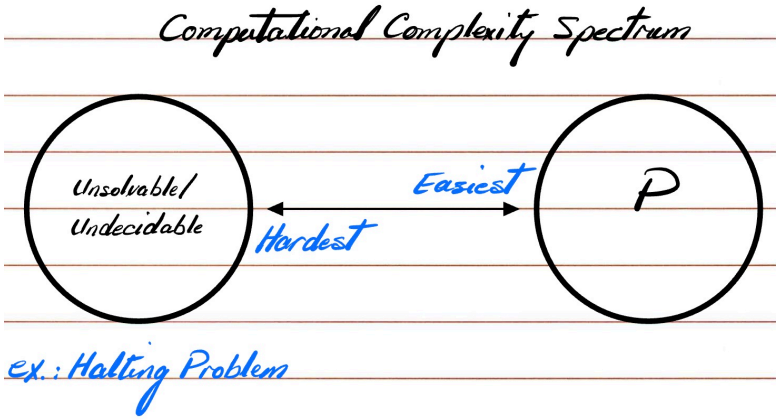
Tractability & Computational Complexity Classes

Julius April 3, 2024

1 Definitions of Computational Complexity

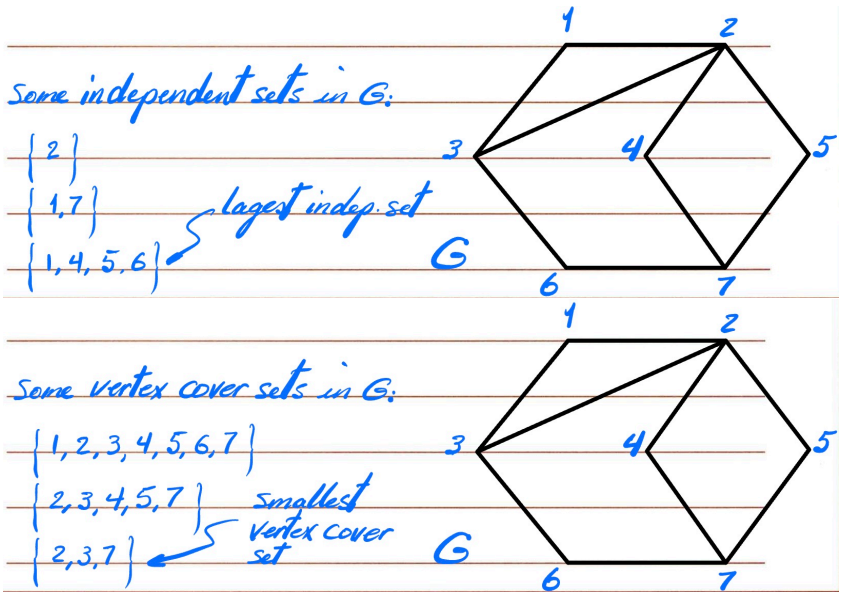
Plan: Explore the space of computationally hard problems to arrive at a mathematical characterization of a large class of them by comparing relative difficulty of different problems.



Loos Definition: If a problem X is at least as hard as problem Y, then if we could solve X, we could also solve Y.
Formal Definition: There is a **polynomial time reduction from Y to X**, i.e. $Y \leq_p X$ if Y can be solved using a **polynomial number of standard computational steps** + **a polynomial number of calls** to a blackbox that solves X.
Then suppose $Y \leq_p X$. If X can be solved in polynomial time, then Y can be solved in polynomial time as well. If Y can't be solved in polynomial time, then X can't be solved in polynomial time either.

2 Independent Set & Vertex Cover Problem

Independent Set: In a graph $G = (V, E)$, we say that a set of vertices $S \subseteq V$ is "independent" if no two nodes in S are joined by an edge.
Independent Set Problem: Find the largest independent set in graph G. (Optimization Version).
Given a graph G and a number k, does G contain an independent set of size at least k? (Decision Version).
Vertex Cover: In a graph $G = (V, E)$, we say that a set of vertices $S \subseteq V$ is a "vertex cover" if every edge in E is incident to at least one node in S.
Vertex Cover Problem: Find the smallest vertex cover set in graph G. (Optimization Version).
Given a graph G and a number k, does G contain a vertex cover of size at most k? (Decision Version).



FACT: Let $G = (V, E)$ be a graph, then S is an independent Set iff its complement $V - S$ is a vertex cover set.

Proof. **Only If Direction:** If S is an independent set, then consider any edge $e = (u, v) \in E$. Since S is independent, there are three possibilities:
1. $u \in S, v \notin S$, then $v \in V - S$, so e is covered by $V - S$.
2. $u \notin S, v \in S$, then $u \in V - S$, so e is covered by $V - S$.
3. $u, v \notin S$, then $u, v \in V - S$, so e is covered by $V - S$.
If Direction: If $V - S$ is a vertex cover, then consider any edge $e = (u, v) \in E$. Since $V - S$ is a vertex cover, there are three possibilities:
1. $u \in V - S, v \notin V - S$, then $u \notin S, v \in S$.
2. $u \notin V - S, v \in V - S$, then $u \in S, v \notin S$.
3. $u, v \in V - S$, then $u, v \notin S$.
So there is no edge in E that can join two nodes in S, so S is an independent set. □

Claim: Independent Set $\leq_p$ Vertex Cover

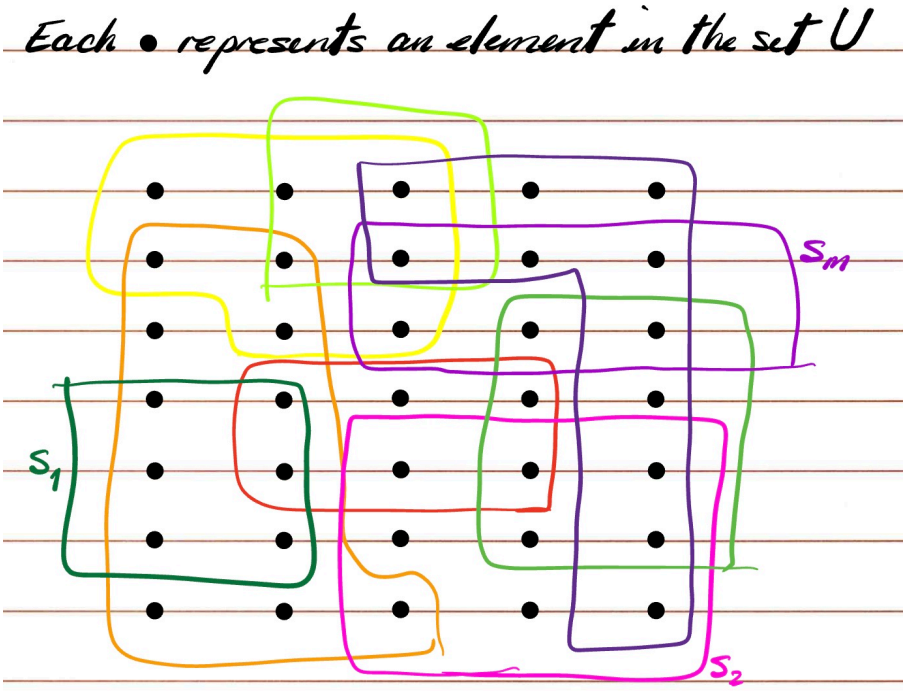
Proof. Having proven that the complement of a vertex cover set is an independent set. Given a blackbox that solves the vertex cover problem, we can decide if G has an independent set of size at least k by asking the blackbox if G has a vertex cover of size at most n-k. □

Claim: Vertex Cover $\leq_p$ Independent Set

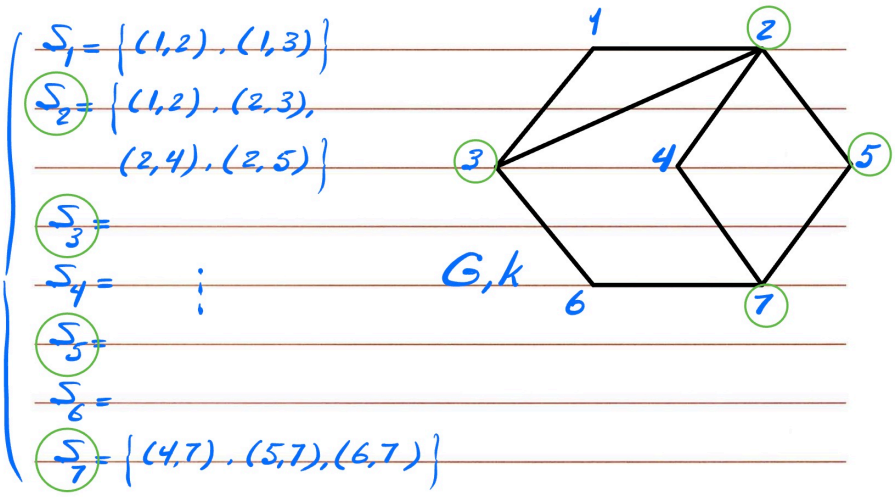
Proof. Having proven that the complement of an independent set is a vertex cover set. Given a blackbox that solves the independent set problem, we can decide if G has a vertex cover of size at most k by asking the blackbox if G has an independent set of size at least n-k. □

3 Set Cover Problem

Set Cover: Given a Set $U = \{1, 2, \cdots, n\}$, a collection of subsets $S_1, S_2, \cdots, S_m$ of U, and a number k. Does there exist a collection of at most k of these subsets whose union is equal to U?



Claim: Vertex Cover $\leq_p$ Set Cover
Given an instance of the vertex cover problem (G, k), we need to construct a set of sets such that there is a vertex cover of size k in G iff there are k sets whose union contains all elements of all sets. Intuitively, we can just create a set for each node in G with all the edges incident to that node as elements.
Then we need to show that G has a vertex cover of size k iff the corresponding set cover instance has k sets whose union contains all edges in G.



Proof. **Only If Direction:** If G has a vertex cover of size k, then the union of the node-corresponding k sets contains all edges in G, i.e. we get a collection of k sets whose union contains all edges in G.
If Direction: If there are k sets from our constructed node-corresponding sets with their union containing all edges in G, then all the incident edges of these sets corresponding k nodes can cover all edges in G, i.e. we get a vertex cover of size k. □

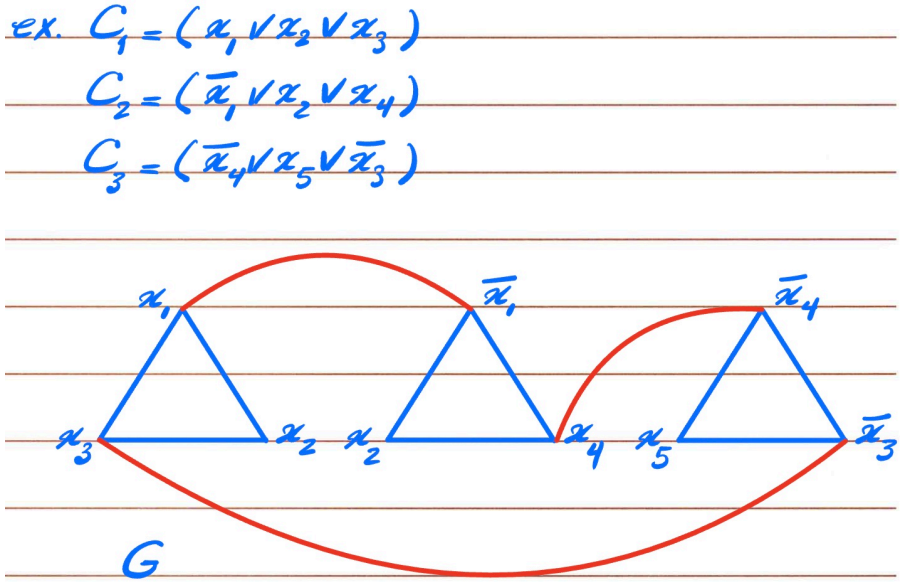
4 SAT (Satisfiability Problem) & 2-SAT & 3-SAT Problem

Clause: Given n Boolean variables $x_1, x_2, \ldots, x_n$, a clause is a disjunction of terms $t_1 \vee t_2 \vee \ldots \vee t_l$, where $t_i \in \{x_1, \ldots, x_n, \bar{x}_1, \ldots, \bar{x}_n\}$
Truth Assignment: an assignment of values 0 or 1 to each $x_i, 1 \leq i \leq n$
Satisfying Truth Assignment: an assignment satisfies a clause truth if it cause $C = 1$. An assignment satisfies truth for a collection of clauses if $C_1 \wedge C_2 \wedge \ldots \wedge C_k = 1$.
E.g. $(x_1 \vee \bar{x}_2) \wedge (\bar{x}_1 \vee \bar{x}_3) \wedge (x_2 \vee \bar{x}_3)$, (1, 1, 1) is not a satisfying truth assignment, (0, 0, 0) and (1, 0, 0) are satisfying truth assignments.
SAT (Satisfiability Problem): Given a collection of clauses $C_1 \ldots C_k$ over a set of variables $X = \{x_1, x_2, \ldots, x_n\}$, is there a Satisfying Truth Assignment?
2-SAT Problem: Given a collection of clauses $C_1 \ldots C_k$ over a set of variables $X = \{x_1, x_2, \ldots, x_n\}$, where each clause has 2 terms, is there a Satisfying Truth Assignment?
2-SAT Problem can be solved in Polynomial even Linear Time through the use of implication graphs and strongly connected components (SCC). Let's construct a Directed Graph with 2n vertices, where i^{th} node means $x_i = 1$ and $(i+n)^{th}$ node means $\bar{x}_i = 1$. Then any clause $(x_i \vee x_j)$ means we have to choose at least one of them to be 1, so we add two edges $(\bar{x}_i \rightarrow x_j)$ and $(\bar{x}_j \rightarrow x_i)$ into the graph. Then Use an algorithm like Tarjan's Algorithm to find Strongly Connected Components (SCC) of the constructed graph. These algorithms work in $O(V+E)$. Then check if there is a variable and its negation in the same SCC, if yes, then there is no satisfying truth assignment, otherwise, there is a satisfying truth assignment.
Finally, we can process the SCCs in Reverse Topological Order to consider dependences and assign true to all literals in an SCC if not already determined by their dependencies.
3-SAT Problem: Given a collection of clauses $C_1 \ldots C_k$ over a set of variables $X = \{x_1, x_2, \ldots, x_n\}$, where each clause has 3 terms, is there a Satisfying Truth Assignment?
Remember a SAT instance is AND of some clauses; Each clause is OR of some literals
2-SAT is P and a special case of SAT, and 3-SAT is NP-Complete and a generalization of SAT.

Proof: SAT $\leq_p$ ***3-SAT.*** The The reduction takes an arbitrary SAT instance ϕ as input, and transforms it to a 3-SAT instance ϕ'. Let C is an arbitrary clause in ϕ:
If C has only one literal l, that is either x_i or $\bar{x}_i$ for some i. Let z_1 and z_2 be two new dummy variables. We replace C by the following four clauses: $(l \vee z_1 \vee z_2) \wedge (l \vee \bar{z}_1 \vee z_2) \wedge (l \vee z_1 \vee \bar{z}_2) \wedge (l \vee \bar{z}_1 \vee \bar{z}_2)$
If C has two literals l_1, l_2: Each of l_1 and l_2 is either a variable x_i or a negated variable $\bar{x}_i$. Let z_1 be a new dummy variable. Replace C by the following two clauses: $(l_1 \vee l_2 \vee z_1) \wedge (l_1 \vee l_2 \vee \bar{z}_1)$
If C has three literals, we do nothing.
If C has more than three literals, Let k > 3 and $C = l_1 \vee l_2 \vee \ldots \vee l_k$, where each l_i either a variable x_i or a negated variable $\bar{x}_i$. Let $z_1, z_2, \ldots, z_{k-3}$ be k-3 dummy variables. We replace C by the following k-2 clauses: $(l_1 \vee l_2 \vee z_1) \wedge (\bar{z}_1 \vee l_3 \vee z_2) \wedge (\bar{z}_2 \vee l_4 \vee z_3) \wedge \ldots \wedge (\bar{z}_{k-3} \vee l_{k-1} \vee l_k)$. **If satisfy truth assignment in the clause of size k**, there must be at least one of the terms in the clause to be true, we can then use the k-3 dummy variables

to set the remaining clauses to be true and therefore find a satisfying truth assignment for the chain; **If we have a satisfying truth assignment in the chain of clauses of size 3**, it must be that at least one of the l_i variables is true (because the dummy variables on their own can set at most k-3 clauses to true). And a satisfying truth assignment in the clause of size k requires only one of l_i to be true. So this is a satisfying truth assignment. □

Proof: 3-SAT $\leq_p$ ***Independent Set.*** Given an instance of 3-SAT with k clauses, we will construct a graph G that has an independent set of size k iff the 3-SAT instance of k clauses is satisfiable.



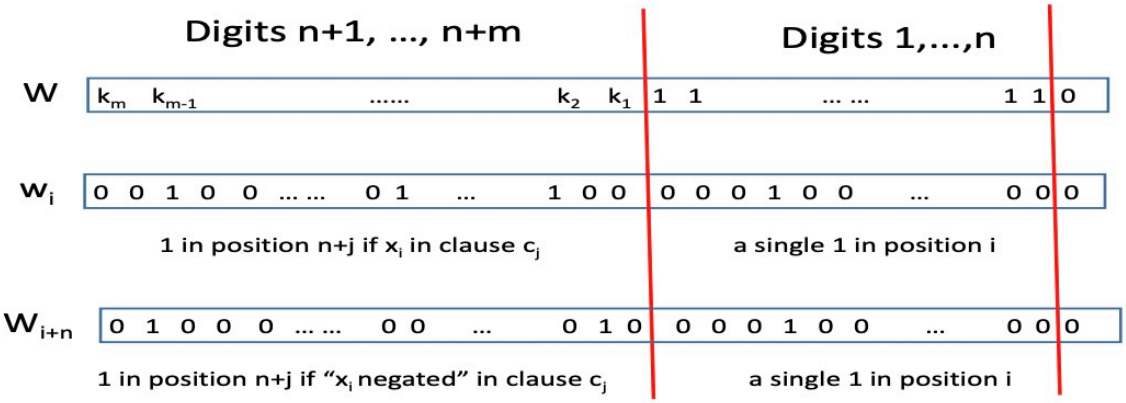
If the 3-SAT instance is satisfiable, then there is at least one node per triangle that evaluates to 1. Let S be a set containing such nodes with **only one from each triangle**. Since x_i and $\bar{x}_i$ can't evaluate to 1 at same time, then nodes in S (one per triangle) can't have an edge between them, i.e. S is an independent set of size k.
If G has an independent set S of size at least k. If x_i appears as a label in S, set $x_i = 1$; If $\bar{x}_i$ appears as a label in S, set $x_i = 0$; If neither x_i nor $\bar{x}_i$ appears as a label in S, then set $x_i = 0/1$. Since S is an independent set, then there is one node set to 1 for each clause. And $x_i, \bar{x}_i$ can't appear in S at the same time, so there is no assignment collision. So the 3-SAT instance is satisfiable. □

5 Subset Sum Problem & Knapsack Problem

Subset Sum: Given n non-negative integers $w_1, \ldots, w_n$ and a target sum W, the question is to decide if there is a subset $I \subset \{1, 2, \ldots, n\}$ such that $\sum_{i \in I} w_i = W$
Knapsack Problem: Given n items with weights $w_1, \ldots, w_n$ and values $v_1, \ldots, v_n$, and a knapsack of capacity W, the goal is to select a subset I to maximize $\sum_{i \in I} v_i$ subject to $\sum_{i \in I} w_i \leq W$.
If we set $v_i = w_i$ for all i, Subset Sum is a special case of the Knapsack problem.
Subset Sum is in NP:
Given a subset I, we can easily check if $\sum_{i \in I} w_i = W$ in $O(nlogW)$ polynomial time.
SAT $\leq_p$ Subset Sum:
Given a SAT instance Φ with n variables $x_1, \ldots, x_n$, m clauses $c_1, \ldots, c_m$, where clause c_j has k_j literals. Then we will define Subset Sum problem using a very large base $B = 2 \max_j\{k_j\}$ with digits $i = 1, \ldots, n$ with $n + m$ digits so we can write numbers as $\sum_{j=0}^{n+m} a_j B^j$, which make sure the additions among our numbers will never cause a carry.
The $1, 2, \ldots, n$ digit will correspond to the n variables $x_1, \ldots, x_n$. For each variable x_i, we set two numbers w_i and w_{i+n} corresponding x_i being true or false. And we set the i^{th} digits of target W, w_i, w_{i+n} to be 1, and set i^{th} digit in all other numbers to be 0, so as to make sure that we use one of w_i or w_{i+n} in any solution.
The $n + 1, n + 2, \ldots, n + m$ digit will correspond to the m clauses with $(n + j)^{th}$ digit to make sure j^{th} clause is satisfied by our setting of the variables.
We will start by define 2n numbers for each of literals $x_i, \bar{x}_i$. The numbers are as follows:

$$w_i = B_i + \sum_{j: x_i \in c_j} B^{n+j} \quad w_{i+n} = B_i + \sum_{j: \bar{x}_i \in c_j} B^{n+j}$$

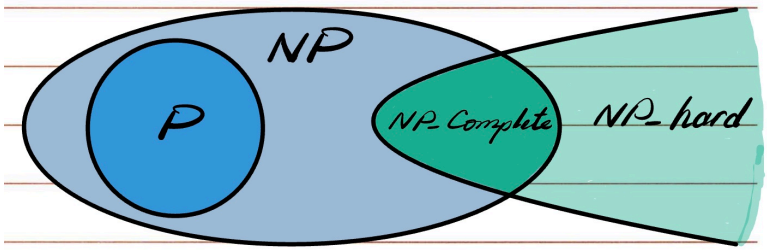
If we get a set of n numbers from the above 2n numbers to satisfy truth assignment for Φ, we can get a sum of the form $\sum_{i=1}^{n} B_i + \sum_{j=1}^{m} b_j B^{n+j}$, where where $b_j \geq 1$ is the number of true literals in clause c_j.



Besides, we need to add $k_j - 1$ copies of B^{n+j} for each clause c_j.
Finally, the target value will be $W = \sum_{i=1}^{n} B_i + \sum_{j=1}^{m} k_j B^{n+j}$. This is our Subset Sum Problem with W and $2n + \sum_{j=1}^{m}(k_j - 1) = 2n - m + \sum_{j=1}^{m} k_j$ numbers defined.

6 Non-deterministic Polynomial Time & Efficient Certification

Efficient Certification: Given a solution, the correctness of the solution can be confirmed in polynomial time.
To show efficient certification, we need **Polynomial Length Certificate & Polynomial Time Certifier**.
3-SAT: Certificate t is an assignment of truth values to variables x_i;
Certifier: Evaluate the clauses if all of them evaluate to 1.
Independent Set: Certificate t is a set of nodes of size at least k in G.
Certifier: Check each edge to make sure no edges have both ends in the set. Check size of set is at least k and there is no repeating nodes.
NP(Non-deterministic Polynomial): the set of all Decision Problems for which there exists an efficient certifier.
Then we can easily show that $P \subseteq NP$, since if a problem is in P, then we can just solve it in polynomial time and return true. However, it is still an open question whether $P = NP$? i.e. Are all problems in NP in P?
NP-Complete: A problem X is in NP and every problem Y in NP is reducible to X in polynomial time, i.e. $Y \leq_p X$.
Such problems are the hardest problems in NP. 3-SAT has been proven to be one of them.
NP-hard: the class of problems that are at least as hard as the NP-complete problems.



Transitivity of Polynomial Time Reduction: If $Z \leq_p Y$ and $Y \leq_p X$, then $Z \leq_p X$.
Then 3-SAT $\leq_p$ Independent Set $\leq_p$ Vertex Cover $\leq_p$ Set Cover.
Set Cover $\leq_p$ 3-SAT since every problem in NP is polynomial reducible to 3-SAT, so by the transitivity property we can get Set Cover $\leq_p$ Vertex Cover.

Basic Strategy to prove a problem is NP-Complete:
1. Prove $X \in NP$ by providing a certificate and a certifier.
2. Choose a known NP-Complete problem Y.
3. Prove $Y \leq_p X$ by providing a polynomial time reduction from Y to X.

7 Set Packing Problem

Given m sets $S_1, S_2, \ldots, S_m$ and an integer k. Our goal is to select k of the m sets such that no selected pair have any elements in common.
Prove NP: Certificate: a subset of k sets (out of the m sets given) that have no elements in common.
Certifier: Can easily check in polynomial time that there are k sets in the certificate and the k sets have no

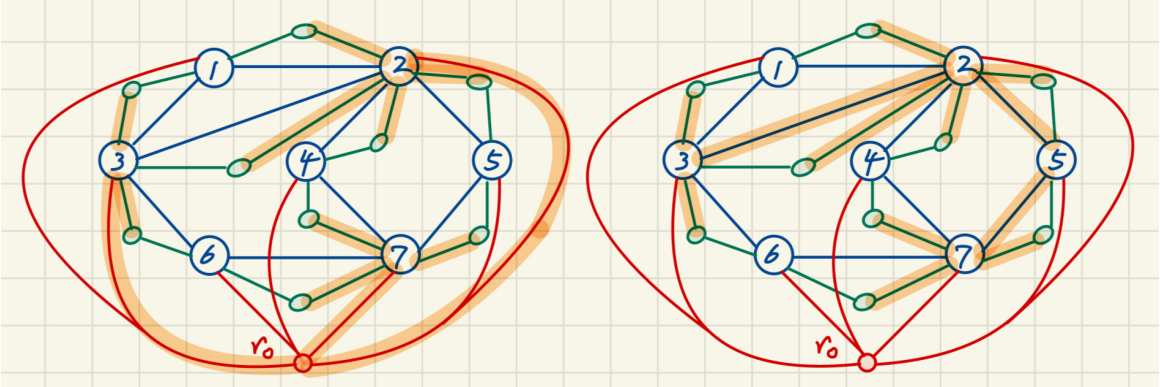
elements in common, which can be easily done in polynomial time.
Claim: Independent Set $\leq_p$ Set Packing.
Start with any instance of Independent Set Problem, we will create a set S_i for each node i with the edges incident on it in G. Then we get a set of Sets $\{S_i\}$, and we can prove that there is an independent set of size k in G iff there are k of the constructed $\{S_i\}$ having no elements in common.

Proof. **If we have an independent set of size k in G**, we can use that to find k sets that have no common elements. The reason is that since the k nodes in G are independent they do not share any edges, therefore the sets corresponding to these k nodes will not have any elements in common.
If we have k sets that have no elements in common, we can find an independent set of size k in G. The reason is that since these sets do not have any elements in common, their corresponding nodes in G will have no edges in common, i.e. they will be independent, and will form an independent set of size k. □

8 Steiner Tree problem

Given an undirected graph $G = (V, E)$ with nonnegative edge costs and whose vertices are partitioned into two sets R and $V - R$. Find a tree $T \subseteq G$ such that $R \subseteq T$ with total cost at most C. That is, the tree that contains every vertex in R (and possibly some in $V - S$) with a total edge cost of at most C.
Prove NP: Certificate: a tree of cost at most C that covers all nodes in R.
Certifier: Can easily run BFS on the tree in polynomial time to check it covers on nodes in R with cost at most C.
Claim: Vertex Cover $\leq_p$ Steiner Tree Problem.
Start with an instance of the Vertex Cover problem (Is there an vertex cover of size at most k in G) and will construct G' such that G has a vertex cover of size at most k iff G' has a Steiner Tree of cost at most m+k.
For each edge e = (u, v) in G, create a new node r_e in G' and connect it to the two endpoints (u, v) of e (Add m nodes in this Process and **all these added nodes belong to Set R, |R| = m**). Add one more node r_0 and connect it to all the original nodes in G. Set the cost of all the edges in G' to be 1.

Proof. **If we have a vertex cover of size k in G, we can produce a Steiner tree of cost m+k in G'**
First add k edges that connect each of k nodes in Vertex Cover to r_0 into the Tree. Then for each node v in Vertex Cover Set, if it adjoints to an edge e in G, then add the edge (v, r_e) into the Tree. (There should be m edges added because the vertex cover set covers all m edges in G). This will result in a tree in G' of cost m+k that covers all nodes in the set R.



If we have a Steiner tree of cost m+k in G' we can produce a vertex cover of size k in G.
Just put all the nodes in the original Graph that are part of the Steiner tree to the Vertex Cover Set. Because m edges in the Tree will be needed to connect all r_e nodes to some other nodes in original G, the other k edges in the Tree will be connecting the nodes in G (May be through node r_0 or Not), corresponding to the k nodes in the Vertex Cover Set. Since these k nodes have direct connections to all nodes r_e, these k nodes will form a vertex cover of size k in G. □