

1. Vazirani, Exercise 2.2

**2.2** Consider the following algorithm for the maximum cut problem, based on the technique of *local search*. Given a partition of  $V$  into sets, the basic step of the algorithm, called *flip*, is that of moving a vertex from one side of the partition to the other. The following algorithm finds a *locally optimal solution* under the flip operation, i.e., a solution which cannot be improved by a single flip.

The algorithm starts with an arbitrary partition of  $V$ . While there is a vertex such that flipping it increases the size of the cut, the algorithm flips such a vertex. (Observe that a vertex qualifies for a flip if it has more neighbors in its own partition than in the other side.) The algorithm terminates when no vertex qualifies for a flip. Show that this algorithm terminates in polynomial time, and achieves an approximation guarantee of  $1/2$ .

2. Vazirani, Exercise 3.1

**3.1** The hardness of the Steiner tree problem lies in determining the optimal subset of Steiner vertices that need to be included in the tree. Show this by proving that if this set is provided, then the optimal Steiner tree can be computed in polynomial time.

**Hint:** Find an MST on the union of this set and the set of required vertices.

3. Vazirani, Exercise 3.3

**3.3** Give an approximation factor preserving reduction from the set cover problem to the following problem, thereby showing that it is unlikely to have a better approximation guarantee than  $O(\log n)$ .

**Problem 3.14 (Directed Steiner tree)**  $G = (V, E)$  is a directed graph with nonnegative edge costs. The vertex set  $V$  is partitioned into two sets, *required* and *Steiner*. One of the required vertices,  $r$ , is special. The problem is to find a minimum cost tree in  $G$  rooted into  $r$  that contains all the required vertices and any subset of the Steiner vertices.

**Hint:** Construct a three layer graph: layer 1 contains a required vertex corresponding to each element, layer 2 contains a Steiner vertex corresponding to each set, and layer 3 contains  $r$ .

4. Vazirani, Exercise 4.1.

**4.1** Show that Algorithm 4.3 can be used as a subroutine for finding a  $k$ -cut within a factor of  $2 - 2/k$  of the minimum  $k$ -cut. How many subroutine calls are needed?