

- [10] 1. Which of the following statements are true? Just indicate for each statement whether it is true or false; you don't need to justify your answers.
- (a) Every  $k$ -tape TM running in time  $t = t(n)$  can be simulated on a 1-tape TM in time  $t \log t$ .
  - (b) The class  $NL$  is closed under complementation.
  - (c) The class  $NL$  is contained in  $P$ .
  - (d) If  $\Pi_2^p = \Sigma_2^p$ , then SAT has superpolynomial circuit complexity.
  - (e) If a language  $L \in NP$  and another language  $L'$  is polytime reducible to  $L$ , then  $L'$  is also in  $NP$ .

**Solution:** The following items are True: (b), (c), (e); the rest are False.

- [10] 2. Show that  $Space(\log n) \neq Time(n)$ .

**Solution:** For the sake of contradiction, assume that  $Space(\log n) = Time(n)$ . Take a language  $L \in Time(n^2) \setminus Time(n)$ ; such a language exists by the Deterministic Time Hierarchy Theorem. Consider the padded version of  $L$ :

$$L_{pad} = \{x\#^{|x|^2} \mid x \in L\}.$$

It is easy to see that  $L_{pad} \in Time(n)$ .

Since  $Time(n) = Space(\log n)$ , we get that  $L_{pad} \in Space(\log n)$ . Note that  $L$  is logspace reducible to  $L_{pad}$ :  $x \in L \Leftrightarrow x\#^{|x|^2} \in L_{pad}$ , and the string  $x\#^{|x|^2}$  can be computed from  $x$  by a logspace Turing machine (with an output tape). Also note that the class  $Space(\log n)$  is closed under logspace reductions: if some language  $A \in Space(\log n)$  and some other language  $B$  is logspace reducible to  $A$ , then  $B$  is also in  $Space(\log n)$ . It follows that  $L \in Space(\log n)$ . Using our assumption that  $Space(\log n) = Time(n)$ , we get that  $L \in Time(n)$ . A contradiction.

### 3. Polynomial-time Hierarchy

- [5] (a) Define the classes  $\Sigma_i^p$ , for  $i \geq 0$ .

**Solution:** For any  $i \geq 0$ , we say that  $L \in \Sigma_i^p$  if there is a polybalanced polytime  $(i + 1)$ -ary relation  $R$  such that: for every  $x$ ,

$$x \in L \Leftrightarrow \exists y_1 \forall y_2 \dots Q y_i R(x, y_1, y_2, \dots, y_i),$$

where  $Q \in \{\exists, \forall\}$  depending the parity of  $i$ .

- [5] (b) State the Karp-Lipton Theorem by completing the following sentence: If  $SAT \in P/poly$ , then ...

**Solution:**  $PH = \Sigma_2^p$  (i.e., PolyTime hierarchy collapses to the second level).

- [10] (c) Sketch the proof of the Karp-Lipton theorem stated above.

**Solution:**

By “Search-to-Decision” reduction for SAT, we get that if SAT has polysize circuits, then there are also polysize circuits for Search-SAT. Such a circuit  $C$  takes as input a formula  $\phi(x_1, \dots, x_n)$ , and outputs a satisfying assignment for  $\phi$ , if it exists; otherwise, the circuit outputs the all-zero assignment.

To show  $PH = \Sigma_2^p$ , it suffices to show that  $\Sigma_2^p = \Pi_2^p$ . For the latter, it just suffices to show that  $\Pi_2^p \subseteq \Sigma_2^p$ . Let  $L \in \Pi_2^p$  be arbitrary language. By definition, there is some polytime, polybalanced relation  $R$  such that, for every  $x$ ,

$$x \in L \Leftrightarrow \forall y \exists z R(x, y, z).$$

For any given  $x$  and  $y$ , define  $f(x, y)$  to be the SAT-formula (in the variables  $z$ ) corresponding to the formula  $\exists z R(x, y, z)$ . That is,  $f(x, y) \in SAT$  iff the formula  $\exists z R(x, y, z)$  is true. Note that for our Search-SAT circuit  $C$ , we have  $\exists z R(x, y, z)$  is true iff  $R(x, y, C(f(x, y)))$  is true.

We claim that  $x \in L$  iff

$$\exists C \forall y R(x, y, C(f(x, y))),$$

where  $C$  is a polysize circuit. Indeed, if  $x \in L$ , then  $C$  solving the SAT-Search problem will satisfy the above formula. Conversely, if some  $C$  satisfies the above formula, then  $\forall y \exists z R(x, y, z)$ , and so  $x \in L$ .

[20] 4. Show that the following languages are  $NP$ -complete. You can use reductions from any  $NP$ -complete language that you know.

(a) Double-SAT =  $\{\phi \mid \phi \text{ is a propositional formula with at least two satisfying assignments}\}$ .

**Solution:** Reduce from SAT. Given  $\psi(x_1, \dots, x_n)$ , produce  $\phi = \psi \wedge (y \vee \bar{y})$ .

To argue the correctness, observe that any satisfying assignment for  $\psi$  can be extended in two ways to a satisfying assignment for  $\phi$  (by setting  $y = T$  and by setting  $y = F$ ). So if  $\psi$  is satisfiable, then  $\phi$  is doubly satisfiable. Conversely, if  $\phi$  is satisfiable, then the same assignment (omitting the variable  $y$ ) will also satisfy  $\psi$ . Thus, if  $\phi$  is doubly satisfiable, then  $\psi$  is satisfiable.

(b) Half-IS =  $\{G \mid G \text{ is an undirected graph on } 2n \text{ vertices with an independent set of size at least } n\}$ .

**Solution:** Reduce from IS. Given a graph  $G$  on  $n$  vertices, and parameter  $k$ , produce  $G'$  where  $G'$  contains  $G$  plus  $n$  new vertices, where exactly  $k$  of the new vertices are connected to each other and to every vertex in  $G$ .

Clearly, if  $G$  has an independent set of size  $k$ , then  $G'$  has an independent set of size  $k + (n - k) = n$ . Conversely, if  $G'$  has an independent set of size  $n$ , only  $n - k$  of the vertices in the independent set can come from the new vertices; the remaining at least  $k$  vertices must be from  $G$ . Hence,  $G$  must have an independent set of size  $k$ .

[10] 5. Give a search-to-decision reduction for the problem Independent Set =  $\{(G, k) \mid \text{undirected graph } G \text{ has an independent set of size at least } k\}$ . That is, state the search version of this problem, and then describe a polytime algorithm solving the search version, given an algorithm that solves the decision version.

**Solution:** The search version is: Given  $G, k$ , find an independent set in  $G$  of size  $k$ , if such a set exists.

Assuming an algorithm for the decision version, we can solve the search version as follows: First check if  $G$  has an independent set of size  $k$ , using the decision algorithm; if not, output “No”. Otherwise, for each vertex  $v$  in  $G$ , if  $G - \{v\}$  has an independent set of size  $k$ , then set  $G = G - \{v\}$ . After the “for” loop is done, output the set of vertices left in  $G$ .

Clearly the running time of the described algorithm is polynomial in the size of  $G$ . To argue the correctness, observe that as long as the size of  $G$  is bigger than  $k$ , there will exist a vertex that can be removed from  $G$  so that the resulting graph still has an independent set of size  $k$  (simply take any vertex not in the independent set). Thus our algorithm will terminate only when the size of the new graph  $G$  is exactly  $k$ . Also, any set of vertices output by our algorithm will be an independent set also in the original graph since we never removed any edge between vertices in the final set.

[10] 6. **Bonus** Consider the following language

Twice-SAT =  $\{(\phi, a) \mid \text{a cnf } \phi \text{ is satisfiable by a truth assignment } a, \text{ and there is at least one other truth assignment satisfying } \phi\}$ .

Is the language Twice-SAT in  $P$  or is it  $NP$ -complete? Justify your answer. (If you say in  $P$ , please outline a polytime algorithm; if you say  $NP$ -complete, please give a reduction from some known  $NP$ -complete problem, and prove the correctness of your reduction.)

**Solution:** The problem is NP-complete. Reduce from SAT. Given a formula  $\phi(x_1, \dots, x_n)$ , first check if  $\phi$  is satisfied by an all-True assignment. If yes, output some trivial tautology (e.g.,  $(x \vee \bar{x})$ ) and any truth assignment. Otherwise, construct  $\phi' = \phi \vee (x_1 \wedge \dots \wedge x_n)$ . Let  $\psi$  be the cnf version of  $\phi'$  obtained by applying distributive laws (to convert  $\phi'$  into a cnf). Observe that  $\phi$  is satisfiable iff  $(\psi, 11 \dots 1)$  is in Twice-SAT.