Reductions

Reduction: Problem A reduces to Problem B if, given a "black box" (subroutine) for B, one can solve A using a (polynomial) number of calls to the subroutine.

Trivial Example:

- B is addition – $B(x, y) = x + y$
- A multiplication by 3.
- A reduces to B because we can multiply by 3 : $A(z) = B(z, B(z, z))$.

Reductions

Reduction: Problem A **reduces to** Problem B if, given a "black box" (subroutine) for B, one can solve A using a (polynomial) number of calls to the subroutine.

Trivial Example:

- B is addition – $B(x, y) = x + y$
- A multiplication by 3.
- A reduces to B because we can multiply by 3 : $A(z) = B(z, B(z, z))$.

Multby3 (x)

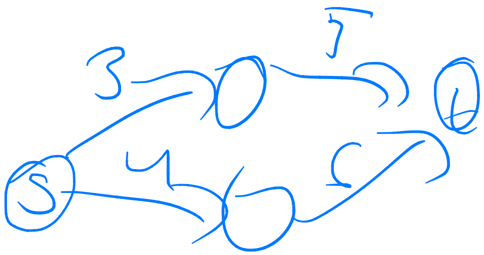
y = x + x

z = y + x

Return z

More Reduction Examples

- A is max flow, B is linear programming
- A is $1||\Sigma C_j$, B is $1||\Sigma w_j C_j$
- A is $P||C_{\max}$, B is $P|prec|\Sigma w_j C_j$



$\underbrace{\text{MaxFlow}(G, \text{cap } u, s, t)}$

Write LP

$\max \sum f_{sj}$

s.t.

$f_{ij} \le u_{ij} \quad \forall (i,j) \in E$

$\sum_{j \in V} f_{ij} = \sum_{j \in V} f_{ji} \quad \forall i \in V - \{s,t\}$

$f_{ij} \ge 0 \quad \forall (i,j) \in E$

Call an LP solver

Return the answer

Have code for $1||\sum w_j C_j$

j	p_j
1	2
2	6
3	1
n	3

$\min \sum C_j$

If all $w_j = 1$
$\sum C_j = \sum w_j C_j$

$\boxed{\text{Solve}1w_jC_j(n, w_1 \ldots w_n, p_1, \ldots p_n) \text{ output schedule}}$

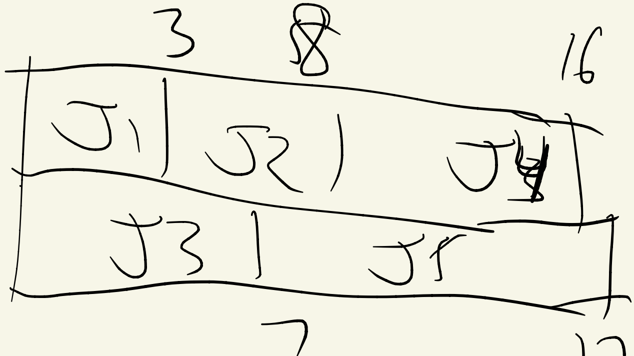
Solve $1|C_j(n, p_1, \ldots p_j)$

Return Solve $1w_jC_j(n, 1, \ldots 1, p_1 \ldots p_n)$

$P||C_{max}$ reduces to $P|prec|\sum w_j C_j$

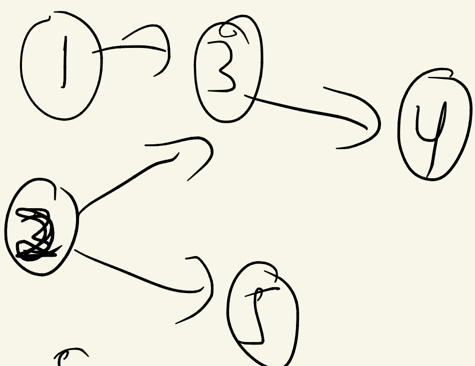
$m = 2$

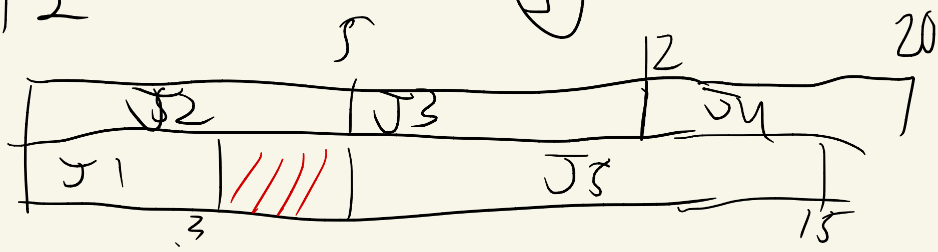
j	p_j
1	3
2	5
3	7
4	8
5	10



$C_{max} = 17$

j	p_j	w_j
1	3	2
2	5	3
3	7	5
4	8	1
5	10	2



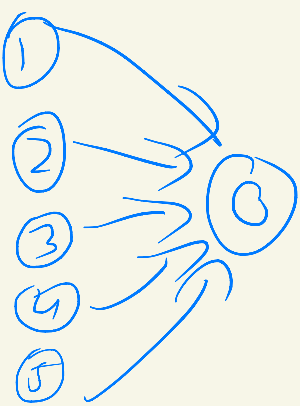


$P||C_{max}$ reduces to $P|prec|\sum w_j C_j$

j	p_j	w_j
1	3	0
2	5	0
3	7	0
4	8	0
5	10	0
0	0	1

if there is a job that has to come at end & it has ~~wt~~ all the wt. then obj are the same

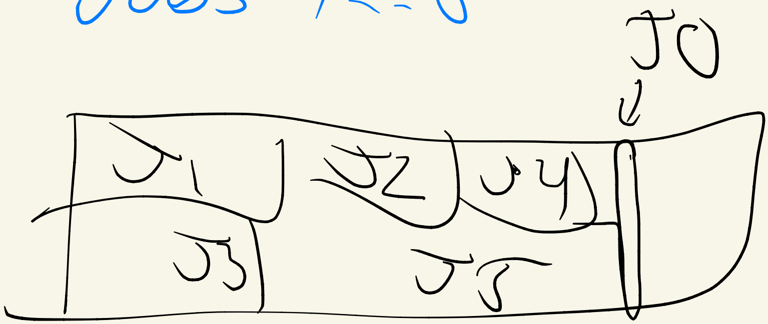
$$\sum w_j C_j = \cancel{w_1} C_1 + \cancel{w_2} C_2 + \cancel{w_3} C_3 + \cancel{w_4} C_4 + \cancel{w_5} C_5 + w_0 C_0 = C_0$$



schedule minimizing $P|prec|\sum w_j C_j$ will put job 0 as early as possible (obj func = C_0)

$\Rightarrow$ finish jobs 1...5 as early as possible

$\Rightarrow$ minimize C_{max} for jobs 1...5



Summary

- Form input for $P|prec|\sum w_j C_j$ by setting $w_j = 0$ $\forall$ jobs and a new job 0 with $p_0 = 0, w_0 = 1$
- Prec constr

- Solve the $P|prec|\sum w_j C_j$ instance
- Return C_0 ($= C_{max}$ of $P||C_{max}$ instance)

Reductions for NP-completeness

- **For technical reasons, We will only consider decision versions of problems.**
- **e.g.** $P||C_{\max}$ **; Given** m **machines,** n **jobs and a number B, does the optimal schedule have makespan less than B.**
- **e.g. Shortest Paths: Given a graph** G **with weights on the edges, two distinguished vertices** s **and** t **and a number B, is the shortest path from** s **to** t **of length less than** B **.**
- **The decision version and the optimization version of a problem are "equivalent," that is they each reduce to each other.**

Reduction Example

Vertex Cover A **vertex cover** of a graph **G=(V,E)** is a set of vertices **V'**, such that for every edge (x,y), at least one of x and y is in **V'**. The vertex cover problem is given a graph **G** and a number **k** and asks whether **G** has a vertex of size at most **k**.

Clique A **clique** is a set of vertices such that each pair of vertices has an edge between them. The **clique problem** is given a graph and a number ℓ and asks when a graph has a clique of size at least ℓ.

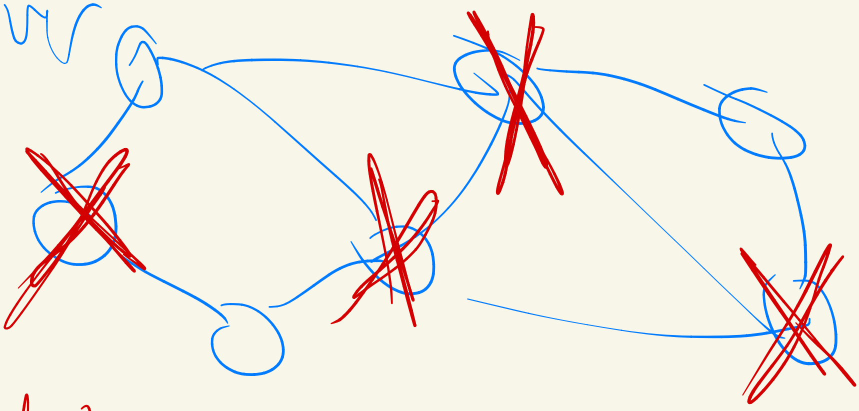
Question: Show that vertex cover reduces to clique.

VC reduces to clique

VC

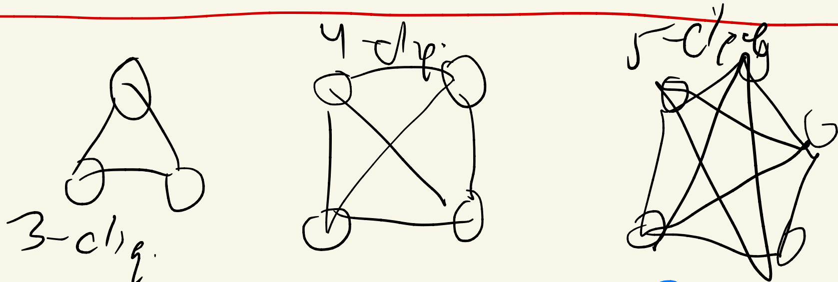
clique

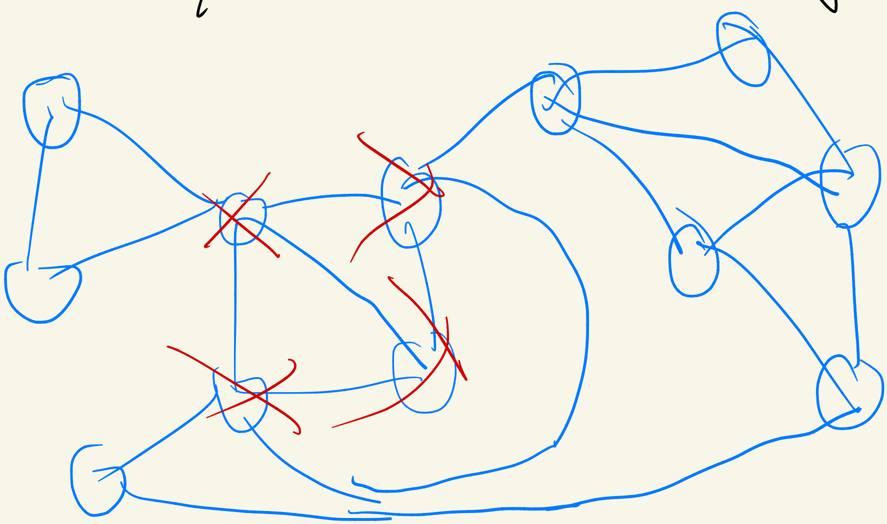
vertex Cover



red X form a V.C.

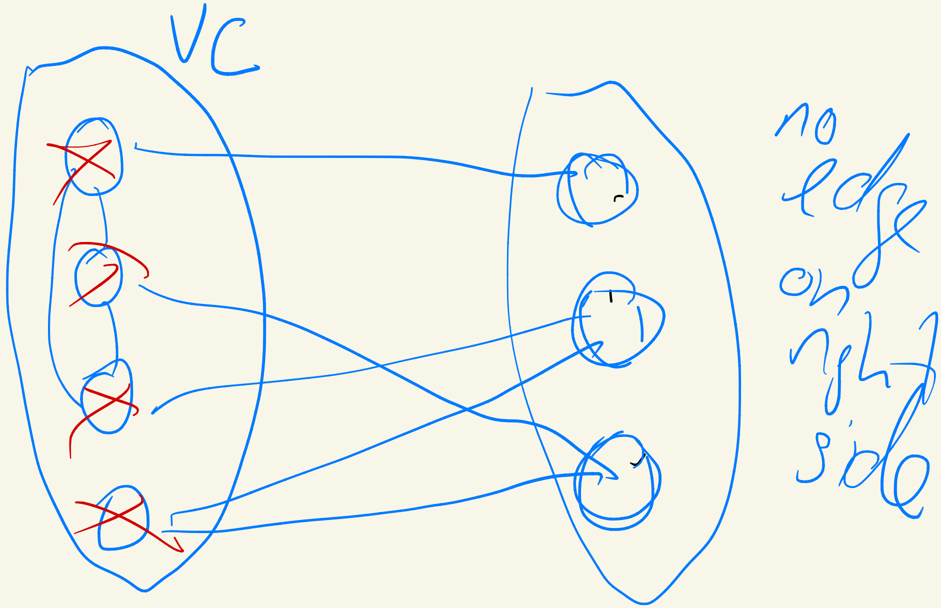
Clique





Given a graph (G, k)

- Form some new graph G'
- Find some l
- Solve Clique (G', l)
- use solution to clique to find a V.C.



- set of vertices not in VC have no edges between them

- set of vertices in a clique have all edges between them

Reduce VC to clique

Input (G, k)

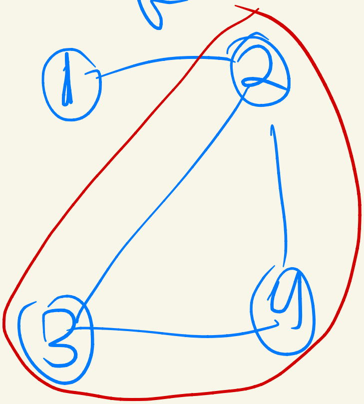
Compute $G' =$ compliment of G.

Set $\ell = |V(G)| - k$

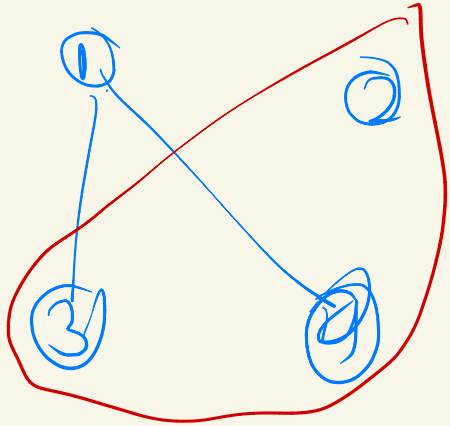
Solve $\text{Clique}(G', \ell)$

output yes/no from $\text{Clique}(G', \ell)$

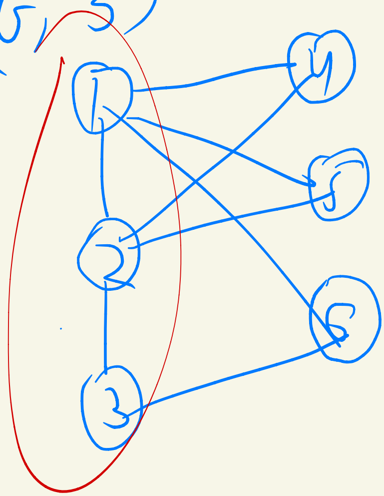
G



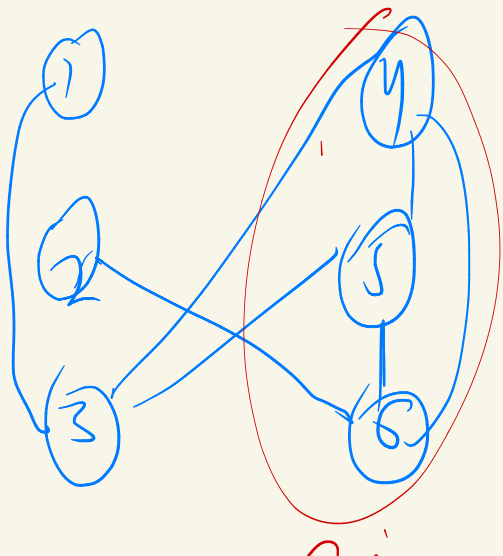
complement G



(G, 3) $\Rightarrow$ (G comp, 3)

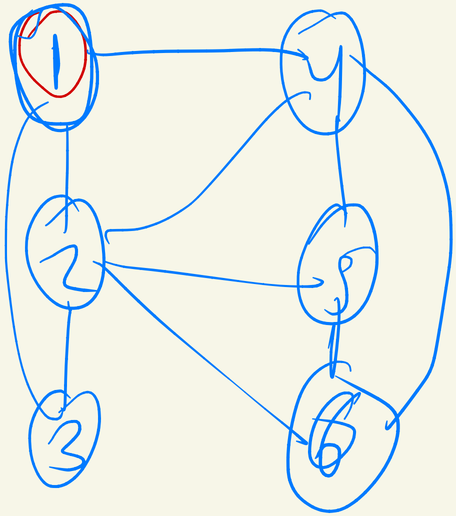


VC.



Cliqe

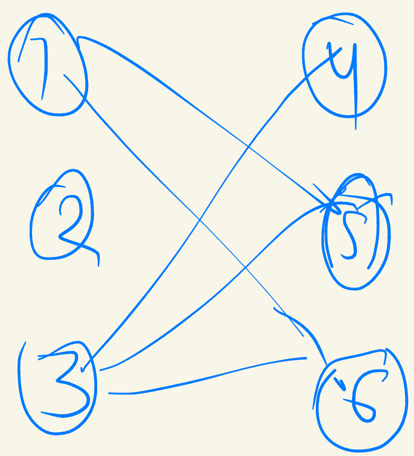
is there a V.C. of size 3



no VC of size 3

VC of size 4

G comp



no clique of size 3

Clique of size 2

Claim VC reduces to clique