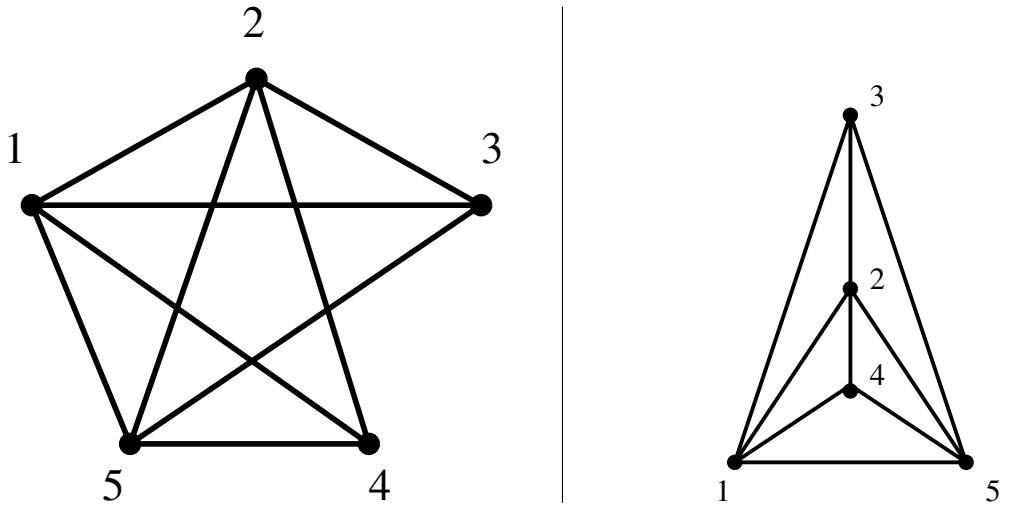

INPUT OUTPUT

15.12 Planarity Detection and Embedding

Input description: A graph G.

Problem description: Can G be drawn in the plane such that no two edges cross? If so, produce such a drawing.

Discussion: Planar drawings (or *embeddings*) make clear the structure of a given graph by eliminating crossing edges, which can be confused as additional vertices. Graphs defined by road networks, printed circuit board layouts, and the like are inherently planar because they are completely defined by surface structures.

Planar graphs have a variety of nice properties that can be exploited to yield faster algorithms for many problems. The most important fact to know is that every planar graph is *sparse*. Euler's formula shows that $|E| \leq 3|V| - 6$ for every nontrivial planar graph $G = (V, E)$. This means that every planar graph contains a linear number of edges, and further that every planar graph contains a vertex of degree ≤ 5. Every subgraph of a planar graph is planar, so there must always a sequence of low-degree vertices to delete from G, reducing it to the empty graph.

To gain a better appreciation of the subtleties of planar drawings, I encourage the reader to construct a planar (noncrossing) embedding for the graph $K_5 - e$, shown on the input figure above. Then try to construct such an embedding where all the edges are straight. Finally, add the missing edge to the graph and try to do the same for K_5 itself.

The study of planarity has motivated much of the development of graph theory. It must be confessed, however, that the need for planarity testing arises relatively infrequently in applications. Most graph-drawing systems do not explicitly seek planar embeddings. *"Planarity Detection"* proved to be among the least frequently hit pages of the Algorithm Repository (*http://www.cs.sunysb.edu/∼algorith*) [Ski99]. That said, it is still very useful to know how to deal with planar graphs when you encounter them.

Thus, it pays to distinguish the problem of planarity testing (does a graph have a planar drawing?) from constructing planar embeddings (actually finding the drawing), although both can be done in linear time. Many efficient planar graph algorithms do not make any use of the drawing, but instead exploit the low-degree deletion sequence described above.

Algorithms for planarity testing begin by embedding an arbitrary cycle from the graph in the plane and then considering additional paths in G, connecting vertices on this cycle. Whenever two such paths cross, one must be drawn outside the cycle and one inside. When three such paths mutually cross, there is no way to resolve the problem, so the graph cannot be planar. Linear-time algorithms for planarity detection are based on depth-first search, but they are subtle and complicated enough that you are wise to seek an existing implementation.

Such path-crossing algorithms can be used to construct a planar embedding by inserting the paths into the drawing one by one. Unfortunately, because they work in an incremental manner, nothing prevents them from inserting many vertices and edges into a relatively small area of the drawing. Such cramping is a major problem, for it leads to ugly drawings that are hard to understand. Better algorithms have been devised that construct *planar-grid embeddings*, where each vertex lies on a $(2n - 4) \times (n - 2)$ grid. Thus, no region can get too cramped and no edge can get too long. Still, the resulting drawings tend not to look as natural as one might hope.

For nonplanar graphs, what is often sought is a drawing that minimizes the number of crossings. Unfortunately, computing the crossing number of a graph is NP-complete. A useful heuristic extracts a large planar subgraph of G, embeds this subgraph, and then inserts the remaining edges one by one to minimize the number of crossings. This won't do much for dense graphs, which are doomed to have a large number of crossings, but it will work well for graphs that are almost planar, such as road networks with overpasses or printed circuit boards with multiple layers. Large planar subgraphs can be found by modifying planarity-testing algorithms to delete troublemaking edges when encountered.

Implementations: LEDA (see Section 19.1.1 (page 658)) includes linear-time algorithms for both planarity testing and constructing straight-line planar-grid embeddings. Their planarity tester returns an obstructing Kuratowski subgraph (see notes) for any graph deemed nonplanar, yielding concrete proof of its nonplanarity.

JGraphEd (*http://www.jharris.ca/JGraphEd/*) is a Java graph-drawing framework that includes several planarity testing/embedding algorithms, including both the Booth-Lueker PQ-tree algorithm and the modern straight-line grid embedding.

PIGALE (*http://pigale.sourceforge.net/*) is a C++ graph editor/algorithm library focusing on planar graphs. It contains a variety of algorithms for constructing planar drawings as well as efficient algorithms to test planarity and identify an obstructing subgraph ($K_{3,3}$ or K_5), if one exists.

Greedy randomized adaptive search (GRASP) heuristics for finding the largest planar subgraph have been implemented by Ribeiro and Resende [RR99] as Algorithm 797 of the *Collected Algorithms of the ACM* (see Section 19.1.6 (page 659)). These Fortran codes are also available from *http://www.research.att.com/∼mgcr/src/*.

Notes: Kuratowski [Kur30] gave the first characterization of planar graphs, namely that they do not contain a subgraph homeomorphic to $K_{3,3}$ or K_5. Thus, if you are still working on the exercise to embed K_5, now is an appropriate time to give it up. Fary's theorem [FÁ48] states that every planar graph can be drawn in such a way that each edge is straight.

Hopcroft and Tarjan [HT74] gave the first linear-time algorithm for drawing graphs. Booth and Lueker [BL76] developed an alternate planarity-testing algorithm based on PQ-trees. Simplified planarity-testing algorithms include [BCPB04, MM96, SH99]. Efficient $2n \times n$ planar grid embeddings were first developed by [dFPP90]. The book by Nishizeki and Rahman [NR04] provide a good overview of the spectrum of planar drawing algorithms.

Outerplanar graphs are those that can be drawn so all vertices lie on the outer face of the drawing. Such graphs can be characterized as having no subgraph homeomorphic to $K_{2,3}$ and can be recognized and embedded in linear time.

Related Problems: Graph partition (see page 541), drawing trees (see page 517).