NP-Hard and NP-Complete Problems

- NP-Hard and NP-Complete Problems
- A first NP-Complete Problem: circuit-sat(C)
- The Cook-Levin Theorem: sat is NP-Complete
- NP-Complete Problems from known Reductions
- NP-Complete ness of dir-ham-cycle and ham-cycle
- tsp is NP-Complete
- subset-sum is NP-Complete
- partition is NP-Complete

Imdad ullah Khan

A problem X is NP-Hard, if every problem in NP is polynomial time reducible to X

$$\forall\, Y \in \mathrm{NP}, \quad Y \ \leq_p \ X$$

A problem $X \in \mathrm{NP}$ is NP-Complete, if every problem in NP is polynomial time reducible to X

$$X \in \mathrm{NP} \quad \text{AND} \quad \forall\, Y \in \mathrm{NP}, \quad Y \ \leq_p \ X$$

A problem $X \in \mathrm{NP}$ is NP-Complete, if every problem in NP is polynomial time reducible to X

$$X \in \mathrm{NP} \quad \text{AND} \quad \forall\, Y \in \mathrm{NP}, \quad Y \leq_p X$$

These problems are at least as hard as any problem in NP

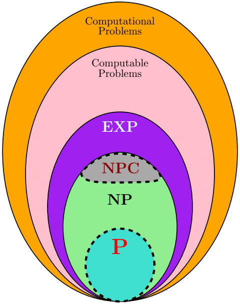
Let NPC be the (sub)class of NP-Complete problems

▷ It is the set of hardest problems in NP

If any NP-complete problem can be solved in poly time, then all problems in NP can be, and thus $\mathrm{P} = \mathrm{NP}$

A problem X is **NP-Complete**, if

1. $X \in \mathrm{NP}$
2. $\forall\ Y \in \mathrm{NP}\ Y \leq_p X$



$\mathrm{P} \subseteq \mathrm{NP} \qquad \mathrm{NPC} \subseteq \mathrm{NP}$

- Take any $X \in \mathrm{NP}$ and prove that it cannot be solved in poly time
  - You proved $\mathrm{P} \neq \mathrm{NP}$ Why?
  - By definition of $\subset$
- Take any $X \in \mathrm{NPC}$ and solve it in poly time
  - You proved $\mathrm{P} = \mathrm{NP}$ Why?
  - By definition of $\leq_p$

NP-COMPLETE Problems

A problem X is NP-COMPLETE, if

1. $X \in \mathrm{NP}$
2. $\forall\ Y \in \mathrm{NP}\ Y \leq_p X$

No polynomial time algorithm for any NP-COMPLETE problem yet

▷ People did and do try, as many practical problems are in NPC

No impossibility proof of poly-time solution for a NP-COMPLETE problem

▷ People did and do try, will prove the widely held belief that $\mathrm{P} \neq \mathrm{NP}$

Let X be any NP-COMPLETE problem.

X is polynomial time solvable if and only if $\mathrm{P} = \mathrm{NP}$

NP-COMPLETE Problems

A problem X is NP-COMPLETE, if

1. $X \in \mathrm{NP}$
2. $\forall\ Y \in \mathrm{NP}\ Y \leq_p X$

Why should you prove a problem to be NP-COMPLETE?

- Good evidence that it is hard
- Unless your interest is proving $\mathrm{P} = \mathrm{NP}$ stop trying finding efficient algorithm

▷ Instead focus on special cases, heuristic, approximation algorithm

What to tell your boss if you fail to find a fast algorithm for a problem?

1. I am too dumb! ▷ You are fired
2. There is no fast algorithm! *You claim that* $\mathrm{P} \neq \mathrm{NP}$ ▷ Need a proof
3. I cannot solve it, but neither can anyone in the world! ▷ Need reduction

NP-Complete Problems

Dealing with Hard Problems

- What to do when we find a problem that looks hard…

I couldn't find a polynomial-time algorithm;
I guess I'm too dumb.

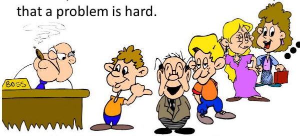
Dealing with Hard Problems

- Sometimes we can prove a strong lower bound… (but not usually)

I couldn't find a polynomial-time algorithm,
because no such algorithm exists!

Dealing with Hard Problems

- NP-completeness let's us show collectively that a problem is hard.



I couldn't find a polynomial-time algorithm,
but neither could all these other smart people.

source: slideplayer.com via Google images

NP-Complete Problems

A problem X is NP-Complete, if

1. $X \in \mathrm{NP}$
2. $\forall\ Y \in \mathrm{NP}\ Y \leq_p X$

NP-Complete problems capture the essential difficulty of all NP problems

Could there be any NP-Complete problem at all?

- Not very hard to imagine (an almost formal proof later)
- Let A be a polynomial time algorithm working on bit-strings that outputs **Yes/No** based on some unknown but consistent logic
- H is the problem: "Is there any polynomial sized bit-string on which A outputs **Yes**?" Clearly $\mathrm{H} \in \mathrm{NP}$?
- Any problem $Y \in \mathrm{NP}$ is reducible to H
- $Y \in \mathrm{NP}$ means there is a poly-sized certificate that can be verified. An instance $\mathcal{I}$ of Y can be transformed to an instance of H with same answer

How to prove NP-Completeness

A problem X is NP-Complete, if

1. $X \in \mathrm{NP}$
2. $\forall\ Y \in \mathrm{NP}\ Y \leq_p X$

How to prove a problem NP-Complete ?

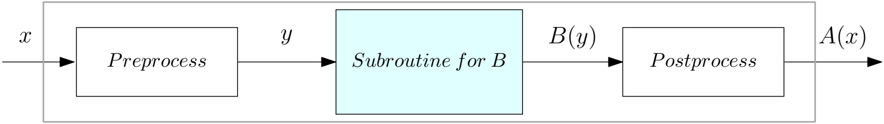
- Proving NP is relatively easy (in many cases)
- Can we do so many reductions?

Polynomial Time Reduction: Algorithm Design Paradigm

Problem A is polynomial time reducible to Problem B, $A \leq_p B$

If any instance of problem A can be solved using a polynomial amount of computation plus a polynomial number of calls to a solution of problem B

Subroutine for B takes an instance y of B and returns the solution $B(y)$



Algorithm for A transforms an instance x of A to an instance y of B. Then transforms $B(y)$ to $A(x)$

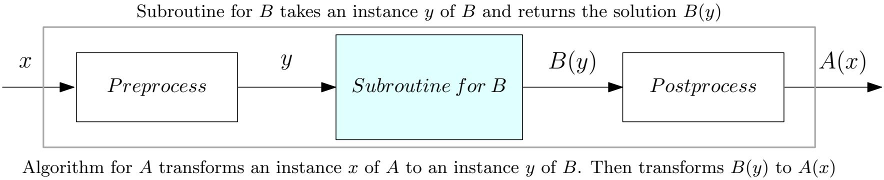
Suppose $A \leq_p B$.

If B is polynomial time solvable, then A can be solved in polynomial time

Polynomial Time Reduction: Tool to Prove Hardness

Problem A is polynomial time reducible to Problem B, $A \leq_p B$

If any instance of problem A can be solved using a polynomial amount of computation plus a polynomial number of calls to a solution of problem B



Suppose $A \leq_p B$.

If A is NP-Complete, then B is NP-Complete

▷ **Why?**

Proving NP-Complete Problems

A problem X is NP-Complete, if

1. $X \in \mathrm{NP}$
2. $\forall\ Y \in \mathrm{NP}\ Y \leq_p X$

To prove X NP-Complete, reduce an NP-Complete problem Z to X

If Z is NP-Complete, and

1. $X \in \mathrm{NP}$
2. $Z \leq_p X$

then X is NP-Complete

1. $X \in \mathrm{NP}$ is explicitly proved
2. $\forall\ Y \in \mathrm{NP},\quad Y \leq_p X$ follows by transitivity

   $\forall\ Y \in \mathrm{NP},\quad Y \leq_p Z$ is true as Z is NP-Complete

   $[Y \leq_p Z \ \wedge \ Z \leq_p X] \implies Y \leq_p X$

A problem X is NP-COMPLETE, if

1. $X \in \mathrm{NP}$
2. $\forall\ Y \in \mathrm{NP}\ Y \leq_p X$

How to prove a problem NP-COMPLETE?

- Proving NP is relatively easy
- Can we do so many reductions?

Template of proving problems to be NP-COMPLETE

We proved that	CLIQUE(G, k) is NP-COMPLETE
Suppose we have the theorem	CLIQUE$(G, k) \leq_p$ IND-SET(G, k)
Then we can conclude that	IND-SET(G, k) is NP-COMPLETE