CS 420 Spring 2019
Supplemental Homework Solutions

1. **Exercise 7.1**

   (a) TRUE

   (b) FALSE

   (c) FALSE

   (d) TRUE

   (e) TRUE (since $3^n = (2^{\log_2 3})^n = 2^{(\log_2 3)n}$.)

   (f) TRUE

2. **Exercise 7.4**

		j			
		1	2	3	4
i	1	T	R, T	S	S, R, T
	2	–	R	S	S
	3	–	–	T	R, T
	4	–	–	–	R

   Since S is in the $(1, 4)$ entry of the table, w is in $L(G)$.

3. **Exercise 7.6**

   Let A and B be two languages in P, say M_A and M_B are polynomial time Turing machines that decide A and B, respectively. Let M_A run in time $O(n^k)$ and M_B run in time $O(n^l)$. We give polynomial time bounded Turing machines M_U, M_{CON} and M_{COM} that decide the languages $A \cup B$, AB, and $\overline{A}$. This will show that P is closed under union, concatenation and complementation.

   M_U = "On input w

   1. Run M_A on w. If M_A accepts, then *accept*.
   2. Run M_B on w. If M_B accepts, then *accept*. Otherwise, *reject*."

   M_U decides $A \cup B$ and runs in time $O(n^{\max\{k,l\}})$.

   M_{CON} = "On input $w = w_1 \cdots w_n$

   1. For $k = 0, \ldots, n$, do the following:
   2. Let $u = w_1 \cdots w_k$ and $v = w_{k+1} \cdots w_n$.
   3. Run M_A on u and M_B on v. If both accept, then *accept*.
   4. If none of the executions of stage 3 leads to acceptance, then *reject*."

M_{CON} decides AB. Each execution of stage 3 takes time $O(n^k)+O(n^l) = O(n^{\max\{k,l\}})$. Since stage 3 is executed $O(n)$ times, the running time for M_{CON} is $O(n^{\max\{k,l\}+1})$.

M_{COM} = "On input w

1. Run M_A on w.
2. If M_A accepts, then *reject*. If M_A rejects, then *accept*."

M_{COM} decides $\overline{A}$ and runs in time $O(n^k)$.

4. **Exercise 7.7**

   Let A and B be languages in NP, say N_A and N_B are polynomial time nondeterministic Turing machines that decide A and B, respectively. We give polynomial time nondeterministic Turing machines N_U and N_{CON} that decide the languages $A \cup B$ and AB.

   N_U = "On input w

   1. Nondeterministically chose one of the machines N_A and N_B.
   2. Simulate a computation of the chosen machine on the input w.
   3. If the computation accepts, then *accept*. Otherwise, *reject*."

   N_U decides $A \cup B$ and since N_A and N_B both run in polynomial time, N_U runs in polynomial time.

   N_{CON} = "On input w

   1. Nondeterministically chose strings u and v with $w = uv$.
   2. Simulate a computation of N_A on u and a computation of N_B on v.
   3. If both of the simulated computations accept, then *accept*. Otherwise, *reject*."

   N_{CON} decides the language AB and runs in polynomial time because both N_A and N_B run in polynomial time.

5. **Exercise 7.8**

   Let m be the number of nodes of G. Stages 1 and 4 are executed only once. Stage 3 runs at most m times because each time except the last it marks an additional node in G. Thus, the total number of stages used is at most $m+2$, which is a polynomial in the size of G. Stage 1 is easily implemented in polynomial time. Stages 3 and 4 involve a scan of the input and a test if certain nodes are marked. This can also be implemented in polynomial time. Thus, both the number of stages executed and the running time of each stage is polynomial in the input size, so the running time of the algorithm is polynomial.

6. **Exercise 7.9**

   TRIANGLE is decided by the following Turing machine.

   M = "On input $\langle G \rangle$, where G is a graph

   1. For all triples p, q, r of nodes of G, do the following.
   2. See if $(p, q), (q, r)$ and (p, r) are all edges of G. If so, then *accept*.
   3. If none of the triples leads to acceptance, then *reject*."

   If there are m nodes in G, then the number of triples of distinct nodes of G is $\frac{m(m-1)(m-2)}{6}$. This number is no bigger than m^3, so stage 2 is executed only a polynomial number of times. Since each execution of a stage can be done in polynomial time, the algorithm runs in polynomial time.

   (A similar argument shows that for any fixed k, the k-CLIQUE problem is in P. However, the CLIQUE problem is NP-complete and therefore not known to be in P.)

7. **Exercise 7.10** Show that ALL_{DFA} is in P.

   Solution: ALL_{DFA} is decided by the following deterministic Turing machine.

   T = "On input $\langle M \rangle$ where M is a DFA

   1. Mark the start state of M.
   2. Repeat until no new states are marked.
   3. Mark any state that has a transition coming into it from a marked state.
   4. If every marked state is an accept state, *accept*.

   Steps 1 and 4 are done once. Each time through the loop in Steps 2 and 3 except the last time, an unmarked state is marked, so the loop is executed no more than m times, where m is the number of states in M. Thus, the total number of steps executed is no more than $2m + 2$ and this is polynomial in the size of $\langle M \rangle$.

   Step 1 involves finding the start state in the list of states and marking it. Step 3 involves processing each transition once and for each transition checking if it goes from a marked state to an unmarked state. Step 4 involves checking if each marked state is in the list of accept states. Each of these steps can be implemented in time polynomial in the size of $\langle M \rangle$, so the algorithm runs in polynomial time.

8. **Exercise 7.12**

   The following nondeterministic Turing machine decides ISO.

   N = "On input $\langle G, H \rangle$, where G and H are graphs

1. If G and H have different numbers of nodes, then *reject*.
2. Let the nodes of G be $g_1, \ldots, g_m$ and the nodes of H be $h_1, \ldots, h_m$.
3. Guess a permutation (i.e., a reordering) $g'_1, \ldots, g'_m$ of the nodes of G.
4. For each pair of numbers i, j with $1 \leq i < j \leq m$ do the following:
5. If G has an edge connecting g'_i and g'_j but H has no edge connecting h_i and h_j, or G has no edge connecting g'_i and g'_j but H has an edge connecting h_i and h_j, then *reject*.
6. If none of the executions of stage 5 lead to rejection, then *accept*."

N decides ISO. The maximum number of times stage 5 is executed is $\frac{n(n-1)}{2} = O(n^2)$ since this is the number of pairs i, j of distinct numbers between 1 and n. Thus the number of stages executed is polynomial, and each stage can be implemented in polynomial time, so N runs in nondeterministic polynomial time.

9. **Problem 7.15**

Let A be in P and let M be a Turing machine that decides A in polynomial time. If $y = y_1 \cdots y_n$ with each y_i in Σ, define a table of Boolean values where for $1 \leq i \leq j \leq n$, $table(i, j)$ is true if and only if the string $y_i \cdots y_j$ is in A^*. We can fill in table recursively using the following easy to verify fact: $table(i, j)$ is true if and only if either $y_i \cdots y_j$ is in A or there is a k with $i \leq k < j$ such that $table(i, k)$ and $table(k + 1, j)$ are both true.

A Turing machine M_* that decides A^* is given by

M_* = "On input $y = y_1 \cdots y_n$

1. For $l = 1$ to n,
2. For $i = 1$ to $n - l + 1$,
3. Let $j = i + l - 1$,
4. Run M on input $y_i \cdots y_j$. If M accepts, then set $table(i, j) = true$.
5. For $k = i$ to $j - 1$,
6. If $table(i, k)$ and $table(k + 1, j)$ are both $true$, then set $table(i, j) = true$.
7. If $table(i, j)$ was not set to $true$ above, then set $table(i, j) = false$.
8. If $table(1, n) = true$, then *accept*. Otherwise, *reject*."

Since M runs in polynomial time, each stage runs in polynomial time. Stage 6 is executed no more than n^3 times and the other stages are executed no more often than stage 6, so the number of stages executed is polynomial. Thus M_* decides A^* in polynomial time.

10. **Problem 7.18**

Suppose that $P = NP$ and that A is a language in P different from $\emptyset$ and Σ^*. There are strings x_0 and x_1 such that $x_1 \in A$ and $x_0 \notin A$.

To show that A is NP-complete, we have to show two things. First, we have to show that $A \in NP$. But we are given that $A \in P$ and we know $P \subseteq NP$, so $A \in NP$. Second, we have to show that for every $B \in NP$, we have $B \leq_p A$. So suppose that $B \in NP$. Since we are assuming that $P = NP$, we actually have $B \in P$. Let M be a deterministic Turing machine that decides B in polynomial time. We can now define a polynomial time reduction f of B to A. The Turing machine F computes f.

$F =$ "On input w

1. Run M on w.
2. If M accepts, then output x_1. If M rejects, then output x_0."

If w is in B, then M will accept w and $f(w) = x_1$ which is in A. If w is not in B, then M will reject w and $f(w) = x_0$ which is not in A. Thus, f is a reduction from B to A. Since M runs in polynomial time, so does F, and thus f is a polynomial time reduction.

This shows that every set in NP is polynomial time reducible to A and hence A is NP-complete.

11. **Problem 7.20**

There are two parts to this problem. First, explain why it is believed that PATH is not NP-complete, and second, show that if PATH is proved to not be NP-complete, then it is proved that $P \neq NP$.

For the first part, we know from Theorem 7.14 that $PATH \in P$. If PATH were NP-complete, then by Theorem 7.35, $P = NP$. Since it is believed that $P \neq NP$, it is believed that PATH is not NP-complete.

For the second part, it follows from Problem 7.18 that if $P = NP$, then PATH is NP-complete. Thus, if it is proved that PATH is not NP-complete, then it is proved that $P \neq NP$.

12. (a) A is decided by the following one-tape TM:
$M =$"On input w
    1. Repeat as long as there are at least two a's and one b on the tape:
    2. Scan the tape and cross off two a's and one b.
    3. If there are no a's and b's left, *accept*, else *reject*."

    Each scan takes time $O(n)$ and there are no more than $\lfloor n/3 \rfloor$ scans since each scan crosses off three symbols, so the total running time is $O(n^2)$.

    (b) The following three tape TM decides A in time $O(n)$:
    $N =$"On input w

1. Scan the first tape. For each a read, write on a on tape 2. For each b read, write two b's on tape 3.
2. Scan tapes 2 and 3. If the number of a's on tape 2 equals the number of b's on tape 3, *accept*, else *reject*."