CSCI3160: Regular Exercise Set 12

Prepared by Yufei Tao

Problem 1 (Reduction from Vertex Cover to Set Cover). Prove: If the set cover problem admits a polynomial time algorithm (for finding an optimal solution), then the vertex cover algorithm admits a polynomial time algorithm (for finding an optimal solution).

Remark. This means that if the vertex cover cannot be solved in polynomial time, neither can set cover.

Problem 2. Let $\mathcal{C}^*$ be an optimal universe cover for the set cover problem. Consider running the set cover algorithm discussed in the lecture. In particular, consider the moment right before the algorithm is to choose the i-th set S_i, having chosen already $S_1, S_2, ..., S_{i-1}$. Let z_{i-1} be the number of elements in the universe that have not been covered by $S_1 \cup S_2 \cup ... \cup S_{i-1}$. Let $\mathcal{C} = \{S_1, S_2, ..., S_{i-1}\} \cap \mathcal{C}^*$, i.e., $\mathcal{C}$ includes all the sets of $\mathcal{C}^*$ that "happened" to have been selected by our algorithm so far. Prove:

- $\mathcal{C} \neq \mathcal{C}^*$.
- S_i has benefit at least $z_{i-1}/(|\mathcal{C}^* \setminus \mathcal{C}|)$.

Problem 3. Let $\mathcal{I}$ be a set of n intervals in 1D space (i.e., each interval has the form $[x, y]$) and P be a set of n 1D points. A subset $S \subseteq P$ is a *stabbing set of* $\mathcal{I}$ if every interval of $\mathcal{I}$ covers at least one point in S. Let OPT be the size of the smallest stabbing set of $\mathcal{I}$, and you are guaranteed that OPT ≥ 1. Design an algorithm to find a stabbing set of size at most OPT $\cdot\, O(\log n)$. Your algorithm must run in time polynomial to n.

Problem 4*. Let $\mathcal{I}$ be a set of n intervals in 1D space (i.e., each interval has the form $[x, y]$). A set S of 1D points is a *stabbing set of* $\mathcal{I}$ if every interval of $\mathcal{I}$ covers at least one point in S. Let OPT be the size of the smallest stabbing set of $\mathcal{I}$. Design an algorithm to find a stabbing set of size at most OPT $\cdot\, O(\log n)$. Your algorithm must run in time polynomial to n.

Remark. The problem is similar to Problem 3 except that we can form a stabbing set with arbitrary 1D points, rather than using only points from a given set P.