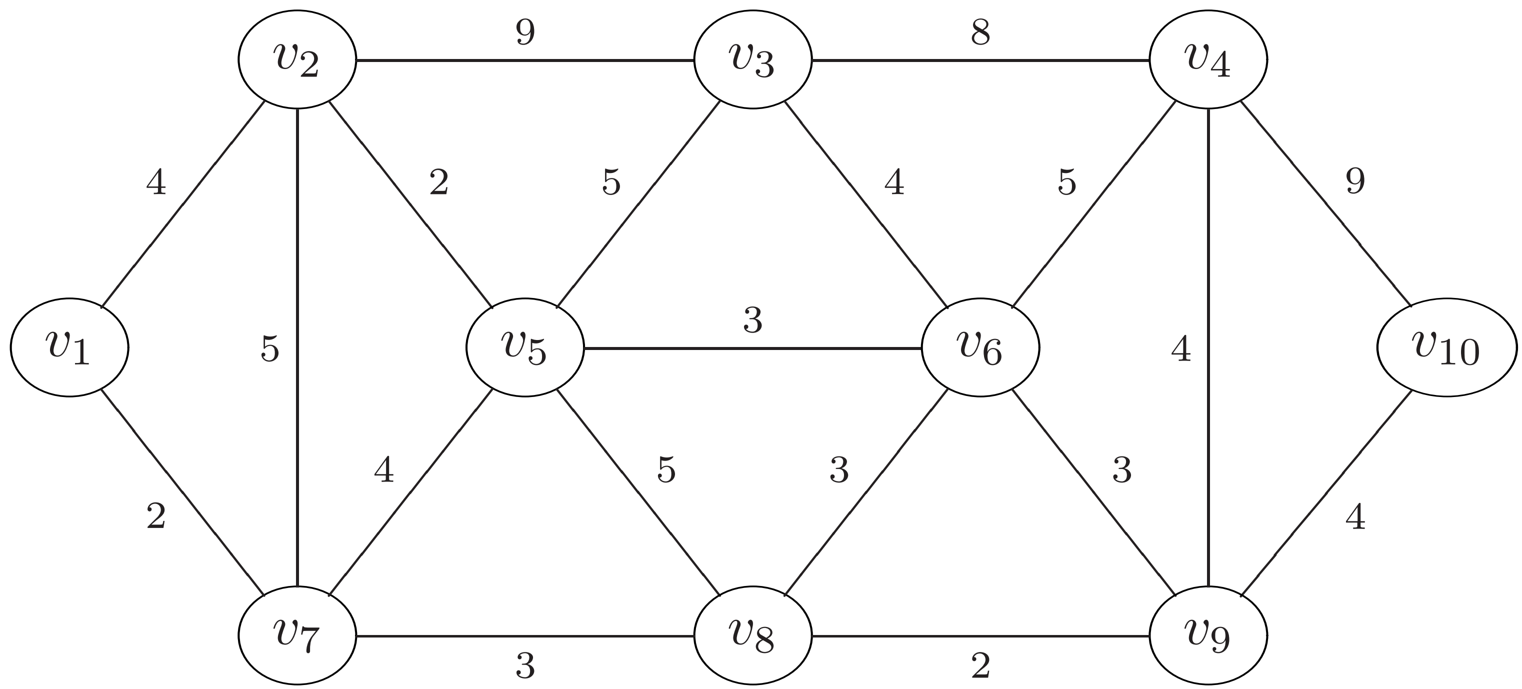
NP-Completeness

Traveling Salesman Problem: Example



Find the minimum cost path from v_1 to v_{10} which visits each vertex exactly once.

Traveling Salesman Problem

Given a weighted graph G, find the minimum cost path from v_1 to v_n which visits each vertex exactly once.

- No known polynomial time algorithm for solving this problem.

Traveling Salesman: Decision Problem

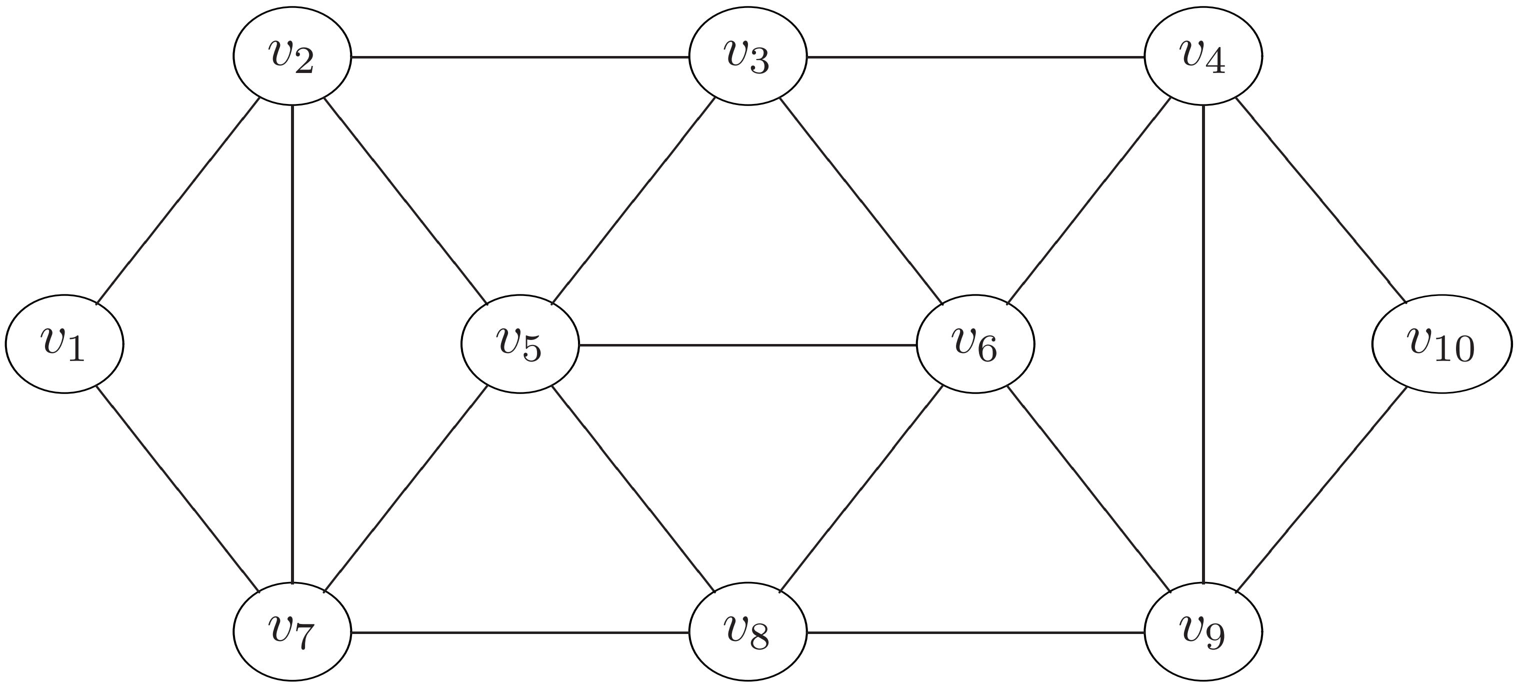
Optimization problem: Given a weighted graph G, find the minimum cost path from v_1 to v_n which visits each vertex exactly once.

Decision problem: Given a weighted graph G and a cost C, is there a path from v_1 to v_n which visits each vertex exactly once and has cost less than C?

A **decision problem** has a **yes** or **no** answer.

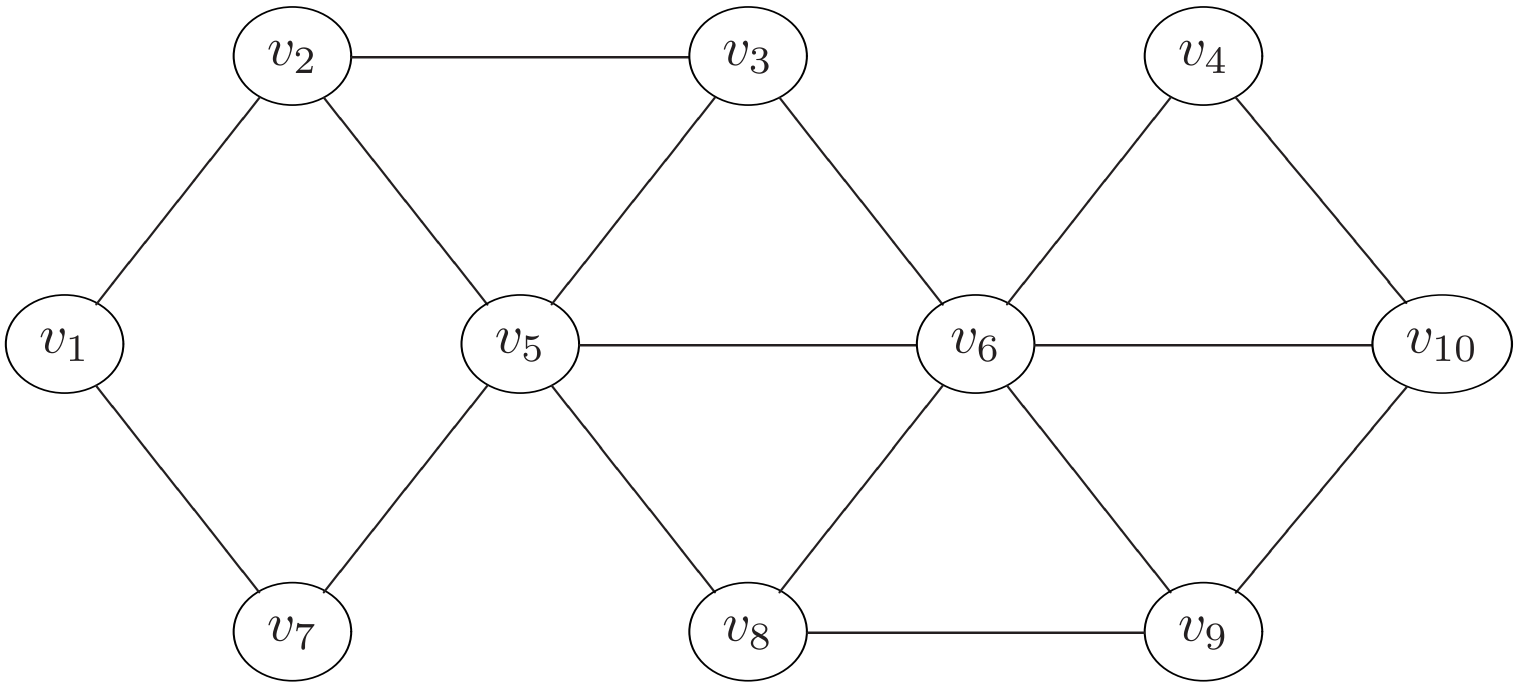
- No known polynomial time algorithm for solving the decision problem.

Hamilton Path Problem: Example



Is there a path from v_1 to v_{10} which visits each vertex exactly once?

Hamilton Path Problem: Example 2



Is there a path from v_1 to v_{10} which visits each vertex exactly once?

Reduction

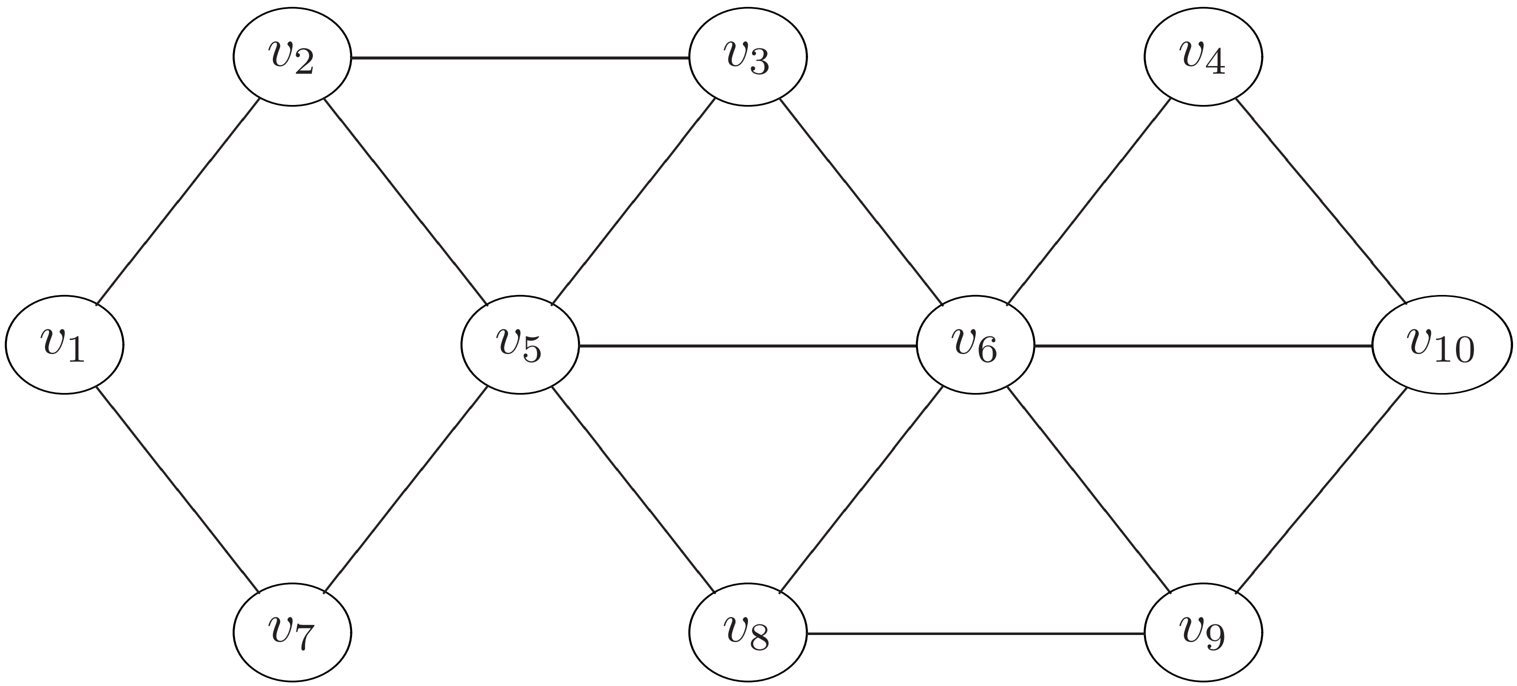
A **reduction** is a transformation of one problem into another.

Hamiltonian Path Problem: Given a graph G, is there a path from v_1 to v_n which visits each vertex exactly once?

Traveling Salesman Problem: Given a weighted graph G and a cost C, is there a path from v_1 to v_n which visits each vertex exactly once and has cost less than C?

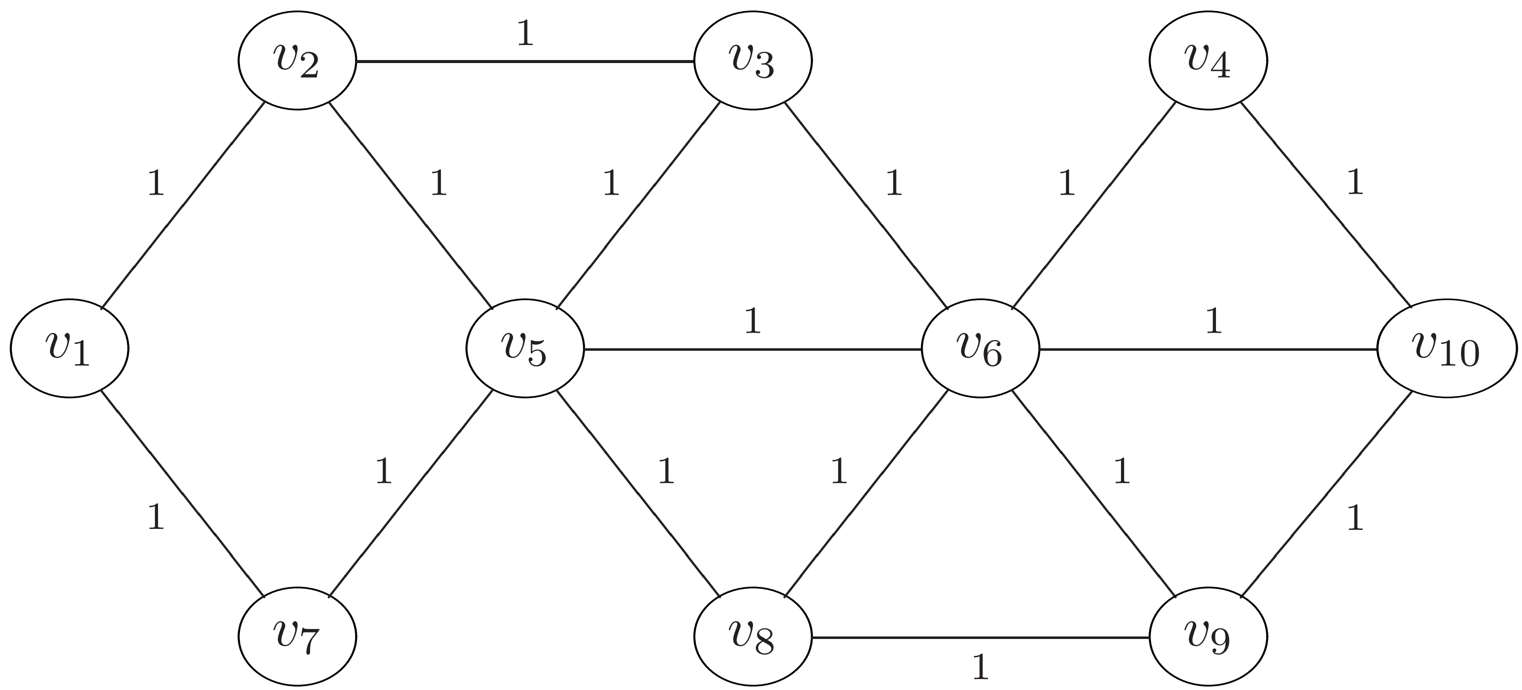
The Hamiltonian Path Problem can be reduced to the Traveling Salesman Problem: Given a graph G, assign the weight 1 to each edge of G.
Let C equal n.
Graph G has a Hamiltonian path from v_1 to v_n if and only if G has a traveling salesman path with cost less than n.

Hamiltonian Path



Hamiltonian Path Problem: Is there a path from v_1 to v_{10} which visits each vertex exactly once?

Hamiltonian Path and Traveling Salesman



Hamiltonian Path Problem: Is there a path from v_1 to v_{10} which visits each vertex exactly once?

Traveling Salesman Problem: Is there a path from v_1 to v_{10} which visits each vertex exactly once and has cost less than 10?

Reduction

Definition. A **reduction** of decision problem Q_1 to decision problem Q_2 is a mapping of every instance q_1 of problem Q_1 to an instance q_2 of problem Q_2 such that q_1 is **yes** if and only if q_2 is **yes**.

Example:

- An instance of the Hamiltonian Path Problem is a graph G with vertices v_1 and v_n.
- Construct a weighted graph G' by adding weight 1 to all edges of G.
- An instance of the Traveling Salesman Problem is graph G' and cost n.
- There is a path from v_1 to v_n visiting all the vertices in G if and only if there is a path with cost less than n from v_1 to v_n visiting all the vertices in G'.

Other Problems Reducible to Traveling Salesman

3-Coloring: Given a graph G, is there a coloring of the vertices with 3 colors so that no two adjacent vertices have the same color?

Independent Set: Given a graph G and an integer k, is there a set S of k vertices in G such that no two vertices in S are adjacent?

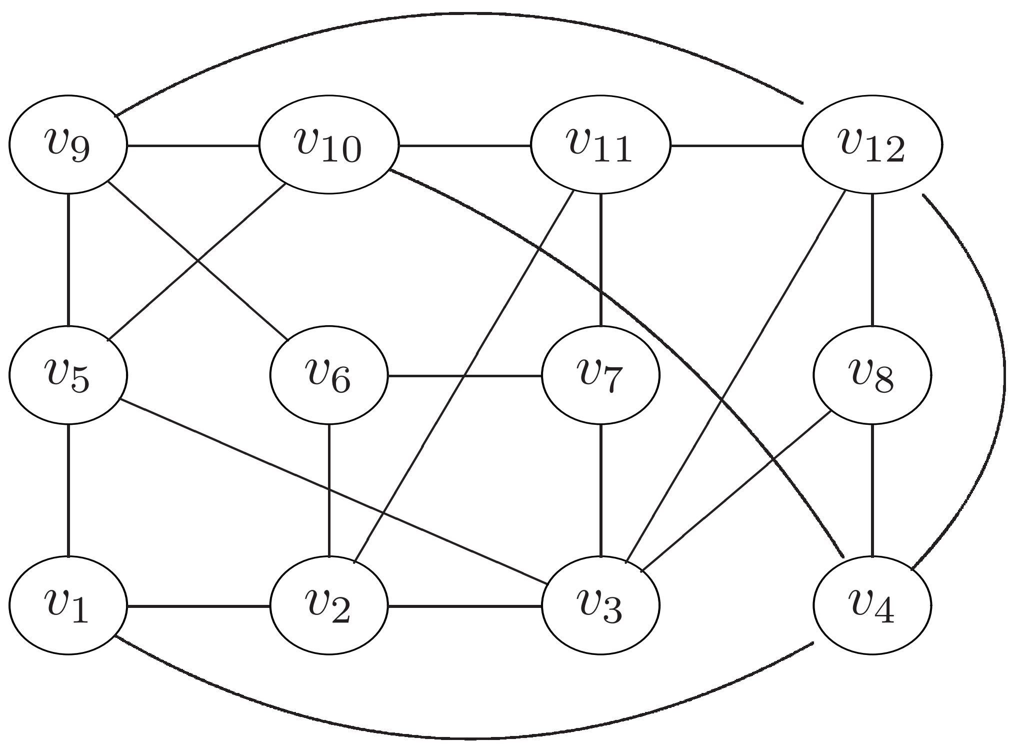
Dominating Set: Given a graph G and an integer k, is there a set S of k vertices in G such that every vertex of G is adjacent (or equals) a vertex in S?

Longest Simple Path: Given a weighted graph G and a distance D is there a simple path from v_1 to v_n in G whose distance is greater than D? (A *path* is simple if each vertex along the path appears only once.)

Subgraph Isomorphism Given two graphs G and G', is G' a subgraph of G?

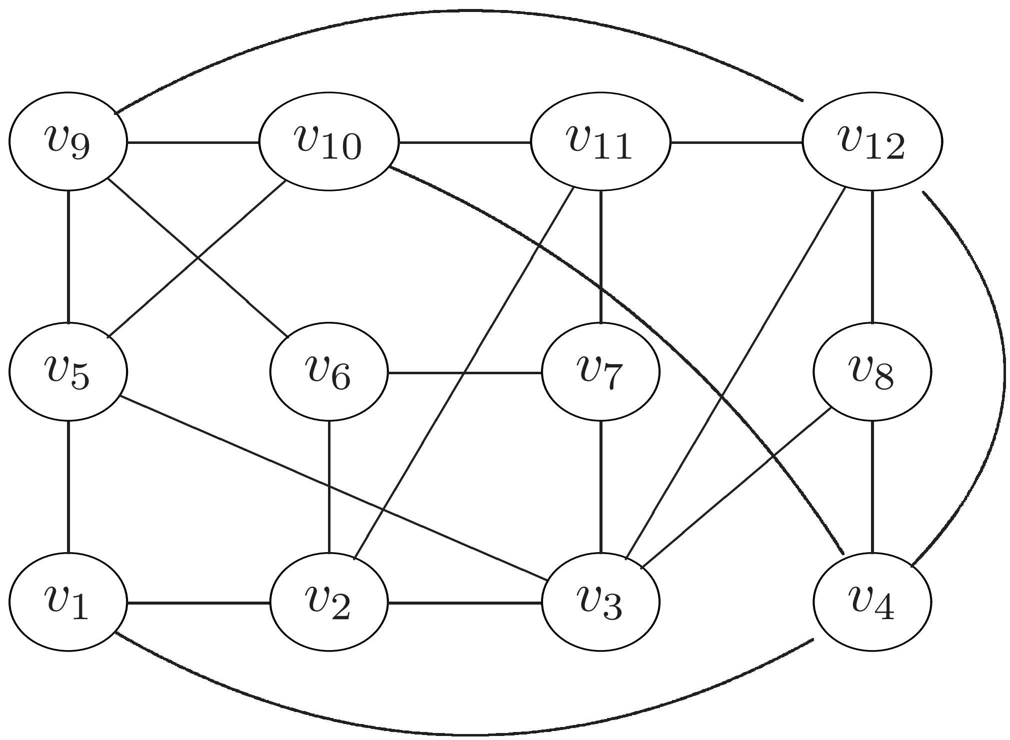
3-Coloring

3-Coloring: Given a graph G, is there a coloring of the vertices with 3 colors so that no two adjacent vertices have the same color?



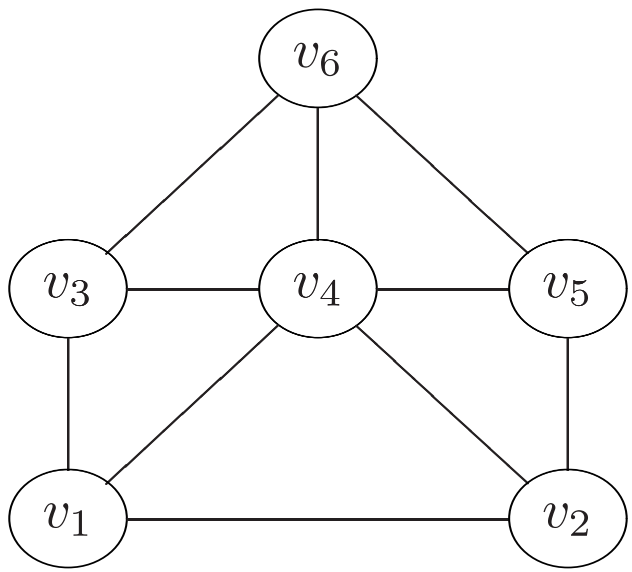
3-Coloring (Copy of previous slide)

3-Coloring: Given a graph G, is there a coloring of the vertices with 3 colors so that no two adjacent vertices have the same color?



3-Coloring

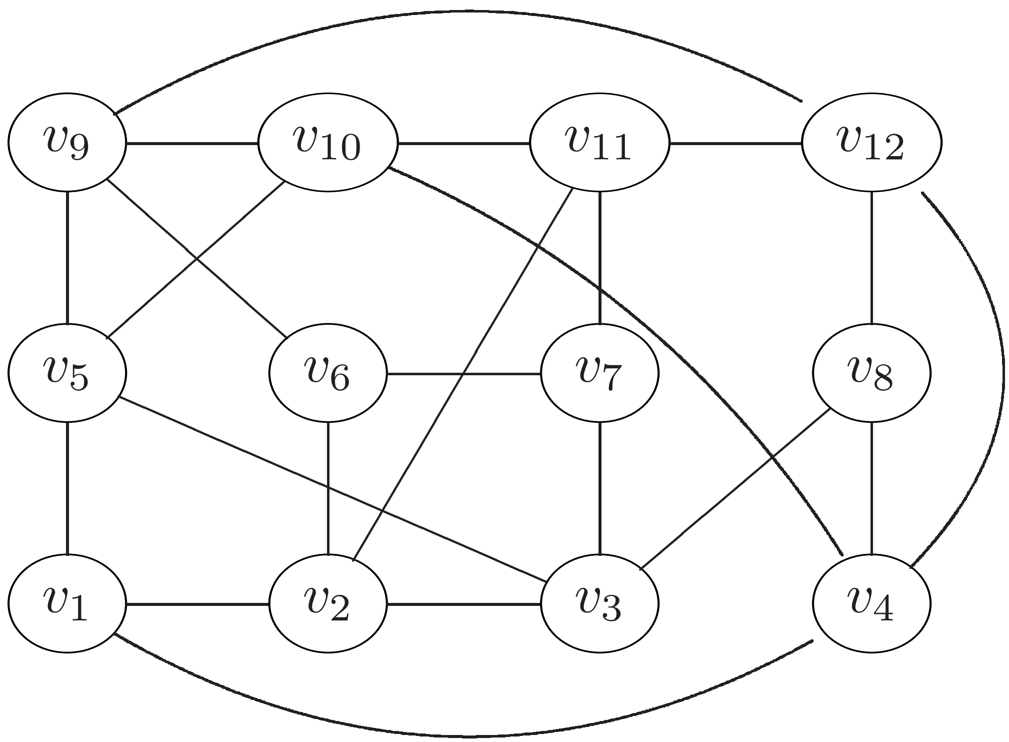
3-Coloring: Given a graph G, is there a coloring of the vertices with 3 colors so that no two adjacent vertices have the same color?



Independent Set

Independent Set Given a graph G and an integer k, is there a set S of k vertices in G such that no two vertices in S are adjacent?

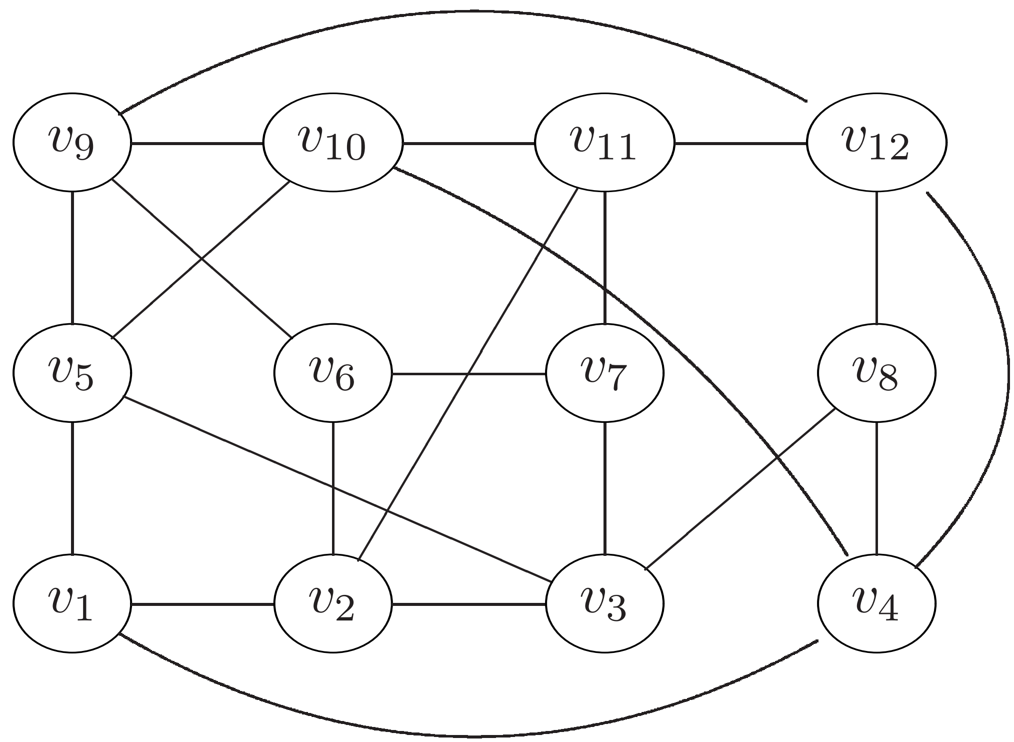
Is there an independent set of 4 vertices?



Independent Set

Independent Set Given a graph G and an integer k, is there a set S of k vertices in G such that no two vertices in S are adjacent?

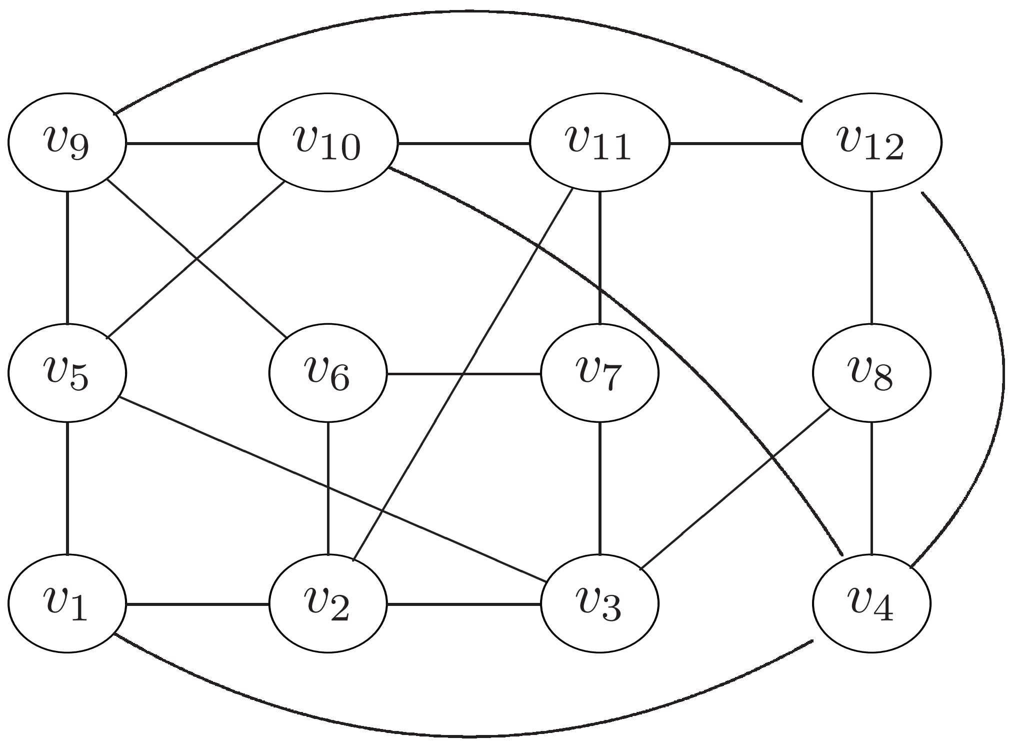
Is there an independent set of 5 vertices?



Independent Set

Independent Set Given a graph G and an integer k, is there a set S of k vertices in G such that no two vertices in S are adjacent?

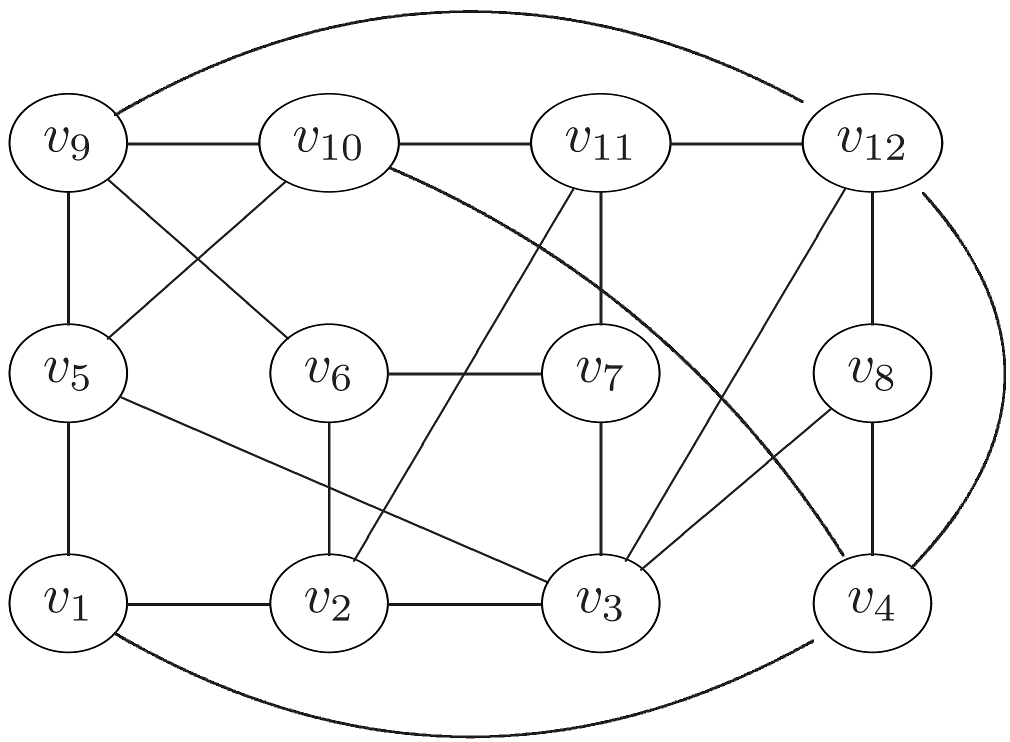
Is there an independent set of 6 vertices?



Dominating Set

Dominating Set: Given a graph G and an integer k, is there a set S of k vertices in G such that every vertex of G is adjacent (or equals) a vertex in S?

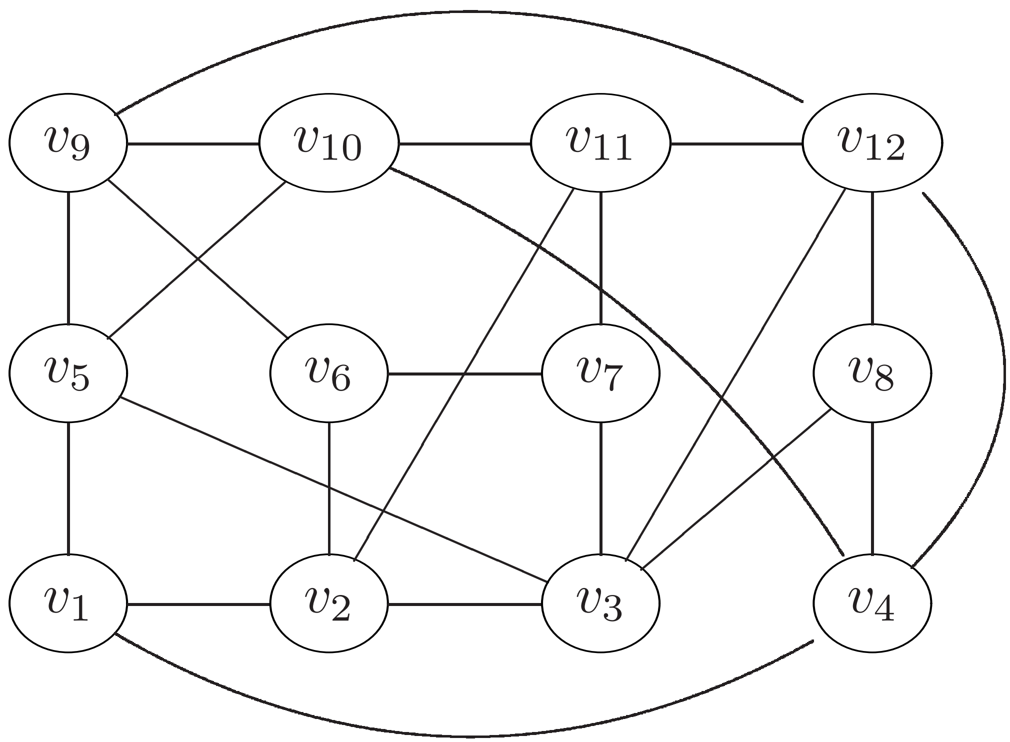
Is there an dominating set of 4 vertices?



Dominating Set

Dominating Set: Given a graph G and an integer k, is there a set S of k vertices in G such that every vertex of G is adjacent (or equals) a vertex in S?

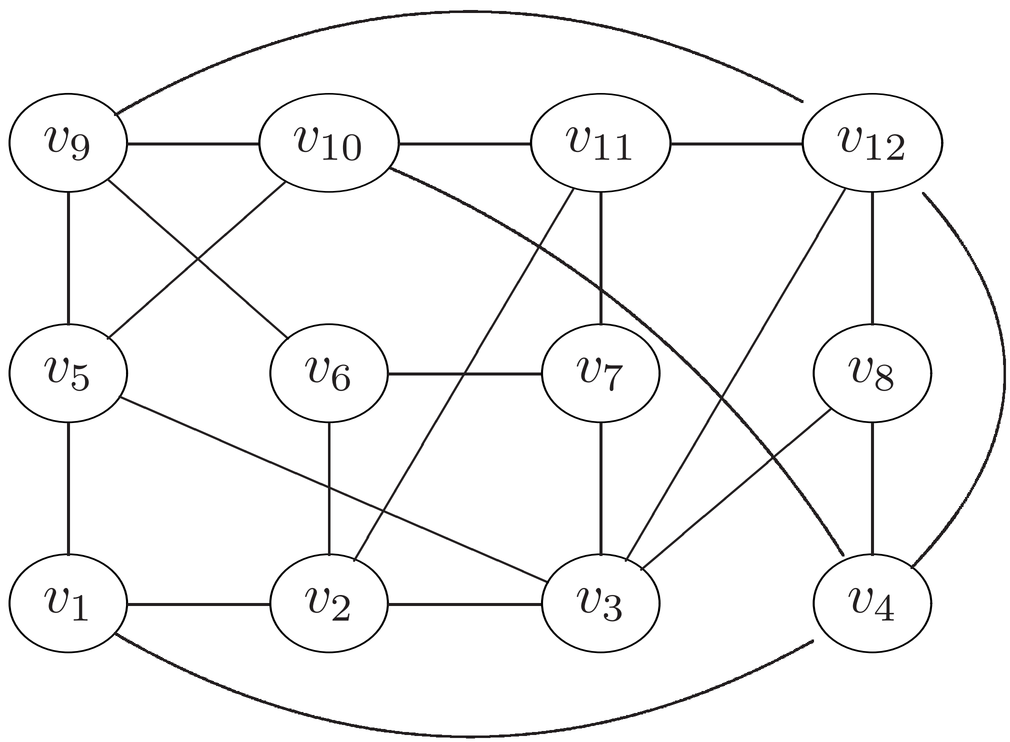
Is there an dominating set of 3 vertices?



Dominating Set

Dominating Set: Given a graph G and an integer k, is there a set S of k vertices in G such that every vertex of G is adjacent (or equals) a vertex in S?

Is there an dominating set of 2 vertices?



Other Problems Reducible to Traveling Salesman

3-Coloring: Given a graph G, is there a coloring of the vertices with 3 colors so that no two adjacent vertices have the same color?

Independent Set: Given a graph G and an integer k, is there a set S of k vertices in G such that no two vertices in S are adjacent?

Dominating Set: Given a graph G and an integer k, is there a set S of k vertices in G such that every vertex of G is adjacent (or equals) a vertex in S?

Longest Simple Path: Given a weighted graph G and a distance D is there a simple path from v_1 to v_n in G whose distance is greater than D? (A *path* is simple if each vertex along the path appears only once.)

Subgraph Isomorphism Given two graphs G and G', is G' a subgraph of G?

Problems Reducible to/from Traveling Salesman

- Traveling Salesman;
- Hamiltonian Path;
- 3-Coloring;
- Independent Set;
- Domination Set;
- Longest Simple Path;
- Subgraph Isomorphism.

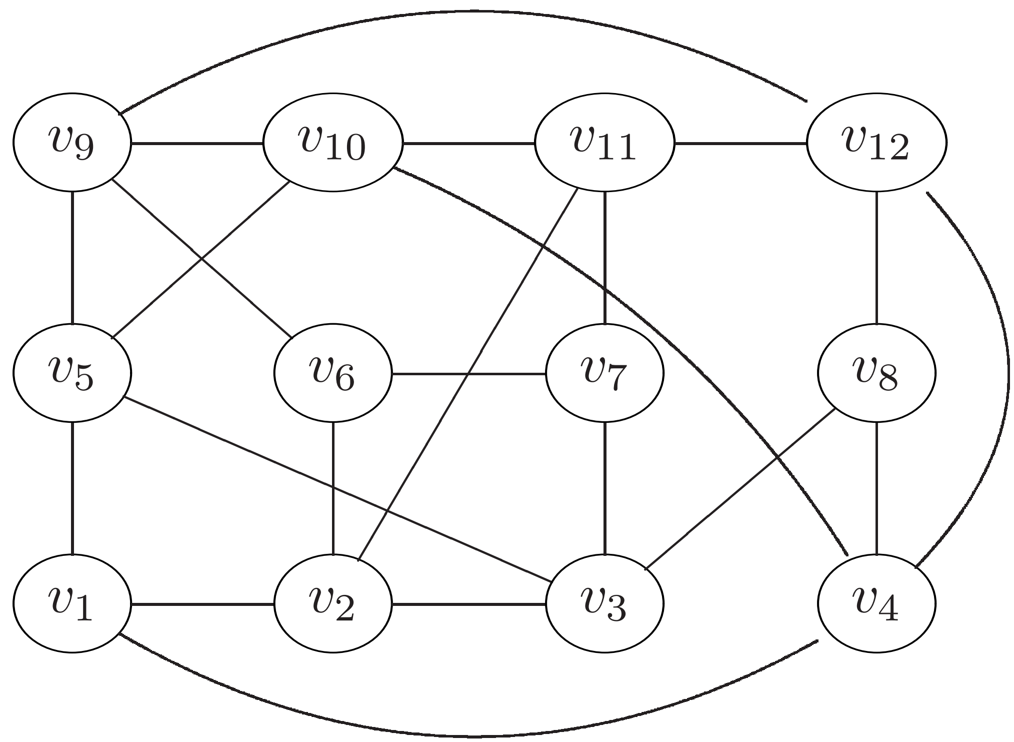
– No known polynomial time algorithm for solving any of these problems;

– If any of these problems can be solved in polynomial time, then they can all be solved in polynomial time.

Independent Set

Independent Set Given a graph G and an integer k, is there a set S of k vertices in G such that no two vertices in S are adjacent?

Is there an independent set of 5 vertices?



Independent Set

Independent Set: Given a graph G and an integer k, is there a set S of k vertices in G such that no two vertices in S are adjacent?

Independent Set of Size 5: Given a graph G, is there a set S of 5 vertices in G such that no two vertices in S are adjacent?

Independent Set of Size 5

Independent Set of Size 5: Given a graph G, is there a set S of 5 vertices in G such that no two vertices in S are adjacent?

Claim: Independent Set of Size 5 can be solved in $O(n^5)$ time!

(n = number of vertices)

$O(n^5)$ Algorithm for Independent Set of Size 5

Input : Graph G with n vertices.

```
Build an adjacency matrix for G ;                /* O(n^2) time */
foreach set S of 5 vertices of G do              /* O(n^5) sets */
    foreach pair v_i, v_j ∈ S do                 /* O(5^2) pairs */
        Check if (v_i, v_j) is an edge of G;
    end
end
```

$O(n^k)$ Algorithm for Independent Set

Input : Graph G with n vertices.

Build an adjacency matrix for G ; /* $O(n^2)$ *time* */
foreach set S of k vertices of G **do** /* $O(n^k)$ *sets* */
    foreach pair $v_i, v_j \in S$ **do** /* $O(5^2)$ *pairs* */
        Check if (v_i, v_j) is an edge of G;
    end
end

Algorithm runs in $O(n^k)$ time.
This algorithm does NOT run in polynomial time, since k is part of the input and is not fixed.

Problems Reducible to/from Traveling Salesman

- Traveling Salesman;
- Hamiltonian Path;
- 3-Coloring;
- Independent Set;
- Domination Set;
- Longest Simple Path;
- Subgraph Isomorphism.

– No known polynomial time algorithm for solving any of these problems;

– If any of these problems can be solved in polynomial time, then they can all be solved in polynomial time.

Verification

Definition. A decision problem is **verifiable** in polynomial time if for every instance with the answer **yes**, there is a solution S which we can use to check in polynomial time that the answer is **yes**.

Example:

- An instance of the Hamiltonian Path Problem is a graph G with vertices v_1 and v_n.
- A solution S is a sequence of vertices starting with v_1 and ending with v_n;
- In polynomial time, we can check if S contains every vertex exactly once and if every consecutive (w, w') in S is an edge of G.

Class NP

Definition. A decision problem is **verifiable** in polynomial time if for every instance with the answer **yes**, there is a solution S which we can use to check in polynomial time that the answer is **yes**.

Definition. A decision problem is in the class NP, if it is verifiable in polynomial time.

(NP stands for non-deterministic polynomial.)

NP-Complete

Definition. A decision problem is in the class NP, if it is verifiable in polynomial time.

Definition. A decision problem Q is NP-complete if

- Q is in NP;
- Every problem in NP can be reduced to Q in polynomial time.

NP-Complete Problems

The following problems (and many others) are NP-complete:

- Traveling Salesman;
- Hamiltonian Path;
- 3-Coloring;
- Independent Set;
- Domination Set;
- Longest Simple Path;
- Subgraph Isomorphism.

If you can solve ANY NP-complete problem in polynomial time, then you can solve EVERY NP-complete problem (and every problem in NP) in polynomial time!?!

More NP-Complete Problems

Boolean Satisfiability: Given a boolean expression (i.e. $(x_1 \vee x_2) \wedge (\neg x_2 \vee x_3 \vee \neg x_4) \wedge \ldots$), is there an assignment of true or false to the variables x_i so that the expression is true?

Subset Sum: Given a set of positive integers $K = \{k_1, k_2, \ldots, k_n\}$ and an integer M, is there a subset of K whose sum equals M?

Set Packing: Given a collection C of finite sets and an integer M, does C contain at least K mutually disjoint sets?

Quadratic Diophantine Equations: Given integers a, b and c, are there positive integers x and y such that $ax^2 + by = c$?

Integer Programming: Given a set of linear inequalities of the form $a_{i,1}x_1 + a_{i,2}x_2 + \ldots + a_{i,n}x_n \leq b_i$, are there INTEGERS $x_1, x_2, \ldots, x_n$ which satisfy all these inequalities?

See *Computers and Intractibility: A Guide to the Theory of NP-Completeness* by Garey and Johnson, 1979.

Independent Set

Independent Set: Given a graph G and an integer k, is there a set S of k vertices in G such that no two vertices in S are adjacent?

Independent set is an NP-complete problem.

Independent Set of Size 5: Given a graph G, is there a set S of 5 vertices in G such that no two vertices in S are adjacent?

Independent Set of Size 5 can be solved in $O(n^5)$ time and is a problem in P.

Reductions

NP-Complete

Definition. A decision problem Q is NP-complete if

- Q is in NP;
- Every problem in NP can be reduced to Q in polynomial time.

Reduction

A **reduction** is a transformation of one problem into another.

Definition. A **reduction** of decision problem Q_1 to decision problem Q_2 is a mapping of every instance q_1 of problem Q_1 to an instance q_2 of problem Q_2 such that q_1 is **yes** if and only if q_2 is **yes**.

NP-Complete

Definition. A decision problem Q is NP-complete if

- Q is in NP;
- Every problem in NP can be reduced to Q in polynomial time.

How is it possible to know that every problem in NP can be reduced to Q in polynomial time?

The "First" NP-Complete Problem

Boolean Satisfiability: Given a boolean expression (i.e. $(x_1 \vee x_2) \wedge (\neg x_2 \vee x_3 \vee \neg x_4) \wedge \ldots$), is there an assignment of true or false to the variables x_i so that the expression is true.

Theorem (Cook's Theorem, 1971)**.** Boolean Satisfiability is NP-complete.

NP-Complete Problems

To show that a decision problem Q is NP-complete:

- Show that Q is verifiable in polynomial time;
- Show that Boolean satisfiability reduces to Q in polynomial time.

Reduction Transitivity

Proposition. If Q_1 reduces to Q_2 in polynomial time and Q_2 reduces to Q_3 in polynomial time, then Q_1 reduces to Q_3 in polynomial time.

Proposition. If Q_1 reduces to Q_2 in polynomial time and Q_2 reduces to Q_3 in polynomial time, then Q_1 reduces to Q_3 in polynomial time.

Proof. Let f_1 be the mapping of Q_1 to Q_2.
Let f_2 be the mapping of Q_2 to Q_3.
Define $f_3(q_1) = f_2(f_1(q_1))$.
Function f_3 maps $q_1 \in Q_1$ to $f_3(q_1) \in Q_3$.
Since f_1 and f_2 can be computed in polynomial time, $f_3(q_1)$ can be computed in polynomial time.
q_1 is yes $\Rightarrow f_1(q_1)$ is yes $\Rightarrow f_2(f_1(q_1))(= f_3(q_1))$ is yes.
q_1 is no $\Rightarrow f_1(q_1)$ is no $\Rightarrow f_2(f_1(q_1))(= f_3(q_1))$ is no.
Therefore, Q_1 reduces to Q_3 in polynomial time. $\square$

NP-Complete Problems

To show that a decision problem Q is NP-complete:

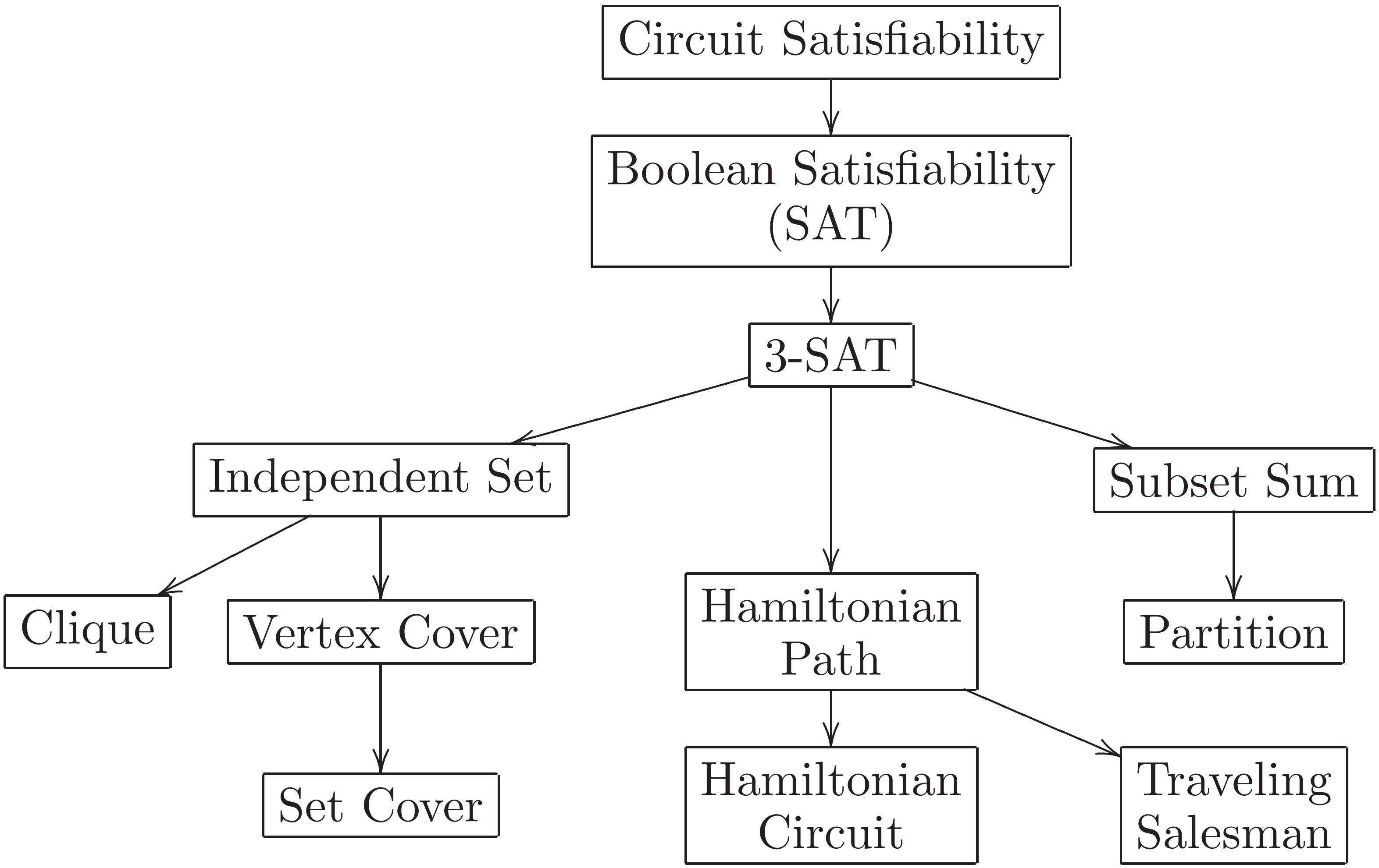
- Show that Q is verifiable in polynomial time;
- Show that Boolean satisfiability reduces to Q in polynomial time.

OR

To show that a decision problem Q is NP-complete:

- Show that Q is verifiable in polynomial time;
- Show that SOME NP-complete problem reduces to Q in polynomial time.

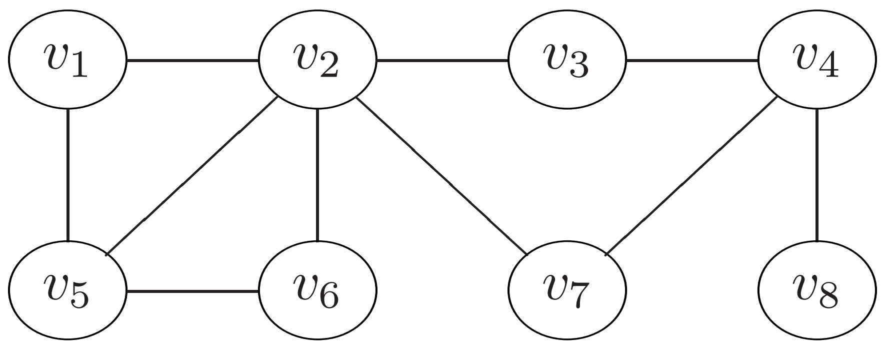
NP-Complete Reductions



Independent Set

Definition. An **independent set** is a subset V' of the vertices of graph G such that if both u and v are in V' then (u, v) is NOT an edge of G.

Example:



Set $\{v_2, v_4\}$ is an independent set.
Set $\{v_1, v_6, v_3, v_8\}$ is an independent set.
Vertex set $\{v_1, v_6, v_4, v_7\}$ is NOT an independent set since (v_4, v_7) is an edge of G.

Independent Set

Independent Set Problem: Given a graph G and an integer K, does G contain an independent set of size K?

Proposition. The Independent Set Problem is in NP.

Proof. A solution to the independent set problem is a subset V' of the vertices of G.

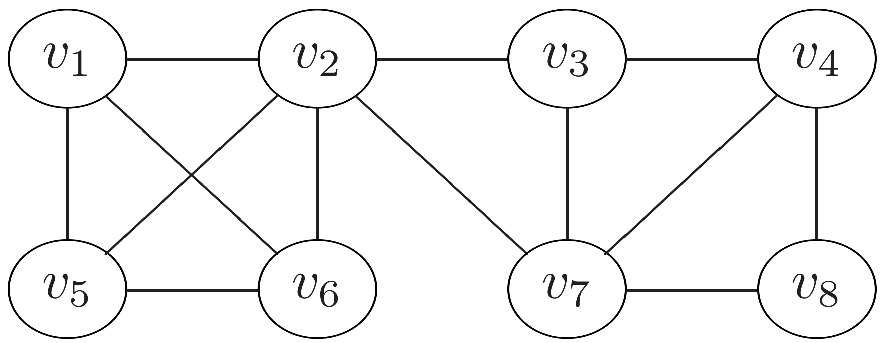
Let n be the number of vertices of G.
Let m be the number of edges of G.

Given a subset V', check if V' has K elements and for each edge (u, v) of G whether both u and v are in V'.
Checking each edge against V' takes $O(mn)$ time which is polynomial in the size of the input. $\square$

Clique

Definition. A **clique** is a subset V' of the vertices of graph G such that for each $u, v \in V'$, pair (u, v) is an edge of G.

Example:



Vertex set $\{v_1, v_2, v_5, v_6\}$ is a clique.
Vertex set $\{v_2, v_3, v_7\}$ is a clique.
Vertex set $\{v_3, v_4, v_7, v_8\}$ is NOT a clique since (v_3, v_8) is not an edge.

Clique

Clique Problem: Given a graph G and an integer M, does G contain a clique of size M?

Proposition. The Clique Problem is in NP.

Proof. A solution to the clique problem is a subset V' of the vertices of G.

Let n be the number of vertices of G.
Let m be the number of edges of G.

Given a subset V', check if V' has M elements and if for every $u, v \in V'$, the pair (u, v) is an edge of G.
Checking if every $u, v \in V'$ is an edge takes $O(n^2 m)$ time which is polynomial in the size of the input. □

Independent Set and Clique

Independent Set Problem: Given a graph G and an integer K, does G contain an independent set of size K?

Clique Problem: Given a graph G and an integer M, does G contain a clique of size M?

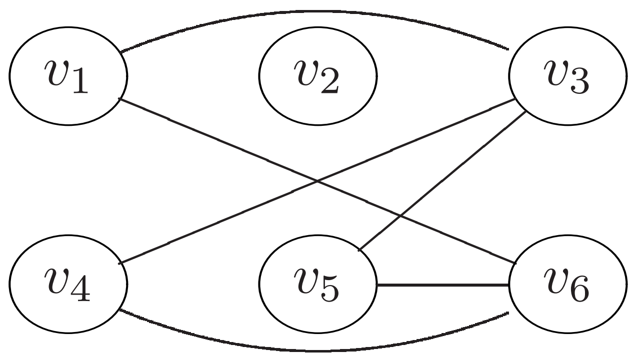
Proposition. The Independent Set Problem reduces to the Clique Problem in polynomial time.

Graph Complement

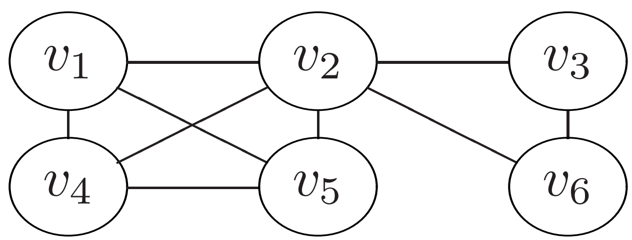
Definition. The **complement** of graph G is a graph $\overline{G}$ with the same vertices as G such that (u, v) is an edge of $\overline{G}$ if and only if (u, v) is NOT an edge of G.

Example:

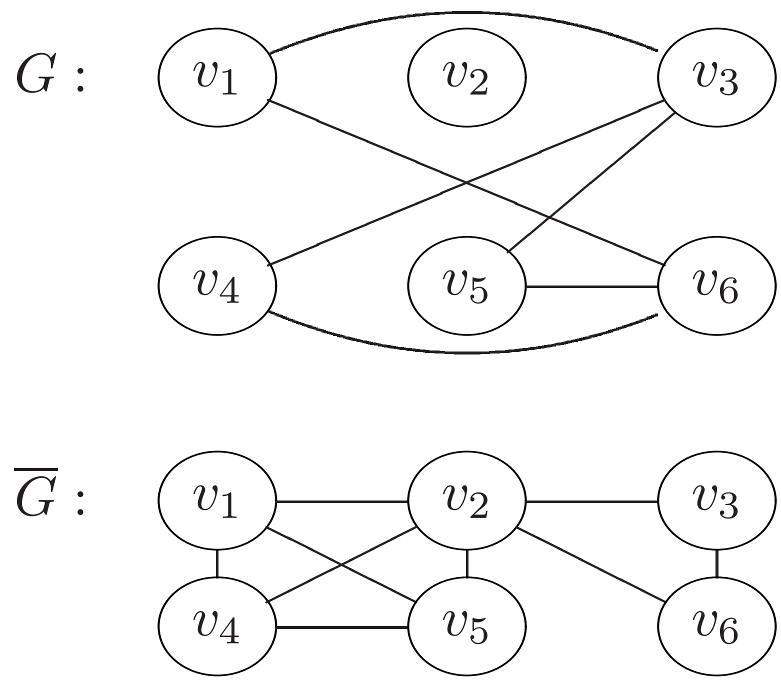
G :



$\overline{G}$:



Graph Complement



$\{v_1, v_2, v_4, v_5\}$ is an independent set of G.

$\{v_1, v_2, v_4, v_5\}$ is a clique of $\overline{G}$.

Graph Complement

Lemma.
If set V' is an independent set of G, then V' is a clique of $\overline{G}$.

Proof. Let V' be an independent set of G.
For every $u, v \in V'$, pair (u, v) is not an edge of G.
If (u, v) is not an edge of G, then (u, v) is an edge of $\overline{G}$.
For every $u, v \in V'$, pair (u, v) is an edge of $\overline{G}$.
Thus, V' is a clique of $\overline{G}$. $\square$

Graph Complement

Lemma.
If set V' is a clique of $\overline{G}$, then V' is an independent set of G.

Proof. Let V' be a clique of $\overline{G}$.
For every $u, v \in V'$, edge (u, v) is in $\overline{G}$.
If (u, v) is an edge of $\overline{G}$, then (u, v) is not an edge of G.
For every $u, v \in V'$, pair (u, v) is NOT an edge of G.
Thus, V' is an independent set of G. $\square$

Reduce Independent Set to Clique

Proposition. The Independent Set Problem reduces to the Clique Problem in polynomial time.

Proof. Let graph G and integer K be an instance of the independent set problem.
Let graph $\overline{G}$ be the complement of graph G.
Graph G has an independent set of size K if and only if graph $\overline{G}$ has a clique of size K.
Mapping (G, K) to $(\overline{G}, K)$ is a reduction from the independent set problem to the clique problem.
Since $\overline{G}$ can be computed $O(n^2)$ time, this is a polynomial time reduction. □

Reduction

Definition. A **reduction** of decision problem Q_1 to decision problem Q_2 is a mapping f of every instance q_1 of problem Q_1 to an instance $f(q_1)$ of problem Q_2 such that q_1 is **yes** if and only if $f(q_1)$ is **yes**.

(1) If q_1 is **yes**, then $f(q_1)$ is **yes**.

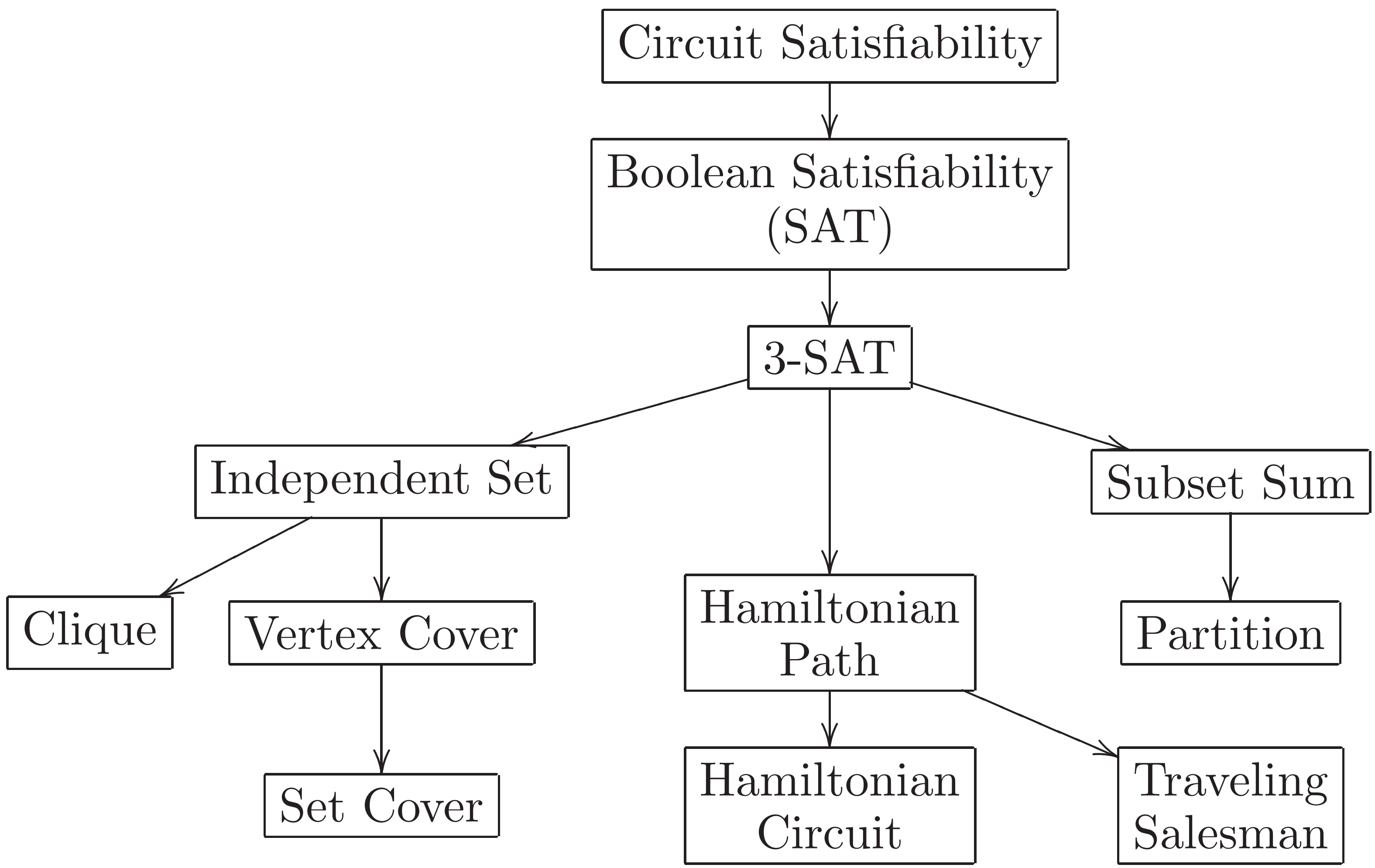
(2) If q_1 is **no**, then $f(q_1)$ is **no**.

OR

(1') If q_1 is **yes**, then $f(q_1)$ is **yes**.

(2') If $f(q_1)$ is **yes**, then q_1 is **yes**. (Contrapositive of 2.)

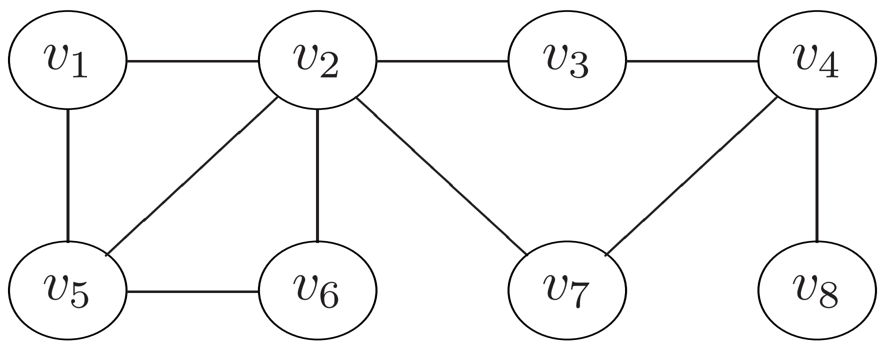
NP-Complete Reductions



Vertex Cover

Definition. A **vertex cover** is a subset V' of the vertices of graph G such that for each edge $(u, v) \in E(G)$, either u or v (or both) are in V'.

Example:



Vertex set $\{v_1, v_3, v_5, v_6, v_7, v_8, \}$ is a vertex cover.
Vertex set $\{v_2, v_4, v_5\}$ is a vertex cover.
Vertex set $\{v_2, v_3, v_8\}$ is NOT a vertex cover since edge (v_4, v_7) is not covered.

Vertex Cover

Vertex Cover Problem: Given a graph G and an integer M, does G contain a vertex cover of size M?

Proposition. The Vertex Cover Problem is in NP.

Proof. A solution to the vertex cover problem is a subset V' of the vertices of G.

Let n be the number of vertices of G.
Let m be the number of edges of G.

Given a subset V', check if V' has M elements and, for every edge (u, v) of G, check that either u or v are in V'.
Checking if either u or v are in V' for every edge (u, v) of G takes $O(mn)$ time which is polynomial in the size of the input.

□

Independent Set and Vertex Cover

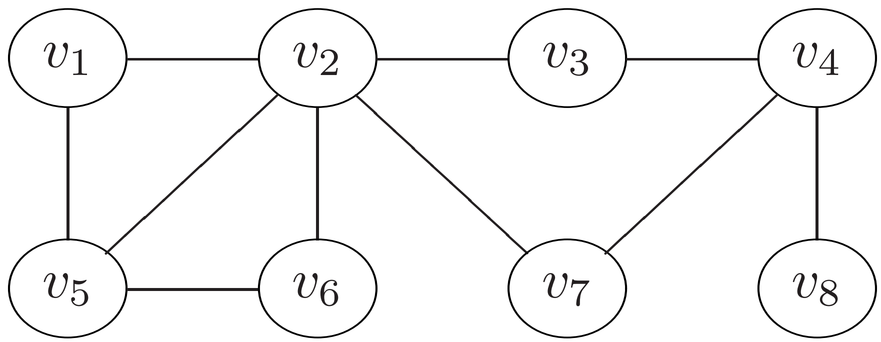
Independent Set Problem: Given a graph G and an integer K, does G contain an independent set of size K?

Vertex Cover Problem: Given a graph G and an integer M, does G contain a vertex cover of size M?

Proposition. The Independent Set Problem reduces to the Vertex Cover Problem in polynomial time.

Independent Set and Vertex Cover

Example:



$V' = \{v_1, v_3, v_6, v_7, v_8\}$ is an independent set of G.
$V(G) - V' = \{v_2, v_4, v_5\}$ is a vertex cover of G.

Reduce Independent Set to Vertex Cover

Lemma. If set V' is an independent set of G, then $V(G) - V'$ is a vertex cover of G.

Proof. Let V' be an independent set of G.
Let $W = V(G) - V'$.
Let (u, v) be an edge of G.
Since (u, v) is an edge of G, either u or v is not in V' (or both are not in V'.)
Since u or v is not in V', either u or v is in W.
Thus, for each edge $(u, v) \in E(G)$, either u or v is in W.
Thus, W is a vertex cover of G. $\square$

Reduce Independent Set to Vertex Cover

Lemma. If set $V(G) - V'$ is a vertex cover of G, then V' is an independent set of G.

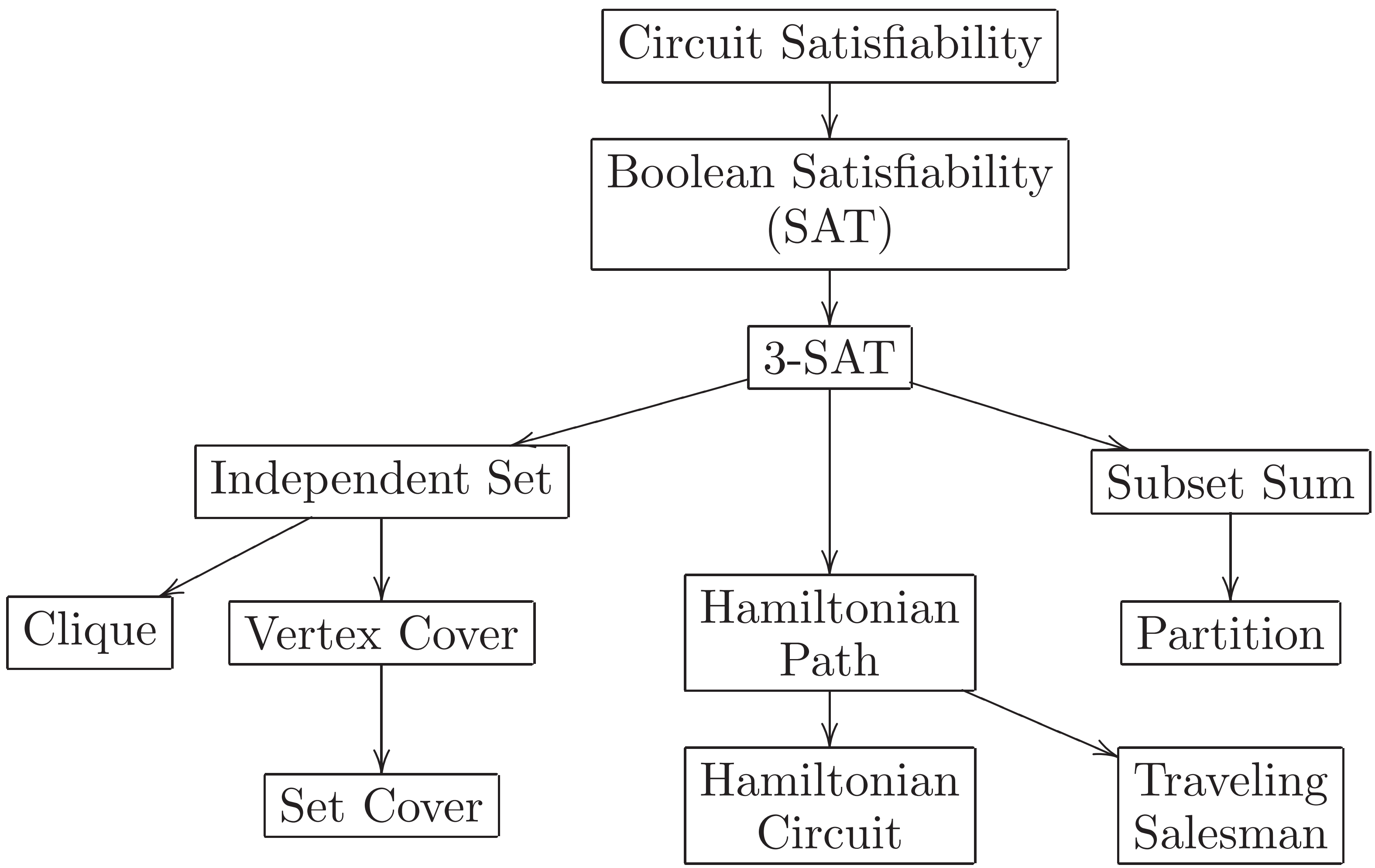
Proof. Let $W = V(G) - V'$ be a vertex cover of G.
Let u and v be two vertices in V'.
Since u and v are in V', they are not in W.
Since W is a vertex cover of G, pair (u, v) is not an edge in G.
Since (u, v) is not an edge of G for every $u, v \in V'$, set V' is an independent set. □

Reduce Independent Set to Vertex Cover

Proposition. The Independent Set Problem reduces to the Vertex Cover Problem in polynomial time.

Proof. Let graph G and integer K be an instance of the independent set problem.
Graph G has an independent set of size K if and only if graph G has a vertex cover of size $n - K$ where n is the number of vertices of G.
Mapping (G, K) to $(G, n - K)$ is a reduction from the independent set to the vertex cover problem.
Since G can be copied in $\Theta(n + m)$ time this is a polynomial time reduction. □

NP-Complete Reductions



Set Cover

Definition. Let U be a set of elements, and let $C = \{S_1, S_2, \ldots, S_n\}$ be a collection of subsets of U. A **set cover** of U is a subcollection $C' \subseteq C$ of these subsets whose union equals U.

Example:

$$U = \{1, 2, 3, 4, 5, 6, 7, 8\}$$

$$S_1 = \{1, 2, 3\} \qquad S_2 = \{1, 4, 5, 8\}$$

$$S_3 = \{2, 3, 4\} \qquad S_4 = \{6, 7\}$$

$$S_5 = \{2, 4, 6\} \qquad S_6 = \{2, 3, 5, 8\}$$

Subcollection $\{S_1, S_2, S_4, S_6\}$ is a set cover.
Subcollection $\{S_2, S_3, S_4\}$ is a set cover.
Subcollection $\{S_1, S_3, S_6\}$ is NOT a set cover since it is missing elements 6 and 7.

Set Cover

Set Cover Problem: Given a set U, a collection $S_1, S_2, \ldots, S_n$ of subsets of U and an integer K, is there a set cover of U of size K?

Proposition. The Set Cover Problem is in NP.

Vertex Cover and Set Cover

Vertex Cover Problem: Given a graph G and an integer M, does G contain a vertex cover of size M?

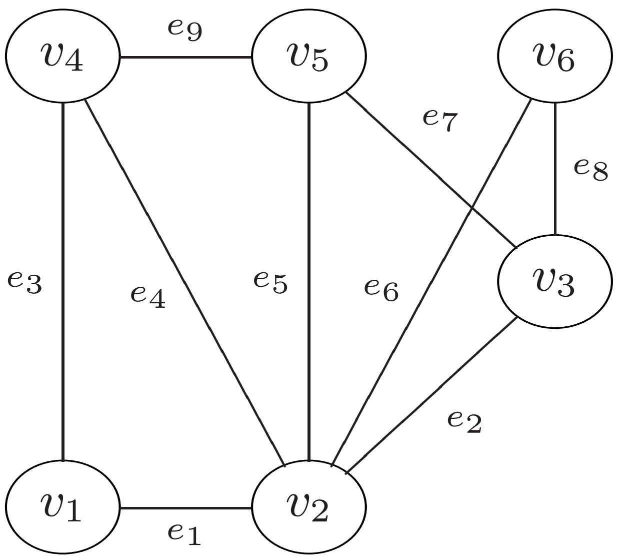
Set Cover Problem: Given a set U, a collection $S_1, S_2, \ldots, S_n$ of subsets of U and an integer K, is there a set cover of U of size K?

Proposition. The Vertex Cover Problem reduces to the Set Cover Problem in polynomial time.

Reduce Vertex Cover to Set Cover

Vertex Cover Problem: Given a graph G and an integer M, does G contain a vertex cover of size M?

Set Cover Problem: Given a set U, a collection $S_1, S_2, \ldots, S_n$ of subsets of U and an integer K, is there a set cover of U of size K?



$U = \{e_1, e_2, e_3, e_4, e_5, e_6, e_7, e_8, e_9\}$

$S_1 = \{e_1, e_3\}$
$S_2 = \{e_1, e_2, e_4, e_5, e_6\}$
$S_3 = \{e_2, e_7, e_8\}$
$S_4 = \{e_3, e_4, e_9\}$
$S_5 = \{e_5, e_7, e_9\}$
$S_6 = \{e_7, e_8\}$

Reduce Vertex Cover to Set Cover

Proposition. The Vertex Cover Problem reduces to the Set Cover Problem in polynomial time.

Proof. Let graph G and integer M be an instance of the vertex cover problem.

Let U equal $E(G)$, the edges of G.

Let $S_i = \{e_j : v_i \text{ is a vertex of } e_j\}$.

Let the collection C be $\{S_1, S_2, \ldots, S_n\}$.

G has a vertex cover of size M if and only if U has a set cover of size M.

Mapping (G, M) to (U, C, M) is a reduction from the vertex cover to the set cover problem.

Since U and C can be computed in $O(nm)$ time, this is a polynomial time reduction. □

Reduction of Vertex Cover to Set Cover

Vertex Cover Problem: Given a graph G and an integer M, does G contain a vertex cover of size M?

Set Cover Problem: Given a set U, a collection $S_1, S_2, \ldots, S_n$ of subsets of U and an integer K, is there a set cover of U of size K?

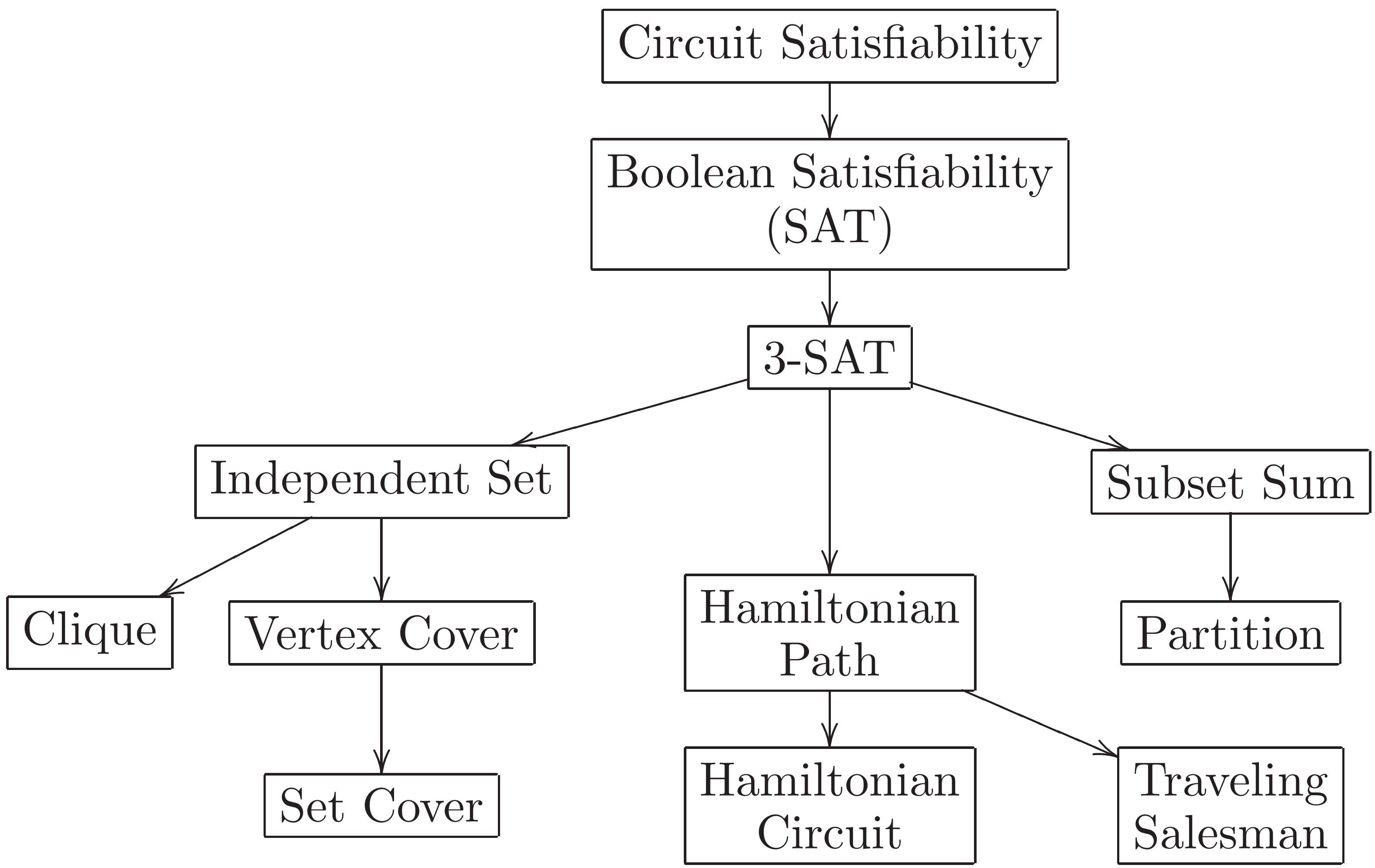
The reduction in the previous slide actually reduces Vertex Cover to a special case of Set Cover Problem, where every element is in exactly two sets.

The reduction is not onto (not all instances of the Set Cover Problem are in images of an instance of Vertex Cover.)

The reduction is not reversible (i.e., it cannot be used to map Set Cover to Vertex Cover.)

Note: Since Vertex Cover is NP-complete, there is some reduction of Set Cover to Vertex Cover but it's much more complicated than the reduction of Vertex Cover to Set Cover.

NP-Complete Reductions



Boolean Expressions

x_1 AND x_2: $x_1 \wedge x_2$

x_1 OR x_2: $x_1 \vee x_2$

NOT x: $\overline{x}$

A literal is x_i or $\overline{x_i}$.

A boolean clause is an "OR" of literals:

$$(x_1 \vee \overline{x_3} \vee x_4).$$

A boolean expression in conjunctive normal form is an "AND" of boolean clauses:

$$(x_1 \vee \overline{x_3} \vee x_4) \wedge (\overline{x_2} \vee x_3) \wedge (\overline{x_1} \vee x_2 \vee x_3).$$

Boolean Expressions

A boolean expression in 3-CNF (conjunctive normal form) is an "AND" of boolean clauses where each clause has EXACTLY 3 literals.

$$(x_1 \vee \overline{x_3} \vee x_4) \wedge (\overline{x_2} \vee x_3 \vee \overline{x_4}) \wedge (\overline{x_1} \vee x_2 \vee x_3).$$

A truth assignment is an assignment of true (T) or false (F) to each variable.

$$x_1 = T \qquad x_2 = T \qquad x_3 = F \qquad x_4 = F$$

$$(x_1 \vee \overline{x_3} \vee x_4) \wedge (\overline{x_2} \vee x_3 \vee \overline{x_4}) \wedge (\overline{x_1} \vee x_2 \vee x_3)$$
$$(T \vee T \vee F) \wedge (F \vee F \vee T) \wedge (F \vee T \vee F) = T.$$

Boolean Expressions

$$(x_1 \vee \overline{x_3} \vee x_4) \wedge (\overline{x_2} \vee x_3 \vee \overline{x_4}) \wedge (\overline{x_1} \vee x_2 \vee x_3).$$

A truth assignment is an assignment of true (T) or false (F) to each variable.

$$x_1 = F \qquad x_2 = T \qquad x_3 = F \qquad x_4 = T$$

$$(x_1 \vee \overline{x_3} \vee x_4) \wedge (\overline{x_2} \vee x_3 \vee \overline{x_4}) \wedge (\overline{x_1} \vee x_2 \vee x_3)$$

$$(F \vee T \vee T) \wedge (F \vee F \vee F) \wedge (T \vee T \vee F) = F.$$

Boolean Expressions

A truth assignment **satisfies** a boolean expression is it makes the boolean expression evaluate to true.

Give a truth assignment which satisfies the following expression:

$$(\overline{x_1} \vee x_2 \vee x_4) \wedge (x_1 \vee \overline{x_2} \vee x_3) \wedge (x_2 \vee \overline{x_3} \vee \overline{x_4}).$$

$x_1 =$

$x_2 =$

$x_3 =$

$x_4 =$

Satisfiability

A truth assignment **satisfies** a boolean expression is it makes the boolean expression evaluate to true.

Not all boolean expressions can be satisfied.

$$x_1 \wedge (\overline{x_1} \vee \overline{x_2}) \wedge (\overline{x_1} \vee x_2 \vee x_3) \wedge \overline{x_3}$$

3-SAT

3-SAT: Given a boolean expression in 3-CNF (conjunctive normal form), is there a truth assignment which makes the boolean expression true?

(A truth assignment **satisfies** a boolean expression is it makes the boolean expression evaluate to true.)

Proposition. 3-SAT is in NP.

3-SAT and Independent Set

3-SAT: Given a boolean expression in 3-CNF (conjunctive normal form), is there a truth assignment which makes the boolean expression true?

Independent Set Problem: Given a graph G and an integer K, does G contain an independent set of size K?

(An **independent set** is a subset V' of the vertices of graph G such that if both u and v are in V' then (u, v) is NOT an edge of G.)

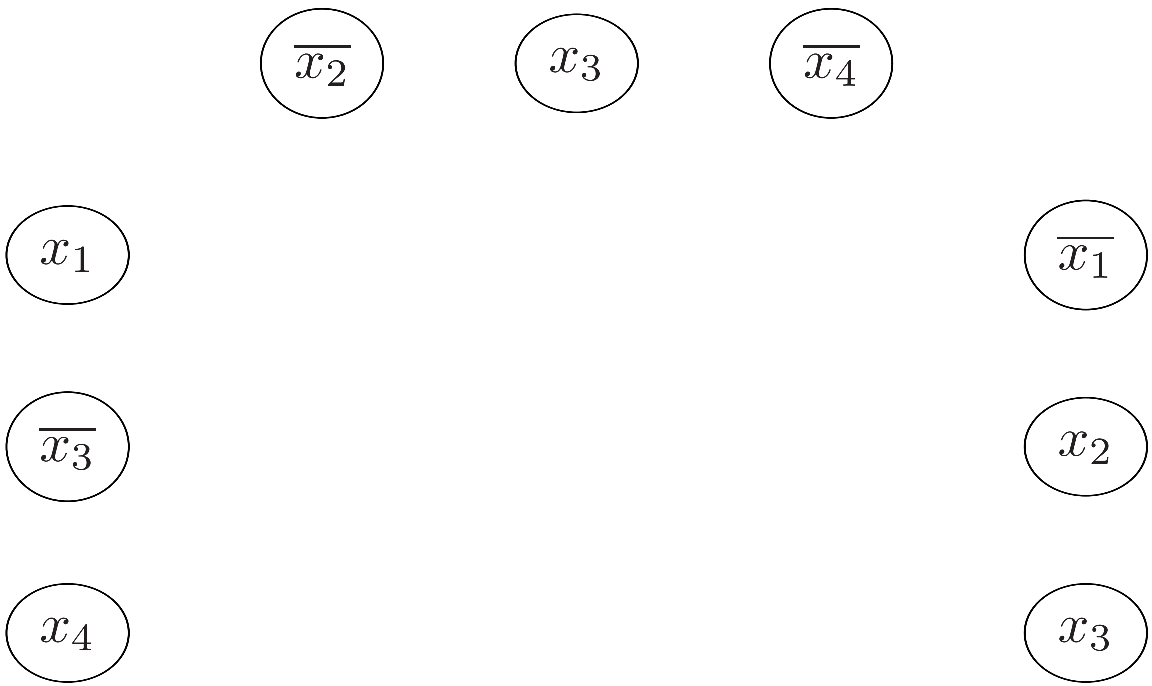
Proposition. 3-SAT reduces to the Independent Set Problem in polynomial time.

Reduce 3-SAT to Independent Set

An instance of 3-SAT is a boolean expression in 3-CNF form.

- For each literal create a vertex;

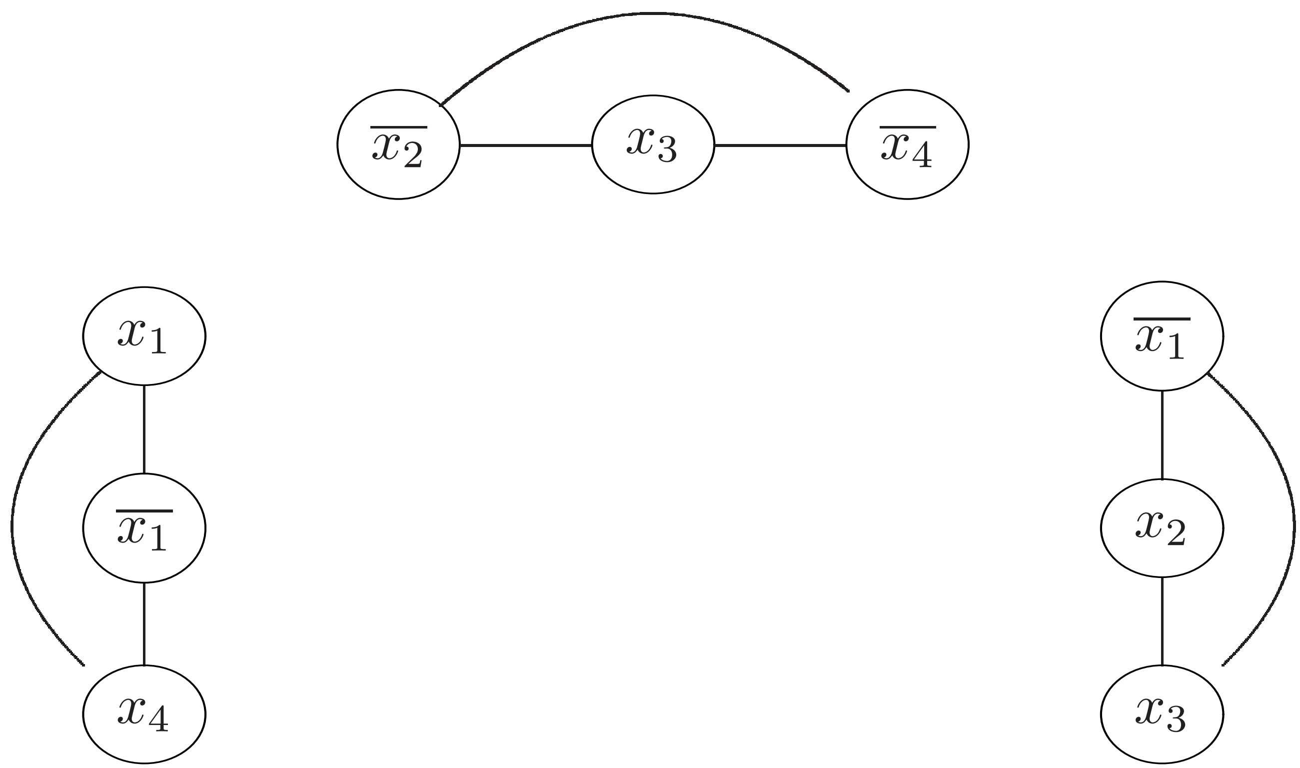
$$(x_1 \vee \overline{x_3} \vee x_4) \wedge (\overline{x_2} \vee x_3 \vee \overline{x_4}) \wedge (\overline{x_1} \vee x_2 \vee x_3).$$



Reduce 3-SAT to Independent Set

- Connect each literal to the two other literals in the same clause;

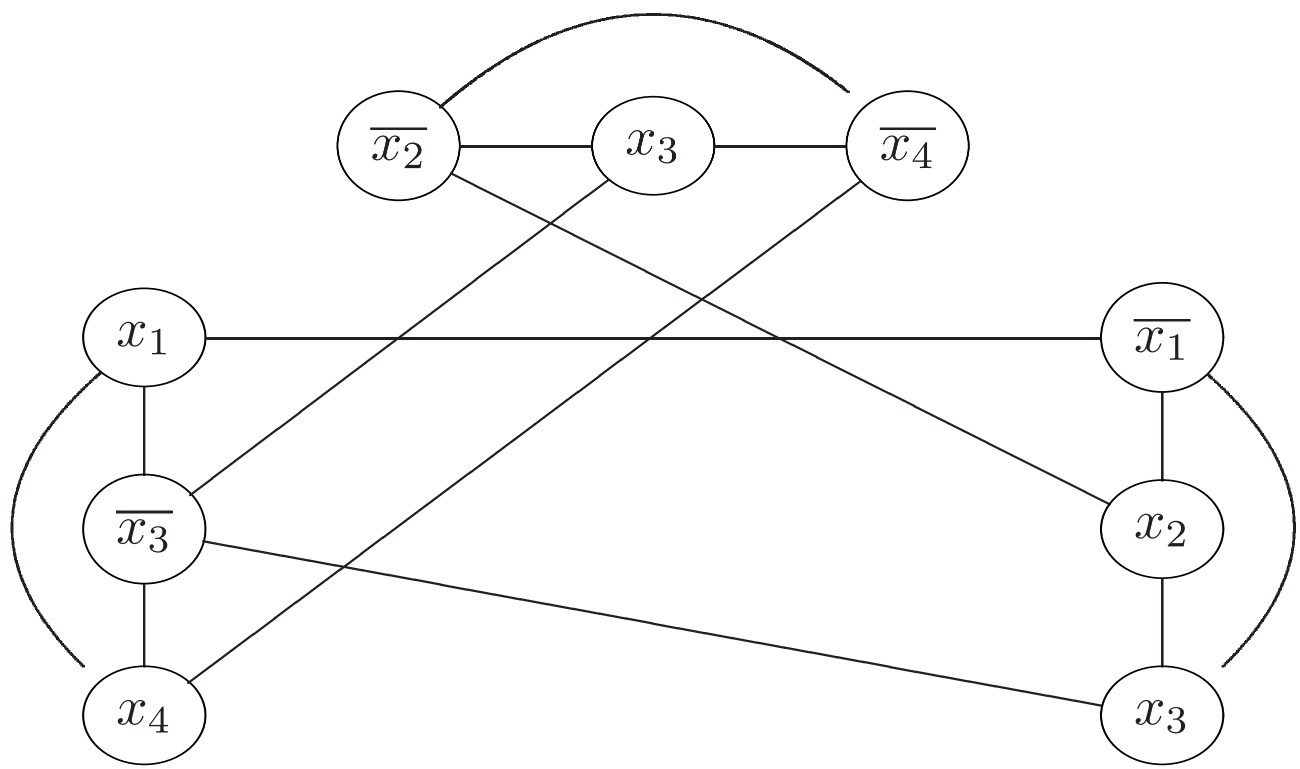
$$(x_1 \vee \overline{x_3} \vee x_4) \wedge (\overline{x_2} \vee x_3 \vee \overline{x_4}) \wedge (\overline{x_1} \vee x_2 \vee x_3).$$



Reduce 3-SAT to Independent Set

- Connect each literal to the two other literals in the same clause;
- Connect each literal x_i to $\overline{x_i}$;

$$(x_1 \vee \overline{x_3} \vee x_4) \wedge (\overline{x_2} \vee x_3 \vee \overline{x_4}) \wedge (\overline{x_1} \vee x_2 \vee x_3).$$



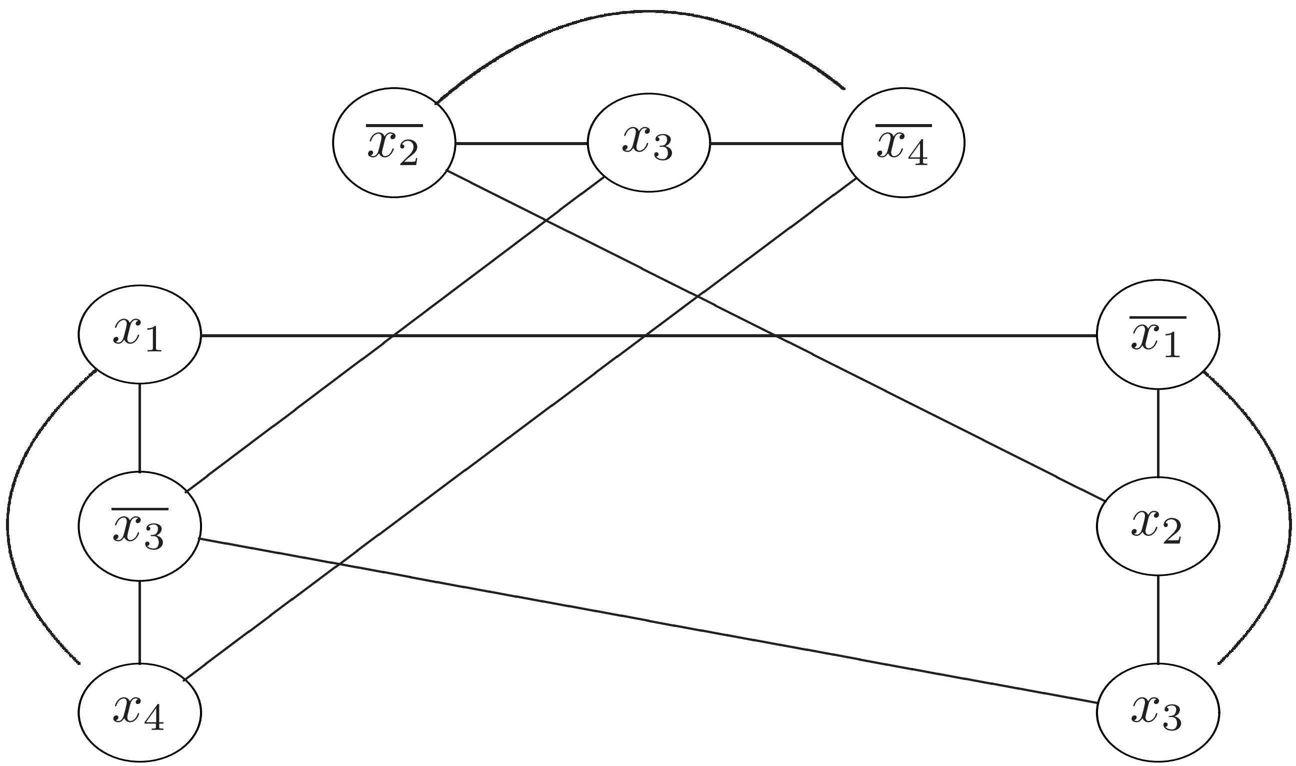
Reduce 3-SAT to Independent Set

An instance of 3-SAT is a boolean expression in 3-CNF form.

- For each literal create a vertex;
- Connect each literal to the two other literals in the same clause;
- Connect each literal x_i to $\overline{x_i}$;

Reduce 3-SAT to Independent Set

$$(x_1 \vee \overline{x_3} \vee x_4) \wedge (\overline{x_2} \vee x_3 \vee \overline{x_4}) \wedge (\overline{x_1} \vee x_2 \vee x_3).$$



Reduce 3-SAT to Independent Set

Proposition. 3-SAT reduces to the Independent Set Problem in polynomial time.

Proof. An instance of 3-SAT is a boolean expression ϕ in 3-CNF form.
$n =$ number of variables in ϕ.
$m =$ number of clauses in ϕ.

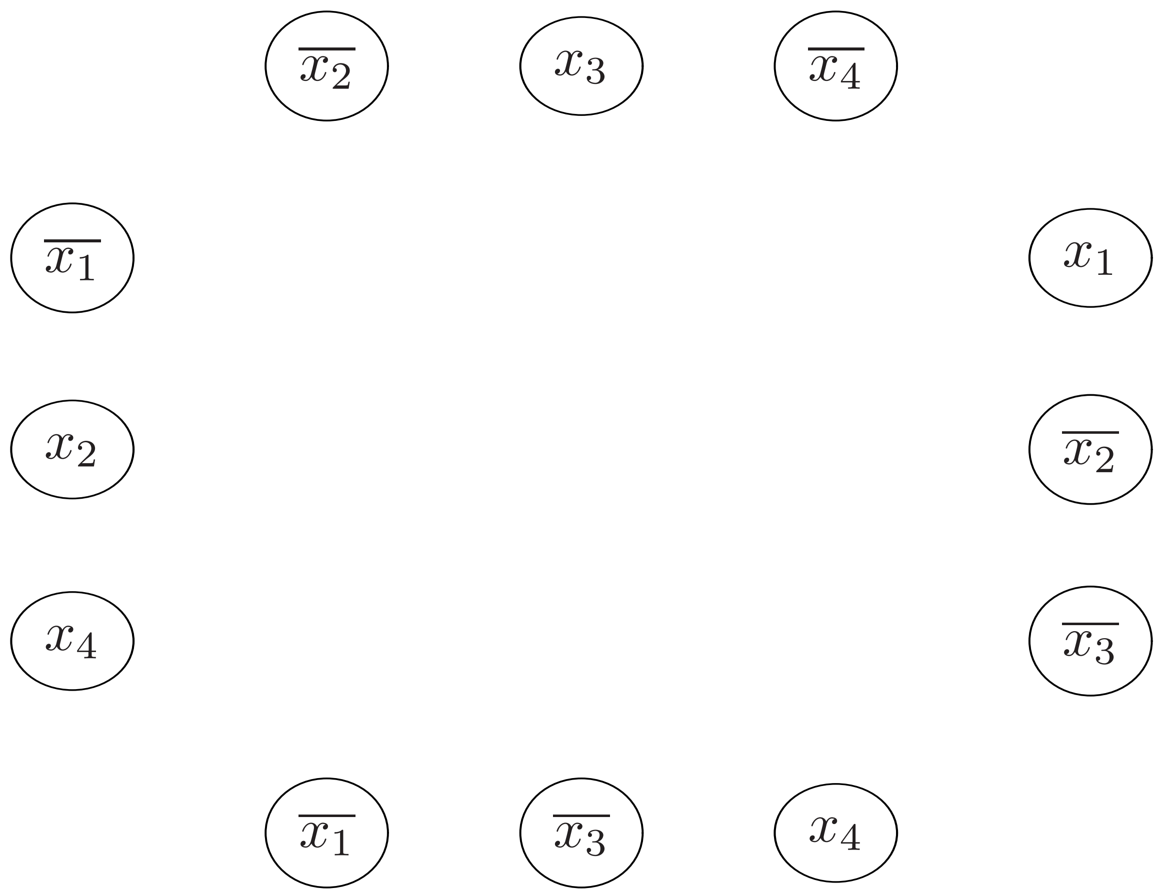
- For each literal create a vertex;
- Connect each literal to the two other literals in the same clause;
- Connect each literal x_i to $\overline{x_i}$;

There an assignment of true or false to the variables x_i so that ϕ is true if and only if the graph has an independent set of size m.

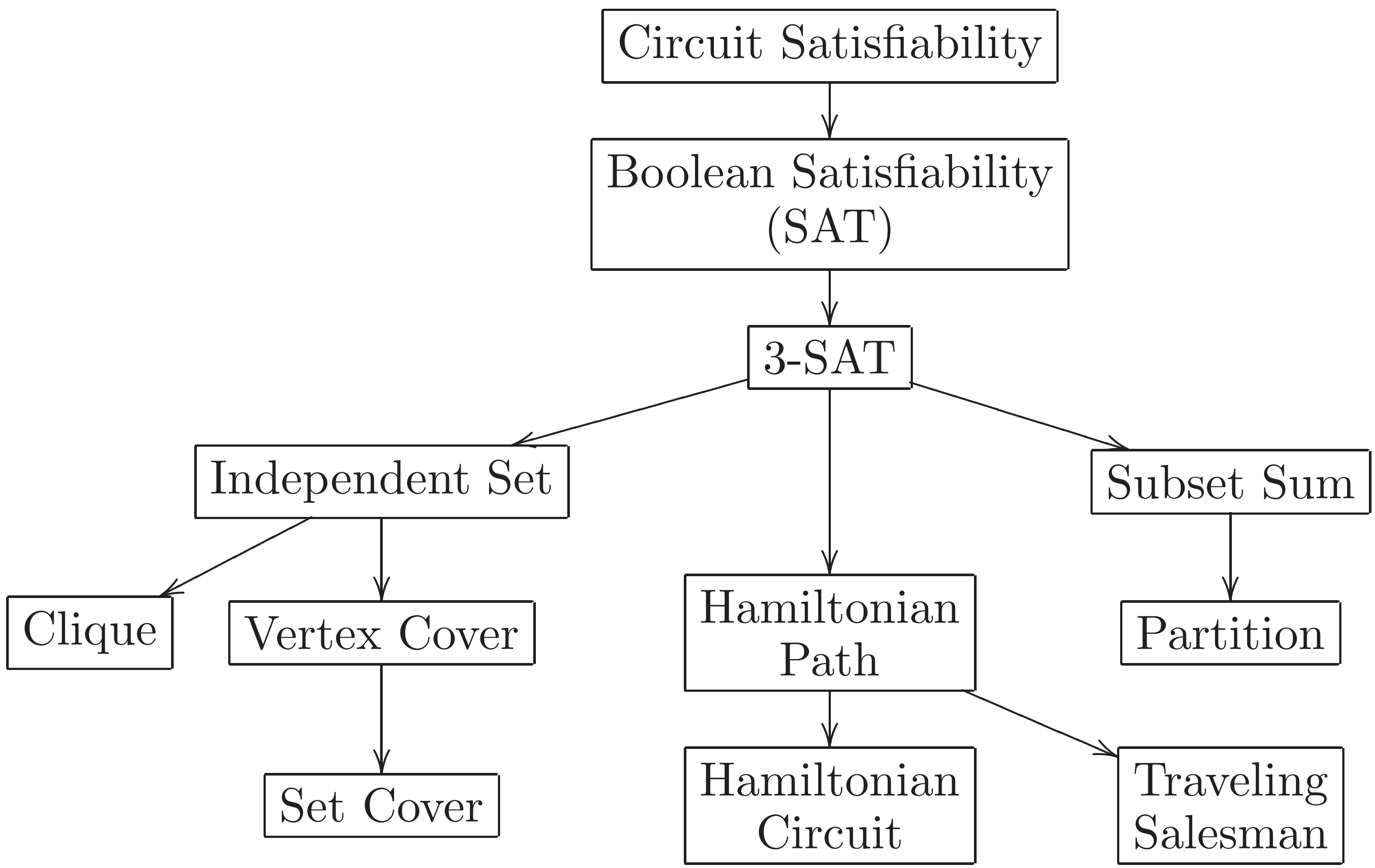
Since the graph can be constructed in polynomial time, the reduction takes polynomial time. □

Reduce 3-SAT to Independent Set: Exercise

$$(\overline{x_1} \vee x_2 \vee x_4) \wedge (\overline{x_2} \vee x_3 \vee \overline{x_4}) \wedge (x_1 \vee \overline{x_2} \vee \overline{x_3}) \wedge (\overline{x_1} \vee \overline{x_3} \vee x_4)$$



NP-Complete Reductions



Hamiltonian Cycle

Hamiltonian Path Problem: Given a graph G, is there a path from v_1 to v_n which visits each vertex exactly once?

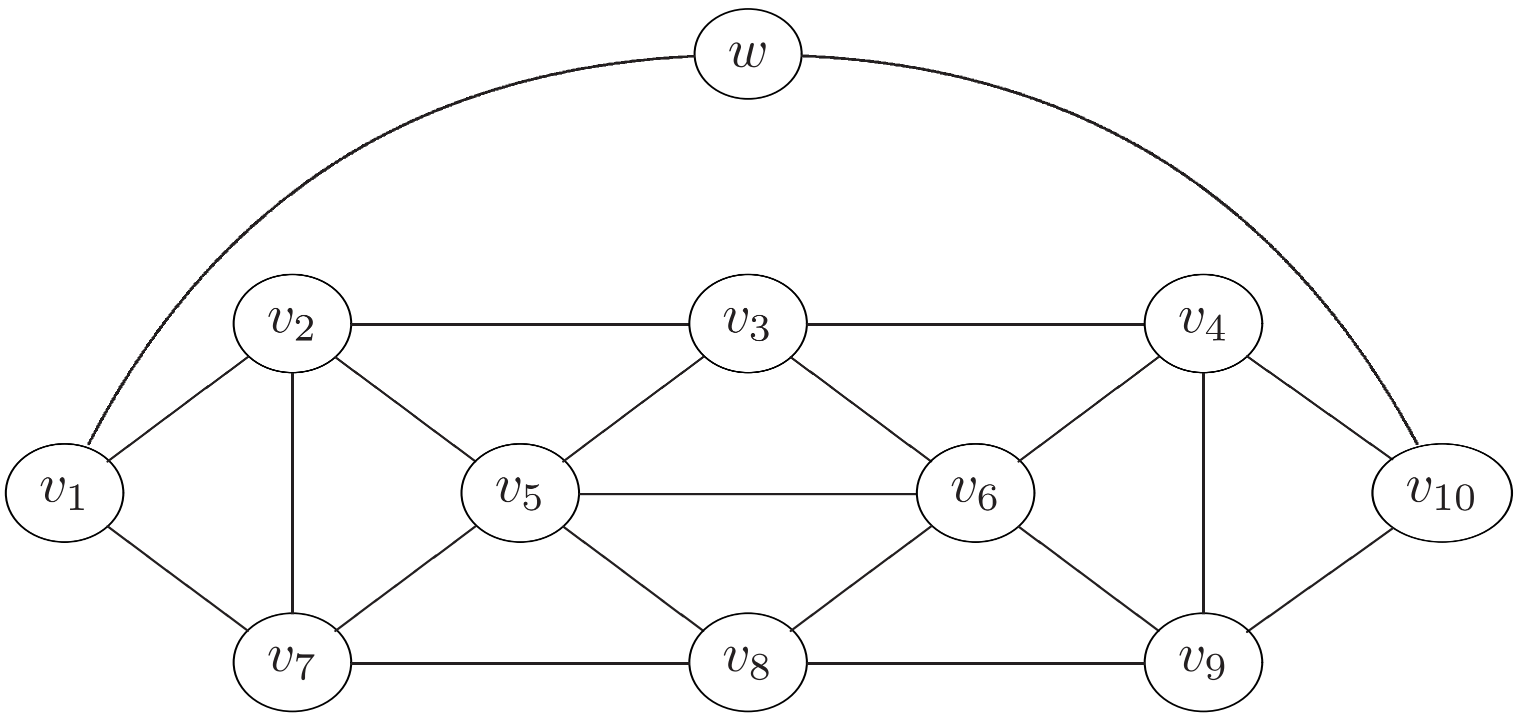
Hamiltonian Cycle: Given a graph G, is there a cycle which visits each vertex exactly once?

Proposition. The Hamiltonian path problem reduces to the Hamiltonian cycle problem in polynomial time.

Reduce Hamiltonian Path to Hamiltonian Cycle

An instance of the Hamiltonian path problem is a graph G and vertices v_1 and v_n.

Form a new graph G' by adding a new vertex w to G and connecting w to v_1 and v_n.



Reduce Hamiltonian Path to Hamiltonian Cycle

Proposition. The Hamiltonian path problem reduces to the Hamiltonian cycle problem in polynomial time.

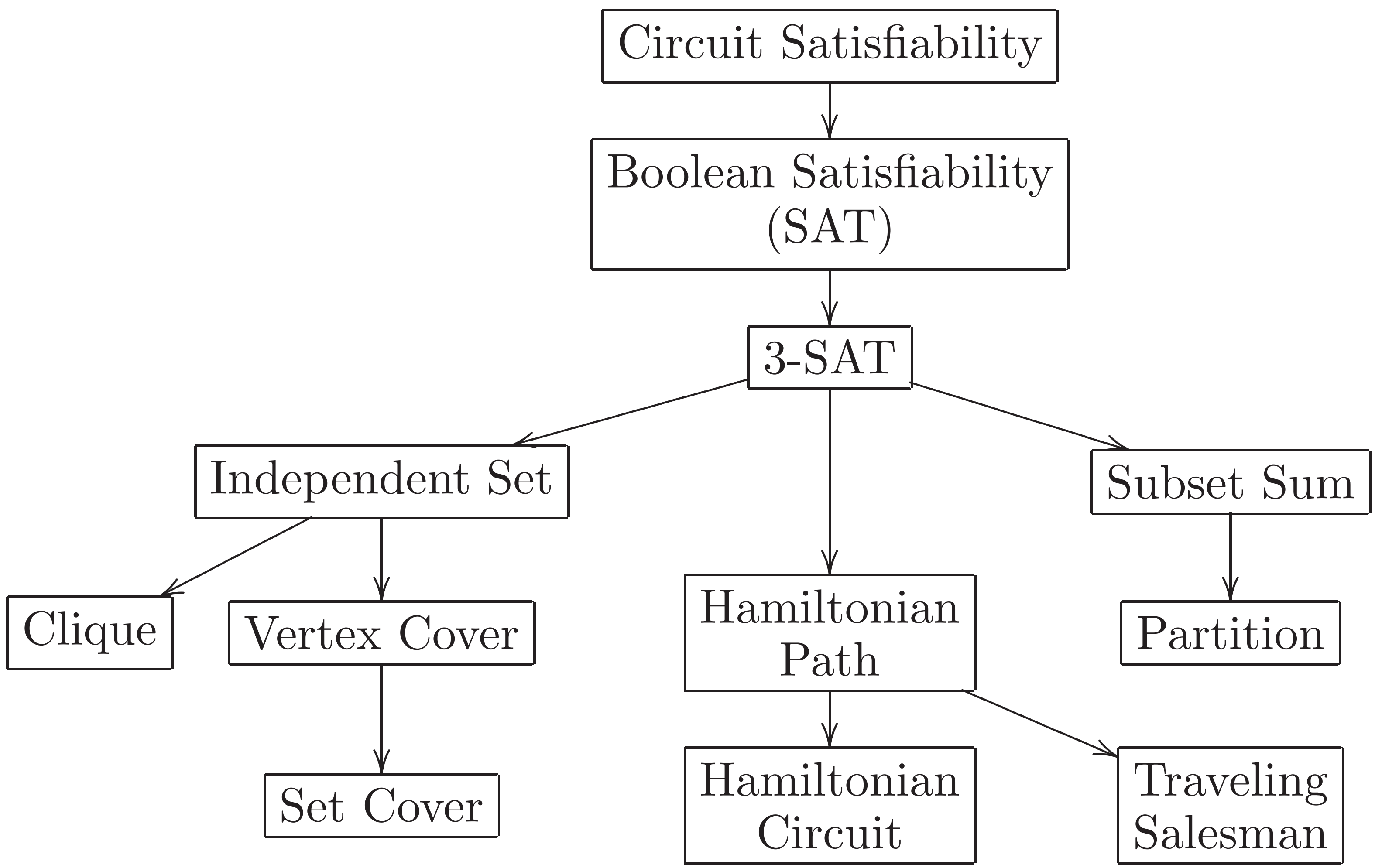
Proof. An instance of the Hamiltonian path problem is a graph G and vertices v_1 and v_n.

Form a new graph G' by adding a new vertex w to G and connecting w to v_1 and v_n.

Graph G has a path from v_1 to v_n which visits each vertex once if and only if graph G' has a Hamiltonian cycle.

Since graph G' can be computed in polynomial time, this is a polynomial time reduction. □

NP-Complete Reductions



Subset Sum

Subset Sum: Given a set of positive integers $K = \{k_1, k_2, \ldots, k_n\}$ and an integer M, is there a subset of K whose sum equals M?

Example:

$K = \{12, 15, 23, 32, 41, 61, 66\}$.
$M = 140$.

Is there a subset J of K whose sum equals 140?

Partition

Partition: Given a set of positive integers $A = \{a_1, a_2, \ldots, a_n\}$, is there a partition of A into subsets B_1 and B_2 such that $\sum_{x \in B_1} x = \sum_{y \in B_2} y$?

Example:

$K = \{12, 16, 29, 34, 41, 50, 58\}$.

Is there a partition of K into two subsets B_1 and B_2 such that $\sum_{x \in B_1} x = \sum_{y \in B_2} y$?

Subset Sum and Partition

Subset Sum: Given a set of positive integers $K = \{k_1, k_2, \ldots, k_n\}$ and an integer M, is there a subset of K whose sum equals M?

Partition: Given a set of positive integers $A = \{a_1, a_2, \ldots, a_n\}$, is there a partition of A into subsets B_1 and B_2 such that $\sum_{x \in B_1} x = \sum_{y \in B_2} y$?

Proposition. The Subset Sum problem reduces to the Partition problem in polynomial time.

Reduce Subset Sum to Partition: Example

$K = \{12, 15, 23, 32, 41, 61, 66\}$. $M = 140$.

Is there a subset J of K whose sum equals 140?

$\sum_{k \in K} k = 12 + 15 + 23 + 32 + 41 + 61 + 66 = 250$.

$\left(\sum_{k \in K} k\right) - 140 = 250 - 140 = 110$.

Let $A = K \cup \{1110, 1140\} = \{12, 15, 23, 32, 41, 61, 66, 1110, 1140\}$.

$J = \{15, 23, 41, 61\}$ and $15 + 23 + 41 + 61 = 140$.

$B_1 = \{15, 23, 41, 61, 1110\}$ and $B_2 = \{12, 32, 66, 1140\}$ is a partition of A.

$(15 + 23 + 41 + 61) + 1110 = 140 + 1110 = 1250$.

$(12 + 32 + 66) + 1140 = (250 - 140) + 1140 = 110 + 1140 = 1250$.

Reduce Subset Sum to Partition: Example

Lemma. Let K be a set of positive integers, let M be an integer and let $I = \sum_{k \in K} k$.
Let $A = K \cup \{4 * I + M, 5 * I - M\}$.
If K has a subset J whose sum is M, then A has a partition into subsets B_1 and B_2 such that $\sum_{x \in B_1} x = \sum_{y \in B_2} y$.

Proof.

Let $B_1 = J \cup \{5 * I - M\}$.
Let $B_2 = K - J \cup \{4 * I + M\}$.
Sets B_1 and B_2 partition A.
$\sum_{x \in B_1} x = \left(\sum_{k \in J} k\right) + (5 * I - M) = M + (5 * I - M) = 5 * I$.
$\sum_{y \in B_2} y = \left(\sum_{k \in K-J} k\right) + (4 * I + M) = (I - M) + (4 * I + M) = 5 * I$.
Therefore, $\sum_{x \in B_1} x = \sum_{y \in B_2} y$. □

Reduce Subset Sum to Partition: Example

Lemma. Let K be a set of positive integers, let M be an integer and let $I = \sum_{k \in K} k$.
Let $A = K \cup \{4 * I + M, 5 * I - M\}$.
If A has a partition into subsets B_1 and B_2 such that $\sum_{x \in B_1} x = \sum_{y \in B_2} y$, then K has a subset J whose sum is M.

Proof. $\sum_{a \in A} a = \left(\sum_{k \in K}\right) + (5 * I - M) + (4 * I + M) = 10 * I$.
$\sum_{x \in B_1} x + \sum_{y \in B_2} y = \sum_{a \in A} a = 10 * I$.
$\sum_{x \in B_1} x + \sum_{y \in B_2} y = 2 \sum_{x \in B_1} x$ since $\sum_{x \in B_1} x = \sum_{y \in B_2} y$.
Thus, $\sum_{x \in B_1} x = 10 * I/2 = 5 * I$.
Since $\sum_{x \in B_1} x = 5 * I$, set B_1 contains $(4 * I + M)$ or $(5 * I - M)$ but not both.
Without loss of generality, assume that B_1 contains $(5 * I - M)$.

Let $J = B_1 - \{5 * I - M\}$.
$\sum_{k \in J} k = \left(\sum_{x \in B_1} x\right) - (5 * I - M) = 5 * I - (5 * I - M) = M$. □

Reduce Subset Sum to Partition

Proposition. The Subset Sum problem reduces to the Partition problem in polynomial time.

Proof. An instance of the Subset Sum problem is a set K of positive integers and an integer M.
Let $I = \sum_{k \in K} k$.
Let $A = K \cup \{5 * I - M, 4 * I + M\}$.
Set K has a subset J whose sum equals M, if and only if A has a partition into subsets B_1 and B_2 such that $\sum_{x \in B_1} x = \sum_{y \in B_2}$.
Since A can be computed in polynomial time, this is a polynomial time reduction. $\square$

NP-Completeness Summary

- A **decision problem** has a **yes** or **no** answer.
- A decision problem is in the class NP, if it is verifiable in polynomial time. (NP = non-deterministic polynomial.)
- A **reduction** is a transformation of one problem into another.
- A decision problem Q is NP-complete if
  - Q is in NP;
  - Every problem in NP can be reduced to Q in polynomial time.
- If any NP-complete problem can be solved in polynomial time, then all NP-complete problems (and all problems in NP) can be solved in polynomial time;
- There is no known polynomial time algorithm for any NP-complete problem.

Factoring

Factoring

Factoring Problem: Given an integer X, find its factors.

Example:

What are the factors of 169,627,128,197?

Factoring

Factoring Problem: Given an integer X, find its factors.

- The factoring problem is in NP;
- No known polynomial time algorithm for factoring large numbers (polynomial in the number of digits);
- NP-complete problems do NOT seem to reduce to the factoring problem. (Even if NP-complete problems cannot be solve in polynomial time, there may be a polynomial time factoring algorithm.)

Cryptography

Factoring Problem: Given an integer X, find its factors.

- Modern cryptographic systems are based on the assumption that factoring is hard;
- Public key cryptosystems are based on the assumption that factoring is hard;
- If NP-complete problems can be solved in polynomial time, then numbers can be factored in polynomial time and these cryptographic systems can be broken in polynomial time!

Heuristics for Optimization Problems

Dominating Set Optimization Problem

Dominating Set (Optimization): Given a graph G, find the small set S of vertices of G such that every vertex of G is adjacent (or equals) a vertex in S.

Give a heuristic for the Dominating Set Optimization Problem.

Clique Optimization Problem

Definition. A **clique** is a subset V' of the vertices of graph G such that for each $u, v \in V'$, pair (u, v) is an edge of G.

Clique Prolem (Optimization): Given a graph G, find the largest clique of G.

Give a heuristic for the Clique Optimization Problem.

Independent Set Optimization Problem

Definition. An **independent set** is a subset V' of the vertices of graph G such that if both u and v are in V' then (u, v) is NOT an edge of G.

Independent Set (Optimization): Given a graph G, find the largest independent set of G.

Give a heuristic for the Independent Set Optimization Problem.

Vertex Coloring Problem

Definition. An **independent set** is a subset V' of the vertices of graph G such that if both u and v are in V' then (u, v) is NOT an edge of G.

Vertex Coloring (Optimization): Given a graph G, find the coloring of the vertices of G with the fewest number of colors so that no two adjacent vertices have the same color.

Give a heuristic for the Vertex Coloring Optimization Problem.