3. P. Gillett, *Calculus and Analytic Geometry*, Heath, Lexington, MA, 1984, pp. 194–195.
4. D. Hughes-Hallett et al., *Calculus*, Wiley, New York, 1994.
5. G. Strang, *Calculus*, Wellesley Cambridge Press, Wellesley, MA, 1991, p. 164.

———o———

The "Join the Club" Interpretation of Some Graph Algorithms

Harold Reiter and Isaac Sonin, University of North Carolina-Charlotte, Charlotte, NC 28223

An important part of graph theory is concerned with algorithms. Finding shortest paths, constructing spanning trees with desirable properties, matching and coloring vertices are just a few examples of problems whose solutions are algorithms. Hence, nearly every course in discrete mathematics, combinatorics, or graph theory contains some material on graph algorithms.

Some students find algorithms hard to remember. Though usually based on relatively simple ideas, their formal presentation may be lengthy, and understanding them may require substantial mathematical maturity. Another difficulty is that algorithms are often described in pseudocode—fine for constructing computer programs, but hard for students to internalize. What is needed is something akin to a mnemonic device to help students remember and understand these algorithms. Our approach is to present certain algorithms of graph theory using a context highly familiar to most students: joining a club.

Let $G = (V, E)$ be a graph with vertex set V and edge set E. We assume throughout this paper that our graph is connected; otherwise, we would simply consider each component separately. Recall that a graph H is called a *subgraph* of G if it is obtained by selecting some vertices of G and some edges of G joining vertices of H. A *tree* is a connected graph that contains no cycles. A *spanning tree* of a graph G is a tree that is a subgraph of G containing all the vertices of G. Each of our algorithms results in the construction of a spanning tree T of G. Each spanning tree construction starts at a designated vertex, which we call the *root* of the tree.

Our idea is to interpret V as the people living in a community. An edge between two people means that they are acquainted. There is a club in the community which everyone wishes to join, and eventually, one by one, everyone will. A nonmember u can join only when recommended by a *sponsor*: that is, by a member v who is acquainted with u. A club member is *open* to being a sponsor if the member knows a nonmember in the community. Of course the status of a club member may change from open to non-open at some time during the club's growth, but not vice versa. The rules given below specify at each stage who can join the club and who among open members can be their sponsor.

Our basic rule for joining the club produces a large class of rooted spanning subtrees. Imposing additional conditions with the basic rule yields several types of spanning trees—Breadth First Search (BFS), Depth First Search (DFS), Minimum Total (MT), and Shortest Path (SP)—all of which are useful in solving many problems. For example, a BFS spanning tree can be used to determine the length of a shortest path (i.e., one with the fewest edges) from the root to any other vertex. A DFS spanning tree can be used to locate the bridges in a graph and, when no bridges exist, to establish a strongly connected orientation of the graph. (A *bridge* is an edge whose removal disconnects the graph. An *orientation* is an assignment of a direction to each edge; the directed graph obtained is *strongly connected* if it is possible to get from any vertex to any other vertex along directed

edges without violating the direction, as in Figure 3.) For formal discussions of these algorithms and their many applications, see any of the references [1]–[6].

Basic Rule. *There is one founding member, and all nonmembers join the club one at a time until no one else can join. At each stage, a nonmember joins on the recommendation of an open member.*

The Basic Rule yields a sequence of subgraphs. We begin with the arbitrarily selected root (the founder u_0), and at each stage one vertex (the new member) and one edge (from the new member to the sponsor) are added to the subgraph; see Figure 1. The algorithms we will discuss are different prescriptions for choosing successive new members and their sponsors. Whenever two or more new member–sponsor pairs have equal priority for admission to the club, we choose whichever one we please. This convention, for breaking ties arbitrarily, applies to all the algorithms.

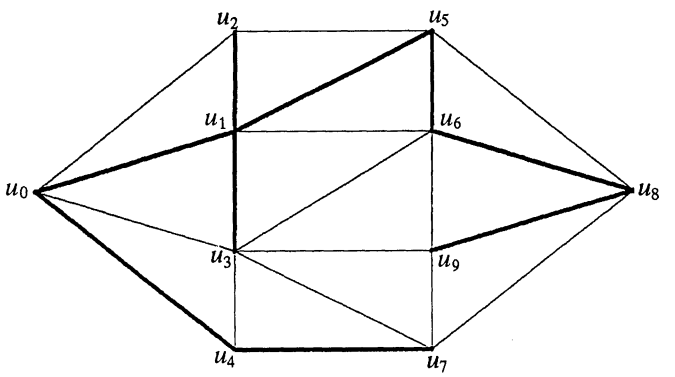

Figure 1. The Basic Rule yields a spanning tree (bold lines). The subscripts indicate the order in which the vertices became members of the club.

Let us show that a subgraph that results from the Basic Rule (or any of its variations) must be a spanning tree. Notice that at each stage the set of members (vertices), together with the edges connecting them with their sponsors, is a connected subgraph of G. What if at some stage a cycle were formed? Let u be the last member of the cycle to join the club. As the newest member, u cannot be a sponsor of either of the adjacent vertices. Hence u must be sponsored by both of them, contrary to the Basic Rule. Therefore, we have an acyclic graph at each stage of the club formation process. Finally, if at any stage there are still nonmembers, then—because G is connected—some member of the club must still be open. Hence the process continues until all nonmembers have joined the club. At this point the club is a tree that includes all the vertices of G; that is, a spanning tree.

Our rules also provide a corresponding *tree labeling*, that is, a labeling of the n vertices with the numbers $0, 1, \ldots, n-1$ so that, on the unique path from the founder to each member, the labels occur in increasing order.

The next two rules depend on the concept of *seniority*. If u joined the club before v then we say that u is *senior* to v (and v is *junior* to u).

Seniority Rule. *At each stage, a nonmember joins the club on the recommendation of the most senior open member.*

A spanning tree that results from the Seniority Rule is called a BFS spanning tree; see Figure 2.

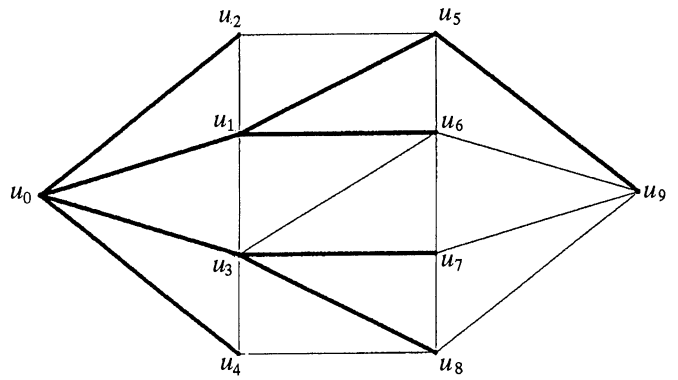

Figure 2. The Seniority Rule yields a Breadth First Search spanning tree (bold lines).

Juniority Rule. *At each stage, a nonmember joins the club on the recommendation of the most junior open member.*

A resulting spanning tree is called a DFS spanning tree. Figure 3 shows a DFS spanning tree with the additional structure of a strongly connected orientation. Each tree edge is directed from senior member to junior member. The other edges are directed from junior member to senior member. Any graph without bridges has a strongly connected orientation obtained by first constructing a DFS spanning tree (Juniority Rule) and then orienting it in this way.

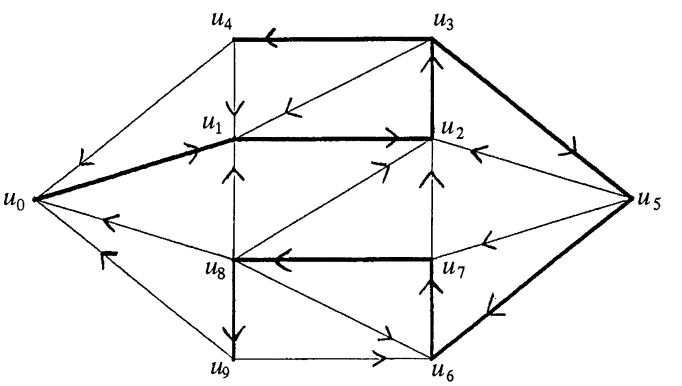

Figure 3. The Juniority Rule yields a Depth First Search spanning tree (bold lines).

Some algorithms apply not only to connected graphs but also to *networks*, that is, graphs with a nonnegative *weight* (length) $f(e)$ assigned to each edge e. The shortest path from vertex u to vertex v means a path the sum of whose weights is

minimal (i.e., no larger than any other such sum). We stipulate for the next rule that each new member must pay an *initiation fee* equal to the weight on the edge between this member and the member's sponsor. Note that a nonmember u who is approached by several open members might owe different fees depending on which sponsor u is selected.

Minimum Fee Rule. *At each stage, a nonmember with the lowest possible initiation fee joins the club, using whichever sponsor makes that fee possible.*

A resulting spanning tree T is called an MT spanning tree (see Figure 4), and its total weight is given by $\Sigma_{e \in T} f(e)$. The Minimum Fee Rule is a simple way of describing Prim's algorithm for finding a spanning tree with minimum total weight. A proof that T is indeed an MT spanning tree can be found in [3] or [5].

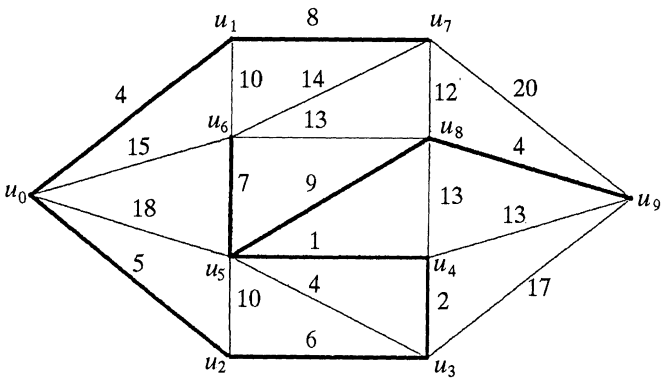

Figure 4. The Minimum Fee Rule yields a spanning tree with minimum total sum of weights (lengths).

Our final rule can be used to produce a spanning tree that describes the shortest paths from the root vertex to any other vertex. To introduce this rule, we require a new member to pay a *reimbursement* as follows. Every new member has a sponsor who in turn has a sponsor, and so on, up to the founder. Suppose each new member u must pay a reimbursement to the club equal to the sum of the fees on all these edges; that is, the sum $\Sigma f(e)$ is taken along the unique list of edges joining the founder to u. As before, if a nonmember has several potential sponsors, the corresponding reimbursements might be different.

Minimum Reimbursement Rule. *At each stage, a nonmember with the lowest possible reimbursement joins the club, using whichever sponsor makes that reimbursement possible.*

The Minimum Reimbursement Rule is our way of describing the rule used in Dijkstra's algorithm to find an SP spanning tree. See Figure 5.

Our algorithms apply to directed graphs as well, on the condition that, if (u, v) is a directed edge, u can sponsor v, but not vice versa. Other graph algorithms also fit the "join the club" paradigm. For example, both the Ford-Fulkerson max-flow

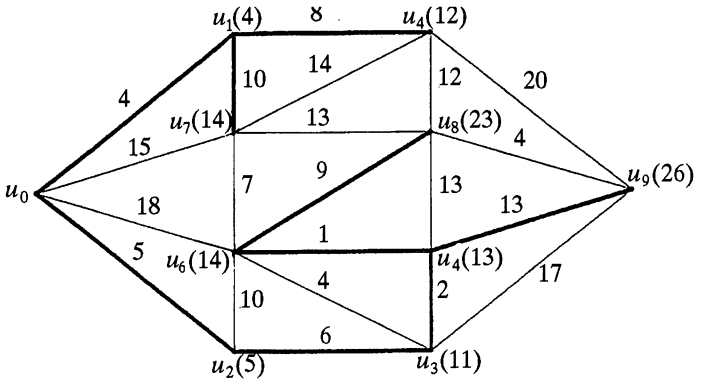

Figure 5. The Minimum Reimbursement Rule yields a spanning tree with shortest paths between u_0 and any point in the graph. Reimbursements are in parentheses.

min-cut algorithm and the "maximum matching" algorithm construct and use spanning trees as part of an iterated process. See [3] or [5]. Having a simple way to remember the algorithms for finding spanning trees with desired properties makes these more complex algorithms easier to grasp. We hope our readers will be happy to join the club of those who know and can apply the algorithms of graph theory.

References

1. K. P. Bogart, *Introductory Combinatorics*, 2nd ed., Harcourt Brace Jovanovich, New York, 1990.
2. G. Chartand and L. Lesniak, *Graphs and Digraphs*, 2nd ed., Wadsworth & Brooks/Cole, Monterey, CA, 1986.
3. A. Gibbons, *Algorithmic Graph Theory*, Cambridge University Press, Cambridge, 1985.
4. F. Harary, *Graph Theory*, Addison-Wesley, Reading, MA, 1969.
5. J. A. McHugh, *Algorithmic Graph Theory*, Prentice Hall, Englewood Cliffs, NJ, 1990.
6. F. S. Roberts, *Applied Combinatorics*, Prentice Hall, Englewood Cliffs, NJ, 1984.

First Things First

One who is naive with respect to the question of possible differences in consequences resulting from application of operators in different orders may profit by thinking a little about the operations of (i) insuring an automobile and (ii) driving the automobile into collision with that of a struggling lawyer.

Ralph Palmer Agnew, *Differential Equations*, 2nd ed., McGraw-Hill, New York, 1960, p. 218.
Contributed by D. Bushaw, Washington State University.