WE HAVE BEEN SOLVING PROBLEM A BY SOLVING SMALLER VERSIONS OF PROBLEM A
GENERAL IDEA:
SOLVE PROBLEM A BY SOLVING PROBLEM B
D&C, DP, or Greedy
Instance of size N
Your Algorithm

D&C, DP, or Greedy
Instance of size N

D&C, DP, or Greedy
Instance of size <N

n/2

S_L

S_R

solutions to smaller instance
Your Algorithm

D&C, DP, or Greedy
Instance of size <N

D&C, DP, or Greedy
Instance of size <N

D&C, DP, or Greedy
Instance of size N

S
solution to original problem

S_L, S_R
solutions to smaller instance

merge
Bipartite Matching Instance
Bipartite Matching Instance

Max flow Instance

$(\zeta, \epsilon)$
Instances of Bipartite matching
Instances of Bipartite matching $(L,R,E)$

Instances of Max Flow $(G,c)$

Your Algorithm

Ford-Fulkerson

$f$

flow
Instances of Bipartite matching

Matching

Your Algorithm

Instances of Max Flow

Flow

Ford-Fulkerson
Reduction

\[ \text{\textsc{Problem}_a} \leq_{f(n)} \text{\textsc{Problem}_b} \]

"as hard as solving problem A."

Instances of Problem A \rightarrow \text{Your Algorithm} \rightarrow \text{Instances of Problem B}

\[ S_A \rightarrow \downarrow \rightarrow S_B \]

any solution to problem B.
party problem

Draw an edge between people who do not get along.

Find the largest set of people who "get along" in this graph.
independent set

A set \( S \subseteq V \) is an independent set if no two nodes in \( S \) are joined by an edge.
example

- No two nodes joined by an edge.
Given a graph $G$, find the **max independent set**.
baseball
how to defend such a graph??

→ want some defender on every edge

Smallest such set
A vertex cover of a graph is a subset $S \subseteq V$ such that for each edge $e = (x, y) \in E$, either $x \in S$ or $y \in S$. 
a vertex cover of a graph is a set $C \subseteq V$
such that $\forall (x, y) \in E$
either $x \in C$ or $y \in C$
goal:
given a graph $G$, find the minimum size vertex cover for $G$. 
A solution to VC can be used to solve INDSET.
Thm: \( S \) is a VC of graph \( G \) if and only if \( V-S \) is an induced set of \( G \).
Thm: set $S$ is an independent set of $G$ iff $V-S$ is a vertex cover.
Thm: set $S$ is an independent set of $G$ iff $V-S$ is a vertex cover.

- Suppose $S$ is an independent set.

  Consider any edge $e = (x, y) \in E$.

  1. $x \in S \Rightarrow y \in S$ b/c $S$ is in $V$.

  2. $y \in V-S$, edge $e$ is covered.

  3. $x \notin S \Rightarrow x \in V-S$, edge $e$ is covered.

  Therefore, $V-S$ covers every edge in $G$ and so $V-S$ is a VC.
Thm: set $S$ is an independent set of $G$ iff $V-S$ is a vertex cover.

Suppose $V-S$ is a vc.

Consider an $x \in S$, and consider any edge $(x,y) \in E$.

$\Rightarrow$ since $x \notin V-S$, but $V-S$ is a vertex cover, this implies that $y \in V-S$ to cover edge $(x,y)$.

$\Rightarrow y \notin S$. This holds for any $v \in S$ and any edge $(v,w) \in E$.

$\Rightarrow S$ must be an independent set.
Instances of MinVertex Cover

Instances of MaxIndSet

Your Algorithm

Vertex Cover

V-S

S

Ind Set

UC \leq \text{INDSET}

\text{INDSET} \leq \text{UC}

\Theta(U) \text{ time}

Using any solver for \text{NP} set.
3sat problem

input: formula in 3CNF form: logical $\overline{\text{AND}}$ of clauses of 3-variables

$((a \lor b \lor c) \overline{\text{AND}} (\overline{a} \lor e \lor d \lor f) \overline{\text{AND}} (\overline{\text{variable}}))$

output: “Assignment $A: V \rightarrow \text{True or False}$ s.t. the formula is True.”
3sat example

\((x \lor y \lor z) \land (x \lor \overline{y} \lor y) \land (u \lor y \lor \overline{z}) \land (z \lor \overline{x} \lor u) \land (\overline{x} \lor \overline{y} \lor \overline{z}) = \phi\) 5 clause formula

\[ A(x) \rightarrow T \]
\[ (y) \rightarrow T \]
\[ (u) \rightarrow T \]
\[ (z) \rightarrow F \]
\[3\text{SAT} \leq_p \text{INDSET}\]

\[(x \lor y \lor z) \land (x \lor \overline{y} \lor y) \land (u \lor y \lor z) \land (z \lor \overline{x} \lor u) \land (\overline{x} \lor \overline{y} \lor z)\]

what must we do to?

3SAT $\phi$

Assignment $A$

G

INDSET

solv

S
Instance of 3SAT $\phi$  \rightarrow \text{Your Algorithm}  \rightarrow \text{Instances of MaxIndSet} (G)
Instance of 3SAT

Your Algorithm

\[ |S| \leq K \]

A satisfying assignment

Ind Set
gadgets
\[3\text{SAT} \leq_p \text{INDSET}\]

\[(x \lor y \lor z) \land (x \lor \overline{y} \lor y) \land (u \lor y \lor \overline{z}) \land (z \lor \overline{x} \lor u) \land (\overline{x} \lor \overline{y} \lor \overline{z})\]
$3\text{SAT} \leq_p \text{INDSET}$

$((x \lor y \lor z) \land (x \lor \overline{y} \lor y) \land (u \lor y \lor \overline{z}) \land (z \lor \overline{x} \lor u) \land (\overline{x} \lor \overline{y} \lor \overline{z}))$
\((x \lor y \lor z) \land (x \lor \overline{y} \lor y) \land (u \lor y \lor \overline{z}) \land (z \lor \overline{x} \lor u) \land (\overline{x} \lor \overline{y} \lor \overline{z})\)
\( \phi \in \text{SAT} \implies \exists \text{ a satisfying assignment } A \text{ for } \phi \) \\
\implies \text{ each clause has a true variable (can be } \geq 1) \\
\text{pick one true variable per clause.} \\
\text{Let } S = \bigcap}\text{ set } 3. \quad |S| \geq K. \text{ b/c there are } K \text{ clauses.} \\
\text{And if a variable is clause } i \text{ is selected, then its} \\
\text{negation is false, & not selected. Node per triangle}
By definition, only 1 node per triangle is selected. Make that variable true. Assign any unassigned variable to be T.

1. The resulting A assigns every variable.
2. The assignment A is consistent. If \( A(k) = \text{true} \), then no vertex for \( x \) is in the indset. \( \Rightarrow \overline{x} \) is assigned F.
3. \( \phi \) is satisfied by A. By each clause, \( \phi \) is satisfied.
Road Map

SAT ≥ p
3SAT ≥ p
CLIQUE ≥ p
HAMPATH ≥ p
3COL ≥ p
SET COVER ≥ p
IND SET ≥ p
VERTEX COVER ≥ p
SUBSET-SUM ≥ p
clique

Social graph
clique = \{ (G, K) \mid \text{ determine if } G \text{ has a clique of size } K \}

a complete graph on \( K \) nodes
all \( K \) vertices are connected to each other

3SAT \leq \text{ CLIQUE}

"clique is as hard as 3SAT up to polytime"
Instance of 3SAT

\[
\phi = \left( x_1 \lor x_2 \lor x_3 \right) \land \left( \overline{x_1} \lor \overline{x_2} \lor x_3 \right) \land \left( x_1 \lor x_2 \lor \overline{x_3} \right)
\]

Instances of Clique

Your Algorithm

\((G,k)\)

Instances of Clique
$\phi = (x_1 \lor x_2 \lor x_3) \land (\overline{x_1} \lor \overline{x_2} \lor x_3) \land (x_1 \lor \overline{x_2} \lor \overline{x_3})$

Instance of 3SAT

Your Algorithm

A satisfying assignment

$\mathcal{S}$

Ind Set

Instances of Clique

$(G,k)$
CLIQUE

formula \( k_{\text{clauses}} \)

\[ \phi = (x_1 \lor x_2 \lor x_3) \land (\overline{x_1} \lor \overline{x_2} \lor x_3) \land (x_1 \lor x_2 \lor \overline{x_3}) \]

(\text{Graph, } k) \quad k = \# \text{ clauses}
CLIQUE

\[ \phi = (x_1 \lor x_2 \lor x_3) \land (x_1 \lor x_2 \lor x_3) \land (x_1 \lor x_2 \lor x_3) \]

Create 3 nodes/clause

Graph, k

\[ k = \# \text{ clauses} \]
CLIQUE

\[ \phi = (x_1 \lor x_2 \lor x_3) \land (\overline{x_1} \lor \overline{x_2} \lor x_3) \land (x_1 \lor x_2 \lor \overline{x_3}) \]

Create 3 nodes/clause
Connect nodes to “non-opposites”

Graph, k
\[ k = \# \text{ clauses} \]
CLIQUE

$\phi = (x_1 \lor x_2 \lor x_3) \land (x_1 \lor x_2 \lor x_3) \land (x_1 \lor x_2 \lor x_3)$

Create 3 nodes/clause

Connect nodes to "non-opposites"
CLIQUE

\[ \phi = (x_1 \lor x_2 \lor x_3) \land (\overline{x_1} \lor \overline{x_2} \lor \overline{x_3}) \land (x_1 \lor x_2 \lor x_3) \]

Create 3 nodes/clause

Connect nodes to “non-opposites”

Graph, \( G \)

\( k = \# \) clauses

order \( 3k \cdot V \) time

if \( G \) has a \( k \) clique
\( \phi = (x_1 \lor x_2 \lor x_3) \land (\overline{x_1} \lor \overline{x_2} \lor x_3) \land (\overline{x_1} \lor x_2 \lor \overline{x_3}) \)
CLIQUE

\[ \phi = (x_1 \lor x_2 \lor x_3) \land (x_1 \lor x_2 \lor x_3) \land (x_1 \lor x_2 \lor x_3) \]

Satisfying assignment = 1 var/clause
CLIQUE

$\phi = (x_1 \lor x_2 \lor x_3) \land (\overline{x_1} \lor \overline{x_2} \lor x_3) \land (\overline{x_1} \lor x_2 \lor \overline{x_3})$

Satisfying assignment = 1 var/clause

k “non-opposite” connected nodes
\( \phi = \left( x_1 \lor x_2 \lor x_3 \right) \land \left( \overline{x_1} \lor x_2 \lor x_3 \right) \land \left( x_1 \lor x_2 \lor \overline{x_3} \right) \)

k-clique

1 node/clause is true
$\phi = (x_1 \lor x_2 \lor x_3) \land (\overline{x_1} \lor x_2 \lor x_3) \land (x_1 \lor x_2 \lor \overline{x_3})$

k-clique

1 node/clause is true

Satisfying assignment
\[ \phi = (x_1 \lor x_2 \lor x_3) \land (\overline{x_1} \lor \overline{x_2} \lor x_3) \land (x_1 \lor x_2 \lor \overline{x_3}) \]

\[ \phi \in SAT \iff f(\phi) \in CLIQUE \]
Theory of NP
Languages

$L = \{ \text{sets on instance} \}$

$x \in L$
DEF OF NP

A language L belongs to the class NP iff there exists a polynomial time algorithm A and a constant c such that

\[ L = \{ x \in \{0, 1\}^* \mid \exists y \in \{0, 1\}^{x^c} \text{ s.t. } A(x, y) = 1 \} \]

Efficiently test whether a solution is true.
A language $L$ is **NP-Complete** if

1. $L \in \text{NP}$
2. $\forall A \in \text{NP}, A \leq_p L$

“$L$ is among the hardest NP problems”
WHY IS VC IN NP?

vertexcover(G,k)
COOK-LEVIN THEOREM

WHAT IS THE HARDEST PROBLEM IN NP?
Cook-Levin theorem

\[ \forall L \in \text{NP} \]

\[ L \leq_f 3\text{SAT} \]