Introduction to Algorithms	**Problem Set 8**
CS 4820, Spring 2019	**Due 11:59pm Thursday, March 28**

Hand in your solutions electronically using CMS. Each solution should be submitted as a separate file. Collaboration is encouraged while solving the problems, but:

1. **list the names of those with whom you collaborated;**
2. **you must write up the solutions in your own words;**
3. **you must write your own code.**

Remember that when a problem asks you to design an algorithm, you must also prove the algorithm's correctness and analyze its running time. The running time must be bounded by a polynomial function of the input size.

(1) *(15 points)* Define ($\leq k$)-CNF to be a CNF with each clause containing at most k literals, and a k-CNF to be a CNF with each clause containing exactly k literals.

(1a) *(8 points)* Consider a variant of 3SAT, which we call $(2,3)$-LSAT, defined in the following way: An input instance of $(2,3)$-LSAT is a (≤ 3)-CNF on n variables, with the additional restriction that each literal appears at most 2 times, and the output is 'Yes' if the given instance is satisfiable and 'No' otherwise. Either (i) design an efficient algorithm that finds a satisfying assignment to an input instance of $(2,3)$-LSAT or outputs 'Impossible' in the case that the input instance is unsatisfiable, or (ii) prove that it is NP-complete.

(1b) *(7 points)* Consider another variant of 3SAT, which we call $(3,3)$-VSAT, defined in the following way: An input instance of $(3,3)$-VSAT is a 3-CNF on n variables, with the additional restriction that each variable appears at most 3 times, and the output is 'Yes' if the given instance is satisfiable and 'No' otherwise. Either (i) design an efficient algorithm that finds a satisfying assignment to an input instance of $(3,3)$-VSAT or outputs 'Impossible' in the case that the input instance is unsatisfiable, or (ii) prove that it is NP-complete.

(2) *(10 points)* Consider the problem of NDPATH defined in the following way: An input instance is of the form (G,S,T), where G is an undirected graph on n vertices, $S = \{s_1,\dots,s_k\}, T = \{t_1,\dots,t_k\}$ are disjoint subsets of nodes of equal cardinality k (for any integer $k \in \{1,\dots,\lfloor n/2 \rfloor\}$). It is an 'Yes' instance of NDPATH if there are node disjoint paths from s_i to t_i, for all $i = 1,\dots,k$, and is a 'NO' instance otherwise. Prove that NDPATH is NP-complete.

Hint: Reduce from 3SAT. For each clause and each variable, add a pair of s_i's and t_i's. For each clause, introduce a few intermediate nodes, corresponding to the variables appearing in the clause.

(3) *(10 points)* The transactions in a blockchain[1] ledger can be modeled as a directed acyclic graph $G = (V,E)$ whose vertex set is partitioned into subsets $V_1, V_2, \dots, V_p$, where V_i represents the set of transactions pertaining to user i, and an edge (u,v) can be interpreted as meaning that transaction u is a predecessor of transaction v. The graph G and its partition $V_1,\dots,V_p$ are assumed to satisfy the following property:

($*$) *For $i = 1,\dots,p$, V_i contains a node r_i that has no incoming edges in G. For every other $v \in V_i$ there is at least one $u \in V_i$ such that $(u,v) \in E$.*

A set of transactions, S, is called *compatible* if it satisfies the following two properties.

[1] No knowledge of blockchains is necessary for solving this problem.

1. For all $(u, v) \in E$, if $v \in S$ then $u \in S$.
2. For all $i = 1, 2, \ldots, p$, if V_i contains three distinct nodes u, v, w such that u has edges to both v and w in G, then v and w cannot both belong to S.

The first constraint can be interpreted as stating that a transaction cannot be accepted unless all of its predecessors are accepted. The second constraint prevents each user i from "double-spending."

Consider the decision problem COMPAT defined as follows. An input instance consists of a directed acyclic graph $G = (V, E)$, a partition of V into subsets $V_1, \ldots, V_p$ satisfying property $(*)$, and a positive integer $k \leq |V|$. It is a 'Yes' instance of COMPAT if and only if there exists a compatible set of at least k transactions. Prove that COMPAT is NP-complete.

Acknowledgement: we thank Haobin Ni for coming up with this question.