Complexity

CPE 349

Theresa Migler-VonDollen

Complexity

Complexity

Definition

A *decision problem* is a problem for which any proposed solution can be quickly checked for correctness.

Complexity

Definition

A *decision problem* is a problem for which any proposed solution can be quickly checked for correctness.

For a decision problem:

Complexity

Definition

A *decision problem* is a problem for which any proposed solution can be quickly checked for correctness.

For a decision problem:

- There exists a "checking algorithm" C, that takes as input:

Complexity

Definition

A *decision problem* is a problem for which any proposed solution can be quickly checked for correctness.

For a decision problem:

- There exists a "checking algorithm" C, that takes as input:
  - The given instance of the problem, I, and

Complexity

Definition

A *decision problem* is a problem for which any proposed solution can be quickly checked for correctness.

For a decision problem:

- There exists a "checking algorithm" C, that takes as input:
    - The given instance of the problem, I, and
    - The proposed solution, S.

Complexity

Definition

A *decision problem* is a problem for which any proposed solution can be quickly checked for correctness.

For a decision problem:

- There exists a "checking algorithm" C, that takes as input:
  - The given instance of the problem, I, and
  - The proposed solution, S.
- C outputs `true` if and only if S is a solution for instance I.

Complexity

Definition

A *decision problem* is a problem for which any proposed solution can be quickly checked for correctness.

For a decision problem:

- There exists a "checking algorithm" C, that takes as input:
  - The given instance of the problem, I, and
  - The proposed solution, S.
- C outputs `true` if and only if S is a solution for instance I.
- Further, the running time of C on instance (I, S), is polynomial in $|I|$.

Complexity

Definition

A *decision problem* is a problem for which any proposed solution can be quickly checked for correctness.

For a decision problem:

- There exists a "checking algorithm" C, that takes as input:
  - The given instance of the problem, I, and
  - The proposed solution, S.
- C outputs `true` if and only if S is a solution for instance I.
- Further, the running time of C on instance (I, S), is polynomial in $|I|$.

I think of C as a "grading algorithm".

Complexity

Definition

A *decision problem* is a problem for which any proposed solution can be quickly checked for correctness.

For a decision problem:

- There exists a "checking algorithm" C, that takes as input:
  - The given instance of the problem, I, and
  - The proposed solution, S.
- C outputs `true` if and only if S is a solution for instance I.
- Further, the running time of C on instance (I, S), is polynomial in $|I|$.

I think of C as a "grading algorithm".

Definition

The class of all decision problems is denoted by NP.

NP

NP

Definition

The class of all decision problems is denoted by NP.

NP

Definition

The class of all decision problems is denoted by NP.

Examples:

NP

Definition

The class of all decision problems is denoted by NP.

Examples:

- Knapsack

NP

Definition

The class of all decision problems is denoted by *NP*.

Examples:

- Knapsack
  - How? It's an optimization problem.

NP

Definition

The class of all decision problems is denoted by *NP*.

Examples:

- Knapsack
  - How? It's an optimization problem.
  - Convert it to a decision problem:

NP

Definition

The class of all decision problems is denoted by NP.

Examples:

- Knapsack
  - How? It's an optimization problem.
  - Convert it to a decision problem:
    - Introduce a threshold and check that your solution meets that threshold.

NP

Definition

The class of all decision problems is denoted by *NP*.

Examples:

- Knapsack
  - How? It's an optimization problem.
  - Convert it to a decision problem:
    - Introduce a threshold and check that your solution meets that threshold.
- Independent Set

NP

Definition

The class of all decision problems is denoted by NP.

Examples:

- Knapsack
  - How? It's an optimization problem.
  - Convert it to a decision problem:
    - Introduce a threshold and check that your solution meets that threshold.
- Independent Set
- Traveling Salesperson Problem

NP

Definition

The class of all decision problems is denoted by *NP*.

Examples:

- Knapsack
  - How? It's an optimization problem.
  - Convert it to a decision problem:
    - Introduce a threshold and check that your solution meets that threshold.
- Independent Set
- Traveling Salesperson Problem
- Minimum Spanning Tree

NP

Definition

The class of all decision problems is denoted by *NP*.

Examples:

- Knapsack
  - How? It's an optimization problem.
  - Convert it to a decision problem:
    - Introduce a threshold and check that your solution meets that threshold.
- Independent Set
- Traveling Salesperson Problem
- Minimum Spanning Tree
- Matching

NP

Definition

The class of all decision problems is denoted by *NP*.

Examples:

- Knapsack
  - How? It's an optimization problem.
  - Convert it to a decision problem:
    - Introduce a threshold and check that your solution meets that threshold.
- Independent Set
- Traveling Salesperson Problem
- Minimum Spanning Tree
- Matching
- Many, many others.

P

P

Definition

P is the class of all decision problems that can be solved in polynomial time.

P

Definition

P is the class of all decision problems that can be solved in polynomial time.

Examples of decision problems in P:

P

Definition

P is the class of all decision problems that can be solved in polynomial time.

Examples of decision problems in P:

- Minimum Spanning Tree

P

Definition

P is the class of all decision problems that can be solved in polynomial time.

Examples of decision problems in P:

- Minimum Spanning Tree
- Longest Increasing Subsequence

P

Definition

P is the class of all decision problems that can be solved in polynomial time.

Examples of decision problems in P:

- Minimum Spanning Tree
- Longest Increasing Subsequence
- Independent Set on Trees

P

Definition

P is the class of all decision problems that can be solved in polynomial time.

Examples of decision problems in *P*:

- Minimum Spanning Tree
- Longest Increasing Subsequence
- Independent Set on Trees
- Bipartite Matching

P

Definition

P is the class of all decision problems that can be solved in polynomial time.

Examples of decision problems in P:

- Minimum Spanning Tree
- Longest Increasing Subsequence
- Independent Set on Trees
- Bipartite Matching
- Many, many others

P = NP?

P = NP?

By definition, $P \subseteq NP$.

P = NP?

By definition, $P \subseteq NP$.
But is $P = NP$?

P = NP?

By definition, $P \subseteq NP$.
But is $P = NP$?

- Are there any decision problems that can't be solved in polynomial time?

P = NP?

By definition, $P \subseteq NP$.
But is $P = NP$?

- Are there any decision problems that can't be solved in polynomial time?
  - If so:

P = NP?

By definition, $P \subseteq NP$.
But is $P = NP$?

- Are there any decision problems that can't be solved in polynomial time?
  - If so:
    - $P \neq NP$

P = NP?

By definition, $P \subseteq NP$.
But is $P = NP$?

- Are there any decision problems that can't be solved in polynomial time?
  - If so:
    - $P \neq NP$
  - If not:

P = NP?

By definition, $P \subseteq NP$.
But is $P = NP$?

- Are there any decision problems that can't be solved in polynomial time?
  - If so:
    - $P \neq NP$
  - If not:
    - $P = NP$

P = NP?

By definition, $P \subseteq NP$.
But is $P = NP$?

- Are there any decision problems that can't be solved in polynomial time?
  - If so:
    - $P \neq NP$
  - If not:
    - $P = NP$

The answer is unknown.

P = NP?

By definition, $P \subseteq NP$.
But is $P = NP$?

- Are there any decision problems that can't be solved in polynomial time?
  - If so:
    - $P \neq NP$
  - If not:
    - $P = NP$

The answer is unknown.

- Most mathematicians believe that $P \neq NP$.

P = NP?

By definition, $P \subseteq NP$.
But is $P = NP$?

- Are there any decision problems that can't be solved in polynomial time?
  - If so:
    - $P \neq NP$
  - If not:
    - $P = NP$

The answer is unknown.

- Most mathematicians believe that $P \neq NP$.
- This is one of the most important unsolved problems in mathematics.

NP-Complete

NP-Complete

Definition

A problem X is *NP-complete* if:

NP-Complete

Definition

A problem X is *NP-complete* if:

1. X is in NP and

NP-Complete

Definition

A problem X is *NP-complete* if:

1. X is in NP and
2. Every problem in NP is *reducible* to X in polynomial time.

NP-Complete

Definition

A problem X is *NP-complete* if:

1. X is in *NP* and
2. Every problem in *NP* is *reducible* to X in polynomial time.

Note: A problem satisfying condition 2 is *NP*-hard if it doesn't satisfy condition 1.

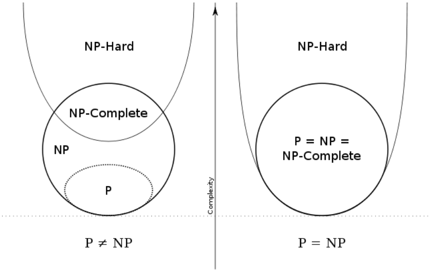
NP-Complete

Definition

A problem X is *NP-complete* if:

1. X is in NP and
2. Every problem in NP is *reducible* to X in polynomial time.

Note: A problem satisfying condition 2 is NP-hard if it doesn't satisfy condition 1.



Reductions

Reductions

Definition

A *reduction* from a decision problem A to a decision problem B ($A \rightarrow B$) is a polynomial time algorithm, f:

Reductions

Definition

A *reduction* from a decision problem A to a decision problem B ($A \to B$) is a polynomial time algorithm, f:

- That transforms an instance, I, of A into an instance, $f(I)$, of B.

Reductions

Definition

A *reduction* from a decision problem A to a decision problem B ($A \rightarrow B$) is a polynomial time algorithm, f:

- That transforms an instance, I, of A into an instance, $f(I)$, of B.

Together with another polynomial time algorithm, h:

Reductions

Definition

A *reduction* from a decision problem A to a decision problem B ($A \to B$) is a polynomial time algorithm, f:

- That transforms an instance, I, of A into an instance, $f(I)$, of B.

Together with another polynomial time algorithm, h:

- That maps any solution S of $f(I)$ back into a solution $h(S)$ of I.

Reductions

Definition

A *reduction* from a decision problem A to a decision problem B ($A \to B$) is a polynomial time algorithm, f:

- That transforms an instance, I, of A into an instance, $f(I)$, of B.

Together with another polynomial time algorithm, h:

- That maps any solution S of $f(I)$ back into a solution $h(S)$ of I.

If $f(I)$ has no solution, then neither does I.

Reductions

Definition

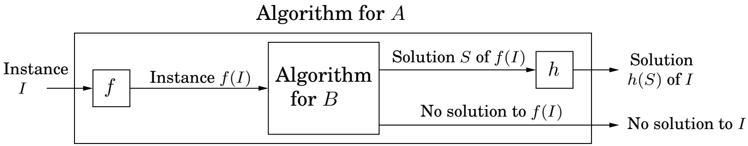
A *reduction* from a decision problem A to a decision problem B ($A \to B$) is a polynomial time algorithm, f:

- That transforms an instance, I, of A into an instance, $f(I)$, of B.

Together with another polynomial time algorithm, h:

- That maps any solution S of $f(I)$ back into a solution $h(S)$ of I.

If $f(I)$ has no solution, then neither does I.



Reductions

Definition

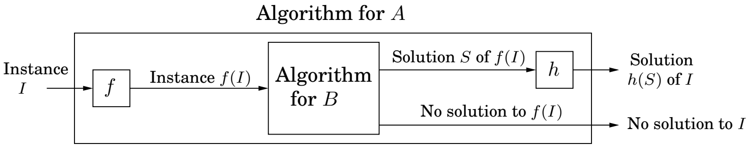
A *reduction* from a decision problem A to a decision problem B ($A \rightarrow B$) is a polynomial time algorithm, f:

- That transforms an instance, I, of A into an instance, $f(I)$, of B.

Together with another polynomial time algorithm, h:

- That maps any solution S of $f(I)$ back into a solution $h(S)$ of I.

If $f(I)$ has no solution, then neither does I.



- If such a reduction exists, it implies that B is at least as hard as A.

SAT

SAT

Definition

A *Boolean formula in conjunctive normal form (CNF)* is a collection of *clauses*, each consisting of the disjunction (logical **or** denoted $\vee$) of several literals.

SAT

Definition

A *Boolean formula in conjunctive normal form (CNF)* is a collection of *clauses*, each consisting of the disjunction (logical **or** denoted $\vee$) of several literals.
A literal is either a Boolean variable (such as x) or the negation of a Boolean variable ($\bar{x}$).

SAT

Definition

A *Boolean formula in conjunctive normal form (CNF)* is a collection of *clauses*, each consisting of the disjunction (logical **or** denoted $\vee$) of several literals.
A literal is either a Boolean variable (such as x) or the negation of a Boolean variable ($\bar{x}$).

Example:
$(x \vee y \vee z) \wedge (x \vee \bar{y}) \wedge (y \vee \bar{z}) \wedge (z \vee \bar{x}) \wedge (\bar{x} \vee \bar{y} \vee \bar{z})$

SAT

Definition

A *Boolean formula in conjunctive normal form (CNF)* is a collection of *clauses*, each consisting of the disjunction (logical **or** denoted $\vee$) of several literals.
A literal is either a Boolean variable (such as x) or the negation of a Boolean variable ($\bar{x}$).

Example:
$(x \vee y \vee z) \wedge (x \vee \bar{y}) \wedge (y \vee \bar{z}) \wedge (z \vee \bar{x}) \wedge (\bar{x} \vee \bar{y} \vee \bar{z})$

SAT

SAT

Definition

A *Boolean formula in conjunctive normal form (CNF)* is a collection of *clauses*, each consisting of the disjunction (logical **or** denoted $\vee$) of several literals.
A literal is either a Boolean variable (such as x) or the negation of a Boolean variable ($\bar{x}$).

Example:
$(x \vee y \vee z) \wedge (x \vee \bar{y}) \wedge (y \vee \bar{z}) \wedge (z \vee \bar{x}) \wedge (\bar{x} \vee \bar{y} \vee \bar{z})$

SAT

Input: A Boolean formula.

SAT

Definition

A *Boolean formula in conjunctive normal form (CNF)* is a collection of *clauses*, each consisting of the disjunction (logical **or** denoted $\vee$) of several literals.
A literal is either a Boolean variable (such as x) or the negation of a Boolean variable ($\bar{x}$).

Example:
$(x \vee y \vee z) \wedge (x \vee \bar{y}) \wedge (y \vee \bar{z}) \wedge (z \vee \bar{x}) \wedge (\bar{x} \vee \bar{y} \vee \bar{z})$

SAT

Input: A Boolean formula.
Goal: Determine if the variables can be assigned in such a way as to make the formula evaluate to TRUE, or determine that no such assignment exists.

SAT

SAT

SAT

Input: A Boolean formula.
Goal: Determine if the variables can be assigned in such a way as to make the formula evaluate to TRUE, or determine that no such assignment exists.

SAT

SAT

Input: A Boolean formula.
Goal: Determine if the variables can be assigned in such a way as to make the formula evaluate to TRUE, or determine that no such assignment exists.

$(x \vee y \vee z) \wedge (x \vee \bar{y}) \wedge (y \vee \bar{z}) \wedge (z \vee \bar{x}) \wedge (\bar{x} \vee \bar{y} \vee \bar{z})$

SAT

SAT

Input: A Boolean formula.
Goal: Determine if the variables can be assigned in such a way as to make the formula evaluate to TRUE, or determine that no such assignment exists.

$(x \vee y \vee z) \wedge (x \vee \bar{y}) \wedge (y \vee \bar{z}) \wedge (z \vee \bar{x}) \wedge (\bar{x} \vee \bar{y} \vee \bar{z})$
Is there an assignment for x, y, and z that makes the following statement TRUE?

SAT

SAT

Input: A Boolean formula.
Goal: Determine if the variables can be assigned in such a way as to make the formula evaluate to TRUE, or determine that no such assignment exists.

$(x \vee y \vee z) \wedge (x \vee \bar{y}) \wedge (y \vee \bar{z}) \wedge (z \vee \bar{x}) \wedge (\bar{x} \vee \bar{y} \vee \bar{z})$
Is there an assignment for x, y, and z that makes the following statement TRUE?

Definition

3-SAT is a special case of k-SAT, when each clause contains exactly $k = 3$ literals. It was one of Karp's 21 NP-complete problems.

SAT

SAT

Input: A Boolean formula.
Goal: Determine if the variables can be assigned in such a way as to make the formula evaluate to TRUE, or determine that no such assignment exists.

$(x \vee y \vee z) \wedge (x \vee \bar{y}) \wedge (y \vee \bar{z}) \wedge (z \vee \bar{x}) \wedge (\bar{x} \vee \bar{y} \vee \bar{z})$
Is there an assignment for x, y, and z that makes the following statement TRUE?

Definition

3-SAT is a special case of k-SAT, when each clause contains exactly $k = 3$ literals. It was one of Karp's 21 NP-complete problems.

We will use 3-SAT to show that Independent Set is NP-complete.

Independent Set

Independent Set

Definition

A subset $S \subset V$ of vertices forms an independent set of a graph $G = (V, E)$ if there are no edges between them.

Independent Set

Definition

A subset $S \subset V$ of vertices forms an independent set of a graph $G = (V, E)$ if there are no edges between them.

Independent Set

Independent Set

Definition

A subset $S \subset V$ of vertices forms an independent set of a graph $G = (V, E)$ if there are no edges between them.

Independent Set

Input: A graph $G = (V, E)$.

Independent Set

Definition

A subset $S \subset V$ of vertices forms an independent set of a graph $G = (V, E)$ if there are no edges between them.

Independent Set

Input: A graph $G = (V, E)$.
Goal: Find a largest independent set (an independent set with the most vertices).

Independent Set

Definition

A subset $S \subset V$ of vertices forms an independent set of a graph $G = (V, E)$ if there are no edges between them.

Independent Set

Input: A graph $G = (V, E)$.
Goal: Find a largest independent set (an independent set with the most vertices).

Now state as a decision problem:

Independent Set

Definition

A subset $S \subset V$ of vertices forms an independent set of a graph $G = (V, E)$ if there are no edges between them.

Independent Set

Input: A graph $G = (V, E)$.
Goal: Find a largest independent set (an independent set with the most vertices).

Now state as a decision problem:

Independent Set

Independent Set

Definition

A subset $S \subset V$ of vertices forms an independent set of a graph $G = (V, E)$ if there are no edges between them.

Independent Set

Input: A graph $G = (V, E)$.
Goal: Find a largest independent set (an independent set with the most vertices).

Now state as a decision problem:

Independent Set

Input: A graph $G = (V, E)$ and a value g.

Independent Set

Definition

A subset $S \subset V$ of vertices forms an independent set of a graph $G = (V, E)$ if there are no edges between them.

Independent Set

Input: A graph $G = (V, E)$.
Goal: Find a largest independent set (an independent set with the most vertices).

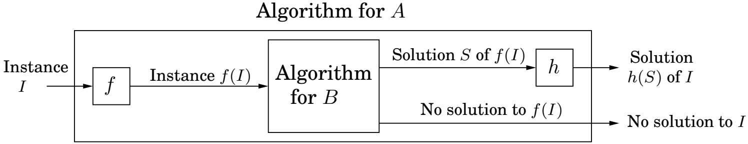
Now state as a decision problem:

Independent Set

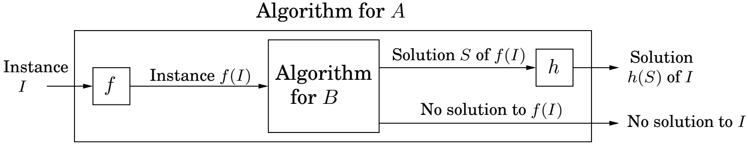
Input: A graph $G = (V, E)$ and a value g.
Goal: Find an independent set with g vertices.

3SAT $\rightarrow$ Independent Set

3SAT → Independent Set

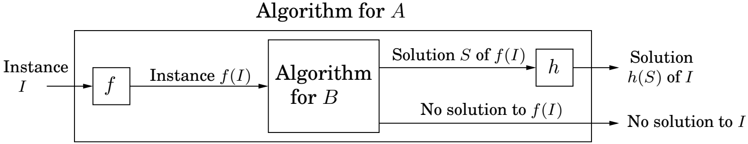


3SAT $\rightarrow$ Independent Set



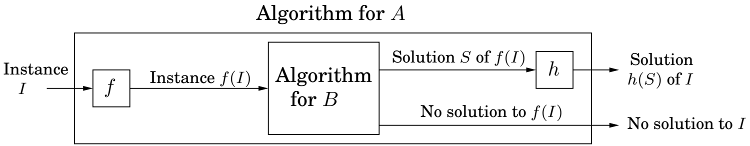
We wish to show that Independent Set is at least as hard as 3SAT.

3SAT → Independent Set



We wish to show that Independent Set is at least as hard as 3SAT. These are two seemingly unrelated problems.

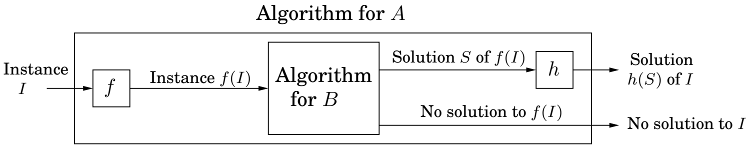
3SAT $\rightarrow$ Independent Set



We wish to show that Independent Set is at least as hard as 3SAT. These are two seemingly unrelated problems.

- One deals with Boolean variables.

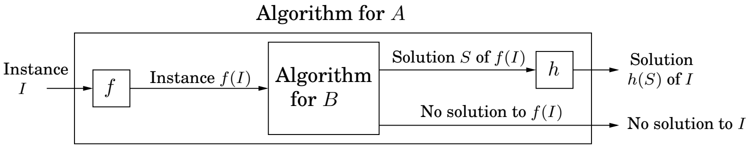
3SAT → Independent Set



We wish to show that Independent Set is at least as hard as 3SAT. These are two seemingly unrelated problems.

- One deals with Boolean variables.
- The other deals with graphs.

3SAT $\rightarrow$ Independent Set



We wish to show that Independent Set is at least as hard as 3SAT. These are two seemingly unrelated problems.

- One deals with Boolean variables.
- The other deals with graphs.

We need to relate Boolean logic with graphs.

3SAT $\rightarrow$ Independent Set

3SAT → Independent Set

- Begin by relating each clause to a graph

3SAT $\rightarrow$ Independent Set

- Begin by relating each clause to a graph - a triangle.

3SAT → Independent Set

- Begin by relating each clause to a graph - a triangle.
  - Why?

3SAT → Independent Set

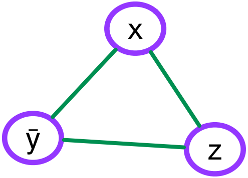
- Begin by relating each clause to a graph - a triangle.
  - Why?
- Label the vertices with the variables.

3SAT → Independent Set

- Begin by relating each clause to a graph - a triangle.
  - Why?
- Label the vertices with the variables.

Example:

$(x \vee \bar{y} \vee z) \rightarrow$

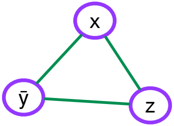


3SAT $\rightarrow$ Independent Set

3SAT → Independent Set

Example:

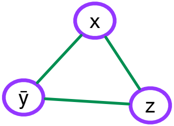
$(x \vee \bar{y} \vee z)$ →



3SAT $\rightarrow$ Independent Set

Example:

$(x \vee \bar{y} \vee z)$ →

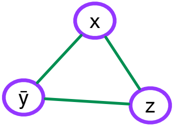


- A triangle has all three vertices maximally connected.

3SAT → Independent Set

Example:

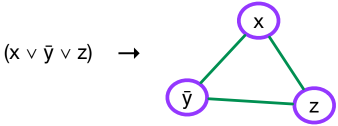
$(x \vee \bar{y} \vee z)$ →



- A triangle has all three vertices maximally connected.
  - This requires that only one vertex is selected for the independent set.

3SAT → Independent Set

Example:

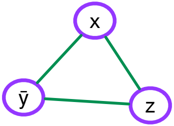


- A triangle has all three vertices maximally connected.
  - This requires that only one vertex is selected for the independent set.
- Repeat this construction for all clauses.

3SAT → Independent Set

Example:

$(x \vee \bar{y} \vee z)$ →

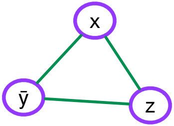


- A triangle has all three vertices maximally connected.
  - This requires that only one vertex is selected for the independent set.
- Repeat this construction for all clauses.
  - The independent set for this graph has to pick at most one literal from each clause.

3SAT → Independent Set

Example:

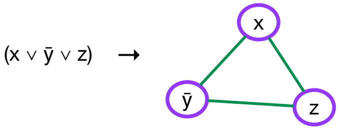
$(x \vee \bar{y} \vee z) \rightarrow$



- A triangle has all three vertices maximally connected.
  - This requires that only one vertex is selected for the independent set.
- Repeat this construction for all clauses.
  - The independent set for this graph has to pick at most one literal from each clause.
- To force exactly one choice from each clause, take g to be the number of clauses.

3SAT → Independent Set

Example:

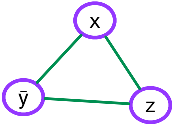


- A triangle has all three vertices maximally connected.
  - This requires that only one vertex is selected for the independent set.
- Repeat this construction for all clauses.
  - The independent set for this graph has to pick at most one literal from each clause.
- To force exactly one choice from each clause, take g to be the number of clauses.
  - How can we make sure not to include both x and $\bar{x}$?

3SAT → Independent Set

Example:

$(x \vee \bar{y} \vee z)$ →

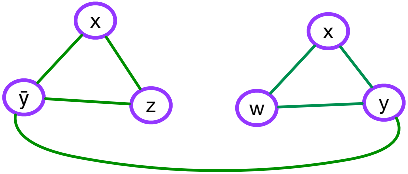


- A triangle has all three vertices maximally connected.
  - This requires that only one vertex is selected for the independent set.
- Repeat this construction for all clauses.
  - The independent set for this graph has to pick at most one literal from each clause.
- To force exactly one choice from each clause, take g to be the number of clauses.
  - How can we make sure not to include both x and $\bar{x}$?
    - Add an edge between every variable and its negation.

3SAT $\rightarrow$ Independent Set

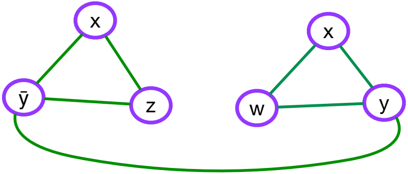
3SAT → Independent Set

$(x \vee \bar{y} \vee z) \wedge (w \vee x \vee y)$ →



3SAT → Independent Set

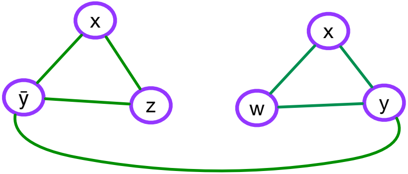
$(x \vee \bar{y} \vee z) \wedge (w \vee x \vee y) \rightarrow$



f is the function that takes an input, I, to 3SAT (a boolean formula) and converts it to an input, $f(I)$, for Independent Set (a graph and a value, g).

3SAT → Independent Set

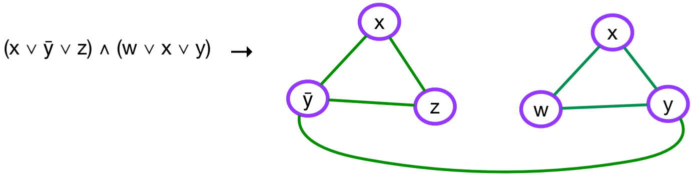
$(x \vee \bar{y} \vee z) \wedge (w \vee x \vee y) \rightarrow$



f is the function that takes an input, I, to 3SAT (a boolean formula) and converts it to an input, $f(I)$, for Independent Set (a graph and a value, g).

- For each clause, f creates a triangle where each vertex has label corresponding to each variable in the clause.

3SAT $\rightarrow$ Independent Set

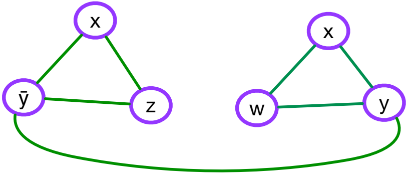


f is the function that takes an input, I, to 3SAT (a boolean formula) and converts it to an input, $f(I)$, for Independent Set (a graph and a value, g).

- For each clause, f creates a triangle where each vertex has label corresponding to each variable in the clause.
- f also adds an edge between vertices representing variable x, and all vertices representing the negation of x, $\bar{x}$.

3SAT → Independent Set

$(x \vee \bar{y} \vee z) \wedge (w \vee x \vee y) \rightarrow$



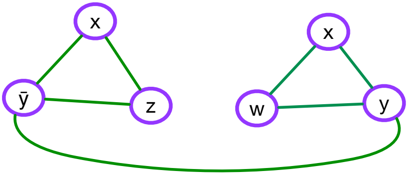
f is the function that takes an input, I, to 3SAT (a boolean formula) and converts it to an input, $f(I)$, for Independent Set (a graph and a value, g).

- For each clause, f creates a triangle where each vertex has label corresponding to each variable in the clause.
- f also adds an edge between vertices representing variable x, and all vertices representing the negation of x, $\bar{x}$.
- If there are n clauses in I, $f(I)$ is the graph described above with value $g = n$.

3SAT $\rightarrow$ Independent Set

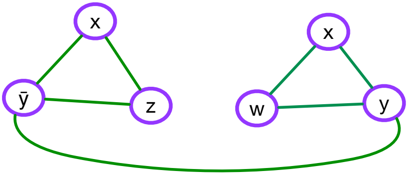
3SAT → Independent Set

$(x \vee \bar{y} \vee z) \wedge (w \vee x \vee y)$ →



3SAT → Independent Set

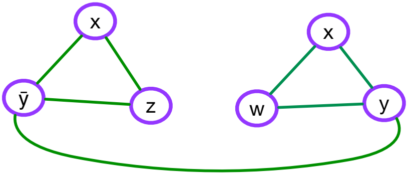
$(x \vee \bar{y} \vee z) \wedge (w \vee x \vee y)$ →



h is the function that takes a solution, S, for input $f(I)$, for Independent Set and converts it to a solution, $h(S)$, for input I for 3SAT.

3SAT → Independent Set

$(x \vee \bar{y} \vee z) \wedge (w \vee x \vee y) \rightarrow$

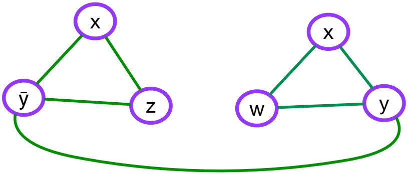


h is the function that takes a solution, S, for input $f(I)$, for Independent Set and converts it to a solution, $h(S)$, for input I for 3SAT.

- A solution for Independent Set is a set of vertices (if this set has size g).

3SAT $\rightarrow$ Independent Set

(x ∨ ȳ ∨ z) ∧ (w ∨ x ∨ y) →

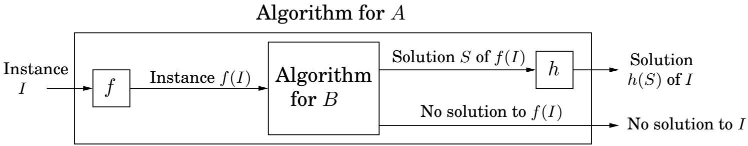


h is the function that takes a solution, S, for input $f(I)$, for Independent Set and converts it to a solution, $h(S)$, for input I for 3SAT.

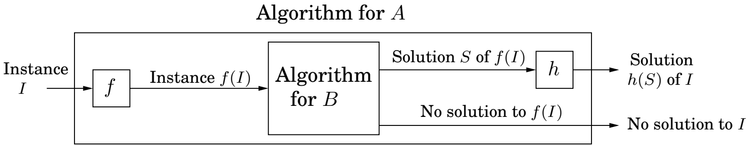
- A solution for Independent Set is a set of vertices (if this set has size g).
- h simply takes the labels of all of the vertices in the independent set and sets their value to `true`.

3SAT $\rightarrow$ Independent Set

3SAT $\rightarrow$ Independent Set

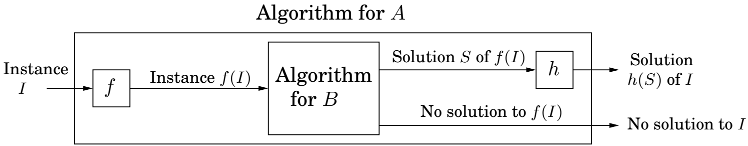


3SAT $\rightarrow$ Independent Set



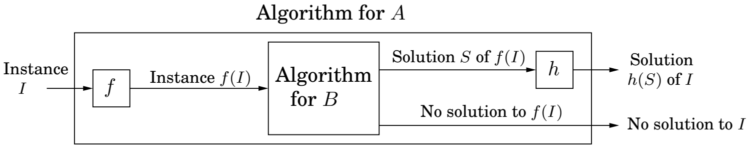
- f is the construction that takes a Boolean formula to a graph and threshold value.

3SAT → Independent Set



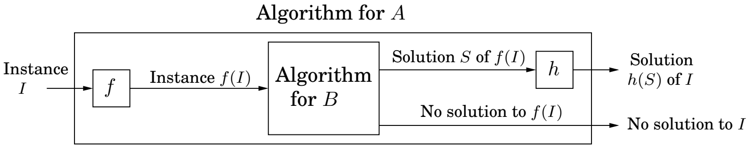
- f is the construction that takes a Boolean formula to a graph and threshold value.
  - This is a polynomial time construction.

3SAT → Independent Set



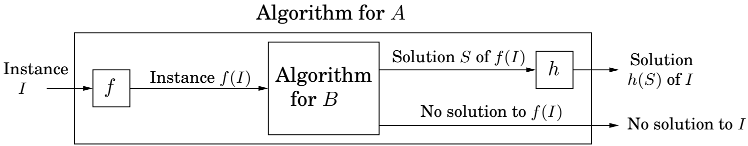
- f is the construction that takes a Boolean formula to a graph and threshold value.
  - This is a polynomial time construction.
- Suppose that there was a polynomial time algorithm for Independent Set.

3SAT $\rightarrow$ Independent Set



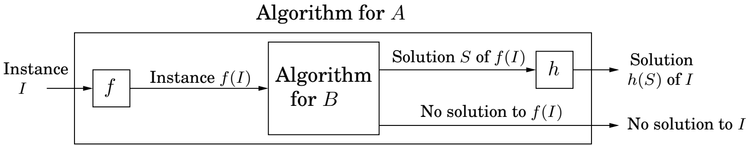
- f is the construction that takes a Boolean formula to a graph and threshold value.
  - This is a polynomial time construction.
- Suppose that there was a polynomial time algorithm for Independent Set.
- h takes a solution for Independent Set to a solution for 3SAT.

3SAT → Independent Set



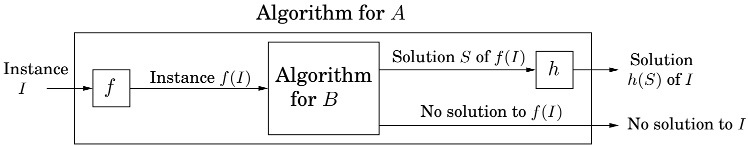
- f is the construction that takes a Boolean formula to a graph and threshold value.
  - This is a polynomial time construction.
- Suppose that there was a polynomial time algorithm for Independent Set.
- h takes a solution for Independent Set to a solution for 3SAT.
  - This is a polynomial time conversion.

3SAT → Independent Set



- f is the construction that takes a Boolean formula to a graph and threshold value.
  - This is a polynomial time construction.
- Suppose that there was a polynomial time algorithm for Independent Set.
- h takes a solution for Independent Set to a solution for 3SAT.
  - This is a polynomial time conversion.
- If G has no independent set of size g, what does that mean?

3SAT $\rightarrow$ Independent Set



- f is the construction that takes a Boolean formula to a graph and threshold value.
  - This is a polynomial time construction.
- Suppose that there was a polynomial time algorithm for Independent Set.
- h takes a solution for Independent Set to a solution for 3SAT.
  - This is a polynomial time conversion.
- If G has no independent set of size g, what does that mean?
  - The Boolean formula is not satisfiable.

3SAT $\rightarrow$ Independent Set - Example

3SAT $\rightarrow$ Independent Set - Example

Suppose that the input to 3SAT is the following Boolean formula:

3SAT $\rightarrow$ Independent Set - Example

Suppose that the input to 3SAT is the following Boolean formula:
$(x \vee y \vee w) \wedge (\bar{x} \vee z \vee \bar{w}) \wedge (\bar{y} \vee v \vee \bar{z}) \wedge (\bar{x} \vee v \vee z)$

3SAT $\rightarrow$ Independent Set - Example

Suppose that the input to 3SAT is the following Boolean formula:
$(x \vee y \vee w) \wedge (\bar{x} \vee z \vee \bar{w}) \wedge (\bar{y} \vee v \vee \bar{z}) \wedge (\bar{x} \vee v \vee z)$
What is the corresponding graph and threshold for the independent set problem?

Vertex Cover

Vertex Cover

Definition

A *vertex cover* is a subset of vertices of a graph such that every edge is incident to at least one vertex in the set.

Vertex Cover

Definition

A *vertex cover* is a subset of vertices of a graph such that every edge is incident to at least one vertex in the set.

Vertex Cover

Vertex Cover

Definition

A *vertex cover* is a subset of vertices of a graph such that every edge is incident to at least one vertex in the set.

Vertex Cover

Input: A graph $G = (V, E)$.

Vertex Cover

Definition

A *vertex cover* is a subset of vertices of a graph such that every edge is incident to at least one vertex in the set.

Vertex Cover

Input: A graph $G = (V, E)$.
Goal: Find a vertex cover of minimum size.

Vertex Cover

Definition

A *vertex cover* is a subset of vertices of a graph such that every edge is incident to at least one vertex in the set.

Vertex Cover

Input: A graph $G = (V, E)$.
Goal: Find a vertex cover of minimum size.

Now state as a decision problem:

Vertex Cover

Definition

A *vertex cover* is a subset of vertices of a graph such that every edge is incident to at least one vertex in the set.

Vertex Cover

Input: A graph $G = (V, E)$.
Goal: Find a vertex cover of minimum size.

Now state as a decision problem:

Vertex Cover

Vertex Cover

Definition

A *vertex cover* is a subset of vertices of a graph such that every edge is incident to at least one vertex in the set.

Vertex Cover

Input: A graph $G = (V, E)$.
Goal: Find a vertex cover of minimum size.

Now state as a decision problem:

Vertex Cover

Input: A graph $G = (V, E)$, and a value x.

Vertex Cover

Definition

A *vertex cover* is a subset of vertices of a graph such that every edge is incident to at least one vertex in the set.

Vertex Cover

Input: A graph $G = (V, E)$.
Goal: Find a vertex cover of minimum size.

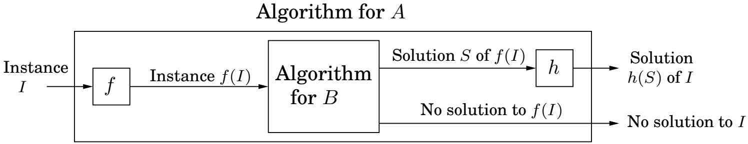
Now state as a decision problem:

Vertex Cover

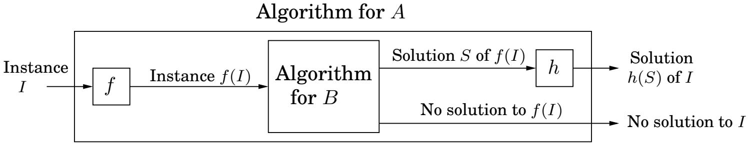
Input: A graph $G = (V, E)$, and a value x.
Goal: Find a vertex cover with size x.

Independent Set $\rightarrow$ Vertex Cover

Independent Set → Vertex Cover



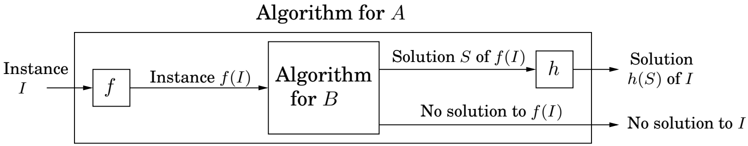
Independent Set → Vertex Cover



Theorem

A set of vertices S is a vertex cover of a graph G if and only if the vertices $V \setminus S$ are an independent set of G.

Independent Set → Vertex Cover

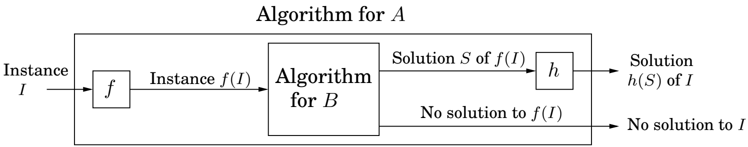


Theorem

A set of vertices S is a vertex cover of a graph G if and only if the vertices $V \setminus S$ are an independent set of G.

- Let $I = (G, g)$, then $f((G, g)) = (G, |V| - g)$.

Independent Set → Vertex Cover

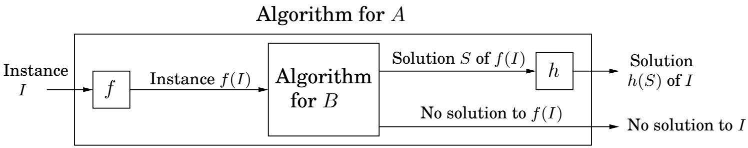


Theorem

A set of vertices S is a vertex cover of a graph G if and only if the vertices $V \setminus S$ are an independent set of G.

- Let $I = (G, g)$, then $f((G, g)) = (G, |V| - g)$.
- Suppose that there is a polynomial time algorithm for Vertex Cover that returns solution $W \subseteq V$ (if one exists).

Independent Set $\rightarrow$ Vertex Cover

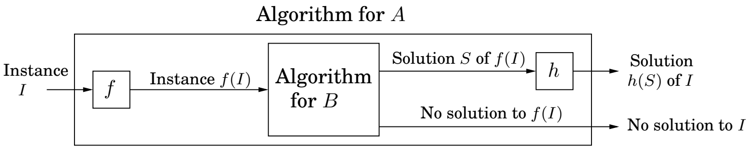


Theorem

A set of vertices S is a vertex cover of a graph G if and only if the vertices $V \setminus S$ are an independent set of G.

- Let $I = (G, g)$, then $f((G, g)) = (G, |V| - g)$.
- Suppose that there is a polynomial time algorithm for Vertex Cover that returns solution $W \subseteq V$ (if one exists).
- $h(W) = V \setminus W$.

Independent Set $\rightarrow$ Vertex Cover



Theorem

A set of vertices S is a vertex cover of a graph G if and only if the vertices $V \setminus S$ are an independent set of G.

- Let $I = (G, g)$, then $f((G, g)) = (G, |V| - g)$.
- Suppose that there is a polynomial time algorithm for Vertex Cover that returns solution $W \subseteq V$ (if one exists).
- $h(W) = V \setminus W$.
- No solution for $f(I)$ implies no solution to I.

Clique

Clique

Definition

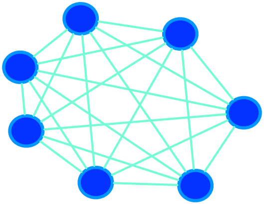
A *clique* is a graph in which there is an edge between every two vertices.

Clique

Definition

A *clique* is a graph in which there is an edge between every two vertices.

Example (K_7):

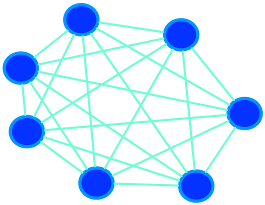


Clique

Definition

A *clique* is a graph in which there is an edge between every two vertices.

Example (K_7):



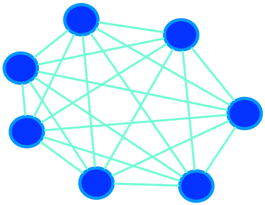
Clique

Clique

Definition

A *clique* is a graph in which there is an edge between every two vertices.

Example (K_7):



Clique

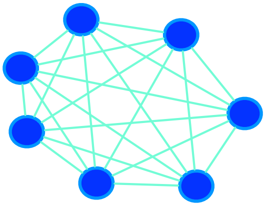
Input: A graph $G = (V, E)$ and a value k.

Clique

Definition

A *clique* is a graph in which there is an edge between every two vertices.

Example (K_7):



Clique

Input: A graph $G = (V, E)$ and a value k.
Goal: Find a clique of size k in G.

Independent Set $\rightarrow$ Clique

Independent Set $\rightarrow$ Clique

Definition

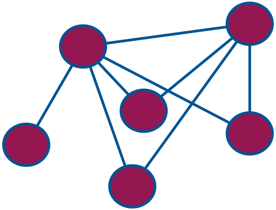
The *complement* of a graph $G = (V, E)$ is the graph $\bar{G} = (V, \bar{E})$ where $\bar{E}$ contains the possible edges of G that are not in E.

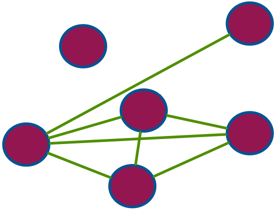
Independent Set $\rightarrow$ Clique

Definition

The *complement* of a graph $G = (V, E)$ is the graph $\bar{G} = (V, \bar{E})$ where $\bar{E}$ contains the possible edges of G that are not in E.

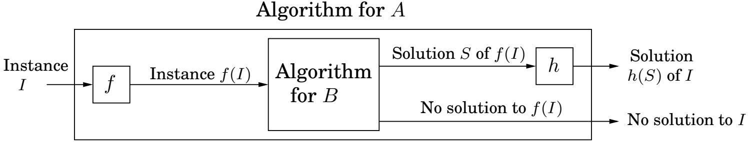
Example:



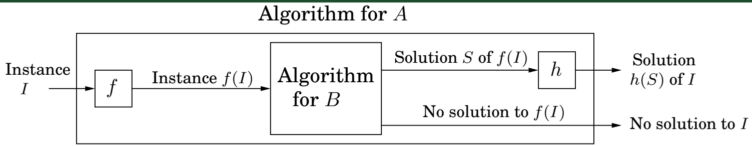


Independent Set $\rightarrow$ Clique

Independent Set $\rightarrow$ Clique



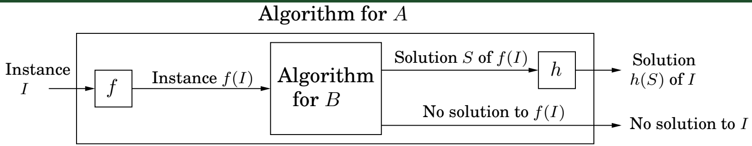
Independent Set → Clique



Theorem

A set of vertices S is an independent set of a graph G if and only if the S is a clique of $\bar{G}$.

Independent Set → Clique

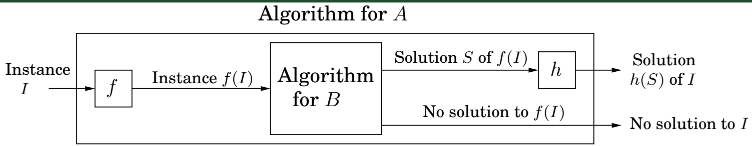


Theorem

A set of vertices S is an independent set of a graph G if and only if the S is a clique of $\bar{G}$.

- Let $I = (G, g)$, then $f((G, g)) = (\bar{G}, g)$.

Independent Set $\rightarrow$ Clique

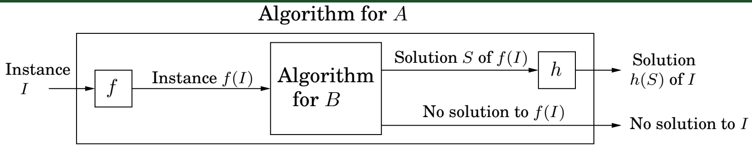


Theorem

A set of vertices S is an independent set of a graph G if and only if the S is a clique of $\bar{G}$.

- Let $I = (G, g)$, then $f((G, g)) = (\bar{G}, g)$.
- Suppose that there is a polynomial time algorithm for Clique that returns solution $W \subseteq V$ (if one exists).

Independent Set $\rightarrow$ Clique

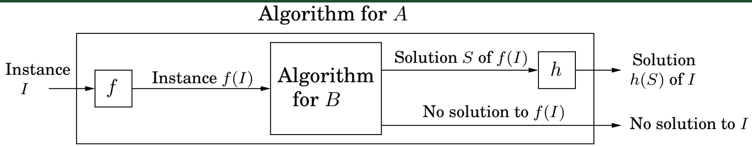


Theorem

A set of vertices S is an independent set of a graph G if and only if the S is a clique of $\bar{G}$.

- Let $I = (G, g)$, then $f((G, g)) = (\bar{G}, g)$.
- Suppose that there is a polynomial time algorithm for Clique that returns solution $W \subseteq V$ (if one exists).
- $h(W) = W$.

Independent Set → Clique



Theorem

A set of vertices S is an independent set of a graph G if and only if the S is a clique of $\bar{G}$.

- Let $I = (G, g)$, then $f((G, g)) = (\bar{G}, g)$.
- Suppose that there is a polynomial time algorithm for Clique that returns solution $W \subseteq V$ (if one exists).
- $h(W) = W$.
- No solution for $f(I)$ implies no solution to I.

What have we shown?

What have we shown?

We have shown:

What have we shown?

We have shown:

- Independent Set is at least as hard as 3SAT.

What have we shown?

We have shown:

- Independent Set is at least as hard as 3SAT.
- Vertex Cover is at least as hard as Independent Set.

What have we shown?

We have shown:

- Independent Set is at least as hard as 3SAT.
- Vertex Cover is at least as hard as Independent Set.
- Clique is at least as hard as Independent Set.

What have we shown?

We have shown:

- Independent Set is at least as hard as 3SAT.
- Vertex Cover is at least as hard as Independent Set.
- Clique is at least as hard as Independent Set.