COS 487
Fall 2003

Problem Set 7 Solutions

Problem 1:

We show that P is closed under the star operation by dynamic programming. Let A be any language in P, and let M be the TM deciding A in polynomial time. The following procedure decides A^*.

ST = "On input $w = w_1w_2\ldots w_n$:

1. If w is the empty string, accept.
2. Initialize $T[i, j] = 0$ for $1 \le i \le j \le n$.
3. For $i = 1$ to n,
4. Set $T[i,i] = 1$ if w_1 is in A.
5. For $l = 2$ to n,
6. For $i = 1$ to $n - l + 1$,
7. Let $j = i + l - 1$,
8. If $w_i\ldots w_j$ is in A, set $T[i,i] = 1$.
9. For $k = i$ to $j - 1$,
10. If $T[i,k] = 1$ and $T[k,j] = 1$, set $T[i,j] = 1$.
11. Accept if $T[1,n] = 1$; otherwise reject."

Each stage takes polynomial time, and ST runs for $O(n^3)$ stages, so the algorithm runs in polynomial time.

Problem 2:

Initialize the permutation p to the identity. Read t from left to right, one bit at a time. If the current bit is 0, assign $p = pp$. Otherwise, let $p = ppq$. Since composition is associative, the end result is p^t. The algorithm runs in $O(klog(t))$ time, which is polynomial.

Problem 3:

(a.) A $\neq$-assignment assigns to each clause at least one true literal and at least one false literal. The negation will thus assign to each clause at least one true literal and at least one false literal.

(b.) Suppose a clause $(y_1 \vee y_2 \vee y_3)$ is true. Then, setting $z_i = (\neg(y_1 \wedge y_2)) \wedge y_3$ and $b = 0$ gives a $\neq$-assignment to $(y_1 \vee y_2 \vee z_i)$ and $(\neg z_i \vee y_3 \vee b)$.

If $(y_1,\ldots,y_n, z_1,\ldots,z_m,b)$ is a $\neq$-assignment to the reduced problem, then either $(y_1,\ldots,y_n)$ (if $b = 0$) or its negation (if $b = 1$) is a satisfying assignment to the original 3SAT problem.

(c.) Since ≠SAT is in NP (nondeterministically guess an assignment and verify), and since a polynomial time reduction exists from 3SAT to ≠SAT, ≠SAT is NP-complete.

Problem 4:

We use the reduction from ≠SAT to MAX-CUT described in the problem, and ask for a cut of size $(3k)^2v+2k$. If a ≠-assignment to the original problem exists, a cut of the requested size exists. Place a node x on the left side of the cut if x is assigned true, and place it on the right side if x is assigned false. All $(3k)^2$ edges of the variable gadgets are cut, and since each clause gadget contains a node on each side of the cut, the cut contains two edges from each clause gadget.

Conversely, the cut may contain only two edges from each clause gadget, and thus must contain all variable gadget edges. The cut must be of the form specified in the first part, and can be transformed to a ≠-assignment to the original problem.

Problem 5:

Let $L = \{ \langle x, a, b\rangle \mid x$ has a factor in the range $[a, b]\}$. This is clearly in NP, so by assumption, this language is in P as well. The factors of x can then be extracted using a binary search.

Problem 7:

The clause $(x \vee y)$ is logically equivalent to each of the expressions $(\neg x \rightarrow y)$ and $(\neg y \rightarrow x)$. We represent the 2cnf formula ϕ on the variables $x_1,\ldots,x_n$ by a directed graph G on $2m$ nodes labled with the literals over these variables. For each clause in ϕ, place two edges in the graph corresponding to the two implications above. ϕ is satisfiable iff G doesn't contain a cycle containing both x_i and $\neg x_i$ for some i. Testing for such a cycle is easily done in polynomial time with a depth-first search algorithm.

Problem 8:

First we show that Z is in DP. Consider the following two languages:

$Z_1 = \{\langle G_1, k_1, G_2, k_2\rangle \mid G_1$ has a k_1 clique, G_2 is a graph, and k_2 is an integer $> 2\}$
$Z_2 = \{\langle G_1, k_1, G_2, k_2\rangle \mid G_2$ has a k_2 clique, G_1 is a graph, and k_1 is an integer $> 2\}$

Clearly Z_1 and Z_2 are in NP, and $Z = Z_1 \cap$ (complement of Z_2), so Z is in DP.

To show that Z is complete for DP we need to show that for all A in DP, A is polytime reducible to Z. Let $A = A_1 \cap$ (complement of A_2) for NP languages A_1 and A_2. By the NP completeness of CLIQUE, A_1 and A_2 are polytime reducible to CLIQUE. Let f_1 and f_2 denote the corresponding polytime reduction mappings. Then $f = (f_1, f_2)$ is a polytime reduction from A to Z.