1. Show that the minimum $k$-cut problem (Problem 4.2) is polynomial time solvable for fixed $k$.

2. 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?

3. 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 a minimum spanning tree on the union of this set and the set of required vertices.

4. Let $G = (V, E)$ be a graph with nonnegative edge costs. $S$, the senders and $R$, the receivers, are disjoint subsets of $V$. The problem is to find a minimum cost subgraph of $G$ that has a path connecting each receiver to a sender (any sender suffices). Partition the instances into two cases: $S \cup R = V$ and $S \cup R \neq V$. Show that these two cases are in $P$ and $NP$-hard, respectively.

   **Hint:** Add a new vertex that is connected to each sender by a zero cost edge. Consider the new vertex and all receivers as required and the remaining vertices as Steiner, and find a minimum Steiner tree.

5. Given an undirected graph $G = (V, E)$, the cardinality maximum cut asks for a partition of $V$ into sets $S$ and $\bar{S}$ so that the number of edges running between these sets is maximized. Consider the following greedy algorithm for this problem. Here $v_1$ and $v_2$ are arbitrary vertices in $G$, and for $A \subset V$, $d(v, A)$ denotes the number of edges running between vertex $v$ and set $A$.

   1. initialize
      
      $A \leftarrow \{v_1\}$
      
      $B \leftarrow \{v_2\}$

   2. for $v \in V \setminus \{v_1, v_2\}$ do
      
      if $d(v, A) \geq d(v, B)$ then $B \leftarrow B \cup \{v\}$
      
      else $A \leftarrow A \cup \{v\}$

   3. output $A$ and $B$

   Show that this is a factor $1/2$ approximation algorithm and give a tight example. What is the upper bound on OPT that you are using? Give examples of graphs for which this upper bound is as bad as twice OPT. Generalize this problem and the algorithm to weighted graphs.
6. Consider the following algorithm for the maximum cut problem, based on the technique of \textit{local search}. Given a partition of $V$ into sets, the basic step of the algorithm, called \textit{flip}, is that of moving a vertex from one side of the partitions to the other. The following algorithm finds a \textit{locally optimal solution} under the flip operation, i.e., a solution that 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 the algorithm terminates in polynomial time, and achieves an approximation guarantee of $1/2$.

7. (Bonus; this is a lot of work!)

Implement Algorithm 4.7 for the minimum $k$-cut problem. Look at exercises 4.3, 4.4, 4.5, and 4.6 to see how to implement the Gomory-Hu algorithm.

8. (Bonus for those students who took the class “Probabilistic Analysis and Randomized Algorithms”)

Find a simple Monte Carlo algorithm for the $k$-cut problem for fixed $k$. 

Page 2 of 2