Wen Liang(Wen_Liang@student.uml.edu) 01724877

Note:The extra question I did was 6.2, 7.9, 7.10, 7.12

Extra Question:

6.2

Assume for sake of contradiction that there exists some infinite subset of MIN_{TM} that is Turing-Recognizable. Let this subset be S. Since S is Turing-recognizable, we have that there exists some enumerator E that enumerates S. Consider the following turing machine M:
M = On input x:
1. Obtain own source code $< M >$
2. Use E to enumerate elements of S until you find a TM T output by E such that $|< M >| \quad < \quad |< T >|$
3. Output T(x).

We have that step 1 is possible by the recursion theorem. In step 2 will eventually find such a TM T because S is infinite and hence its elements lengths are not bounded. Hence, we have that the above Turing machine M effectively simulates T on all its inputs. i.e., M accepts, rejects and loops on exactly those inputs that are accepted, rejected and looped on respectively by T. Therefore M is equivalent to T. But we have that $|< M >| \quad < \quad |< T >|$. This is a contradiction as T is supposed to be minimal. Hence, MIN_{TM} has no infinite subset that is Turing recognizable.

7.9

We construct a TM M that decides $TRIANGLE$ in polynomial time.
M = " On input $< G >$ where G is a graph:
1. For each triple of vertices v_1, v_2, v_3 in G:
2. If edges $(v_1, v_2), (v_1, v_3)$, and (v_2, v_3), are all edges of G,accept.
3. No triangle has been found in G, so reject."

A graph with m vertices has $m!/(3!(m-3)!) = O(m^3)$ triples of vertices. Therefore, stage 2 will be repeated at most $O(m^3)$ times. In addition, each stage can be implemented to run in polynomial time. Therefore, $TRIANGLE \in P$.

7.10

ALL_{DFA} is the set of every language that is deterministically computed. In other words, there is always a finite number of steps to be computed and the machine never has to make any decisions, it always knows its next state. The class P is defined as a class of

languages that are decidable in polynomial time on a deterministic single tape Turing machine. So, given a TM where we run an input string w, if it is accepted as decidable in polynomial time, then it is Deterministic and therefore in Class P.

M = "On input$< B, w >$, where B is a DFA and w is a string:
1. Simulate B on input w.
2. If the simulation ends in an accept state,accept. If it ends in a non-accepting state, reject."

If input is accepted, then it is in class P.

7.12

A nondeterministic polynomial time algorithm for ISO operates as follows:
"On input $< G, H >$ where G and H are undirected graphs:
1. Let m be the number of nodes of G and H. If they do not have the same number of nodes, reject.
2. Nondeterministically select a permulation π of m elements.
3. For each pair of nodes x and y of G check that (x,y) is an edge of G iff $(\pi(x), \pi(y))$ is an edge of H. If all agree, accept. If any differ, reject."

Stage 2 can be implemented in polynomial time nondeterministically. Stage 3 takes polynomial time. Therefore $ISO \in$NP.

Required Question:

6.1

LISP:
((LAMBDA (X) (LIST X (LIST (QUOTE QUOTE) X)))
(QUOTE (LAMBDA (X) (LIST X (LIST (QUOTE QUOTE) X)))))

Python:
a = "print 'a = ', repr(a), '\n', repr(a)[1:-5]', a"
print 'a = ', repr(a),'\n', repr(a)[1:-5]

C language:
#include <stdio.h>

int main() { char *s = "#include <stdio.h>%cint main() { char *s = %c%s%c; printf(s, 10, 34, s, 34); return 0; }"; printf(s, 10, 34, s, 34); return 0;}

6.6

To solve this problem, we have to be careful about circular definitions, that is, defining M in terms of N, and N in terms of M. Remember that q(w) is a T.M. that prints w on its tape, and halts.

M = "On input w:

(a) Ignore input.
(b) Obtain $< M >$ by the recursion theorem.
(c) Compute $q(< M >)$, and write it to the tape.
(d) Halt"

We let N = $q(< M >)$. When M runs, it writes N = $< q(< M >) >$ in its tape, and when N runs, it writes $< M >$ in its tape.

7.6

P is closed under union. For any two P-languages L_1 and L_2, let M_1 and M_2 be the TMs that decide them in polynomial time. We construct a TM $M^/$ that decides the union of L_1 and L_2 in polynomial time:
$M^/$="On input $< w >$:
1. Run M_1 on w. If it accepts, accept.
2. Run M_2 on w. If it accepts, accept. Otherwise, reject."
$M^/$ accepts w if and only if either M_1 and M_2 accept w. Therefore, $M^/$ decides the union of L_1 and L_2. Since both stages take polynomial time, the algorithm runs in polynomial time.

P is closed under concatenation. For any two P-language L_1 and L_2, let M_1 and M_2 be the TMs that decide them in polynomial time. We construct a TM $M^/$ that decides the concatenation of L_1 and L_2 in polynomial time:
$M^/$=" On input $< w >$:
1. For each way to cut w into two substrings w = $w_1 w_2$:
2. Run M_1 on w_1 and M_2 on w_2. If both accept, accept.
3. If w is not accepted after trying all the possible cuts, reject."
$M^/$ accepts w if and only if w can be written as $w_1 w_2$ such that M_1 accepts w_1 and M_2 accepts w_2. Therefore, $M^/$ decides the concatenation of L_1 and L_2. Since stage 2 runs in polynomial time and is repeated at most O(n) times, the algorithm runs in polynomial time.

P is closed under complement. For any P-language L, let M be the TM that decides it in polynomial time. We construct a TM $M^/$ that decides the complement of L in polynomial time:
$M^/$ = " On input $< w >$:
1. Run M on w.
2. If M accepts, reject. If it rejects, accept."
$M^/$ decides the complement of L. Since M runs in polynomial time, $M^/$ also runs in polynomial time.

7.7

NP is closed under union. For any two NP-languages L_1 and L_2, let M_1 and M_2 be the NTMs that decide them in polynomial time. We construct a NTM $M^/$ that decides the union of L_1 and L_2 in polynomial time:
$M^/$ = " On input $< w >$:
1. Run M_1 on w. If it accepts, accept.
2. Run M_2 on w. If it accepts, accept. Otherwise, reject."
In both stages 1 and 2. $M^/$ uses its nondeterminism when the machines being run make nondeterministic steps. $M^/$ accepts w if and only if either M_1 and M_2 accept w. Therefore, $M^/$ decides the union of L_1 and L_2. Since both stages take polynomial time, the algorithm runs in polynomial time.

NP is closed under concatenation. For any two NP-languages L_1 and L_2, let M_1 and M_2 be the NTMs that decide them in polynomial time. We construct a NTM $M^/$ that decides the concatenation of L_1 and L_2 in polynomial time:
$M^/$ = " On input$< w >$:
1. For each way to cut w into two substrings w = $w_1 w_2$:
2. Run M_1 on w_1.
3. Run M_2 on w_2. If both accept, accept; otherwise continue with the next choice of w_1 and w_2.
4. if w is not accepted after trying all the possible cuts, rejects."
In both stages 2 and 3, $M^/$ uses its nondeterminism when the machines being run make nondeterministic steps. $M^/$ accepts w if and only if w can be expressed as $w_1 w_2$ such that M_1 accepts w_1 and M_2 accepts w_2. Therefore, $M^/$ decides the concatenation of L_1 and L_2. Since stage 2 , stage 3 runs in polynomial time and is repeated for at most O(n) time, the algorithm runs in polynomial time.

7.8

Here is the high-level description of TM M that decides $CONNECTED$
M = " On input $< G >$, the encoding of a graph G:
1. Select the first node of G and mark it.
2. Repeat the following stage until no new nodes are marked:
3. For each node in G, mark it if it is attatched by an edge to a node that is already marked.
4. Scan all the nodes of G to determine whether they are all marked. If they are, accept; otherwise, reject."

Here we give a high-level analysis of the algorithm. This analysis is sufficient to allow us to conclude that it runs in polynomial time. A more detailed analysis would go into the head motion of the TM, but we do not do that here. The algorithm runs in $O(n^3)$ time.
Stage 1 takes at most $O(n)$ steps to locate and mark the start node. Stage 2 causes at most n+1 repititions, because each repetition except for the last repetition marks at least one additional node. Each execution of stage 3 uses at most $O(n^2)$ steps by examining all adjacent nodes to see whether any have been marked. Therefore in total, stage 2 and 3 take $O(n^3)$ time. Stage 4 uses $O(n)$ steps to scan all nodes.
Therefore the algorithm runs in $O(n^3)$ time and $CONNECTED$ is in P.