Instructor: Rocco A. Servedio

Computer Science 4236: Introduction to Computational Complexity
Problem Set #1 Spring 2023

Due 11:59pm Wednesday, February 8, 2023

See the course Web page for instructions on how to submit homework.
Important: To make life easier for the TAs, **please start each problem on a new page.**

Problem 1 (a) First, write a clear and precise definition of what each of the following mean for functions $f, g : \mathbb{N} \to \mathbb{N}$: (i) $f(n) = O(g(n))$; (ii) $f(n) = \Omega(g(n))$; (iii) $f(n) = \Theta(g(n))$.

Then, for each pair of the following functions (i.e. (a) and (b), (a) and (c), etc.), determine whether (i) $f(n) = O(g(n))$; (ii) $f(n) = \Omega(g(n))$; (iii) $f(n) = \Theta(g(n))$; or (iv) none of the above. Give **brief** justifications of your answers. (Logarithms are base two.)

(a) $\log(n^{n/10})$; (b) $(1 + 1/n)^{10n^2}$; (c) $n/\log n$ if n is even, $n \cdot \log n$ if n is odd;
(d) $n!$; (e) $2^{(\log n)+(\log \log n)}$.

(b) You are working on an algorithm for deciding your favorite language $L \in \Sigma^*$. After some enjoyable effort, you succeed in designing a multitape Turing machine M to decide language L with the following running time: for length-n inputs when $n \leq 100$, your machine runs in time $3 \cdot 2^n$, and for length-n inputs with $n > 100$, your machine runs in time $3 \cdot n^2$. What is the smallest time complexity class $TIME(T(n))$ for which you can say that $L \in TIME(T(n))$? Justify your answer.

(c) Let L be a language that is decided in time $t(n)$ and space $s(n) \geq n$ by a Turing machine M that has an input tape and seven worktapes. Show that L is also accepted in space $s(n)$ by a Turing machine M' that has an input tape and a single worktape. What is the running time of M'? (Use big-Oh notation; justify your answer.)

Problem 2 (a) The language SQUARE-ROOT-COLORING is defined as $\{G : G$ is an undirected graph on n nodes which can be properly vertex-colored using at most $\sqrt{n}$ colors$\}$. (Recall that a proper vertex coloring of a graph is an assignment of a color to each vertex of the graph such that no edge has both of its endpoints being the same color.)

Show that SQUARE-ROOT-COLORING is NP-complete. You may use the fact that GRAPH-COLORING is NP-complete; recall that GRAPH-COLORING is the language of all pairs (G, k) where G is an undirected graph, k is a positive integer, and G can be properly vertex-colored using at most k colors.

(b) The language LOG-CLIQUE is defined as $\{G : G$ is an undirected graph on n nodes which contains a clique of size $\log n$. Do you think that LOG-CLIQUE is NP-complete? Explain why or why not (a few sentences suffice.)
(c) The language QUARTER-CNF-SAT is defined as $\{\phi : \phi$ is a Boolean CNF formula on $4n$ variables such that ϕ has a satisfying assignment with exactly n variables set to TRUE$\}$. Show that QUARTER–CNF-SAT is NP-complete. You may use the fact that CNF-SAT (the language of all satisfiable Boolean CNF formulas) is NP-complete.

Problem 3 In this problem you'll establish that the search and decision versions of some problems in NP are in fact equivalent.

(a) Show that if P=NP then there is a polynomial-time algorithm which, given a Boolean formula ϕ, outputs a satisfying assignment for ϕ (if one exists) or outputs "unsatisfiable" if there is no satisfying assignment.
(b) Recall that a *clique* of size k in an undirected n-node graph $G = (V, E)$ is a subset $V' \subseteq V$ with $|V'| = k$ such that for every $u, v \in V'$ the edge (u, v) is present in E. Show that if P=NP then there is a polynomial-time algorithm which, given an undirected graph G, outputs a clique of size k, where k is the largest value such that G contains a k-clique.

Problem 4 Recall that a *vertex cover* in a graph with vertex set V is a set of nodes $V' \subseteq V$ such that every edge in the graph touches (at least) one of the nodes in the vertex cover (so it is a set of nodes that "cover" all of the edges). The language VERTEX-COVER is $\{(G, k) : G$ has a vertex cover of size at most $k\}$.

Both VERTEX-COVER and CLIQUE are NP-complete, by easy polynomial-time reductions from one to the other (a set V' is a vertex cover in G if and only if $V \setminus V'$ is an independent set in G, and an independent set in G is the same as a clique in the complement of G). This means that the worst-case time complexity of the two problems is equivalent, but there are several other senses in which VERTEX-COVER appears to be considerably easier than CLIQUE. In this problem you'll explore one such sense, namely the running time *as a function of* k.

Suppose that n is very large and k is rather small. Despite intensive research effort, the fastest known algorithms for determining whether an n-node graph G has a k-clique run in time $n^{\Theta(k)}$, and there is reason to believe that no faster algorithms can exist.

In contrast, show that there is a $O(2^k) \cdot \text{poly}(n)$-time algorithm for determining whether G has a size-k vertex cover.

Problem 5 Consider the language FACTORING $= \{(N, k) : N$ and k are numbers in binary notation which are such that N is divisible by some integer in the range $\{2, \ldots, k\}.\}$.

(a) Show that FACTORING is in NP.
(b) (Search is no harder than decision) Show that if P=NP, then there is a poly(n)-time algorithm which, given as input an n-bit integer N, outputs the prime factorization of N. (Hint: Use part (a).)

(c) Now define UNARY-FACTORING to be the analogue of FACTORING but where N, k are given in *unary* notation. Show that UNARY-FACTORING is in P. (The moral here is that binary notation is the "right" way to represent numbers as inputs to algorithms.)

Problem 6 This problem asks you to fill in a missing piece from our proof of Ladner's theorem (recall that Ladner's theorem states that if $P \neq NP$, then there are languages in NP that are neither in P nor NP-complete). Recall that the proof uses the following definition and claim:

Definition: *Let $H : \{2, 3, \dots\} \to \mathbb{N}$ be defined as follows: $H(n)$ is the smallest number $i \leq \log\log n$ such that for every $x \in \{0,1\}^*$ of length $|x| \leq \log n$, machine M_i outputs $SAT_H(x)$ within $i|x|^i$ time steps (and if there is no such number i, then $H(n) = \log\log n$). Here*

$$SAT_H := \bigcup_{n \geq 1} \{\psi 0 1^{n^{H(n)}} \ : \ \psi \in SAT, |\psi| = n\}.$$

Claim: *The function H is well defined and there is an algorithm which, given an input number n, runs in* $\text{poly}(n)$ *time and computes $H(n)$.*

Prove the above claim.