Theory of Computation

Final Examination on January 13, 2022
Fall Semester, 2021

Problem 1 (15 points) Prove that VALIDITY is coNP-complete. You need to describe the reduction and prove that it works. (Recall that language L is in coNP when there is a nondeterministic polynomial-time algorithm M such that (1) if $x \in L$, then $M(x) =$ "yes" for all computation paths, and (2) if $x \notin L$, then $M(x) =$ "no" for some computation path.)

Proof: See p. 481 of the lecture notes. ■

Problem 2 (20 points) Suppose that algorithm C runs in *expected* time $T(n)$ and always gives the right answer. Describe how to turn it into a randomized algorithm which runs *within* time $O(T(n))$ and gives a wrong answer with probability at most, say, 1/3?

Proof: Consider an algorithm which runs C for time $kT(n)$. If C stops before the time bound, output its answer; otherwise, reject the input (alternatively, accept the input). By Markov's inequality, this new algorithm runs in time $kT(n)$ and gives the wrong answer with probability $\leq 1/k$. Pick $k = 3$. ■

Problem 3 (20 points) A randomized algorithm runs in polynomial time $T(n)$ by flipping $O(\log n)$ coins. Furthermore, this algorithm makes an error with probability at most 1/4. How to turn it into a deterministic polynomial-time algorithm which is always correct?

Proof: Try all possible coin clips. There are only polynomially many random sequences. Count the number of "yes" and that of "no". Output the majority answer. This algorithm is always correct because the original algorithm promises the number of paths leading to the majority answer is the correct one. ■

Problem 4 (20 points) The following interactive proof of QR for $x \in Z_n^*$ differs from the one given in the lecture by swapping two statements. Why does it not work as an interactive proof? (Recall that an interactive proof for language L requires, besides those requirements on running times, that (1) (completeness) if $x \in L$, then x is accepted by the verifier with high probability, and (2) (soundness) if $x \notin L$, then x is accepted by the verifier with *any* prover with low probability.)

Algorithm 1

1: **for** $m = 1, 2, \ldots, \log_2 n$ **do**
2: Victor chooses a random bit i and sends it to Peggy;
3: Peggy chooses a random $v \in Z_n^*$ and sends $y = v^2 \bmod n$ to Victor;
4: Peggy sends $z = u^i v \bmod n$, where u is a square root of x;
5: Victor checks if $z^2 \equiv x^i y \bmod n$;
6: **end for**
7: Victor accepts x if Line 5 is confirmed every time;

Proof: Because Victor sends i first before Peggy sends z, a malicious Peggy can pick a random z then solve $z^2 \equiv x^i y \bmod n$ for y. Thus when x is a quadratic nonresidue, Peggy may still convince Victor that x is a quadratic residue with probability one. This violates the soundness condition of IP. ■

Problem 5 (25 points) Let V be a set of points and $d(u, v)$ be the distance between $u, v \in V$. Assume that the distances satisfy the triangle inequality. The k-CENTER problem is to find a subset $S \subseteq V$ with size $k > 0$ that minimizes the maximum distance from any point $v \in V$ to the closest point $u \in S$:

$$S = \operatorname*{arg\,min}_{S \subseteq V, |S| = k} \left(\max_{v \in V} d(v, S) \right),$$

where $d(v, S) = \min_{u \in S} d(v, u)$. Note that $d(v, S) = 0$ if $v \in S$. Prove that the following greedy algorithm is a 0.5-approximation algorithm: It produces a subset S whose maximum distance from any point $v \in V$ to the closest point $u \in S$ is at most 2 times the optimal subset's. (The following definitions may help the analysis: Let $O = \{ o_1, o_2, \ldots, o_k \}$ be an optimal solution with the optimal distance OPT. O partitions V into k clusters $C_i = \{ v \in V \mid d(v, O) = d(v, o_i) \}$ for $i = 1, 2, 3, \ldots, k$. What if C_i contains one member of S or at least 2 members of S?)

Algorithm 2

1: Let S start with $\{ s_1 \}$ for an arbitrary node $s_1 \in V$;
2: **while** $| S | < k$ **do**
3: Find a node $v \in V$ which maximizes $d(v, S)$;
4: Add v to S;
5: **end while**

Proof: Let $O = \{ o_1, o_2, \ldots, o_k \}$ be an optimal solution with the optimal distance OPT. By definition, $d(v, O) \leq \text{OPT}$ for any $v \in V$. O partitions V into k clusters $C_j = \{ v \in V \mid d(v, O) = d(v, o_j) \}$ for $j = 1, 2, 3, \ldots, k$. Note that $\bigcup_j C_j = V$. Let the algorithm add $s_1, s_2, \ldots, s_k$ to S in that sequence. We now proceed to consider the two cases.

(1) Suppose that each cluster contains exactly one point from the algorithm's output S. Fix a cluster C_j. Any point $v \in C_j$ is at most OPT away from o_j by the definition of C_j. Note also that $d(s, o_j) \leq d(s, O) \leq \text{OPT}$ for $s \in S \cap C_j$ by the definition of C_j. The triangle inequality implies that $d(v, s) \leq d(v, o_j) + d(o_j, s) \leq 2 \times \text{OPT}$. As every node belongs in exactly one cluster, the claim is proved.

(2) Otherwise, there are two points $s_m, s_n \in S$, where $1 \leq m < n \leq k$, from S lying in some cluster, say C_j. If $v \in S$, then $d(v, S) = 0 \leq 2 \times \text{OPT}$ and we are done. So assume $v \notin S$ from now on. Define $S_i = \{ s_1, s_2, s_3, \ldots, s_i \}$ for $i = 1, 2, 3, \ldots, k$, the S after the ith iteration of the while loop. Note that S_k is the final output S. Any point $v \in V \setminus S$ has distance $d(v, S) \leq d(s_n, S_{n-1})$; otherwise, s_n would not have been added to S. Note that $d(s_j, S_{j-1}) \leq d(s_j, S_i) \leq d(s_j, s_i)$ for $1 \leq i < j \leq k$ because the algorithm added the point s_j to S which was farthest away from S_{j-1} and adding more points to S does not increase the distance of any point to S. In summary, $d(v, S) \leq d(s_n, S_{n-1}) \leq d(s_n, s_m) \leq d(s_n, o_j) + d(o_j, s_m) \leq 2 \times \text{OPT}$. This completes the proof.

■