innovate achieve lead

BITS Pilani
Pilani Campus

CS/IS F214
Logic in Computer Science

MODULE: INTRODUCTION

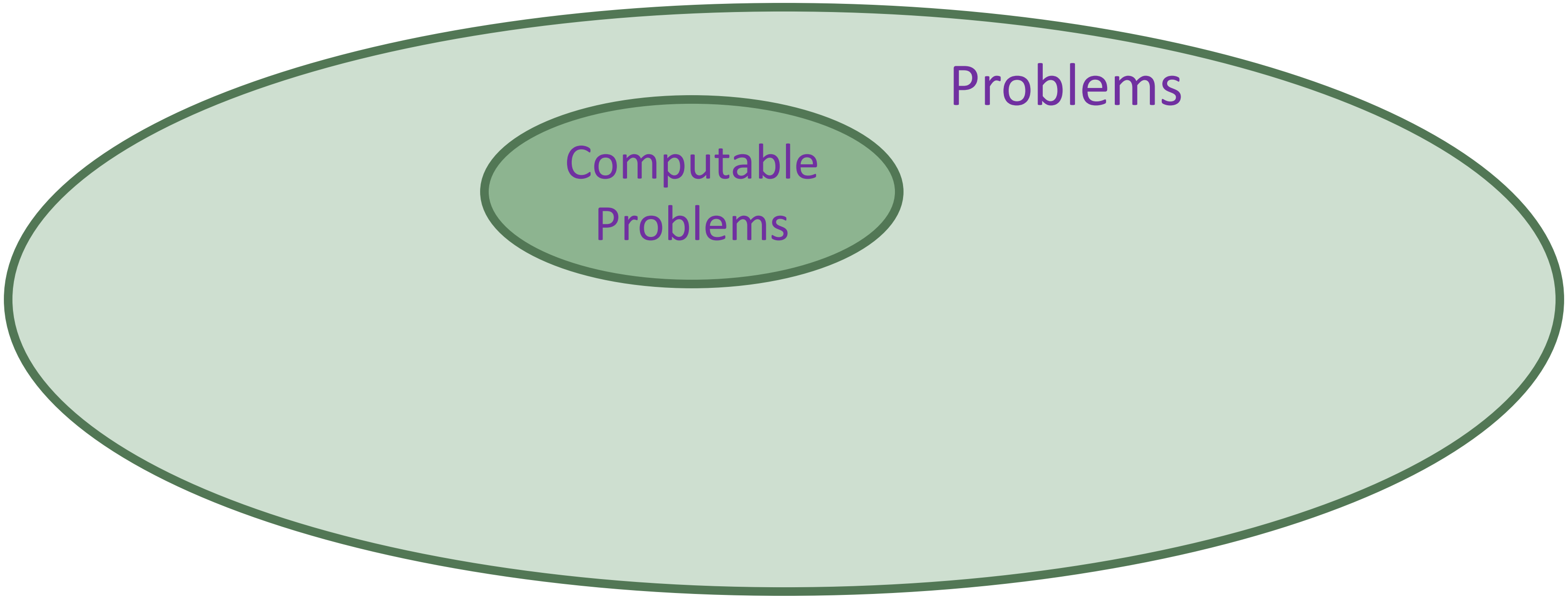
Complexity Classes – P and NP

Time Complexity

- We talk about the ***time complexity*** of an algorithm (or a program) as
  - the time taken to run the algorithm on its ***worst case*** input, as a *function of the size of the input*
    - alternatively, the maximum time taken by the algorithm on any input, as a function of the size of the input
- One way of strictly defining the complexity of an algorithm is to
  - count the number of steps the Turing machine (designed for that algorithm)

RECALL: Computability

- Questions:
  - Does a Turing machine exist for every algorithm (or program)?
  - Is there a Turing machine to solve any problem?
- Recall:
  - **Computable** is same as **Turing-Computable,** which is same as *computable using a general purpose computer*.



Complexity Classes

- Define
  - **TIME(f(n))** =

    { π | π is a problem that can be solved in time at most **f(n)** where **n** is the input size }
- Note:
  - "π can be solved" means "that a solution for π can be computed for a given input"
    - i.e. there exists an algorithm to solve π
    - i.e. there exists a Turing machine for π that takes a given input and produces the solution as output
- Define
  - **P = TIME(f(n))** where **f** is a polynomial function in **n**.

Problems and Solutions

- Intuitively:
  - *solving a problem is harder than verifying a given solution.*
- e.g.
  - Consider the problem of computing factors of a given integer:
    - i.e. given **N**, find its factors, say **x** and **y** such **x * y = N**.
  - Alternatively, consider the problem of verifying whether given numbers are factors of another number:
    - i.e. given **N**, **x**, and **y**, verify whether **x * y = N**

Complexity Classes

- Define
  - **NTIME(f(n))** =

{ π | π is a problem that can be verified in time at most **f(n)** where **n** is the input size }

- Note:
  - *π can be verified means* that
    - given an input **x** and a (purported) solution $\mathbf{S_{\pi}(x)}$
    - *it can be decided* (i.e. *computed*)
      - whether indeed $\mathbf{S_{\pi}(x)}$ is a solution for π on **x**
- Define
  - **NP = NTIME(f(n))** where **f** is a polynomial function in **n**.

P ?= NP

- **P $\subseteq$ NP** ?
  - Why?
    - If a problem π is in P,
      - then it can be solved in polynomial time and the length of the solution cannot be more than polynomial time (why?)
      - i.e. a certificate (i.e. as evidence for the solution) exists such that
        - it can be verified in polynomial time.
      - i.e. π is in P

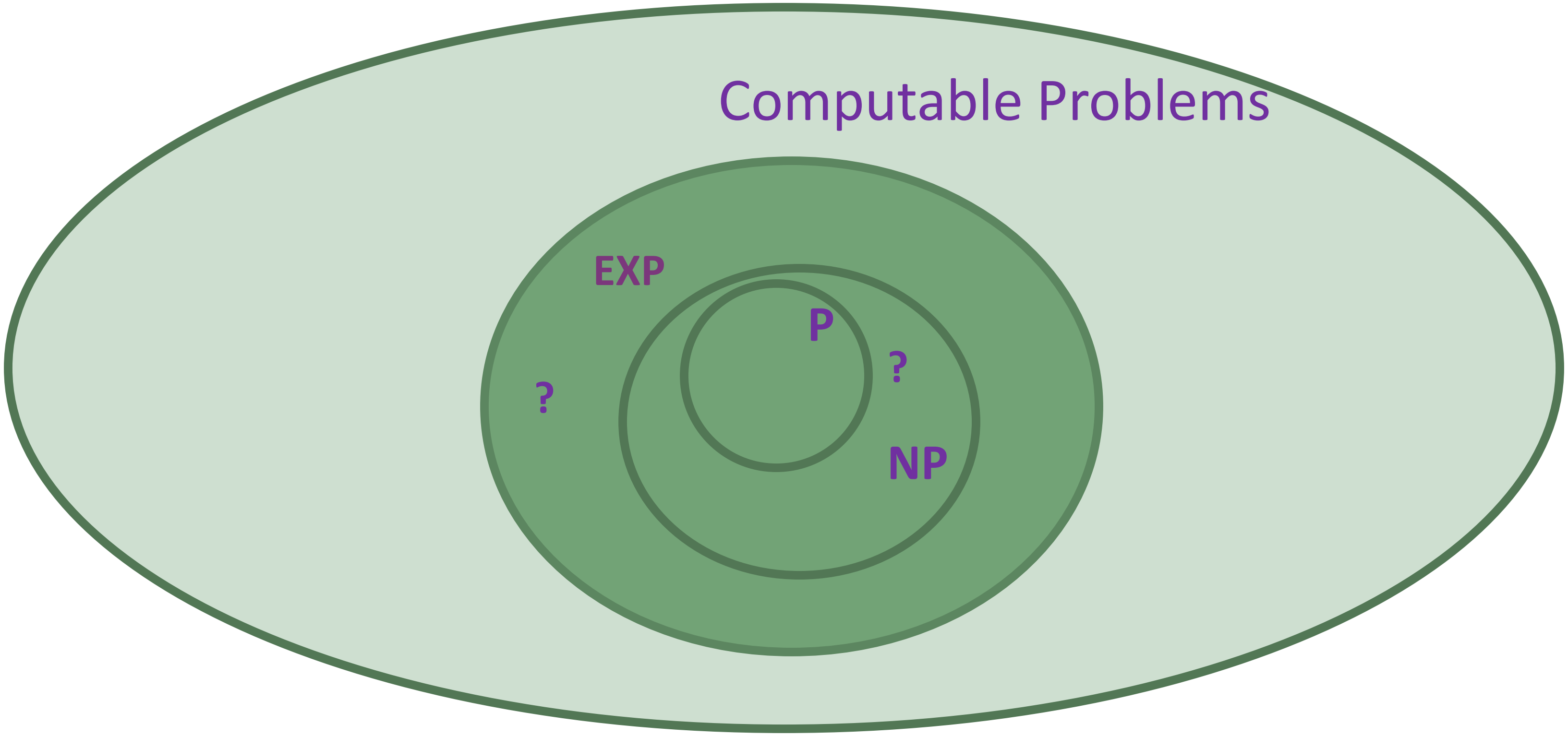
P ?= NP

- Is **P = NP** ?
  - An open question:
    - There is no known problem proven to be:
      - in **NP \ P**
    - There are plenty of problems
      - proven to be in **NP** but for which
      - there are no known polynomial time algorithms.

NP ?= EXP

- Define **EXP** =
  - {π | π is a problem that can be solved in exponential time i.e. *in time proportional to* 2^n *for input size* n }
- Is **NP** $\subseteq$ **EXP** ?
  - If a problem π is in **NP**,
    - then a certificate (i.e. as evidence for the solution) exists such that
      - it can be verified in polynomial time, say g(n)
    - i.e. a certificate of length f(n) exists where f is polynomial in input size n
  - Construct all potential certificates (i.e. all possible bit strings) of length at most f(n) for a given input of size n:
    - each potential certificate can be verified in polynomial time
      - i.e. in time $2^{f(n)} * g(n)$ one can find the correct certificate (corresponding to the solution)
    - i.e. π is in **EXP**

Computability and Complexity



- We know **P** $\subseteq$ **NP** $\subseteq$ **EXP**
- We know **P** $\subset$ **EXP** i.e. *there are problems that can be solved in exponential time but not in polynomial time (e.g. board-games like chess)*
- From this we can infer **P** $\subset$ **NP** or **NP** $\subset$ **EXP** but we do not have a proof of either!

- Although these ideas were discussed as early as 1950s by Turing, Godel, and von Neumann:
  - these complexity classes and the question (Is **P = NP** ?) were formulated in the early 70s.