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

The Class P of Problems

The Class P: Decision problems that can be **solved** in polynomial time

▷ There exists an algorithm that correctly outputs **Yes/No** on any instance

Recall that polynomial time is a good notion of "reasonable/efficient time"

- Mainly because polynomials are closed under composition (reduction)
- In practice degrees of polynomials are small

(Appropriately defined decision versions of) all these problems are in P

- MST(G, k)
- SHORTEST-PATH(G, s, t, k)
- PRIME(n)
- BIPARTITE-VERTEX-COVER(G, k)
- MAX-FLOW(G, t)

The Class NP of Problems

The Class NP: Decision problems that can be **verified** in polynomial time

A problem X is efficiently verifiable if

- The claim: "$\mathcal{I}$ is a **Yes** instance of X" can be made in polynomial bits
  - There exists a polynomial sized certificate for **Yes** instances of X
- A certificate can be verified in polynomial time
  - There exists a polynomial time algorithm $\mathcal{V}$ that takes the instance $\mathcal{I}$ and the certificate $\mathcal{C}$ such that $\mathcal{V}(\mathcal{I},\mathcal{C}) =$ **Yes** iff $X(\mathcal{I}) =$ **Yes**

▷ NP stands for "Non-deterministic Polynomial Time"

- 3-SAT(f)
- HAMILTONIAN-CYCLE(G)
- KNAPSACK(U, w, v, C)
- INDEPENDENT-SET(G, k)

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

Let $X \in \mathrm{P}$, we show that $X \in \mathrm{NP}$

By definition, there exists a polynomial time algorithm $\mathcal{A}$, which decides X

We argue existence of a poly-sized certificate for **Yes** instances of X and poly-time verifier for X

- The certificate could be an empty string
- Given an instance $\mathcal{I}$ of X and a certificate $\mathcal{C}$ to witness that $X(\mathcal{I}) =$ **Yes**
- $\mathcal{V}$ **can be** $\mathcal{V}(\mathcal{I}, \mathcal{C}) := \mathcal{A}(\mathcal{I})$ ▷ polynomial time
- Essentially ignore the certificate, decide the instance using $\mathcal{A}$ if the output is **Yes** declare verified else not verified

Notice that the output of this $\mathcal{V}$ is $\mathcal{V}(\mathcal{I}, C) =$ **Yes** iff $\mathcal{A}(\mathcal{I}) =$ **Yes**

P = NP?

The following problems we know or can be easily shown to be in P and NP.

Notice the corresponding problems are of similar flavor to each other

P	NP
2-SAT	3-SAT
EULER-TOUR	HAMILTONIAN-CYCLE
MST	TSP
SHORTEST-PATH	LONGEST-PATH
INDEPENDENT-SET-TREE	INDEPENDENT-SET
BIPARTITE-MATCHING	3D-MATCHING
BIPARTITE-VERTEX-COVER	VERTEX-COVER
LINEAR PROGRAM	INTEGER LINEAR PROGRAM
PRIME	FACTOR

$\mathrm{P} = \mathrm{NP}$?

Many problems in CS, Math, OR, Engineering, etc. are polynomial time verifiable but have no known polynomial time algorithm

Polynomial time verifiability **seems like** a weaker condition than polynomial time solvability

- No proof that it is weaker (i.e. NP describes a larger class of problems)

So it is unknown whether P = NP

$\mathrm{P} = \mathrm{NP}$?

Is $\mathrm{P} = \mathrm{NP}$?

The biggest open problem in computer science

Is verifying a candidate solution is easier than solving a problem?

- Majority believes that $\mathrm{P} \neq \mathrm{NP}$
- One can check if any of possible candidate solutions verifies
- But candidate space can be exponential
  - $n!$ possible Hamiltonian cycles are candidates for $\textsc{tsp}(G, k)$
  - $\binom{n}{k} = O(n^k)$ possible subsets for $\textsc{clique}(G, k)$
- **No known " better way" than this**
- **No proof that there is no better way than this**

$\mathrm{P} = \mathrm{NP}$?

> To say that "P vs NP is the central unsolved problem in computer science" is a comical understatement. P vs NP is one of the deepest questions that human beings have ever asked.
>
> Scott Aaronson

- There is a reason it is one of 7 million-dollar prize problem of the Clay Mathematical Institute (now one of the 6)
- If $\mathrm{P} = \mathrm{NP}$, then mathematical creativity can be automated (the ability to verify a proof would be the same as the ability to find a proof)
- Since verification seems to be way easier, every verifier would have the reasoning power of Gauss and the like
- By just programming your computer in polynomial time you can solve (perhaps) the other 5 Clay Institute problems
- "just because I can appreciate good music, doesn't mean that I would be able to create good music"

$\mathrm{P} = \mathrm{NP}$?

Then why isn't it obvious that $\mathrm{P} \neq \mathrm{NP}$

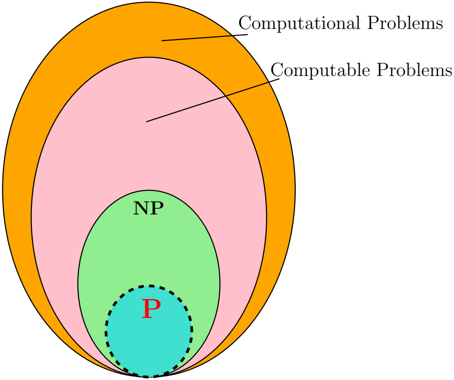
- Intuition tells us that brute-force search is unavoidable
- It is generally believed that there is no general and significantly better than brute-force method to solve NP problems
- Why can't we prove it?
- It is said that the great physicist Richard Feynman had trouble even being convinced that P vs NP was an open problem
- There are many many problems where we could avoid brute-force search

  ▷ See the list of "hard" problems and their easier "counterparts"

- Though not a decision problem, recall that we discussed that (to impress your boss) you can say that your algorithm for SORTING finds that one unique permutation out of the $n!$ possible ones

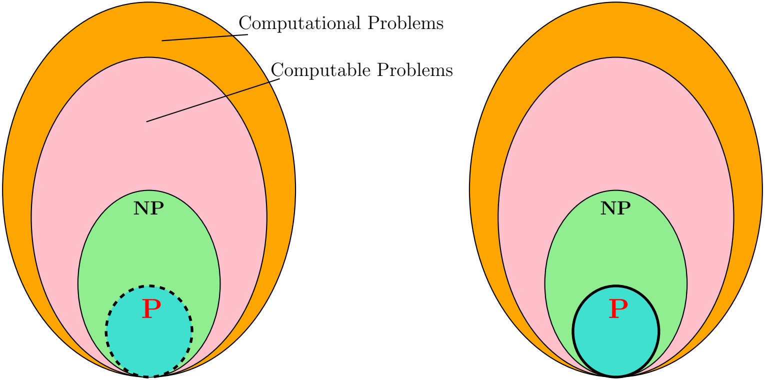
We try to characterize these hard problems and say that almost all of them all essentially the same

$\mathrm{P} = \mathrm{NP}$?



- For $X \in \mathrm{NP}$ prove that there is no polynomial time algorithm
- You proved $\mathrm{P} \neq \mathrm{NP}$ (You get a million dollars and *A* in this course)

$\mathrm{P} = \mathrm{NP}$?



- For $X \in \mathrm{NP}$ prove that there is no polynomial time algorithm
- You proved $\mathrm{P} \neq \mathrm{NP}$ (You get a million dollars and *A* in this course)