Trees

Today: introduction to trees

Following lectures:

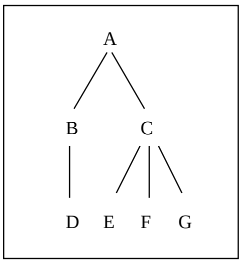
- binary search trees
- height balanced search trees
- B-trees
- heaps

Trees

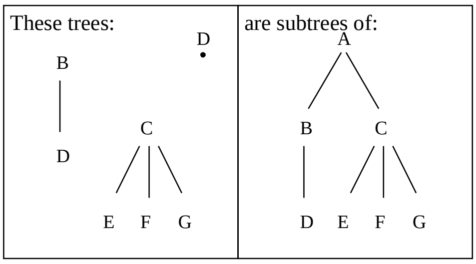
- Common data structure for non-linear collections. In today's lecture:
  - Tree terminology
  - Kinds of binary trees
  - Size and depth of trees

Trees

- Root: topmost node (A)
- Parents & children (mothers & daughters) (B daughter of A)
- Descendants & siblings (sisters) (B and C)
- Leaf nodes: nodes with no children (D)

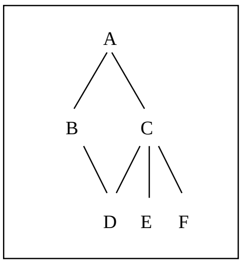


Subtrees



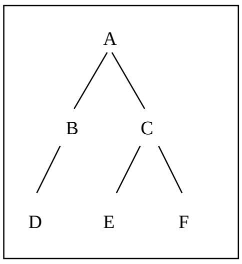
Not a Tree

- Nodes cannot have more than one parent



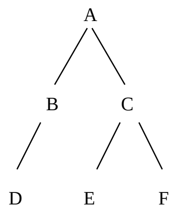
Binary Trees

- Nodes cannot have more than two children (can have 0,1, or 2 children).
- In binary trees each daughter is either left or right (even if there is only one).



Size of Tree

is just the number of nodes in the tree. Size 6:



Depth (Height) of Tree

- is the length of the longest path from root to a leaf node.
- Length of path = number of nodes on path.
- Level (depth) of a node: length of path from root to that node (not counting the node).

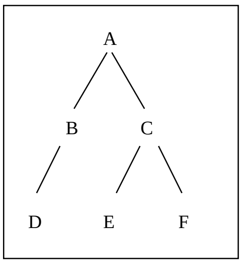
Level of a node

- This tree has depth 3:

Level 0

Level 1

Level 2



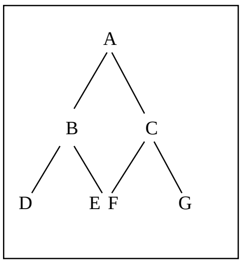
Various types of balanced trees

Useful in tree searching algorithms

- Perfectly balanced binary trees
- Complete binary trees
- Height balanced binary trees

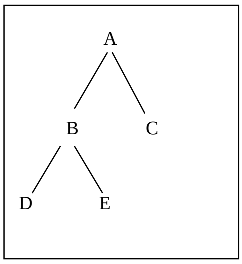
Perfectly Balanced Binary Trees

- Left and right subtrees have same depth, and are themselves perfectly balanced.
- In other words, as full as possible. (Different from *full trees* in Shaffer!)



Perfectly Balanced Binary Trees

- Not perfectly balanced:



How Many Nodes in a Perfectly Balanced Tree?

- Running time of most tree algorithms depends on the height of the tree.
- Perfectly balanced tree is the best possible case for those algorithms.
- Useful to know how many items we can put in a perfectly balanced tree of height n,
- and vice versa: how many levels does a perfectly balanced tree of size x have?

First some simple facts

- Fact 1: each level in a perfectly balanced binary tree contains twice more nodes than the previous level (apart from level 0 which does not have a previous level).
- Fact 2: level k contains 2^k nodes.

  Proof of fact 2: by induction on k.
  - Base case: for k=0 Fact 2 holds ($2^0 = 1$).
  - Inductive step: assume level k-1 has 2^{k-1} nodes. By the Fact 1, level k has $2*2^{k-1}$ nodes which is 2^k nodes.

How many nodes?

Theorem: A perfectly balanced binary tree of depth n contains $2^n - 1$ nodes.

Proof: by induction on n.

- Base case: n=1. The tree contains $2^1 - 1 = 1$ node.
- Inductive step: assume a tree of depth n-1 contains $2^{n-1} - 1$ nodes. A tree of depth n has one more level (n-1) which contains 2^{n-1} nodes (Fact 2). The total number of nodes in the tree of depth n is:
  $2^{n-1} - 1 + 2^{n-1} = 2^n - 1$.

How many levels?

Given that perfectly balanced binary tree of depth n contains $2^n - 1$ nodes, how many levels does a tree of size x have?

How many levels?

Given that perfectly balanced binary tree of depth n contains $2^n - 1$ nodes, how many levels does a tree of size x have?

$x = 2^n - 1$

How many levels?

Given that perfectly balanced binary tree of depth n contains $2^n - 1$ nodes, how many levels does a tree of size x have?

$x = 2^n - 1$

$2^n = x + 1$

How many levels?

Given that perfectly balanced binary tree of depth n contains $2^n - 1$ nodes, how many levels does a tree of size x have?

$x = 2^n - 1$

$2^n = x + 1$

$n = \log_2 (x + 1)$

How many levels?

Given that perfectly balanced binary tree of depth n contains $2^n - 1$ nodes, how many levels does a tree of size x have?

$x = 2^n - 1$

$2^n = x + 1$

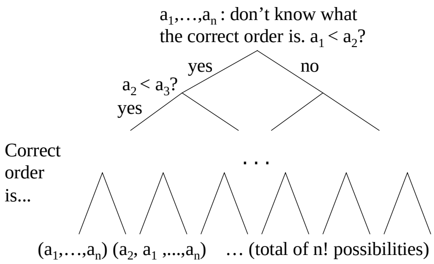
$n = \log_2 (x + 1)$

The number of levels is $\log_2 (x + 1)$.

Lower bound for comparison sorting

- Can model sorting which depends on comparisons between elements as a binary decision tree.
- At each node, a comparison between two elements is made; there are two possible outcomes and we find out a bit more about the correct order of items in the array.
- Finally arrive at full information about the correct order of the items in the array.

Comparison sorting



Comparison sorting

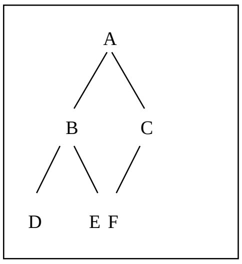
- If a binary tree has n! leaves than the minimal number of levels (assuming the tree is perfect) is log n! +1.
- This shows that O(n log n) sorting algorithms are essentially optimal (log n! is not equal to n log n but has the same growth rate modulo some hairy constants).

Problem with Perfectly Balanced Trees

- Perfectly balanced binary trees only come in $2^n - 1$ sizes: 0, 1, 3, 7, 15, …
- If we need a tree of a different size, have to compromise a bit.
- Use complete trees instead.

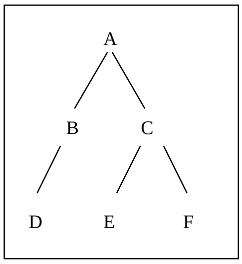
Complete Binary Trees

- Perfectly balanced, except possibly at the lowest level, and
- All the leaves at the lowest level are as far to the left as possible (it is filled from left to right level by level)



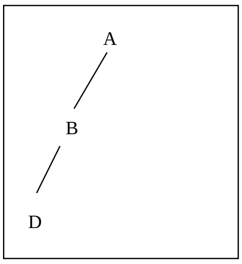
Complete Binary Trees

- NOT A COMPLETE BINARY TREE



Complete Binary Trees

- NOT A COMPLETE BINARY TREE



Informal exercise

- Prove by mathematical induction that a perfectly balanced binary tree with k levels contains 2^{k-1} leaves.
- Calculate how many levels a complete binary tree of size x has.

Reading

- Shaffer, Chapter 5.1, 8.9 (lower bounds for sorting - optional).