Standard NP-complete problem

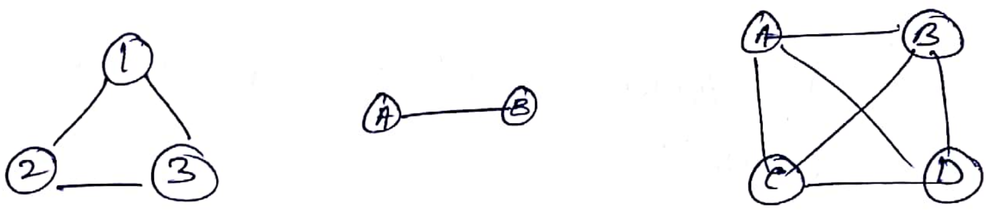
① Clique Decision problem

complete Graph : If a vertex is connected to all other vertices in a graph, then it is called a complete Graph.

property: for "n" vertices,

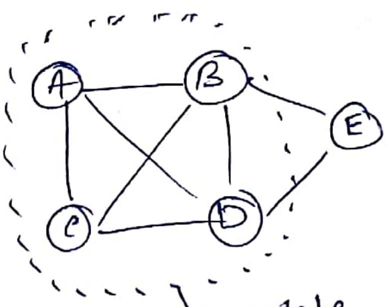
Total no. of edges in a complete Graph $= \dfrac{n * (n-1)}{2}$

Example of complete graph.



clique : It is a sub-graph of a graph, which is complete.

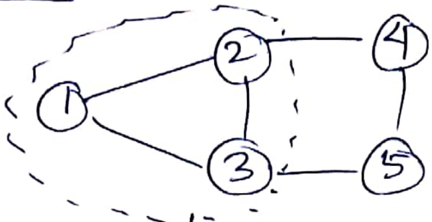
A clique, c, in an undirected graph $G(V,E)$ is a subset of the vertices $(c \subseteq V)$ such that every two distinct vertices are adjacent.



complete

Here the graph is not complete, where as the subgraph is complete.
Hence, the subgraph is a clique.
(size-4)

Example 2 :



clique. (size - 3)

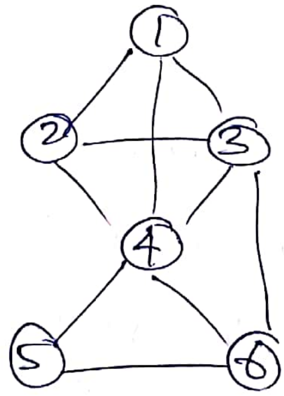
		Size(K)
$C_1 = (1, 2, 3)$	→	3
$C_2 = (1, 2)$	—	2
$C_3 = (1, 3)$	—	2
⋮		

clique Decision problem (CDP)

A decision problem to test whether a given set of vertices is a clique or not.

Example

Is there a clique of size 3 in the following graph?



Answer: yes.

clique optimization problem

An optimization problem to find the maximum size of the clique in a graph.

CDP is NP-complete.

To prove, CDP is NP-complete; we need to show

① CDP ∈ NP

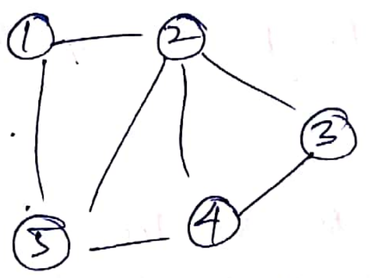
② CDP ∈ NP-Hard.

① CDP ∈ NP (CDP is NP)

By using the adjacency matrix representation of the graph, we can verify the subgraph is complete or not in polynomial time.

So, CDP ∈ NP.

Example



Adjacency matrix Representation

A	1	2	3	4	5
1	0	1	0	0	1
2	1	0	1	1	1
3	0	1	0	1	0
4	0	1	1	0	1
5	1	1	0	1	0

To test C is a clique or not

```
flag = 1
for each edge pair (u, v) ∈ vertex set of C
{
    if A[u, v] == 0      (not adjacent)
    {
        flag = 0
        break
    }
}
if flag == 1
    returns True
else
    returns False.
```

This algo. is a polynomial time algo.

Hence,

CDP ∈ NP.

② CDP $\in$ NP-Hard

To show CDP is NP-hard, we need to take an example of known NP-Complete problem and convert that to a clique decision problem (CDP.) in polynomial time.

e.g. SAT $\propto$ CDP

Hence, CDP $\in$ NP-Hard.

Example

Let x_1, x_2, x_3 are 3-variables of a boolean Formula/expression.

Literal

$$F = \underbrace{(x_1 \vee x_3)}_{\substack{\text{clause 1} \\ (c_1)}} \wedge \underbrace{(\bar{x}_1 \vee \bar{x}_3)}_{\substack{\text{clause 2} \\ (c_2)}} \wedge \underbrace{(x_1 \vee x_2)}_{\substack{\text{clause 3} \\ (c_3)}}$$

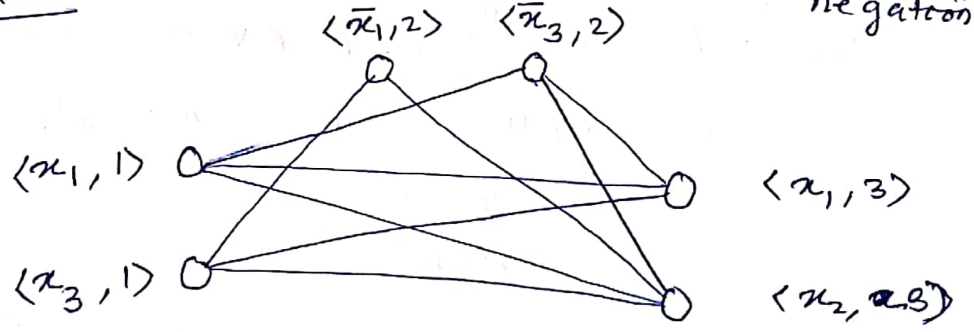
$\boxed{F = \bigwedge_{i=1}^{k} c_i}$

Step 1: Represent the formula F in a Graph. G.

Rules for connecting edges

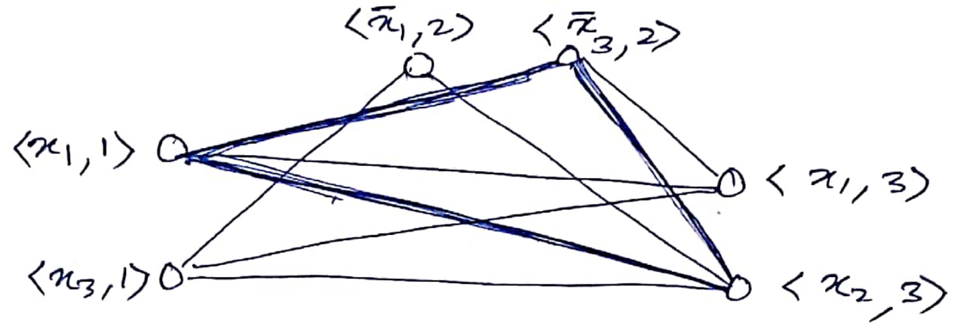
Rule 1: Do not connect the vertices of same clause in the formula.

Rule 2: Do not connect a literal with its negation.



G(V, E)

Step 2: Find cliques with size k.
where $k = \#$ of clauses.



$x_1, x_2, \bar{x}_3$ is a clique having size 3.
$(k = 3)$

Step 3: show that solving the clique problem also solves the SAT problem.

$x_1, \; x_2, \; \bar{x}_3$

$1 \quad 1 \quad 1$

$$F = (x_1 \vee x_3) \wedge (\bar{x}_1 \vee \bar{x}_3) \wedge (x_1 \vee x_2)$$
$$= (1 \vee 0) \wedge (0 \vee 1) \wedge (1 \vee 1)$$
$$= 1 \wedge 1 \wedge 1$$
$$= 1.$$

So, we have proved that solving CDP also solves SAT problem.

~~Hence~~ ~~CDP~~ $SAT \underset{\text{Reduces to}}{\propto} CDP$.

Hence, CDP is NP-Hard.

Conclusion: CDP $\in$ NP and CDP $\in$ NP-Hard.
Hence, CDP is NP-complete.