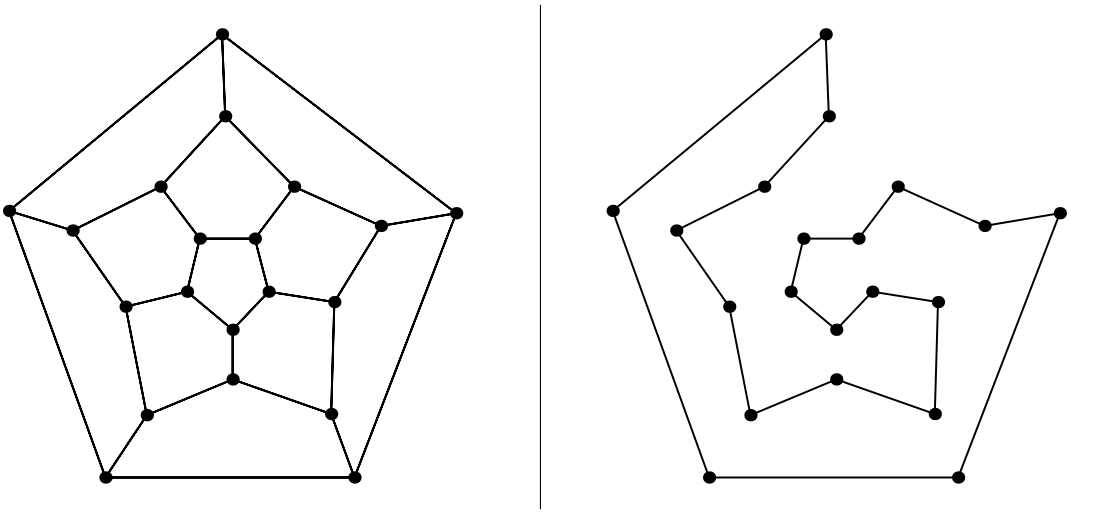

INPUT OUTPUT

16.5 Hamiltonian Cycle

Input description: A graph $G = (V, E)$.

Problem description: Find a tour of the vertices using only edges from G, such that each vertex is visited exactly once.

Discussion: Finding a Hamiltonian cycle or path in a graph G is a special case of the traveling salesman problem G'—one where each edge in G has distance 1 in $'G$. Non-edge vertex pairs are separated by a greater distance, say 2. Such a weighted graph has TSP tour of cost n in G' iff G is Hamiltonian.

Closely related is the problem of finding the longest path or cycle in a graph. This arises often in pattern recognition problems. Let the vertices in the graph correspond to possible symbols, with edges linking pairs of symbols that might occur next to each other. The longest path through this graph is a good candidate for the proper interpretation.

The problems of finding longest cycles and paths are both NP-complete, even on very restrictive classes of unweighted graphs. There are several possible lines of attack, however:

- *Is there a serious penalty for visiting vertices more than once?* – Reformulating the Hamiltonian cycle problem instead of minimizing the total number of vertices visited on a complete tour turns it into an optimization problem.

This allows possibilities for heuristics and approximation algorithms. Finding a spanning tree of the graph and doing a depth-first search, as discussed in Section 16.4 (page 533), yields a tour with at most $2n$ vertices. Using randomization or simulated annealing might bring the size of this down considerably.

- *Am I seeking the longest path in a directed acyclic graph (DAG)?* – The problem of finding the longest path in a DAG can be solved in linear time using dynamic programming. Conveniently, the algorithm for finding the *shortest* path in a DAG (presented in Section 15.4 (page 489)) does the job if we replace min with max. DAGs are the most interesting case of longest path for which efficient algorithms exist.

- *Is my graph dense?* – Sufficiently dense graphs always contain Hamiltonian cycles. Further, the cycles implied by such sufficiency conditions can be efficiently constructed. In particular, any graph where all vertices have degree $\geq n/2$ must be Hamiltonian. Stronger sufficient conditions also hold; see the Notes section.

- *Are you visiting all the vertices or all the edges?* – Verify that you really have a vertex-tour problem and not an edge-tour problem. With a little cleverness it is sometimes possible to reformulate a Hamiltonian cycle problem in terms of Eulerian cycles, which instead visit every edge of a graph. Perhaps the most famous such instance is the problem of constructing de Bruijn sequences, discussed in Section 15.7 (page 502). The benefit is that fast algorithms exist for Eulerian cycles and many related variants, while Hamiltonian cycle is NP-complete.

If you *really* must know whether your graph is Hamiltonian, backtracking with pruning is your only possible solution. Certainly check whether your graph is biconnected (see Section 15.8 (page 505)). If not, this means that the graph has an articulation vertex whose deletion will disconnect the graph and so cannot be Hamiltonian.

Implementations: The construction described above (weight 1 for an edge and 2 for a non-edge) reduces Hamiltonian cycles to a symmetric TSP problem that obeys the triangle inequality. Thus we refer the reader to the TSP solvers discussed in Section 16.4 (page 533). Foremost among them is `Concorde`, a program for the symmetric traveling salesman problem and related network optimization problems, written in ANSI C. `Concorde` is available for academic research use from *http://www.tsp.gatech.edu/concorde*. It is the clear choice among available TSP codes.

An effective program for solving Hamiltonian cycle problems resulted from the masters thesis of Vandegriend [Van98]. Both the code and the thesis are available from *http://web.cs.ualberta.ca/∼joe/Theses/vandegriend.html*.

Lodi and Punnen [LP07] put together an excellent survey of available TSP software, including the special case of Hamiltonian cycle. Current links to all programs are maintained at *http://www.or.deis.unibo.it/research_pages/tspsoft.html*.

The football program of the Stanford GraphBase (see Section 19.1.8 (page 660)) uses a stratified greedy algorithm to solve the asymmetric longest-path problem. The goal is to derive a chain of football scores to establish the superiority of one football team over another. After all, if Virginia beat Illinois by 30 points, and Illinois beat Stony Brook by 14 points, then by transitivity Virginia would beat Stony Brook by 44 points if they played, right? We seek the longest simple path in a graph where the weight of edge (x, y) denotes the number of points by which x beat y.

Nijenhuis and Wilf [NW78] provide an efficient routine to enumerate all Hamiltonian cycles of a graph by backtracking. See Section 19.1.10 (page 661). Algorithm 595 [Mar83] of the *Collected Algorithms of the ACM* is a similar Fortran code that can be used as either an exact procedure or a heuristic by controlling the amount of backtracking. See Section 19.1.5 (page 659).

Notes: Hamiltonian cycles apparently first arose in Euler's study of the knight's tour problem, although they were popularized by Hamilton's "Around the World" game in 1839. See [ABCC07, GP07, LLKS85] for comprehensive references on the traveling salesman problem, including discussions on Hamiltonian cycle.

Most good texts in graph theory review sufficiency conditions for graphs to be Hamiltonian. My favorite is West [Wes00].

Techniques for solving optimization problems in the laboratory using biological processes have attracted considerable attention. In the original application of these "biocomputing" techniques, Adleman [Adl94] solved a seven-vertex instance of the directed Hamiltonian path problem. Unfortunately, this approach requires an exponential number of molecules, and Avogadro's number implies that such experiments are inconceivable for graphs beyond $n \approx 70$.

Related Problems: Eulerian cycle (see page 502), traveling salesman (see page 533).