# NP

08/04/24

Roll No: 220002029, 220002063, 220002081

Verbally speaking, NP is the set containing the problems that are possibly verified in polynomial time

Defining mathematically,
Let $L$ be a problem, $L \in$ NP if there exists a certificate $y$ such that $|y| = O(|x|^c)$ (where $x$ is the input and $c$ is a constant) such that there exists a polynomial-time algorithm $A$ that can verify whether $y \in L$.

In simpler terms, to state problem L is NP then one is supposed to find a certificate y whose length is polynomial order of the input length , such that a polynomial time algorithm A verifies if $y \in L$ .

Let us understand the notion of NP problems by considering the following example:

## Vertex Cover (VC)

Let $G(V, E)$ be an unweighted undirected graph. A set $VC \subseteq V$ is said to be a vertex cover if for all $(u, v) \in E$, either $u \in VC$ or $v \in VC$.
In this regard the optimisation problem and decision problem are defined in the following manner

- **Optimization problem:**

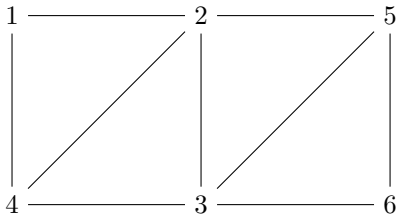
  Let $G(V, E)$ be an unweighted undirected graph. Find the minimum vertex cover, denoted as $VC_{\min}$, such that for all vertex covers $VC$ of $G(V, E)$, the cardinality of $VC_{\min}$ is less than or equal to the cardinality of $VC$, i.e., $|VC_{\min}| \leq |VC|$.

- **Decision problem:**

  $L_{\text{vc}} = \{\langle G, k\rangle \mid$ whether there exists a vertex cover $VC$ of cardinality at most $k\}$

  The output of the Decision problem is a boolean value.

Understanding the vertex cover and decision problem through an example:




Vertex covers for the given graph:

- $VC = \{2, 4, 5, 6\}$
- $VC = \{2, 3, 4, 5\}$

Note that there is no vertex cover for the above graph with cardinality less than 4.

Therefore <G,4> = True; <G,3> = False

**To Show:** $L_{\text{vc}} \in \text{NP}$

**Solution:** $\mathbf{L_{vc}}$ is defined as the decision problem for the above scenario. To demonstrate that $L_{\text{vc}}$ belongs to NP, we need to construct or find a certificate, denoted as $V'$, satisfying the problem conditions.
Since a vertex cover can at most include all the vertices, we can assert that the certificate $V' \subseteq V$ and its cardinality is $O(N)$ (where $N$ is the number of vertices). Since $V' \subseteq V$ and $|V'| \leq N$, we have a certificate with a polynomial order length of the input size.
For each $(u, v) \in E$, we check whether $u$ or $v$ is in $V'$. This verification can be achieved by traversing the edge set. Therefore, the complexity to verify whether $V'$ is a possible solution (vertex cover) is $O(|E|)$, which is of polynomial order. Hence, verifying whether $V'$ is a possible solution to the given problem can be done in polynomial time.
**Conclusion:** Therefore, since there exists a certificate with a polynomial order length (i.e., $|V'| = O(N)$), and it can be verified whether $V'$ is a possible solution in polynomial time, we can conclude that $L_{\text{vc}} \in \text{NP}$.

# NP Hard

**To Show:** $\forall L' \in \text{NP}$, if $L' \leq_p L$ (polynomial reducible), then $L$ is called NP-Hard.

**Solution:**
Polynomial-time reductions provide a formal means for showing that one problem is at least as hard as another, to within a polynomial-time factor. That is, if $L_1 \leq_p L_2$ implies that when $L_1$ is hard then $L_2$ is also hard. A language $L$ is NP-complete if:

1. $L$ belongs to NP, and
2. $L' \leq_p L$ for every $L' \in \text{NP}$.

If a language $L$ satisfies property 2, but not necessarily property 1, we say that $L$ is NP-hard.

The very first problem that is proved to be NP Hard is circuit-satisfiability problem (SAT) problem. Since $L'$ is NP-complete, for all $L'' \in \text{NP}$, we have $L'' \leq_p L'$. We will use the direct proof that $L \leq_P$ CIRCUIT-SAT for every language $L \in \text{NP}$.

$\leq_P$ relation is a transitive relation on languages. That is, if $L_1 \leq_P L_2$ and $L_2 \leq_P L_3$, then $L_1 \leq_P L_3$.

Therefore, $L' \leq_P$ SAT for all $L' \in \text{NP}$ and if SAT $\leq_P L$, using transitivity $L' \leq_P L$. Here the status of $L$ is unknown but the status of $L'$ and SAT are known.

To prove $L$ is NP-complete:

1. Prove $L \in \text{NP}$.
2. Select a known NP-complete language $L'$.
3. Describe an algorithm that computes a function $f$ mapping every instance $x$ of $L'$ to an instance $f(x)$ of $L$.
4. Prove that the function $f$ satisfies $x \in L'$ if and only if $f(x) \in L$.
5. Prove that the algorithm computing $f$ runs in polynomial time.

(Steps 2-5 show that $L$ is NP-hard.) Knowing that the circuit-satisfiability problem is NP-complete now allows us to prove much more easily that other problems are NP-complete. Moreover, as we develop a catalog of known NP-complete problems, we will have more and more choices for languages from which to reduce.

# 3SAT $\leq_P$ Vertex Cover (VC)

**To Show:** To prove that the 3SAT problem is polynomial time reducible to the Vertex Cover problem, we will establish a reduction from 3SAT to Vertex Cover.
**Solution:** Let's recap the key elements:

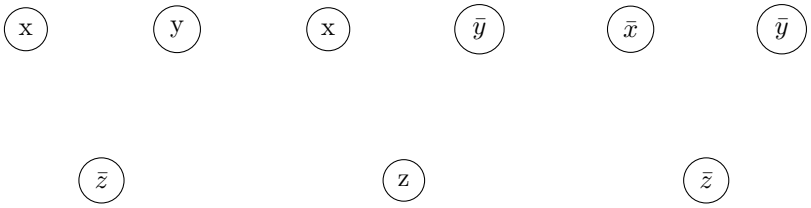
1. **3SAT Problem**: Given a Boolean formula in Conjunctive Normal Form (CNF) where each clause has at most three literals, determine if there exists an assignment of truth values to variables that satisfies the formula.
2. **Vertex Cover Problem**: Given an undirected graph $G = (V, E)$, find the smallest set of vertices such that each edge in $E$ is incident to at least one vertex in the set.

We'll prove that 3SAT is polynomial time reducible to the Vertex Cover problem by showing that for any instance of 3SAT, we can construct an equivalent instance of Vertex Cover in polynomial time.
Let's denote the instance of 3SAT with $f(n, m)$, where $n$ is the number of variables and $m$ is the number of clauses.

**Example**: $(x \vee y \vee \neg z) \wedge (x \vee \neg y \vee z) \wedge (\neg x \vee \neg y \vee \neg z)$

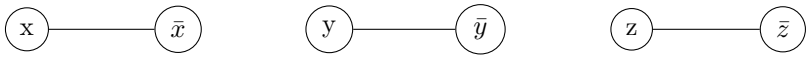
1. **Creation of Vertices for Variables and Clauses**:
   - For each variable $x_i$, we create two vertices: one representing $x_i$ and the other representing its negation, $\neg x_i$. This results in $2n$ vertices for $n$ variables.


   - For each clause $c_j$, we create three vertices, one for each literal in the clause. This results in $3m$ vertices for $m$ clauses.

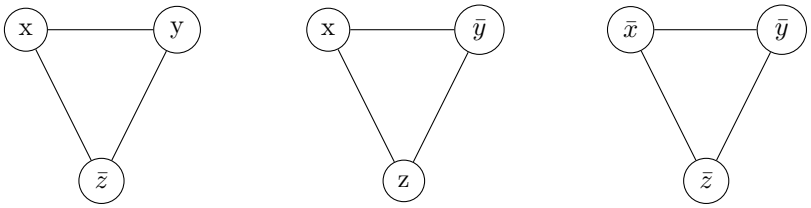



   - The total number of vertices created is $2n + 3m$.

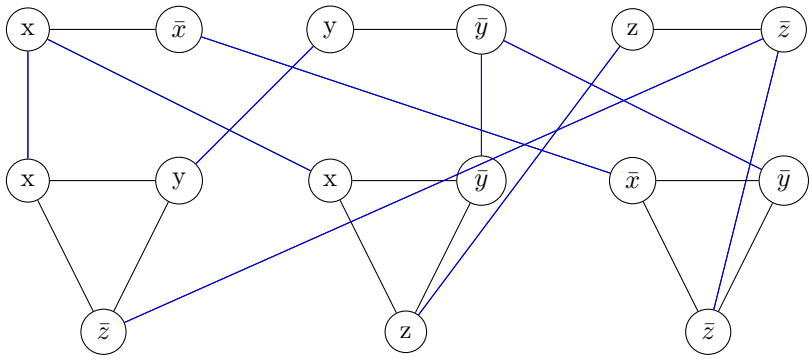
2. **Connection of Complementary Literals**:
   - For each variable $x_i$, we have vertices representing both $x_i$ and $\neg x_i$. These pairs of vertices are connected by an edge to denote that they are complementary literals.
   - This step requires $n$ edges.




3. **Connection of Clause Vertices**:
   - For each clause $c_j$, we connect each literal vertex to the three vertices representing the clause.
   - Since each clause has three literals, this step requires $3m$ connections.



4. **Total Edges**:
   - We connect each variable $x_i$ to corresponding $x_i$ literal in the clause, this step requires $3m$ connections.
   - The total number of edges in the graph is the sum of edges from all three steps: $n + 3m + 3m = n + 6m$.




5. **Time Complexity Analysis**:
   - Creating vertices: $O(n + m)$ time.
   - Connecting complementary literals: $O(n)$ time.
   - Connecting clause vertices: $O(m)$ time.
   - Overall, the construction of the graph takes polynomial time since it involves simple operations like addition and iteration, which are executed a polynomial number of times.
6. **Size of the Graph**:
   - The size of the graph $G$ is polynomial in the size of the input 3SAT instance $f(n, m)$ because both the number of vertices and edges are polynomial functions of $n$ and $m$.

Therefore, we have demonstrated that the construction of the graph $G$ from a 3SAT instance $f(n, m)$ can be done in polynomial time, making the reduction from 3SAT to Vertex Cover polynomial time.

**7. An expression $f$ in 3-SAT form is satisfiable if and only if $G$ has a vertex cover of size $k$ where $k$=(n+2m)**:

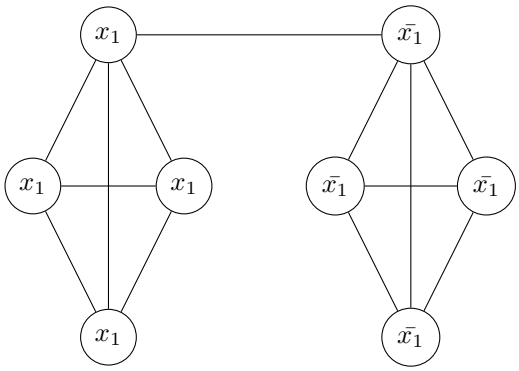
Forward direction proof ( $f$ in 3-SAT form is satisfiable implies a vertex cover of size k exists)

- Let a clause has three literals, say $x_1$, $x_2$, $x_3$.
- If expression f is true, at least on of the three literals in a triplet is true.
- Assume without loss of generality that $x_1$ is true.
- Except $x_1$ take other two vertices in the vertex cover. If more than one literal is true you can decide any one of them not to be included in the vertex cover.
- Take the corresponding $x_1$ in the duplet to be in the vertex cover.
- Repeat the above steps for the remaining clauses.
- Thus for each step we take two literals in a triplet in the vertex cover and a literal from the duplet in the vertex cover.
- Including two vertices from a triplet ensures all edges have been covered in the triplet. Including a vertex from the duplet ensures that the duplets edges have been included. The interconnection edges are also simultaneously included as each edge in interconnection have been taken care by including a vertex either from triplet or from duplet.

- This results in a vertex cover of size n+2m. n vertices, one vertex from each duplet and 2m vertices, 2 vertices from each triplet.
- This proves the forward direction.

Backward direction proof ( A vertex cover of size k exists implies $f$ is satisfiable. )

- We can prove the above statement by providing an example of an unsatisfiable expression and showing that a vertex cover of size k does not exist in that graph.
- Consider the unsatisfiable expression. $(x_1 \vee x_1 \vee x_1) \wedge (\neg x_1 \vee \neg x_1 \vee \neg x_1)$.
- Corresponding graph will be:




- We need a vertex cover of size 1+(2*2)=5
- We need to ensure there are at least two vertices from the first triplet and two from the second triplet. Also we would need both the vertices from the duplet in order to ensure interconnecting edges between duplet and triplet have been considered. Therefore the minimum size of vertex cover would be (2+2+2)=6 (two from first triplet, two from second triplet and both vertices in the duplet ).
- Since, the expression is unsatisfiable, a vertex cover of size 5 does not exist.
- This proves the backward implication

Hence we can say if an expression $f$ is satisfiable there exists a vertex cover of size n+2m is in the corresponding graph $G$.

**Therefore we can say vertex cover is an NP Hard problem. Sice it is both NP and NP Hard it also belongs to the category of "NP COMPLETE".**