Introduction
Theory of Computation

Course Code: 10B11CI513 **Credit: 4**

L-T-P Scheme: 3-1-0

Course Coordinator:- Dr. Ajay Kumar
Department of CSE

Textbook

- K.L.P.Mishra: Theory of Computer Science 3rd Edition.
- Michael Sipser: *Introduction to the Theory of Computation*, second edition, 2006

Supplementary Readings:

- J. Hopcroft, R. Motwani and J. Ullman, *Introduction to Automata, Languages, and Computation*, Third edition, China Machine Press, 2007
- C. Baier and J. Katoen, *Principles of Model Checking*, 2008, MIT Press.

Prerequisites:

- Discrete Mathematics and
- Basic knowledge of algorithms

Focus

- Automata
- Computability
- Complexity

➢ Each area is interpreted differently and the answer varies according to the interpretation.

Why study Theory of Computing?

- What are the fundamental capabilities and limitations of computers?
  - (*All the previous area can be link to this question*)

Complexity

- What makes some problems computationally hard and others easy?

    - Example : Sorting problem and scheduling problem

- ✓ Scheduling thousand classes make take centuries in order to find best schedule
- ✓ We do not know the exact answer (past 35 year)

Outcome of Complexity Theory

- We can demonstrate that certain problem are computationally hard

If not:

- Which aspect of the problem is at the root of the difficulty(we can alter it)
- Settle for less perfect solution
- Some problem are hard in only in the worst case situation and easy most of the time

(Helps in designing hard problem by cryptographer)

Computability theory

- What can and can not be computed?

Example: The problem of determining whether a mathematical statement is true or false

(Can be solved by mathematician but no computer algorithm can solve it)

Outcome of both studies

- Complexity: Classifying problem into easy and hard
- Computability: Solvable and non-solvable problem

Automata theory

- Definition, properties of mathematical model of computation
- FSM: Text processing, complier design and hardware design
- CFG: Programming language and AI

Outcome

- What is a computer?
  - Formal definition of computation
- Computational models
  - Turing machines (= real computers)
  - Simpler computing devices: finite state automata, push-down automata

What is Theory Good for?

- Elegant way of thinking
- Expanding your minds
- Useful tools in practice
    - finite automata, regular expressions: text processing
    - grammars: programming language design and specification
    - NP-hardness: approximate solutions or randomized computation

A Brief History: Hilbert's Program

- Hilbert's program (1920's)

① Axiomatization of all mathematics

② Completeness: all true mathematical statements can be proved in the formalism

③ Consistency: no contradiction can be obtained in the formalism of mathematics

④ Decidability: there should be an *algorithm* for deciding the truth or falsity of any mathematical statement

"We must know. We will know".

History: Godel’s Incomplete Theorems

Godel’s first incomplete Theorem (1931)

- no consistent system of axioms whose theorems can be listed by an "effective procedure" (essentially, a computer program) is capable of proving all facts about the natural numbers.

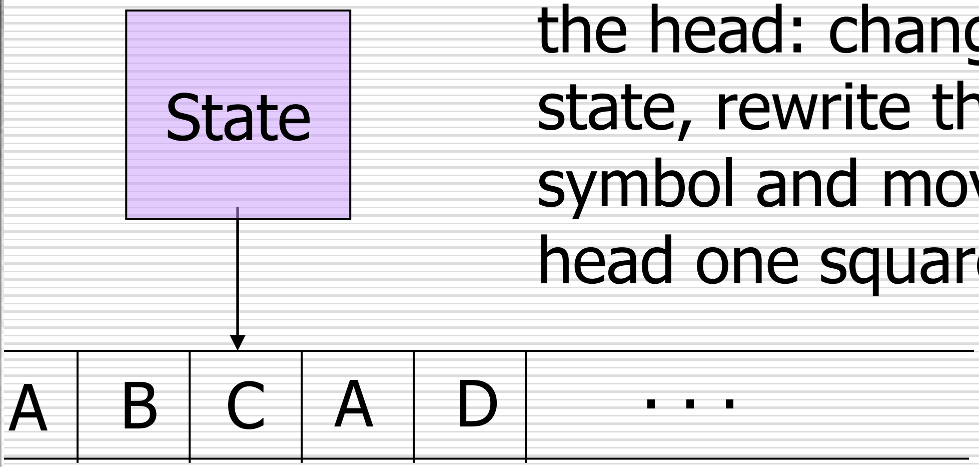
- However, in 1940’s Tarski showed that the first order theory of the real numbers with addition and multiplication is decidable. In this sense, number theory is more difficult than real analysis to computer scientists.

Goedel and Turing (1931-1936)

Goedel (1906-1978)

Turing (1912-1954)

History: Turing Machine (1936)



Action: based on the state and the tape symbol under the head: change state, rewrite the symbol and move the head one square.

Infinite tape with squares containing tape symbols chosen from a finite alphabet

Church and Kleene

(1903-1995)

(1909-1994)

History: Church-Turing Thesis

Intuitive notion of algorithm	=	Turing machine of Algorithm

- An algorithm is an effective method expressed as a finite list of well-defined instructions for calculating a function.

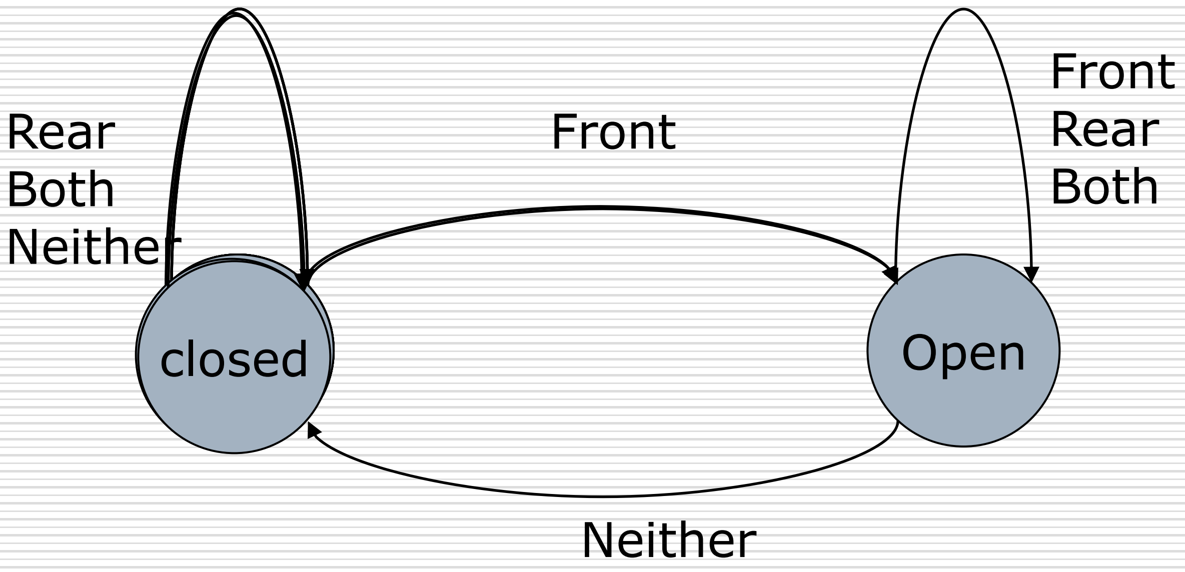
History: Automata

- Scott & Rabin: Finite Automata and their Decision Problems, *1959*
- Kleene: Regular languages

In a nonderministic machine, several choices may exist for the next state at any point.

A Simple Automata

Automatic Door as an automaton



	Neither	Front	Rear	Both
Closed	Closed	Open	Closed	Closed
open	Open	Closed	Open	Open

History: P and NP

Classify into two kinds of problems:

① Those that can be solved efficiently by computers

② Those that can be solved in principles, but in practice take so much time that computers are useless for all but very small instances of the problem.

Why Study Automata Theory

- Software for designing and checking the behavior of digit circuits
- Software for verifying systems of all types that have a finite number of distinct states, such as communication protocols or protocols for secure exchange of information.

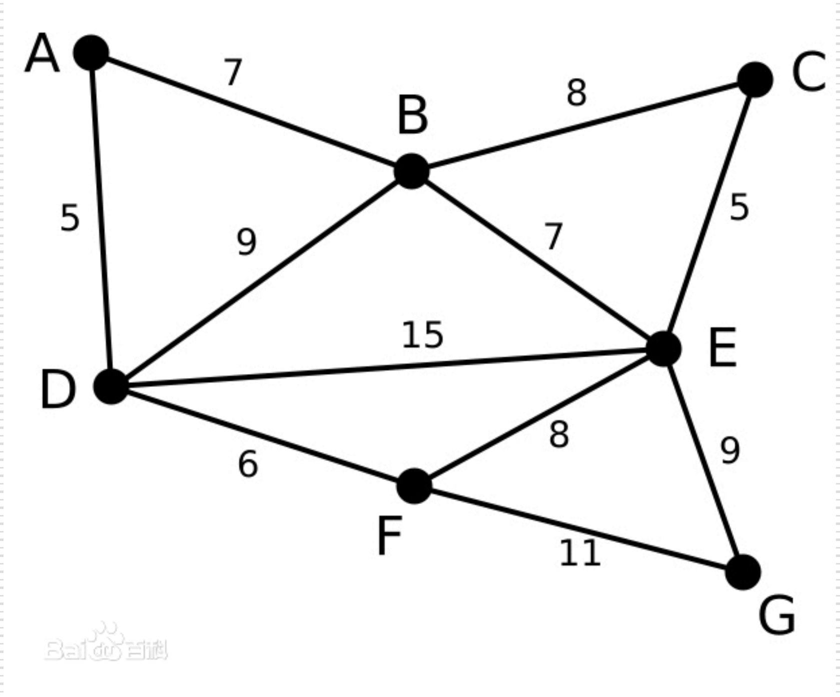
Why Study Computability Theory (1)

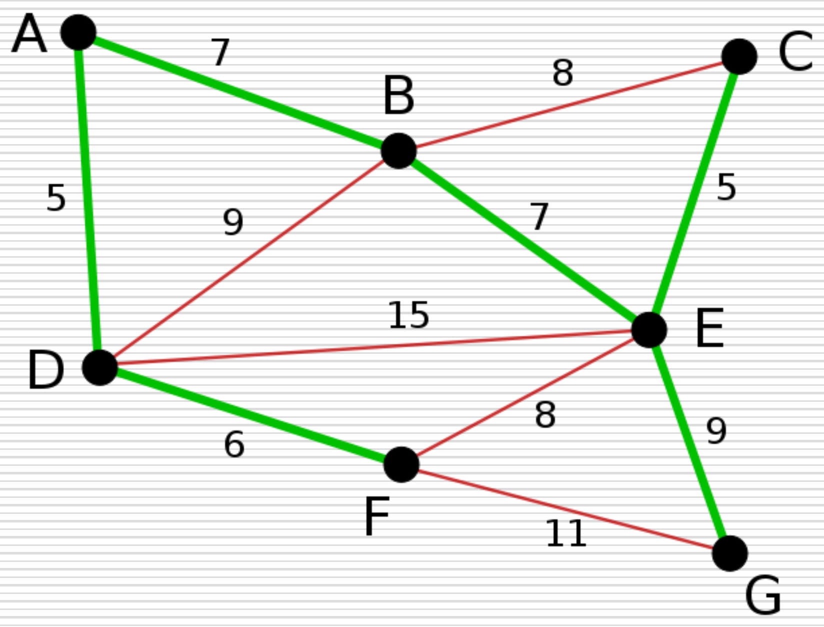
- Halting Problem (Turing):

  Given a description of a program, decide whether the program finishes running or will continue to run, and thereby run forever. This is equivalent to the problem of deciding, given a program and an input, whether the program will eventually halt when run with that input, or will run forever.

Why Study Complexity Theory (P)

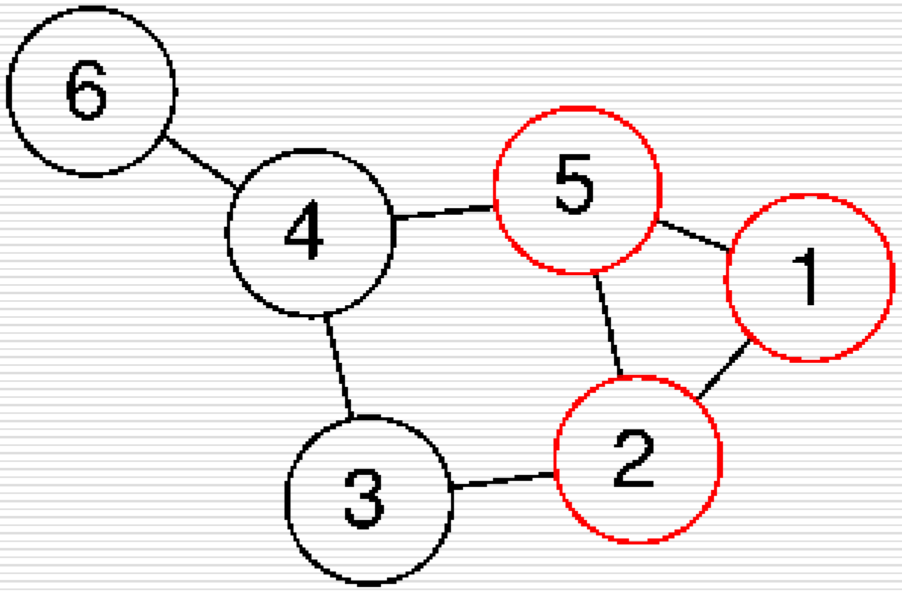
Kruskal Algorithm (Minimal spanning tree problem)





Why Study Complexity Theory (NP)

Clique Problem



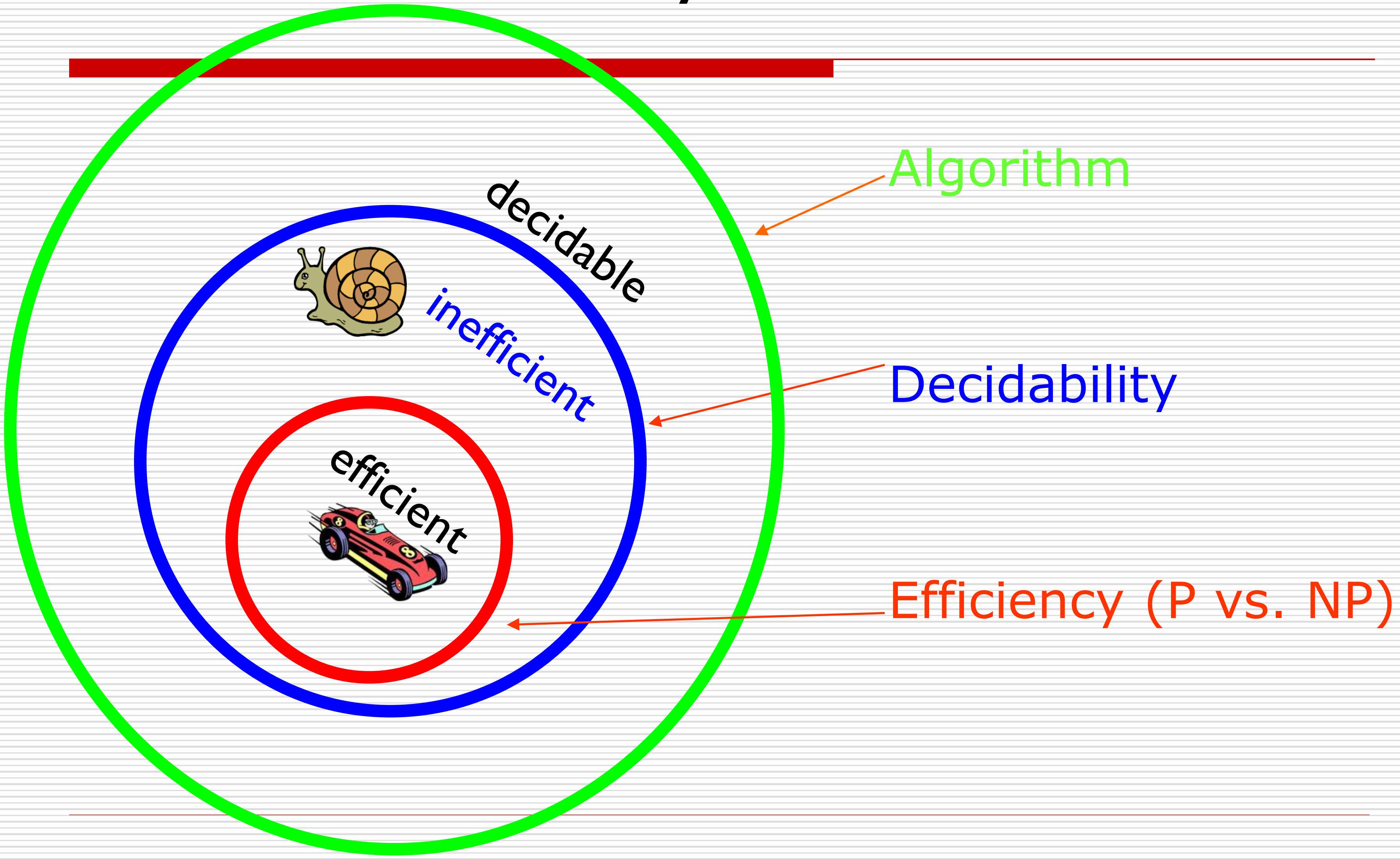
Why Study Complexity Theory (Application)

- Primality Testing and Cryptography

  There are a number of techniques based on RSA codes that enhance computer security, for which the most common methods in use today rely on the assumption that it is hard to factor numbers, that is, given a composite number, to find its prime factors.

- Shor's algorithm and Quantum Computation

Classical Theory



Welcome to Turing’s World

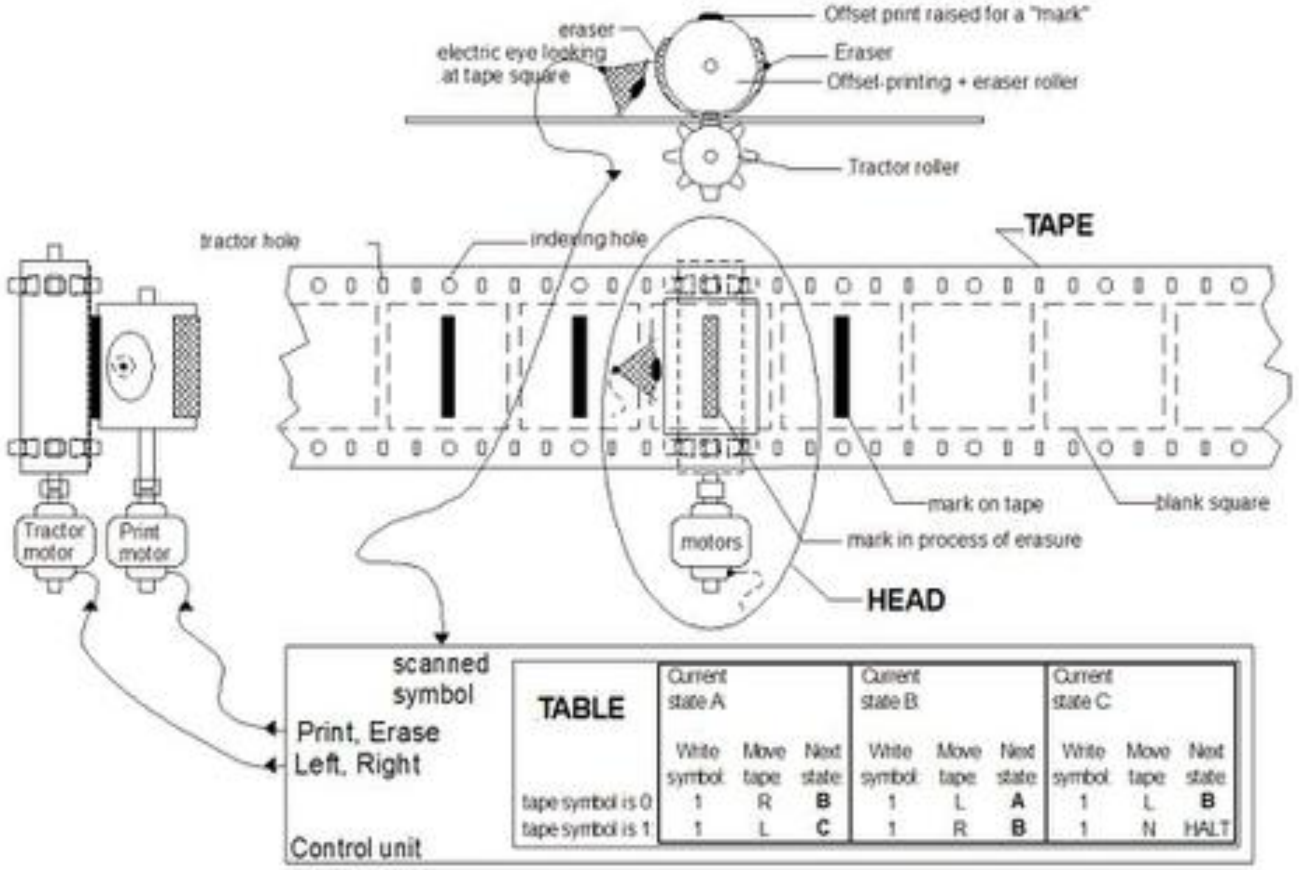


TABLE	Current state A			Current state B			Current state C		
	Write symbol	Move tape	Next state	Write symbol	Move tape	Next state	Write symbol	Move tape	Next state
tape symbol is 0	1	R	B	1	L	A	1	L	B
tape symbol is 1	1	L	C	1	R	B	1	N	HALT

A fanciful mechanical Turing machine's TAPE and HEAD. The TABLE instructions might be on another "read only" tape, or perhaps on punch-cards. Usually a "finite state machine" is the model for the TABLE.

Automata Theory

- Automata theory deals with the definitions and properties of mathematical models of computation.
- The theories of computability and complexity require a precise definition of a *computer.* Automata theory allows practice with formal definitions of computation

Computability Theory And Complexity Theory

- What can and can not be computed?
- How quickly can a problem be computed?

Next Class

- Basics of discrete Mathematics