

Lec27-NP-hardness

KT 8.1, 8.2, 8.3, 8.4

Sunday, November 12, 2023 7:26 PM

Polynomial time: e.g. $O(m^2n)$

Often we want poly-time because exp algs $O(2^n)$ are too slow.

Mistake: NP-hard \neq non-polynomial time
not the correct definition

Decision vs. Optimization problems

Optimization: output answer that minimizes/maximizes an objective function

Ex All of linear programming.

Shortest path

Minimum spanning tree.

Max - Flow

Decision: Yes/No answer to whether there exists a solution that is good enough

Ex \exists s-t path shorter than ϵ
 \exists MST with weight ≤ 100

Ex. Circulation with demands

Optimization \rightarrow Decision

Problem (MST) Given a graph $G=(V, E)$, with weights w_e on each edge in E , find a spanning tree of smallest total weight.

Problem (MST-Decision) Given a number C , and a graph $G=(V, E)$, with weights w_e on each edge in E , is there a spanning tree of total weight less than C

Glossary

Problem: a decision problem in general terms

Instance: A particular set of input into a problem:

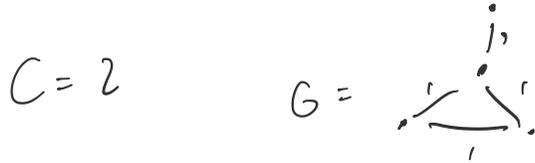
e.g. $C = 5$

$G =$



Yes-Instance: An instance with answer "YES"

No-Instance: An instance with answer "NO"



deciding: determining if an instance has answer "YES" or "NO"

Define: The class $P =$ set of problems.

A problem $Q \in P$ if

\exists algorithm A_Q s.t. \forall instances $I \in Q$,

$A_Q(I)$ runs in fewer than $\text{poly}(|I|)$ steps, and s.t.

if I is a YES-instance, $A_Q(I) = \text{YES}$

if I is a NO-instance, $A_Q(I) = \text{NO}$

on a deterministic Turing machine

P is the set of problems that can be decided in polynomial number of steps. i.e. "efficient" solutions.

Define: The class $NP =$ set of problems.

A problem $Q \in NP$ if a non-deterministic Turing machine can return YES on a YES instance in poly-time

Easier Definition: A problem $Q \in NP$ if ^{certificate} evidence of a YES-instance can be verified in poly-time. (on normal Turing machine)

Ex. (Traveling Salesman-Decision)

Given distances d_{ij} between n cities, and a number C ,

Ex. (Traveling Salesman)

Given distances d_{ij} between n cities, and a number C ,
is there a way to visit each city exactly once where
the total length of the route is $\leq C$.

If someone gives you a route, it is easy to check
if it provides a solution that makes this a YES-instance.

Ex. (Independent Set - Decision)

Given graph G , $\exists S \subseteq V$, $|S| \geq k$ s.t. no two nodes in
 S are connected by an edge?

Finding set is hard. (later)

Checking a set S^* I give you is easy.

S^* acts as a certificate that $\langle G, k \rangle$ is a Yes-instance.

Define: NP = set of problems Q for which

\exists algorithm C_Q s.t.

\forall instances $I \in Q$,

if I is yes-instance, $C_Q(I, S) = \text{YES}$ for some S

if I is no-instance, $C_Q(I, S) = \text{NO}$ $\forall S$.

and $C_Q(I, S)$ runs in $\text{poly}(|I|)$ time.

Verifier

(efficient certifier)

certificate

(nondeterministic polynomial)

(nondeterministic polynomial)

NP - set of problems with efficient certifier

P - set of problems with efficient certifier
(polynomial) that ignores the certificate

Thm $P \subseteq NP$.

proof. Suppose $X \in P$. Then \exists poly-time alg A that decides X .
to show $X \in NP$, need to design efficient certifier $C_X(I, S)$.

Just take $C_X(I, S) = A(I)$.



every problem with poly-time alg is in NP.

Question: $P \stackrel{?}{=} NP$

Know $P \subseteq NP$.

Does there \exists a problem in NP that is NOT in P?

Is checking a solution actually easier than finding a solution? Intuitively maybe,

No one knows answer, but most people believe it to be true.

Reductions as tool for hardness

We want to say some problems are hard.

We want to say some problems are hard.

Easier 1st step: Problem X is at least as hard as Problem Y.

Proof technique: we reduce Problem Y to Problem X:

Suppose we have black box that solves instances of X.

How can you solve instances of Y using poly steps

+ poly calls to black box.

If Y can be reduced to X, $Y \leq_p X$

\uparrow
Y is poly-time reducible to X

\Rightarrow X is at least as hard as Y because if you can solve X, you can solve Y.

Note: reduce to problem we want to show is harder

Examples:

Max Bipartite Matching \leq_p Max Network Flow

Image Segmentation \leq_p Min-Cut

Max Network Flow \leq_p Linear Programming

Circulation w/ Demands + Lower Bounds \leq_p Circulation w/ Demands \leq_p Max Flow

If ILP can be reduced to TSP, $ILP \leq_p TSP$

\Rightarrow if we can solve TSP, we can solve ILP.

If $Y \leq_p X$ can be reduced to X , then $Y \in P$.

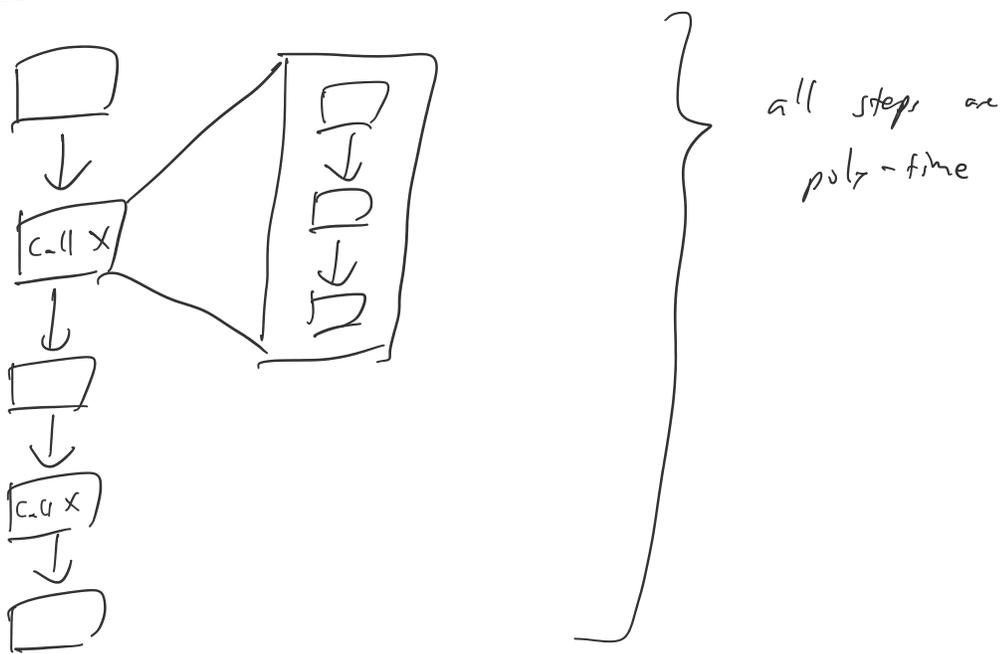
\Rightarrow if we can solve TSP, we can solve ILP.

Suppose $Y \leq_p X$ and \exists poly-time alg for X .

Then \exists poly-time alg for Y .

Why? Polynomials compose

Alg for Y



Thm: If $Y \leq_p X$, and Y CANNOT be solved in poly-time, then X cannot be solved in poly-time.

proof: Suppose we have alg. A that solves X in poly-time. Then we reduce Y to X , giving a solution to Y in poly time, a contradiction.

So if we can find a hard problem Y , can show

Recall: An independent set in a graph $G=(V,E)$ is a set $S \subseteq V$ s.t. $\forall x,y \in S, (x,y) \notin E$.

Thm If $G=(V,E)$ is a graph, then S being an ind. set $\Leftrightarrow V-S$ is a vertex cover.

proof. (\Rightarrow) Suppose S is an independent set, and let $e=(u,v) \in E$.

Only one of u,v can be in S .

\Rightarrow At least one of u,v is in $V-S$.

$\Rightarrow V-S$ is a vertex cover.

(\Leftarrow) Suppose $V-S$ is a vertex cover, and let $u,v \in S$.

$\Rightarrow (u,v) \notin E$, as that edge isn't covered by $V-S$.

$\Rightarrow S$ is an ind. set. □

So vertex covers and independent sets are complements.

Given an instance of Independent Set $\langle G, k \rangle$,

we ask our Vertex Cover black box if \exists a vertex cover C of size $\leq |V| - k$.

If so, we define $S = V - C$ as our sol to Independent Set, as S is an ind. set of size k .

If not, then no vertex cover of size $\leq |V| - k$, so no independent set of size $\geq k$.

\Rightarrow Ind Set \leq_p Vertex Cover.

Claim: Vertex Cover \leq_p Ind Set.

... of Γ . Decide if G has a vertex cover of size k_2

Claim: \forall

proof. To decide if G has a vertex cover of size k , we ask if it has an ind. set of size $|V| - k$. □

\Rightarrow Vertex Cover & Ind Set are equivalently difficult.

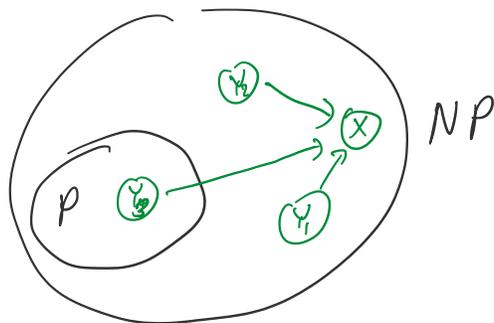
Define:

X is NP-complete, if

• $X \in NP$

• X is NP-hard.

} X can be used to solve every other problem in NP.



Thm. If X is NP-complete, then X is solvable in poly time iff $P = NP$.

proof. If $P = NP$, then $X \in NP = P$, so X is solvable in poly-time.

Suppose $X \in P$. Then $\forall Y \in NP$,

$Y \leq_p X$ so we can solve Y by reducing it in polytime to X , which we can also solve in poly-time, so we can solve Y in poly-time.

$\Rightarrow Y \in P$, so $P = NP$.

Thm If Y is NP-complete, $X \in NP$, and $Y \leq_p X$, then X is NP-complete.

proof. Let $Z \in NP$. Since Y is NP-complete, $Z \leq_p Y \leq_p X$ 

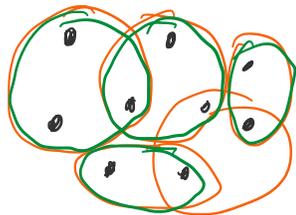
Fact: Vertex Cover is NP-complete.

(easy to verify Vertex Cover is in NP because checking a proposed vertex cover is easy)

\Rightarrow Ind Set is NP-complete.

Problem (Set Cover) Given a set U of elements, and a collection S_1, \dots, S_m of subsets of U , is there a collection of at most k of these subsets whose union is U ?

Set cover



subsets

Lemma: Set Cover $\in NP$.

proof. The certificate is a list of k sets from S_1, \dots, S_m .

We can take the union of these sets and check that the union is equal to U in poly-time 

Lemma: Vertex Cover \leq_p Set Cover.

proof: Let $G = (V, E)$ and k be an instance of Vertex Cover.

Create an instance of Set Cover:

- $U = E$
- Create a $S_u \forall u \in V$ where S_u contains the edges adjacent to u .

U can be covered by $\leq k$ sets iff G has a vertex cover of size $\leq k$.

Why? If k sets S_{u_1}, \dots, S_{u_k} cover U , then every edge is adj to at least one of vertices u_1, \dots, u_k , giving a vertex cover.

Conversely, if u_1, \dots, u_k is a vertex cover, then sets S_{u_1}, \dots, S_{u_k} cover U . 

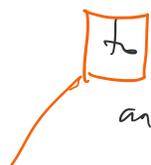
\Rightarrow Set Cover is NP-hard

\Rightarrow Set Cover is NP-complete.

Punchline: If you can reduce

Vertex Cover
Ind. Set
or Set Cover

} known NP-complete

 some problem X , then X is NP-hard, and if additionally, $X \in NP$, then X is NP-complete.

and if additionally, $X \in NP$, then X is NP-complete.
careful about direction of reduction