Problem 1: [True of False, with justification] (30 points)
For each of the following questions, state TRUE or FALSE. Justify your answer in brief, indicating only the "proof idea" or counterexample, drawing a diagram if needed.

a) If $L \in \mathbf{P}$ and $L' \in \mathbf{NP}$-complete, then there exists a polynomial-time reduction from L to L'.

True. $L \in P \Rightarrow L \in NP$, and all problems in NP reduces in polynomial time to L' Since $L' \in$ NP-complete

b) The empty set is a language which is **NP**-complete.

False. If ϕ were NP-complete then, ~~then~~ for all $L \in NP$, we would have $L \leq_p \phi$. But ϕ has no language and one cannot map $w \in L$ such that $f(w) \in \phi$.

c) It holds that $2^{O(\log n)} = n^{O(1)}$.

True. $2^{O(\log n)} = 2^{\log_2 n \, O(1)} = n^{O(1)}$.

d) *All* the languages contained in $\{0^i \mid i \geq 0\}$ are in **P**.

False. the set of all languages in $\{0^i \mid i \geq 0\}$ is Uncountable. ~~there~~ therefore there are languages there that are not even Turing-recognizable.

Problem 2: (20 points)
For this question we consider the following language:

$$\text{MAXCLIQUE} = \{\langle G, k\rangle \mid G \text{ is a graph and } G \text{ has } \textit{no} \text{ clique with } k \text{ or more vertices}\}.$$

Recall that co−**NP** is the set of languages for which their complement is in **NP** and note that MAXCLIQUE is *different* than the language CLIQUE in Sipser's book.

a) Show that every problem in co−**NP** can be reduced to MAXCLIQUE in polynomial time.

CLIQUE $= \{\langle G, k\rangle \mid$ G has clique of size $k\}$

$\overline{\text{MAXCLIQUE}} = \{\langle G, k\rangle \mid$ G is not a graph or G has clique of size $\geq k\}$

Note that CLIQUE $\leq_p \overline{\text{MAXCLIQUE}}$ since every clique of size $\geq k$ has a clique of size k.

then, $\forall L \in NP$, $L \leq_p$ CLIQUE $\leq_p \overline{\text{MAXCLIQUE}}$.

therefore $\overline{L} \leq_p$ MAXCLIQUE.

b) Show that, if **P** = **NP**, then MAXCLIQUE is in **P**.

MAXCLIQUE $\in$ co-NP.

$P = NP \Rightarrow$ co-NP $= NP$ since, for any $\bar{L} \in$ co-NP,

$$L \in NP \Rightarrow L \in P \Rightarrow \bar{L} \in P$$

Since P is closed under complementation.

Problem 3: (25 points)
Let k be the number of colors. We have 3 balls of each color, and each ball is labelled with a number from the set $\{1, 2, 3, \ldots, 2n-1, 2n\}$. Note that multiple balls can have the same label.

a) Show that it is in **P** to decide that there is a color c such that there are no two balls of color c whose labels add to $2n+1$.

We must construct an algorithm that is polynomial in k.

For each color c, look at each pair of ~~not~~ balls of ~~that~~ color c and check if their labels sum to $2n+1$. If no such pair of balls is found for color C return TRUE.

If ~~we~~ "for" loop finishes without returning TRUE, we return FALSE.

the program above runs in $O(k^3)$ time.

b) Show that it is **NP**-complete to decide that there is a subset of the balls such that there is exactly one ball for each color in the subset and there are no two balls in the subset whose labels add to $2n+1$. Hint. Reduce 3-SAT to this problem.

Each clause is a color.

If X_i is in clause j then create ball with label i and color j.

If $\overline{X_i}$ is in clause j then create ball with label $2n+1-i$ and color j, where n is the number of variables.

A SAT solution has all clauses satisfied, with at least one literal per clause being TRUE. Those are the balls we add to the subset. If $X_i = T$ then $\overline{X_i}$ = FALSE, but any ball for $\overline{X_i}$ has label $2n+1-i$ and is not in the subset. So this gives a valid distribution of balls.

Now we show that a valid distribution of balls give a SAT solution. For each ball with label i in the subset we set $X_i = T$ if $i \leq n$ and $X_i = F$ if $i \geq n+1$. this satisfies the clause j where j is the color of ball i. Since we have a ball from each color, we satisfies all clauses.

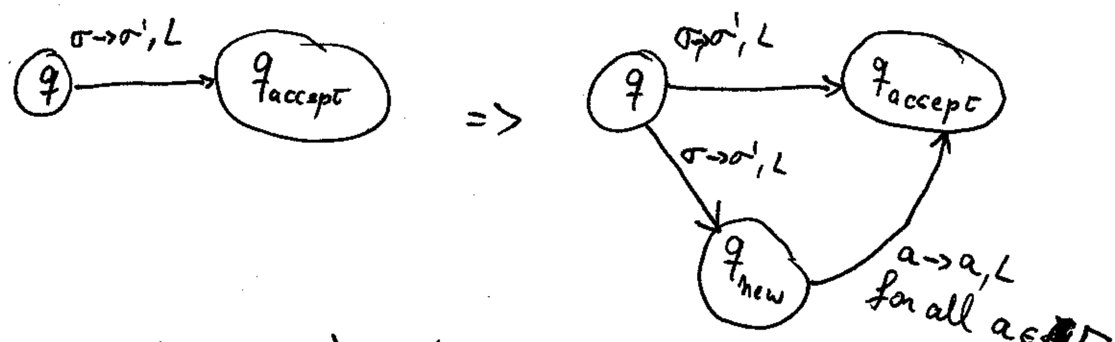
Problem 4: (25 points)

We define a *two-path* Turing machine M as a non-deterministic Turing machine that accepts an input string w if and only if there exists *at least* two distinct accepting computation paths for w.

Recall that an accepting computation path is a sequence of configurations $C_0, C_1, \ldots, C_k$ such that, C_0 is the initial configuration of M, C_k is an accepting configuration of M, and for all $0 \le i < k$, C_i can be obtained from C_{i+1} via a transition of M.

a) Show that M is equivalent to a standard (deterministic) Turing machine.

1) Transform TM M into 2-path TM M'
take each transition to accept STATE of M and duplicates it to a new state q_{new} that has a transition to the accept state



2) Take 2-path TM M' and transforms it into TM M.
M simulates M' as a TM simulates a non-deterministic TM.
the first time a branch of computation is detected to reach an accept STATE, we remember that (encoding on the states of M) and continues until we find another branch of computation that reaches an accept STATE. then M halts and accept.
Otherwise, if M' halts, M reject, and if M' does not halt, M also does not halt.

b) Show that the language

$$A_{2\mathrm{TM}} = \{\langle M, w\rangle \mid M \text{ is a two-path Turing machine and } M \text{ accepts } w\}$$

is undecidable.

We show $A_{TM} \leq_m A_{2TM}$.

We employ the same change in TM M in letter (a) part 1 to ~~show~~ transform M into a two-path TM.

Problem 5: (Bonus question, 10 points)

Prove that the language

$$\{\langle\phi\rangle \mid \phi \text{ is a 3-SAT formula and } \phi \text{ is true for all possible assignments to the variables}\}$$

is in **P**.

Let L be language above.

If $\phi \in L \Rightarrow \overline{\phi}$ is an UNSATISFIABLE DNF formula.

But SATISFIABILITY of DNF formulas is in P.