Design and Analysis of Algorithms, MTech-I (1st semester)

Chapter 6: NP Theory - II

October 7, 2022

Devesh C Jinwala,

Professor in CSE, SVNIT, Surat and Adjunct Professor, IITJammu & Dean (R&C), SVNIT

Department of Computer Science and Engineering, SVNIT, Surat

Broad Contents of the talks

- Talk1: Understanding How to Analyze an Algorithm. Efficient and Inefficient Algorithms. Complexity Classes of Problems.

Broad Contents of the talks

- Talk1: Understanding How to Analyze an Algorithm. Efficient and Inefficient Algorithms. Complexity Classes of Problems.
- Talk2: Understanding and Working with Problem Reductions.

Broad Contents of the talks

- Talk1: Understanding How to Analyze an Algorithm. Efficient and Inefficient Algorithms. Complexity Classes of Problems.
- Talk2: Understanding and Working with Problem Reductions.
- Talk3: Non-determinism, Working with NPHard, NPComplete.

Topics to be discussed

A. **Talk1: Algorithm Analysis, Problem Complexity Classes.**
    1. What does solving problems algorithmically, mean?
    2. Analyzing an algorithm
    3. How to relate time complexity to input size?
    4. Classifying problems
    5. A Motivating Example to illustrate hardness
    6. Some Hard Problems

Topics to be discussed...

- B **Talk2: Relating Problem Hardness & Polynomial Reductions**
  1. Reductions
  2. How can we relate hardness of two problems ?
  3. How can we relate solvability of two problems ?
  4. Polynomial Reduction of one problem to the other
  5. Polynomial Equivalence of one problem to the other
  6. Three methods of reductions: illustrations

Topics to be discussed...

- **Talk3: Non-determinism, Working with NPHard, NPComplete.**
  1. The concept of non-determinism

Topics to be discussed...

- **Talk3: Non-determinism, Working with NPHard, NPComplete.**
  1. The concept of non-determinism
  2. Designing Non-deterministic algorithms

Topics to be discussed...

- **Talk3: Non-determinism, Working with NPHard, NPComplete.**
    1. The concept of non-determinism
    2. Designing Non-deterministic algorithms
    3. The Class P, NP, EXP of problems

Topics to be discussed...

- **Talk3: Non-determinism, Working with NPHard, NPComplete.**
  1. The concept of non-determinism
  2. Designing Non-deterministic algorithms
  3. The Class P, NP, EXP of problems
  4. The NP-Hard, NP-Complete problems

Topics to be discussed...

- **Talk3: Non-determinism, Working with NPHard, NPComplete.**
  1. The concept of non-determinism
  2. Designing Non-deterministic algorithms
  3. The Class P, NP, EXP of problems
  4. The NP-Hard, NP-Complete problems
  5. Proofs associated

Topics to be discussed...

- **Talk3: Non-determinism, Working with NPHard, NPComplete.**
  1. The concept of non-determinism
  2. Designing Non-deterministic algorithms
  3. The Class P, NP, EXP of problems
  4. The NP-Hard, NP-Complete problems
  5. Proofs associated
  6. Summarizing

Reviewing Objectives

- To understand

Reviewing Objectives

- To understand
  - that many of the problems that have polynomial time algorithms are computationally related.

Reviewing Objectives

- To understand
  - that many of the problems that have polynomial time algorithms are computationally related.
  - a few of the real world problems defy classification. . . ..

Reviewing Objectives

- To understand
  - that many of the problems that have polynomial time algorithms are computationally related.
  - a few of the real world problems defy classification. . . ..
  - how nondeterminism property can be exploited to understand the classes of problems.

Reviewing Objectives

- To understand
  - that many of the problems that have polynomial time algorithms are computationally related.
  - a few of the real world problems defy classification. . . ..
  - how nondeterminism property can be exploited to understand the classes of problems.
  - the classification of the problems viz. NP-Hard and NP-Complete

Reviewing Objectives

- To understand
  - that many of the problems that have polynomial time algorithms are computationally related.
  - a few of the real world problems defy classification. . . . .
  - how nondeterminism property can be exploited to understand the classes of problems.
  - the classification of the problems viz. NP-Hard and NP-Complete
  - how one problem can be reduced to another. . . . . . rendering them to be similar in nature to one another. . . . .

Reductions, broadly

Consider two problems P and Q such that suppose we wish to solve P but, we have an algorithm to solve Q.

- Then, if we can devise a function T

Then, we have been able to solve P using the algorithm for Q.

Reductions, broadly

Consider two problems P and Q such that suppose we wish to solve P but, we have an algorithm to solve Q.

- Then, if we can devise a function T
- that takes as input x - that P is supposed to take,

Then, we have been able to solve P using the algorithm for Q.

Reductions, broadly

Consider two problems P and Q such that suppose we wish to solve P but, we have an algorithm to solve Q.

- Then, if we can devise a function T
- that takes as input x - that P is supposed to take,
- produces T(x) – that Q accepts as input and

Then, we have been able to solve P using the algorithm for Q.

Reductions, broadly

Consider two problems P and Q such that suppose we wish to solve P but, we have an algorithm to solve Q.

- Then, if we can devise a function T
- that takes as input x - that P is supposed to take,
- produces T(x) – that Q accepts as input and
- Q produces the output that actually is the output of P

Then, we have been able to solve P using the algorithm for Q.

Reductions, broadly

Consider two problems P and Q such that suppose we wish to solve P but, we have an algorithm to solve Q.

- Then, if we can devise a function T
- that takes as input x - that P is supposed to take,
- produces T(x) – that Q accepts as input and
- Q produces the output that actually is the output of P
- then, we say that P reduces to Q

Then, we have been able to solve P using the algorithm for Q.

Reductions, pictorially

- P reduces to Q means Problem P is solved using Q and a function T and so

Reductions, pictorially

- P reduces to Q means Problem P is solved using Q and a function T and so
- *P reduces to Q* $\Longrightarrow$ Q is at least as hard as P

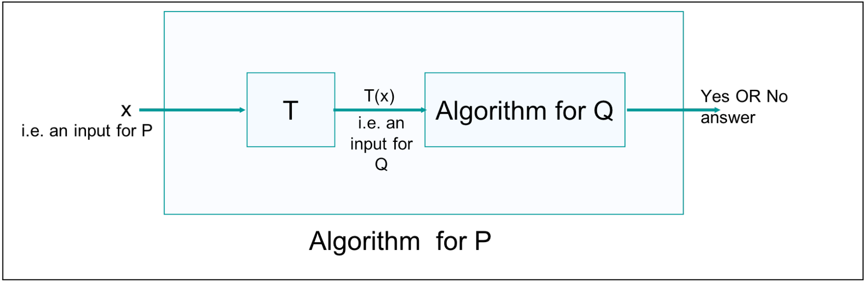


Figure: Problem P is solved using an algorithm for Q and a converter

Reductions, pictorially

- P reduces to Q means Problem P is solved using Q and a function T and so
- *P reduces to Q* $\implies$ Q is at least as hard as P

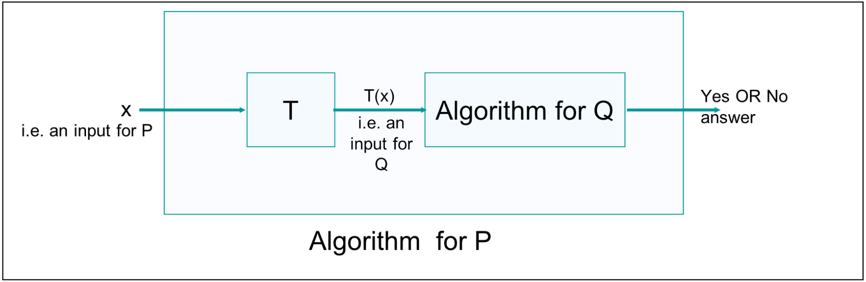


Figure: Problem P is solved using an algorithm for Q and a converter

- Note that, here T is nothing but a converter..........

Reductions, formally defining

- def: Let P_1 and P_2 be two problems. Then problem P_1 reduces to P_2 (i.e. $P_1 \ \alpha \ P_2$)if and only if there is a way to solve P_1 by a deterministic polynomial time algorithm P_2 i.e. using a deterministic algorithm that also solves P_2 in polynomial time.
- P reduces to Q is denoted as either P α_P Q OR P $\leq_P$ Q

Understanding Reductions, further

- Informally, given that a problem P_1 is reducible to P_2 (i.e. $P_1 \leq_P P_2$), it implies that if P_2 is solvable then P_1 also is solvable.

Understanding Reductions, further

- Informally, given that a problem P_1 is reducible to P_2 (i.e. $P_1 \leq_P P_2$), it implies that if P_2 is solvable then P_1 also is solvable.
- How would you relate then, the problems SELECTION[1] and SORTING ?

Understanding Reductions, further

- Informally, given that a problem P_1 is reducible to P_2 (i.e. $P_1 \leq_P P_2$), it implies that if P_2 is solvable then P_1 also is solvable.
- How would you relate then, the problems SELECTION[1] and SORTING ?
- Does SELECTION $\leq_P$ SORTING i.e. if SORTING is solvable then SELECTION also is solvable. Is this true ? OR

Understanding Reductions, further

- Informally, given that a problem P_1 is reducible to P_2 (i.e. $P_1 \leq_P P_2$), it implies that if P_2 is solvable then P_1 also is solvable.
- How would you relate then, the problems SELECTION[1] and SORTING ?
- Does SELECTION $\leq_P$ SORTING i.e. if SORTING is solvable then SELECTION also is solvable. Is this true ? OR
- Does SORTING $\leq_P$ SELECTION i.e. if SELECTION is solvable then SORTING also is solvable. Is this true ?

Understanding Reductions, further...

- Very very importantly, note that given that a problem P_1 is reducible to P_2 (i.e. $P_1 \leq_P P_2$),
  it also implies that P_2 is at least as difficult as P_1

Understanding Reductions, further...

- Very very importantly, note that given that a problem P_1 is reducible to P_2 (i.e. $P_1 \leq_P P_2$),
  it also implies that P_2 is at least as difficult as P_1
- This means that

SELECTION α SORTING
implies that
SORTING is as difficult as SELECTION

Intuitively, how can we use reduction to prove a new problem to be hard ?

Figure: ???

Understanding Reductions, further...

- How would you define the converter functions for reducing the SELECTION to the SORTING, in the previous example ?

Understanding Reductions, further...

- How would you define the converter functions for reducing the SELECTION to the SORTING, in the previous example ?
- Now, Input for T(x) – we simply input N different numbers viz. $(x_1, x_2, x_3, \ldots\ldots\ldots\ x_n)$ to T_1 hiding the i^{th} largest value that is to be selected from these N different numbers.

Understanding Reductions, further...

- How would you define the converter functions for reducing the SELECTION to the SORTING, in the previous example ?
- Now, Input for T(x) – we simply input N different numbers viz. $(x_1, x_2, x_3, \ldots\ldots.\ x_n)$ to T_1 hiding the i^{th} largest value that is to be selected from these N different numbers.
- i.e. let T be defined as $\mathrm{T}(x_1, x_2, x_3, \ldots\ldots.\ x_n) = (y_1, y_2, y_3, \ldots\ldots y_n)$, such that $y_i = x_i$

Understanding Reductions, further...

- How would you define the converter functions for reducing the SELECTION to the SORTING, in the previous example ?
- Now, Input for T(x) – we simply input N different numbers viz. $(x_1, x_2, x_3, \ldots\ldots\ldots x_n)$ to T_1 hiding the i^{th} largest value that is to be selected from these N different numbers.
- i.e. let T be defined as
  T$(x_1, x_2, x_3, \ldots\ldots\ldots x_n) = (y_1, y_2, y_3, \ldots\ldots y_n)$, such that $y_i = x_i$
- Thus, Q would sort these N different numbers

Understanding Reductions, further...

- How would you define the converter functions for reducing the SELECTION to the SORTING, in the previous example ?
- Now, Input for T(x) – we simply input N different numbers viz. $(x_1, x_2, x_3, \ldots\ldots\ x_n)$ to T_1 hiding the i^{th} largest value that is to be selected from these N different numbers.
- i.e. let T be defined as
  $\mathrm{T}(x_1, x_2, x_3, \ldots\ldots\ x_n) = (y_1, y_2, y_3, \ldots\ldots y_n)$, such that $y_i = x_i$
- Thus, Q would sort these N different numbers
- Let T_2 select only the i^{th} value from these and return as the answer.

Understanding Reductions, further...

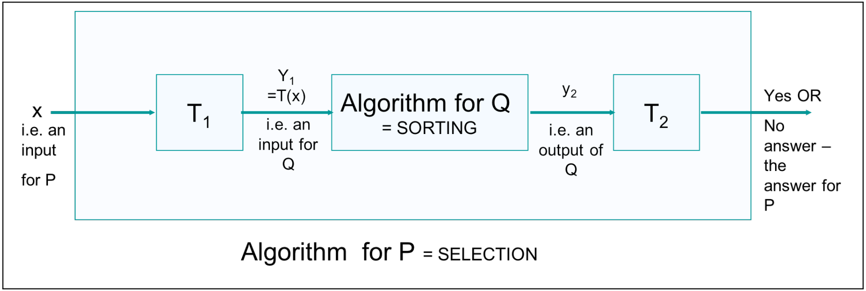


Figure: Solving SELECTION using SORTING and T1 and T2

Reductions, another example

- Consider Two problems
  - SumPair - Compute the sum of two numbers
  - SumList - Compute the sum of n numbers in a vector
- Is it possible to use SumList to find the sum of n numbers in a vector i.e. the answer to SumPair ?
- Hence, we would say $Sum_{Pair} \leq_P Sum_{List}$
- However, in this case we can also say that $Sum_{List} \leq_P Sum_{Pair}$
- Therefore, we can say that $Sum_{Pair} \equiv_P Sum_{List}$

Polynomial Time Reduction

- It is essential to ensure that the functions T_1, T_2 and the algorithm for Q are all polynomial time to ensure Polynomial Time Reducibility.
- That is, reducibility is useful only if it is polynomial time reducibility.
- def: We say that a problem P_1 polynomially reduces to another problem P_2 only if the time for CONVERTER 1 plus the time for CONVERTER 2 is DETERMINISTIC POLYNOMIAL TIME.

Tutorial Problem

- Desiderata : Suppose we could solve X in polynomial-time. What else could we solve in polynomial time?

Tutorial Problem

- Desiderata : Suppose we could solve X in polynomial-time. What else could we solve in polynomial time?
- Reduction

Tutorial Problem

- Desiderata : Suppose we could solve X in polynomial-time. What else could we solve in polynomial time?
- Reduction
    - Problem X polynomial reduces to problem Y if arbitrary instances of problem X can be solved using

Tutorial Problem

- Desiderata : Suppose we could solve X in polynomial-time. What else could we solve in polynomial time?
- Reduction
  - Problem X polynomial reduces to problem Y if arbitrary instances of problem X can be solved using
  - Polynomial number of standard computational steps, plus

Tutorial Problem

- Desiderata : Suppose we could solve X in polynomial-time. What else could we solve in polynomial time?
- Reduction
  - Problem X polynomial reduces to problem Y if arbitrary instances of problem X can be solved using
  - Polynomial number of standard computational steps, plus
  - Polynomial number of calls to oracle that solves problem Y.

Reduction By Simple Equivalence

Vertex Cover $\equiv_P$ IS

- Minimal Vertex Cover of a graph is a minimum subset of the vertices of G which contains at least one of the two endpoints of each edge in G

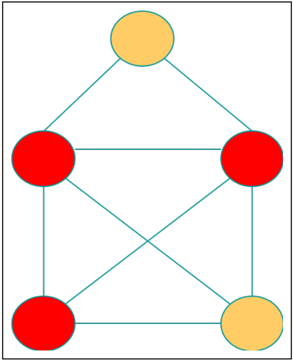


Figure: What is the Vertex Cover?

Vertex Cover $\equiv_P$ IS...

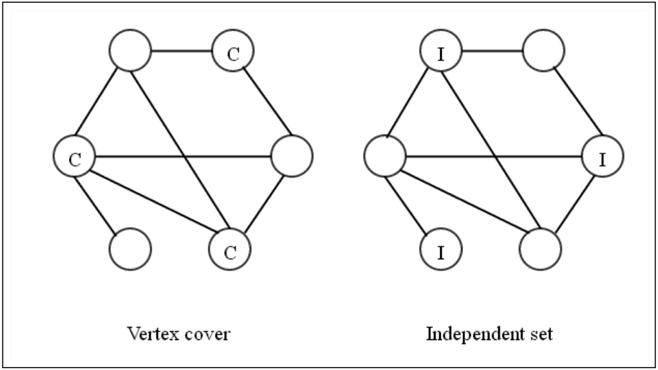


Figure: Vertex Cover?

Vertex Cover $\equiv_P$ IS...

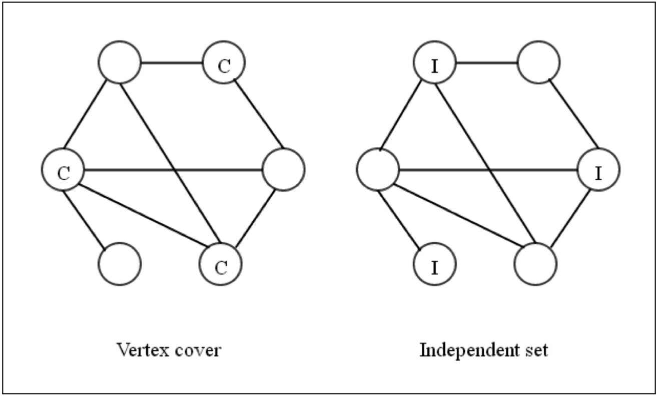


Figure: Independent Set?

Vertex Cover $\equiv_P$ IS...

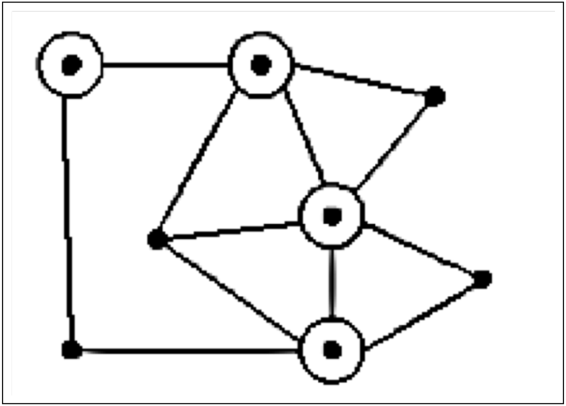


Figure: Independent Set? Vertex Cover?

Vertex Cover $\equiv_P$ IS...

- What is an independent set of a graph ?

Vertex Cover $\equiv_P$ IS...

- What is an independent set of a graph ?
- Problem : Given a graph G = (V, E) and an integer k, is there a subset of vertices S $\leq_P$ V such that $|S| \leq_P$ k, and for each edge at the most only one of its endpoints is in S?

Vertex Cover $\equiv_P$ IS...

- What is an independent set of a graph ?
- Problem : Given a graph G = (V, E) and an integer k, is there a subset of vertices S $\leq_P$ V such that $|S| \leq_P$ k, and for each edge at the most only one of its endpoints is in S?
- e.g. Is there an independent set of size $\leq$6? Yes. ...of size $\leq$ 7? No.

Vertex Cover $\equiv_P$ IS...

- What is an independent set of a graph ?
- Problem : Given a graph G = (V, E) and an integer k, is there a subset of vertices S $\leq_P$ V such that $|S| \leq_P$ k, and for each edge at the most only one of its endpoints is in S?
- e.g. Is there an independent set of size $\leq$6? Yes. ...of size $\leq$ 7? No.

Vertex Cover $\equiv_P$ IS...

- What is an independent set of a graph ?
- Problem : Given a graph G = (V, E) and an integer k, is there a subset of vertices S $\leq_P$ V such that $|S| \leq_P$ k, and for each edge at the most only one of its endpoints is in S?
- e.g. Is there an independent set of size $\leq$6? Yes. ...of size $\leq$ 7? No.

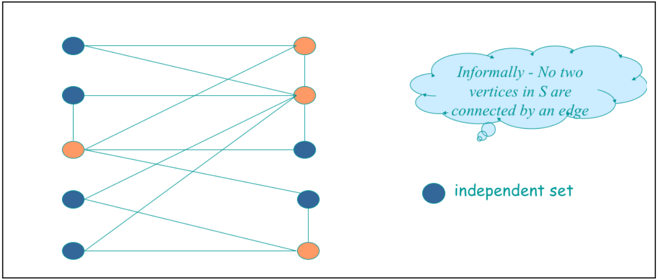


Figure: Independent Set? Vertex Cover?

Vertex Cover $\equiv_P$ IS...

- Minimum VERTEX COVER : informally a set of vertices that include all the edges. . . ..
- Given a graph G = (V, E) and an integer k, is there a subset of vertices S $\leq$ V such that $|S| \leq$ k, and for each edge, at least one of its endpoints is in S?
- Is there a vertex cover of size $\leq$ 4? Yes. size $\leq$ 3? No.

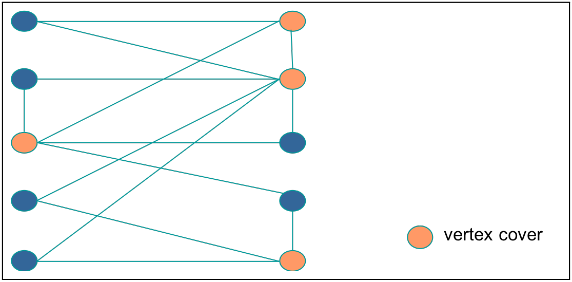


Figure: Independent Set? Vertex Cover?

Vertex Cover $\equiv_P$ IS...

- Claim. VERTEX-COVER $\equiv_P$ INDEPENDENT-SET.
- We show S is an independent set iff V - S is a vertex cover.

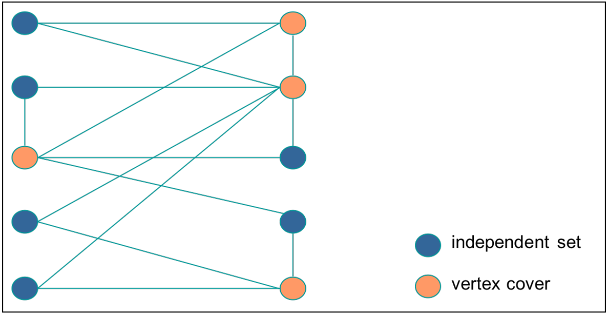


Figure: Independent Set? Vertex Cover?

Vertex Cover $\equiv_P$ IS...

- Part I : We prove that if S is an independent set, then V-S is a vertex cover

independent...
a subset of vertices
$S \subseteq V$ such that
$|S| \geq k$, and for
each edge
at the most
only one of
its endpoints is in S

Vertex Cover $\equiv_P$ IS...

- Part I : We prove that if S is an independent set, then V-S is a vertex cover
- Claim. VERTEX-COVER $\leq_P$ INDEPENDENT-SET

independent...
a subset of vertices
$S \subseteq V$ such that
$|S| \geq k$, and for
each edge
at the most
only one of
its endpoints is in S

Vertex Cover $\equiv_P$ IS...

- Part I : We prove that if S is an independent set, then V-S is a vertex cover
- Claim. VERTEX-COVER $\leq_P$ INDEPENDENT-SET
- Proof:

independent...
a subset of vertices
$S \subseteq V$ such that
$|S| \geq k$, and for
each edge
at the most
only one of
its endpoints is in S

Vertex Cover $\equiv_P$ IS...

- Part I : We prove that if S is an independent set, then V-S is a vertex cover
- Claim. VERTEX-COVER $\leq_P$ INDEPENDENT-SET
- Proof:
  - Note that we are given that S be any independent set.

independent...
a subset of vertices
$S \subseteq V$ such that
$|S| \geq k$, and for
each edge
at the most
only one of
its endpoints is in S

Vertex Cover $\equiv_P$ IS...

- Part I : We prove that if S is an independent set, then V-S is a vertex cover
- Claim. VERTEX-COVER $\leq_P$ INDEPENDENT-SET
- Proof:
  - Note that we are given that S be any independent set.
  - Consider an arbitrary edge (u, v).

independent...
a subset of vertices
$S \subseteq V$ such that
$|S| \geq k$, and for
each edge
at the most
only one of
its endpoints is in S

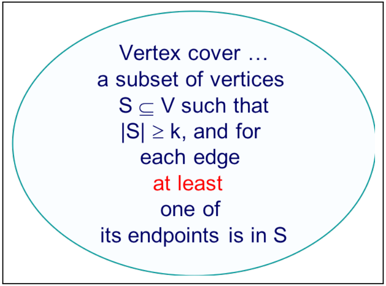
Vertex Cover $\equiv_P$ IS...

- Part I : We prove that if S is an independent set, then V-S is a vertex cover
- Claim. VERTEX-COVER $\leq_P$ INDEPENDENT-SET
- Proof:
  - Note that we are given that S be any independent set.
  - Consider an arbitrary edge (u, v).
  - ... ...

independent...
a subset of vertices
$S \subseteq V$ such that
$|S| \geq k$, and for
each edge
at the most
only one of
its endpoints is in S

Vertex Cover $\equiv_P$ IS...:

- Part II : We prove that if V-S is a vertex cover then S is an independent set



Vertex Cover $\equiv_P$ IS...:

- Part II : We prove that if V-S is a vertex cover then S is an independent set
- Claim : INDEPENDENT-SET $\leq_P$ VERTEX-COVER

Vertex cover ...
a subset of vertices
$S \subseteq V$ such that
$|S| \geq k$, and for
each edge
at least
one of
its endpoints is in S

Vertex Cover $\equiv_P$ IS...:

- Part II : We prove that if V-S is a vertex cover then S is an independent set
- Claim : INDEPENDENT-SET $\leq_P$ VERTEX-COVER
- Proof:

Vertex cover ...
a subset of vertices
$S \subseteq V$ such that
$|S| \geq k$, and for
each edge
at least
one of
its endpoints is in S

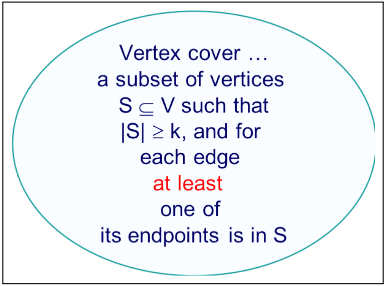
Vertex Cover $\equiv_P$ IS...:

- Part II : We prove that if V-S is a vertex cover then S is an independent set
- Claim : INDEPENDENT-SET $\leq_P$ VERTEX-COVER
- Proof:
  - Consider two vertices u and v belonging to V. Then, there are three cases for the edge (u,v).

Vertex cover ...
a subset of vertices
$S \subseteq V$ such that
$|S| \geq k$, and for
each edge
at least
one of
its endpoints is in S

Vertex Cover $\equiv_P$ IS...:

- Part II : We prove that if V-S is a vertex cover then S is an independent set
- Claim : INDEPENDENT-SET $\leq_P$ VERTEX-COVER
- Proof:
  - Consider two vertices u and v belonging to V. Then, there are three cases for the edge (u,v).
  - What can be said about the edge (u,v) and the set V-S?



Vertex Cover $\equiv_P$ IS...:

- Part II : We prove that if V-S is a vertex cover then S is an independent set
- Claim : INDEPENDENT-SET $\leq_P$ VERTEX-COVER
- Proof:
  - Consider two vertices u and v belonging to V. Then, there are three cases for the edge (u,v).
  - What can be said about the edge (u,v) and the set V-S?
  - Three cases: ... ...

Vertex cover ...
a subset of vertices
S $\subseteq$ V such that
|S| $\geq$ k, and for
each edge
at least
one of
its endpoints is in S

Independent Set: An interesting observation

- An interval scheduling problem can be described by an intersection graph,

Independent Set: An interesting observation

- An interval scheduling problem can be described by an intersection graph,
- In an intersection graph, each vertex is an interval, and there is an edge between two vertices if and only if their intervals overlap.

Independent Set: An interesting observation

- An interval scheduling problem can be described by an intersection graph,
- In an intersection graph, each vertex is an interval, and there is an edge between two vertices if and only if their intervals overlap.
- Remember we studied the interval scheduling problem first up in the Greedy design chapter.

Independent Set: An interesting observation

- An interval scheduling problem can be described by an intersection graph,
- In an intersection graph, each vertex is an interval, and there is an edge between two vertices if and only if their intervals overlap.
- Remember we studied the interval scheduling problem first up in the Greedy design chapter.
- So, now in this representation, how could the interval scheduling problem be modelled?

Independent Set: An interesting observation

- An interval scheduling problem can be described by an intersection graph,
- In an intersection graph, each vertex is an interval, and there is an edge between two vertices if and only if their intervals overlap.
- Remember we studied the interval scheduling problem first up in the Greedy design chapter.
- So, now in this representation, how could the interval scheduling problem be modelled?
- Well, the interval scheduling problem is equivalent to finding the maximum independent set in this intersection graph.

Independent Set: An interesting observation

- An interval scheduling problem can be described by an intersection graph,
- In an intersection graph, each vertex is an interval, and there is an edge between two vertices if and only if their intervals overlap.
- Remember we studied the interval scheduling problem first up in the Greedy design chapter.
- So, now in this representation, how could the interval scheduling problem be modelled?
- Well, the interval scheduling problem is equivalent to finding the maximum independent set in this intersection graph.
- Finding a maximum independent set is NP-hard in general graphs, but it can be done in polynomial time in the special case of intersection graphs (ISMP).

K-Clique $\equiv_P$ Vertex Cover

- Claim. K-Clique $\equiv_P$ VERTEX-COVER

K-Clique $\equiv_P$ Vertex Cover

- Claim. K-Clique $\equiv_P$ VERTEX-COVER
- That is, a graph $G = (V, E)$ has a clique of size k, iff the compliment graph G_C has a vertex cover of size $|V| - k$.

K-Clique $\equiv_P$ Vertex Cover

- Claim. K-Clique $\equiv_P$ VERTEX-COVER
- That is, a graph $G = (V, E)$ has a clique of size k, iff the compliment graph G_C has a vertex cover of size $|V| - k$.

K-Clique $\equiv_P$ Vertex Cover

- Claim. K-Clique $\equiv_P$ VERTEX-COVER
- That is, a graph $G = (V, E)$ has a clique of size k, iff the compliment graph G_C has a vertex cover of size $|V| - k$.

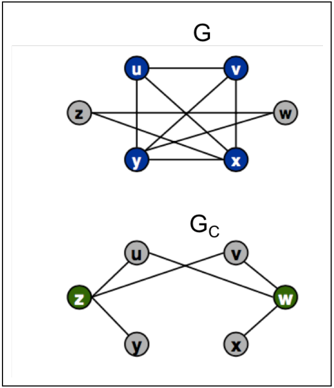


Figure: K-Clique? Vertex Cover?

K-Clique $\equiv_P$ Vertex Cover...

- Part I : We prove that if the graph G has a clique of size k, the compliment graph G_C has a vertex cover of size $|V| - k$.

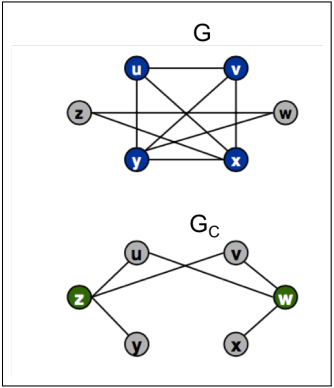


Figure: K-Clique? Vertex Cover?

K-Clique $\equiv_P$ Vertex Cover...

- Part I : We prove that if the graph G has a clique of size k, the compliment graph G_C has a vertex cover of size $|V| - k$.
- Claim : Vertex-Cover $\leq_P$ K-Clique

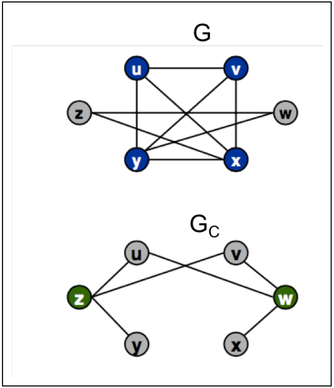


Figure: K-Clique? Vertex Cover?

K-Clique $\equiv_P$ Vertex Cover...

- Part I : We prove that if the graph G has a clique of size k, the compliment graph G_C has a vertex cover of size $|V| - k$.
- Claim : Vertex-Cover $\leq_P$ K-Clique
- Proof: Let S be the k-clique in G then we have to show that $V - S$ is a vertex cover in G_C of size $|V| - k$

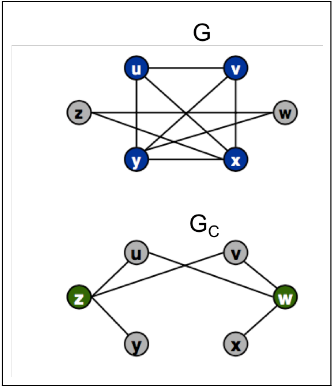


Figure: K-Clique? Vertex Cover?

K-Clique $\equiv_P$ Vertex Cover...

- Part I : We prove that if the graph G has a clique of size k, the compliment graph G_C has a vertex cover of size $|V| - k$.
- Claim : Vertex-Cover $\leq_P$ K-Clique
- Proof: Let S be the k-clique in G then we have to show that $V - S$ is a vertex cover in G_C of size $|V| - k$
    - First, consider an edge (v, w) in G_C

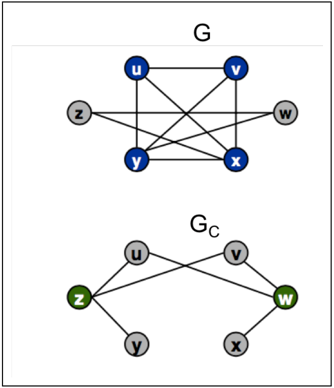


Figure: K-Clique? Vertex Cover?

K-Clique $\equiv_P$ Vertex Cover...

- Part I : We prove that if the graph G has a clique of size k, the compliment graph G_C has a vertex cover of size $|V| - k$.
- Claim : Vertex-Cover $\leq_P$ K-Clique
- Proof: Let S be the k-clique in G then we have to show that $V - S$ is a vertex cover in G_C of size $|V| - k$
    - First, consider an edge (v, w) in G_C
    - What about this edge in G ?

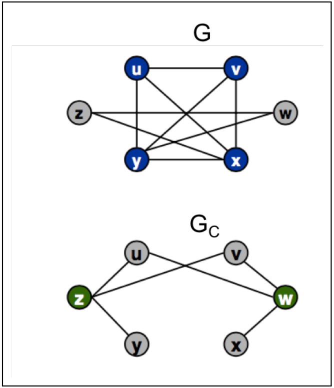


Figure: K-Clique? Vertex Cover?

K-Clique $\equiv_P$ Vertex Cover...

- Part I : We prove that if the graph G has a clique of size k, the compliment graph G_C has a vertex cover of size $|V| - k$.
- Claim : Vertex-Cover $\leq_P$ K-Clique
- Proof: Let S be the k-clique in G then we have to show that $V - S$ is a vertex cover in G_C of size $|V| - k$
  - First, consider an edge (v, w) in G_C
  - What about this edge in G ?
  - ... ...

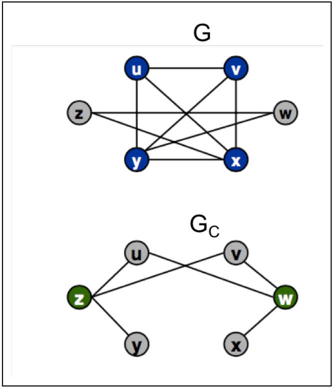


Figure: K-Clique? Vertex Cover?

K-Clique $\equiv_P$ Vertex Cover...

- Part I : We prove that if the graph G has a clique of size k, the compliment graph G_C has a vertex cover of size $|V| - k$.
- Claim : Vertex-Cover $\leq_P$ K-Clique
- Proof: Let S be the k-clique in G then we have to show that $V - S$ is a vertex cover in G_C of size $|V| - k$
  - First, consider an edge (v, w) in G_C
  - What about this edge in G ?
  - ... ...
  - Lastly, what about the sizes k and $|V| - k$? Keep in mind that the vertex sets of both G and Gc are the same i.e. V.

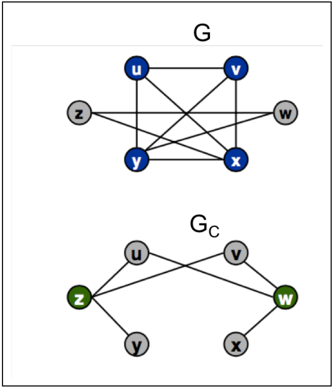


Figure: K-Clique? Vertex Cover?

K-Clique $\equiv_P$ Vertex Cover...

- Part II : We prove that if the graph G_C has a a vertex cover of size $|V| - k$, then the compliment graph G has a clique of size k.

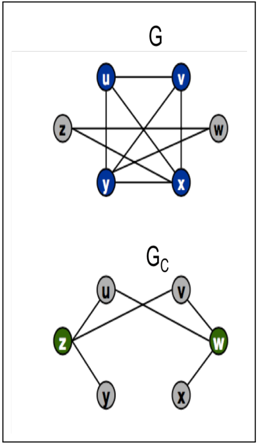


Figure: K-Clique? Vertex Cover?

K-Clique $\equiv_P$ Vertex Cover...

- Part II : We prove that if the graph G_C has a a vertex cover of size $|V| - k$, then the compliment graph G has a clique of size k.
- Claim : K-Clique $\leq_P$ Vertex-Cover

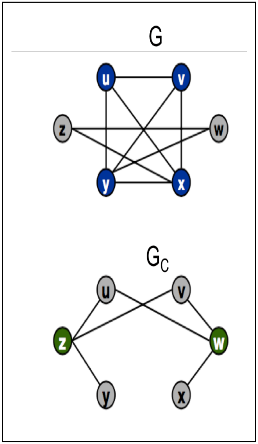


Figure: K-Clique? Vertex Cover?

K-Clique $\equiv_P$ Vertex Cover...

- Part II : We prove that if the graph G_C has a a vertex cover of size $|V| - k$, then the compliment graph G has a clique of size k.
- Claim : K-Clique $\leq_P$ Vertex-Cover
- Proof: Let $V - S$ be the vertex cover of size $|V| - k$ in G_C then we have to show that S is a clique of size k in G_C of size $|V| - k$

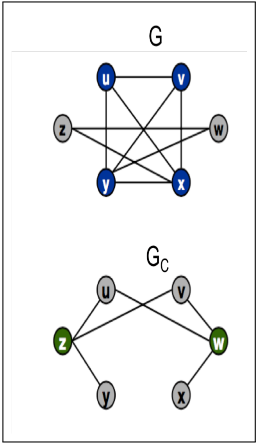


Figure: K-Clique? Vertex Cover?

K-Clique $\equiv_P$ Vertex Cover...

- Part II : We prove that if the graph G_C has a a vertex cover of size $|V| - k$, then the compliment graph G has a clique of size k.
- Claim : K-Clique $\leq_P$ Vertex-Cover
- Proof: Let $V - S$ be the vertex cover of size $|V| - k$ in G_C then we have to show that S is a clique of size k in G_C of size $|V| - k$
- We start with the following:

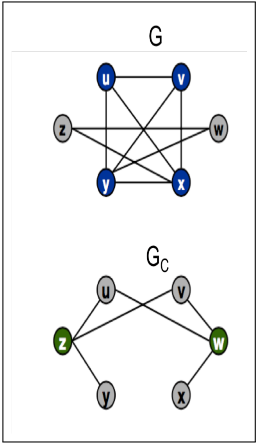


Figure: K-Clique? Vertex Cover?

K-Clique $\equiv_P$ Vertex Cover...

- Part II : We prove that if the graph G_C has a a vertex cover of size $|V| - k$, then the compliment graph G has a clique of size k.
- Claim : K-Clique $\leq_P$ Vertex-Cover
- Proof: Let $V - S$ be the vertex cover of size $|V| - k$ in G_C then we have to show that S is a clique of size k in G_C of size $|V| - k$
- We start with the following:
  - Consider an (edge (v, w) belonging to G_C. Now since $V - S$ is forming a vertex cover in G_C and since edge (v, w) belongs to G_C we have the following two cases:

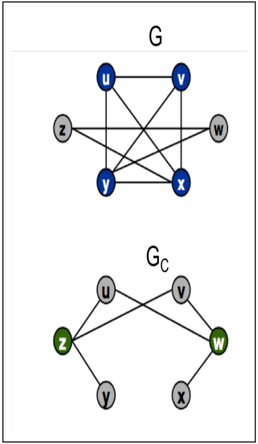


Figure: K-Clique? Vertex Cover?

K-Clique $\equiv_P$ Vertex Cover...

- Part II : We prove that if the graph G_C has a a vertex cover of size $|V| - k$, then the compliment graph G has a clique of size k.
- Claim : K-Clique $\leq_P$ Vertex-Cover
- Proof: Let $V - S$ be the vertex cover of size $|V| - k$ in G_C then we have to show that S is a clique of size k in G_C of size $|V| - k$
- We start with the following:
  - Consider an (edge (v, w) belonging to G_C. Now since $V - S$ is forming a vertex cover in G_C and since edge (v, w) belongs to G_C we have the following two cases:
  - ... ...

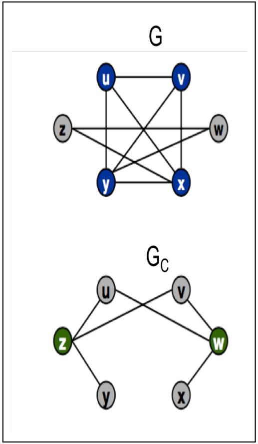


Figure: K-Clique? Vertex Cover?

K-Clique $\equiv_P$ Vertex Cover...

- Part II : We prove that if the graph G_C has a a vertex cover of size $|V| - k$, then the compliment graph G has a clique of size k.
- Claim : K-Clique $\leq_P$ Vertex-Cover
- Proof: Let $V - S$ be the vertex cover of size $|V| - k$ in G_C then we have to show that S is a clique of size k in G_C of size $|V| - k$
- We start with the following:
    - Consider an (edge (v, w) belonging to G_C. Now since $V - S$ is forming a vertex cover in G_C and since edge (v, w) belongs to G_C we have the following two cases:
    - ... ...
    - ... ...

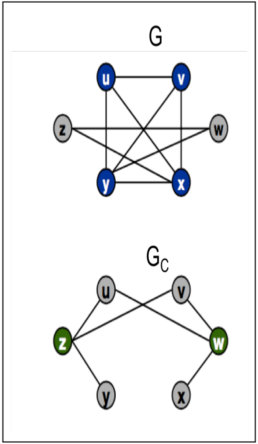


Figure: K-Clique? Vertex Cover?

Reduction By Using Gadgets

The Set Cover problem

- An instance of Set Cover is given by a ground set $U = x_1, x_2, x_3, \ldots\ldots.x_n$, subsets S $\subseteq$ U of that ground set, and an integer k.
- The question is, is it possible to select a collection C of at most k of these subsets such that taken together, they *cover* all of U?
- That is, is there a n set $C \subseteq 1, 2, \ldots\ldots, m$ such that $|C| = k$ and $\cup_{i \epsilon C}\ S_i =$ U ?

U = { 1, 2, 3, 4, 5, 6, 7 }
k = 2

S_a = {3, 7}　　S_b = {2, 4}

S_c = {3, 4, 5, 6}　　S_d = {5}

S_e = {1}　　S_f = {1, 2, 6, 7}

Figure: What are S_c and S_f shown in red?

The Set Cover problem: Applications

- Sample application.
  - m available pieces of software.
  - Set U of n capabilities that we would like our system to have.
  - The i_{th} piece of software provides the set $S_i \subseteq$ U of capabilities.
  - Goal achieve all n capabilities using fewest pieces of software.

U = { 1, 2, 3, 4, 5, 6, 7 }
k = 2

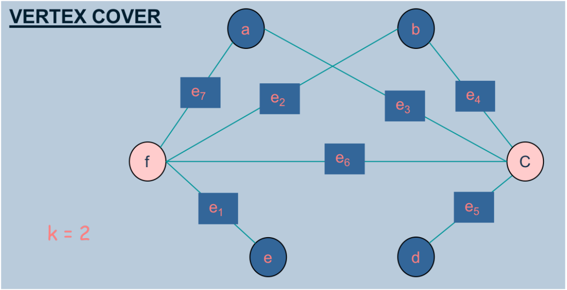
S_a = {3, 7} S_b = {2, 4}

S_c = {3, 4, 5, 6} S_d = {5}

S_e = {1} S_f = {1, 2, 6, 7}

Figure: What are S_c and S_f shown in red?

Vertex Cover $\equiv_P$ SetCover



SET COVER

U = { 1, 2, 3, 4, 5, 6, 7 }, k = 2

S_a = {3, 7}, **S_b = {2, 4},** **S_c = {3, 4, 5, 6},** **S_d = {5},** **S_e = {1},**

S_f= {1, 2, 6, 7}

Figure: Vertex Cover $\equiv_P$ SetCover.

K-Clique $\equiv_P$ SAT

- Given a graph G=(V,E) it has a clique of size k if and only if there is a SAT expression μ with k clauses that is satisfiable.

K-Clique $\equiv_P$ SAT

- Given a graph G=(V,E) it has a clique of size k if and only if there is a SAT expression μ with k clauses that is satisfiable.
- As an illustration consider that we are given a SAT expression μ as follows: $\mu = (x_1 + \bar{x_2})(x_3 + x_4 + \bar{x_2})(\bar{x_1} + \bar{x_4})$

K-Clique $\equiv_P$ SAT

- Given a graph G=(V,E) it has a clique of size k if and only if there is a SAT expression μ with k clauses that is satisfiable.
- As an illustration consider that we are given a SAT expression μ as follows: $\mu = (x_1 + \bar{x_2})(x_3 + x_4 + \bar{x_2})(\bar{x_1} + \bar{x_4})$
- μ is satisfiable if and only if we have a graph G=(V,E) has a clique has size of k.

K-Clique $\equiv_P$ SAT

- Given a graph G=(V,E) it has a clique of size k if and only if there is a SAT expression μ with k clauses that is satisfiable.
- As an illustration consider that we are given a SAT expression μ as follows: $\mu = (x_1 + \bar{x_2})(x_3 + x_4 + \bar{x_2})(\bar{x_1} + \bar{x_4})$
- μ is satisfiable if and only if we have a graph G=(V,E) has a clique has size of k.
- Approach is to create an appropriate gadget that can be used as a tool to prove the equivalence of the two problems.

K-Clique $\equiv_P$ SAT

- Given a graph G=(V,E) it has a clique of size k if and only if there is a SAT expression μ with k clauses that is satisfiable.
- As an illustration consider that we are given a SAT expression μ as follows: $\mu = (x_1 + \bar{x_2})(x_3 + x_4 + \bar{x_2})(\bar{x_1} + \bar{x_4})$
- μ is satisfiable if and only if we have a graph G=(V,E) has a clique has size of k.
- Approach is to create an appropriate gadget that can be used as a tool to prove the equivalence of the two problems.
- Design rules:

K-Clique $\equiv_P$ SAT

- Given a graph G=(V,E) it has a clique of size k if and only if there is a SAT expression μ with k clauses that is satisfiable.
- As an illustration consider that we are given a SAT expression μ as follows: $\mu = (x_1 + \bar{x_2})(x_3 + x_4 + \bar{x_2})(\bar{x_1} + \bar{x_4})$
- μ is satisfiable if and only if we have a graph G=(V,E) has a clique has size of k.
- Approach is to create an appropriate gadget that can be used as a tool to prove the equivalence of the two problems.
- Design rules:
    - For nodes

K-Clique $\equiv_P$ SAT

- Given a graph G=(V,E) it has a clique of size k if and only if there is a SAT expression μ with k clauses that is satisfiable.
- As an illustration consider that we are given a SAT expression μ as follows: $\mu = (x_1 + \bar{x_2})(x_3 + x_4 + \bar{x_2})(\bar{x_1} + \bar{x_4})$
- μ is satisfiable if and only if we have a graph G=(V,E) has a clique has size of k.
- Approach is to create an appropriate gadget that can be used as a tool to prove the equivalence of the two problems.
- Design rules:
    - For nodes
        - the number of nodes in the graph will be equal to the number of literals in the CNF with vertices grouped in k classes.

K-Clique $\equiv_P$ SAT

- Given a graph G=(V,E) it has a clique of size k if and only if there is a SAT expression μ with k clauses that is satisfiable.
- As an illustration consider that we are given a SAT expression μ as follows: $\mu = (x_1 + \bar{x_2})(x_3 + x_4 + \bar{x_2})(\bar{x_1} + \bar{x_4})$
- μ is satisfiable if and only if we have a graph G=(V,E) has a clique has size of k.
- Approach is to create an appropriate gadget that can be used as a tool to prove the equivalence of the two problems.
- Design rules:
  - For nodes
    - the number of nodes in the graph will be equal to the number of literals in the CNF with vertices grouped in k classes.
  - For edges

K-Clique $\equiv_P$ SAT

- Given a graph G=(V,E) it has a clique of size k if and only if there is a SAT expression μ with k clauses that is satisfiable.
- As an illustration consider that we are given a SAT expression μ as follows: $\mu = (x_1 + \bar{x_2})(x_3 + x_4 + \bar{x_2})(\bar{x_1} + \bar{x_4})$
- μ is satisfiable if and only if we have a graph G=(V,E) has a clique has size of k.
- Approach is to create an appropriate gadget that can be used as a tool to prove the equivalence of the two problems.
- Design rules:
    - For nodes
        - the number of nodes in the graph will be equal to the number of literals in the CNF with vertices grouped in k classes.
    - For edges
        - do not connect any two vertices in the same class

K-Clique $\equiv_P$ SAT

- Given a graph G=(V,E) it has a clique of size k if and only if there is a SAT expression μ with k clauses that is satisfiable.
- As an illustration consider that we are given a SAT expression μ as follows: $\mu = (x_1 + \bar{x_2})(x_3 + x_4 + \bar{x_2})(\bar{x_1} + \bar{x_4})$
- μ is satisfiable if and only if we have a graph G=(V,E) has a clique has size of k.
- Approach is to create an appropriate gadget that can be used as a tool to prove the equivalence of the two problems.
- Design rules:
  - For nodes
    - the number of nodes in the graph will be equal to the number of literals in the CNF with vertices grouped in k classes.
  - For edges
    - do not connect any two vertices in the same class
    - do not connect any two vertices if they are complement of each other, even if they are across different classes

K-Clique $\equiv_P$ SAT

- Given a graph G=(V,E) it has a clique of size k if and only if there is a SAT expression μ with k clauses that is satisfiable.
- As an illustration consider that we are given a SAT expression μ as follows: $\mu = (x_1 + \bar{x_2})(x_3 + x_4 + \bar{x_2})(\bar{x_1} + \bar{x_4})$
- μ is satisfiable if and only if we have a graph G=(V,E) has a clique has size of k.
- Approach is to create an appropriate gadget that can be used as a tool to prove the equivalence of the two problems.
- Design rules:
  - For nodes
    - the number of nodes in the graph will be equal to the number of literals in the CNF with vertices grouped in k classes.
  - For edges
    - do not connect any two vertices in the same class
    - do not connect any two vertices if they are complement of each other, even if they are across different classes
    - otherwise connect all the other vertices across different classes.

Vertex Cover $\equiv_P$ SetCover.

Gadget design for $\mu = (x_1 + \bar{x_2})(x_3 + x_4 + \bar{x_2})(\bar{x_1} + \bar{x_4})$

Blank

Blank