UNIVERSITY OF LIBERAL ARTS BANGLADESH

Automata and Theory of Computation

CSE 417

Lecture 1-2

Dr. Nafees Mansoor

Student's Introduction

- **Name**
  - I will try to remember your names. But if you have a Long name, please let me know how should I call you 😆
- **Major and Academic status**
- **Thoughts on Programming**
  - Java, C/C++, VB, Matlab, Scripts etc.
- **Expectation from this course**

ULAB UNIVERSITY OF LIBERAL ARTS BANGLADESH

Instructor

DR. NAFEES MANSOOR

Assistant Professor

Department of Computer Science and Engineering

Email: nafees.mansoor@ulab.edu.bd

Office: Room PC315

Class Hours: 11:30 - 13:00 (S/T)

Consultation Hours: 13:30 – 14:30 (S/T)

Note: Available by appointment at other hours (e.g. email)

Focus

- Introduction to the theory of computation
- Topics that will be covered in this course
  - Finite automata (DFA, NFA)
  - Regular expressions
  - Minimization of DFA, equivalences of DFA and NFA
  - Regular expressions
  - Context free grammar
  - Push Down Automata
  - Turing machine

Contents

ILO	Topic	Teaching Strategy	Assessment Strategy	Number of Sessions
1-2	Introduction to Automata and Theory of Computation	Lecture Exercise	Q/A Assignment	4
1-3	Finite State Machines	Lecture Exercise	Q/A	6
1-3	Regular Expressions	Lecture Exercise	Q/A	3
1-5	Context Free Grammar	Lecture Exercise	Q/A Assignment	5
1-5	Push Down Automata	Lecture Exercise	Q/A	2
1-5	Turing Machine & Decidability	Lecture Exercise	Q/A Assignment	4
			Total	24

Weight Assessments

- Assignments 10%
- Quiz 10%
- Class Participation 10%
- Midterm 20%
- Final Exam 50%

Grading

A relative or bell-curve grading system will be followed, so that the majority will receive a middle grade, and only a few will get A/A-, or F. The course teacher will assign mark ranges to each letter grade, taking into account the assessment components and assigned weights, difficulty level, average academic ability of the class, etc

I have a simple policy

Grade is something

That shouldn't be given

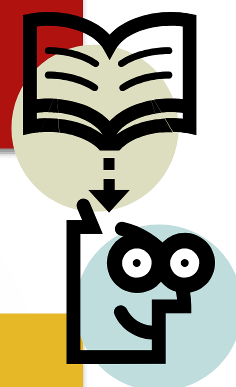
It should be

earned

Reference Books

Introduction to the theory of computation

by Michael Sipser



Introduction to automata theory, languages, and computation

by John E. Hopcroft, Rajeev Motwani and Jeffrey D. Ullman

Questions

Why to study Automata Theory

To understand the limits of computation

- Some problems require more resources to compute, and others may be computed with less.
- To study these issues we need mathematical notions of “resource” and “compute”.
- We’ll study different “machine models” (finite automata, pushdown automata). . .

Why to study Automata Theory

Direct application in creating

- Compilers,
- Text editors,
- Communications protocols,
- Hardware design, . . .

- First compilers took several person-years; now can be written by a single student in one semester 😊 thanks to theory of parsing.

To learn to think analytically about computing

Divided into three areas

- **Computability Theory**
  - Classifies problems as solvable or not solvable
- **Complexity Theory**
  - Classifies problems as easy ones and hard ones
- **Automata Theory**
  - Deals with definitions and properties of mathematical models of computation

Mathematical Notions and Terminology

Set Theory

- What are Sets?

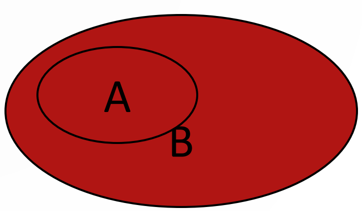
  Sets are collections of objects

- S={a, b, c} refers to the set
  - whose elements are a, b and c.
- $a \in S$
  - means "a is an element of set S".
- $d \notin S$
  - means "d is *not* an element of set S".
- $\{x \in S \mid P(x)\}$
  - is the set of all those x from S such that P(x) is true.
  - *E.g.*, $T=\{x \in \mathbf{Z} \mid 0<x<10\}$.

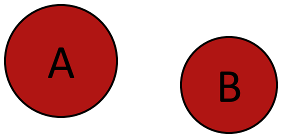
Set Theory: Relation between Sets

- **Definition:** Suppose A and B are sets. Then A is called a **subset** of B: **$A \subseteq B$ iff** every element of A is also an element of B.

  Symbolically,
- $A \subseteq B \Leftrightarrow \forall x$, if $x \in A$ then $x \in B$.
- $A \not\subset B \Leftrightarrow \exists x$ such that $x \in A$ and $x \notin B$.



$A \subseteq B$



$A \not\subset B$

Set Theory: Relation between Sets

- **Definition:** Suppose A and B are sets. Then

  A **equals** B: **A = B**

  iff every element of A is in B and

  every element of B is in A.

  Symbolically,

  $A=B \Leftrightarrow A \subseteq B$ and $B \subseteq A$.

- **Example:** Let $A = \{m \in \mathbf{Z} \mid m=2k+3$ for some integer $k\}$;

  B = the set of all odd integers.

  Then A=B.

Set Theory: Operations

Definition: Let A and B be subsets of a set U.

1. Union of A and B: $A \cup B = \{x \in U \mid x \in A \text{ or } x \in B\}$
2. Intersection of A and B:

$$A \cap B = \{x \in U \mid x \in A \text{ and } x \in B\}$$

3. Difference of B minus A: $B - A = \{x \in U \mid x \in B \text{ and } x \notin A\}$
4. Complement of A: $A^c = \{x \in U \mid x \notin A\}$

Ex.: Let $U=\mathbf{R}$, $A=\{x \in \mathbf{R} \mid 3<x<5\}$, $B=\{x \in \mathbf{R} \mid 4<x<9\}$. Then

1) $A \cup B = \{x \in \mathbf{R} \mid 3<x<9\}$.
2) $A \cap B = \{x \in \mathbf{R} \mid 4<x<5\}$.
3) $B - A = \{x \in \mathbf{R} \mid 5 \le x<9\}$, $A - B = \{x \in \mathbf{R} \mid 3<x \le 4\}$.
4) $A^c = \{x \in \mathbf{R} \mid x \le 3 \text{ or } x \ge 5\}$, $B^c = \{x \in \mathbf{R} \mid x \le 4 \text{ or } x \ge 9\}$

Set Theory: Properties

- Commutative Laws:

$$(a)\ A \cap B = B \cap A$$

$$(b)\ A \cup B = B \cup A$$

- Associative Laws:

$$(a)\ (A \cap B) \cap C = A \cap (B \cap C)$$

$$(b)\ (A \cup B) \cup C = A \cup (B \cup C)$$

- Distributive Laws:

$$(a)\ A \cup (B \cap C) = (A \cup B) \cap (A \cup C)$$

$$(b)\ A \cap (B \cup C) = (A \cap B) \cup (A \cap C)$$

- Double Complement Law:

$$(A^c)^c = A$$

- De Morgan's Laws:

$$(a)\ (A \cap B)^c = A^c \cup B^c$$

$$(b)\ (A \cup B)^c = A^c \cap B^c$$

- Absorption Laws:

$$(a)\ A \cup (A \cap B) = A$$

$$(b)\ A \cap (A \cup B) = A$$

Set Theory: Empty Set

- The unique set with no elements

  is called **empty set** and denoted by ∅.
- Set Properties that involve ∅ **.**

  For all sets A,

1. $\emptyset \subseteq A$
2. $A \cup \emptyset = A$
3. $A \cap \emptyset = \emptyset$
4. $A \cap A^c = \emptyset$

Set Theory: Disjoint Set

- A and B are called **disjoint** iff

$$A \cap B = \emptyset$$

- Sets $A_1, A_2, \ldots, A_n$ are called **mutually disjoint** *iff* for all i,j = 1,2,…, n

$$A_i \cap A_j = \emptyset \quad \text{whenever } i \neq j.$$

- *Examples:*

1) A={1,2} and B={3,4} are disjoint.

2) The sets of even and odd integers are disjoint.

3) A={1,4}, B={2,5}, C={3} are mutually disjoint.

4) A−B, B−A and A∩B are mutually disjoint.

Set Theory: Power Set

- **Definition:** Given a set A,

  the **power set** of A, denoted $\mathcal{P}$**(A)** ,

  is the set of all subsets of A.

- *Example:* $\mathcal{P}(\{a,b\}) = \{\emptyset, \{a\}, \{b\}, \{a,b\}\}$.

Properties:

1) *If* $A \subseteq B$ *then* $\mathcal{P}(A) \subseteq \mathcal{P}(B)$.
2) *If* a set A has n elements
   then $\mathcal{P}(A)$ has 2^n elements.

Graph Theory

- A graph *G(V,E)* is two sets of object
  - Vertices (or nodes) , set *V*
  - Edges, set *E*
- A graph is represented with dots or circles (vertices) joined by lines (edges)
- The magnitude of graph *G* is characterized by number of vertices *|V|* (called the order of *G*) and number of edges |*E*| (size of *G)*

Graph Theory: Applications

Graph (Network)	Vertexes (Nodes)	Edges (Arcs)	Flow
Communications	Telephones exchanges, computers, satellites	Cables, fiber optics, microwave relays	Voice, video, packets
Circuits	Gates, registers, processors	Wires	Current
Financial	Stocks, currency	Transactions	Money
Transportation	Airports, rail yards, street intersections	Highways, railbeds, airway routes	Flights, vehicles, passengers

- **Types of graphs**
  - **Directed graphs**

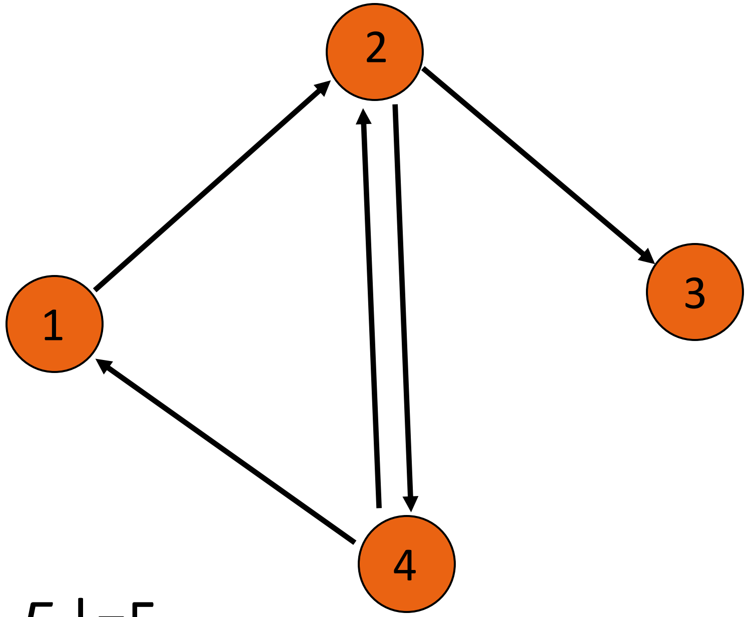
    G=(V,E) where *E* is composed of ordered pairs of vertices; i.e. the edges have direction and point from one vertex to another.
  - **Undirected graphs**

    G=(V,E) where *E* is composed of unordered pairs of vertices; i.e. the edges are bidirectional.

Directed Graph

An edge $e \in E$ of a directed graph is represented as an ordered pair (u,v), where $u, v \in V$.

Here u is the initial vertex and v is the terminal vertex, assuming that $u \neq v$

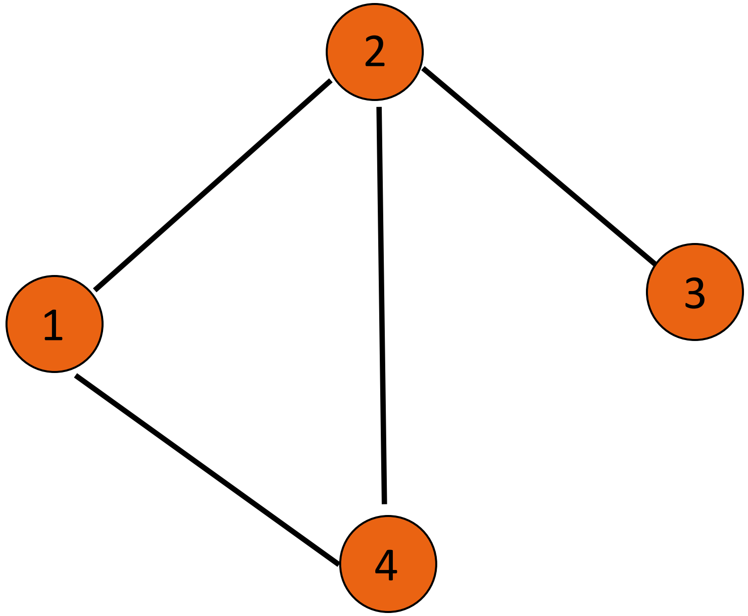


$V = \{ 1, 2, 3, 4\}$, $|V| = 4$

$E = \{(1,2), (2,3), (2,4), (4,1), (4,2)\}$, $|E| = 5$

Undirected Graph

An edge $e \in E$ of an undirected graph is represented as an **unordered pair** $(u,v)=(v,u)$, where $u, v \in V$. Also assumed that $u \neq v$

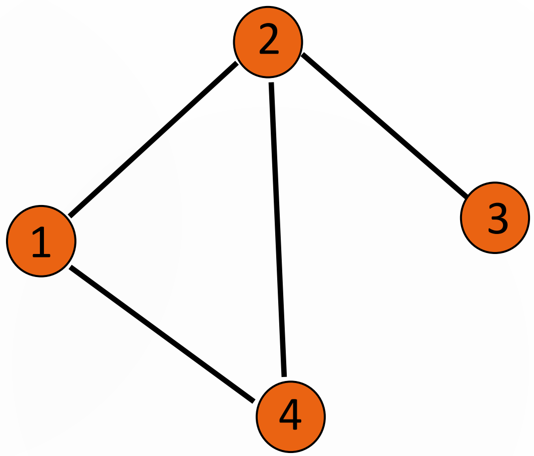


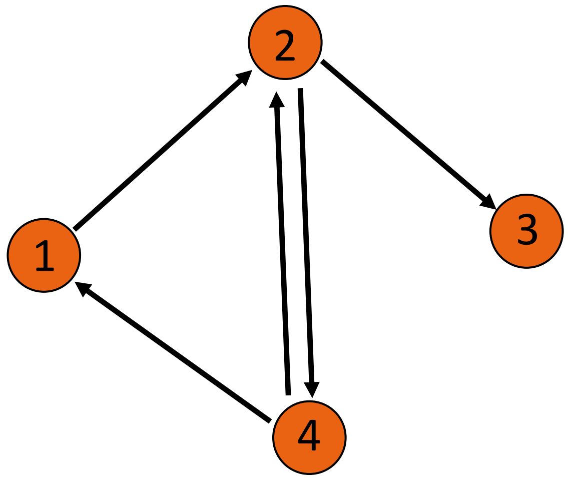
$V = \{1, 2, 3, 4\}$, $|V| = 4$
$E = \{(1,2), (2,3), (2,4), (4,1)\}$, $|E| = 4$

Degree of a Vertex

Degree of a vertex in an undirected graph is the number of edges incident on it.

In a directed graph, the *out degree* of a vertex is the number of edges leaving it and the *in degree* is the number of edges entering it





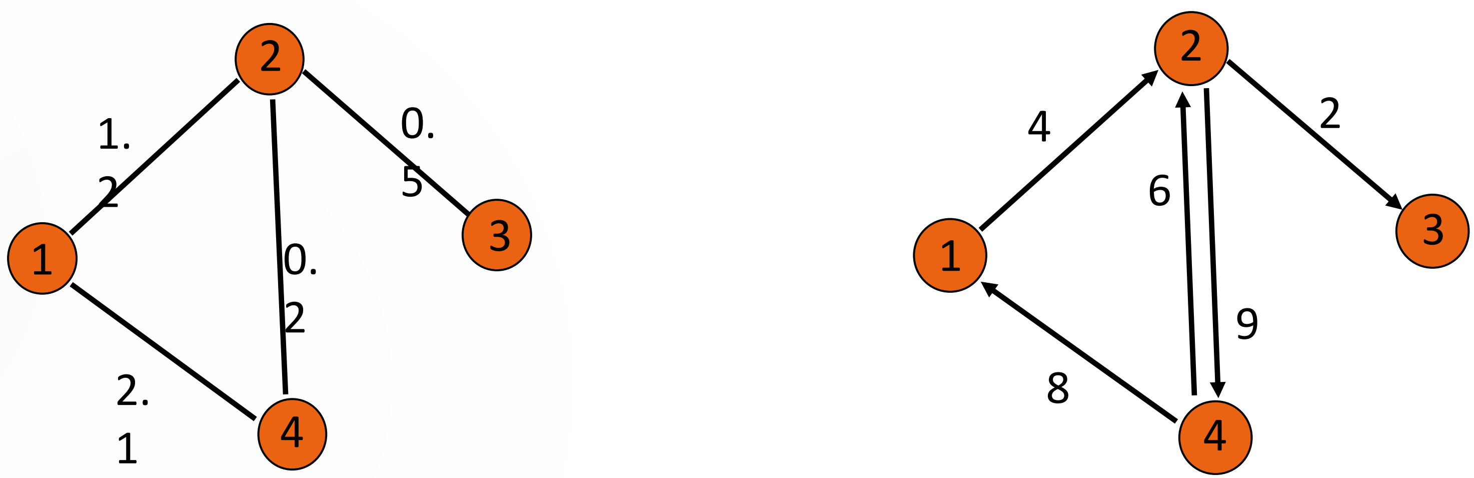
The *degree* of vertex 2 is ???

Ans: 3

The *in degree* of vertex 2 is 2 and the *in degree* of vertex 4 is 1

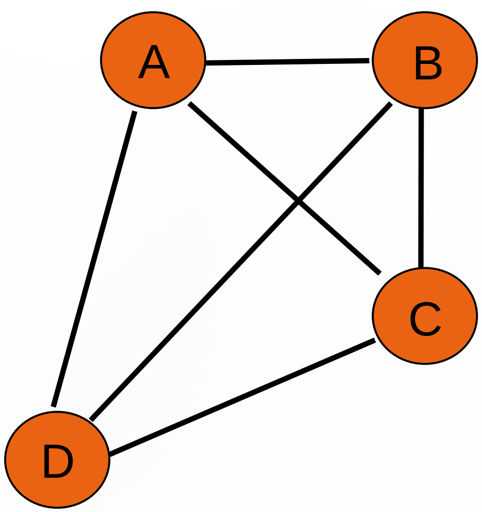
Weighted Graph

A *weighted graph* is a graph for which each edge has an associated *weight*, usually given by a *weight function* $w: E \rightarrow R$



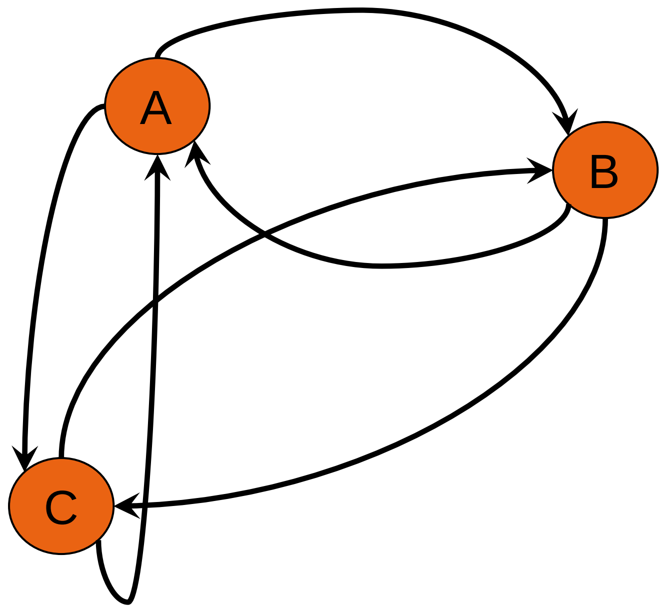
Complete Graph

A *complete graph* is an undirected/directed graph in which every pair of vertices is *adjacent*. If (*u*, *v*) is an edge in a graph *G*, we say that vertex *v* is *adjacent* to vertex *u*.



If an undirected graph G has V nodes, how many edges are required to define it as a complete graph

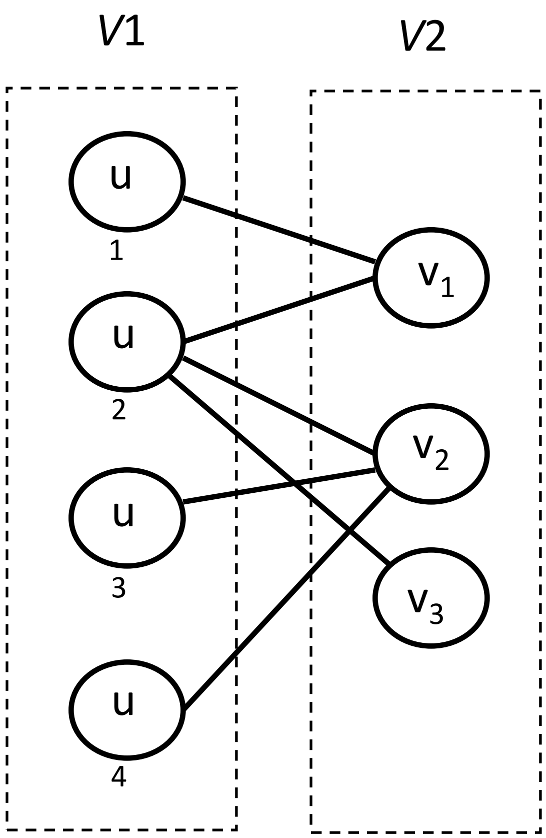
*Ans: V**(*V*-1)/2 edges



If an directed graph G has V nodes, how many edges are required to define it as a complete graph

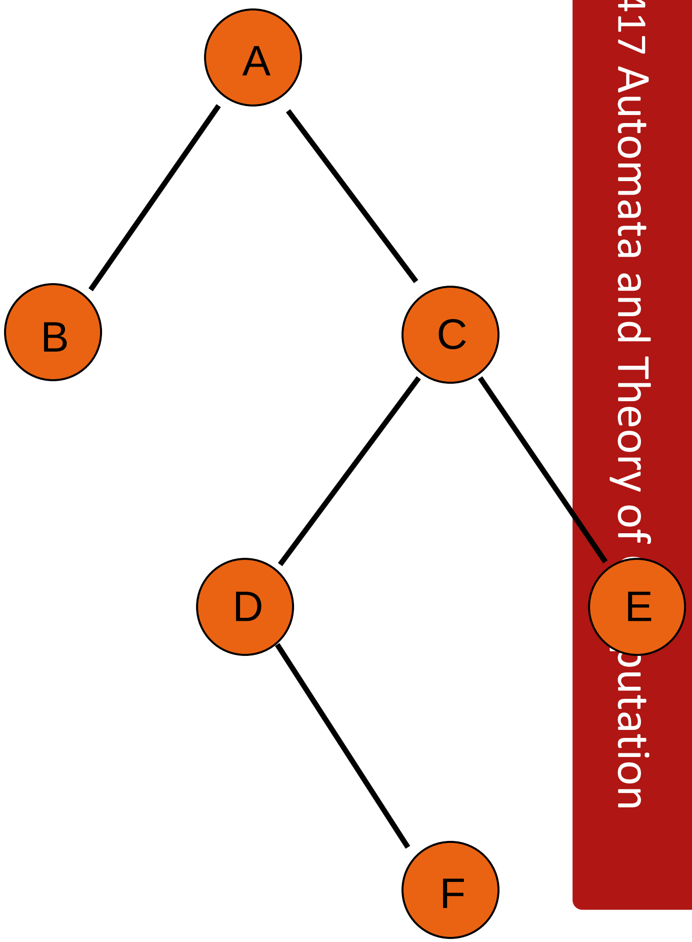
*Ans: V**(*V*-1) edges

A *bipartite graph*
is an undirected graph
$G = (V,E)$ in which V can
be partitioned into 2
sets $V1$ and $V2$ such that
$(u,v) \in E$ implies either
$u \in V1$ and $v \in V2$
OR
$v \in V1$ and $u \in V2$.



Let $G = (V, E)$ be an undirected graph. The following statements are equivalent,

1. G is a tree
2. Any two vertices in G are connected by unique simple path
3. G is connected, but if any edge is removed from E, the resulting graph is disconnected
4. G is connected, and $|E| = |V|$ -1
5. G is acyclic, and $|E| = |V|$ -1
6. G is acyclic, but if any edge is added to E, the resulting graph contains a cycle



Finite Automata

- Finite automata are finite collections of states with transition rules that take you from one state to another.
- Original application was sequential switching circuits, where the "state" was the settings of internal bits.
- Today, several kinds of software can be modeled by FA.

Representing FA

- Simplest representation is often a graph.
  - Nodes = states.
  - Arcs indicate state transitions.
  - Labels on arcs tell what causes the transition.

Finite Automata

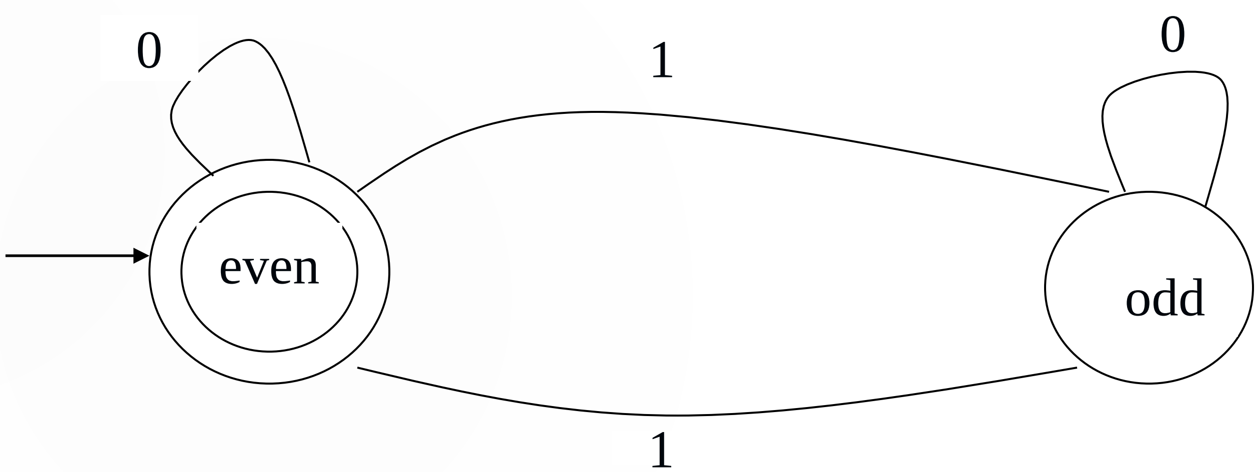
- Has some number of states
- Has a start state and at least one end state
- Accepts input that advances it through its states
- Can be Deterministic (DFA) or Non-Deterministic (NFA)

Finite Automata

- FINITE AUTOMATA IS ALSO KNOWN AS THE FINITE STATE MACHINE
- IT IS A 5-TUPLE {Q, Σ, Δ, Q_0, F}WHERE
  - Q is a finite set of states in the machine
  - Σ is the input alphabet
  - δ is the transition from one state to the next state
  - q_0 is the initial state
  - F is the set of all accepting states

For example: language consist of all strings have even number of 1's

L={11, 011, 101, 110, 0011, 00…011 …}



Finite Automata

For example: language consist of all strings have even number of 1's

- Finite Automaton, M:
- M ={Q, Σ, δ, q_0, F}
  - Q={even, odd}
  - Σ={0,1}
  - δ is described as

	0	1
even	{even}	{odd}
odd	{odd}	{even}

  - q_0 =even
  - F={even}

Example: Recognizing Strings Ending with "ing"