Introduction to Algorithms	**Guide to HW 8**
CS 4820, Spring 2019	

(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.

Solution. **(1a)** We prove that $(2,3)$-LSAT is NP-complete. First observe that $(2,3)$-LSAT is in NP since given any input instance ϕ and a truth assignment $\mathcal{A}$ to the variables $x_1, \ldots, x_n$, it can be can be verified in $O(n+\ell)$ time wether $\mathcal{A}$ satisfies ϕ (where ℓ is the number of clauses in ϕ).

We prove NP-completeness by reducing 3-SAT to $(2,3)$-LSAT. Let ψ be an input instance of 3-SAT with ℓ clauses and n variables. Our reduction f is the following:

- For each variable x_i that appears $r \geq 3$ times in ϕ, replace each occurrence of x_i with a fresh new variable, i.e, replace the first occurrence of x_i with $y_{i,1}$, the second occurrence of x_i with $y_{i,2}$, and so on (where $y_{i,j}$'s are new variables).
- For each such x_i add the clauses $(y_{i,1} \vee \overline{y_{i,2}}), (y_{i,2} \vee \overline{y_{i,3}}), \ldots, (y_{i,r-1} \vee \overline{y_{i,r}}), (y_{i,k} \vee \overline{y_{i,1}})$ to the existing set of clauses.
- Let ψ' denote the CNF obtained from ϕ by the above process. Set $f(\psi) = \psi'$.

We note that the above reduction can be done in time $O(n+\ell)$.

We first prove that ψ' is an input instance of $(2,3)$-LSAT. To prove this, we must show that each literal appears at most 2 times in ϕ'. Clearly, if a variable x_i appears at most 2 times, then each of the literals x_i and $\overline{x_i}$ appear at most 2 times. Now consider a variable x_i that appears at least $r \geq 3$ times. We replace each occurrence of x_i with fresh variables $y_{i,1}, \ldots, y_{i,r}$, and also our construction ensure that each of the literals $y_{i,1}, \ldots, y_{i,r}, \overline{y_{i,1}}, \ldots, \overline{y_{i,r}}$ appears exactly 2 times. Thus, it follows that each literal in ψ' appears at most 2 times.

Now we prove correctness of our reduction by proving that ψ is satisfiable iff ψ' is satisfiable. Let ψ be satisfiable, and $\mathcal{A}$ be a satisfying truth assignment of ψ. Construct a truth assignment $\mathcal{A}'$ for ϕ' as follows: If variable x_i appears at most 2 times in ψ, then set the truth value of this variable according to $\mathcal{A}$. If variable x_i appears at least 3 times in ψ, set the truth value of all the variables $y_{i,1}, \ldots, y_{i,r}$ to the same truth value as that of x_i under $\mathcal{A}$. It is easy to check that the only way of satisfying all the clauses $(y_{i,1} \vee \overline{y_{i,2}}), (y_{i,2} \vee \overline{y_{i,3}}), \ldots, (y_{i,r-1} \vee \overline{y_{i,r}}), (y_{i,k} \vee \overline{y_{i,1}})$ is by ensuring that the variables $y_{i,1}, y_{i,2}, \ldots, y_{i,r}$ are all set the the same truth value. Since $\mathcal{A}'$ ensures this, all such clauses are satisfied. Now consider

the clause C_p in ψ containing the j'th occurrence of the vairable x_i. Since x_i is replaced with $y_{i,j}$ which has the same truth value under $\mathcal{A}'$ as x_i under $\mathcal{A}$. This is true for the other variables appearing in C_p as well. Thus, the clause corresponding to C_p in ϕ' must be satisfied under $\mathcal{A}'$.

In the other direction, we show that if ψ' is satisfiable then so is ψ. Let $\mathcal{A}'$ be a satisfying truth assignment for the variables in ψ'. We now show a satisfying truth assignment for ψ. If a variable x_i appears in ψ', then set the truth value of this variable according to $\mathcal{A}'$. If the variables $y_{i,1}, \ldots, y_{i,r}$ appear in ψ', then assign the truth value of the variable x_i to be the same as that of $y_{i,1}$ under $\mathcal{A}'$. As argued above, any satisfying assignment of ψ' must set the same truth value to all the variables $y_{i,1}, \ldots, y_{i,r}$. Further, since each of these variables replaced an occurrence of x_i in ψ, it follows that $\mathcal{A}$ satisfies ψ. This completes the proof of correctness of our reduction. Hence $(2,3)$-LSAT is NP-complete.

(1b) We show that $(3,3)$-VSAT can be solved in polynomial time. Given an instance $\phi = C_1 \vee C_2 \ldots \vee C_\ell$ of $(3,3)$-VSAT on variables $x_1, \ldots, x_n$, where each C_i is a clause containing exactly 3 variables, construct a bipartite graph $G_\phi = (A \cup B, E)$, with A containing a node for clauses $C_1, \ldots, C_\ell$, and B containing a node for variables $x_1, \ldots, x_n$. Further, there is add an edge $e = (a, b) \in E$ if variable a appears in clause b. Thus, $|A| = \ell$ and $|B| = n$.

We claim that the graph G_ϕ has a matching containing all nodes in A. This can be shown using Hall's condition as follows: let $X \subseteq A$ be any subset of A, and let $\Gamma(X)$ denote the set of neighbors of X. Since each each clause has exactly 3 variables, the number of edges going into $\Gamma(X)$ is exactly $3|X|$. Further, since each variable has degree at most 3, the it follows that the number of edges out of $\Gamma(X)$ is at most $3|\Gamma(X)|$. Thus, it must be that $|\Gamma(X)| \geq |X|$, which shows that G_ϕ satisfies Halls' condition, and hence contains a matching containing all nodes in A.

The following algorithm to find the satisfying assignment to an input instance of $(3,3)$-VSAT:

1. Given as input ϕ, construct the graph G_ϕ.
2. Find a matching M in G_ϕ that contains all nodes in A, or output 'Impossible' if no such matching exists.
3. For any (C_i, x_j) in the matching M, assign a truth value to x_j which makes C_i 'True'.
4. Set truth values to the unset variables arbitrarily, and output the truth assignment to $x_1, \ldots, x_n$.

The first step of the algorithm takes time ℓ, and the second step can be done in time $O(\ell^2)$ using the Ford-Fulkerson algorithm. Finally, the other steps can be performed in time $O(n + \ell)$, which gives a total running time of $O(n + \ell^2)$.

The proof of correctness of our algorithm is as follows. By the argument above, we know that we always find a matching in G_ϕ that contains all nodes in A. Thus, the assignment of variables in Step 3 ensures that all the clauses in ϕ are satisfied, and hence it follows that any instance of $(3,3)$-VSAT is satisfiable. We output the corresponding truth assignment in Step 4.

Note that our proof shows that every instance of $(3,3)$-VSAT is satisfiable.

(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, $S = \{s_1, \ldots, s_k\}, T = \{t_1, \ldots, t_k\}$ are disjoint subsets of nodes of equal cardinality k (for any integer $k \in \{1, \ldots, n\}$). It is an 'Yes' instance of NDPATH if there are node disjoint paths from s_i to t_i, for all $i = 1, \ldots, k$, and is a 'NO' instance otherwise. Prove that NDPATH is NP-complete.

Solution. We first claim that NDPATH is in NP. Indeed, given (G, S, T), where $S = \{s_1, \ldots, s_k\}, T = \{t_1, \ldots, t_k\}$, and k paths $P_1, \ldots, P_k$, we can check in polynomial time that the path P_i from s_i to t_i,

and the fact that they are node-disjoint. (We can check node-disjointness by walking on each path, and marking the edges seen. No node should be seen more than once if the paths are node-disjoint.)

We prove NP-completeness of NDPATH by reducing 3SAT to NDPATH. Let ϕ be an input instance of 3SAT. Let $\phi = C_1 \wedge C_2 \wedge \ldots \wedge C_\ell$, using the variables $x_1, \ldots, x_n$, where each C_i is a clause containing exactly 3 literals. We create a graph G_ϕ in the following way: for each clause C_i, create nodes s_i and t_i. Further, for each variable x_j appearing in C_i, create 2 nodes $u_{i,j}, \overline{u_{i,j}}$ corresponding to the literals x_j and $\overline{x_j}$. (Thus, we create 6 such nodes).

Finally, for each variable x_j create nodes $s_{\ell+j}$ and $t_{\ell+j}$. Set $S = \{s_1, \ldots, s_{n+\ell}\}$ and $T = \{t_1, \ldots, t_{n+\ell}\}$.

We now add edges to G_ϕ as follows: For each clause C_i and variable x_j that appears in C_i, add an edge between the node s_i and either $u_{i,j}$ or $\overline{u_{i,j}}$ depending on whether the literal x_j or $\overline{x_j}$ appears in C_j. Add another edge between t_i and either $u_{i,j}$ or $\overline{u_{i,j}}$ depending on whether the literal x_j or $\overline{x_j}$ appears in C_j. (e.g., if $\overline{x_j}$ appears in C_i, then the edges we add are $(s_i, \overline{u_{i,j}})$ and $(\overline{u_{i,j}}, t_i)$).

Further, corresponding to each variable x_j: add two node-disjoint paths from s_{n+j} to t_{n+j}, one path using all nodes from the set $\{u_{i,j} : \ i \in \{1, \ldots, \ell\}$ such that x_j is a literal in $C_i\}$, and the other path using all nodes from the set $\{\overline{u_{i,j}} : \ i \in \{1, \ldots, \ell\}$ such that $\overline{x_j}$ is a literal in $C_i\}$.

We note that given ϕ, the graph G_ϕ can be constructed in time $O(n + \ell)$. Thus the above reduction from ϕ to (G_ϕ, S, T) is efficient.

Now, we prove correctness of our reduction. We claim that ϕ is satisfiable iff G_ϕ has node disjoint paths from s_i to t_i for $i = 1, \ldots, n + \ell$. Suppose ϕ has a satisfying assignment $\mathcal{A}$. Consider a clause C_i. Since C_i is satisfied by A, some literal in C_i must evaluate to 1 (True). We use the intermediate node corresponding to this literal construct a path from s_i to t_i (to be more specific, use the lowest numbered literal that evaluates to 'True'). For instance if $\overline{x_j}$ is the lowest numbered literal in C_i that evaluates to 'True', then the path we use from s_i to t_i uses the edges $(s_i, \overline{u_{i,j}})$ and $(\overline{u_{i,j}}, t_i)$. Since each C_i is satisfied, we can construct edge-disjoint paths from s_i to t_i for $I = 1, \ldots, \ell$. Now consider any variable x_j, and the nodes $s_{\ell+j}$ and $t_{\ell+j}$ corresponding to this. By construction, there are just 2 paths from $s_{\ell+j}$ and $t_{\ell+j}$. If $\mathcal{A}$ assigns x_j to 'TRUE', we use the path from $s_{\ell+j}$ to $t_{\ell+j}$ that uses all nodes from the set $\{\overline{u_{i,j}} : \ i \in \{1, \ldots, \ell\}$ such that $\overline{x_j}$ is a literal in $C_i\}$. In the other case, when $\mathcal{A}$ assigns x_j to 'FALSE', we use the path $s_{\ell+j}$ to $t_{\ell+j}$ that uses all nodes from the set $\{u_{i,j} : \ i \in \{1, \ldots, \ell\}$ such that x_j is a literal in $C_i\}$. Suppose $\mathcal{A}$ assigns x_j to 'TRUE'. Then, none of the paths from s_i to t_i, $i \leq \ell$ use the node $\overline{u_{i,j}}$, by our construction above. Thus, indeed in this case the path from $s_{\ell+j}$ to $t_{\ell+j}$ that uses all nodes from the set $\{\overline{u_{i,j}} : \ i \in \{1, \ldots, \ell\}$ such that $\overline{x_j}$ is a literal in $C_i\}$ is node-disjoint from any s_i to t_i path we constructed above, for $i \leq \ell$. We can argue similarly in the case x_i is assigned 'FALSE'. Since this is true for any $j \in \{1, \ldots, n\}$ (and all the paths from $s_{\ell+j}$ to $t_{\ell+j}$ for different values of j are node disjoint), it follows that G_ϕ has node disjoint paths from s_i to t_i for $i = 1, \ldots, n + \ell$.

Now, suppose that G_ϕ has node disjoint paths from s_i to t_i for $i = 1, \ldots, n + \ell$. For all $j = 1, \ldots, n$, we set the variable to x_j to be 'TRUE' if the path used to connect $s_{\ell+j}$ to $t_{\ell+j}$ uses all nodes from the set $\{\overline{u_{i,j}} : \ i \in \{1, \ldots, \ell\}$ such that $\overline{x_j}$ is a literal in $C_i\}$. We set x_j to be 'FALSE' if the path used to connect $s_{\ell+j}$ to $t_{\ell+j}$ uses all nodes from the set $\{u_{i,j} : \ i \in \{1, \ldots, \ell\}$ such that x_j is a literal in $C_i\}$. We claim that this truth assignment, say $\mathcal{A}$, is a satisfying assignment for ϕ. To see this, consider any clause C_i, and let the node-disjoint path that connects s_i to t_i use $u_{i,j}$ as the intermediate node. It must mean that the node disjoint path that connects s_{n+j} to t_{n+j} uses all nodes from the set $\{\overline{u_{i,j}} : \ i \in \{1, \ldots, \ell\}$ such that $\overline{x_j}$ is a literal in $C_i\}$, and hence x_j is set to 'TRUE' by $\mathcal{A}$. Further, by our construction, a path from s_i to t_i with $u_{i,j}$ as intermediate node indicates that the literal x_j appears in the clause C_i, and thus C_i gets set to 'TRUE' under the assignment $\mathcal{A}$. Since this holds for each clause C_i, it follows that $\mathcal{A}$ is a satisfying assignment for ϕ. This completes the proof of correctness of our reduction, and proves that NDPATH is NP-complete.

(3) *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* $V_1, V_2, \ldots, V_p$*, 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* $V_1, \ldots, V_p$ *are assumed to satisfy the following property:*

(*) For $i = 1, \ldots, p$, 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. *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.*

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

Solution. COMPAT is in NP because given a COMPAT instance $(G, V_1, \ldots, V_p, k)$ and a set S of transactions (vertices of G), a verifier can check that:

1. $|S| \geq k$; this takes $O(n)$ time where n is the number of vertices of G
2. there is no $e = (u, v) \in E$ with $u \notin S$, $v \in S$; this takes $O(m)$ time where m is the number of edges of G
3. none of the sets $V_1, \ldots, V_p$ contains three vertices u, v, w such that (u, v) and (u, w) belong to E and v, w both belong to S. This takes $O(|V_1|^3 + \cdots + |V_p|^3)$ time, which is bounded above by $O(n^3)$.

To prove that COMPAT is NP-complete we reduce from INDEPENDENT SET. Given an instance of INDEPENDENT SET represented by a graph H with n vertices and m edges, and target independent set size ℓ, we create an instance of COMPAT described as follows.

- Let $p = n + m$
- Choose an arbitrary numbering of the vertices of H as $h_1, \ldots, h_n$.
- Choose an arbitrary numbering of the edges of H as $e_{n+1}, e_{n+2}, \ldots, e_{n+m} = e_p$.
- For $i = 1, \ldots, n$ create vertex set $V_i = \{s_i, t_i\}$ and directed edge (s_i, t_i).
- For $i = n+1, \ldots, p$ let h_j and h_k be the endpoints of edge e_i. Create vertex set $V_i = \{u_i, v_{ij}, v_{ik}\}$ and directed edges (u_i, v_{ij}), (u_i, v_{ik}), (v_{ij}, t_j), (v_{ik}, t_k).
- The vertex set of G is $V = V_1 \cup \cdots \cup V_p$. The edge set consists of the directed edges described in the preceding two steps.

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

- Set $k = \ell + n + 2m$.

This reduction runs in polynomial time. In fact, each step can be implemented by doing a constant amount of work per vertex and edge of H, so the entire reduction runs in $O(n + m)$ time.

To prove the reduction is correct, we must show the following two things.

1. **If H contains an independent set T of size at least ℓ, then G contains a compatible set of size at least k.** In fact, a compatible set S can be constructed as follows. First, we select one endpoint of each edge of H according to the following procedure: if $e_i \in E(H)$ has an endpoint $h_j \in T$ then there is a unique such endpoint (since T is an independent set) and we select it; otherwise we select an endpoint of e_i arbitrarily. For each pair (e_i, h_j) select in this way, we insert the vertex $v_{ij} \in V(G)$ into a set P. Now define

$$S = P \cup \{s_i : i = 1, \ldots, n\} \cup \{t_i : i \in T\} \cup \{u_i : i = 1, 2, \ldots, m\}.$$

   By construction, S has $m + n + |T| + m \geq k$ elements. It is compatible because

   - the only pairs of edges that share the same tail are pairs of the form (u_i, v_{ij}) and (u_i, v_{ik}), and for each such pair, only one of v_{ij}, v_{ik} belongs to P by construction;
   - for each edge (s_i, t_i) in $E(G)$, s_i is included in S;
   - for each edge (u_i, v_{ij}) in $E(G)$, u_i is included in S;
   - for each edge (v_{ij}, t_j) in $E(G)$, if t_j is included in S then $h_j \in T$ so e_i selected h_j when we ran the procedure to select one endpoint of each edge; this means $v_{ij} \in P \subset S$.

2. **If G contains a compatible set S of size at least k then H contains an independent set of size at least ℓ.** In fact, $T = \{h_j \mid t_j \in S\}$ is an independent set because if $e_i = (h_j, h_k)$ then at most one of v_{ij}, v_{ik} can belong to S, and if $v_{ik} \notin S$ then $t_k \notin S$ due to the presence of edge (v_{ik}, t_k) in G and the first criterion in the definition of a compatible set. Thus, for each edge $e_i = (h_j, h_k)$ in H at least one of h_j, h_k does not belong to T, which confirms that T is an independent set. Finally, to deduce that T has at least ℓ elements, note that any compatible set can contain at most two elements from each of the sets $V_{n+1}, \ldots, V_{n+m}$ (by the second criterion in the definition of a compatible set) so at most $2m$ elements in total from $V_{n+1} \cup \cdots \cup V_{n+m}$. Since $|S| \geq k = \ell + n + 2m$, the remaining $\ell + n$ elements of S come from $V_1 \cup \cdots \cup V_n$. At most n of those elements can be of the form s_j, so at least ℓ of them are of the form t_j, and each such $t_j \in S$ gives rise to a corresponding element $h_j \in T$, for a total of at least ℓ distinct elements in T as claimed.