5.12

First, let assume that it is decidable N is decider of the L(M,w) then, we construct another Turing machine A as follow:

replace write blank symbols to X in all transitions.

when it reads X, do the same thing as read blank in M.

Before going to accept state, write x, then blank overwriting X, then go to accept state.

Output of A will be the same as input of M Therefore, N is decider of A_{TM}. But A_{TM} is known as undecidable. So N does not exist and the problem is undecidable.

5.23

Let $B = 0^*1^*$

Because it is a iff statement, there are two things needed to be proved.

1, if A is decidable, then $A \leq_m B$

2, if $A \leq_m B$ then A is decidable

Assume that A is decidable, there is a decider M for it.

Because we have a desider, we get

$f(s) \Rightarrow x$ for s in A

x in B

$f(\neg s) \Rightarrow \neg x$

So (1) is Proved

Since $A \leq_m B$, there is a f(x) mapping A to B.

We can build a M on input w

1. compute f(w)
2. if f(w) in form $0^* 1^*$, accept else reject

for $w \in A$

f. will check whether f(x) in form $0^* 1^*$. then M accept w

M therefore decides A and

(2) is proven.

5.24

Lets assume that there is a $z \in \Sigma^*$ and $f(z) = \langle z \rangle$. By definition of mapping reduction, $z \in \overline{A_{TM}}$ iff $\langle z \rangle \in J$, therefore $\overline{A_{TM}} \leq_m J$ and by the corollary 5.29 $\overline{A_{TM}}$ is not Turing recognizable, implying J is also not turing recognizable.

again, assume there is a string $s \in \Sigma^*$ that $f(s) = \langle s \rangle$. By definition of mapping reduction, $s \in A_{TM}$ iff $\langle s \rangle \in J$. Therefore $A_{TM} \leq_m J$, because J and $\overline{J}$ share the mapping property $\overline{A_{TM}} \leq_m \overline{J}$. Again, By corollary 5.29, $\overline{A_{TM}}$ is not turing recognizable imply $\overline{J}$ is also not turing recognizable.

5.25

A_{TM} is known to be undecidable.

To Prove that $A_{TM} \leq \overline{A_{TM}}$, we need to prove that there is a f that

$f(x) = y$ for $x \in A_{TM}$ and $y \in \overline{A_{TM}}$

Let f be:

if M accepts w, reject

if M rejects or doesn't halt, accept.

Then $f(x) = y$ for $x \in A_{TM}$ and $y \in \overline{A_{TM}}$ is satisfied.

A_{TM} is undecidable and $A_{TM} \leq_m \overline{A_{TM}}$

7.5

To be satisfiable, the expression can be evaluated as 1 with given assignment of X Y.

First, assume $X=1$, $Y=0$

the original formula

$= (1 \vee 0) \wedge (1 \vee 1) \wedge (0 \vee 0) \wedge (0 \vee 0) \wedge (0 \vee 1)$

$= 1 \wedge 1 \wedge 0 \wedge 0 \wedge 1$

$= 0$ (not satisfy)

Second assume $X=0$ $Y=1$

original formula:

$= (0 \vee 1) \wedge (0 \vee 0) \wedge (1 \vee 1) \wedge (1 \vee 1) \wedge (1 \vee 0)$

$= 1 \wedge 0 \wedge 1 \wedge 1 \wedge 1$

$= 0$ (not satisfy)

Therefore, this expression is not satisfiable

1.6 P is decidable in $\bigcup_k TIME(n^k)$

Union: Assume we have $P_1, P_2 \in P$

Build a turing machine:

M = "on input w:

① check $w \in P_1$

② if ① is false, check $w \in P_2$

③ if either ① or ② is true, accept

else reject.

The run time of this machine is the combination of 2 polynomial time which is another polynomial time. So P is closed under union.

Concatenation:

Build a TM M:

M = " on input w of length n

① split n on index from first to last

② check if $s_1 \in P_1$ and $s_2 \in P_2$

③ if ② is true, accept

else, go back to 1 and increase index. until it is at last index.

④ reject.

runtime is $n \cdot (P_1 + P_2)$. this is still polynomial time and P is closed under concatenation.

Complement.

To prove complement, we just need to switch the accept states of P to reject state and reject state to accept state. This results in $\overline{P}$ with the same overall runtime. So P is closed under complement.

7.)

Suppose we have 2 machines $T_1, T_2 \in NP$ that both of them have non deterministic Poly run time.

Build a TM M

M = " on input w:

1. simulate T_1 on w, if T_1 accepts, accepts.
2. simulate T_2 on w, if T_2 accepts, accepts.
3. reject.

The runtime of this T_{AUB} machine is runtime of T_1 plus T_2. which is also NP

So NP is closed under union.

Concatenation:

Build a machine $T_{A \circ B}$:

$T_{A \circ B}$ = " on input w of length n

1. split w on index, from first to last
2. simulate T_1 on S_1, if accepts, accept.
3. simulate T_2 on S_2, if accepts, accept.
4. Back to 1 unless, it is at last index.
5. reject.

This is running $O(T_1 + T_2)$ n times. therefore it is still NP and NP is closed under concatenation.

7.4 TRIANGLE = $\{\langle G \rangle \mid G$ contains triangle$\}$

To show that TRIANGLE $\in P$, we build a TM T

T = " on input $G(V, E)$

① find all combination of X, Y, Z, where X, Y, Z are different edges in the graph.

② check any $[(X,Y)(Y,Z)(Z,X)]$ all in V.

③ accept if ② is true, accept else, reject.

Finding all combination of X, Y, Z cost $O(V^3)$ time and checking for ② cost $O(E)$. Together the total run time is $O(V^3E) \in P$

Therefore TRIANGLE $\in P$.

7.13;

To show that modular expression is under running time P, we should know that $a^b \equiv c \bmod p$ means that $\frac{a^b - c}{p}$ has remainder as 0. Then we can build a TM M:

M = "on input $\langle a, b, c, p \rangle$

① let $x = a \bmod p$, $i = 0$, $y = 1$

② for b_i in $b \{b_1 b_2 \cdots b_i \cdots b_{12}\}$ from b_1 to b_{12}

if $b_i = 1$, $x = x^2 \bmod p$, $y = y \cdot x \bmod p$, $i+=1$

then calculate next b_i

③ if $y - c \bmod p$ is 0, accept $\left(y = a^b - c\right)$

①③ is constant runtime, ② is running n times of loop for n = length of b in binary. Each loop cost O(1) running time. So, in total Modexp = $O(n+1+1) = O(n)$