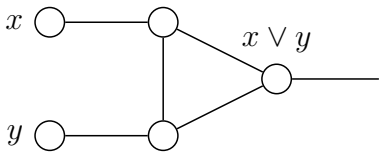
Homework 7

Problem 1. Let D-SAT be the problem of deciding if a CNF formula φ has at least two satisfying assignment. Prove that D-SAT is NP-complete.

Solution. By the NP-completeness of SAT, we only need to prove that SAT $\leq_{\mathrm{P}}$ D-SAT. To decide whether any CNF $\varphi \in$ SAT, we can construct a new CNF $\phi = \varphi \wedge (y \vee \neg y)$ by introducing a new variable y. Apparently this construction can be done in polynomial time. Now we will prove that $\varphi \in$ SAT $\iff$ $\phi \in$ D-SAT. If φ is satisfiable by assignment $A = \{x_1, x_2, ..., x_n\}$, Then $B_1 = \{x_1, ..., x_n, y = 0\}$ and $B_1 = \{x_1, ..., x_n, y = 1\}$ are both satisfying assignments of ϕ. If φ isn't satisfiable, which means $\varphi \equiv 0$, then there will be $\phi = \varphi \wedge (y \vee \neg y) \equiv 0$, since $0 \wedge 1 \equiv 0$. Therefore, $\varphi \in$ SAT $\iff$ $\phi \in$ D-SAT, which means SAT $\leq_{\mathrm{P}}$ D-SAT and D-SAT is NP-complete.

Problem 2. A coloring of a graph is an assignment of colors to vertices in a graph so that no adjacent vertices have the same color. Let 3-COLORING be the problem of deciding if a given graph G has a coloring using three colors.

(a) We will consider two graph gadgets. First, the triangle graph enforces that the vertices have different colors and you can use two of the colors T and F to encode true and false. Second, the OR gadget given in the following graph implements the logical OR operation. Prove that the OR gadget has the property that (1) if both vertices x and y have color F, then so is the vertex labeled by $x \vee y$; and (2) if one of them has color T, then it is possible to assign T to the vertex $x \vee y$ in the gadget.

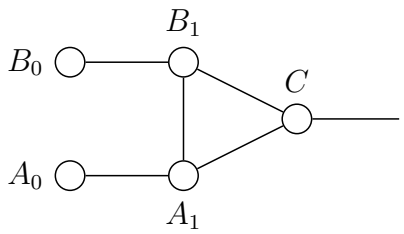


(b) Use the above graph gadgets to prove that 3-COLORING is NP-complete.

Solution.

(a)

(a) Denote the five vertices of the gadget with letters shown in the graph below.



Assume the three colors are T, F, and N, Then if A_0 and B_0 are both colored T, $\{A_1, B_1\} = \{T, N\}$. So C can't be colored with either T or N, which means it can only be colored with F.

(b) Suppose A_0 is colored T, and B_0 with some unknown color $x \in \{F, N\}$. We can color A_1 with x, and B_1 with the third color (apart from T and x). Then C can be colored with T.

(b) We will prove that 3-COLORING $\leq_{\mathrm{P}}$ SAT, which means 3-COLORING is NP-complete. Given a CNF formula $\varphi(x_1, x_2, ..., x_n)$, we can construct a graph G with the following rules:

(a) Create node N to symbolize color N.

(b) Create node F to symbolize color F.

(c) For each variable x_i, create two nodes symbolizing x_i and $\neg x_i$. Draw lines between x_i, $\neg x_i$, N, so $\{x_i, \neg x_i\} = \{\mathrm{T}, \mathrm{F}\}$.

(d) For each clause $C_j = l_1 \wedge l_2 \wedge ... \wedge l_k$, $l_i \in \{x_1, x_2, ..., x_n, \neg x_1, ..., \neg x_n\}$, use the gadget in (a) to connect the relevant variables. First connect the nodes symbolizing l_1 and l_2 using the gadget, and then connect the output node $l_1 \wedge l_2$ with l_3 using the gadget. Repeat this process and we can get a node symbolizing C_j.

(e) For the nodes symbolizing each clauses C_j, connect it with node N and node F, which restricts the clause to be colored T.

Now, we will prove that $\varphi \in$ SAT $\iff$ $G \in$ 3-COLORING.

First, if φ is satisfiable, then we can color the variables with T and F according to the satisfying assignment and color the clause nodes with T. In the satisfying assignment, every clause has at least one true variable, so the output node of the gadget can be colored with T since in (b) we've proved that the output node can be colored with T if one of its input is T. We color node N to N and node F to F, and since all clause nodes are colored T there are no adjacent vertices with the same color. Therefore, $G \in$ 3-COLORING.

Second, if $G \in$ 3-COLORING, we name the color of node N as N, and the color of node F as F. Since all clause nodes are connected to N

and F, so they all need to be colored in T. Meanwhile, in (a) we've proved that the output node will be F if all input nodes are F, so at least one of the input nodes of each clause must be colored T. If node symbolizing x_i is colored T, Then the node for $\neg x_i$ must be F since they are connected with each other and both with N. Therefore, we can assign "true" to all variables whose corresponding nodes are colored with T in G. This will ensure that $\forall i, x_i \neq \neg x_i$ and that all clauses have at least one true variable, which satisfies φ.
Therefore, $\varphi \in$ SAT $\iff$ $G \in$ 3-COLORING, which indicates 3-COLORING is NP-complete.

Problem 3. Write a complete proof for the Claim 1 in the proof of Ladner's theorem discussed in class. That is, show that both $Z(n)$ and $n^{Z(n)}$ are computable in time polynomial in n. The definition of $Z(n)$ is given in the lecture notes.

Solution. We compute $Z(n)$ recursively. Suppose $Z(n-1) = i$, then if M_i can compute $\text{SAT}_Z(x)$ for all x with $|x| < \log n$ in $|x|^i$ time, let $Z(n) = Z(n-1)$. Else, let $Z(n) = \min\{Z(n-1)+1, \log\log n\}$.

There exist at most $O(2^{\log n}) = O(n)$ kinds of input with length $\log n$, so for the simulation to get $Z(n)$ from $Z(n-1)$, we need at most $O(|x|^i \times n) \leq O(n(\log n)^{Z(n-1)}) \leq O(n(\log n)^{\log logn})$ time. By the fact that $\log x < \sqrt{x}$ if x is large enough, we have, with large enough n, $\log\Big((\log n)^{\log\log n}\Big) = (\log\log n)^2 < \log n$ (take $x = \log n$ in the above inequality). So, the above time complexity $O(n(\log n)^{\log logn}) = O\Big(n \times 2^{\log((\log n)^{\log\log n})}\Big) < O(n \times 2^{\log n}) = O(n^2)$.

Therfore, to compute $Z(n)$ from scratch, we need to compute $Z(1)$, $Z(2)$, ..., $Z(n)$ in order. So the total time complexity is at most $\sum_{k=1}^{n} O(k^2) = O(n^3)$, which means $Z(n)$ is polynomial-time (of n) constructible.

Since $Z(n)$ can be computed in $O(n^3)$ time, we can compute $n^{Z(n)}$ by the fast-exponentiation algorithm. We only need $O(\log(Z(n))) \leq O(\log\log\log n)$ amount of multiplication, each with multipliers less than $n^{Z(n)} \leq n^{\log\log n}$. So the length of the multipliers are at most $l = \log(n^{\log\log n}) = \log n \log\log n$, and each multiplication can be done in $O(l^2) < O((\log n)^4)$ complexity. Therefore, the total complexity is at most $O(\log\log\log n \times (\log n)^4) < O((\log n)^5) < O(n)$, which means $n^{Z(n)}$ is also computable in polynomial time of n.