NP-Completeness

Story so far

For a wide range of Problems we have developed **Polynomial time** algorithms.

On input of size n, their worst-case running time is $O(n^c)$ for some constant c.

Q: Whether all Problems can be solved in Polynomial time?

Ans. No,

For example "Halting Problem" can not be solved by any Computer, no matter how much time we allow.

There are also problems that can be solved, but not in time $O(n^k)$ for any constant k.

"Generally, we think of Problems that are solvable in Polynomial time algorithms as being tractable, or easy and Problems that require Superpolynomial time as being intractable or hard."

This Class

We are not worried about exact running times.

We only interested whether the running time is Polynomial or Super-Polynomial.

Decision Problem

A decision Problem is a Computational Problem that can be Posed as a YES-NO question of the input values.

Examples

① Input: a natural number n

Q: Is n prime?

② Input: a natural numbers x and y

Q: Does x divide y?

Shortest Path Problem

Input: An undirected graph G and two vertices s and t of G.

Output: Find the shortest s to t path.

Shortest Path Problem (Optimization version)

Input: An undirected graph G and two vertices s and t of G.

Question: Find the shortest s to t path.

Shortest Path Problem: (Decision version)

Input: An undirected graph G and two vertices s and t of G, and an integer k?

Q: Is there a shortest s to t path of length at most k?

Question:

Write optimization version & decision version of Travelling Salesman Problem (TSP).

Next we look at few graph theoretic Problems.

VERTEX COVER

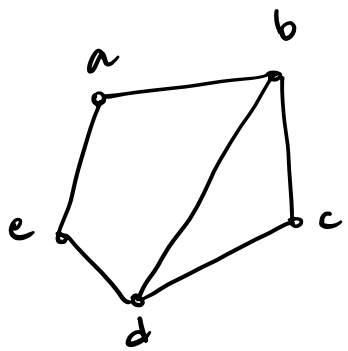
INDEPENDENT SET

CLIQUE

Graph theoretic Problems

Vertex cover:

A Vertex cover of a graph G is a set $X \subseteq V(G)$ that contains at least one endpoint of every edge.



The size of a Vertex cover is the number of vertices it contains.

The Vertex cover Problem is to find a Vertex Cover minimum size in a graph G.

MVC — Minimum (Size) VertexCover

Warmup examples:

- Complete graphs
- Path (P_n)
- Cycle (C_n)

VERTEX COVER PROBLEM (Optimization Version)

Input: A graph G (undirected)

Q: Find a vertex cover of minimum size in G.

Brute-force Algorithm for V.C:

```
VC(G)
1.   MVC = |V(G)|
2.   For each X ⊆ V(G)
3.       For all e = uv ∈ E(G)
4.           if both u and v ∉ E(G)
5.               return False
6.       if |X| < MVC
7.           MVC = |X|
8.   return MVC
```

Running time : $O(2^n m)$

VERTEX COVER (Decision version)

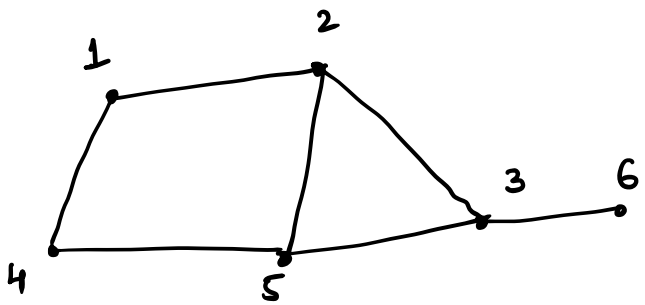
Input: A graph G and an integer k

Question: Does G contain a vertex cover of size at most k?

Independent set :

In a graph $G=(V,E)$, we say subset $I \subseteq V$ is independent if no two vertices in I are adjacent.

The **independent set Problem** is to find the largest independent set in a graph G.



The largest size independent set $= \{2, 4, 6\}$

Size $= 3$.

Independent set (Decision version)

Input: A graph G and an integer k

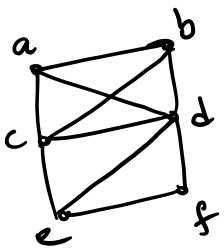
Question: Does G contain an independent set of size at least k?

CLIQUE in a graph:

A subset $C \subseteq V(G)$ is called a clique in a graph G if for all $u, v \in C$, $uv \in E(G)$ (i.e., C is a complete subgraph of G).

The **CLIQUE Problem** is to find the size of a largest clique in a given graph (optimization version).

Ex:-



$C = \{a, b, c, d\}$ is a clique of size 4 (also largest)

CLIQUE Problem [Decision version]

Input: Graph G, and an integer k

Question: Decide whether G has a clique of size k?

CLIQUE

Decision version vs Optimization version

Q: Which one is easier to solve?

Given a graph G,

Q: ① Is there a relation between Vertex Cover and Independent Sets in G?

True | False :

① If X is a VertexCover of G, then

$I = V(G) - X$ is an independent set of G

② If I is an independent set of G then

$X = V(G) - I$ is a VertexCover of G.

Theorem:

X is a minimum size vertex cover of G

if and only if

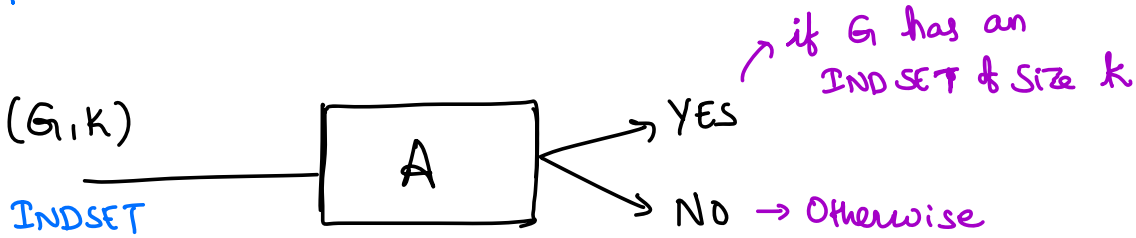
$I = V(G) - X$ is a maximum size independent set in G.

In otherwords.

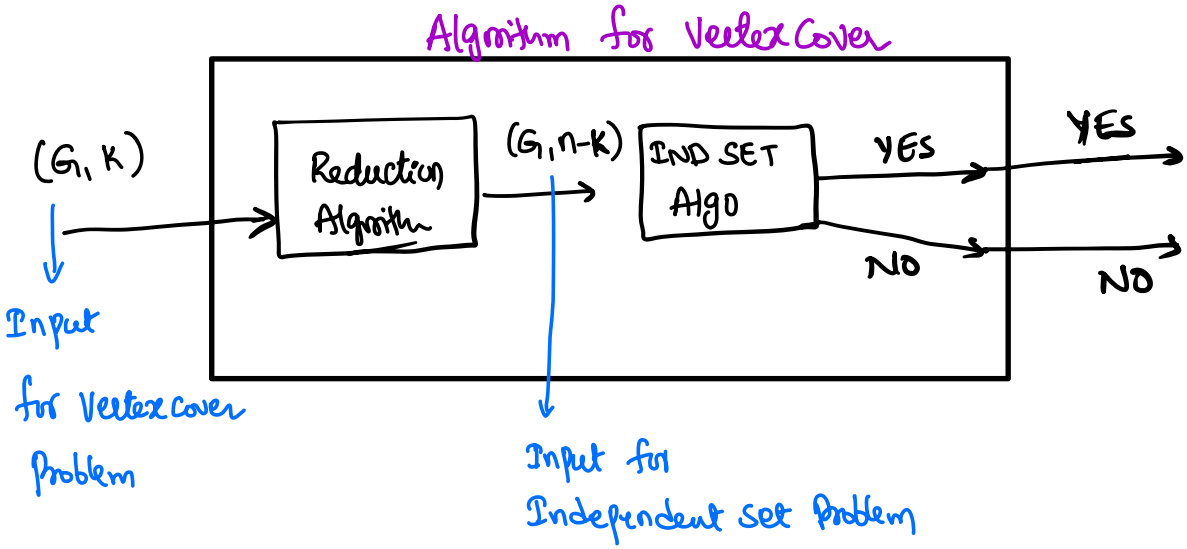
Size of MVC + Size of Maximum ind set

= Number of vertices.

Suppose you have an algorithm A that solves Independent Set Problem



Q: Can we use A to solve VERTEX COVER Problem.


Algorithm for Vertex Cover
(G, K)
Reduction Algorithm
(G, n-K)
IND SET Algo
YES
YES
NO
NO
Input for Vertex cover Problem
Input for Independent set Problem

Given a graph G,

Q: ① Is there a relation between ~~Vertex Cover~~ Clique and Independent Set in G?

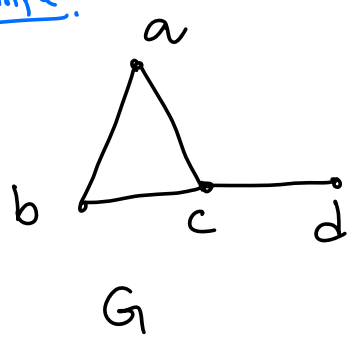
Complement of a graph:

The Complement of G is denoted by $\overline{G}$, defined as follows.

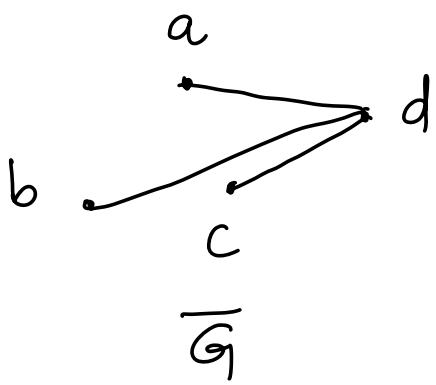
$$V(\overline{G}) = V(G)$$

$$E(\overline{G}) = \{ uv \mid u, v \in V(G) \& uv \notin E(G) \}$$

Example:



G



$\overline{G}$

True | False :

If I is an independent set in G

then I is a clique in $\overline{G}$

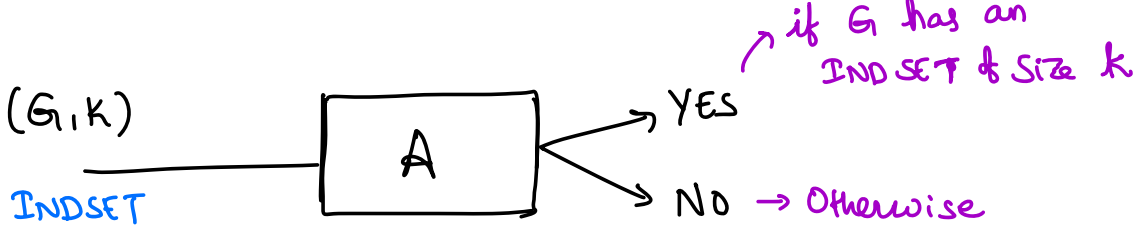
Theorem:

I is a maximum size independent set in G

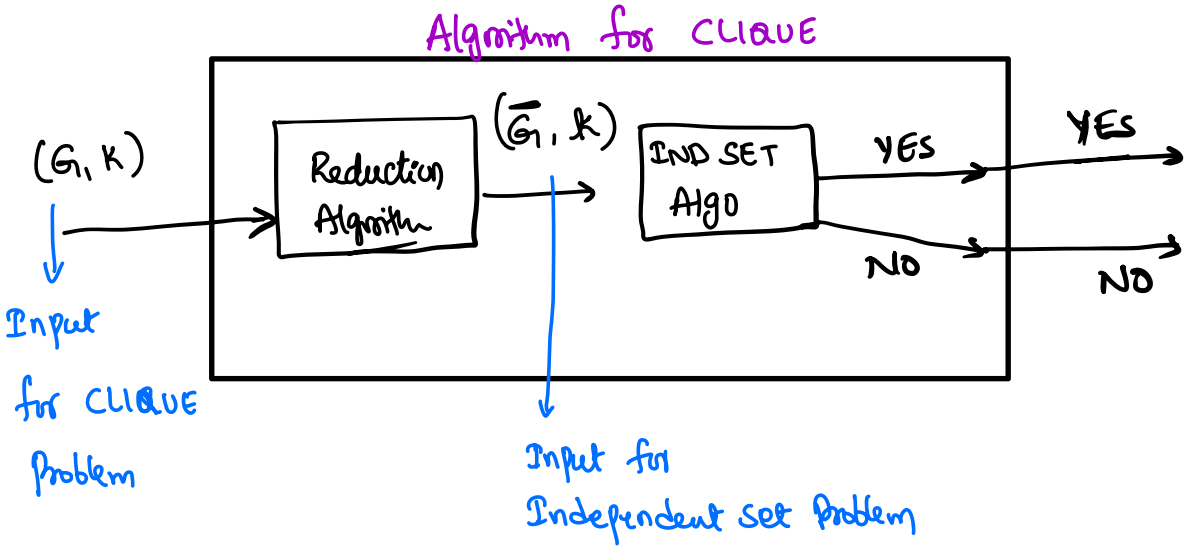
if and only if

I is a maximum size clique of $\overline{G}$.

Suppose you have an algorithm A that solves Independent Set Problem



Q: Can we use A to solve ~~VERTEX COVER~~ CLIQUE Problem.


Algorithm for CLIQUE
(G, K)
Reduction Algorithm
(Ḡ, k)
IND SET Algo
YES
YES
NO
NO
Input for CLIQUE Problem
Input for Independent set Problem

Summary So far

VERTEX COVER
IND SET
CLIQUE

} If you have a Polynomial time algorithm that solves one of them then we can also solve the other two.

We do not know of Polynomial-time algorithms for these three Problems (even today).

We cannot Prove that no Polynomial-time algorithm exists.

Same is true for many other Problems like

- Hamiltonian Path
- TSP
- 3-SAT
- longest Path
- etc

The good thing is :

Researchers working in this area have found a large class of Problems with the following Property.

"A Polynomial time algorithm for any one of them would imply the existence of a Polynomial time algorithm for all of them"

[These are NP-Complete Problems, formally we define it later]

Complexity classes P and NP

P = Set of all decision Problems that can be solved in Polynomial time.

(YES/NO type questions)

Eg: Sorting, Searching, MST etc

NP = Set of all decision Problems for which a solution can be verified in Polynomial time.

Eg: Vertex Cover, Hamiltonian cycle, SAT etc

Clearly $P \subseteq NP$

(Q) Is P equal to NP? (Unsolved Problem)

belief: $P \neq NP$

Means there are Problems in NP that are harder to Compute than to verify: they could not be solved in Polynomial time, but the answer could be verified in Polynomial time.

NP-Complete :

A Problem Π is NP-Complete if

"A Polynomial time algorithm for Π would imply the existence of a Polynomial time algorithm for all Problems in NP".

More on this topic will be in

next Part – Algorithms – II