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.

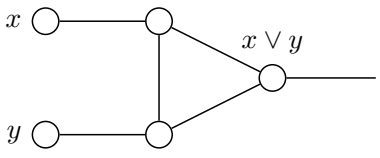
Solution. Firstly, it is easy to see that D-SAT is in NP. Given two different sets of assignments, we can verify if they are both satisfying assignments in polynomial time.

Now we prove that SAT $\leq_P$ D-SAT. Given a CNF formula φ, we construct a new CNF formula $\varphi' = \varphi \vee \varphi^-$. Here, if $\varphi = \varphi(x_1, x_2, \ldots, x_n)$, then $\varphi^- = \varphi(\neg x_1, \neg x_2, \ldots, \neg x_n)$. If $\varphi \in$ SAT, then φ' has at least two satisfying assignments, namely, the satisfying assignments of φ and the satisfying assignments of φ^-. If $\varphi \notin$ SAT, then φ^- has no satisfying assignments either, so φ' has no satisfying assignments. Therefore, we have constructed a computable function $f : \varphi \mapsto (\varphi \vee \varphi^-)$ such that $\varphi \in$ SAT if and only if $f(\varphi) \in$ D-SAT. Thus, have shown that SAT $\leq_P$ D-SAT.

Since SAT is NP-complete, we have 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.

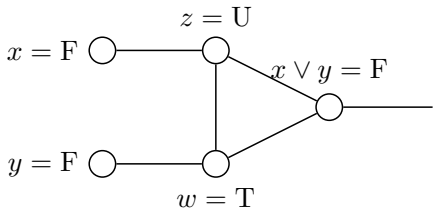
(a) We will consider two graph gadgets. First, the triangle graph enforces that the vertices have different colors and you can use two of the colors T and F to encode true and false. Second, the OR gadget given in the following graph implements the logical OR operation. Prove that the OR gadget has the 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.

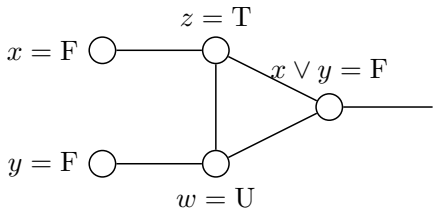


(b) Use the above graph gadgets to prove that 3-COLORING is NP-complete.

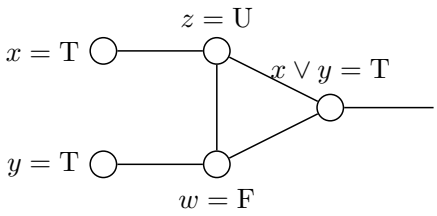
Solution. (a) Let the third color be U.

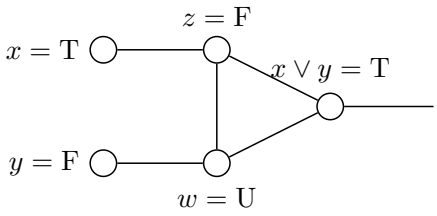
(1) There are only two cases, and in either case, the vertex $x \vee y$ can only be colored F.





(2) There are two cases, and in either case, the vertex $x \vee y$ can be colored T.

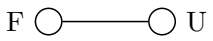




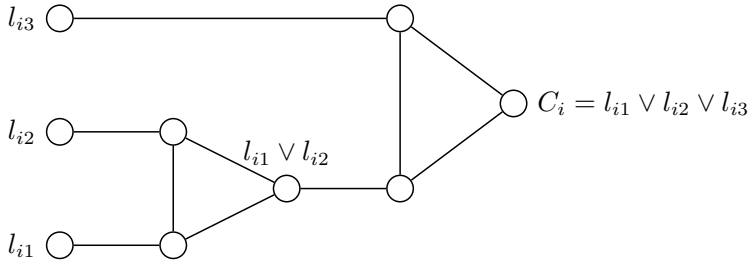
(b) Firstly, it is easy to see that 3-COLORING is in NP. Given a coloring of the vertices, we can verify if it is a valid coloring in polynomial time.

Now we prove that 3-SAT $\leq_P$ 3-COLORING. Given a 3-SAT instance φ with variables $x_1, x_2, \ldots, x_n$ and clauses $C_1, C_2, \ldots, C_m$, we construct a graph G as follows.

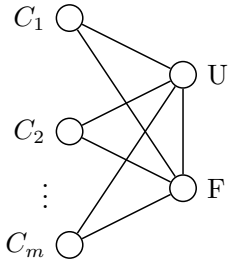
First of all, we can assign two vertices to have color F and U respectively.



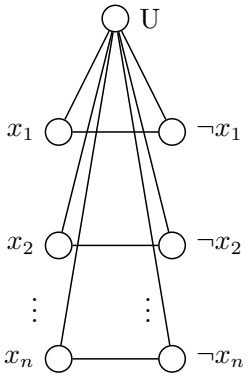
Suppose $C_i = l_{i1} \vee l_{i2} \vee l_{i3}$, where l_{ij} is a literal, meaning that either $l_{ij} = x_k$ or $l_{ij} = \neg x_k$. We can construct a gadget C_i for each clause C_i as follows.



Then we connect all the gadgets C_i to enforce that all of them are true.



Finally, we connect all the variables to enforce that all of them are colored.



If φ is satisfiable, then the graph G has a three-coloring. If φ is not satisfiable, then the graph G does not have a three-coloring. Therefore, we have constructed a computable function $f : \varphi \mapsto G$ such that $\varphi \in$ 3-SAT if and only if $f(\varphi) \in$ 3-COLORING. Thus, have shown that 3-SAT $\leq_P$ 3-COLORING.

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. Definition of Z: Enumerate all TMs $M_1, M_2, \cdots$. Define $Z(1) = 1$. Suppose $Z(n-1) = i$. If for all x of length at most $\log n$, M_i computes $\mathrm{SAT}_Z(x)$, $Z(n) = i$. Otherwise, define $Z(n) = i + 1$.

Firstly, we will show that $Z(n)$ is computable in time polynomial in n. We will prove this by induction on n. It is apparent that $Z(1)$ could be computed in constant time, thus is polynomial in n. Suppose that $Z(n-1)$ is computable in time polynomial in $n-1$, which is also polynomial in n. Now let's consider $Z(n)$. Since we can compute $Z(n-1)$ in polynomial time, we can enumerate all TMs $M_1, M_2, \cdots, M_{Z(n-1)}$, and get $M_{Z(n-1)}$. Then we can compute $\mathrm{SAT}_Z(x)$ for all x of length at most $\log n$. Note that the alphabet of x, denoted as Σ, is finite, so the number of x of length at most $\log n$ is finite, which is on the order of $O(|\Sigma|^{\log n}) = O(n^{\log |\Sigma|})$, and can be computed in polynomial time. Then, we can verify if $M_{Z(n-1)}$ computes $\mathrm{SAT}_Z(x)$ for all x of length at most $\log n$. Hence, we can compute $Z(n)$ in polynomial time.

Secondly, we will show that $n^{Z(n)}$ is computable in time polynomial in n. Since $Z(n)$ is represented in binary format, the length of $Z(n)$ is on the order of $O(\log Z(n))$. Each multiplication can be done in polynomial time in n, and we need to multiply n by itself $Z(n)$ times, where $Z(n) = O(\log \log n)$. Therefore, $n^{Z(n)}$ is computable in time polynomial in n.

Therefore, both $Z(n)$ and $n^{Z(n)}$ are computable in time polynomial in n.