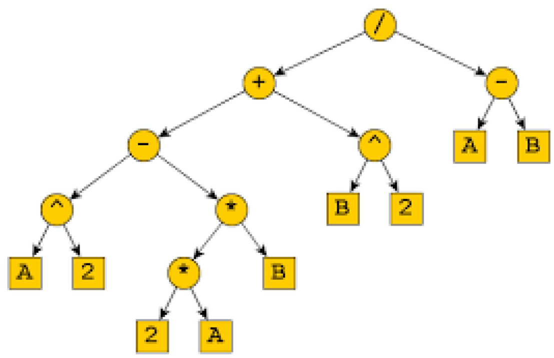
Lecture 29 Rooted Trees
Copyright Michael Monagan

November 27, 2020 11:14 AM

# Lecture 29 Rooted Trees


Grimaldi 12.2

Assignment #8 due Mon. Dec 7th.
Worth 3% of your grade.
Final exam
Part I 20% rest.
Part II 20% A7 and A8



What formula does this tree encode?

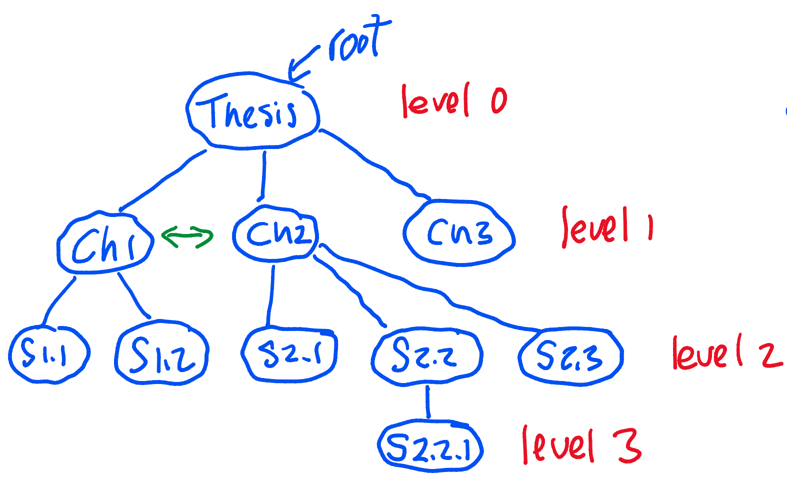


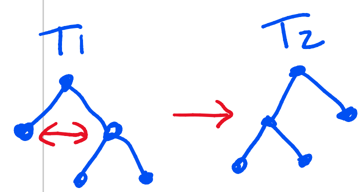
# Ordered rooted trees

For some applications it is essential to have not just a rooted tree, but also an ordering of the children for each internal vertex.

Example

Thesis
  Ch1
    S1.1
    S1.2
  Ch2
    S2.1
    S2.2
      S2.2.1
    S2.3
  Ch3



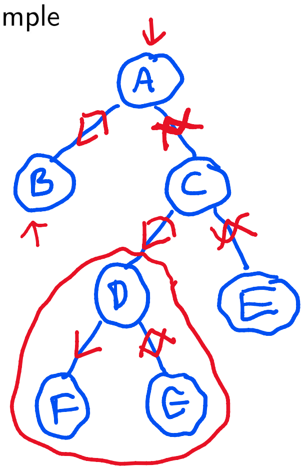



How do we walk through and process a rooted tree?

## Definition ( preorder, postorder tree traversals )

A **preorder traversal** of a tree $T$ first visits the root vertex then visits, in preorder, the vertices of the subtrees $T_1, T_2, \ldots, T_k$ of $T$.
A **postorder traversal** of a tree $T$ visits, in postorder, the vertices of the subtrees $T_1, T_2, \ldots, T_k$ of $T$ then visits the root.

Example



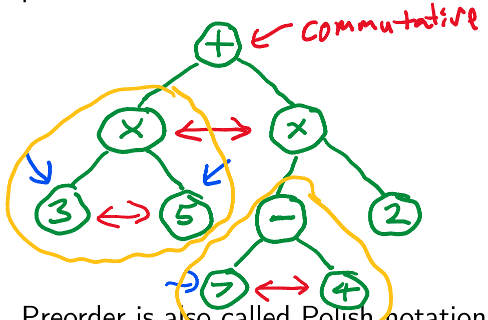

preorder: A B C (D F G) E

postorder: B (F G D) E C A



Exercise. Draw the expression tree for $(3 \times 5) + ((7 - 4) \times 2)$ and give the postorder traversal.

15 + 3×2 = 21.

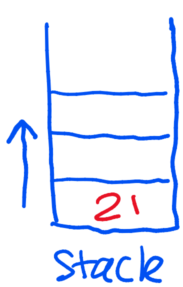



postorder: (3 5 ×) (7 4 −) 2 × +

Preorder is also called Polish notation and postorder is also called reverse Polish notation. HP calculators used postorder and a stack to evaluate expressions.

Don't need brackets!

3 5 × 7 4 − 2 × +

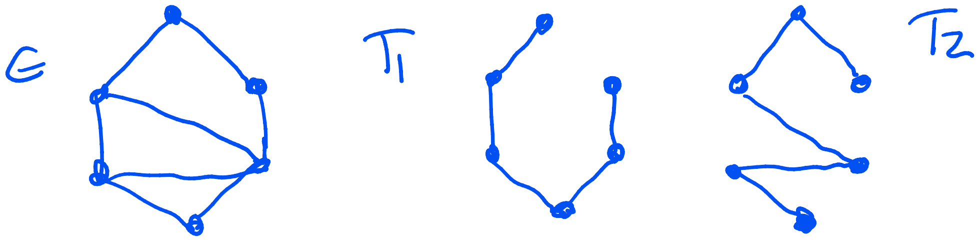



Rules: Number? Put it on the top of the stack.
Operator? Take the top 2 items off the stack, operate, and put the result on the top of the stack.

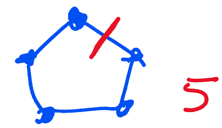
## Definition ( spanning tree )

Let $G$ be a connected multigraph. A subgraph $T$ of $G$ is a **spanning tree** if $T$ spans $G$ (so $T$ contains all vertices in $G$) and $T$ is a tree.

Example




How many spanning trees does a cycle have?






## Theorem ( existence of spanning trees )

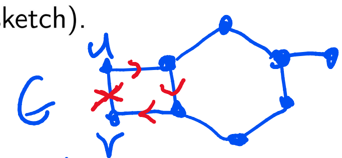
*Every connected multigraph $G = (V, E)$ has a spanning tree.*
*Here are two algorithms to select a spanning tree in $G$:*

(1) *Start from $G$. If there is a cycle $C$ in $G$ delete an edge from $C$. Repeat this until $G$ has no cycles. Output $G$.*

(2) *Create the graph $H = (V, \phi)$. For each edge $e$ in $G$ add $e$ to $H$ if it does not make $H$ have a cycle. Output $H$.*

Proof (sketch).

(1)



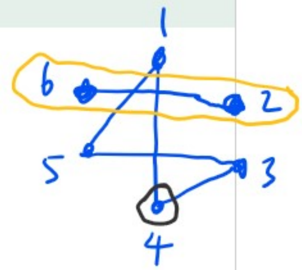

is connected.

Deleting an edge on a cycle in G preserves connectivity.
End up with a connected graph with no cycles. A tree by def. with all the vertices.

(2) Exercise.

# Graph Algorithms

Consider the graph $G = (V, E)$ where $V = \{1, 2, 3, 4, 5, 6\}$ and $E = \{\{1, 5\}, \{5, 3\}, \{4, 1\}, \{4, 3\}, \{2, 6\}\}$.




How can a computer test if $G$ is connected? planar?
If $G$ is connected, how can it find a spanning tree in $G$?
If $G$ is planar, how can it find a planar embedding of $G$?

Most algorithms need to visit the **neighbors** of a vertex.
**Question:** What is a good way to store the edges?

## Definition ( list of neighbors )

The **list of neighbors** representation for $E$ is an array $A$ of size $n = |V|$ where $A_i$ is the set of neighbors of vertex $i$ (the vertices adjacent to $i$).

Example

A

| 1 | 2 | 3 | 4 | 5 | 6 |
|---|---|---|---|---|---|
| {4,5} | {6} | {4,5} | {1,3} | {1,3} | {2} |

**Question:** How do algorithms walk through the graph?



**The Depth-First Search (DFS) algorithm**

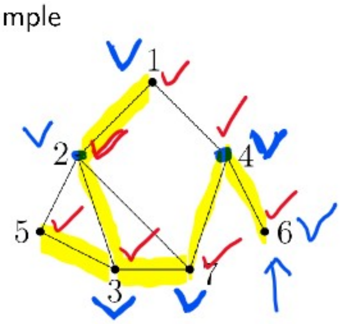
Input. A graph $G = (V, E)$.
Output. A set $E_T$ of edges such that $(V, E_T)$ is a spanning tree of $G$.

1. Let $v = 1$, $E_T = \phi$ and and mark vertex 1 as visited.
2. If all neighbors of $v$ have been visited Then
   a) If $v = 1$ Then Return $(V, E_T)$.
   b) Else (backtrack step) Let $v = parent(i)$ and Goto step 2.
3. Else
   a) Let $i$ be the smallest neighbor of $v$ that has not been visited.
   b) Mark $i$ as visited.
   c) Add the edge $\{v, i\}$ to $E_T$ and Let $parent(i) = v$.
   d) Let $v = i$ and Goto step 2.

error
i should be v

Modify this DFS algorithm to test if G is connected, has a cycle.

Example




Depth First Search Spanning Tree.

$E_T = \{\{1,2\}, \{2,3\}, \{3,5\}, \{3,7\}, \{4,7\}, \{4,6\}\}$

parent

| 1 | 2 | 3 | 4 | 5 | 6 | 7 |
|---|---|---|---|---|---|---|
| | 1 | 2 | 7 | 3 | 4 | 3 |

mark array.

## Additional Space