(10 p) Under the assumption $\text{NP} \subseteq \text{P/poly}$, describe how to construct a polynomial-size family of circuits $\{C_{m,n}\}_{m,n \in \mathbb{N}^+}$ that take any CNF formula $\phi(x, y) = \phi(x_1, x_2, \ldots, x_n, y_1, y_2, \ldots, y_n)$ of size $m$ over $2n$ variables and any assignment $\alpha = (\alpha_1, \alpha_2, \ldots, \alpha_n) \in \{0, 1\}^n$ as inputs, and output an assignment $\beta = (\beta_1, \beta_2, \ldots, \beta_n) \in \{0, 1\}^n$ such that it holds that $\phi(\alpha, C_{m,n}(\phi, \alpha)) = \phi(\alpha_1, \alpha_2, \ldots, \alpha_n, \beta_1, \beta_2, \ldots, \beta_n) = 1$ if such a $\beta$ exists. That is, fill in the details in the construction that we claimed without proof when showing the Karp-Lipton theorem.

Remark: You do not need to provide an exact gate-by-gate specification of the circuits (especially since we do not believe $\text{NP} \subseteq \text{P/poly}$), but you should describe in reasonable detail what subcircuits you use and how they are glued together. Also, make sure to argue why the size is polynomial.
2. We say that a language $L \subseteq \{0,1\}^*$ is sparse if there is a polynomial $p$ such that it holds for every $n \in \mathbb{N}^+$ that $|L \cap \{0,1\}^n| \leq p(n)$. Show that if $L$ is sparse, then $L \in \text{P/poly}$.

3. Show that $\text{ZPP} = \text{RP} \cap \text{coRP}$.

4. During the second half of the course we will be interested in proving size lower bounds for AC$^0$ circuits. When doing so, it can be convenient to use that any AC$^0$ circuit $C$ can be modified to obtain an equivalent circuit $C'$ with the following properties:

1. All gates in $C'$ have fan-out 1 (i.e., it is what is known as a formula, with a DAG structure that is a tree).
2. All NOT ($\neg$) gates are at the input level of $C'$ (i.e., they only apply to variables).
3. The AND ($\wedge$) and OR ($\vee$) gates alternate, so that at each level of $C'$ all gates are either AND or OR.
4. The bottom level has AND gates of some small, bounded fan-in (for the purposes of this problem, let us say some global constant $K$).

Show how these modifications can be done without increasing the circuit depth by more than a constant and the size more than polynomially (so that $C'$ is also a bounded-depth polynomial-size circuit computing the same function as $C$). If $C$ is a circuit of size $S$ and depth $d$, what size and depth do you get for $C'$?

5. We say that a language $L$ is in the complexity class $\Sigma_i^p$ if there exists a polynomial-time Turing machine $M$ and a polynomial $q(n)$ such that $x \in L$ if and only if

$$\exists u_1 \in \{0,1\}^{q(n)} \forall u_2 \in \{0,1\}^{q(n)} \exists u_3 \in \{0,1\}^{q(n)} \cdots Q_i u_i \in \{0,1\}^{q(n)} M(x, u_1, u_2, u_3, \ldots, u_i) = 1$$

(in words, there is a “witness” $u_1$ such that for all “challenges” $u_2$ there is a “witness” $u_3$ such that . . . the Turing machine $M$ accepts $x$ given this extra information if and only if $x$ is in the language).

Let $\Sigma_i \text{SAT}$ be the set of true formulas $\psi$ on the form

$$\psi = \exists u_1 \in \{0,1\}^{q(n)} \forall u_2 \in \{0,1\}^{q(n)} \exists u_3 \in \{0,1\}^{q(n)} \cdots Q_i u_i \in \{0,1\}^{q(n)} \phi(u_1, u_2, u_3, \ldots, u_i)$$

where all $u_i$'s denote sets of variables and $\phi$ is a formula in propositional logic. Show that $\Sigma_i \text{SAT}$ is a complete problem for the class $\Sigma_i^p$.

6. Recall that we write $\Pi_i^p = \text{co} \Sigma_i^p$ to denote the complement of the complexity class $\Sigma_i^p$ and that the union of all such classes form the polynomial hierarchy $\text{PH} = \bigcup_{i \in \mathbb{N}^+} \Sigma_i^p = \bigcup_{i \in \mathbb{N}^+} \Pi_i^p$. If it would hold that $\text{PH} = \Sigma_i^p$ one says that “the polynomial hierarchy collapses to the $i$th level” (which, as we said in class, is generally not believed to be the case).

Prove that if $\Sigma_i^p = \Pi_i^p$, then the polynomial hierarchy collapses to the $i$th level.

---

Note that since all variables in $\phi$ are bound by quantifiers the formula $\psi$ has a fixed truth value which is true or false. Hence, the “SAT” in $\Sigma_i \text{SAT}$ might seem like a bit of a misnomer. But on the other hand the standard SAT problem of deciding whether $\phi(u)$ is satisfiable is the same problem as whether the formula $\exists u \phi(u)$ is true, and from this point of view is is natural to talk about $\Sigma_i \text{SAT}$ problems higher up in the polynomial hierarchy.
In our lectures on Boolean circuits we defined $\text{DTIME}(T(n))/a(n)$ as the class of languages decided by Turing machines $M$ running in time $O(T(n))$ that also get a specific advice string $\alpha_n \in \{0,1\}^{a(n)}$ for inputs of size $n$. We then proved (or at least outlined a proof) that $\text{P/poly} = \bigcup_{c,d \in \mathbb{N}^+} \text{DTIME}(n^c)/n^d$.

Is it possible to change the definition so that not only the advice string $\alpha_n$ depends on the size of the input, but so that we can also pick different Turing machines $M_n$ for different input sizes (while still maintaining that all running times be bounded by a common polynomial $p(n)$), and prove that $\text{P/poly}$ is equal to the set of languages decided by such sequences of Turing machines $\{M_n\}_{n \in \mathbb{N}^+}$ with advice strings $\{\alpha_n\}_{n \in \mathbb{N}^+}$? Work out the details to show that this alternative definition is just as fine, or give a clear mathematical argument why it seems problematic.

Recall that the language $\text{CircuitEval} = \{\langle C, \alpha \rangle \mid C \text{ is a circuit that evaluates to } 1 \text{ on } \alpha\}$ is $\text{P}$-complete (we did not prove this in class, but this fact can be used freely in this problem). We want to understand the complexity of the closely related language $\text{CNFEval} = \{\langle F, \alpha \rangle \mid F \text{ is a CNF formula that evaluates to } 1 \text{ on } \alpha\}$ (in both of the descriptions above we are assuming that the domain of $\alpha$ matches the number of variables in $C$ or $F$, or else we have a no-instance due to syntax error).

Your task is either to prove that $\text{CNFEval}$ is also $\text{P}$-complete, just as $\text{CircuitEval}$ is, or else to explain why this seems unlikely and what problems you run into when trying to prove $\text{P}$-completeness.

A decision tree $T$ is a binary tree with edges directed from the root to the leaves and with leaves labelled 0/1, non-leaves labelled by variables $x_i$, and the two edges out of every non-leaf labelled 0 and 1, respectively. We say that $T$ represents the Boolean function $f : \{0,1\}^n \to \{0,1\}$ if for every assignment $\alpha$ it holds that when starting in the root of $T$ and following the edge labelled by $\alpha(x_i)$ out of every non-leaf labelled by $x_i$ we end up in a leaf labelled by the value $f(\alpha)$.

The depth of a decision tree $T$ is the length of a longest root-to-leaf path in $T$.

In this problem we want to study some connections between decision trees, CNF formulas, and DNF formulas.

Suppose that a Boolean function $f$ can be represented as a decision tree of depth $d$. Show that $f$ can also be represented as a $d$-CNF formula and as a $d$-DNF formula.

Suppose that a Boolean function $f$ can be written both as a $k$-CNF formula and as an $\ell$-DNF formula. Show that this implies that $f$ also can be represented as a decision tree of depth at most $k\ell$. 
10 (50 p) Given a (multi)set $A = \{a_1, a_2, \ldots, a_m\}$ of integer terms and a target sum $B$, does there exist a subset $S \subseteq [m]$ such that $\sum_{i \in S} a_i = B$? This problem is known as SubsetSum and is NP-complete. We want to analyse a purported proof of NP-hardness and study what happens when one tinkers with the reduction.

10a (15 p) Consider the following suggested reduction of 3-Sat to SubsetSum. We are given a 3-CNF formula $F$ with $m$ clauses $C_1, \ldots, C_m$ over $n$ variables $x_1, \ldots, x_n$. We build from this $F$ a SubsetSum instance with $2(n + m)$ integer terms and target sum as follows, where all numbers below have $n + m$ decimal digits each:

- For each variable $x_i$, construct numbers $t_i$ and $f_i$ such that:
  - the $i$th digit of $t_i$ and $f_i$ is equal to 1;
  - for $n + 1 \leq j \leq n + m$, the $j$th digit of $t_i$ is equal to 1 if the clause $C_{j-n}$ contains the literal $x_i$;
  - for $n + 1 \leq j \leq n + m$, the $j$th digit of $f_i$ is equal to 1 if $C_{j-n}$ contains $\overline{x_i}$, and
  - all other digits of $t_i$ and $f_i$ are 0.

- For each clause $C_j$, construct numbers $u_j$ and $v_j$ such that:
  - the $(n + j)$th digit of $u_j$ and $v_j$ is equal to 1 and
  - all other digits of $u_j$ and $v_j$ are 0.

- The target sum $B$ has:
  - $j$th digit equal to 1 for $1 \leq j \leq n$ and
  - $j$th digit equal to 3 for $n + 1 \leq j \leq n + m$.

Is the the above a correct reduction from 3-Sat to SubsetSum that proves the NP-hardness of the latter problem? If your answer is yes, then give a full proof of correctness showing that the reduction (i) is polynomial-time computable, (ii) maps yes-instances to yes-instances, and (iii) maps no-instances to no-instances. If your answer is no, then show how at least one of these conditions fails to hold.

10b (15 p) Given a 3-CNF formula $F$ with $m$ clauses over $n$ variables, run the same reduction as in problem 10a except that the numbers $u_j$ and $v_j$ are omitted and the target sum $B$ has all digits equal to 1. Formulas that map into satisfiable instances of SubsetSum under this modified reduction have a very specific form of satisfying assignments. Describe what such assignments look like.

10c (20 p) Consider the language HACKEDSat consisting of 3-CNF formulas that map to satisfiable SubsetSum instances under the reduction in problem 10b. What is the complexity of deciding this language? Is it in NP? In P? Or NP-complete? For full credit, provide either a polynomial-time algorithm or a reduction from some problem proven NP-complete in chapter 2 in Arora-Barak or during the lectures.