1

(Sipser 9.12) Describe the error in the following "proof" that $P \neq NP$. Assume that $P = NP$, then $SAT \in P$ and so for some $k, SAT \in TIME(n^k)$. Because every language in NP is polynomial time reducible to SAT, you have $NP \subset TIME(n^k)$. By the time hierarchy theorem, $TIME(n^{k+1})$ contains a language that is not in $TIME(n^k)$, which contradicts $P \subset TIME(n^k)$. Therefore $P \neq NP$

> The error in this proof comes in the line that "every language in NP is polynomial time rediculble to SAT, then $NP \subset TIME(n^k)$". While it is true that every language is in NP is polynomial time reducible to SAT, it is not guarenteed that the function f that reduces L to SAT is less than n^k. The correct conclusion to their sentence is $NP \subset TIME(O(f(n)+n^k))$. This then makes their following argument incorrect because we cannot determine if $TIME(O(f(n) + n^k)) \subset TIME(n^{k+1})$ because $f(n)$ is not a constant term for all languages in NP.

2

Prove that $USAT \in P^{SAT}$

> We need to show there is an oracle machine M that queries SAT to solve $USAT$ in polynomial time.
>
> M on input w:
>
> > Check if input is a CNF, reject if not
> >
> > Query SAT on w, if SAT rejects then reject.
> >
> > Otherwise:
> >
> > For each variable x: $>>$ Set $x = T$ and create a new CNF based on w. If a clause in w did not contain $x \vee \neg x$ then add it to our new CNF. If a clause contained $\neg x$ then remove that literal from the clause and add the new clause to the new CNF. If the clause contained x (not the negation) then do not add this clause. Query SAT with out new CNF. $>>$ Set $x = F$ and create a new CNF based on w. If a clause in w did not contain $x \vee \neg x$ then add it to our new CNF. If a clause contained x then remove that literal from the clause and add the new clause to the new CNF. If the clause contained $\neg x$ then do not add this clause. Query SAT with out new CNF. $>>$ If both queries accept or if both queries reject then M rejects, otherwise continue.
> >
> > M accepts

Proof of Correctness

If $w \in USAT$ then M accepts w

> $w \in USAT$ then there is only one assignment that makes the CNF true. Then for each variable, one assignment results in a satisfiable CNF and one does not. So for each variable x SAT will say the new CNF is satisfiable for the assignment of x that makes the CNF true and reject the other assignment. This means M accepts w because at each variable the queries were opposing.

If $w \notin USAT$ then M rejects w

> $w \notin USAT$ then there is either more than one assignment that makes the CNF true or no assignment to make it true. If there is no assignment then SAT returns false for the first query so M rejects. Otherwise there exists at least one variable whose assignment can be either T or F and results in a satisfiable CNF. So for that variable x SAT will say the new CNF is satisfiable for both assignments of x. This means M rejects w because at at least one variable the queries were the same.

3

Prove that an oracle C exists for which $NP^C \neq coNP^C$

> Suppose we have language $L = \{w | \exists x \in C \wedge |x| = |w|\}$
>
> The compliment of this language is $\bar{L} = \{w | \forall x \notin C \vee |x| \neq |w|\}$
>
> Let M_L be a NTM that decides L, we need to prove that M_L halts in polynomial time.
>
> M_L simply guesses a string x with size $|x| = |w|$ and accepts or rejects if oracle accepts or rejects. $M_L \in TIME(n)$ because constructing a guess x is $O(n)$ and the guess itself is constant time. If $w \in L$ then M_L will guess $x \in C$ and accept because C accepts x. If $w \notin L$ then M_L will guess a random x and reject because $x \notin C$.
>
> Therefore $L \in NP^C$ because there exists an oracle NTM that decides it in polynomial time.
>
> Since $L \in NP^C$ then $\bar{L} \in coNP^C$
>
> Now we need to show that $\bar{L}$ cannot be decided in polynomial time by an NTM.
>
> Suppose $\bar{L} \in NP$ then $\exists$ NTM $M_{\bar{L}}$ that decides $\bar{L}$ in polynomial time.

Remember $\bar{L}$ accepts all strings w for which there is no string of equal size in C. However the number of strings of equal size to w is $O(2^n)$ which is the number of queries we would need to make to C in order to accept.

Therefore since $M_{\bar{L}}$ can make up to $O(n^k)$ queries it will not be able to say with certainty whether or not a string w is in the language.

Therefore $\exists C(NP^C \neq coNP^C)$

4

Prove that in an interactive proof, if the verifier is required to be a deterministic, polynomial time algorithm with no access to random bits, then the class of languages this system can decide is equal to NP, even if we allow an arbitrary number of queries to the prover.

Call this class IP_P, we need to prove $\exists L \in NPcomplete \wedge L \in IP_P$ and that $\forall L \notin NP \rightarrow L \notin IP_P$

Proof $NP \subset IP_P$

Lets do $3SAT \in IP_P$.

Reminder that $3SAT = \{w | w$ is a CNF formula where each clause is of size 3 and there exists an assignm

The idea for this is very similar to how we prove any language is in NP. Given a string w, we ask the verifier what the assignment is to make the CNF true. Then the verifier simply checks if that assignment is true.

Proof of Correctness

If $w \in 3SAT$ then $\exists P$ that is truthful that convinces V because it returns a valid assignment and our verifier will find that the assignment is valid so it returns true.

If $w \notin 3SAT$ then $\forall P, P$ cannot convince V that $w \in 3SAT$ because P will not be able to come up with a valid assignment for the CNF because one does not exist.

Therefore since $3SAT \in IP_P$ then $NP \in IP_P$

Proof $NP = IP_P$ **(**$IP_P \subset NP$**)**

We need to show that IP_P in NP. That is, given $L \in IP_P$ prove $L \in NP$. If $L \in IP_P$ then there exists a verifier V_P such that:

1. If $w \in L \rightarrow \exists P$ s.t. V_P querying P will accept with prob $> 2/3$
2. If $w \notin L \rightarrow \forall P, V_P$ querying P qill accept with prob $< 1/3$

Create an NTM M that decides L.

M on input w:

Simulate V_P on input w, whenever V would query the prover, instead nondeterministically guess the output of P.

If this leads to a prob $> 2/3$ accept other wise reject.

Create a verifier V to verify M in polynomial time:

V gets the certificate of M which contains the transcript of the query history. This certificate is $O(n^k)$ because there has to be a polynomial amount of guesses ($V_P \in P$) and the output of the prover at each point must also be $O(n^k)$ (again because $V_P \in P$).

V verifies that the resulting probability is $> 2/3$ for an accept.

Proof of Correctness

If $L \in IP_P$ then there exists a V_P that decides it within polynomial time.

If $w \in L$ then $\exists P$ that convinces V_P to accept, which means that there is a possible sequence of query outputs that make V_P accept so M non-deterministically chooses these outputs so V_P will accept so M accepts.

If $w \notin L$ then $\forall P, V$ accepts with probability greater than $1/3$, in other words, there is no sequence of query outputs to make V_P accept with probability greater than $2/3$ so M rejects.

Therefore if $L \in IP_P \rightarrow L \in NP$

If $L \notin IP_P$ then there does not exist a V_P that decides L within polynomial time. Simply either the number of queries required for V_P is $\omega(n^k)$ or the length of each queries is $\omega(n^k)$ (in other words we need exponential runtime). Thus the certificate required for V would also be exponential so V cannot exist so $L \notin NP$

Therefore if $L \notin IP_P \rightarrow L \notin NP$

Thus we have proved $NP \subset IP_P$ and $IP_P \subset NP$ so $NP = IP_P$