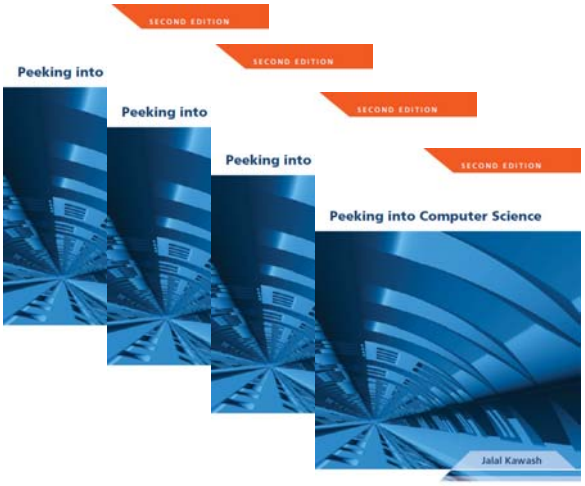
Graphs

Peeking into Computer Science

Reading Assignment

- Mandatory: Chapter 3 – Sections 3.3 & 3.4

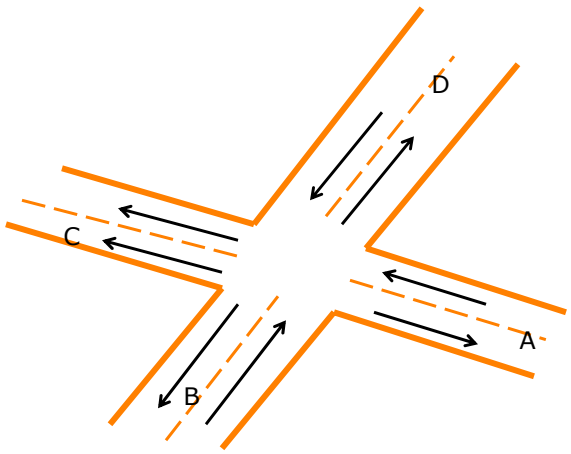


Graph Coloring

At the end of this section, you will be able to:

1. Understand (once more) how graphs are used to represent data
2. Learn the graph coloring algorithm
3. Apply graph coloring to solve for various scheduling problems

Objectives

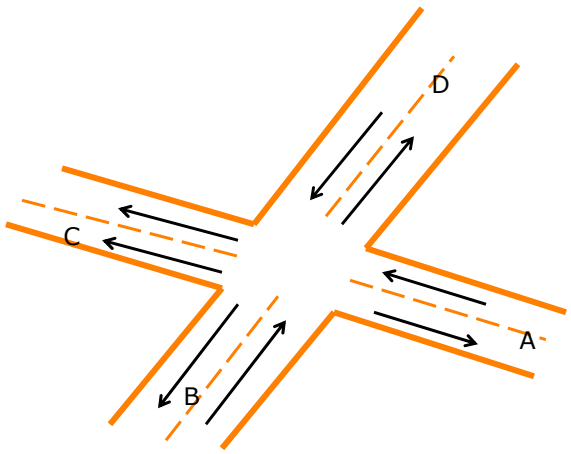


Based on Aho, Hopcroft, and Ullman, *Data Structures and Algorithms,* Addison-Wesley

Simple Intersection

- Design a traffic light for a complex intersection
- Identify allowed turns
  - Going straight is a turn!
- Group turns so that permitted simultaneous turns are in the same group
- Find the smallest number of groups
  - This means a traffic light with the smallest number of phases

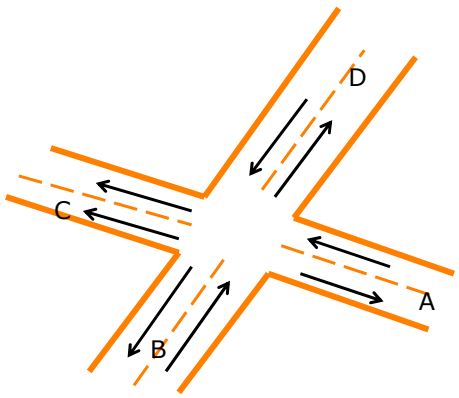
Another Example Problem



Based on Aho, Hopcroft, and Ullman, *Data Structures and Algorithms,* Addison-Wesley

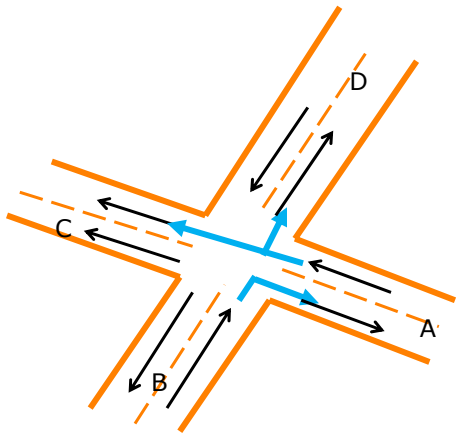
Simple Intersection

- From A to B (denoted AB)
- From A to C (AC)
- From A to D (AD)
- BA
- BC
- BD
- DA
- DB
- DC



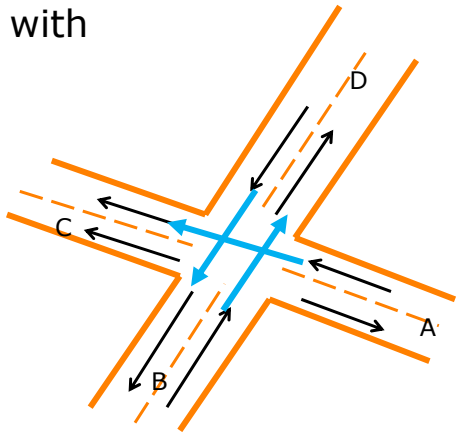
9 Possible Turns

- AC
- AD
- BA



Example Simultaneous Turns

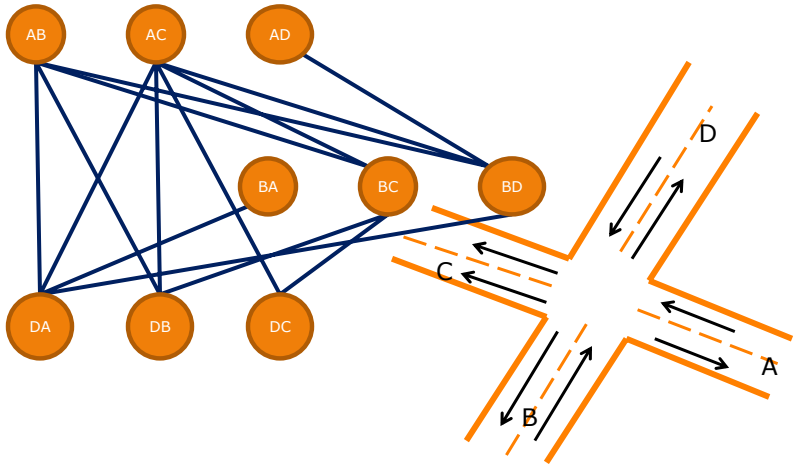
- AC conflicts with
- DB
- BD
- BC
- DC



Example Conflicting Turns

- Build a graph model
- Vertices are turns
- An edge connects conflicting turns

Objective



Graph Model of Conflicting Turns

- Color the vertices of the graph
- No two adjacent vertices can have the same color
  - Adjacent: and edge connects them
- Each color represents a group of turns that can be simultaneous
- Note: more than one solution

Graph Coloring

- Repeat the following two steps until the graph is colored

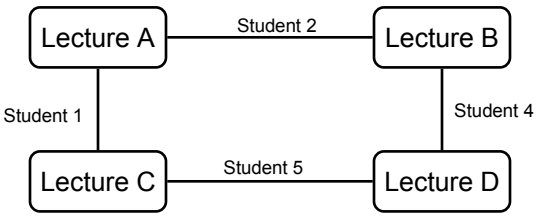
1. Select an uncolored vertex and color it with a ***new*** color, **C**
2. For each uncolored vertex,
   - Determine if it has an edge with a vertex that is colored with color **C**
   - If not, color it with color **C**
   - If yes, skip it

Graph Coloring Algorithm

- You are to schedule final exams so that a student will not have two final exams scheduled at the same time.
- Although you could schedule each exam in it's own individual time slot (i.e., no two exams run simultaneously) this would be highly inefficient.
  - Another constraint is to schedule exams with the minimum number of time slots.
- Examinations for a lecture will be represented with vertices.
- A pair of vertices will be connected if there is at least one student who is registered in both.

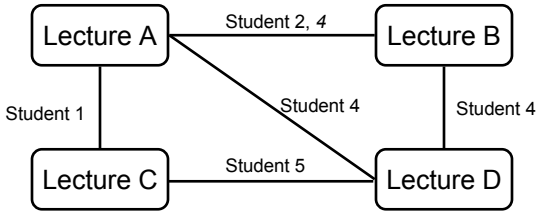
JT's Extra, Graph Coloring Example 1: Final Exams

- Lecture A:
  - Student 1, student 2
- Lecture B:
  - Student 2, student 4, student 6
- Lecture C:
  - Student 1, student 3, student 5
- Lecture D:
  - Student 4, student 5

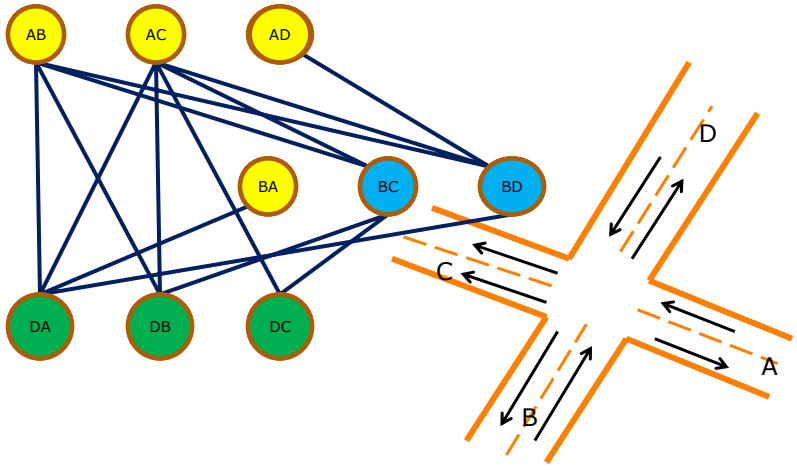


JT's Extra, Graph Coloring Example 1: Final Exams (2)

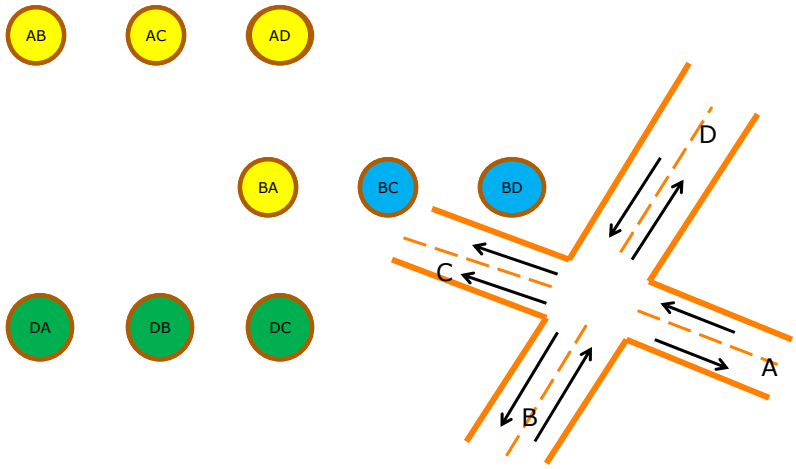
- Lecture A:
  - Student 1, student 2, *student 4*
- Lecture B:
  - Student 2, student 4, student 6
- Lecture C:
  - Student 1, student 3, student 5
- Lecture D:
  - Student 4, student 5



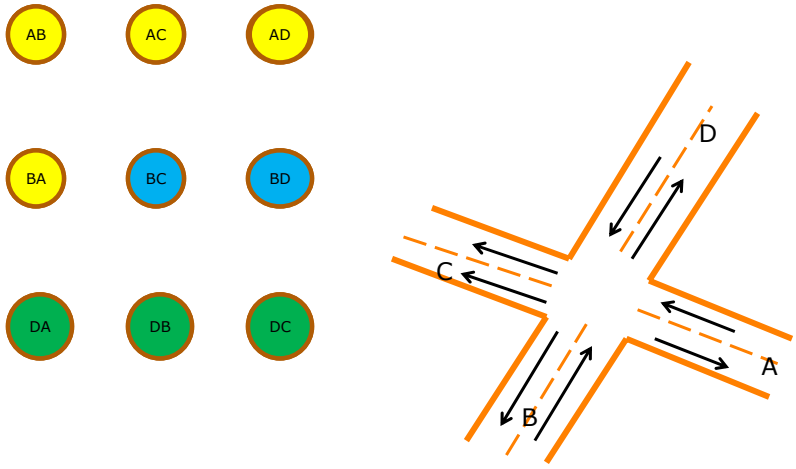
JT's Extra, Graph Coloring Example 2: Final Exams



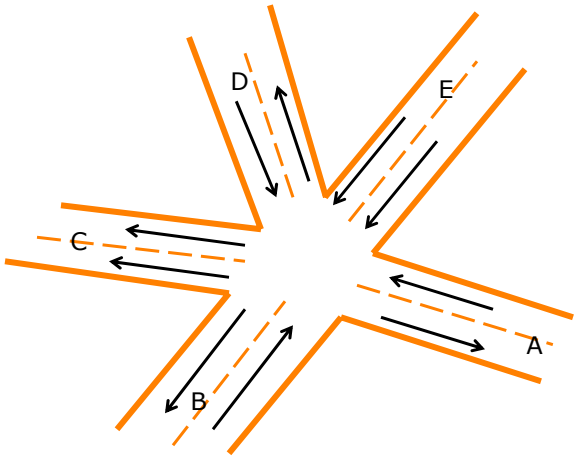
Graph Model of Conflicting Turns



Graph Model of Conflicting Turns

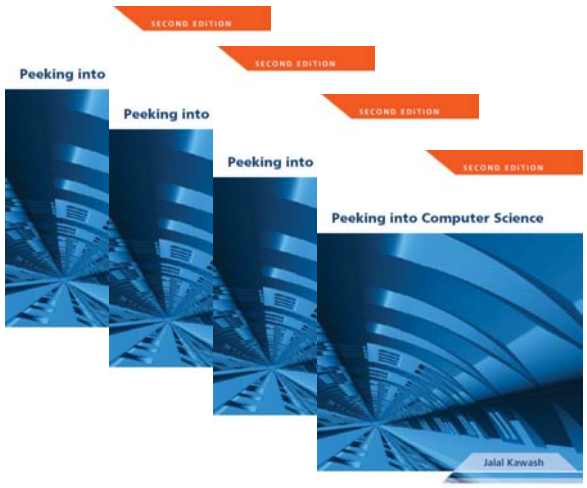


Graph Model of Conflicting Turns



From Aho, Hopcroft, and Ullman, *Data Structures and Algorithms,* Addison-Wesley

Complex Intersection



Trees

A popular special case of graphs

At the end of this section, the student will be able to:

1. Identify when a graph is a tree
2. Use tree terminology
3. Identify different applications of trees
4. Define binary trees

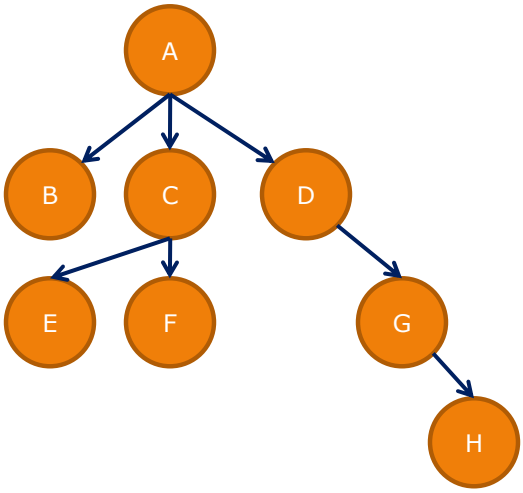
Objectives

- Trees are a special case of graphs
- A *tree* is a (directed) graph with the properties:

1. There is a designated vertex, called the *root* of the tree
2. There is a unique directed path from the root to every other vertex in the tree

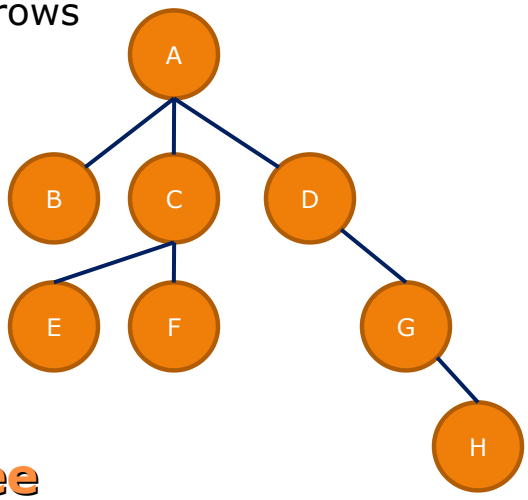
- Grow downwards!

Computer Science Trees



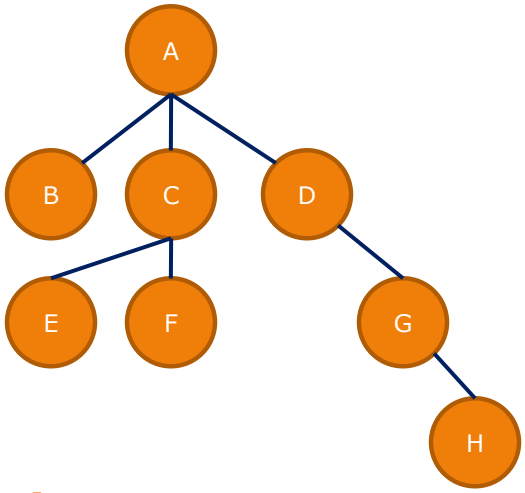
Example Tree

- A tree always "grows" downward
- We omit the arrows

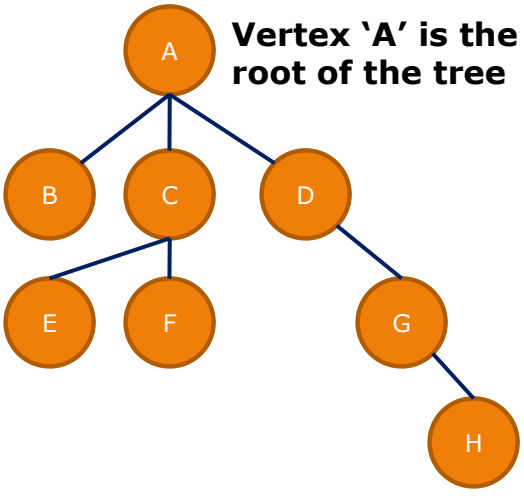


Example Tree

- Root
- Child
- Parent
- Ancestor
- Descendant
- Distance
- Siblings
- Leaf



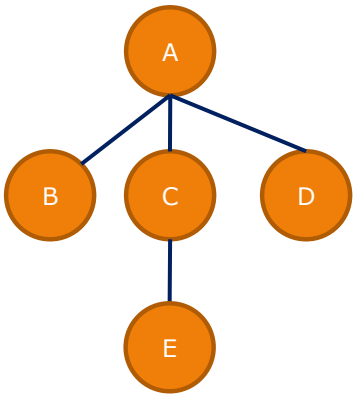
Tree Terminology



Vertex 'A' is the root of the tree

Root

- A is the parent of B, C, D.
- B, C, D are the children of A.
- B,C,D,E are descendants of A.
- A is ancestor of B, C, D, E

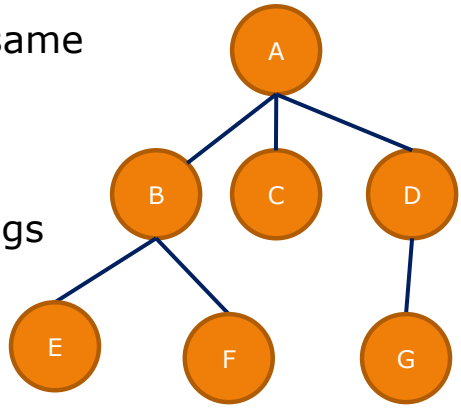


Parent-Child (Above/Below), Ancestor-Descendant

- All vertices have exactly one parent except for the root (which has none).

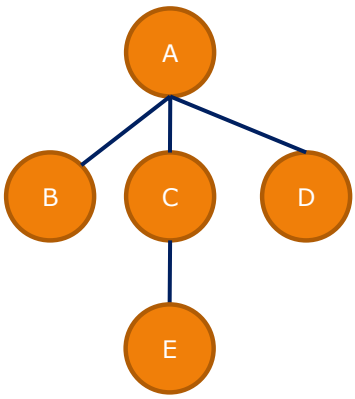
Parent-Child

- Siblings have the same parent.
- B,C,D are siblings
- E, F are siblings
- A, G have no siblings



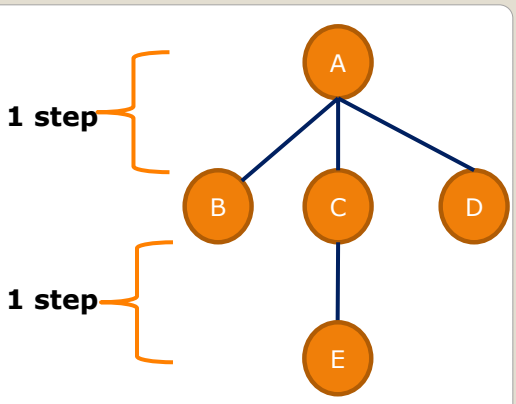
Siblings

- Leaves are vertices with no children
- B, D, E are leaves
- "Bottom level"

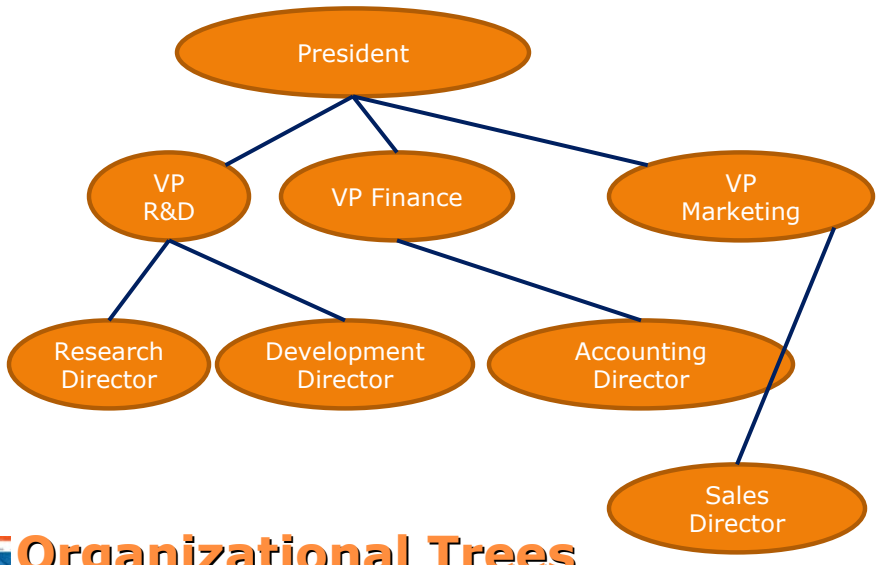


Leaf

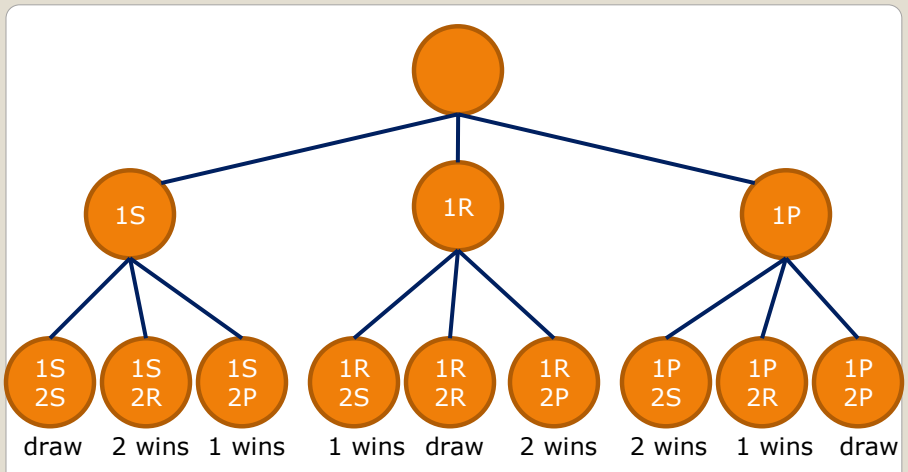
- Moving up from a node to the parent is one 'step'.
- Moving up to the parent of the parent is another 'step'.



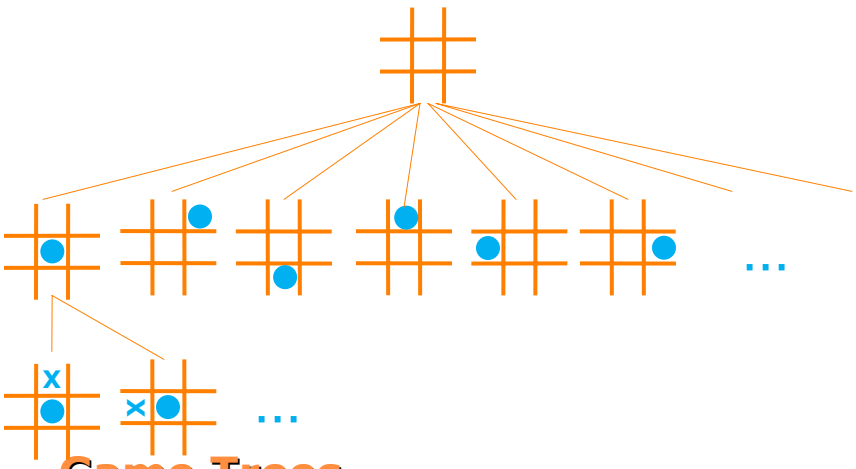
Distance



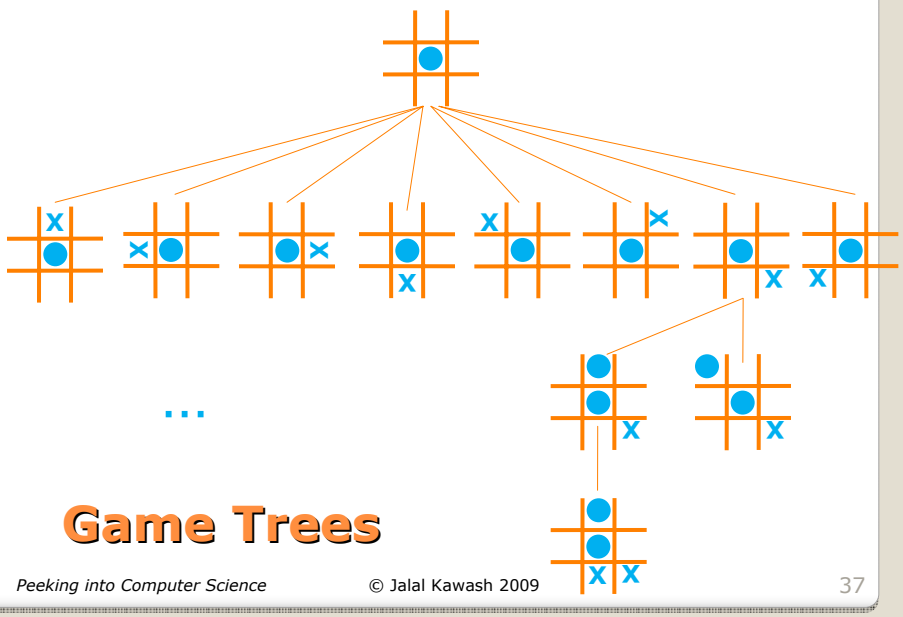
Organizational Trees



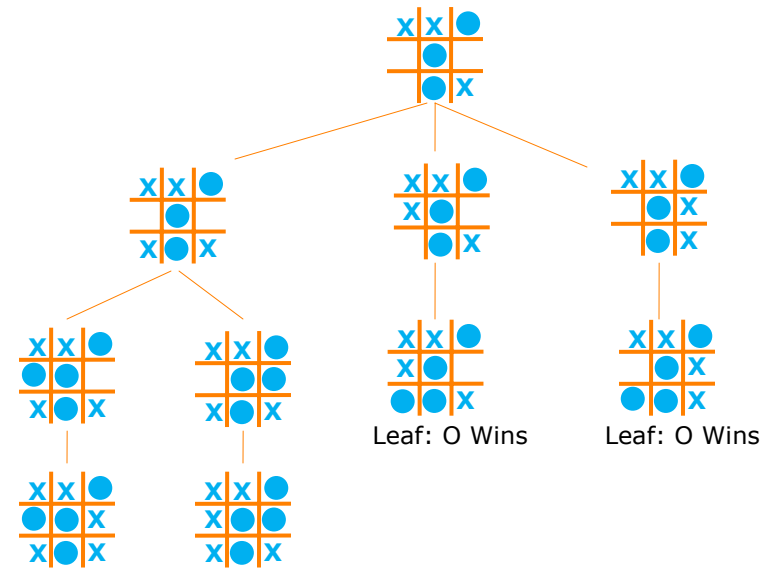
Game Trees
Paper-Rock-Scissors



Game Trees
Tic-Tac-Toe



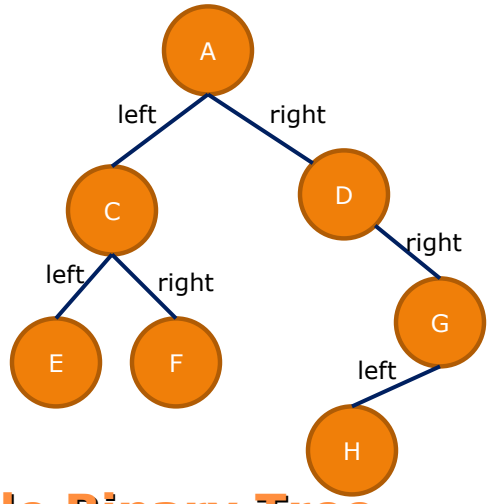
Game Trees



- A binary tree is a tree with the following properties:

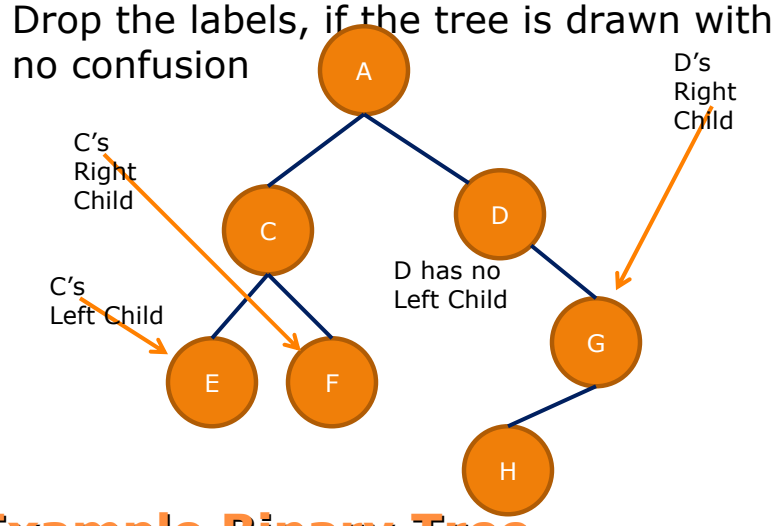
1. The edges are labeled with the labels *left* or *right*
2. Every vertex has at most two children; if both children exist, then one edge must be labeled with *left* and the other *right*.

Binary Trees



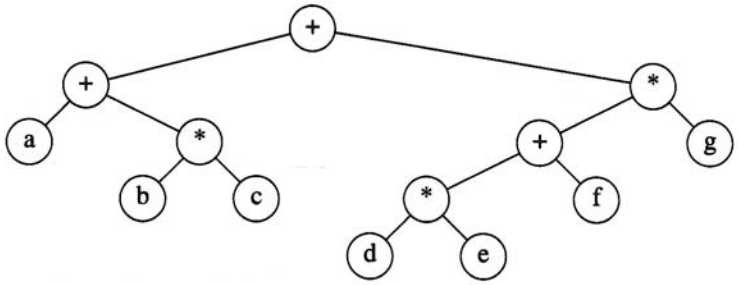
Example Binary Tree

- Drop the labels, if the tree is drawn with no confusion



Example Binary Tree

Figure 4.14 Expression tree for $(a + b * c) + ((d * e + f) * g)$ [1]



1 From: "Data Structures and Algorithm Analysis in C++", Weiss M.A.

JT's Extra: Application Binary Trees s2

Slide 42

s2 Weiss page 123
sysman, 9/28/2012