# EECS3101 Algorithms (Summer'26)

## Digest of Lecture Tutorial 5, June 8

Induction Proofs: Proper Binary Trees and Spanning Trees

Jackie Wang

### How to use this Digest

This digest **is** meant to help you identify the main ideas, important examples, common pitfalls, and practice targets from the lecture. It is **not** a replacement for attending class, watching the lecture recording, reviewing the slides and iPad notes, or doing your own practice and reflection. The digest intentionally summarizes only the roadmap and highlights; **you are expected to fill in the details** by revisiting the lecture materials, tracing examples, writing code, solving problems, and checking your own understanding. A good way to use this digest is as a checklist: after reviewing the lecture, you should be able to explain and apply the items listed under "*What you should be able to do*".

This digest is an AI-assisted study roadmap reviewed by Jackie. It was produced from the lecture transcript, a carefully designed tool workflow, and Jackie's actual hand-written iPad notes. Although the digest is reviewed before being posted, it **might** still contain typos, inconsistencies, omissions, or errors, just like any manually prepared course material. Therefore, use it as a study guide, **not** as an unquestionable authority. If something seems unclear or inconsistent, you are responsible for asking for clarification and verifying your understanding. Possible errors in the digest should **not** be treated as permission to submit incorrect work, or as a substitute for understanding the course materials.

### Table of Contents

# Main Ideas

- The tutorial focused on how to use mathematical induction for structured objects, first proper binary trees and then spanning trees.
- The first proof showed that for any non-empty proper binary tree, the number of external nodes is exactly one more than the number of internal nodes: $N_E = N_I + 1$.
- The key tree-proof idea was to choose a structure-preserving minimal extension: convert one external node into an internal node and add two new external children.
- The second proof showed that if a graph is a spanning tree, then its number of edges satisfies $m = n - 1$, where $n = |V|$ and $m = |E|$.
- The graph-proof idea was to extend a spanning tree by adding one new vertex and exactly one new edge, because zero edges breaks connectedness and two or more edges create a cycle.
- Jackie emphasized that implication direction matters: $G$ being a spanning tree implies $m = n - 1$, but $m = n - 1$ alone does not imply that $G$ is a spanning tree.
- Assignment 1 was briefly connected to reading one-line `BSTNode` construction expressions in JUnit-style tests; a short guide was to be posted for students who need help parsing those expressions.

# Roadmap

## Tutorial agenda and where this fits

Jackie began by locating Tutorial 5 under the graph category on the lecture site and explaining that Tutorials 5 through 8 would continue graph-related work, while Tutorial 9 would be for dynamic programming. The main goal of this tutorial was not to introduce a new algorithm, but to practise proof structure: one induction proof on trees and one induction proof on graphs. Jackie also noted that, time permitting, Assignment 1's one-line tree construction style would be discussed because it appears in the starter tests and is useful for later programming tests.

## Warm-up: induction pattern

The tutorial reused the induction template from the earlier graph lecture: check small cases, state an inductive hypothesis, and then prove a strictly larger case by making the smallest valid extension. Jackie stressed that the extension must preserve the defining structure. For graphs this may mean adding one vertex; for proper binary trees, adding just one node is not valid because it can break properness.

## Proper binary trees and the target property

The first problem concerned a non-empty proper binary tree. Proper means that every internal node has exactly two children; every node therefore has either zero children or two children. The target property was written using $N_I$ for the number of internal nodes and $N_E$ for the number of external nodes:

$$N_E = N_I + 1.$$

Jackie first checked the property on a concrete tree in the notes, where counting gave $N_I = 5$ and $N_E = 6$. This example was used only to build intuition before the proof.

## Tree proof: base cases and inductive hypothesis

The smallest non-empty proper binary tree has one external node and no internal nodes, so $N_E = 1$ and $N_I = 0$, and the property holds. Jackie also checked the next smallest proper binary tree: one internal root with two external children, so $N_E = 2$ and $N_I = 1$. The inductive hypothesis then assumed that the property holds for a proper binary tree larger than these base cases.

## Tree proof: the structure-preserving extension

For the inductive step, Jackie used the smallest valid way to make a larger proper binary tree: choose an external node, convert it into an internal node, and attach two new external children. This increases the total size by two, not one. The before/after count is the core of the proof: $N'_I = N_I + 1$, because the chosen external node becomes internal, and $N'_E = N_E - 1 + 2 = N_E + 1$, because one external node is lost and two new ones are gained. Substituting the inductive hypothesis $N_E = N_I + 1$ gives $N'_E = N'_I + 1$, which completes the step.

## Translating the spanning-tree statement

The second problem was the first property from the graph-property slide: the number of edges is the number of vertices minus one if the graph is a spanning tree. Jackie paused to translate this English sentence into a logical implication. The direction is

$$G \text{ is a spanning tree} \Rightarrow m = n - 1.$$

This is not the same as the converse. A graph can have $n - 1$ edges and still fail to be a spanning tree if the chosen edges create a cycle or leave a vertex disconnected.

## Reviewing what a spanning tree must preserve

Before proving the property, Jackie reviewed the defining properties of a spanning tree: it is undirected, it is a spanning subgraph, it is connected, and it is acyclic. These are not decorative words; they control what a valid inductive extension may do. Any extension used in the proof must still cover the relevant vertices, remain undirected, stay connected, and avoid cycles.

### Spanning-tree proof: base cases

The proof was by induction on the number of vertices. The empty case was not used because the formula would give $0 = 0 - 1$, which is false. Jackie then checked small spanning trees: for $n = 1$, $m = 0$; for $n = 2$, $m = 1$; for $n = 3$, $m = 2$. These examples support the pattern $m = n - 1$ and also show why the graph has to be connected and acyclic.

### Spanning-tree proof: exactly one new edge

For the inductive step, Jackie drew the old spanning tree as a cloud with $n$ vertices and $m = n - 1$ edges. To make a strictly larger spanning tree, add one new vertex $x$. The new vertex cannot be left isolated, because the result would not be connected. It also cannot be connected to two or more existing vertices, because the old tree already has a path between those existing vertices; adding two edges through $x$ would create a cycle. Therefore the valid minimal extension adds exactly one new edge.

### Spanning-tree proof: completing the count

After the extension, the new graph has $|V'| = n + 1$ vertices and $|E'| = |E| + 1$ edges. By the inductive hypothesis, $|E| = n - 1$, so $|E'| = (n - 1) + 1 = n$. Since $|V'| - 1 = (n + 1) - 1 = n$, the larger graph also satisfies $|E'| = |V'| - 1$. Jackie emphasized that this proof works because the inductive hypothesis is explicitly used.

### Assignment 1 connection

Near the end, Jackie pointed to the Assignment 1 starter tests and the large one-line expression used to construct a binary tree with nested `BSTNode` constructor calls. The tutorial did not fully walk through that code, but Jackie noted that students who find it hard to parse should review the short guide to be posted before the next lecture. The point is to be able to read the tree shape encoded by nested constructor calls, not merely to copy the expression.

## Key Terms, Notation, and Structures

- **Mathematical induction**: A proof method with base cases, an inductive hypothesis, and an inductive step that proves a strictly larger case by using the hypothesis.
- **Structure-preserving extension**: The operation used in the inductive step must keep the object inside the same category, such as proper binary tree or spanning tree.
- **Proper binary tree**: A non-empty binary tree in which every internal node has exactly two children.
- **Internal node**: A node with children; in a proper binary tree, it has exactly two children.
- **External node**: A leaf node with zero children.
- $N_I$: The number of internal nodes in a binary tree before extension.

- $N_E$: The number of external nodes in a binary tree before extension.
- $N_I'$, $N_E'$: The corresponding internal-node and external-node counts after an extension.
- **Proper binary tree property**: For any non-empty proper binary tree, $N_E = N_I + 1$.
- **Spanning tree**: An undirected, connected, acyclic spanning subgraph.
- $n = |V|$: The number of vertices in a graph.
- $m = |E|$: The number of edges in a graph.
- **Spanning-tree edge property**: If $G$ is a spanning tree, then $m = n - 1$, equivalently $|E| = |V| - 1$.
- **Implication direction**: The statement proved was $G$ is a spanning tree $\Rightarrow m = n - 1$; the reverse direction was explicitly not being proved.
- **Cloud diagram**: Jackie's abstraction for an already-valid tree or graph instance whose exact internal shape is not important, only its properties and counts.
- `BSTNode`: The Assignment 1 node class used in nested constructor expressions for building binary trees in starter tests.

# Key Examples and Demonstrations

## Checking the proper binary tree property on a concrete tree

**Setup:** The notes showed a proper binary tree and counted internal and external nodes.

**Main point:** The example had $N_I = 5$ and $N_E = 6$, matching $N_E = N_I + 1$. This does not prove the property, but it gives the right target before writing the induction proof.

**What to practise:** Redraw the example tree from the notes, count internal and external nodes without looking at the answer, and then explain why every internal node satisfies the proper-tree requirement.

## Induction proof for $N_E = N_I + 1$

**Setup:** Start with a non-empty proper binary tree and prove the relationship between external and internal nodes.

**Main point:** The inductive step must increase the tree by two nodes, not one. Converting an external node $n$ into an internal node and adding two external children changes the counts to $N_I' = N_I + 1$ and $N_E' = N_E + 1$, which preserves $N_E = N_I + 1$ after using the inductive hypothesis.

**What to practise:** Reproduce the proof from the notes with the before/after variables. Then try to prove the equivalent statement from the tutorial prompt about the total number of nodes being odd.

### Formulating the spanning-tree property

**Setup:** The graph-property slide stated that the number of edges is the number of vertices minus one if the graph is a spanning tree.

**Main point:** The correct implication is $G$ is a spanning tree $\Rightarrow m = n-1$. Jackie separated this from the false converse and reminded students to pay attention to "if", "only if", and "if and only if".

**What to practise:** Write the statement in both English and symbolic form. Then draw a small graph with $n$ vertices and $n-1$ edges that is not a spanning tree because it is disconnected or cyclic.

### Induction proof for spanning trees

**Setup:** Assume an existing spanning tree has $n$ vertices and, by the inductive hypothesis, $n-1$ edges. Add a new vertex $x$.

**Main point:** A valid extension to a larger spanning tree must add exactly one edge from $x$ to the old tree. Zero edges makes $x$ disconnected. Two or more edges create a cycle because the old tree already has a path between any two existing vertices.

**What to practise:** Redraw the cloud diagram from the notes. Explain why one new edge preserves connectedness and acyclicity, then redo the final count $|E'| = |V'| - 1$ without looking.

### Assignment 1 one-line `BSTNode` construction

**Setup:** The notes showed the `BSTNode` constructors and a starter-test expression with deeply nested constructor calls.

**Main point:** The expression encodes a tree recursively: each internal-node constructor contains a key/value and two child subtrees, while each external-node constructor marks a leaf placeholder. Reading this expression is part of understanding the test cases.

**What to practise:** Take a small nested `BSTNode` expression and draw the corresponding tree. Then reverse the exercise: draw a small tree and write the corresponding constructor expression.

## Common Pitfalls and Misconceptions

### Extending a proper binary tree by only one node

**Pitfall:** Adding a single child to an external node and treating the result as a proper binary tree.

**Why it causes problems:** The resulting internal node would have one child, violating the definition of proper binary tree.

**How to avoid it:** In the inductive step for proper binary trees, convert an external node into an internal node by adding exactly two external children.

## Using node names as if they were numeric counts

**Pitfall:** Writing an expression such as subtracting node $n$ or adding nodes $x, y$ as if the symbols themselves were numbers.

**Why it causes problems:** Nodes are objects or positions; counts must change by numeric amounts such as $-1$, $+1$, or $+2$.

**How to avoid it:** Use count variables such as $N_E$, $N_I$, $N'_E$, and $N'_I$, and explain the numerical change: lose one external node, gain two external nodes.

## Forgetting to use the inductive hypothesis

**Pitfall:** Completing an inductive step without ever substituting or invoking the assumed property for the smaller case.

**Why it causes problems:** The proof no longer connects the larger case to the smaller case, so it is not an induction proof.

**How to avoid it:** In each inductive step, identify the exact line where the inductive hypothesis is used, such as substituting $N_E = N_I + 1$ or $|E| = |V| - 1$.

## Reversing the implication for spanning trees

**Pitfall:** Concluding that any graph with $m = n - 1$ must be a spanning tree.

**Why it causes problems:** The edge count alone does not guarantee connectedness or acyclicity. Jackie explicitly treated the converse as not being proved and generally false.

**How to avoid it:** Remember the proven direction: spanning tree $\Rightarrow m = n-1$. To identify a spanning tree, also check spanning, connectedness, acyclicity, and undirectedness.

## Adding too many edges in the spanning-tree induction step

**Pitfall:** Adding a new vertex to the old spanning tree and connecting it to two or more old vertices.

**Why it causes problems:** The old tree is connected, so there is already a path between those old vertices; the two new edges through the new vertex create a cycle.

**How to avoid it:** For the spanning-tree proof, add exactly one edge from the new vertex to the existing tree.

## Treating nested constructor code as unreadable text

**Pitfall:** Skipping the large one-line `BSTNode` expression because it looks too long.

**Why it causes problems:** The starter tests specify tree shapes using those expressions, so misunderstanding them can lead to misunderstanding what the test expects.

**How to avoid it:** Parse the expression recursively: identify the root constructor first, then the left subtree, then the right subtree, and draw the tree as you go.

## What You Should Be Able to Do

After this tutorial, you should be able to:

- explain what makes a binary tree proper and distinguish internal nodes from external nodes;
- count $N_I$ and $N_E$ on a concrete proper binary tree and check whether $N_E = N_I + 1$ holds;
- reproduce the induction proof that every non-empty proper binary tree satisfies $N_E = N_I + 1$;
- explain why the proper-binary-tree inductive step must add two nodes rather than one;
- translate "$m = n - 1$ if $G$ is a spanning tree" into the implication $G$ is a spanning tree $\Rightarrow$ $m = n - 1$;
- list the properties that define a spanning tree: undirected, spanning, connected, and acyclic;
- justify why extending a spanning tree by one new vertex requires exactly one new edge;
- reproduce the induction proof that a spanning tree with $n$ vertices has $n - 1$ edges;
- give a counterexample showing that $m = n - 1$ alone does not guarantee a spanning tree;
- read a nested `BSTNode` constructor expression well enough to draw the intended binary tree.

## Suggested Practice

- Redraw the proper binary tree example from the notes and recount $N_I$ and $N_E$ before reviewing the proof.
- Reproduce the proof of $N_E = N_I + 1$ on paper, making the two before/after equations $N_I' = N_I + 1$ and $N_E' = N_E + 1$ explicit.
- Try the tutorial prompt's equivalent claim that the total number of nodes in a non-empty proper binary tree is odd, using the same induction idea.
- Redraw the spanning-tree cloud diagram and explain in words why zero new edges and two new edges are both invalid extensions.
- Reproduce the proof that a spanning tree satisfies $|E| = |V| - 1$, and mark the exact line where the inductive hypothesis is used.
- Complete the graph taxonomy table from the previous lecture before the next class, since Jackie planned to continue from it.

- For Assignment 1, take the one-line `BSTNode` expression from the starter tests and draw the tree it constructs.
- Create a small false-converse example: a graph with $n$ vertices and $n-1$ edges that is not a spanning tree, and explain which defining property fails.

## Where to Review

- Review the Tutorial 5 iPad/PDF notes page 1 for the tutorial scope: mathematical induction, Assignment 1, induction on trees and graphs, and the one-line BST construction topic.
- Review page 2 for the graph property slide and the annotated proof sketch for spanning trees, including implication direction, base cases, and the before/after edge count.
- Review page 3 for the cloud diagram showing why adding two edges from a new vertex creates a cycle and is not a valid spanning-tree extension.
- Review page 5 for the proper binary tree proof, especially the conversion of an external node into an internal node with two new external children.
- Review page 6 for the `BSTNode` constructors and the starter-test one-line construction style for Assignment 1.
- Revisit Lecture 9's mathematical induction template and Lecture 10's definitions of spanning subgraph, connected graph, forest, tree, and spanning tree.
- Review the math review slide on implication language if "if", "only if", and "if and only if" are still confusing.

## Closing Reminder

This tutorial digest is a roadmap for the two induction proofs and the Assignment 1 code-reading connection. It is not enough to read the final equations: you need to redo the proofs, justify the extension steps, and practise converting between diagrams, formulas, and code. Revisit Jackie's notes and recording, complete the taxonomy table, and work through the Assignment 1 tree-construction examples actively before relying on the digest for review.