COMP 330 - Fall 2010 - Assignment 6

Solutions

General rules: In solving this you may consult books and you may also consult with each other, but you must each find and write your own solution. In each problem list the people you consulted. This list will not affect your grade. There are in total 115 points, but your grade will be considered out of 100. You should drop your solutions in the assignment drop-off box located in the Trottier Building on the 3rd floor left of the elevators.

1. Recall that for $f, g : \mathbb{N} \to \mathbb{R}^+$, we say $f = O(g)$ if and only if

$$\exists c, n_0 > 0 \forall n > n_0 \qquad f(n) \leq cg(n). \tag{1}$$

In each one of the following cases show that $f = O(g)$ by proving (1).

(a) (5 Points) $f(n) = (n+5)^3$ and $g(n) = n^3$.

Solution: Note that $f(n) = n^3 + 15n^2 + 75n + 125 \leq n^3 + 15n^3 + 75n^3 + 125n^3 = 216n^3$ for $n \geq 1$. Thus for $c = 216$ and $n_0 = 1$ we have $f(n) \leq cg(n)\ \forall n > n_0$ as required.

(b) (10 Points) $f(n) = 10^{\sqrt{n}}$ and $g(n) = 2^n$.

Solution: Note that $10^{\sqrt{n}} \leq 2^n \Leftrightarrow \sqrt{n} \log_2 10 \leq n \Leftrightarrow n \geq (\log_2(10))^2$. Let $c = 1$ and $n_0 = \left\lceil (\log_2(10))^2 \right\rceil = 12$ to obtain $f(n) \leq cg(n)\ \forall n > n_0$ as required.

(c) (10 Points) $f(n) = \log_2 n$ and $g(n) = \sqrt{n}$.

Solution: Define $h(n) = g(n) - f(n)$. Differentiating with respect to n we obtain $h'(n) = \frac{1}{2\sqrt{n}} - \frac{1}{n \ln(2)}$. Setting $h'(n) = 0$ we note that the only critical point is $\frac{4}{\ln(2)^2} \approx 8.325$ and $h'(n) > 0$ for all $n \geq 9$.
Now $h(n) = 0$ for $n = 16$ and $h(n)$ is increasing for all $n \geq 9$ hence letting $c = 1$ and $n_0 = 16$ we obtain $f(n) \leq cg(n)\ \forall n > n_0$ as required.

2. Recall that for $f, g : \mathbb{N} \to \mathbb{R}^+$, we say $f = o(g)$ if and only if

$$\forall \epsilon > 0 \exists n_0 > 0 \forall n > n_0 \qquad f(n) \leq \epsilon g(n). \tag{2}$$

In each one of the following cases show that $f = o(g)$ by proving (2).

(a) (10 Points) $f(n) = 342n^2 + n + 10$ and $g(n) = n^2 \log_2 n$

Solution: Let $\epsilon > 0$ be given. Note that $342n^2 + n + 10 \leq 342n^2 + n^2 + 10n^2 = 353n^2$ for $n \geq 1$. And $353n^2 \leq \epsilon n^2 \log_2 n \Leftrightarrow 353 \leq \epsilon \log_2 n \Leftrightarrow n \geq 2^{\frac{353}{\epsilon}}$. Hence choosing $n_0 = 2^{\frac{353}{\epsilon}}$ gives $f(n) \leq \epsilon g(n)\ \forall n \geq n_0$ as required.

(b) (10 Points) $f(n) = \sqrt{n} + 10$ and $g(n) = \sqrt{n \log_2 n}$.

Solution: Let $\epsilon > 0$ be given. Note that $\sqrt{n} + 10 \leq \sqrt{n} + 10\sqrt{n} = 11\sqrt{n}\ \forall\ n \geq 1$. Next $11\sqrt{n} \leq \epsilon\sqrt{n \log_2 n} \Leftrightarrow 11 \leq \epsilon\sqrt{\log_2 n} \Leftrightarrow n \geq 2^{\frac{121}{\epsilon^2}}$. Hence choosing $n_0 = 2^{\frac{121}{\epsilon^2}}$ gives $f(n) \leq \epsilon g(n)\ \forall n \geq n_0$ as required.

3. Consider the following algorithm:

On input w which is the binary representation of a positive integer:

- For $i = 2, 3, \ldots, w - 1$
- If w is divisible by i, reject.
- If w was not divisible by any of the above values of i, accept.

(a) (5 Points) What is the language of the above Turing Machine?

Solution: The language of this Turing Machine is the set of prime numbers.

(b) (5 Points) Is the running time of this Turing Machine polynomial? (Explain)

Solution: No. Suppose we have an input w consisting of n bits where each bit is a 1. Then the size of w is $2^n - 1$. In the case that w is a prime number the machine would have to perform $O(2^n)$ steps to check if w is divisible by any positive integer smaller than itself. Hence the running time of this machine is exponential in the input size n.

4. (15 Points) A cycle of size m in a graph G is a set of <u>distinct</u> vertices $v_1, \ldots, v_m$ such that v_1 is adjacent to v_2, v_2 is adjacent to v_3,...,v_{m-1} is adjacent to v_m, and v_m is adjacent to v_1. Show that the following language is in P:

$$C_{100} = \{\langle G \rangle : G \text{ is a graph which contains a cycle of size at least } 100\}.$$

Solution: To show that C_{100} is in P we will construct a TM that decides C_{100} in polynomial time.

M = On input $\langle G \rangle$ where $G = (V, E)$ with $|V| = n$:
1. If $n < 100$ reject $\langle G \rangle$.
2. Let $k = n!/(n - 100)!$ and $V_1, \cdots, V_k$ be all the possible permutations of 100 vertices in V.

3. For $i = 1, \cdots k$:

   Check if the vertices given by the permutation V_i form a simple path in G. If they do, mark the vertices in V_i except the very first and last vertex. Then check if there exists another simple path from the first vertex in V_i to the last vertex in V_i that does not use any marked vertices. If there is, accept $\langle G \rangle$ and return.

4. If the loop terminates without accepting $\langle G \rangle$ then reject $\langle G \rangle$.

To compute the running time of this TM note that the for loop is repeated $k = O(n^{100})$ times. Inside each iteration we can check if the vertices of a given permutation V_i form a simple path in a constant number of steps since each permutation constains a constant number of vertices. Next we can check if the there exists a path between the first and last vertex consisting of unmarked vertices by using a BFS. This can be done in $O(n^2)$. Therefore the overall running time of the TM is $O(n^{100})$ which is polynomial.

To see that the TM does indeed decide C_{100} note that if $\langle G \rangle \in C_{100}$ then there is a set of vertices $v_1, \cdots, v_m$ with $m \geq 100$ that form a cycle. Hence when the TM performs the for loop for the permutation $v_1, \cdots, v_{100}$ it will find that these vertices form a simple path. The TM will then mark the vertices $v_2, \cdots, v_{99}$ and will proceed to find the simple path v_{100}, v_{101},$\cdots$,v_m, v_1 which connects v_1 to v_{100} without using any marked vertices. Hence the TM will accept $\langle G \rangle$ and terminate.

Now if $\langle G \rangle \notin C_{100}$ then there is no cycle of length at least 100. Hence even if the TM finds a pemutation of 100 vertices that form a simple path there will be no path joining the first and last vertex using only unmarked vertices. Therefore the TM will reject $\langle G \rangle$.

5. (10 Points) Show that if P=NP, then every language in NP except $\emptyset$ and Σ^* is NP-complete.

Solution: Let B be a language in NP such that $B \neq \emptyset$ and $B \neq \Sigma^*$. Note that this implies the existence of two distinct elements $x \neq y$ such that $x \in B$ and $y \notin B$. To show that B is NP-complete, we must show that:

(a) $B \in NP$ and

(b) every A in NP is polynomial-time reducible to B.

The first condition holds by definition. To check the second condition let A be any language in NP. Since we are assuming that P=NP it follows that $A \in$ P. Hence there exists a deterministic TM M that decides A in polynomial time. Then define the function $f : \Sigma^* \to \Sigma^*$ as

$$f(w) = \begin{cases} x & \text{if} w \in A \\ y & \text{if} w \notin A \end{cases}$$

Then note that f is a polynomial time computable function due the the existence of the TM M.

We then obtain

$$w \in A \Rightarrow f(w) = x \Rightarrow f(w) \in B$$
$$w \notin A \Rightarrow f(w) = y \Rightarrow f(w) \notin B$$

Hence $w \in A \Leftrightarrow f(w) \in B$ and therefore A is polynomial-time reducible to B. It follows that B is NP-complete.

6. (10 Points) Show that if L is in NP, then L^* is also in NP.

Solution: First note that since L is in NP there exists a nondeterministic polynomial time TM N that decideds L. To show that L^* is in NP we will construct a nondeterministic polynomial time TM M that decides L^* as follows:

M = On input w with $|w| = n$: 1. If $w = \emptyset$ accept w.
2. For $i = 1, \cdots n$:
    3. Partition w nondeterministically into i non-empty substring $w_1, \cdots, w_i$.
    4. Run N on each substring $w_1, \cdots, w_i$. 5. If each substring is accepted then accept w.
6. Reject w.

If $w \in L^*$ then we can write $w = w_1 \cdots w_k$ for some $k \leq n$ such that each $w_i \in L$. Hence the TM M will accept w. If instead $w \notin L^*$ then there is no way of partitioning w into substrings which are in L hence the TM M will reject w.

Since N runs in polynomial time it follows that M also runs in polynomial time.

7. Show that the following languages are NP-complete.

   (a) (10 Points)

   $\{\langle \phi \rangle \ : \ \phi$ is a CNF with a solution that sets exactly half of the variables to TRUE$\}$.

   Solution: Let L be the language in question. We will first show that L is in NP. To see this consider the TM that chooses half of the variables of ϕ nondeterministically, sets them to TRUE and sets all other variables to FALSE. The TM then verifies if this assignment satisfies ϕ. If it does it accepts and if it does not it rejects. Also if there is an odd number of variables the TM just rejects.

   We now look at $3SAT$ which is NP-complete and we will show that $3SAT \leq_p L$ which together with $L \in$ NP will imply that L is NP-complete. We do this using the following polynomial time reduction:

M = On input $\langle \phi \rangle$ such that ϕ has has variables $x_1, \cdots x_n$:
1. Define new variables $y_1, \cdots, y_n$ and set $\phi' = \phi$.
2. For each clause in C_i in ϕ:
    3. Define a new clause $C_i' = C_i$.
    4. Negate all the literals in C_i'.
    4. For each variable x_j in C_i' replace it with y_j.
    5. Append clause C_i' to ϕ' .
6. Output $\langle \phi' \rangle$.

If $\langle \phi \rangle \in 3SAT$ then there is some assignment of values TRUE / FALSE to the variables $x_1, \cdots, x_n$ which satisfies ϕ. Wlog we can assume that the variable $x_1, \cdots, x_l$ are set to TRUE in this assignment and the variables $x_{l+1}, \cdots, x_n$ are set to FALSE, for some $l \leq n$. We then set $y_1, \cdots, y_l$ to FALSE and $y_{l+1}, \cdots, y_n$ to TRUE. We claim that this assignment satisfies ϕ'. To see this note that every clause C_i which is a clause of ϕ must have a literal which evaluates to TRUE. It follows that the corresponding clause in C_i' will also evaluate to TRUE. In addition ϕ' has $2n$ variables, of which exactly $l + (n - l) = n$ are set to TRUE. Hence $\langle \phi' \rangle \in L$.

If $\langle \phi \rangle \notin 3SAT$ then ϕ always evaluates to FALSE for any assignment of TRUE / FALSE values to the variables. It follows that ϕ' will also evaluate to FALSE and hence $\langle \phi' \rangle \notin L$.

Therefore $\langle \phi \rangle \in 3SAT \Leftrightarrow \langle \phi' \rangle \in L$.

(b) (15 Points) A d-clique in a graph G is a set of vertices such that every two of them are in distance at most d from each other. So a 1-clique is an actual clique. The language in question is

$$\{\langle G, k \rangle \mid G \text{ contains a 2-clique with } k \text{ vertices}\}.$$

Solution: Let L be the language in question. We will first show that L is in NP. Suppose we have a graph $G = (V, E)$ with $|V| = n$ and a given integer $k \leq n$. Consider the TM that chooses a set S of k vertices nondeterministically. For every pair of vertices $u, v \in S$ that are not joined by an edge the TM checks if there exists a vertex $w \in S$ such that $(u, w) \in E$ and $(v, w) \in E$. If it does not find such a w for a pair u, v it reject. If it cycles through all pairs without rejecting, it accepts. This can be done in polynomial time.

We now look at $CLIQUE$ which is NP-complete and we will show that $CLIQUE \leq_p L$ which together with $L \in$ NP will imply that L is NP-complete. We do this using the following polynomial time reduction:

M = On input $\langle G, k \rangle$ where $G = (V, E)$ with $|V| = n$, $|E| = m$ and $k \leq n$:
1. Construct a new graph H by subdividing each edge in G (as decribed in the hint). Let W be the set of new vertices added when subdividing the edges.
2. Add a separate clique of size $100n^2$ to the graph consisting of vertices $u_1, \cdots, u_{100n^2}$.

3. Join u_1 to every vertex in V.
4. Output $\langle H, k + 100n^2 \rangle$.

If $\langle G, k \rangle \in CLIQUE$ then G has a clique of k vertices $v_1, \cdots, v_k$. In this case the vertices $v_1, \cdots, v_k, u_1, \cdots, u_{100n^2}$ form a 2-clique in H of size $k+100n^2$. Hence $\langle H, k + 100n^2 \rangle \in L$.

If $\langle G, k \rangle \notin CLIQUE$ then the maximum clique in G must be of size strictly smaller than k. Let $v_1, \cdots, v_l$ denote the maximum clique in G with $l < k$. Now note that the maximum 2-clique in H cannot include any vertices from the set W since $|W| < n^2 < 100n^2$ and each vertex in W is at a distance 3 from each of the vertices $u_2, \cdots, u_{100n^2}$. Hence the maximum 2-clique in H is $v_1, \cdots, v_l, u_1, \cdots, u_{100n^2}$ and has size $l + 100n^2 < k + 100n^2$. Therefore $\langle H, 100n^2 + k \rangle \notin L$.

It follows that $\langle G, k \rangle \in CLIQUE \Leftrightarrow \langle H, 100n^2 + k \rangle \in L$.