Algorithms - Exercise 6

Due Wednesday 8/12 24:00

1. Let $G = (V, E)$ be an unconnected graph. A cut in the graph is a partition of the vertices in the graph into two disjoint sets A, B such that $A \cup B = V$. The size of the cut, denoted by $s(A, B)$ is the number of edges (from E) that cross between the two sets. More formally:

$$s(A, B) = |\{\{u, v\} \in E : u \in A \text{ and } v \in B\}|$$

The problem MAX-CUT is to find, given a graph G, the largest cut in the graph. I.e. to find $\max\{s(A, B) : A, B \text{ is a cut in } G\}$.

   (a) Consider the following probabilistic process for MAX-CUT: Start with two empty sets A and B. And then For every vertex $v \in V$, with probability $1/2$ put it in A and with probability $1/2$ in B. Do this independently for every vertex.

   Prove that the expected size of the cut you get is $|E|/2$.

   (b) Show how to remove the randomness from the previous algorithm by replacing the (exponentially large) sample space with a polynomial-size sample space that preserves the correctness of your argument. Use this small sample space to show a 2-approximation for MAX-CUT (that runs in polynomial-time).

   (c) Give a different 2-approximation algorithm for MAX-CUT. This time your algorithm should be greedy: order (arbitrarily) your vertices, then at each stage consider the next vertex and do the best thing that you can according to what you did with the previous vertices.

2. We define the problem MIN-HITTING-SET as follows:
Input: m sets $S_1, \ldots, S_m$ where $S_i \subseteq \{1, \ldots, n\}$ for every $1 \leq i \leq m$.
Output: A set H of minimum size that intersects each S_i. I.e. for every $1 \leq i \leq m$ $H \cap S_i \neq \emptyset$.

Note that the problem is a generalization of the VERTEX-COVER problem.

Show a polynomial-time algorithm that gives M-approximation for MIN-HITTING-SET, where M is the size of the maximal set from $S_1, \ldots, S_m$.

Hint: First construct the 0-1 linear programming (LP in short) problem that is equivalent to the problem. Then relax the 0-1 requirement. Solve the LP problem (you can assume that there is a polynomial-time algorithm that does that), and finally round the solution that you get to 0-1.