Hard Problems (NP problems)

Arash Rafiey

26 January 2016

So far we have seen polynomial time problems and we have designed (attempt) efficient algorithm to solve them.

So far we have seen polynomial time problems and we have designed (attempt) efficient algorithm to solve them.

We are going to see that most of the problems are difficult (what do we mean?)

So far we have seen polynomial time problems and we have designed (attempt) efficient algorithm to solve them.

We are going to see that most of the problems are difficult (what do we mean?)

That means there are not known polynomial algorithms for solving them.

So far we have seen polynomial time problems and we have designed (attempt) efficient algorithm to solve them.

We are going to see that most of the problems are difficult (what do we mean?)

That means there are not known polynomial algorithms for solving them.

There has been a lot of works trying to classify hard problems and there has been a lot of progress in this regard.

One way of classification could be the following :

We can verify a given answer to problem X is valid or not by spending a polynomial amount of time $\mathcal{O}(|X|^c)$, c is constant number (independent from $|X|$).

One way of classification could be the following :

We can verify a given answer to problem X is valid or not by spending a polynomial amount of time $\mathcal{O}(|X|^c)$, c is constant number (independent from $|X|$).

Example :

Independent Set Problem : Consider graph G with n nodes. Decide whether a set $I \subseteq V(G)$ is an independent set of size at least k.
If $|I| < k$ then we say no.

One way of classification could be the following :

We can verify a given answer to problem X is valid or not by spending a polynomial amount of time $\mathcal{O}(|X|^c)$, c is constant number (independent from $|X|$).

Example :

Independent Set Problem : Consider graph G with n nodes. Decide whether a set $I \subseteq V(G)$ is an independent set of size at least k.
If $|I| < k$ then we say no.

Otherwise :
We consider every pair of nodes u, v in I and if there is an edge between them then we say no. Otherwise we say yes.

It takes $\mathcal{O}(|I|^2)$ and since $|I| \leq n$, it takes $O(n^2)$ to verify.

Example :

Consider digraph D with n nodes. We can decide in $\mathcal{O}(n)$ whether given sequence $u_1, u_2, \ldots, u_n, u_{n+1}$ of the nodes in D provide a Hamiltonian cycle.

Example :

Consider digraph D with n nodes. We can decide in $\mathcal{O}(n)$ whether given sequence $u_1, u_2, \ldots, u_n, u_{n+1}$ of the nodes in D provide a Hamiltonian cycle.

1) If $u_i u_{i+1}$ is an arc for every $1 \leq i \leq n$,
2) $u_1 = u_{n+1}$,
3) each u_i, $1 \leq i \leq n$ appears once
Then the sequence is a Hamiltonian cycle.
It takes $\mathcal{O}(n)$.

Example :

Consider digraph D with n nodes. We can decide in $\mathcal{O}(n)$ whether given sequence $u_1, u_2, \ldots, u_n, u_{n+1}$ of the nodes in D provide a Hamiltonian cycle.

1) If $u_i u_{i+1}$ is an arc for every $1 \leq i \leq n$,
2) $u_1 = u_{n+1}$,
3) each u_i, $1 \leq i \leq n$ appears once
Then the sequence is a Hamiltonian cycle.
It takes $\mathcal{O}(n)$.

Traveling Salesman Problem (TSP) : Consider digraph D with n nodes where each arc has a non-negative weight. We can decide in $\mathcal{O}(n)$ whether a given sequence $u_1, u_2, \ldots, u_n, u_{n+1}$ of the nodes in D is a Hamiltonian cycle with weight at most W.

Example :

Consider digraph D with n nodes. We can decide in $\mathcal{O}(n)$ whether given sequence $u_1, u_2, \ldots, u_n, u_{n+1}$ of the nodes in D provide a Hamiltonian cycle.

1) If $u_i u_{i+1}$ is an arc for every $1 \leq i \leq n$,
2) $u_1 = u_{n+1}$,
3) each u_i, $1 \leq i \leq n$ appears once
Then the sequence is a Hamiltonian cycle.
It takes $\mathcal{O}(n)$.

Traveling Salesman Problem (TSP) : Consider digraph D with n nodes where each arc has a non-negative weight. We can decide in $\mathcal{O}(n)$ whether a given sequence $u_1, u_2, \ldots, u_n, u_{n+1}$ of the nodes in D is a Hamiltonian cycle with weight at most W.

Check whether the sequence is a Hamiltonian cycle
Check whether its cost is at most W.

Example :

Given a CNF formula with variables $x_1, x_2, \ldots, x_n$ and their negations. Decide whether there is an assignment of the variables (true or false) to satisfy all the clauses.

Example :

Given a CNF formula with variables $x_1, x_2, \ldots, x_n$ and their negations. Decide whether there is an assignment of the variables (true or false) to satisfy all the clauses.

$$F = (\neg x \vee y \vee z) \wedge (x \vee \neg w \vee v) \wedge (\neg y \vee w \vee \neg v) \wedge (x \vee \neg v \vee z) \wedge (w \vee v \vee \neg z)$$

Example :

Given a CNF formula with variables $x_1, x_2, \ldots, x_n$ and their negations. Decide whether there is an assignment of the variables (true or false) to satisfy all the clauses.

$$F = (\neg x \vee y \vee z) \wedge (x \vee \neg w \vee v) \wedge (\neg y \vee w \vee \neg v) \wedge (x \vee \neg v \vee z) \wedge (w \vee v \vee \neg z)$$

For $x = y = z = 0, v = w = 1$ F is not satisfiable.

Example :

Given a CNF formula with variables $x_1, x_2, \ldots, x_n$ and their negations. Decide whether there is an assignment of the variables (true or false) to satisfy all the clauses.

$$F = (\neg x \vee y \vee z) \wedge (x \vee \neg w \vee v) \wedge (\neg y \vee w \vee \neg v) \wedge (x \vee \neg v \vee z) \wedge (w \vee v \vee \neg z)$$

For $x = y = z = 0, v = w = 1$ F is not satisfiable.

For $x = y = 0, z = v = w = 1$ F is satisfiable.

Example :

Given a CNF formula with variables $x_1, x_2, \ldots, x_n$ and their negations. Decide whether there is an assignment of the variables (true or false) to satisfy all the clauses.

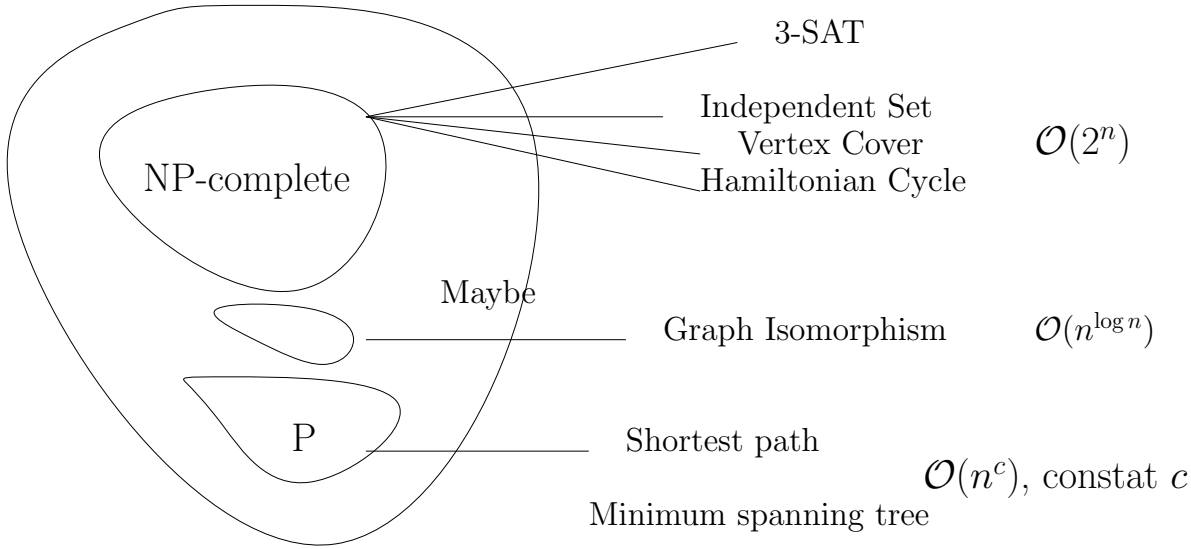
$$F = (\neg x \vee y \vee z) \wedge (x \vee \neg w \vee v) \wedge (\neg y \vee w \vee \neg v) \wedge (x \vee \neg v \vee z) \wedge (w \vee v \vee \neg z)$$

For $x = y = z = 0, v = w = 1$ F is not satisfiable.

For $x = y = 0, z = v = w = 1$ F is satisfiable.

If there are m clauses and n variables in F then we can check in $\mathcal{O}(mn)$ whether F is satisfiable or not (the size of F is the number of clauses + the number of variables in F).

NP is the class of problems X where verifying the solution for an instance $x \in X$ takes $\mathcal{O}(|x|^c)$, c is a constant .



Another way of classification could be the following :

The class of problems $\mathcal{S}$ where each pair of problems in S can be reduced to each other.

Another way of classification could be the following :

The class of problems $\mathcal{S}$ where each pair of problems in S can be reduced to each other.

Can arbitrary instance of problem Y be solved using a polynomial number of computational steps, plus a polynomial number of calls to a black box that solves problem X ?

Another way of classification could be the following :

The class of problems $\mathcal{S}$ where each pair of problems in S can be reduced to each other.

Can arbitrary instance of problem Y be solved using a polynomial number of computational steps, plus a polynomial number of calls to a black box that solves problem X ?

If the answer to this question is yes then we write $Y \leq_p X$; "Y is polynomial time reducible to X" or "X is at least as hard as Y".

Another way of classification could be the following :

The class of problems $\mathcal{S}$ where each pair of problems in S can be reduced to each other.

Can arbitrary instance of problem Y be solved using a polynomial number of computational steps, plus a polynomial number of calls to a black box that solves problem X ?

If the answer to this question is yes then we write $Y \leq_p X$; "Y is polynomial time reducible to X" or "X is at least as hard as Y".

Example : Circular Scheduling (independent set when we have circular arcs) Problem Y.
We use Interval Scheduling solution (procedure, tool box) to solve an instance of Circular Scheduling (call n times).

Suppose $Y \leq_p X$. If X can be solved in polynomial time, then Y can be solved in polynomial time.

Suppose $Y \leq_p X$. If X can be solved in polynomial time, then Y can be solved in polynomial time.

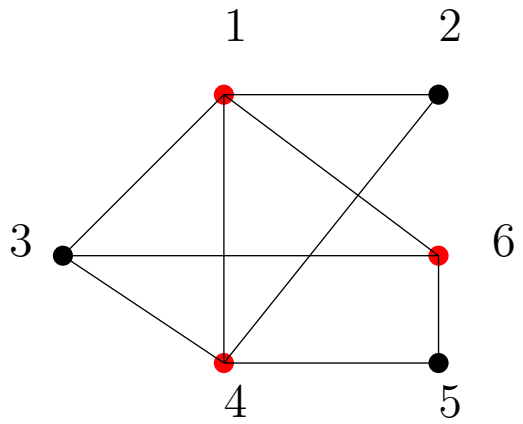
Suppose $Y \leq_p X$. If Y can not be solved in polynomial time, then X can not be solved in polynomial time.

First Reduction Example :

Vertex Cover : Given a graph $G = (V, E)$, we say that a set $S \subseteq V$ is a vertex cover if every edge $e \in E$ has at least one end in S.

First Reduction Example :

Vertex Cover : Given a graph $G = (V, E)$, we say that a set $S \subseteq V$ is a vertex cover if every edge $e \in E$ has at least one end in S.



$S = \{1, 4, 6\}$

Problem

Given a graph G and a number k, does G contain a vertex cover of size at most k ?

Theorem

Let $G = (V, E)$ be a graph. Then S is an independent set if and only if its complement $V - S$ is a vertex cover.

Theorem

Let $G = (V, E)$ be a graph. Then S is an independent set if and only if its complement $V - S$ is a vertex cover.

Proof.

First suppose S is an independent S. Consider an arbitrary edge uv. Since S is independent, it can not contain both u and v; so one of them is in $V - S$. Therefore $V - S$ is a vertex cover for G.

Theorem

Let $G = (V, E)$ be a graph. Then S is an independent set if and only if its complement $V - S$ is a vertex cover.

Proof.

First suppose S is an independent S. Consider an arbitrary edge uv. Since S is independent, it can not contain both u and v; so one of them is in $V - S$. Therefore $V - S$ is a vertex cover for G.

Conversely, suppose $V - S$ is a vertex cover for G. Now for every edges uv of E, at least one of the u, v is in $V - S$ and hence S is an independent set. □

Theorem

Independent Set $\leq_p$ *Vertex Cover.*

Theorem

Independent Set $\leq_p$ *Vertex Cover.*

Proof.

If we have a block box to solve Vertex Cover, then we can decide whether G has an independent set of size at least k by asking the black box whether G has a vertex cover of size at most $n - k$. □

Theorem

Vertex Cover $\leq_p$ *Independent Set.*

Theorem

Vertex Cover $\leq_p$ *Independent Set.*

Proof.

If we have a block box to solve Independent Set, then we can decide whether G has a vertex cover of size at most k by asking the black box whether G has an independent set of size at least $n - k$. □

More General Case Reduction : Vertex Cover to Set Cover

Set Cover : Given a set U of n elements, a collection $S_1, S_2, \ldots, S_m$ of subsets of U, and a number k, does there exists a collection of at most k of these sets whose union is equal to all of U.

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

$$S_1 = \{1, 2\} \qquad S_2 = \{3, 4, 6\} \qquad S_3 = \{1, 7\}$$

$$S_4 = \{2, 4, 5\} \qquad S_5 = \{3, 5, 6, 7\}$$

More General Case Reduction : Vertex Cover to Set Cover

Set Cover : Given a set U of n elements, a collection $S_1, S_2, \ldots, S_m$ of subsets of U, and a number k, does there exists a collection of at most k of these sets whose union is equal to all of U.

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

$$S_1 = \{1, 2\} \qquad S_2 = \{3, 4, 6\} \qquad S_3 = \{1, 7\}$$

$$S_4 = \{2, 4, 5\} \qquad S_5 = \{3, 5, 6, 7\}$$

We should pick S_2, S_3, S_4 or S_1, S_2, S_5.

Theorem

Vertex Cover $\leq_p$ *Set Cover.*

Proof.

From an arbitrary instance of Vertex Cover we construct an instance of Set Cover and we use the block for solving Set Cover, so we can solve vertex cover.

Theorem

Vertex Cover $\leq_p$ *Set Cover.*

Proof.

From an arbitrary instance of Vertex Cover we construct an instance of Set Cover and we use the block for solving Set Cover, so we can solve vertex cover.

Let $G = (V, E)$ be an arbitrary graph and a given number k (look for a vertex cover of size at most k).
When we pick a vertex x in G and place it in the Vertex cover then we cover all the edges incident to x. So we can define U to be set E.

Theorem

Vertex Cover $\leq_p$ *Set Cover.*

Proof.

From an arbitrary instance of Vertex Cover we construct an instance of Set Cover and we use the block for solving Set Cover, so we can solve vertex cover.

Let $G = (V, E)$ be an arbitrary graph and a given number k (look for a vertex cover of size at most k).
When we pick a vertex x in G and place it in the Vertex cover then we cover all the edges incident to x. So we can define U to be set E.

S_i for $i \in V$ be the set of edges $\{ij_1, ij_2, \ldots, ij_t\}$; ($j_1, j_2, \ldots, j_t$ are the neighbors of i in G).

Proof.

Now if we can cover U by picking at most k sets then G has a vertex cover of size at most k.

If $S_{i_1}, S_{i_2}, \ldots, S_{i_\ell}$, $\ell \leq k$ are the sets to cover U, then every edge in E is incident to one of $i_1, i_2, \ldots, i_\ell$. So $i_1, i_2, \ldots, i_\ell$ is a vertex cover.

Proof.

Now if we can cover U by picking at most k sets then G has a vertex cover of size at most k.
If $S_{i_1}, S_{i_2}, \ldots, S_{i_\ell}$, $\ell \leq k$ are the sets to cover U, then every edge in E is incident to one of $i_1, i_2, \ldots, i_\ell$. So $i_1, i_2, \ldots, i_\ell$ is a vertex cover.

Conversely, if $\{i_1, i_2, \ldots, i_\ell\}$ is a vertex cover of size $\ell \leq k$, then the sets $S_{i_1}, S_{i_2}, \ldots, S_{i_\ell}$ is a cover for U.

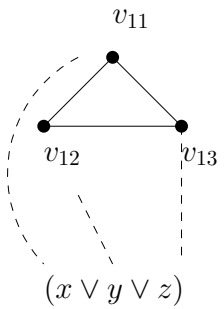
Reduction from 3SAT

3-SAT $\leq_p$ Independent Set. We want to show that Independent Set is at least as hard as 3- Satisfiability .

Reduction from 3SAT

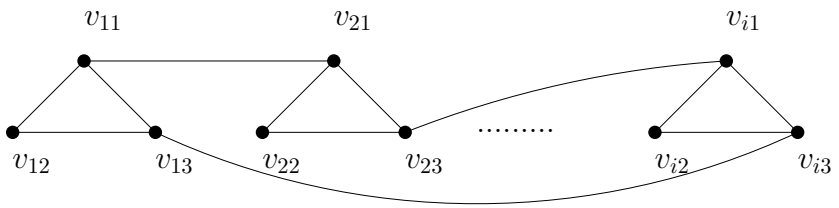
3-SAT $\leq_p$ Independent Set. We want to show that Independent Set is at least as hard as 3- Satisfiability .

We should transform an arbitrary instance of 3-SAT to an instance of an independent set.
Let $F = C_1 \wedge C_2 \wedge \cdots \wedge C_k$.
For each clause $C_i = (x \vee y \vee z)$ we consider three nodes v_{i1}, v_{i2}, v_{i3} representing x, y, z and make a triangle.



Our intuition is if we choose v_{i1} in the independent set then we set $x = 1$.

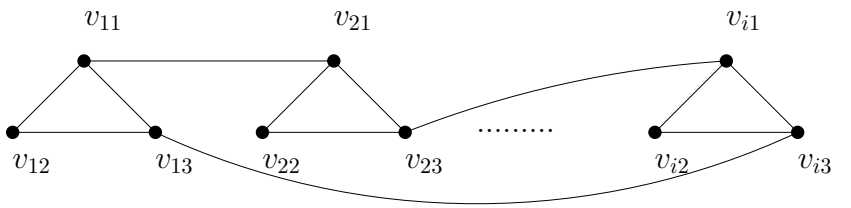
We also note that if we set $x = 1$ then $\neg x = 0$ and the node representing $\neg x$ should not be in the independent set. Therefore : We connect v_{i1} to v_{j1} if $C_j = (\neg x \vee u \vee w)$ or connect v_{i1} to v_{j2} if $C_j = (u \vee \neg x \vee w)$.



$(x \vee y \vee z) \quad \bigwedge \quad (\neg x \vee u \vee w) \qquad \bigwedge \quad (\neg w \vee v \vee \neg z)$

Construct graph G.

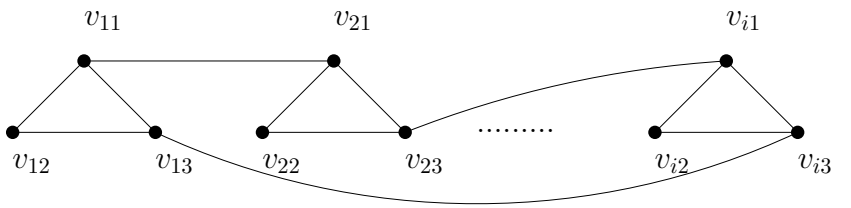
We also note that if we set $x = 1$ then $\neg x = 0$ and the node representing $\neg x$ should not be in the independent set. Therefore : We connect v_{i1} to v_{j1} if $C_j = (\neg x \vee u \vee w)$ or connect v_{i1} to v_{j2} if $C_j = (u \vee \neg x \vee w)$.



$(x \vee y \vee z) \quad \wedge \quad (\neg x \vee u \vee w) \qquad \wedge \quad (\neg w \vee v \vee \neg z)$

Construct graph G. Show that if there is an independent set in G of size k then there is an assignment for the variables to satisfy F.

We also note that if we set $x = 1$ then $\neg x = 0$ and the node representing $\neg x$ should not be in the independent set. Therefore : We connect v_{i1} to v_{j1} if $C_j = (\neg x \vee u \vee w)$ or connect v_{i1} to v_{j2} if $C_j = (u \vee \neg x \vee w)$.

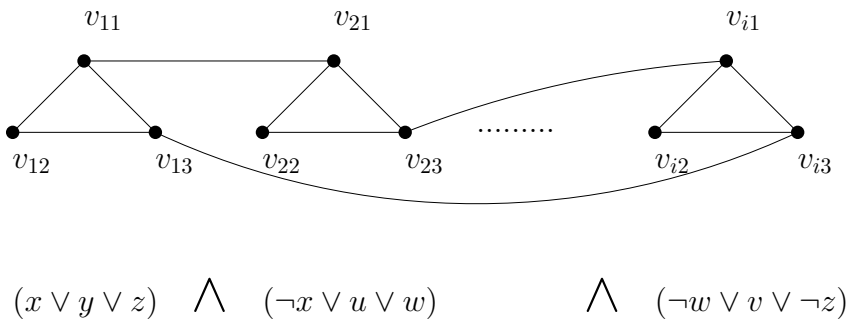


$(x \vee y \vee z) \quad \wedge \quad (\neg x \vee u \vee w) \qquad \wedge \quad (\neg w \vee v \vee \neg z)$

Construct graph G. Show that if there is an independent set in G of size k then there is an assignment for the variables to satisfy F.

If v_{i1} is in the independent then set its corresponding literal to be 1.

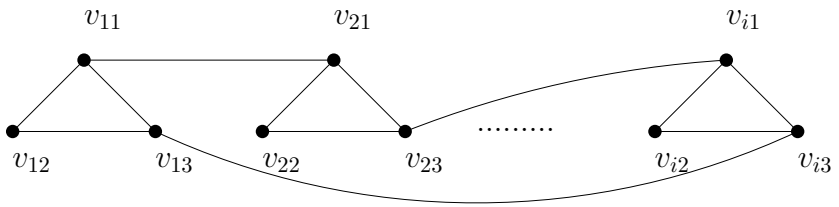
We also note that if we set $x = 1$ then $\neg x = 0$ and the node representing $\neg x$ should not be in the independent set. Therefore : We connect v_{i1} to v_{j1} if $C_j = (\neg x \vee u \vee w)$ or connect v_{i1} to v_{j2} if $C_j = (u \vee \neg x \vee w)$.



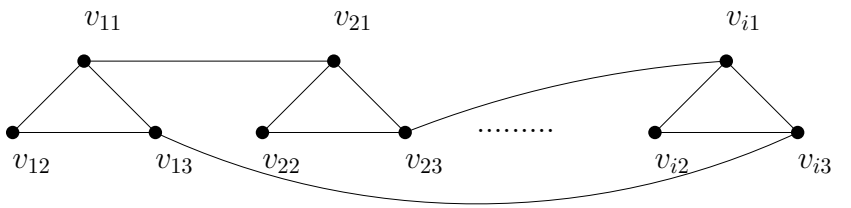
Construct graph G. Show that if there is an independent set in G of size k then there is an assignment for the variables to satisfy F.

If v_{i1} is in the independent then set its corresponding literal to be 1.

From each of the triangles we choose exactly one node in the independent set. Therefore for each clause at least one literal is set to 1 and hence F is satisfied.

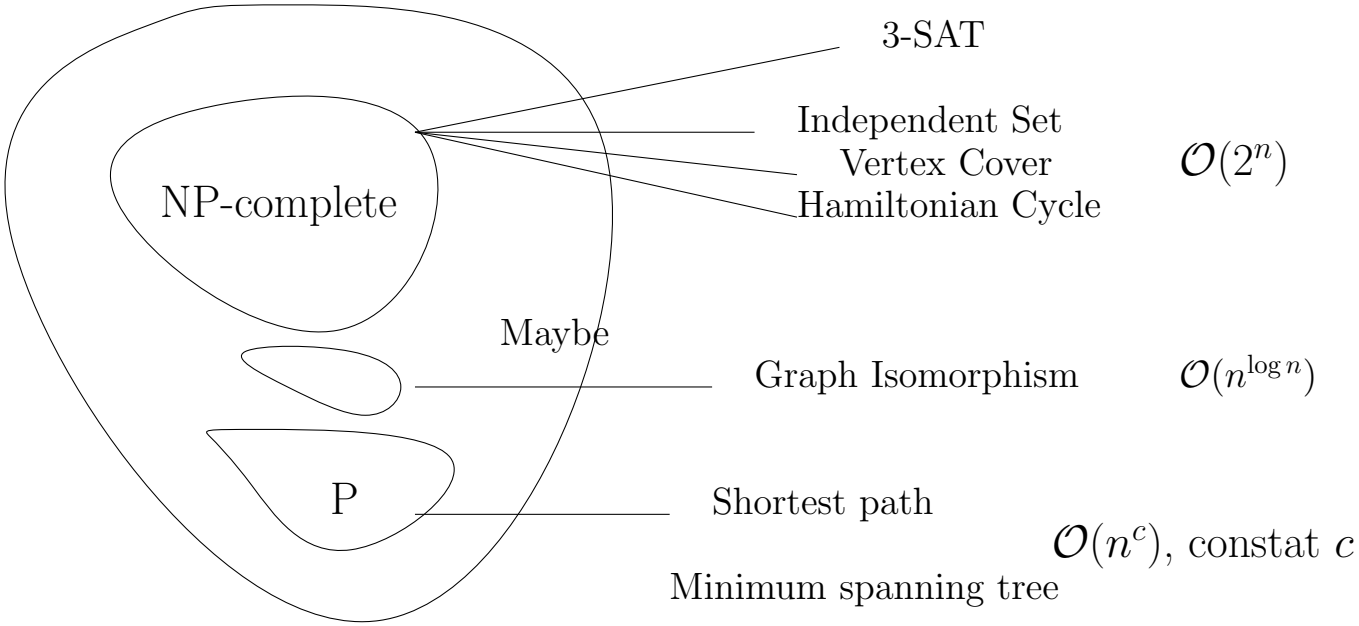


$$(x \vee y \vee z) \quad \bigwedge \quad (\neg x \vee u \vee w) \qquad \bigwedge \quad (\neg w \vee v \vee \neg z)$$



$$(x \vee y \vee z) \quad \bigwedge \quad (\neg x \vee u \vee w) \qquad \bigwedge \quad (\neg w \vee v \vee \neg z)$$

Conversely : We also observe that if F is satisfied then there is an independent set in G of size at least k (if x in clause C_i is 1 then add x to the independent set, and don't add anything else from clause C_i to independent set)


3-SAT
Independent Set
Vertex Cover
Hamiltonian Cycle
$\mathcal{O}(2^n)$
NP-complete
Maybe
Graph Isomorphism
$\mathcal{O}(n^{\log n})$
P
Shortest path
$\mathcal{O}(n^c)$, constat c
Minimum spanning tree

Hitting Set Problem:

Definition

We are given a set $U = \{e_1, e_2, e_3, \cdots, e_n\}$ and $S_1, S_2, \cdots, S_m$ are subsets of U and integer k. The goal is to find a subset of U with size at most k which has intersection with all $S_1, S_2, \cdots, S_m$.

Theorem

Hitting Set Problem is NP complete.

Proof.

We reduce Vertex Cover to Hitting Set!
From an arbitrary instance of vertex cover we construct an instance of hitting set.

□

How??

NP completeness of Hitting Set Problem:

Proof.

1) Consider graph $G = (V, E)$;
2) $U = V(G)$;
3) $S_i = \{u, v\}$; where (u, v) is an edge of G.
4) If C is a vertex cover for G of size $k \Rightarrow$ by definition then, for every edge (u, v) in G either $u \in C$ or $v \in C$. C is solution for Hitting Set because it must intersect every set S_i
5) Now suppose H is a hitting set of U of size k. Since H intersects every set, it has at least one endpoint of every edge. Set C to be H.

□

Dominating set problem is NP-complete

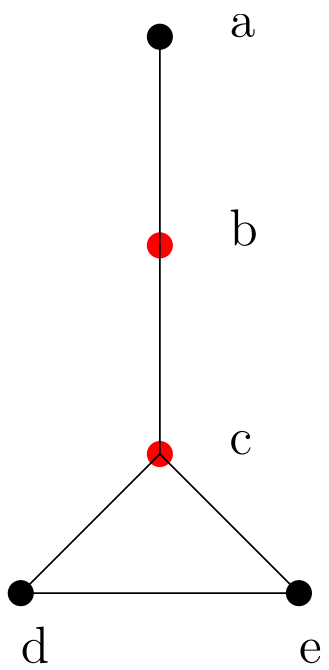
We are give a graph $G = (V, E)$ and an integer k. Decide whether there exists a set $S \subseteq V$ such that :

1. $|S| \leq k$
2. every vertex in $V - S$ has a neighbor in S.

Dominating set problem is NP-complete

We are give a graph $G = (V, E)$ and an integer k. Decide whether there exists a set $S \subseteq V$ such that :

1. $|S| \leq k$
2. every vertex in $V - S$ has a neighbor in S.



b,c is a dominating set

Dominating set problem is in NP. Because we can verify in polynomial time whether a given set S is a dominating set (is a valid solution).

Dominating set problem is in NP. Because we can verify in polynomial time whether a given set S is a dominating set (is a valid solution).
In order to prove dominating set is NP-complete. We should reduce a known NP-complete problem to dominating set (DS) .

Dominating set problem is in NP. Because we can verify in polynomial time whether a given set S is a dominating set (is a valid solution).
In order to prove dominating set is NP-complete. We should reduce a known NP-complete problem to dominating set (DS) .

Y $\leq_p$ Dominating Set

Dominating set problem is in NP. Because we can verify in polynomial time whether a given set S is a dominating set (is a valid solution).
In order to prove dominating set is NP-complete. We should reduce a known NP-complete problem to dominating set (DS) .

Y $\leq_p$ Dominating Set

We know set cover is NP-complete.

Dominating set problem is in NP. Because we can verify in polynomial time whether a given set S is a dominating set (is a valid solution).
In order to prove dominating set is NP-complete. We should reduce a known NP-complete problem to dominating set (DS) .

Y $\leq_p$ Dominating Set

We know set cover is NP-complete.

Set Cover $\leq_p$ Dominating Set

Set Cover : Given a set U of n elements, a collection $S_1, S_2, \ldots, S_m$ of subsets of U, and a number k, does there exists a collection of at most k of these sets whose union is equal to all of U.

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

$$S_1 = \{1, 2\} \qquad S_2 = \{3, 4, 6\} \qquad S_3 = \{1, 7\}$$

$$S_4 = \{2, 4, 5\} \qquad S_5 = \{3, 5, 6, 7\}$$

Set Cover : Given a set U of n elements, a collection $S_1, S_2, \ldots, S_m$ of subsets of U, and a number k, does there exists a collection of at most k of these sets whose union is equal to all of U.

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

$$S_1 = \{1, 2\} \qquad S_2 = \{3, 4, 6\} \qquad S_3 = \{1, 7\}$$

$$S_4 = \{2, 4, 5\} \qquad S_5 = \{3, 5, 6, 7\}$$

We should pick S_2, S_3, S_4 or S_1, S_2, S_5.

To complete a reduction from an arbitrary instance of set cover we construct an instance of a dominating set.

To complete a reduction from an arbitrary instance of set cover we construct an instance of a dominating set.

We can look at the set cover instance as a bipartite graph.

To complete a reduction from an arbitrary instance of set cover we construct an instance of a dominating set.

We can look at the set cover instance as a bipartite graph.

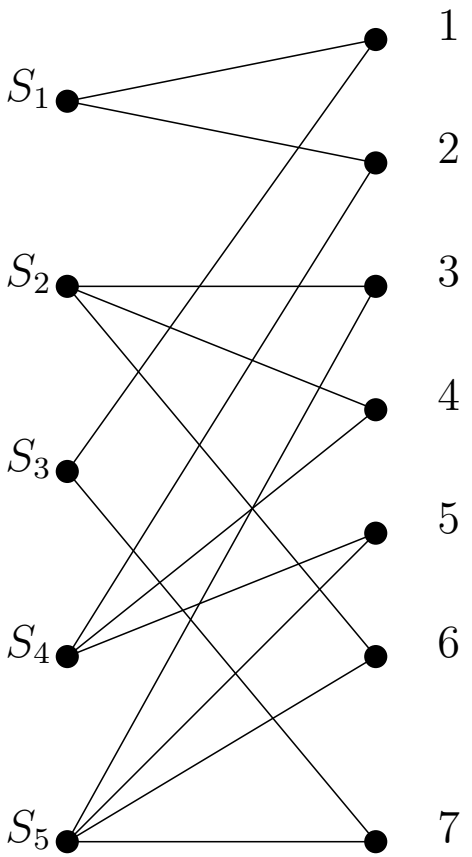
For each set S_i we consider a white node and for each element of the ground set U we consider a black node.

We connect a black node v to white node S_i if $v \in S_i$.

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

$$S_1 = \{1, 2\} \qquad S_2 = \{3, 4, 6\} \quad S_3 = \{1, 7\}$$

$$S_4 = \{2, 4, 5\} \quad S_5 = \{3, 5, 6, 7\}$$



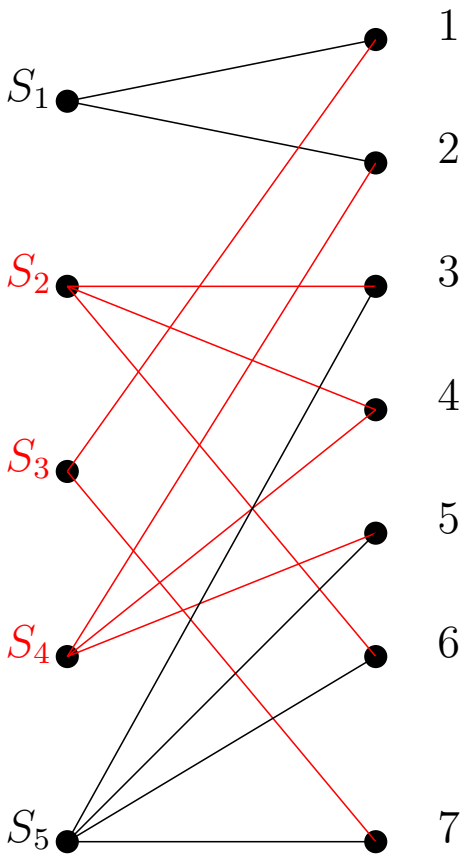
S_2, S_3, S_4 is a set cover since

$S_2 \cup S_3 \cup S_4 = \{1, 2, 3, 4, 5, 6, 7\}$

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

$$S_1 = \{1, 2\} \qquad S_2 = \{3, 4, 6\} \quad S_3 = \{1, 7\}$$

$$S_4 = \{2, 4, 5\} \quad S_5 = \{3, 5, 6, 7\}$$



S_2, S_3, S_4 is a set cover since

$S_2 \cup S_3 \cup S_4 = \{1, 2, 3, 4, 5, 6, 7\}$

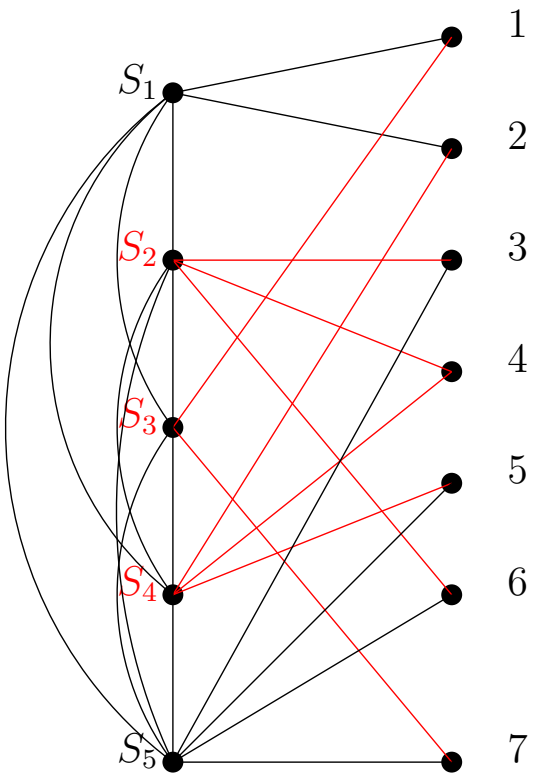
S_2, S_3, S_4 dominates all right nodes

Now we make a clique on the white vertices (on the set nodes).

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

$$S_1 = \{1, 2\} \qquad S_2 = \{3, 4, 6\} \quad S_3 = \{1, 7\}$$

$$S_4 = \{2, 4, 5\} \quad S_5 = \{3, 5, 6, 7\}$$



S_2, S_3, S_4 is a set cover since

$S_2 \cup S_3 \cup S_4 = \{1, 2, 3, 4, 5, 6, 7\}$

S_2, S_3, S_4 dominates all right nodes

S_2, S_3, S_4 domiates all others

Important : We should show that if there is a k collections of sets that covers entire U then there is set of nodes in new graph G that is dominating set.

Important : We should show that if there is a k collections of sets that covers entire U then there is set of nodes in new graph G that is dominating set.

Let $S_{i_1}, S_{i_2}, \ldots, S_{i_k}$ be a set cover for U.

Then set S of white nodes corresponding to $S_{i_1}, S_{i_2}, \ldots, S_{i_k}$ is a dominating set.

Because every black node has a neighbor in S and sice every white node is adjacent to all other black nodes, S is a dominating set.

Conversely : From a dominating set of size k we find a solution for set cover.

Conversely : From a dominating set of size k we find a solution for set cover.

Let DS be a dominating set of size k in G.

Conversely : From a dominating set of size k we find a solution for set cover.

Let DS be a dominating set of size k in G.

If DS contains only white vertices then we can obtain the corresponding sets and we get a set cover.

Conversely : From a dominating set of size k we find a solution for set cover.

Let DS be a dominating set of size k in G.

If DS contains only white vertices then we can obtain the corresponding sets and we get a set cover.

If DS has a black vertex v then we just simply replace it by a white vertex w representing to set S_i containing w

Conversely : From a dominating set of size k we find a solution for set cover.

Let DS be a dominating set of size k in G.

If DS contains only white vertices then we can obtain the corresponding sets and we get a set cover.

If DS has a black vertex v then we just simply replace it by a white vertex w representing to set S_i containing w

So we can assume that DS contains only white vertices.