom{2n}{n}$. ☺

Another way: Find a recurrence for b_n. Note:

$$b_4 = b_0 b_3 + b_1 b_2 + b_2 b_1 + b_3 b_0.$$

Counting Binary Trees

Proof: Every binary tree either:

- Has no vertices (x^0) –or–
- Breaks down as one root vertex (x)
  along with two binary trees beneath ($B(x)^2$).

Therefore, the generating function for binary trees satisfies
$B(x) = 1 + xB(x)^2$. We conclude $b_n = \frac{1}{n+1}\binom{2n}{n}$. ☺

Another way: Find a recurrence for b_n. Note:

$$b_4 = b_0 b_3 + b_1 b_2 + b_2 b_1 + b_3 b_0.$$

In general, $b_n = \sum_{i=0}^{n-1} b_i b_{n-1-i}$.

Counting Binary Trees

Proof: Every binary tree either:

- Has no vertices (x^0) –or–
- Breaks down as one root vertex (x) along with two binary trees beneath ($B(x)^2$).

Therefore, the generating function for binary trees satisfies $B(x) = 1 + xB(x)^2$. We conclude $b_n = \frac{1}{n+1}\binom{2n}{n}$. ☺

Another way: Find a recurrence for b_n. Note:

$$b_4 = b_0 b_3 + b_1 b_2 + b_2 b_1 + b_3 b_0.$$

In general, $b_n = \sum_{i=0}^{n-1} b_i b_{n-1-i}$. Therefore, $B(x)$ equals

$$1 + \sum_{n \geq 1} \left(\sum_{i=0}^{n-1} b_i b_{n-1-i} \right) x^n$$

Counting Binary Trees

Proof: Every binary tree either:

- Has no vertices (x^0) –or–
- Breaks down as one root vertex (x) along with two binary trees beneath ($B(x)^2$).

Therefore, the generating function for binary trees satisfies $B(x) = 1 + xB(x)^2$. We conclude $b_n = \frac{1}{n+1}\binom{2n}{n}$. ☺

Another way: Find a recurrence for b_n. Note:

$$b_4 = b_0 b_3 + b_1 b_2 + b_2 b_1 + b_3 b_0.$$

In general, $b_n = \sum_{i=0}^{n-1} b_i b_{n-1-i}$. Therefore, $B(x)$ equals

$$1 + \sum_{n\geq 1} \left(\sum_{i=0}^{n-1} b_i b_{n-1-i} \right) x^n = 1 + x \sum_{n\geq 1} \left(\sum_{i=0}^{n-1} b_i b_{n-1-i} \right) x^{n-1} =$$

Counting Binary Trees

Proof: Every binary tree either:

- Has no vertices (x^0) –or–
- Breaks down as one root vertex (x) along with two binary trees beneath ($B(x)^2$).

Therefore, the generating function for binary trees satisfies $B(x) = 1 + xB(x)^2$. We conclude $b_n = \frac{1}{n+1}\binom{2n}{n}$. ☺

Another way: Find a recurrence for b_n. Note:

$$b_4 = b_0 b_3 + b_1 b_2 + b_2 b_1 + b_3 b_0.$$

In general, $b_n = \sum_{i=0}^{n-1} b_i b_{n-1-i}$. Therefore, $B(x)$ equals

$$1 + \sum_{n \geq 1} \left(\sum_{i=0}^{n-1} b_i b_{n-1-i} \right) x^n = 1 + x \sum_{n \geq 1} \left(\sum_{i=0}^{n-1} b_i b_{n-1-i} \right) x^{n-1} =$$

$$1 + x \sum_{k \geq 0} \left(\sum_{i=0}^{k} b_i b_{k-i} \right) x^k =$$

Counting Binary Trees

Proof: Every binary tree either:

- Has no vertices (x^0) –or–
- Breaks down as one root vertex (x) along with two binary trees beneath ($B(x)^2$).

Therefore, the generating function for binary trees satisfies $B(x) = 1 + xB(x)^2$. We conclude $b_n = \frac{1}{n+1}\binom{2n}{n}$. ☺

Another way: Find a recurrence for b_n. Note:

$$b_4 = b_0 b_3 + b_1 b_2 + b_2 b_1 + b_3 b_0.$$

In general, $b_n = \sum_{i=0}^{n-1} b_i b_{n-1-i}$. Therefore, $B(x)$ equals

$$1 + \sum_{n \geq 1} \left(\sum_{i=0}^{n-1} b_i b_{n-1-i} \right) x^n = 1 + x \sum_{n \geq 1} \left(\sum_{i=0}^{n-1} b_i b_{n-1-i} \right) x^{n-1} =$$

$$1 + x \sum_{k \geq 0} \left(\sum_{i=0}^{k} b_i b_{k-i} \right) x^k = 1 + x \left(\sum_{k \geq 0} b_k x^k \right) \left(\sum_{k \geq 0} b_k x^k \right) = 1 + xB(x)^2.$$