Solution Sketches for some Homework Problems

7.10 'Enumerate all triples of vertices in G. For each triple (a, b, c), check if (a, b), (b, c), and (c, a) are edges. If so accept.
Reject if ~~we~~ none of the triples causes us to accept.'

Since the number of triples we check is at most $|V|^3$, where $|V|$ is the number of vertices, and $|V| \leq n$, where n is the input size, this algorithm can be implemented to run in polynomial time on a single-tape TM.

7.33 The error is in the assertion that SAT is not in P. It is true that the algorithm described is exponential, but this does not mean that there does not exist another, clever, polynomial-time algorithm.

8.3 Player II has a winning strategy.

In his first move, player I has no choice but to pick 2.

Player II picks 4.

Player I has no choice but to pick 5.

Player II picks 6.

Player I has no move, so he loses.

Player I does not have a winning strategy because Player II does.

8.6 Suppose A is PSPACE-hard.

This means that for any $L \in PSPACE$, $L \leq_p A$.

Now Consider any $L' \in NP$.

Since $NP \subseteq PSPACE$, $L' \in PSPACE$.

Thus $L' \leq_p A$.

Since for any $L' \in NP$, we have $L' \leq_p A$,

A is NP-hard.

22C : 131 Limits of Computation
Spring 2005
Homework 2
Due on March 3

1. [7 points] Exercise 7.10
2. [15 points] Problem 7.26 (You should do a reduction from 3SAT.)
3. [8 points] Problem 7.33
4. [7 points] Exercise 8.3
5. [8 points] Exercise 8.6
6. [15 points] Problem 8.14

8.9

We already know that $NP \subseteq PSPACE$.

Under the hypothesis, we show

$$PSPACE \subseteq NP.$$

Since 3SAT is NP-hard, the hypothesis implies that 3SAT is PSPACE-hard.

This means that for any $L \in PSPACE$, $L \leq_p 3SAT$.

Since $3SAT \in NP$, and $L \leq_p 3SAT$, we conclude (this needs an argument) that $L \in NP$.

Thus $PSPACE \subseteq NP$.

9.1. First observe that

$$2^n \text{ is } O(2^{n+1})$$

and

$$2^{n+1} \text{ is } O(2^n). \quad (\text{Why?})$$

So $L \in TIME(2^n)$.

$\Leftrightarrow$ L is decidable by an algorithm with running time $O(2^n)$

$\Leftrightarrow$ L is decidable by an algorithm with running time $O(2^{n+1})$

$\Leftrightarrow$ $L \in TIME(2^{n+1})$

9.2 Argue that

$$2^n \text{ is } o\left(\frac{2^{2n}}{\log 2^{2n}}\right),$$

2^{2n} is time constructible,

and apply Corollary 9.11.

9.10 The error is in concluding that because every language in NP is polynomial time reducible to SAT, $NP \subseteq TIME(n^k)$.

$L \leq_p SAT$ and $SAT \in TIME(n^k)$ does not mean $L \in TIME(n^k)$. It only means $L \in TIME(n^{k'})$ for some other constant k' (that depends on L).