CSE 555 HW 4 SAMPLE SOLUTION

QUESTION 1.

Short PCP is the decision problem: Given a set $T$ of $n$ tiles $[\frac{t_i}{b_i}]$, $1 \leq i \leq n$, and an integer $k$ written in unary, is there a PCP solution $t_i \ldots t_\ell = b_1 \ldots b_\ell$ in which $\ell \leq k$? Show that short PCP is NP-complete.

Short PCP is in NP. The certificate is a sequence of tiles, $T_s$. The following is a polynomial time verifier.

1. Check that the sequence of tiles has length at most $k$. If not, reject.
2. Check that tiles are valid: Mark each tile in $T_s$ and its corresponding tile in $T$. If any tile in $T_s$ has no corresponding tile in $T$, reject.
3. Check that $t_i \ldots t_\ell = b_1 \ldots b_\ell$. If not, reject. Otherwise, accept.

Let $n$ be the length of input $|\langle T, k \rangle|$, so $n$ is an upper bound on $k$. Step 1 takes time at most $k + 1$ as we reject if the sequence is longer than $k$, so $O(n)$. The second step takes time at most $2kn$, so this step is $O(n^2)$. Let $m$ be the length of the longest tile (either $t_i$ or $b_i$). Step 3 takes time at most $km$ where $n$ is an upper bound on $m$, so this step is $O(n^2)$. Then, the entire verifier is $O(n^2)$.

To show that Short PCP is NP-complete, we mimic the reduction of every language in NP to SAT, instead reducing every language in NP to Short MPCP. If $A$ is in NP, there is a nondeterministic TM, $N$, that decides $A$ in time at most $n^K$ for some constant $K$. Then, we construct the tiles for Short MPCP to simulate the computation history of $N$ on some string $w$ as in the original MPCP problem. As a reminder, we construct the following tiles which forces a simulation of $N$ on $w$:

- $[\# \ldots \#]$ for starting configuration
- $[\frac{a}{a}]$, $[\frac{a \text{ accept}}{\text{accept} a}]$, $[\frac{\text{accept} a}{a \text{ accept}}]$ for $a \in \Gamma$
- $[\frac{q_i \text{ accept} \#}{\# \ldots \#}]$ for $\delta(q, a) = (r, b, R)$ for right tape-head movements
- $[\frac{q_i \text{ accept} \#}{\# \ldots \#}]$ for $\delta(q, a) = (r, b, L)$ for left tape-head movements
- $[\# \ldots \#]$ and $[\# \ldots \#]$, $[\frac{\text{accept} \# \#}{\# \ldots \#}]$ to separate configurations and complete the match

Modeling the original reduction to SAT via the tableau, we note that, if $N$ has an accepting computation history for $w$, there are at $n^K$ configurations and each configuration has length $n^K$. Each tile constructed for the Short MPCP problem has length (the greater of $t_i$ or $b_i$)
at least 1, so each configuration requires at most $n^K$ tiles. Then, there is a solution to the Short MPCP problem with no more than $n^K \times n^K = n^{2K} = k$ tiles if there is an accepting computation history of $N$ on $w$. We note that the conversion from MPCP to PCP requires only a constant number of additional tiles and a constant adjustment to the length of tiles, so the reduction from $A$ to Short PCP differs from the reduction from $A$ to Short MPCP by a constant factor.

**Question 2.**

Suppose that we are provided an oracle to answer the question: Given $\langle T, k \rangle$, does the set $T$ of tiles have a PCP solution with $k$ or fewer tiles (with $k$ written in unary)? The oracle takes zero time to compute, but we must write the question to it, and read its answer, and these do take time.

(a) We want to find a solution (solve the construction problem, not just the decision problem). Show how to use the oracle (repeatedly) to find a PCP solution of size at most $k$, if one exists, in polynomial time.

First modify the tiles in the same way that MPCP was reduced to PCP, so that $T_1$ is necessarily the first tile.

Start with tile set $T = \{T_1, \ldots, T_\ell\}$.

`ISTILE(\langle T, k \rangle)` is the oracle that determines whether there is a solution or not.

`TILE(\langle T, k \rangle)` does the following:

(a) If $T_1 = [t_1b_1]$ has $t_1 = b_1$ return

(b) for $j$ from 1 to $\ell$ if one of $t_1t_j$ or $b_1b_j$ is a prefix of the other then

(i) Obtain $T'$ from $T$ by removing $T_1$ and replacing it by $[t_1t_j]$. 

(ii) If `ISTILE(\langle T', k - 1 \rangle)` output $T_j$, call `TILE(\langle T', k - 1 \rangle)`, and then return

The main program is: If `ISTILE(\langle T, k \rangle)` then output $T_1$ and call `TILE(\langle T, k \rangle)` else output “no solution”.

`TILE` is invoked $k$ times at most, and each makes at most $\ell$ calls to the oracle. `TILE` only invokes itself once.

(b) We want to find a smallest solution (solve the optimization problem, not just the construction problem). Show how to use the oracle (repeatedly) to find a PCP solution with the fewest tiles, if one exists having at most $k$ tiles, in polynomial time.

If `ISTILE(\langle T, k \rangle)` then

$s \leftarrow k$
While \( \text{ISTILE}(T, s - 1) \) do \( s \leftarrow s - 1 \) od;
\( \text{TILE}(T, s) \)
fi

(We could be more efficient by using a binary search, but since \( k \) is represented in unary there is no need to.)

**Question 3.**

An \( n \)-fan is a graph \( G = (V, E) \) where \( V = \{v_0, ..., v_{n-1}\} \) and \( E = \{\{v_i, v_{n-1}\} : 0 \leq i < n - 1\} \cup \{\{v_i, v_{i+1}\} : 0 \leq i < n - 2\} \). We say that \( H = (W, F) \) is a spanning subgraph of \( G = (V, E) \) if \( W = V \) and \( F \subseteq E \). Show that the question “Given input \( \langle G \rangle \), does \( G \) have a spanning fan?” is NP-complete.

Call this problem \( \text{SPANFAN} \). \( \text{SPANFAN} \) is in NP; the spanning fan, \( H = (W, F) \) is the certificate. A verifier runs in polynomial time as follows:

1. Verify that \( W = V \) by scanning through \( W \) and marking corresponding nodes in \( V \) and \( W \). If some node in \( W \) fails to be in \( V \), reject. When all nodes in \( W \) are marked, check to see if there are any unmarked nodes in \( V \). Each pass to mark nodes in \( V \) takes time \( O(2^n) \) and there are at most \( O(n) \) passes. The final check takes \( O(n) \) steps, so this phase runs in \( O(n^2) \) time.

2. Verify that \( F \) has the edge properties of an \( n \)-fan.
   (a) Scan \( F \) to determine if some node has degree \( n - 1 \). (We can do this, for example, by making a list of the nodes and keeping a tally of how many edges each node appears in.) If no node has this property, reject. Otherwise, delete this node and all adjacent edges. Call the remaining graph \( H' = (W', F') \).
   (b) On \( H' \), using a method similar to above, determine if there are two nodes with degree 1 with all other nodes having degree 2. If no, reject. If yes, call the nodes with degree 1 \( s \) and \( t \) and use the polynomial time algorithm that decides \( \text{PATH} \) to determine if there is a path between \( s \) and \( t \). If yes, accept; otherwise, reject.

Scanning \( F \) to determine if a node is the “hub” for the fan takes \( O(E) = O(n^2) \) steps. Scanning \( F' \) to determine if two nodes are possible endpoints of a path among the remaining nodes takes \( O(n^2) \) steps. The final check runs in polynomial time. Since all steps of both phases of this verifier run in time polynomial in \( n \), this is a polynomial time verifier.

Claim: \( \text{SPANFAN} \) is NP-complete.
Proof: We show that \( \text{UHAMPATH} \leq P \ \text{SPANFAN} \), and we know \( \text{UHAMPATH} \) to be NP-complete from Theorem 7.55. From \( G = (V, E) \), construct \( G' = (V', E') \) where \( V' = V \cup \{\infty\} \) and \( E' = E \cup \{\{v_i, \infty\} : v_i \in V\} \). In the construction, we have to write
down $n$ additional edges; assume that we lookup the node from the input tape, scan to the end of the tape to add the edge, and return to lookup the next node. Since we do this for $n$ nodes and the representation is at most $O(n^2)$, this takes $O(n^3)$ steps and is therefore a polynomial reduction. To prove that the reduction yields the correct result, we claim $G' \in SPANFAN \iff G \in UHAMPATH$.

This is obvious from our construction. If $G$ has an undirected Hamiltonian path, it forms the “top” of a spanning fan and $\infty$ along with edges $\{v_i, \infty\}$ form the “hub” and “spokes,” and therefore, $G'$ must have a spanning fan.

Our construction of $G'$ added the hub node, $\infty$, of the spanning fan and all of its adjacent edges (spokes), so if $G'$ has a spanning fan with $\infty$ as the base, then the top of the spanning fan must be an undirected Hamiltonian path in $G$. Note that it is possible that there is a spanning fan in $G'$ where $\infty$ is not the hub node, but rather appears along the path that forms the top. For example, if the hub of the fan is $v_0$, the top of the fan is formed by $\{(v_i, v_{i+1}) : 1 \leq i \leq v_{n-2} \cup \{v_{n-1}, \infty\}\}$, and the spokes of the fan are $\{(v_0, v_i) : 1 \leq i \leq v_{n-1}\} \cup \{v_0, \infty\}$. This corresponds to the case that there is spanning fan in $G$ on $n$ vertices and so there is a spanning fan in $G'$ on $n+1$ vertices, but when there is a spanning fan in $G$, there is an undirected Hamiltonian path; take the top of the fan and a spoke from either of the two nodes with degree 2.

As we have proved both directions, the reduction is complete.

(Note that the above proofs assume $n \geq 4$ as finding a spanning fan for $n < 4$ is trivial.)

**Question 4.**

A maximal path is a set $\{x_0, x_1, ..., x_k\}$ in a graph so that $\{x_i, x_{i+1}\}$ is an edge for $0 \leq i < k$, and there is no edge $\{x_0, y\}$ or $\{x_k, y\}$ unless $y \in \{x_0, x_1, ..., x_k\}$. The value of $k$ is the length of the path. Show that the question “Given graph $G$ on $n$ vertices, does $G$ have a maximal path of length equal to $\lfloor \sqrt{n} \rfloor$?” is NP-complete.

Call this problem $MAXPATH$. $MAXPATH$ is in NP; the maximal path is the certificate $P = (W, F)$. Let $m+1$ be the number of nodes in the certificate. A polynomial time verifier works as follows.

1. Check that $W \subseteq V$, $F \subseteq E$ (analysis is similar to Question 3, (1)). If not, reject.

2. Check that the edges in the certificate form a path. (We can do this in polynomial time by checking that the certificate forms a connected component of $m+1$ nodes, that the number of edges present is $m$, and that there are two endpoint nodes - call them $x_0, x_m$ - with degree 1 and the remaining $m-1$ nodes have degree 2.) If not, reject.
(3) Check that the length of the path is equal to \( \lfloor \sqrt{n} \rfloor \). (We can do this in polynomial time on a TM by verifying the inequality \( m^2 \leq n < (m + 1)^2 \).) If not, reject.

(4) Check that the path in the certificate is maximal. (For all \( v \in V \setminus W \), reject if either \( \{v, x_0\} \in E \) or \( \{v, x_m\} \in E \). We do this for \( O(n) \) nodes and check \( O(n^2) \) edges per node, so it is \( O(n^3) \) and therefore polynomial.)

(5) Accept if all checks have passed successfully.

Claim: \( \text{MAXPATH} \) is NP-complete.

Proof: We show that \( UHAMPATH \leq_P \text{MAXPATH} \). Let \( G = (V, E) \) be a graph on \( k + 1 \) nodes. From \( G \), we construct \( G' = (V \cup V', E) \) as follows. Create \((k - 1)k\) new nodes to form a set \( V' \). Then \( V \cup V' \) has \( n = k^2 + 1 \) nodes. Do not add any edges to \( E \). We are just adding nodes to the representation, so this reduction takes \( O(k^2) \) time. We claim that \( G \in UHAMPATH \iff G' \in \text{MAXPATH} \).

\( \iff \)

If \( G' \) has a maximal path of length \( \lfloor \sqrt{n} \rfloor = \lfloor \sqrt{k^2 + 1} \rfloor = k \), since all of the edges in \( G' \) only connect nodes \( V \), then \( G \) has a maximal path of length \( k \). Since \( G \) was a graph on \( k + 1 \) nodes, a maximal path of length \( k \) is an undirected Hamiltonian path.

\( \implies \)

If \( G \) has an undirected Hamiltonian path, it has length \( k \). By definition, this is a maximal path; there does not exist a \( y \in V : y \not\in \{x_0, \ldots, x_k\} \). Since none of the new nodes created for \( G' \) have any adjacent edges and the edge set in \( G' \) remains unchanged from \( G \), this path does not grow or shrink from \( G \) to \( G' \). Therefore, this is a maximal path in \( G' \). Due to the construction of \( G' \), we are guaranteed the property that \( k = \lfloor \sqrt{n} \rfloor \) where \( k \) is the length of the path and \( n \) is the number of nodes in \( G' \).

As we have proved both directions, the reduction is complete.

**Question 5.**

Earlier we saw that \( \text{ALL}_{\text{DFA}} = \{\langle M \rangle : L(M) = \Sigma^* \text{ for the DFA } M \text{ whose input alphabet is } \Sigma\} \) and \( \text{ALL}_{\text{NFA}} = \{\langle M \rangle : L(M) = \Sigma^* \text{ for the NFA } M \text{ whose input alphabet is } \Sigma\} \) are both decidable.

(a) Show that \( \text{ALL}_{\text{DFA}} \) is in \( P \).

Consider the properties a DFA must have for it’s language to be \( \Sigma^* \). 1) The start state must be a final state to accept \( \epsilon \). 2) Any state reachable from the start state must be a final state, otherwise there is a path on some input string that is non-accepting. It may be the case that the DFA contains some states that are not final states and are not reachable from the start state. (Another property, that the DFA must contain loops allowing it to read any arbitrarily long string over \( \Sigma \) is implied by the nature of DFAs in which every state must have exactly one outgoing
transition on every \( a \in \Sigma \) and the finite set of states, so this property is not unique to DFAs whose language is \( \Sigma^* \) and can be disregarded.) A DFA must satisfy both of these properties to be in \( ALL_{DFA} \); any DFA that fails either of these 2 properties is in \( \overline{ALL}_{DFA} \). A polynomial time TM that decides \( \overline{ALL}_{DFA} \) is as follows.

1. Check the set of final states. If \( q_0 \notin F \), accept.
2. Mark all occurrences of \( q_0 \).
3. Examine the transition function and repeat until no new states are marked:
   a. For \( \delta(q, a) = r \) where \( q \) is marked and \( r \) is not marked, if \( r \notin F \), accept.
      Else, mark all occurrences of \( r \).
4. All states that are reachable from the start state have been marked and all are in \( F \), so reject.

To see that this is a polynomial time algorithm, let \( n = |\langle M \rangle| \). Step 1 and step 2 take time \( O(n) \). A new state is marked for every repetition of step 3 except the last one, so it repeats at most \( |Q| < n \) times. Step 3a can be completed in two passes (right to the end, then left to the beginning) of the input, so takes time \( O(n) \), so step 3 takes time \( O(n^2) \). Step 4 is really a fall-through in the case that step 3 completes without accepting, and so takes \( O(1) \). Then, the entire algorithm is done in \( O(n^2) \) time.

Questions for you to ponder: Is \( ALL_{DFA} \) in NL? Is \( \overline{ALL}_{DFA} \) in NL? in L?

(b) Show that \( \overline{ALL}_{NFA} \) is NP-hard.

We show that \( SAT \leq_P \overline{ALL}_{NFA} \) by constructing an NFA, \( M \), that accepts every string except a satisfying assignment of the Boolean formula. Then, if \( L(M) = \Sigma^* \), \( \langle M \rangle \in \overline{ALL}_{NFA} \), the formula is unsatisfiable; else, if \( L(M) \neq \Sigma^* \), \( \langle M \rangle \notin \overline{ALL}_{NFA} \), there is some string that \( M \) does not accept and this corresponds to a satisfying assignment of the formula. The explanation of the construction is as follows:

- We assume there is an order to the variables; that is, if \( True, False, ...Ttrue \) is a satisfying assignment and \( x_1, x_2, ..., x_l \) is the order of the variables, then \( x_1 = True, x_2 = False, ..., x_l = True \) is the assignment. We represent \( True = 1 \), \( False = 0 \) in the NFA, so \( \Sigma = \{0, 1\} \).
- We assume the number of variables in the formula is \( \ell \) and the number of clauses is \( c \).
- Let \( q_0 \) be the start state.
- A satisfying assignment will have length \( \ell \), so the NFA will accept all strings, \( w \), where \( |w| \neq \ell \) as these strings cannot correspond to satisfying assignment. The set of states that correspond to this are \( q_i \) for \( 0 \leq i \leq \ell + 1 \). We add a
transition on both symbols of $\Sigma$ so that $\delta(q_j, \Sigma) = q_{j+1}$, $0 \leq j \leq \ell$ and define $\delta(q_{\ell+1}, \Sigma) = q_{\ell+1}$. Add $q_k \neq q_\ell$ to the set of final states, $F$.

- We allow strings of length $\ell$ to reach a final state so long as they do not correspond to a satisfying assignment. For each of the $c$ clauses, we construct a path with $\ell$ states. We construct a state for each variable for each clause. For variable $x_i$ and clause $C_j$, label the state $x_{i,j}$ for $1 \leq i \leq \ell$, $1 \leq j \leq c$. If literal $x_i$ appears in clause $C_j$, define $\delta(q_{i-1}, 0) = q_i$. If literal $\overline{x_i}$ appears in clause $C_j$, define $\delta(q_{i-1}, 1) = q_i$. If both or neither literals $x_i$ and $\overline{x_i}$ appear in clause $C_j$, define $\delta(q_{i-1}, \Sigma) = q_i$. Add one additional state, $x_0,j$ to correspond to the “start state” for the clause and add $\delta(q_0, \epsilon) = x_{0,j}$. Add $x_{\ell,j}$ to the set of final states for $1 \leq j \leq c$.

As a small example, consider the formula $(x_1 \lor x_2) \land (\neg x_1 \lor x_2)$. The states $q_i$ will accept all strings with length $\neq 2$. The path for clause $C_1$ will accept string 00 and path for clause $C_2$ will accept string 10, corresponding to non-satisfying assignments. No path, however, will correspond to 11 or 01, the satisfying assignments for the formula, and so $L(M) \neq \Sigma^*$. Now consider the formula $(\neg x_1) \land (x_1)$, again over variables $x_1, x_2$. There is no satisfying assignment for this formula. The path for clause $C_1$ will accept $1(0 \cup 1) = \{10, 11\}$ and the path for clause $C_2$ will accept $0(0 \cup 1) = \{00, 01\}$. As all strings of length other than 2 are accepted by the $q_i$ states and all strings of length 2 are accepted by the clause and variable states, this machine accepts everything, as it should when there is no satisfying assignment.

The NFA we construct has $(c+1)(\ell+1)+1$ states with at most 2 transitions from each state, the running time is $O(c\ell)$. The formula had $c$ clauses over $\ell$ variables with both $c$ and $\ell$ bounded by the input size, $n$, so the reduction is done in $O(n^2)$ time.

Questions for you to ponder: Is $\text{ALL}_{NFA}$ in NP? Is $\text{ALL}_{NFA}$ in PSPACE?