



# Clique Cover Problem

Reduction and Hardness proof

**Group No: 10**

**Student ID:**

1505002

1505044

1505057

1505097

1505101



# Clique Cover Problem

**Input:** An undirected graph  $G(V,E)$  and an integer  $K$ .

**Output:** True if the vertices of  $G$  can be partitioned into  $K$  sets  $S_i$ , whenever two vertices in the same sets  $S_i$  are adjacent.  $S_i$  do not need to be disjoint, they can be non disjoint. But we can make them disjoint by putting common vertices in only one set without any problem. Thus we can think  $S_i$  are disjoint.

**Note:** There is also **edge clique cover** problem but we are only interested in **vertex clique cover**. So if we say clique cover, we are indicating vertex clique cover.

# Complete Graph

A complete graph is a simple undirected graph in which every pair of distinct vertices is connected by a unique edge.

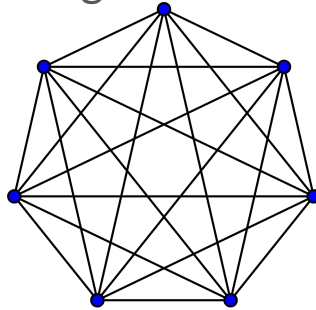
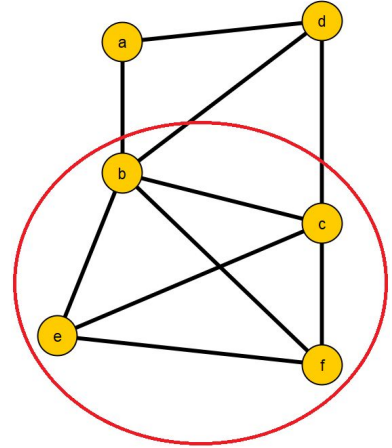
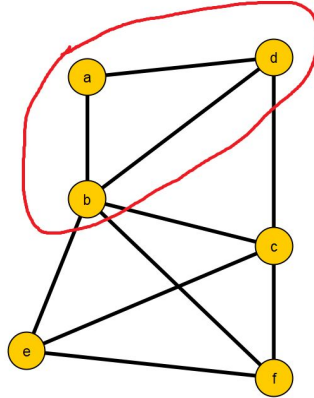
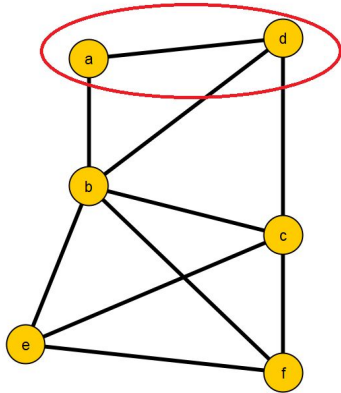
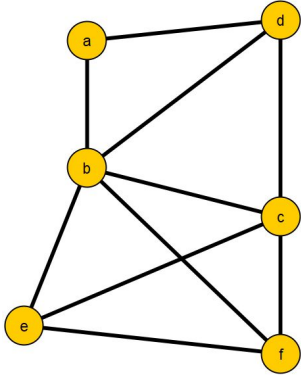


Fig : Complete Graph( $k_7$ )

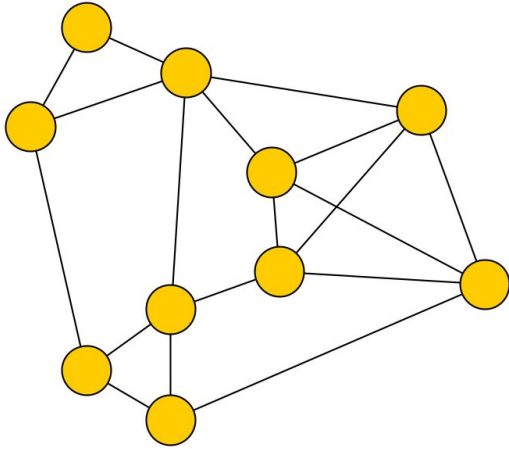


# Clique

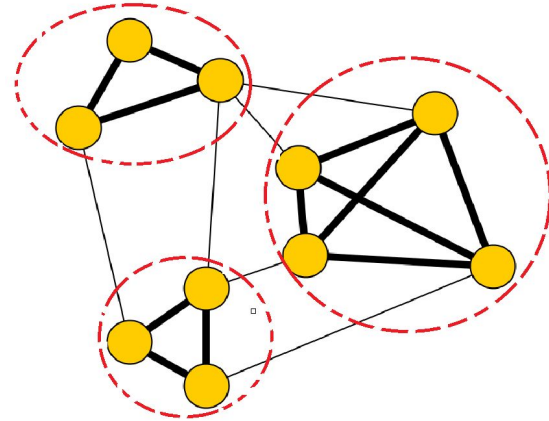
A clique of a graph  $G(V,E)$  is a complete subgraph of  $G$ .



# Clique Cover Problem



Given a graph  $G(V,E)$  and integer  $k = 3$ .

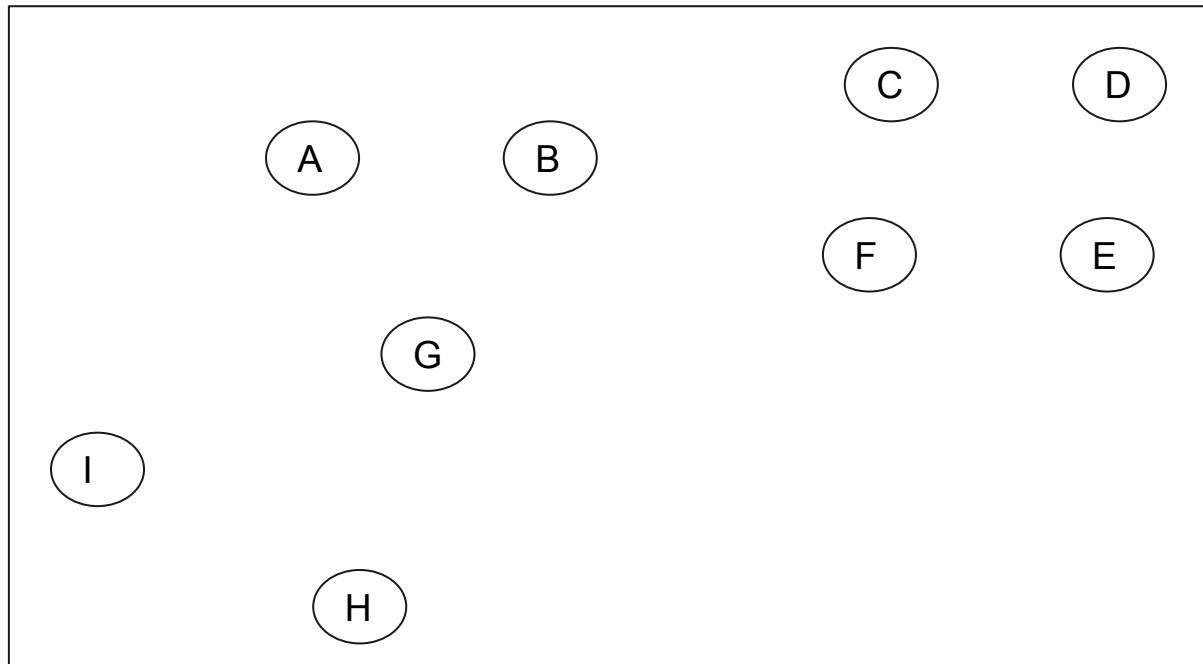


True because the vertices of  $G$  can be partitioned into 3 sets  $S_i$ , where two vertices in the same sets  $S_i$  are adjacent.

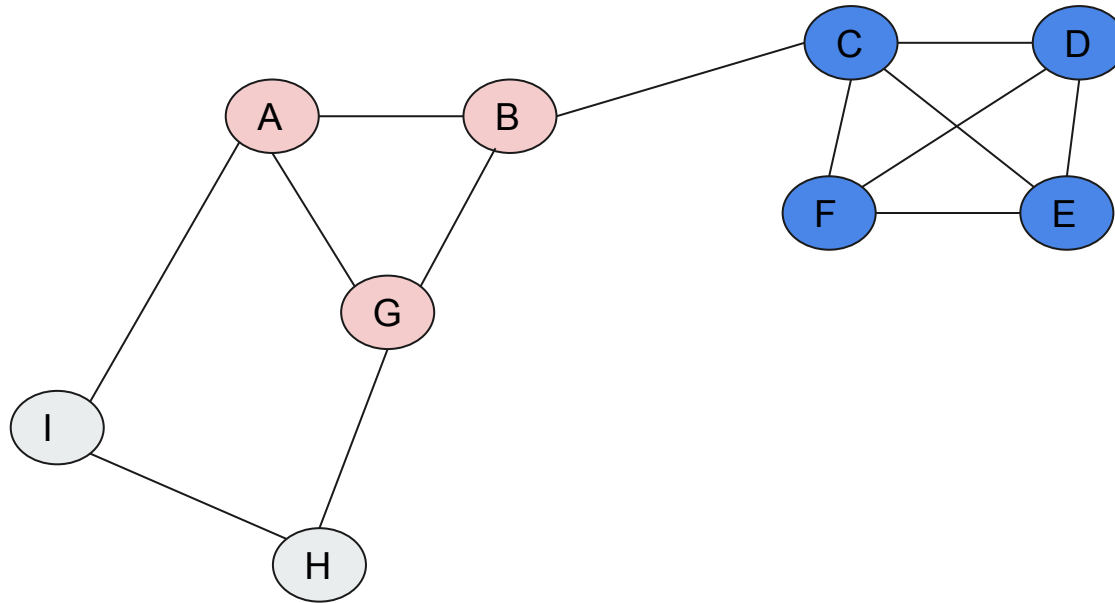


# Why do we need to study “Clique Cover Problem”

# Let's look at a scenario...

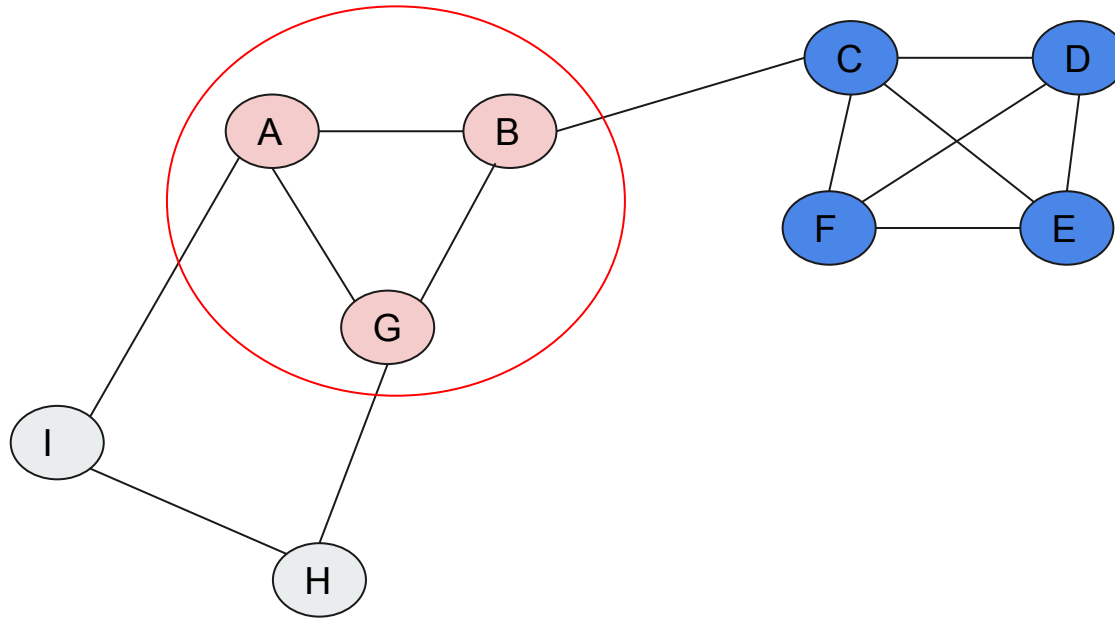


Let's look at a scenario...

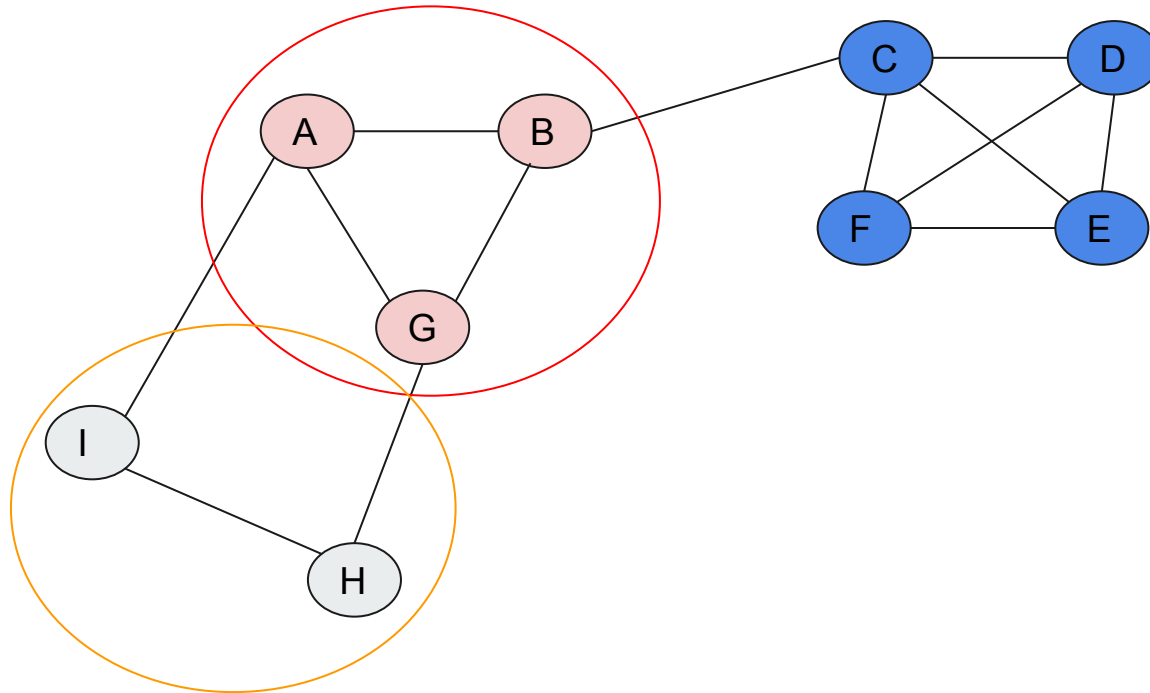




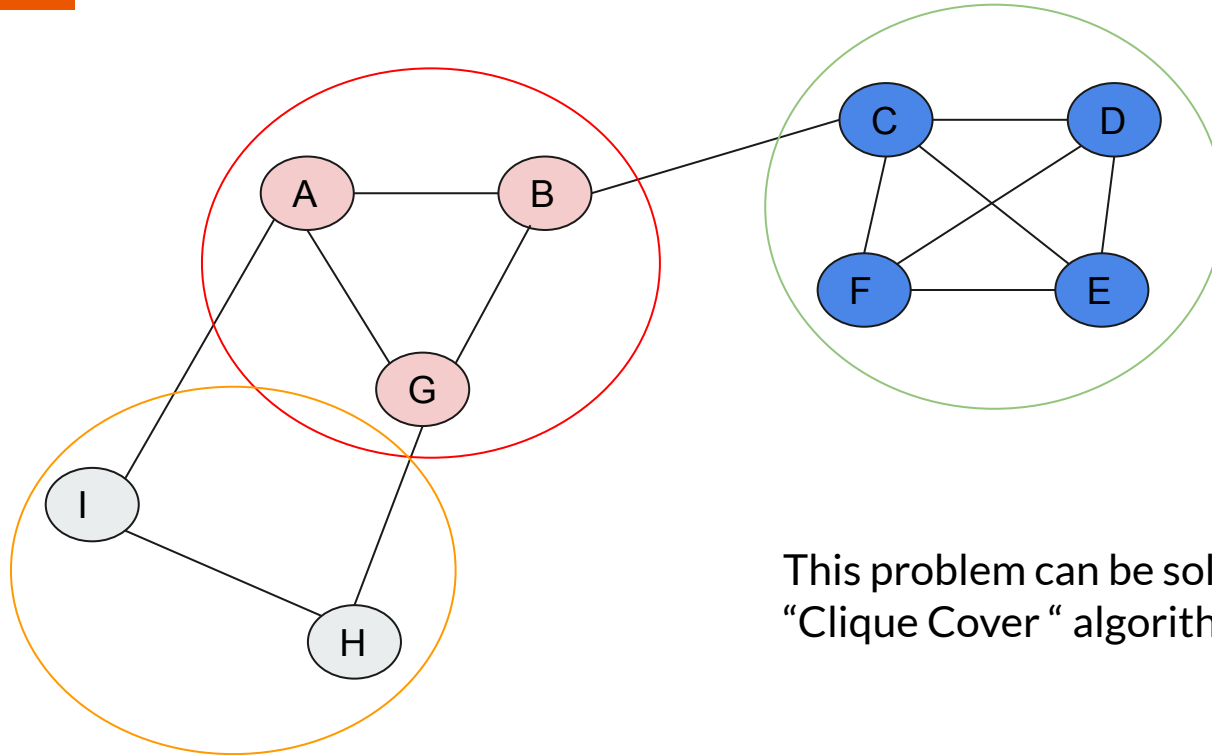
Let's look at a scenario...



Let's look at a scenario...



# Let's look at a scenario...



This problem can be solved using  
“Clique Cover” algorithm.



## Other Applications

- *DNA molecular solution problem, data partitioning problem* in embedded processor-based systems (memory chips), *image processing problems* etc.
- Applications of the vertex clique cover problem arise in *network security, scheduling* and *VLSI design*.
- Algorithms for clique cover can also be used to solve the closely related problem of finding a maximum clique, which has a range of applications in biology, such as identifying related protein sequences.



## Reduction from Clique Cover



## Hardness Status of Clique Cover Problem

The clique cover problem in computational complexity theory is the algorithmic problem of finding a minimum clique cover, or (rephrased as a decision problem) finding a clique cover whose number of cliques is below a given threshold. Finding a minimum clique cover is NP-hard, and its decision version is NP-complete. It was one of Richard Karp's original 21 problems shown NP-complete in his 1972 paper "Reducibility Among Combinatorial Problems".

The equivalence between clique covers and coloring is a reduction that can be used to prove the NP-completeness of the clique cover problem from the known NP-completeness of graph coloring.



## Proof of NP-completeness

- To prove that clique cover is **NP-complete**, first we prove that it is in **NP**.
- We are given a graph  $G(V, E)$  and the clique cover of the graph of size  $k$  that is  $k$  subsets of  $V$ . We first check whether each such subset form a complete graph that is any two vertices of a set are neighbors. If not the answer is no. Otherwise we again check if union of all the sets is equal to  $V$ . If not the answer is no otherwise yes.
- Clearly, this takes at most  $(O(n^2))$  where  $n$  is the number of vertices . So, the clique cover problem is clearly in **NP**.



## Proof of NP-completeness

- The next thing we do is show that it is NP-hard.
- To do so, we show that the **clique cover** set problem is at least as hard as the **k-coloring** problem.

Claim

$$K\text{-coloring} \leq_p \text{Clique Cover}$$





## Reduction from K-coloring

- If a graph is **k-colorable**, then it can be partitioned into **k independent sets** (one for each color class).
- Then we just exploit the normal reduction between **Independent Set** and **Clique Cover** by taking the complement graph (we swap edges for non-edges and vice versa), so any independent set becomes a clique.

Claim

$K\text{-coloring} \leq_p K\text{-Independent Set}$

$K\text{-Independent Set} \leq_p \text{Clique Cover}$

## An Example...

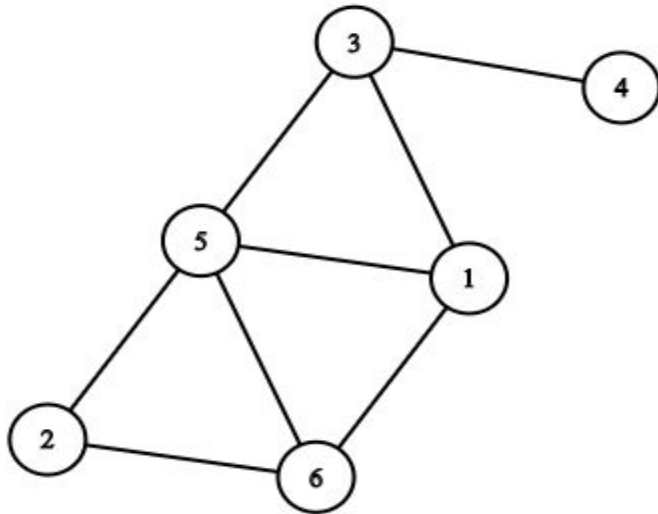


Figure:  $G$

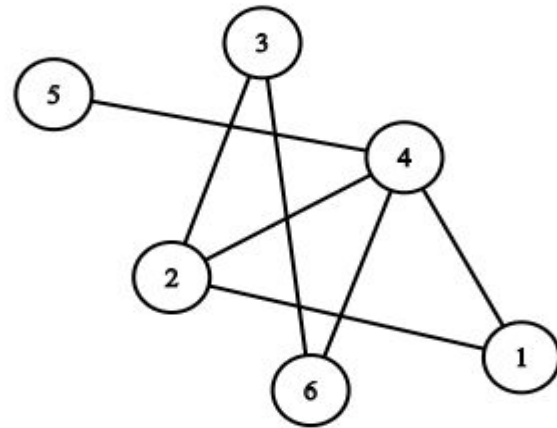


Figure:  $\overline{G}$



## Reduction from K-coloring

- So if  $G(V,E)$  is  $k$ -colourable, it can be partitioned into  $k$  independent sets.
- Hence  $\overline{G(V,E)}$  (complement of  $G(V,E)$ ) can be partitioned into (i.e. covered by)  $k$  cliques.
- Conversely if  $\overline{G(V,E)}$  can be covered by  $k$  cliques,  $G(V,E)$  has a partition into  $k$  independent sets, and hence is  $k$ -colourable.

## An Example...

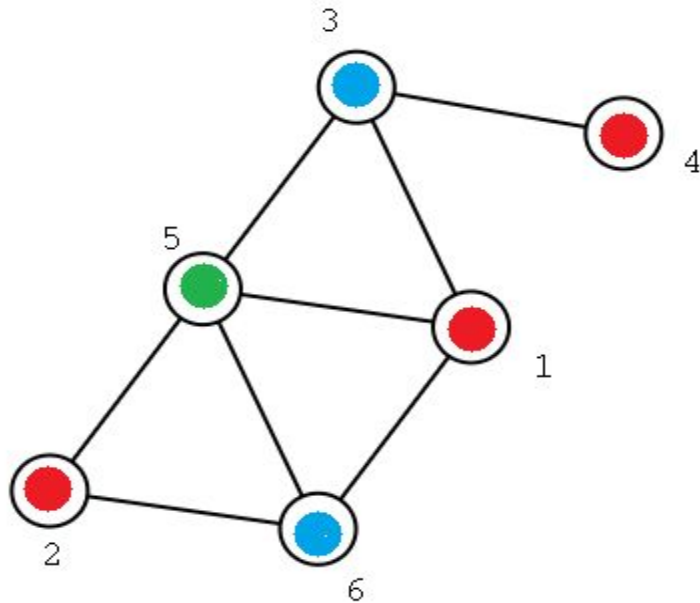


Figure:  $G$

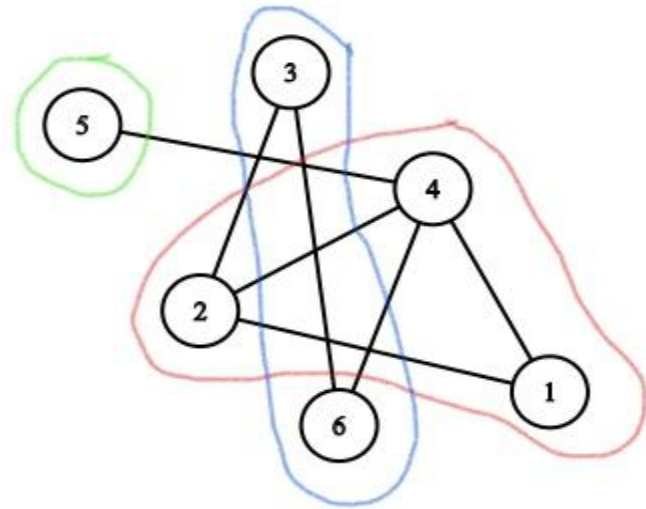


Figure:  $\overline{G}$



# Clique Cover is NP-complete!

Theorem

The Clique Cover problem is NP-complete.



## Transitivity of Reducibility

If A is reducible to B and B is reducible to C, then A is reducible to C.

E.g. Independent Set  $\leq_p$  Vertex Cover and Vertex Cover  $\leq_p$  Hamiltonian Cycle, then  
Independent Set  $\leq_p$  Hamiltonian Cycle .

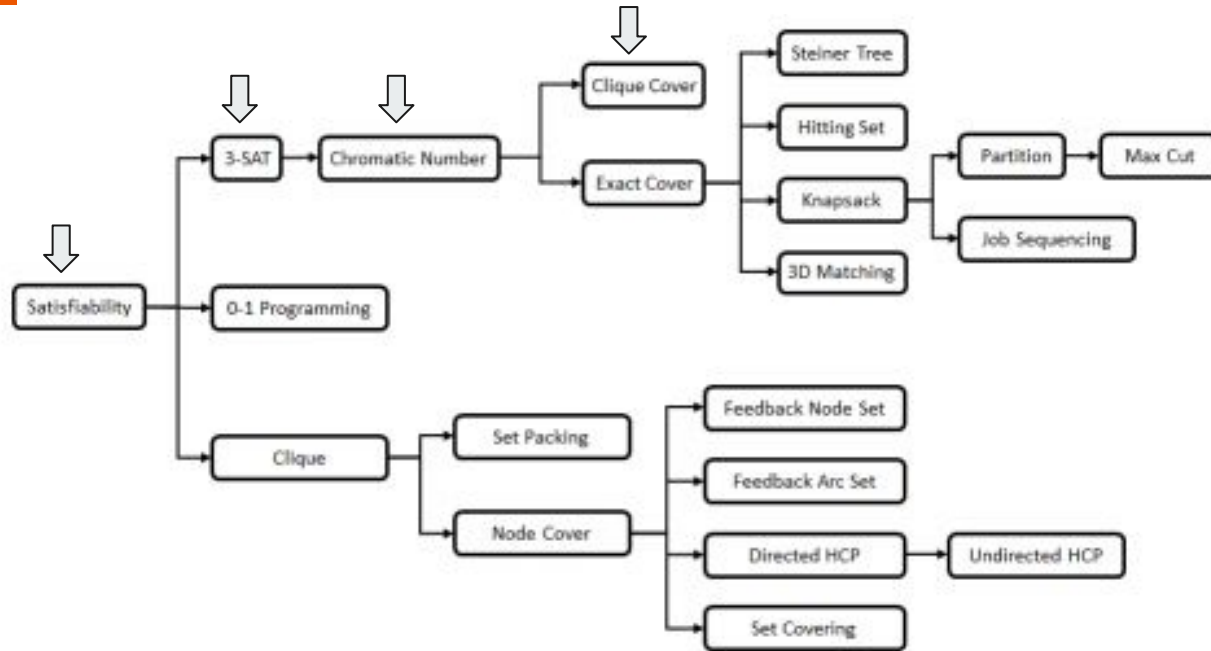



Fig : A tree showing the 21 NP-complete problems identified by Karp, where the edges correspond to individual reductions.



According to the “**Transitivity of Reducibility**” theorem ,  $3\text{-SAT} \leq_p \text{Chromatic Problem} \leq_p \text{Clique Cover Problem}$ .

Unfortunately, we could not find any problem that **Clique Cover Problem** reduces to.

But there is a related *NP-complete* problem named **Clique Edge Cover** problem . It concerns partitioning the edges of a graph, rather than the vertices, into subgraphs induced by cliques.





# Clique Cover and It's Variation

## **Clique Cover Decision Problem:**

Given a graph  $G(V, E)$  and a number  $K$ , we need to answer yes or no if we can partition  $V(G)$  in  $K$  cliques

## **Minimum Clique Cover Problem:**

Given a graph  $G(V, E)$ , we need to output the smallest number for which clique cover exists. This number is called the clique cover number.

Both of these problem are equivalent. That means if we solve one of the problems we can easily construct a solution for the other problem in polynomial time from that solution.



## Solving minimum clique cover solves the clique decision problem

- Suppose, clique cover number is  $X$ , then for each integer between  $X$  and  $|V|$  (inclusive of both), answer for clique decision is yes. Otherwise answer is no.
- It is obvious that answer for  $K = X$  answer is yes.
- Now we can extract a vertex from a clique of size  $p$  ( $p > 1$ ) and make two cliques of size 1 and  $p-1$  respectively. Thus making the answer for  $K = X + 1$  yes.
- We can continue in this way making  $K = X+2$  answer yes and so on till  $K = |V|$ . When  $K = |V|$  we are left with all one length clique so no further reduction is possible.



## Solving clique decision problem solves minimum clique cover

- Suppose, we can solve the clique decision problem with program P.
- Now iterate through  $K = 1$  to  $|V|$  and ask P for each K if answer is yes or no. While iterating, the first number for which answer is yes, is the clique cover number. Thus solving the minimum clique cover problem. The complexity is  $O(|V|)$ \*complexity of clique decision problem.
- Alternatively we can Binary Search in range  $[1, |V|]$  to find the first integer which is yes. We can do this because the sequence is like this: No, No ..... No, Yes, ..... Yes. This gives us significantly lower complexity than first approach. The complexity is  $O(\log|V|)$ \*complexity of clique decision problem.



## **Special Classes of Graph for Which Clique Decision or Minimum Clique Cover is Solvable in Polynomial Time**

- This class includes Path Graph, Cycle Graph, Complete Graph, Triangle Free Graph. Details are discussed in next slide.



## When $K = |V|$

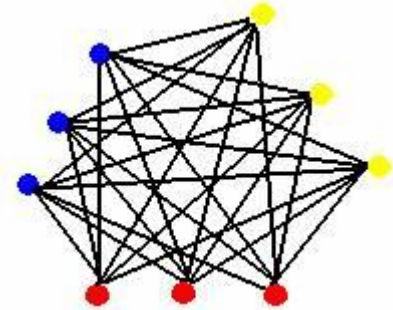
- In this case answer for clique decision problem is always yes for any graph.
- How? We can just make each vertex a clique.



# Complete Graph $K_n$

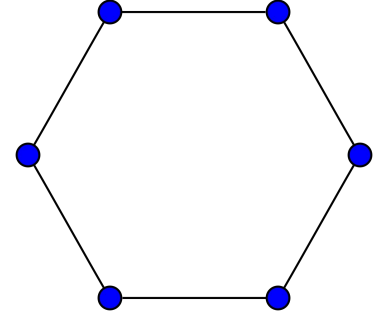
- For complete graph, clique cover number is 1
- It is obvious from first point, for clique decision problem for all  $K = 1$  to  $|V|$  answer is yes.

# Complete K partite graph $K_{n_1, n_2 \dots n_p}$



- Clique cover number is  $\max(n_1, n_2, \dots, n_p)$
- We can easily take one vertex from each partition to make a clique. If there is no vertex left in a partition to take, no need to take any vertex from there. Thus creating a clique cover of size  $\max(n_1, n_2, \dots, n_p)$ . Answer is no less than this because we can not put two vertex from same partition in same clique. Thus giving clique cover number  $\max(n_1, n_2, \dots, n_p)$ .

## Cycle graph $C_n$ or Path Graph $P_n$

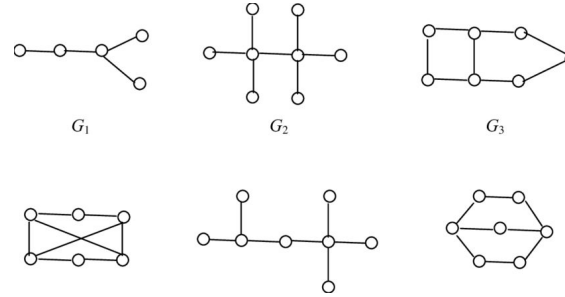


- For a cycle/path graph clique cover number is  $n/2$  if  $n$  is even,  $(n+1)/2$  if  $n$  is odd.
- It is obvious that no clique can have more than 2 vertex in a cycle/path graph. Thus we can take maximum two sequential vertex in a clique. So the clique cover number can not be greater than  $\text{ceil}(n)/2$ .
- Now we have clique cover number  $n/2$  if  $n$  is even. For  $n$  is odd we have a clique of size 1 thus giving a clique cover number  $(n+1)/2$  if  $n$  is odd.





# Triangle Free Graph



- It is obvious that in triangle free graph maximum clique size is two. So we make as much as two size clique possible. How we do that? Maximum matching.
- For vertices not in maximum matching we make each of them one sized clique
- Thus we have a clique cover number =  $|V|$  - maximum matching
- Note that Triangle Free graph is the general case of cycle and path graph.