Theory of Computation
CSC 339 – Spring 2021

Introduction
Week-1&2

King Saud University

Department of Computer Science

Dr. Azzam Alsudais

Outline

- Complexity Theory
- Computability Theory
- Automata Theory
- Mathematical and terminology review

Theories ..

- **Complexity Theory**
- **Computability Theory**
- **Automata Theory**

Theories ..

- **Complexity Theory**
  - What makes some problems computationally hard and others easy?
- **Computability Theory**
- **Automata Theory**

Theories ..

- **Complexity Theory**
  - What makes some problems computationally hard and others easy?
- **Computability Theory**
  - Can we determine if a problem is solvable or not?
- **Automata Theory**

Theories ..

- **Complexity Theory**
  - What makes some problems computationally hard and others easy?
- **Computability Theory**
  - Can we determine if a problem is solvable or not?
- **Automata Theory**
  - Can we design models of computations to solve certain problems?

Complexity Theory

- **The million dollar question**
  - *“What makes some problems computationally hard and others easy?”*
- **This question has been studied for over 40 years**

Complexity Theory

- **The million dollar question**
  - *“What makes some problems computationally hard and others easy?”*
- **This question has been studied for over 40 years**
- **We cannot <u>prove</u> that a given problem is computationally hard**
- **But we can point to some evidence that certain problems are computationally hard..**

Complexity Theory

- **What to do when you encounter a computationally hard problem?**

Complexity Theory

- What to do when you encounter a computationally hard problem?
  - Identify the root of the difficulty, and alter it to make the problem more tractable.

Complexity Theory

- What to do when you encounter a computationally hard problem?
  - Identify the root of the difficulty, and alter it to make the problem more tractable.
  - It might be fine to settle for a less than perfect solution.

Complexity Theory

- What to do when you encounter a computationally hard problem?
  - Identify the root of the difficulty, and alter it to make the problem more tractable.
  - It might be fine to settle for a less than perfect solution.
  - Some problems are hard only in worst case scenario, so design solutions that are optimized for the common case.

Complexity Theory

- **What to do when you encounter a computationally hard problem?**
  - **Identify the root of the difficulty, and alter it to make the problem more tractable.**
  - **It might be fine to settle for a less than perfect solution.**
  - **Some problems are hard only in worst case scenario, so design solutions that are optimized for the common case.**
  - **May consider alternate types (e.g., randomized) of computation to speed up tasks.**

Computability Theory

- **Some “basic” problems cannot be solved by computers**
  - **Determining whether a mathematical statement is true of false.**

Computability Theory

- **Complexity and computability theories are closely related**

Computability Theory

- **Complexity and computability theories are closely related**
  - **In complexity theory, the objective is to classify problems as hard or easy ones.**

Computability Theory

- **Complexity and computability theories are closely related**
  - **In complexity theory, the objective is to classify problems as hard or easy ones.**
  - **In computability theory, the classification of problems is by those that are solvable and those that are not.**

Automata Theory

- **Automata theory deals with definitions and properties of mathematical models of computation.**

Automata Theory

- Automata theory deals with definitions and properties of mathematical models of computation.

- Such models comprise a significant portion of applied areas in computer science.

Automata Theory

- **Automata theory deals with definitions and properties of mathematical models of computation.**
- **Such models comprise a significant portion of applied areas in computer science.**
  - **e.g., finite automata is used in text processing, compilers, and HW design**
  - **Context-free grammer (CFG) is used in PL and AI.**

Mathematical and Terminology Review

- Sets
- Sequences & Tuples
- Functions
- Graphs
- Strings and Languages

Mathematical and Terminology Review

- **Sets**
  - **A set is a group of objects (aka members, elements) represented as a unit. Order of elements does not matter.**
  - **It contains only 1 occurrence of each element**
  - **S1 = {1,2,3} is equivalent to S2 = {2,1,3}**
  - **Is S3 = {1,{2,3}} equivalent to S1 ?**
  - **Is S4 = {1,2,2,} equivalent to S1 ?**
  - **Unions and intersections**

Mathematical and Terminology Review

- **Sequences and tuples**
  - **Order of elements does matter.**
  - **(1,2,3) is not the same as (3,2,1).. whereas {1,2,3} is the same as {3,2,1}**
  - **K-tuple: (1,2,3) is a 3-tuple**

Mathematical and Terminology Review

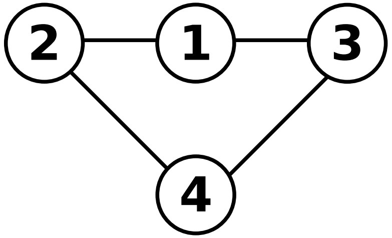
- **Functions**
  - **“a *function* is an object that sets up an input-output relationship.”**
  - **Example..**
    - ***Increment(x)***
    - **Takes x as input, adds 1, and outputs result**
  - **Can take multiple arguments**
    - **Unary functions take single argument**
    - **Binary functions take two**
    - **3-ary functions take three, and so on.**
  - **Predicates are special type of functions (returns True or False)**

Mathematical and Terminology Review

- **Graphs**
  - **nodes and edges**
  - **Path: a sequence of nodes connected by edges**
  - **A graph is *connected* if every two nodes have a path between them**
  - **Graphs can be undirected or directed**
    - **Edges in directed graphs have arrows pointing to their direction**
    - **Outdegree (# of outgoing edges) & indegree (# of incoming edges)**

Mathematical and Terminology Review

- **Graphs**
  - **G = (V,E)**
  - **V = {1,2,3,4}**
  - **E = {{1,2}, {1,3},{2,4},{3,4}}**



Mathematical and Terminology Review

- **Strings and languages**
  - **Alphabet is a non-empty finite set**
  - **A string over an alphabet is a finite sequence of symbols from that alphabet.**
  - **{0,1}**
  - **{a,b,c,d,e,f,g,h,i,j,k,l,m,n,o,p,q,r,s,t,u,v,w,x,y,z}**
  - **{0,1,x,y,z}**

Mathematical and Terminology Review

- **Strings and languages**
  - **Prefix**
    - **xz = y .. string x is a prefix of string y**
    - **x is a proper prefix of y if x $\neq$ y**

Mathematical and Terminology Review

- **Boolean logic**
  - **True or False {0,1}**
  - **Conjunction (AND)**
    - **0 ^ 1 = 0**
    - **1 ^ 1 = 1**
  - **Disjunction (OR)**
    - **0 v 1 = 1**
    - **0 v 0 = 0**
  - **Exclusive or (XOR)**
    - **0 xor 0 = 0**
    - **1 xor 0 = 1**
    - **1 xor 1 = 0**

Homework

- Exercise
  - 0.3, 0.8
- Reading
  - 1.1 (p31 - p47)