Classes of Problems

- Polynomial Time Verification
- The Classes P and NP
- The Classes EXP and coNP
- NP-Hard and NP-Complete Problems
- Proving NP-Hardness
- A first NP-Complete Problem

Imdad ullah Khan

NP-HARD and NP-COMPLETE

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

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

NP-Hard and NP-Complete

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

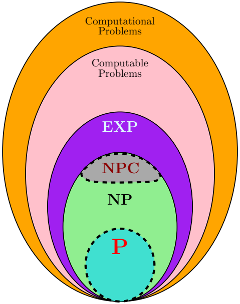
▷ It is the set of hardest problems in NP

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

NP-Complete Problems

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

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



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

- 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?

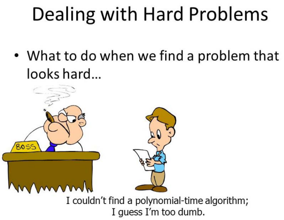
- Good evidence that it is hard
- Unless your interest is proving $\mathrm{P} = \mathrm{NP}$ stop trying finding efficient algorithm

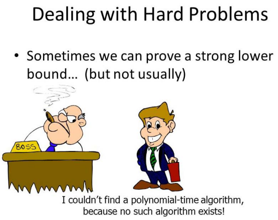
▷ Instead focus on special cases, heuristic, approximation algorithm

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





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?