AMERICAN INTERNATIONAL UNIVERSITY-BANGLADESH

CSC3113: THEORY OF COMPUTATION

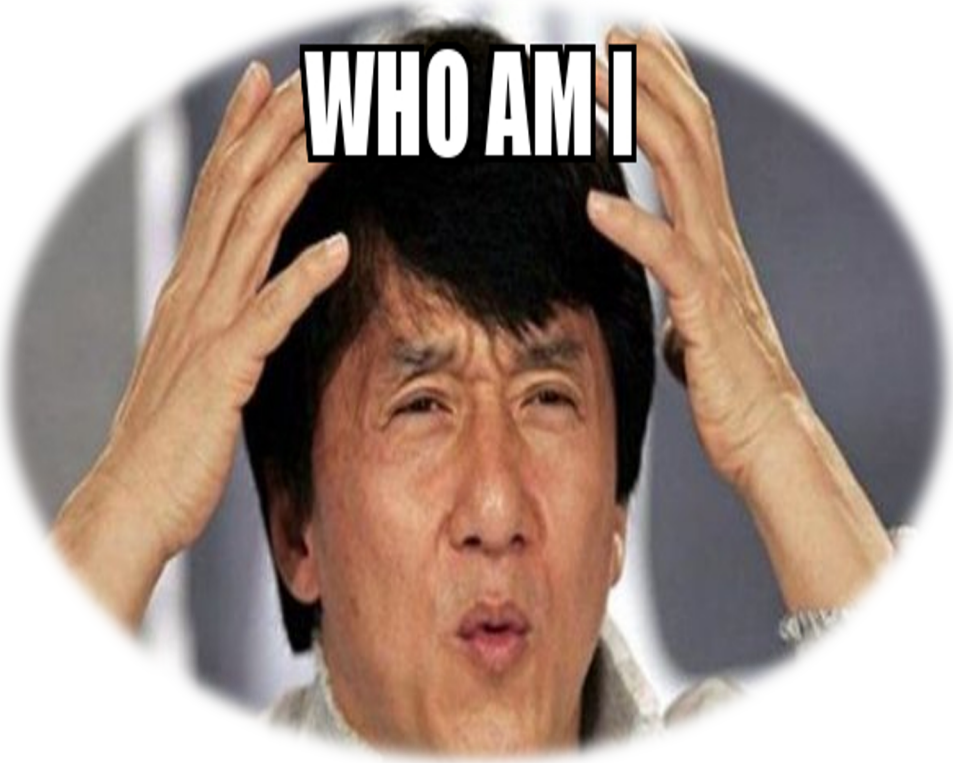
Lecture: # 1 **Week: #** 1 **Semester:** Summer 2020-2021

INTRODUCTION

Instructor: Dr. Hossain Md Shakhawat, Assistant Professor,
Department of Computer Science, Faculty of Science & Technology.
shakhawat@aiub.edu

More knowledge less fear

-theme of this lecture



LECTURE OUTLINE

- Mission and Vision
- Learning Objective & Outcome
- Class & Course Policies
- Exam & Evaluation Policies
- Course Objective
- Course Outcome
- Course Outline
- Pre-requisite
- Introduction to Theory of Computation

Learning Objective

Welcome & Introduction

- To know the vision & mission of AIUB and the FST.
- To understand the policies regarding class, course, exam and evaluation.
- To grasp the course content, outline, objective & outcome as a whole.
- To prepare with the pre-requisite course/topic for this course.
- Introduction to the course Theory of Computation.

LEARNING OUTCOME

WELCOME & INTRODUCTION

- Students will understand the vision & mission of the university and the faculty the belong to.
- Students will know the policies regarding –
  - Class attendance; Classroom behavior & interaction; Q/A session rules
  - Outside class interaction, consultation and communication.
  - Exams/viva/assignment… overall assessment policies and marks distribution
- Students will have a clear idea about the –
  - Objective and outcome of the course; Course outline/syllabus
  - Weekly distribution of the topics; Books and references
  - Pre-requisite course/topics provided as a reading material
- A brief introduction about “Theory” and “Computation” in computer science.

MISSION & VISION

American International University-Bangladesh

Vision: AMERICAN INTERNATIONAL UNIVERSITY-BANGLADESH (AIUB) envisions promoting professionals and excellent leadership catering to the technological progress and development needs of the country.

Mission: AMERICAN INTERNATIONAL UNIVERSITY-BANGLADESH (AIUB) is committed to provide quality and excellent computer-based academic programs responsive to the emerging challenges of the time. It is dedicated to nurture and produce competent world class professional imbued with strong sense of ethical values ready to face the competitive world of arts, business, science, social science and technology.

Department of Computer Science

Vision: Our vision is to be the preeminent Department of Computer Science through creating recognized professionals who will provide innovative solutions by leveraging contemporary research methods and development techniques of computing that is in line with the national and global context.

Mission: The mission of the Department of Computer Science of AIUB is to educate students in a student-centric dynamic learning environment; to provide advanced facilities for conducting innovative research and development to meet the challenges of the modern era of computing, and to motivate them towards a life-long learning process.

Faculty of Science & Technology

Vision: The Faculty of Science and Technology is committed to equip students with world class scientific research and industry-oriented knowledge and skills.

Mission: To create highly skilled and globally competitive professionals with advanced theoretical and applied knowledge responsive to the needs of the society in the discipline of science and technology.

CLASS & COURSE POLICIES

- At least 80% presence. Auto attendance is taken via TEAMS.
- Go through the topics before attending the class (from course outline).
- First 10 minutes of the class will be question/answer session of previous class topics.
- Every 20-25 minutes of the lecture, you will be given time to ask questions.
- Use the "Raise Hand" option to ask question. Or you can post the question on the chat/message box of the TEAMS course group.
- Outside class for any unresolved issue,
  - Consultation time will be provided. Make use of these times.
  - Use the TEAMS message box for appointment
  - You may also post your (topic related) questions on the chat/message option in TEAMS course group. Any student is allowed to answer/discuss the questions.

- **REMEMBER:**
  - **Your feedback is the key to the completion of this (online) course successfully.**
  - **After a topic is completed, prepare & ask QUESTIONS (during/after the class).**
  - **REPETATION is never a good solution as the lecture you heard before will be said again. To answer a question the teacher may repeat the topic in a different way as per your question.**

EXAM & EVALUATION POLICIES

Exam Policies

- All assessment will be conducted online using Microsoft TEAMS.
  - Learn & understand all technical aspect of online quiz and assignment, specially, document upload, time limits, Hand In/Turn In options, submission, etc.
  - When you are appearing for the exam, make sure to switch/turn off all other internet/background activities in your machine.
  - It is strictly recommended to use PC/Laptop for classes & exams, unless there is an emergency.
- There will be at least one assessment (quiz/assignment/viva) each week. Most assignment & quiz will be problem solving in nature. Quiz may also contain theoretical questions.
- Mid Semester assessment and Final Assessment will consist of Assignment, Exam & Viva on the defined week in the academic calendar.
- Any plagiarism will be result in grade "F".

Marks Distribution

- **All evaluation conducted online**
  - Attendance/Performance: 10%
  - Midterm Assessment
    - Quizzes: 40% (best will be counted)
    - Term [Exam1 + Exam 2]: (25 + 25) 50%
  - Final term Assessment
    - Quizzes: 30% (best will be counted)
    - Assignments: 20%
    - Term [Exam + Viva]: (20 + 20) 40%
- **Total:** 100%
- **Final Grade/ Grand Total**
  - Midterm: 50%
  - Final Term: 50%

COURSE OBJECTIVE

- Learn about different mathematical (formal) model for computation.
- Understand how to reason about computation using abstract, formal models.
- Go through the definitions of several specific models of computation including finite automata, context-free grammars, and Turing machines along with the tools for analyzing their power and limitations.
- Design and analyze how the nature of computation (solvable, unsolvable, efficient problems by computers) can be formalized as precise mathematical problems for different computational devices.
- Practice creative mathematical problem solving.

COURSE OUTCOME

AT THE END OF THIS COURSE, THE FOLLOWING OUTCOME SHALL HAVE BEEN ATTAINED

- Basic notations used in computer science literature
- Understand the mathematical model of Computation.
- Use of Computational models to solve problems for different computational devices
- Learn how to formally reason about computation.
- Develop the ability to compose correct, clear, and concise mathematical proofs.
- Understand Computability
- Determine Complexity of problems
- Learn the technology-independent foundations of CS

COURSE OUTLINE

- **Automata and Formal Language Theory:** Deterministic finite automata, nondeterministic finite automata, regular expressions; Pumping Lemma, non-regular languages; Pushdown automata and context-free languages.
- **Computability Theory:** Turing Machines and the Church-Turing thesis; Decidability, halting problem;
- **Complexity Theory:** Time complexity, space complexity; Complexity classes P, NP, PSPACE and the P vs. NP question. Polynomial time reductions and NP-completeness.
- Week wise topic distribution.
- **Books & References:**
  - *Introduction to the Theory of Computation*; (Latest Edition) by Michael Sipser.
  - *Introduction to Automata Theory, Languages, and Computation*; (Latest Edition) by John E. Hopcroft, et al.
  - *Elements of the Theory of Computation*, (Latest Edition) by Harry R. Lewis, Christos H. Papadimitriou.

PRE-REQUISITE

MATHEMATICAL REASONING AND ALGORITHMIC THINKING IS REQUIRED:

- Set theory (sub-set, power set etc)
- Sequence, Tuple
- Function & relation (domain, range etc)
- Propositional logic
- Graph theory, mathematical representation & algorithms
- Asymptotic notation, NP-completeness
- Proof by : Construction, Contradiction, Induction

** some reading material regarding these topics are given at the end.

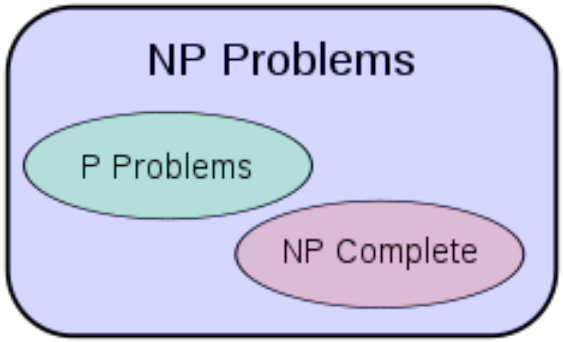
How to earn 1 MILLION dollars?

- The Clay Mathematics Institute of Cambridge, Massachusetts (CMI) established seven *Prize Problems, considered most difficult problems to solve*.

The **P versus NP problem**, one of the seven Millennium Prize Problems.

P: problems which can be solved in polynomial time

NP (nondeterministic P): Problems which can't be solved in polynomial time but if a solution is given it can be verified in polynomial time



- An answer to the **P** versus **NP** question would determine whether problems that can be verified in polynomial time can also be solved in polynomial time. Example of NP: sudoku, chess
- If it turned out that **P ≠ NP**, which is widely believed, it would mean that there are problems in **NP** that are harder to compute than to verify: they could not be solved in polynomial time, but the answer could be verified in polynomial time.

WHY STUDY THEORY OF COMPUTATION?

- Let us consider a scenario –
  - You are in a room full of different objects scattered all over the floor. You need to go from one end of the room to another end crossing over these objects. You may step onto the objects in the room or remove the objects from the path to crossover. But some objects may be too hard to step on or may be too heavy (or fixed on the floor) to be removed from the path.
- Now let us consider 3 conditions –
  - If you are crossing the room blindfolded, you would definitely step onto the objects in the room to crossover, but some objects may be too hard to step on which may result in painful experience on your legs.
  - If you are crossing with eyes open, you may remove/move away some of the objects while stepping towards the other end but may remove something which shouldn't be removed or may try to move something which you can never be moved.
  - If you are crossing the room with your eyes open and knowing the nature of the objects, you may reach the other end without any problem by removing the objects which can be removed and stepping on the (soft) objects which will not give you pain. Still there may be some instances where you need to step onto some hard objects, but this time you would be prepared as you know from before.
- This is known as classical difference between "Walking the path" and "Knowing the path".

...Why study Theory of Computation?

- Computer science is all about solving problem using the computational capacities (software/hardware) of a computing devices/tools. So, it is very important to know the model of these computations, capabilities & limitations of these devices/tools, and the complexity of the solutions of the given problem, which is the content of the theory of computation.
- At the end of your graduation you can be a programmer, developer, software engineer or any other CS professionals with clear concept about your favorite areas of computer science (programming/software engineering etc.), without even knowing the theory of computation.
  - But that would be like the first (blindfolded) group, where you would be dependent on the software tools or device configuration given to you to solve the problem. To reach at the pick level of your profession, you will be stepping onto a lot of hard objects where you need to learn some of the concept of theory of computation to complete your solutions and carry on towards your goal.
  - If you are an alert/talented person and have a reasonable idea about TOC like the second (eyes-open) group, you would still be dependent and need to go through the painful process of re-learning to complete your solutions and achieve your goal.

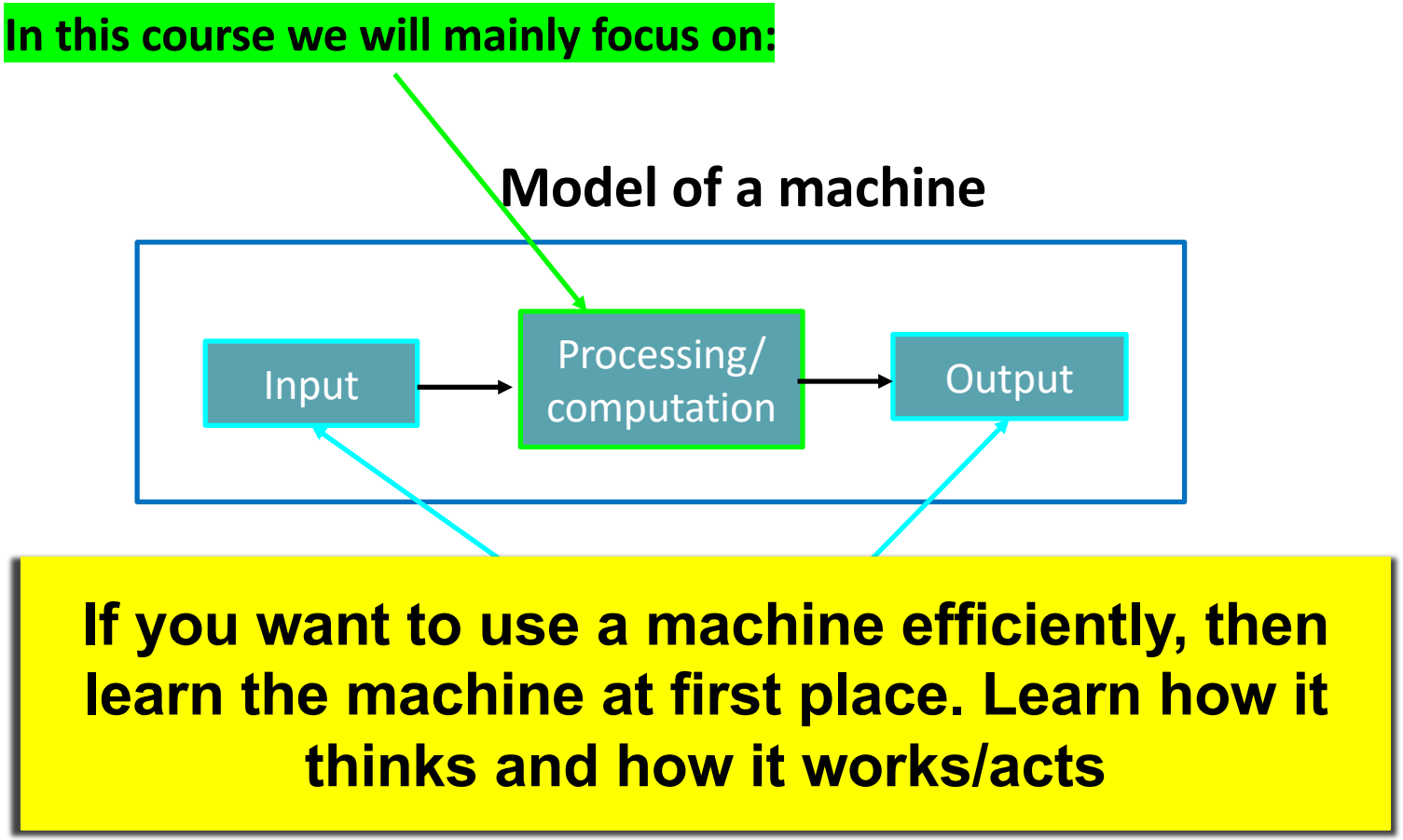
...Why study Theory of Computation?

- If you have a good concept of the theory of computation along with your favorite areas of computer science, you will definitely have a privilege over others to achieve your professional goal. You will not be completely dependent on the given capacities of the devices/tools. As you have the idea of computation, you can come up with your own tools/devices based on the requirement of your solution.
- In present technological world (IoT, BigData, AI, etc.) everything is a blending of hardware and software. To be a better computer professional, you must know the computational models, capabilities, and/or complexities to put a software onto a hardware or engineer a hardware for an intended software.
- In theory of computation you will learn abstract machines, systems, or model of computation, which will be defined mathematically. Every change or enhancement we made on a mathematical computational model of devices/systems, can surely be implemented as the theory gives us fundamental capabilities and limitations of these devices/systems. Now the opposite is not possible. You cannot make a device or system first then come up with a theory to sustain it.

…Why study Theory of Computation?

- The importance to study the theory of computation is to
  - Better understand the development of formal mathematical models of computation that reflect the real-world of computer.
  - To achieve deep understanding about the mathematical properties of computer hardware and software.
  - Mathematical definitions of the computation and the algorithm.
  - To rectify the limitations of computers and answer what kind of problems can be computed?
- **Usability** of TOC**:**
  - Machines Models (different computational devices/SW tools)
  - Problem/solution Models (machines/algorithms can be used to solve)
  - Theorems about what types of machines can solve what types of problems, and at what cost.
- We will work on **Sequential** and **single-processor** computing, Focus on **decision problems** (yes/no answers) on **discrete** inputs.

...Why study Theory of Computation?

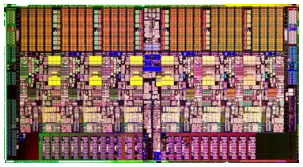


How we study the Computation of Machine?

If we want to learn how the machine works, then we need to study how its parts/hardware functions. But studying hardware is difficult

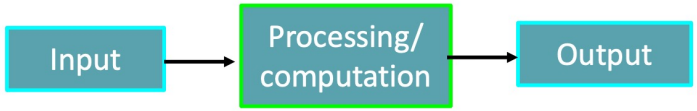
Rather, we create theoretical model of machine, **called abstract machines**. In ToC we study this abstract machine

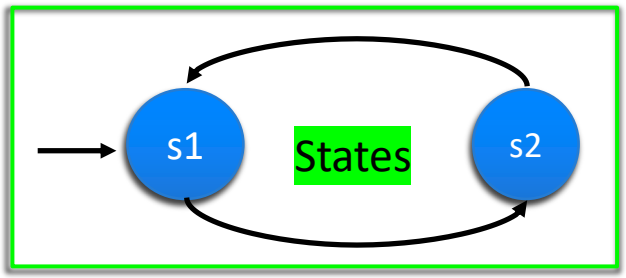
Actual machine



VS

Abstract model of machine





Represent computation as states

What is Theory of Computation?

What is computation?

-nothing but **doing something by a machine (executing algorithm)**

But is the program computable?

Model of computation inside a machine

If yes, then how long it will take ?
100 years or 100 seconds

- The theory of computation "**deals with how efficiently problems can be solved on a model of computation (a machine), using an algorithm**".

- **Central areas of the theory of computation:** automata, computability, and complexity. They are linked by the question: What are the fundamental capabilities and limitations of computers?

"Theory" of "Computation"

- **Computation:** The processing of information by the unlimited application of a finite set of operations or rules.
- **Theory:** General ideas that apply to many different systems (Independence from Technology); Expressed simply, abstractly, and precisely.
  - Precision: Can be used to prove formal mathematical theorems
    - Positive results (what can be computed): correctness of algorithms and system designs
    - Negative results (what cannot be computed): proof that there is no algorithm to solve some problem in some setting (with certain cost)

Linguistics

- Linguistics is the scientific study of language. It involves the analysis of language form, language meaning, and language in context.
- Language theory is a branch of mathematics concerned with describing languages as a set of operations over an alphabet.
- We, the human, communicate with language. In written form, we use different symbols to express. But a machine has no capability to understand such symbols.
- We know from our previous courses, that any computing system works on the format – Input → Process → Output.
- And every input we provide is nothing but a stream of symbols (even a picture or video is converted to streams of symbols) that is later converted to numbers and lastly into signals that a device can process.
- For every models in this course we will be using the symbols (a-z, 0-9, #, $, etc.) as input and try to come up with a model of computation to process these streams into a meaningful output.
- Our main focus would be to process a given language using a given configuration of machine/system (mathematical model) and find out if this language can be processed by this machine or not (decision problem).

THEORY OF COMPUTATION

- We will deal with –
  - Mathematical abstraction of computing devices/systems called a model of computation.
  - What problems can be solved on a model of computation, using some computational steps (algorithm),
  - How efficiently they can be solved or to what degree (e.g., approximate solutions versus precise ones).
- Theory of computation (TOC) is based on analysis of the fundamental capabilities and limitations of computing devices/systems.
- The objective is to **Build** a **theory** out of the **idea** of **computation**.
- Three areas are explored for this purpose –
  - Automata (formal languages)
  - Computability
  - Complexity
- Though these ideas and models are mathematical in nature, each of these three areas has different interpretation of the analysis. And the solution also very according to the interpretation.

AUTOMATA THEORY

- Automata (Αυτόματα) means that something is doing something by itself. Any machine that act automatically called automata. Example: computer, ATM and others.
- **Automatons** are abstract models of machines that perform computations on an input by moving through a series of states
- There are **four major families of automaton** :
    1. **Finite-state machine**
    2. **Pushdown automata**
    3. **Linear-bounded automata**
    4. **Turing machine**

If we can create any of the four models for a problem, then the problem is solvable/computable by a machine/computer; otherwise, the problem is not computable

AUTOMATA THEORY

- Automata comes from the Greek word (Αυτόματα) which means that something is doing something by itself.
- Automata deals with the study of abstract (mathematical model) machines or systems (definition and properties) and the computational problems (defined in terms of formal languages) that can be solved (recognized) using these machines.
  - An automaton can be a finite representation of a formal language that may be an infinite set (language theory). Formal languages are the preferred mode of specification (input) for any problem that must be computed (processed).
  - Automata are used as theoretical models for computing machines (input, process, output),
  - These abstract computing machines are used for proofs about computability (solvability).
- Such models include –
  - *finite automaton*, used in text processing, compilers, and hardware design
  - *Context-free grammar*, used in programming languages and artificial intelligence

COMPUTABILITY THEORY

- The theories of computability and complexity are closely related.

- In complexity theory classifies problems as easy ones and hard ones; whereas computability theory classify problems as solvable or not.
- Computability theory introduces several of the concepts used in complexity theory.

- During the 20th century, mathematicians discovered that certain problems can't be solved by computers. One example: the problem of determining whether a mathematical statement is true or false. This is bread and butter in math. But no computer algorithm can perform this task.

- Development of ideas concerning theoretical models of computers that eventually would help to lead to the construction of actual computers.

Complexity Theory

- The central question of complexity theory: What makes some problems computationally hard and others easy?

- Complexity theory focuses on classifying problems according to their resource usage. A problem is considered hard if its solution requires significant resources. Major form of resources are time and storage (Others: number of gates, circuits).

- One of the roles of complexity theory is to determine the practical limits on what computers can and cannot do. The P versus NP problem, one of the seven Millennium Prize Problems, is dedicated to the field of computational complexity.

Complexity Theory

- Deals with the method and ideas to decide if problems are computationally hard and/or easy.
- There are elegant schemes for classifying problems according to their computational difficulties. For example –
  - Try to alter the aspects of the problem which is at the root of the difficulty so that the problem is more easily solvable.
  - Settle for less than a perfect solution to the problem. In certain cases finding solutions that only approximate the perfect one is relatively easy.
  - Some problems are hard only in the worst-case situation, but easy most of the time.
- Cryptography is one of the examples which requires computational problems that are hard, rather than easy, because secret codes should be hard to break without the secret key or password.

Linking to other areas of science

- Finite automata arise in compilers, AI, coding, chemistry.
- Hard problems are essential to cryptography.
- Computation occurs in cells/DNA, the brain, economic systems, physical systems, social networks, etc.
- Following is a link where you will find many more areas that is linked with this course.

  https://cstheory.stackexchange.com/a/14818

REFERENCES

- Introduction to Theory of Computation, Sipser, (3rd ed), Chapter-1.
- AIUB Lectures, Mashiour Rahman, Pre-requisite topics.

TAKE AWAY?

What is the purpose of studying Theory of computation?

- To learn how a machine works, so that we can develop an efficient algorithm for that machine to use the machine for some work. Or we may develop a better machine in future

What are the major branches of Theory of computation?

- Automata theory
- Computability theory
- Complexity theory

What is the difference between computability and complexity theory?

-Computability theory is concerned with what can be computed/solved vs what cannot; **complexity is** concerned with the resources required to compute the things that are computable

TAKE AWAY?

How to know if a problem is possible to compute/solve using a machine/computer?

- If it is possible to create an automaton model (any of the four) for a problem, then that problem can be computed/solved by a computer

How are machines studied in ToC?

- By creating abstract models of machine and creating abstract models of computation (as states)