Randomized Methods in Computational Complexity

- This course will study computational problems & their complexity.

- Probabilistic methods, or random objects, will be key.

- Course covers the following topics:
  - Algorithms, upper/lower bounds, circuits.
  - Expanders – optimally connected graphs.

- Pseudorandom generators (prg) — boolean functions that "look" random. (hard-to-predict)
- Error-correcting Codes — boolean/algebraic functions that "spread" the essence of a string.
- Extractors — boolean functions to "extract" randomness from an "impure" source.

- Textbook: Arora & Barak "Complexity theory: A Modern Approach".

- <u>Grading Policy:</u>
  - 25% - Assignments
  - 30% - Mid Sem exam
  - 40% - 2nd Sem exam
  - 5% + bonus - participation / extra talk.

- Go to ..../nitin/teaching.html [CS747]
  - Also for other courses, eg. CS640.

Formalize problems & difficulty

- Problem: Set L of strings in $\{0,1\}^*$.
  Decision Problem / Language
- Solution: We call a problem computable if there is a Turing Machine (TM) solving it.
- TM is an abstraction of real computers, or any computing device known to us.

- Computable problems are also called decidable or recursive. ($\exists$ an algorithm)

- Famous Problems:

1) Given a quadratic $f(x_1, \dots, x_n) \in \mathbb{Z}[\bar{x}]$, find an integral root.

   Eg. $f = x_1 - x_2^2$ has a root $(4, -2)$.

2) Given several quadratics $f_1, \dots, f_m \in \mathbb{Z}[x_1, \dots, x_n]$, find an integral zero of $f_1 = f_2 = \dots = f_m = 0$.

   [Hilbert's 10th problem]

▷ (1) is computable in exponential-time!
(2) is uncomputable!

- For a computable problem L, the step-by-step procedure of its TM is an algorithm.

- #Steps is called time-complexity.
- #Cell " " space-complexity.
- Decision problem $L \subseteq \{0,1\}^*$ [boolean problem]
- Functional problem $f : \{0,1\}^* \to \{0,1\}^*$.

- A complexity class is a collection of problems.

Egs. of complexity class

- Let $T: \mathbb{N} \to \mathbb{N}$ be a function & n be the input bit-size $|x|$.

- $\text{Dtime}(T(n)) := \{ L \subseteq \{0,1\}^* \mid \exists \text{ TM that tests } x \in L \text{ in } O(T(|x|))\text{-time} \}$.

- $P := \bigcup_{c>0} \text{Dtime}(n^c)$ [polynomial time]

- $E := \bigcup_{c>0} \text{Dtime}(2^{cn})$ [simple-exponential-time]

- $EXP := \bigcup_{c>0} \text{Dtime}(2^{n^c})$ [exponential-time]

- $SUBEXP := \bigcap_{c>0} \text{Dtime}(2^{n^c})$ [sub-exp-time]

  eg. $\text{Dtime}(2^{\log^2 n})$

- Similarly, $\text{Ntime}(T(n))$ for nondeterministic TM based algorithms.

  (They can guess bits to solve a problem!)

- Define NP, NE, NEXP, SUBNEXP.

- Similarly, we can define Space(T(n)) based on space required by TM.

- Define $\mathbb{L}$ (logspace), Pspace, Expspace.

- We can also use probabilistic TM, where TM steps may use "random coin-flips".
The final answer is required to be correct with decent probability.
practical algorithms?

- Bounded-error probabilistic polynomial-time
  BPP := $\{ L \subseteq \{0,1\}^* \mid \exists$ poly-time prob. TM solving $L \}$

- A classic example in BPP is:
  Polynomial Identity Testing (PIT).

- PIT: Given an arithmetic circuit $C(x_1, \ldots, x_n)$
  over field $\mathbb{F}$, test if $C = 0$?

  $\mathbb{Q}$ or $\mathbb{F}_q$ (finite)