

More Practice Problems for Quiz 3

Borrowed from Prof. Ron Rivest, Matt Lepinski

Please write your name in the upper corner of each page. (2 Points)

Problem 2: (20 points) For any polynomial time computable function, f , let L^f be defined as follows:

$$L^f = \{x | \exists y f(x, y) \in L\}$$

(For example, consider the functions $g(x, y) = x$, and $h(x, y) = y$. Then $L^g = L$ and (assuming L is non-empty) $L^h = \Sigma^*$.) Prove that if $L \in NP$ then $L^f \in NP$ for all polynomial time computable functions, f .

Problem 3: (20 points) Prove that the CLIQUE/ANTI-CLIQUE problem (CAC for short) is NP-Complete:

$$CAC = \{ \langle G, k, m \rangle \mid G \text{ is an undirected graph that contains} \\ \text{either a clique of size } k \text{ or an anti-clique} \\ \text{of size } m \}$$

(Note: An anti-clique of size m is a set of m vertices such that no edge in G connects any two vertices in the anti-clique. An anti-clique is sometimes called an independent set).

Problem 4: (20 points) Paul Porker claims to have a polynomial-time algorithm F such that given any undirected graph G as input, $F(G)$ is the length of the longest cycle in G that contains no repeated vertices.

Paul says, “The reason such an algorithm exists is that most graphs have short cycles which allows this algorithm to be efficient (polynomial time). Also, since the algorithm isn’t solving a decision problem, I don’t have to worry about NP-Completeness. After all, only decision problems can be NP-Complete”.

Argue that Paul Porker’s claim is likely to be incorrect.

Problem 5: (20 points) Disprove the following statement: If $P \neq NP$ then for every polynomial time computable function, f , there exists a polynomial time algorithm A , such that on input $f(y)$ (for some y), A outputs an x such that $f(x) = f(y)$. (Note: we don't care about the behavior of A when it is given an input that is not in the range of f).

Hint: One way to solve this problem is to exhibit a specific function (which is related to an NP-Complete problem) and show that if you could find pre-images under the function, you would be able to solve the related NP-Complete problem. If you solve the problem by exhibiting a specific f to serve as a counter-example, it is very important that you argue why the existence of this function disproves the theorem.

Hint: The function, f , specified in this problem need not be one-to-one. (That is, you can have many inputs which map to the same output. However, the algorithm A succeeds if it finds *any* pre-image of its input, $f(y)$... It is not required to find a specific pre-image).