Sec. 11.1 12, 20, 21, 28

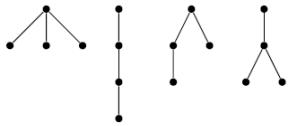
***12. a)** How many nonisomorphic unrooted trees are there with four vertices?

b) How many nonisomorphic rooted trees are there with four vertices (using isomorphism for directed graphs)?

12. We find the answer by carefully enumerating these trees, i.e., drawing a full set of nonisomorphic trees. One way to organize this work so as to avoid leaving any trees out or counting the same tree (up to isomorphism) more than once is to list the trees by the length of their longest simple path (or longest simple path from the root in the case of rooted trees).

a) There are two trees with four vertices, namely $K_{1,3}$ and the simple path of length 3. See the first two trees below.

b) The longest path from the root can have length 1, 2 or 3. There is only one tree with longest path of length 1 (the other three vertices are at level 1), and only one with longest path of length 3. If the longest path has length 2, then the fourth vertex (after using three vertices to draw this path) can be "attached" to either the root or the vertex at level 1, giving us two nonisomorphic trees. Thus there are a total of four nonisomorphic rooted trees on 4 vertices, as shown below.



20. How many leaves does a full 3-ary tree with 100 vertices have?

20. By Theorem 4(i), the answer is $[(m-1)n+1]/m = (2 \cdot 100 + 1)/3 = 67$.

21. Suppose 1000 people enter a chess tournament. Use a rooted tree model of the tournament to determine how many games must be played to determine a champion, if a player is eliminated after one loss and games are played until only one entrant has not lost. (Assume there are no ties.)

21. 999

28. How many vertices and how many leaves does a complete m-ary tree of height h have?

28. This tree has 1 vertex at level 0, m vertices at level 1, m^2 vertices at level 2, ..., m^h vertices at level h. Therefore it has

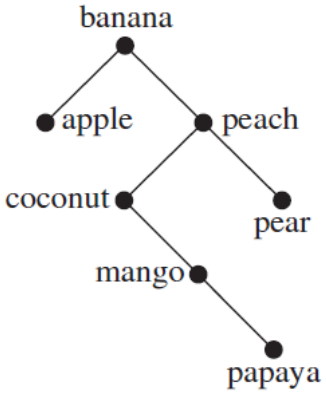
$$1 + m + m^2 + \cdots + m^h = \frac{m^{h+1} - 1}{m - 1}$$

vertices in all. The vertices at level h are the only leaves, so it has m^h leaves.

Sec.11.2 1, 20, 23

1. Build a binary search tree for the words *banana, peach, apple, pear, coconut, mango*, and *papaya* using alphabetical order.

1.

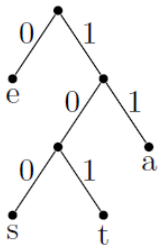


20. Construct the binary tree with prefix codes representing these coding schemes.

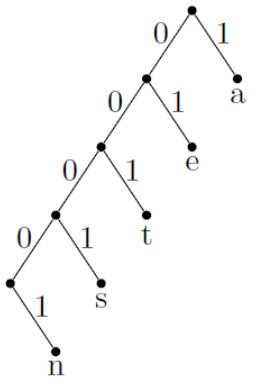
a) *a*: 11, *e*: 0, *t*: 101, *s*: 100
b) *a*: 1, *e*: 01, *t*: 001, *s*: 0001, *n*: 00001
c) *a*: 1010, *e*: 0, *t*: 11, *s*: 1011, *n*: 1001, *i*: 10001

20. The constructions are straightforward.

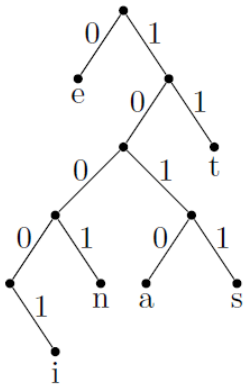
a)



b)



c)



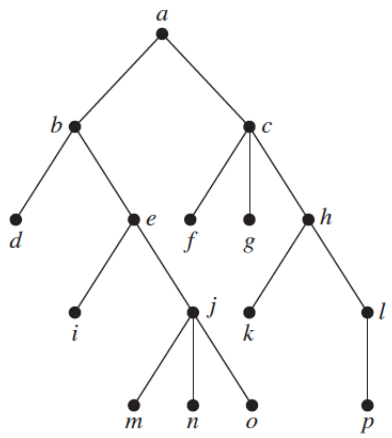
23. Use Huffman coding to encode these symbols with given frequencies: *a*: 0.20, *b*: 0.10, *c*: 0.15, *d*: 0.25, *e*: 0.30. What is the average number of bits required to encode a character?

23. a: 11; b: 101; c: 100; d: 01; e: 00; 2.25 bits (Note: This coding depends on how ties are broken, but the average number of bits is always the same.) **25.** There are four possible

Sec. 11.3 8, 16

In Exercises 7–9 determine the order in which a preorder traversal visits the vertices of the given ordered rooted tree.

8.



8. See the comments in the solution to Exercise 7 for the procedure. The only difference here is that some vertices have more than two children: after listing such a vertex, we list the vertices of its subtrees, in preorder, from left to right. The answer is $a, b, d, e, i, j, m, n, o, c, f, g, h, k, l, p$.

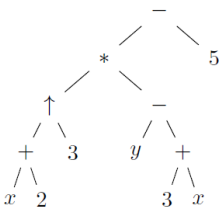
16. a) Represent the expression $((x+2) \uparrow 3) * (y-(3+x)) - 5$ using a binary tree.
Write this expression in

b) prefix notation.

c) postfix notation.

d) infix notation.

16. **a)** We build the tree from the top down while analyzing the expression by identifying the outermost operation at each stage. The outermost operation in this expression is the final subtraction. Therefore the tree has the symbol $-$ at its root, with the two operands as the subtrees at the root. The right operand is clearly 5, so the right child of the root is 5. The left operand is the result of a multiplication, so the left subtree has $*$ as its root. We continue recursively in this way until the entire tree is constructed.



b) We can read off the answer from the picture we have just drawn simply by listing the vertices of the tree in preorder: First list the root, then the left subtree in preorder, then the right subtree in preorder. Therefore the answer is $-*\uparrow+x\,2\,3-y+3\,x\,5$.

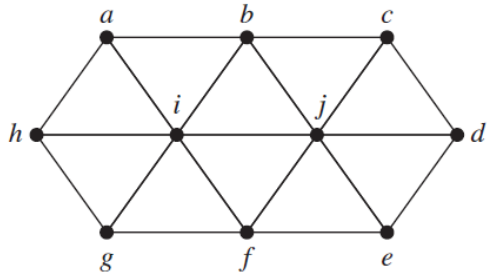
c) We can read off the answer from the picture we have just drawn simply by listing the vertices of the tree in postorder: $x\,2+3\uparrow y\,3\,x+-*5-$.

d) The infix expression is just the given expression, fully parenthesized: $((((x+2)\uparrow 3)*(y-(3+x)))-5)$. This corresponds to traversing the tree in inorder, putting in a left parenthesis whenever we go down to a left child and putting in a right parenthesis whenever we come up from a right child.

Sec. 11.4 4, 14, 16(14), 29

In Exercises 2–6 find a spanning tree for the graph shown by removing edges in simple circuits.

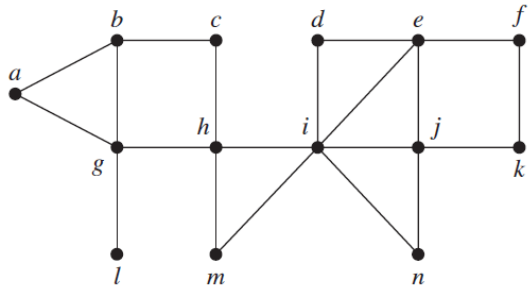
4.



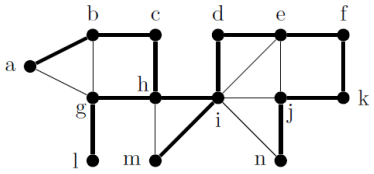
4. We can remove these edges to produce a spanning tree (see comments for Exercise 2): $\{a,i\}$, $\{b,i\}$, $\{b,j\}$, $\{c,d\}$, $\{c,j\}$, $\{d,e\}$, $\{e,j\}$, $\{f,i\}$, $\{f,j\}$, and $\{g,i\}$.

In Exercises 13–15 use depth-first search to produce a spanning tree for the given simple graph. Choose a as the root of this spanning tree and assume that the vertices are ordered alphabetically.

14.

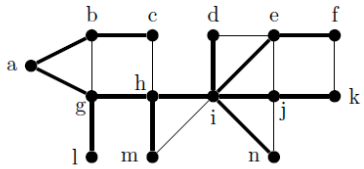


14. The tree is shown in heavy lines. It is produced by starting at a and continuing as far as possible without backtracking, choosing the first unused vertex (in alphabetical order) at each point. When the path reaches vertex l, we need to backtrack. Backtracking to h, we can then form the path all the way to n without further backtracking. Finally we backtrack to vertex i to pick up vertex m.



16. Use breadth-first search to produce a spanning tree for each of the simple graphs in Exercises 13–15. Choose a as the root of each spanning tree.

The tree for the graph in Exercise 14 is shown in heavy lines. It is produced by the same fanning-out procedure as described above.



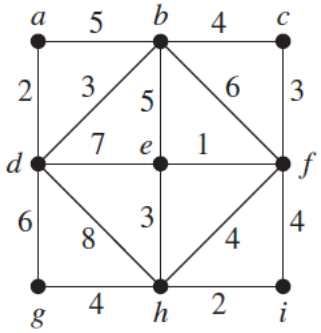
29. Explain how backtracking can be used to find a Hamilton path or circuit in a graph.

Key for 29: Start at a vertex and proceed along a path without repeating vertices as long as possible, allowing the return to the start after all vertices have been visited. When it is impossible to continue along a path, backtrack and try another extension of the current path.

Sec. 11.5 3,7,12

In Exercises 2–4 use Prim's algorithm to find a minimum spanning tree for the given weighted graph.

3.



3. $\{e, f\}$, $\{c, f\}$, $\{e, h\}$, $\{h, i\}$, $\{b, c\}$, $\{b, d\}$, $\{a, d\}$, $\{g, h\}$

7. Use Kruskal's algorithm to find a minimum spanning tree for the weighted graph in Exercise 3.

7. $\{e,f\}$, $\{a, d\}$, $\{h, i\}$, $\{b, d\}$, $\{c,f\}$, $\{e, h\}$, $\{b, c\}$, $\{g, h\}$

12. Devise an algorithm similar to Kruskal's algorithm for constructing a maximum spanning tree of a connected weighted graph.

12. If we simply replace the word "smallest" with the word "largest" (and replace the word "minimum" in the comment with the word "maximum") in Algorithm 2, then the resulting algorithm will find a maximum spanning tree.