Problem Set 12 for CS 170

Formatting

Please use the following format for the top of the solution you turn in, with one line per item below (in the order shown below):

<your username on cory.eecs>
<your full name>
`CS170, Spring 2003`
`Homework #12`
`Section` <your section number>
`Partners:` <your list of partners>

(Remember to write your section number, not the name of your TA or the time of your section.) This will make it easier for us to sort and process your homeworks. Thank you!

Note

When asked for an algorithm you must give (1) a brief informal description of the algorithm, (2) a precise description using pseudo-code, (3) an informal argument for termination and correctness of the algorithm, and (4) an analysis of the running time of the algorithm. Be clear about what the input to the algorithm is, how you measure the size of the input, and what constitutes a "step" in your running-time analysis.

Problem 0. [Any questions?] (5 points)

What's the one thing you'd most like to see explained better in lecture or discussion sections? A one-line answer would be appreciated.

(Sometimes we botch the description of some concept, leaving people confused. Sometimes we omit things people would like to hear about. Sometimes the book is very confusing on some point. Here's your chance to tell us what those things were.)

Problem 1. [Warmup] (15 points)

(a) Most researchers would be surprised if $\mathbf{P} = \mathbf{NP}$. Use this to show that it would be surprising if $\mathit{3SAT} \in \mathbf{P}$.

(b) Show that every problem in $\mathbf{NP}$ can be solved in exponential time (i.e., in time $2^{p(n)}$ for some polynomial $p(n)$).

Problem 2. [One-way Functions] (30 points)

Suppose $f : \{0,1\}^* \to \{0,1\}^*$ is an invertible function that preserves lengths (i.e., given a k-bit string, it produces a k-bit string), and with the properties that

(i) f is computable in polynomial time, and

(ii) f^{-1} is not computable in polynomial time.

Define the language

$$L_f = \{(y,z) : f^{-1}(y) < z \text{ and } |y| = |z|\},$$

where the symbol $<$ refers to the lexicographic ordering on strings and $|y|$ denotes the length of the string y in bits.

Professor Blue makes the following argument:

> Define $V((y,z),w)$ to be true if $|y| = |z| = |w|$ and $f(w) = y$ and $w < z$, and false otherwise. The function V can be computed in polynomial time, hence $L_f \in \mathbf{NP}$.
>
> Moreover, if we could compute f^{-1} in polynomial time, then we could test membership in L_f in polynomial time. But f^{-1} can't be computed in polynomial time, hence $L_f \notin \mathbf{P}$.
>
> Consequently, L_f is a language in $\mathbf{NP}$ but not in $\mathbf{P}$, so under these assumptions about f, $\mathbf{P} \neq \mathbf{NP}$.

(a) Are there any errors in Professor Blue's proof? If so, which paragraph(s), and what type(s) of error?

(b) Her last paragraph contains several claims (i.e., "$L_f \in \mathbf{NP}$", "$L_f \notin \mathbf{P}$", "under these assumptions about f, $\mathbf{P} \neq \mathbf{NP}$"). Which of these are correct? Justify your answer.

Professor Gold makes the following argument:

> Define K_f to be the complement of L_f, i.e., $K_f = \{0,1\}^* \setminus L_f$.
>
> Given (y,z), non-deterministically guess a witness w, then check whether $|y| = |z| = |w|$ and $f(w) = y$ and $w < z$; if so, then we can conclude $(y,z) \notin K_f$; otherwise, we can conclude $(y,z) \in K_f$. Hence $K_f \in \mathbf{NP}$.
>
> Moreover, computing f^{-1} reduces to testing membership in K_f. If K_f were in $\mathbf{P}$, then this reduction would mean that f^{-1} could be computed in polynomial time, in contradiction to our assumptions about f. Hence $K_f \notin \mathbf{P}$.
>
> Consequently, K_f is a language in $\mathbf{NP}$ but not in $\mathbf{P}$, so under these assumptions about f, $\mathbf{P} \neq \mathbf{NP}$.

(A) Are there any errors in Professor Gold's proof? If so, which paragraph(s), and what type(s) of error?

(B) His last paragraph contains several claims. Which of these are correct? Justify your answer.

(C) Have we proven that $\mathbf{P} \neq \mathbf{NP}$? Is it time to collect the \$1,000,000 Claymath prize? Why or why not?

Answer 2 of the following 3 problems. If you hand in answers to all three, you will be graded on the first two problems.

(updated 5/8):
If you like, in Problems 3–5 you may freely assume (without need for proof) that the following problems are all NP-complete: CircuitSAT, SAT, 3SAT, Directed Hamiltonian Cycle, Undirected Hamiltonian Cycle, Directed Hamiltonian Path, Undirected Hamiltonian Path, Vertex Cover, Maximum Clique, Partition, and any problem mentioned/proved in the reader or in lecture as NP-complete.

Problem 3. [Combining Vectors] (25 points)

You are given an integer y and a set of vectors $\mathcal{S} = \{V_1, \ldots, V_K\}$, where $V_i \in \mathbb{N}^n$. In other words, each V_i is a vector of n non-negative integers. Let $v(j)$ denote the j^{th} element of vector v. VECTOR-COMBINE is a decision problem that asks whether there exists a set $X \subseteq \mathcal{S}$ so that $|X| < y$ and:

$$\sum_{j=1}^{n} \max_{v \in X} v(j) \geq n.$$

Show that VECTOR-COMBINE is NP-complete.

HINT: Think of 0-1 vectors.

Problem 4. [Spanning Trees] (25 points)

Given a graph $G = (V, E)$ and an integer k, the CONSTRAINED-SPANNING-TREE problem asks whether there exists a spanning tree on G where each node in the tree has degree less than k. Show that CONSTRAINED-SPANNING-TREE is NP-complete.

Problem 5. [Network Routing] (25 points)

Consider an undirected graph G with source vertices $s_1, \ldots, s_k$ and sink vertices $t_1, \ldots, t_k$. The NETWORK-ROUTING decision problem asks whether there are k node disjoint paths, where the i^{th} path goes from s_i to t_i. Show that this problem is NP-complete.

HINTS:

- Reduce from 3SAT.
- For a 3SAT formula with q clauses and n variables, use $k = q + n$ sources and sinks. Introduce one source/sink pair (s_i, t_i) for each variable (x_i), and one source/sink pair (s'_i, t'_i) for each clause.
- For each 3SAT clause, introduce 6 new intermediate vertices, one for each literal occurring in that clause and one for its complement.
- Notice: If the path from s'_i to t'_i goes through some intermediate vertex representing, say, an occurrence of variable x_i, then no other path can go through that vertex. What vertex would you like the other path to be forced to go through instead?