(1)

Lec. 35. Nov. 22.

Tree: connected acyclic undirected graph.

Thm 1 (p. 527.) A graph G is a tree iff there is a unique path between any two vertices.

pf:
- Suppose G is a tree. Let u, v be any two vertices in G. G connected $\Rightarrow \exists$ path p_1 between u and v. Suppose $\exists$ also a path $p_2 \neq p_1$ from u to v. Then $p_1 + p_2$ form a cycle, contradicting G being a tree. Thus p_2 cannot exist, and p_1 is unique.
- Now suppose G is a graph with a unique path between any u, v. Then G is connected by definition. Suppose G has a cycle and let x, y be on the cycle. Then there are two different paths from x to y (think!) — contradiction. So G has no cycles, and it is a tree.

Def: rooted tree: just pick any vertex as the root r, and let the rest of the tree "hang" from r. Can now speak of parents and children.

Def: leaf: a tree vertex w/ no children.
internal node: a tree vertex w/ children.

(2)

Lec 35
Nov. 22

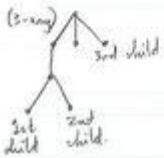
Def. (p. 531).

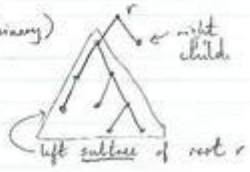
- A tree is m-ary if every vertex has at most m children.
- full m-ary tree: every one (except leaves) has exactly m children.

eg: full binary tree (rooted): 

eg: 3-ary tree (rooted) (not full): 

Def: ordered tree: order of children matters. So, can speak of left and right children (binary trees), or 2nd or 6th child (general m-ary trees).

eg: (3-ary) 

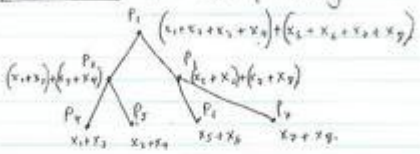
eg: (binary) 

(3)

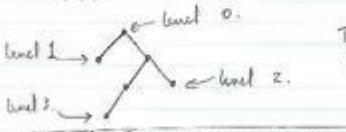
Lec 35.

Example 7, p. 535 File system (eg. UNIX, DOS)

Example 8, p. 535 Parallel processing.



Def:
- The level of a vertex v in a tree T is the length of the path from the root r to v (i.e. # of edges from r to v).
- Height of a tree: maximum level of all vertices.

eg: 

TREE WITH HEIGHT 3

Binary Search Trees: Example 1, p. 541-542.

Decision Trees: Example 2, p. 541.

Prefix trees: Try to save space. High-frequency letters → shorter bit strings. Prefix code: bit string for x would not be the first part of bit string for y.
Figure 4.