10:49
28/09/2024
EVA TARDOS

P, NP, NP-Complete & NP-Hard

*** Polynomial-Time Reductions** : * Problem X is atleast as hard as problem Y.
* If we had a "black box" capable of solving X, then we could also solve Y.

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

Ans: Yes, then we write $Y \leq_p X$; we read this as "Y is polynomial-time reducible to X"
or
"X is at least as hard as Y w.r.t. polynomial time.

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

Ans: → We solved the Bipartite Matching Problem using a polynomial amount of preprocessing + the sol^n of a single instance of the Maximum-Flow Problem. Since the maximum-flow problem can be solved in polynomial time, we concluded that bipartite Matching could as well.

→ Similarly, we solved the foreground/background Image seg. problem using polynomial amount of preprocessing plus the sol^n of single instance of the Minimum-Cut problem

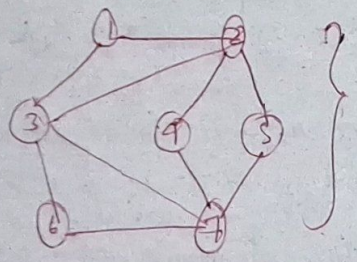
Q: Let $G = (V, E)$ be graph, Then S is an independent set if its complement V-S is a vertex cover.

Proof: Suppose that S is an independent set. Consider an arbitrary edge $e = (u, v)$. Since S is independent, it cannot be the

case that both u & v are in S; so one of them must be in V-S. It follows that every edge has at least one end in V-S & so V-S is a vertex cover.

Conversely, suppose that V-S is a vertex cover, consider any 2 nodes u & v in S. If they were joined by edge e, then neither of them would lie in V-S, contradicting our assumption that V-S is a vertex cover. It follows that no 2 nodes in S are joined by an edge & so S is an independent set.

Q: Independent Set $\leq_p$ Vertex Cover.



Independent Set: if no 2 nodes in $S \subseteq V$ are joined by an edge

Ex: $\{3, 4, 5\}$ → Smallest of size 3

$\{1, 4, 5, 6\}$ → Larger of size 4

Vertex Cover: if every edge $e \in E$ has at least one end in S. $S \subseteq V$

Ex- the set of nodes $\{1, 2, 6, 7\}$ is vertex cover of size 4

" " " " $\{2, 3, 7\}$ " " " " " " 3

Proof: If we have a black 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.

Q: Vertex Cover $\leq_p$ Independent Set.

Proof: If we have a black 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.

Q: Suppose $Y \leq_p X$. If Y cannot be solved in polynomial time, then X cannot be solved in polynomial time.

Proof: If we have a problem Y that is known to be hard & we show that $Y \leq_p X$, then the hardness has "spread" to X; X must be hard or else it could be used to solve Y.

Q: Vertex Cover $\leq_p$ Set cover

Proof: Suppose we have access to a black box that can solve Set cover, & consider an arbitrary instances of vertex cover, specified by a graph $G = (V, E)$ and a number k.

How can we use the black box to help us?

Our goal is to cover the edges in E. So, we formulate an instance of set cover in which the ground set U is equal to E.

Each time we pick a vertex in the Vertex Cover Problem, we cover all the edges incident to it; thus for each vertex $i \in V$, we add a set $S_i \subseteq U$ to our Set cover instance, consisting of all the edges in G incident to i.

We claim that U can be covered with at most k of the sets $S_1, \ldots, S_n$ iff G has a vertex cover of size at most k. This can be proved very easily. For if $S_{i_1} \ldots S_{i_\ell}$ are $\ell \leq k$ set that cover U. Then every edge in G is incident to one of the vertices $i_1, \ldots, i_\ell$ and so the set $\{i_1, \ldots, i_\ell\}$ is a vertex cover in G of size $\ell \leq k$.

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

Thus, given an instance of Vertex Cover, we formulate the instance of Set cover described in previous page & pass it to our black box. We answer yes iff the black box answers yes.

Set Packing Problem: Just as Set cover is a natural generalization of Vertex cover, there is a natural generalization of Independent Set as a packing problem for arbitrary sets.

"Given a set U of n elements, a collection $S_1, \dots, S_m$ of subsets of U, and a number k, does there exist a collection of at least k these sets with the property that no two of them intersect?"

Q: Independent Set $\leq_p$ Set Packing

proof similar to Vertex cover $\leq_p$ Set Cover.

The SAT and 3-SAT problem (The Satisfiability problem)

Suppose, we are given a set X of n Boolean variable $x_1, \dots x_n$ each can take the value 0 or 1 (T/F). By a term over X, we mean one of the variables x_i or its negation $\bar{x}_i$. Finally, a clause is simply a disjunction to distinct terms.

$$t_1 \vee t_2 \vee \dots \vee t_l.$$

Each $t_i \in \{x_1, x_2, \dots x_n, \bar{x}_1, \dots \bar{x}_n\}$. We say the clause has length l if it contains l terms.

or

$$C_1 \wedge C_2 \wedge \dots C_k$$

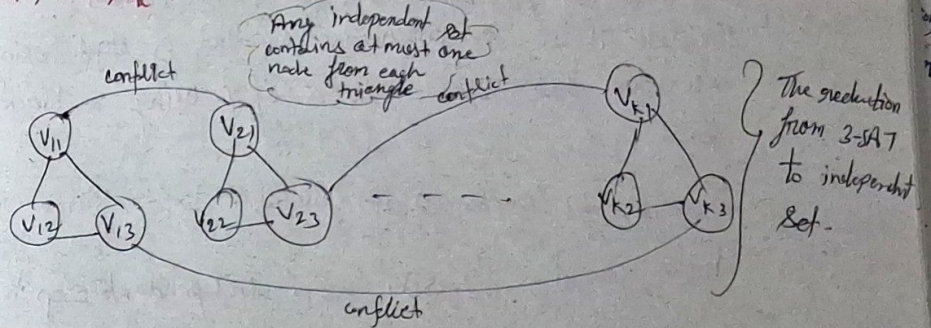
If it causes the conjuction, to evaluate to 1. In this case, we say that u is satisfying assignment with respect to $C_1 \dots C_k$ if that the set of clauses $C_1 \dots C_k$ is satisfiable.

SAT: Given a set of clauses $C_1, \dots C_k$ over a set of variables $X = \{x_1, \dots x_n\}$, does there exist a satisfying truth assignment.

3-SAT: Given a set of clauses $C_1, \dots C_k$, each of length 3, over a set of variables $X = \{x_1, \dots x_n\}$, does there exist a satisfying truth assignment.

Q: 3-SAT $\leq_p$ Independent Set

Proof: We have a black box for Independent set & want to solve an instance of 3-SAT consisting of variables $X = \{x_1, \dots x_n\}$ & clauses $C_1, \dots, C_k$.



2 possible ways:

→① One way to picture the 3-SAT instance was suggested earlier: You have to make an Independent 0/1 decision for each of the n variables & you succeed if you manage to achieve one of three ways of satisfying each clause.

→② A different way to picture the same 3-SAT instance is as follows: You have to choose one term from each clause & then find a truth assignment that causes all these terms to evaluate to 1, thereby satisfying all clauses. So you succeed if you select a term from each clause in such a way that no 2 selected terms "conflict"

we say that 2 term conflict if one is equal to variable x_i the other is equal to its negation $\bar{x}_i$. If we avoid conflicting terms, we can find a truth assignment that makes the selected terms from each clause evaluate to 1.

Q: If $Z \leq_p Y$, and $Y \leq_p X$ then $Z \leq_p X$

Proof:- Given a black box for X, we show how to solve an instance of Z; essentially, we just compose the two algorithms implied by $Z \leq_p Y$ & $Y \leq_p X$.
We run the algorithm for Z using a black box for Y, but each time the black box for Y is called, we simulate it in a polynomial number of steps using the algorithm that solves instances of Y using a black box for X.

Transitivity can be quite useful. Ex:- Since we have proved 3-SAT $\leq_p$ Independent Set $\leq_p$ Vertex Cover $\leq$ Set Cover

Hence, we can conclude that **3-SAT $\leq_p$ Set Cover**

NP: The act of searching for a string t that will cause an efficient certifier to accept the I/p s is often viewed as a nondeterministic search over the space of possible proofs t; for this reason, NP was named as an acronym for "Nondeterministic polynomial time".

Thus we say that B is an efficient certifier for a problem X if the following properties hold.

→ B is a polynomial-time algorithm that takes 2 I/p arguments s & t.

→ There is a polynomial function p so that for every string s. We have $s \in X$ iff $\exists$ a string $t \Rightarrow |t| \leq p(|s|)$ & $B(s,t) = yes$.

Q: $P \subseteq NP$

Proof: Consider a problem $X \in P$; this means that there is a polynomial-time algo. A that solves X. To show that $X \in NP$, we must show that there is an efficient certifier B for X.

We design B as follows: When presented with the I/p pair (s,t), the certifier B simply returns the value $A(s)$.

Think of B as a very "hands-on" manager that ignores the proposed proof t & simply solves the problem on its own. Why is B an efficient certifier for X? clearly it has polynomial running time, since A does. If a string $s \in X$, then for every t of length at most $p(|s|)$, we have $B(s,t) = yes$. On other hand, if $s \notin X$, then for every t of length at most $p(|s|)$, we have $B(s,t) = no$.

Q: Is there a problem in NP that does not belong to P?
Does P = NP?

The general belief is that $P \neq NP$ – and this is taken as a working hypothesis throughout the field – but there is not a lot of hard technical evidence for it. It is mostly based on the sense that P = NP would be too amazing to be true.
These problems simply cannot be solved in polynomial-time.

NP-Complete Problems

Arguably the most natural way to define a "hardest" Problem X is via the following 2 properties:

① $X \in NP$ ② $\forall Y \in NP, Y \leq_p X$.

In other words, we require, that every problem in NP can be reduced to X. We will call such an X an NP-Complete problem.

Q: Suppose X is an NP-Complete problem. Then X is solvable in polynomial time iff P = NP.

Proof: Clearly, if P = NP, then X can be solved in polynomial time since it belongs to NP.
Conversely, suppose that X can be solved in polynomial time, if Y is any other problem in NP, then $Y \leq_p X$ & so it follows that Y can be solved in polynomial time. Hence $NP \subseteq P$;

Q: If Y is an NP-Complete problem, and X is a Problem in NP with the property that $Y \leq_p X$, then X is NP-Complete.

Proof: Since $X \in NP$, we need only verify property (ii) of the definition. So, let Z be any problem in NP. We have $Z \leq_p Y$, by the NP-Completeness of Y, & $Y \leq_p X$ by assumption.

By [If $Z \leq_p X$ & $Y \leq_p X$, then $Z \leq_p X$], it follows that $Z \leq_p X$.
So while proving [circuit Satisfiability is NP-Complete] required the hard work of considering any posible problem in NP, proving further problems NP-Complete only requires a reduction from a single problem already known to be NP-Complete.

Q: 3-SAT is NP-Complete

Proof: Clearly 3-SAT is in NP, since we can verify in polynomial time that a proposed truth assignment satisfies the given set of clauses.
We will prove that it is NP-Complete via the reduction Circuit SAT $\leq_p$ 3-SAT.

Given an arbitrary istance of circuit Satisfiability, we will first construct an equivalent instance of SAT in which each clause contains at most three variables. Then we will convert this SAT instance to an equivalent one in which each clause has exactly three variables. This last collection of clauses will thus be an instance of 3-SAT. & hence will complete the reduction.

So consider an arbitrary circuit k. We associate a variable x_i with each node i of the circuit, to encode the truth value that the circuit holds at that node. There will be three cases depending on the three types of gates:

→① If node v is labeled with $\neg$, & its only entering edge is from node u, then we need to have $x_v = \overline{x_u}$, we gurentee this by adding two clauses $(x_u \vee x_u)$ & $(\overline{x_u} \vee \overline{x_u})$.

→② If node v is labeled with $\vee$, and its 2 entering edges are from nodes u & w, we need to have $x_v = x_u \vee x_w$. We guarantee this by adding the following clauses: $(x_u \vee \overline{x_v})$, $(x_w \vee \overline{x_v})$ and $(\overline{x_u} \vee x_u \vee x_w)$

→③ If node v is labeled with $\wedge$, and its 2 entering edges are from nodes u and w, we need to have $x_v = x_u \wedge x_w$. We gurantee this by adding the following clauses: $(\overline{x_v} \vee x_u)$, $(\overline{x_v} \vee x_w)$ & $(x_v \vee \overline{x_u} \vee \overline{x_w})$.

Finally, we need to gucarantee that the constants at the sources have their specified values and that the o/p evaluate to 1.

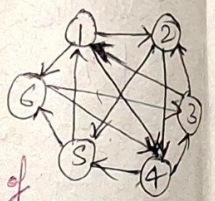
3-SAT $\leq_p$ Independent SAT $\leq_p$ Vertex Cover $\leq_p$ Set Cover.

Q: All of the following problems are NP-complete: Independent Set, set packing, Vertex Cover, Set Cover.

Proof: Each of these problem has the property that it is in NP & that 3-SAT (& hence Circuit Satisfiability) can be reduced to it.

Q: Hamiltonian Cycle is NP-complete.

Proof: We first show that Hamiltonian Cycle is in NP.



Given a directed graph $G = (V, E)$, a certificate that there is a solution would be the ordered list of the vertices on a Hamiltonian cycle. We could then check, in polynomial time, that this list of vertices does contain each vertex exactly once, & that each consecutive pair in the ordering is joined by an edge; this would establish that the ordering defines a Hamiltonian cycle.

Q: Traveling Salesman is NP-Complete.

Proof: It is easy to see that TSP is in NP: The certificate is a permutation of the cities, & a certifier checks that the length of the corresponding tour is at most the given bound.

We now show that Hamiltonian Cycle $\leq_p$ Traveling Salesman.

Given a directed graph $G = (V, E)$, we define the following instance of Traveling Salesman. We have a city v_i' for each node v_i of the following G. We define $d(v_i', v_j')$ to be 1 if there is an edge (v_i, v_j) in G & we define it to be 2 otherwise.

Graph Coloring Problem

Graph coloring refers to the same process on an undirected graph G. with the nodes playing the role of the regions to be colored, & the edges representing pairs that are neighbors. We seek to assign a color to each node of G so that if (u, v) is an edge, then u & v are assigned different colors; & the goal is to do this while using a small set of colors.

→ 3-coloring is NP-Complete

→ Subset Sum is NP-Complete

Q: Does NP = Co-NP?

Again, the widespread belief is that NP ≠ Co-NP: Just because the "yes" instances of a problem have short proofs, it is not clear why we should believe that the "no" instances have short proofs as well.

proving $NP \neq co\text{-}NP$ would be an even bigger step than proving $P \neq NP$.

Q: If $NP \neq co\text{-}NP$, then $P \neq NP$

Proof: We'll actually prove the contrapositive statement: $P = NP$ implies $NP = co\text{-}NP$. Essentially, the point is that P is closed under complementation; so if $P = NP$, then $NP = co\text{-}NP$ i.e. NP would be closed under complementation as well. More formally,

$$X \in NP \Rightarrow X \in P \Rightarrow \overline{X} \in P \Rightarrow \overline{X} \in NP \Rightarrow X \in co\text{-}NP$$

&

$$X \in co\text{-}NP \Rightarrow \overline{X} \in NP \Rightarrow \overline{X} \in P \Rightarrow X \in P \Rightarrow X \in NP$$

Hence, it would follow that $NP \subseteq co\text{-}NP$ & $co\text{-}NP \subseteq NP$

Hence, $NP = co\text{-}NP$.