

CS747: Randomized Methods in Computational Complexity

- This course will study some computational problems & their complexity.
- Probabilistic methods, or random objects, will be key.
- The course covers the following topics:
 - Circuits, lower bounds, algorithms.
 - Expanders - optimally connected graphs.
 - Pseudorandom generators (prg) - "look" random.
 - Error-correcting codes - "spread" the essence of a string around.
 - Extractors - "extract" randomness.

- Probabilistically checkable proofs (PCP), if time permits.
 - Recommended textbook is: Arora & Barak, "Complexity theory: A Modern Approach".
 - Grading:
 - 25% - Assignments
 - 30% - Mid-Sem takehome
 - 40% - End-Sem exam
 - 5% + bonus - participation or extra talk.
 - Go to .../nitin/teaching.html
 - Also, for the lecture notes of CS640.
-

Formalizing problems & difficulty

- We call a problem computable if there is a Turing machine (TM) solving it.

- TM is an abstraction of a Computer, or any computing device known.

- Computable problems are also called decidable / recursive.

- Famous problems:

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

Hilbert's
10th problem

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

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

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

• The #steps is called the time complexity of L .

• The space used by the TM is the space complexity of L .

decision

- A problem L we always formalize as a subset of strings $\{0,1\}^*$, called a language.

Eg. $\text{PRIMES} = \{n \in \mathbb{N} \mid n \text{ is prime}\}$.

- A functional problem is formalized as $f: \{0,1\}^* \rightarrow \{0,1\}^*$.

Eg. $+: \{0,1\}^* \rightarrow \{0,1\}^*$
 $(m,n) \mapsto m+n$

- A complexity class is a collection of problems.

Ex. of complexity classes

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

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

- poly-time P $:= \bigcup_{c > 0} \text{Dtime}(n^c).$

- simply-exp-time E $:= \bigcup_{c > 0} \text{Dtime}(2^{cn}).$

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

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

- Similarly, Ntime $(T(n))$ for nondeterministic TM based algorithms.

(They can guess bits to solve a problem!)

- This defines NP, NE, NEXP, SUBNEXP.

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

• Defines ^{logspace} L, Pspace, Expspace.

- This course will focus on randomized methods, i.e. algorithms are allowed to "flip coins" and proceed accordingly. The final answer is required to be correct with a decent probability.

- Such problems comprise BPP := $\{L \subseteq \{0,1\}^* \mid \exists \text{ poly-time randomized TM } M \text{ solving } L\}$.

- A classic example in BPP is:
Polynomial identity testing (PIT).