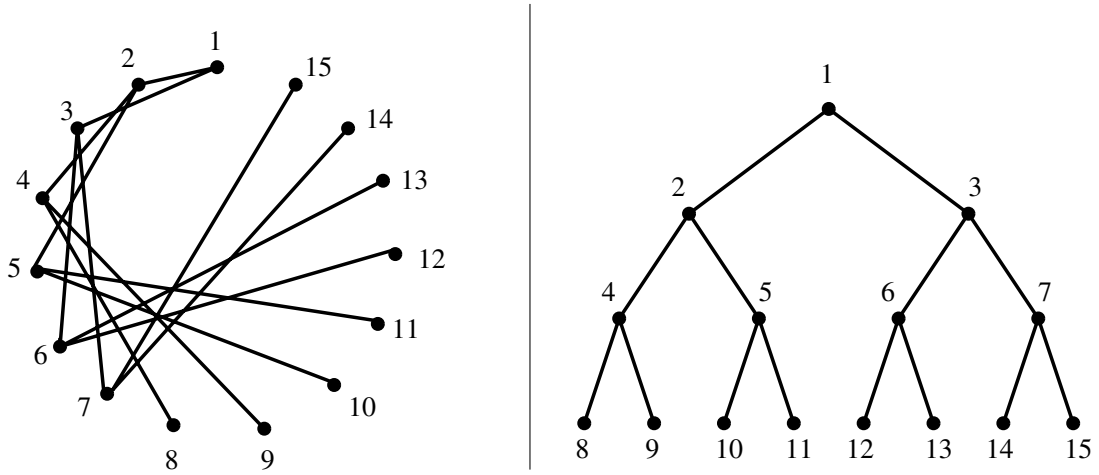

INPUT OUTPUT

15.11 Drawing Trees

Input description: A tree T, which is a graph without any cycles.

Problem description: Create a nice drawing of the tree T.

Discussion: Many applications require drawing pictures of trees. Tree diagrams are commonly used to display and traverse the hierarchical structure of file system directories. My attempts to Google "tree drawing software" revealed special-purpose applications for visualizing family trees, syntax trees (sentence diagrams), and evolutionary phylogenic trees all in the top twenty links.

Different aesthetics follow from each application. That said, the primary issue in tree drawing is establishing whether you are drawing free or rooted trees:

- *Rooted trees* define a hierarchical order, emanating from a single source node identified as the root. Any drawing should reflect this hierarchical structure, as well as any additional application-dependent constraints on the order in which children must appear. For example, family trees are rooted, with sibling nodes typically drawn from left to right in the order of birth.

- *Free trees* do not encode any structure beyond their connection topology. There is no root associated with the minimum spanning tree (MST) of a graph, so a hierarchical drawing will be misleading. Such free trees might well inherit their drawing from that of the full underlying graph, such as the map of the cities whose distances define the MST.

Trees are always planar graphs, and hence can and should be drawn so no two edges cross. Any of the planar drawing algorithms of Section 15.12 (page 520) could be used to do so. However, such algorithms are overkill, because much simpler algorithms can be used to construct planar drawings of trees. The spring-embedding heuristics of Section 15.10 (page 513) work well on free trees, although they may be too slow for certain applications.

The most natural tree-drawing algorithms assume rooted trees. However, they can be used equally well with free trees, after selecting one vertex to serve as the root of the drawing. This faux-root can be selected arbitrarily, or, even better, by using a *center* vertex of the tree. A center vertex minimizes the maximum distance to other vertices. For trees, the center always consists of either one vertex or two adjacent vertices. This tree center can be identified in linear time by repeatedly trimming all the leaves until only the center remains.

Your two primary options for drawing rooted trees are *ranked* and *radial* embeddings:

- *Ranked embeddings* – Place the root in the top center of your page, and then partition the page into the root-degree number of top-down strips. Deleting the root creates the root-degree number of subtrees, each of which is assigned to its own strip. Draw each subtree recursively, by placing its new root (the vertex adjacent to the old root) in the center of its strip a fixed distance down from the top, with a line from old root to new root. The output figure above is a nicely ranked embedding of a balanced binary tree.

  Such ranked embeddings are particularly effective for rooted trees used to represent a hierarchy—be it a family tree, data structure, or corporate ladder. The top-down distance illustrates how far each node is from the root. Unfortunately, such repeated subdivision eventually produces very narrow strips, until most of the vertices are crammed into a small region of the page. Try to adjust the width of each strip to reflect the total number of nodes it will contain, and don't be afraid of expanding into neighboring region's turf once their shorter subtrees have been completed.

- *Radial embeddings* – Free trees are better drawn using a radial embedding, where the root/center of the tree is placed in the center of the drawing. The space around this center vertex is divided into angular sectors for each subtree. Although the same problem of cramping will eventually occur, radial embeddings make better use of space than ranked embeddings and appear considerably more natural for free trees. The rankings of vertices in terms of distance from the center is illustrated by the concentric circles of vertices.

Implementations: GraphViz (*http://www.graphviz.org*) is a popular and well-supported graph-drawing program developed by Stephen North of Bell Laboratories. It represents edges as splines and can construct useful drawings of quite large

and complicated graphs. It has sufficed for all of my professional graph drawing needs over the years.

Graph/tree drawing is a problem where very good commercial products exist, including those from Tom Sawyer Software (*www.tomsawyer.com*), yFiles (*www.yworks.com*), and iLOG's JViews (*www.ilog.com/products/jviews/*). All of these have free trial or noncommercial use downloads.

Combinatorica [PS03] provides Mathematica implementations of several tree drawing algorithms, including radial and rooted embeddings. See Section 19.1.9 (page 661) for further information on Combinatorica.

Notes: All books and surveys on graph drawing include discussions of specific tree-drawing algorithms. The forthcoming *Handbook of Graph Drawing and Visualization* [Tam08] promises to be the most comprehensive review of the field. Two excellent books on graph drawing algorithms are Battista, et al. [BETT99] and Kaufmann and Wagner [KW01]. A third book by Jünger and Mutzel [JM03] is organized around systems instead of algorithms, but provides technical details about the drawing methods each system employs.

Heuristics for tree layout have been studied by several researchers [RT81, Moe90], with Buchheim, et al. [BJL06] reflective of the state-of-the-art. Under certain aesthetic criteria, the problem is NP-complete [SR83].

Certain tree layout algorithms arise from non-drawing applications. The Van Emde Boas layout of a binary tree offers better external memory performance than conventional binary search, at a cost of greater complexity. See the survey of Arge, et al. [ABF05] for more on this and other cache-oblivious data structures.

Related Problems: Drawing graphs (see page 513), planar drawings (see page 520).