Theory of Computation

Introduction

From high level

- From a high level what this course is about?
  - Given a set, say S.
  - Let $A \subseteq S$.
  - Both S and A are well defined.
  - We are given an element $x \in S$, and asked to find whether this x is in A or not.
  - That's all !

Surprises !!

- Surprising things
  - This is related to decision problems.
    - S is set all face images. A is set of images of a particular person. {Face verification}
    - S is set of all graphs. A is set of graphs with Hamiltonian cycle.
  - Sometimes this is an **unsolvable** problem. ??
  - Sometimes this is an **easy** task, sometimes quite a **difficult** one.
    - Recall, $O(n^2)$ is time consuming than $O(n)$ algorithm.

We want general enough set

- Set of strings over some alphabet like {0,1}.
- For example set of strings that end with a 0,
  {0, 00, 10, 000, 010, 100, 110, …}
- Eg2: Each string in the set can be seen as a positive binary number and let the set be the set of prime numbers.
  - Given a number (binary string of 0s and 1s) you want to find whether this is in the set (prime) or not (not a prime).

Why strings are chosen?

- Any data element like number or image or any thing can be represented as a string.
  - Can we say DNA code represents a human being?
- Even a method which solves a problem can be represented as a string.
- A proof can be represented as a string.
- So strings over an alphabet gives us power to represent the things… that is we have ***languages* of strings to represent the things**.

What will you learn from this course?

- How to define a computer? **Automata theory**
- Are there problems that a computer cannot solve? If so, can we find one such problem? **Computability theory**
- For problems that a computer can solve, some problems are easy (e.g., sorting) and some are difficult (e.g., time-table scheduling). Any systematic way to classify problems? **Complexity theory**

Syllabus

- **Syllabus:**
- **Unit – 1** [8 Hours]: Introduction - Alphabets, Strings and Languages, Automata and Grammars; Deterministic finite Automata (DFA) - Formal Definition, Simplified notation, State transition graph, Transition table, Language of DFA; Nondeterministic finite Automata (NFA) - NFA with epsilon transition, Language of NFA, Equivalence of NFA and DFA, Minimization of Finite Automata, Distinguishing one string from other
- **Unit – 2** [8 Hours]: Regular Expression (RE) - Definition, Operators of regular expression and their precedence, Algebraic laws for Regular expressions; Relation with FA - Regular expression to FA, DFA to Regular expression; Non Regular Languages - Pumping Lemma for regular Languages, Application of Pumping Lemma; Properties - Closure properties of Regular Languages, Decision properties of Regular Languages, Applications and Limitation of FA
- **Unit – 3** [8 Hours]: Context Free Grammar (CFG) - Definition, Examples, Derivation, Derivation trees; Ambiguity in Grammar - Inherent ambiguity, Ambiguous to Unambiguous CFG; Normal forms for CFGs - Useless symbols, Simplification of CFGs, CNF and GNF; Context Free Languages (CFL) - Closure properties of CFLs, Decision Properties of CFLs, Emptiness, Finiteness and Membership, Pumping lemma for CFLs
- **Unit – 4** [8 Hours]: Push Down Automata (PDA) - Description and definition, Instantaneous Description, Language of PDA; Variations of PDA - Acceptance by Final state, Acceptance by empty stack, Deterministic PDA; Equivalence of PDA and CFG - CFG to PDA and PDA to CFG
- **Unit – 5** [8 Hours]: Turing machines (TM) - Basic model, definition and representation, Instantaneous Description; Variants of Turing Machine - TM as Computer of Integer functions, Universal TM; Church's Thesis; Language acceptance by TM - Recursive and recursively enumerable languages;
- **Unit – 6** [8 Hours]: Decidability - Halting problem, Introduction to Undecidability, Undecidable problems about TMs; Complexity - Time Complexity, Problem classes - P, NP, NP-Hard, NP-Complete.

Text Books

Text Books:

- John E. Hopcroft, Rajeev Motwani and Jeffrey D. Ullman, Introduction to Automata Theory, Languages and Computation, Pearson Education, 3rd edition, 2014, ISBN: 978-0321455369
- Michael Sipser, Introduction to the Theory of Computation, Cengage Learning, 3rd Edition, 2014, ISBN: 978-8131525296

Reference Books:

- John C. Martin, Introduction to Languages and the Theory of Computation, McGraw-Hill Education, 4th edition, 2010, ISBN: 978-0073191461
- Bernard M. Moret, The Theory of Computation, Pearson Education, 2002, ISBN: 978-8131708705

Evaluation

- Quiz: Best (n-1) 20 marks
- Mid1: 20 marks
- Mid2: 25 marks
- Endsem: 35 marks

- Can be updated (and will be informed).