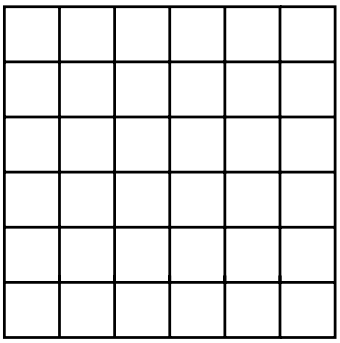

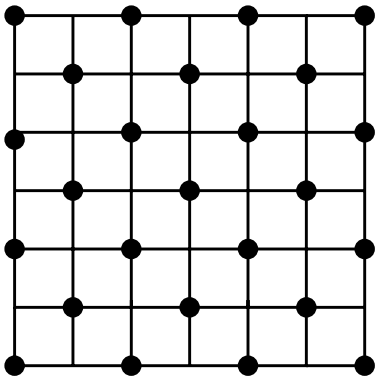

INPUT OUTPUT

16.2 Independent Set

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

Problem description: What is the largest subset S of vertices of V such that for each edge $(x, y) \in E$, either $x \notin E$ or $y \notin E$?

Discussion: The need to find large independent sets arises in facility dispersion problems, where we seek a set of mutually separated locations. It is important that no two locations of our new "McAlgorithm" franchise service be placed close enough to compete with each other. We can construct a graph where the vertices are the set of possible locations, and then add edges between any two locations deemed close enough to interfere. The maximum independent set gives the largest number of franchises we can sell without cannibalizing sales.

Independent sets (also known as *stable sets*) avoid conflicts between elements, and hence arise often in coding theory and scheduling problems. Define a graph whose vertices represent the set of possible code words, and add edges between any two code words sufficiently similar to be confused due to noise. The maximum independent set of this graph defines the highest capacity code for the given communication channel.

Independent set is closely related to two other NP-complete problems:

- *Clique* – Watch what you say, for a clique is what you get if you give an independent set a complement. The *complement* of $G = (V, E)$ is a graph $G' = (V, E')$ where $(i, j) \in E'$ iff (i, j) is not in E. In other words, we replace each edge by a non-edge and vice versa. The maximum independent set in G is exactly the maximum clique in G', so the two problems are algorithmically

identical. Thus, the algorithms and implementations in Section 16.1 (page 525) can easily be used for independent set.

- *Vertex coloring* – The vertex coloring of a graph $G = (V, E)$ is a partition of V into a small number of sets (colors), where no two vertices of the same color can have an edge between them. Each color class defines an independent set. Many scheduling applications of independent set are really coloring problems, since all tasks eventually must be completed.

  Indeed, one heuristic to find a large independent set is to use any vertex coloring algorithm/heuristic, and take the largest color class. One consequence of this observation is that all graphs with small chromatic numbers (such as planar and bipartite graphs) have large independent sets.

The simplest reasonable heuristic is to find the lowest-degree vertex, add it to the independent set, and then delete it and all vertices adjacent to it. Repeating this process until the graph is empty gives a *maximal* independent set, in that it can't be made larger by just adding vertices. Using randomization or perhaps some degree of exhaustive search might result in somewhat larger independent sets.

The independent set problem is in some sense dual to the graph-matching problem. The former asks for a large set of vertices with no edge in common, while the latter asks for a large set of edges with no vertex in common. This suggests trying to rephrase your problem as an efficiently-computable matching problem instead of maximum independent set problem, which is NP-complete.

The maximum independent set of a tree can be found in linear time by (1) stripping off the leaf nodes, (2) adding them to the independent set, (3) deleting all adjacent nodes, and then (4) repeating from the first step on the resulting trees until it is empty.

Implementations: Any program for computing the maximum clique in a graph can find maximum independent sets by just complementing the input graph. Therefore, we refer the reader to the clique-finding programs of Section 16.1 (page 525).

GOBLIN (*http://www.math.uni-augsburg.de/~fremuth/goblin.html*) implements a branch-and-bound algorithm for finding independent sets (called stable sets in the manual).

Greedy randomized adaptive search (GRASP) heuristics for independent set set have been implemented by Resende, et al. [RFS98] as Algorithm 787 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: The proof that independent set is NP-complete is due to Karp [Kar72]. It remains NP-complete for planar cubic graphs [GJ79]. Independent set can be solved efficiently for bipartite graphs [Law76]. This is not trivial—indeed the larger of the "part" of a bipartite graph is not necessarily its maximum independent set.

Related Problems: Clique (see page 525), vertex coloring (see page 544), vertex cover (see page 530).