

2021

# Theory of Computation

**Kun-Ta Chuang**

**Department of Computer Science and Information Engineering  
National Cheng Kung University**



# Outline



Course Preliminaries

---



Mathematical Preliminaries and Notation

---



Three Basic Concepts

---

# Introduction

Theory of computation:  
Formal languages  
Automata theory  
Computability  
Complexity

- Formal Languages
  - Abstraction of the general characteristics of programming language
  - Consists of a set of symbols (**string**) and some rules (**grammar**) of formation by which these symbols can be combined into **sentences**

# Introduction

Theory of computation:  
Formal languages  
Automata theory  
Computability  
Complexity

- Automata Theory
  - A question
    - Do you know how a vending machine works? Can you design one?



# Introduction

Theory of computation:  
Formal languages  
Automata theory  
Computability  
Complexity

- Automata Theory

- An example

- How to design a vending machine?

- Use a *finite automaton*!

Assume (for simplicity):

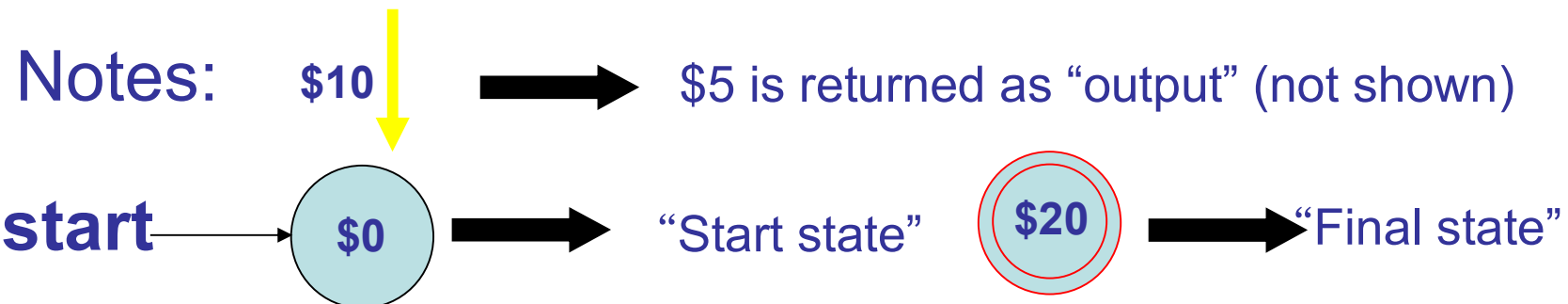
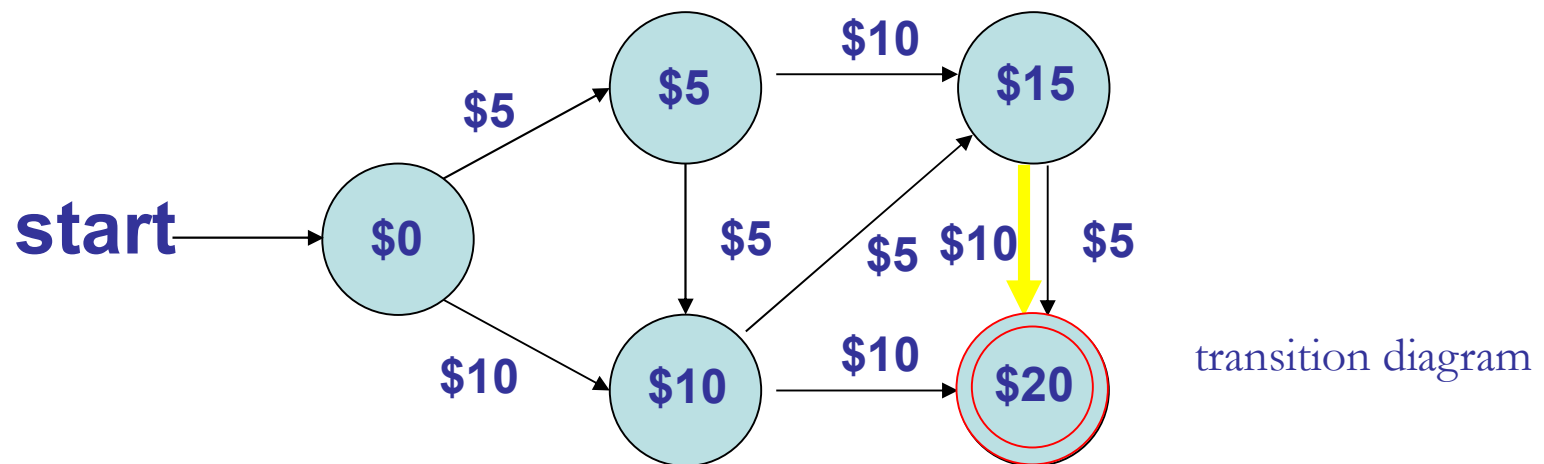
- Only NT 5-dollar and 10-dollar coins are used.
  - Only drinks all of 20 dollars are sold.

# Introduction

Theory of computation:  
Formal languages  
Automata theory  
Computability  
Complexity

- Automata Theory

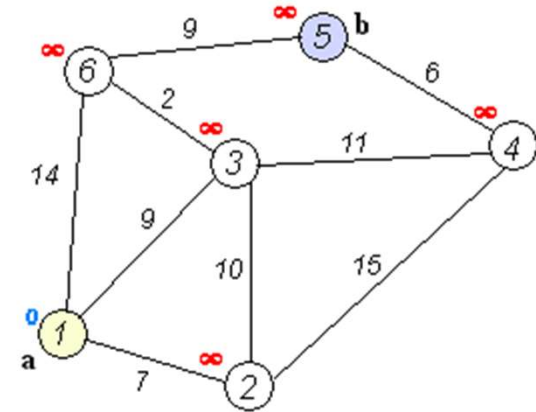
- An example --- need “memory” called “states”



# The Shortest Path Problem

P (Polynomial)

- Given:
  - Directed graph  $G = (V, E)$ 
    - Length  $l_e$  = length of edge  $e = (u, v) \in E$ 
      - Distance; time; cost
      - $l_e \geq 0$
  - Source  $s$
- Goal:
  - Shortest path  $P_v$  from  $s$  to each other node  $v \in V - \{s\}$ 
    - Length of path  $P$ :  $l(P) = \sum_{e \in E} l_e$

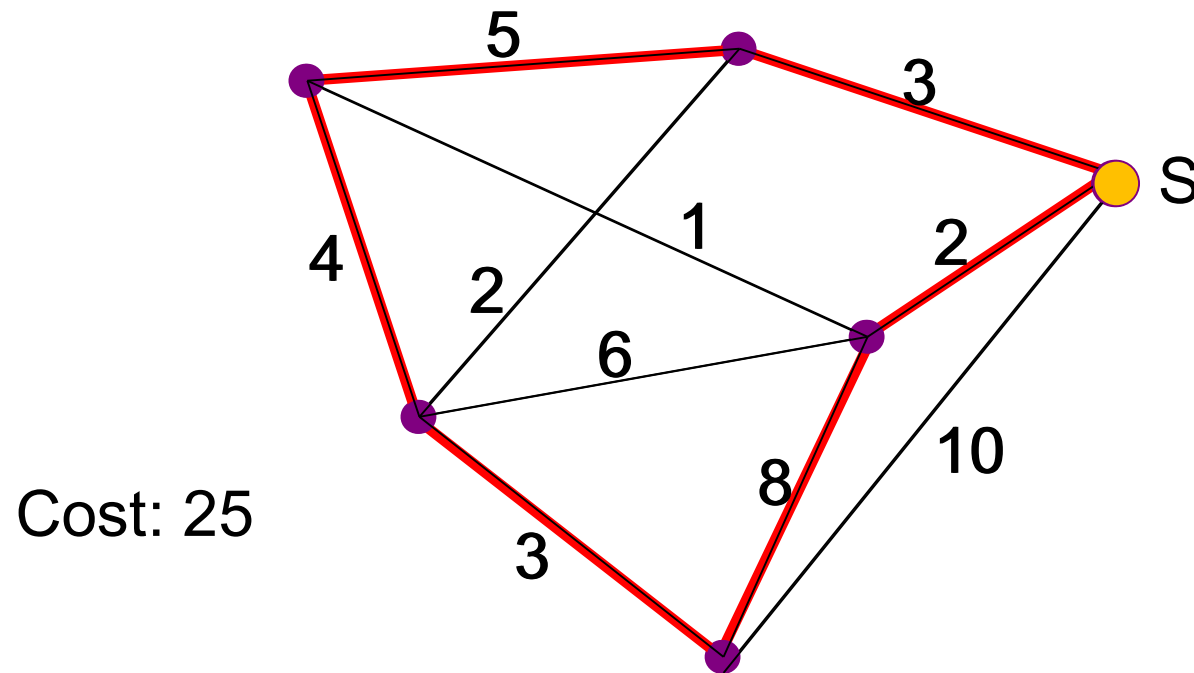


$$\begin{aligned} l(a \rightarrow b) &= l(1 \rightarrow 3 \rightarrow 6 \rightarrow 5) \\ &= 9 + 2 + 9 = 20 \end{aligned}$$

Basic:  $O(|V|^2)$

Fibonacci Heap:  $O(|E| + |V| \log |V|)$

# Example: the Traveling Salesman Problem



What is the least-cost round-trip route that visits each city exactly once and then returns to the starting city?

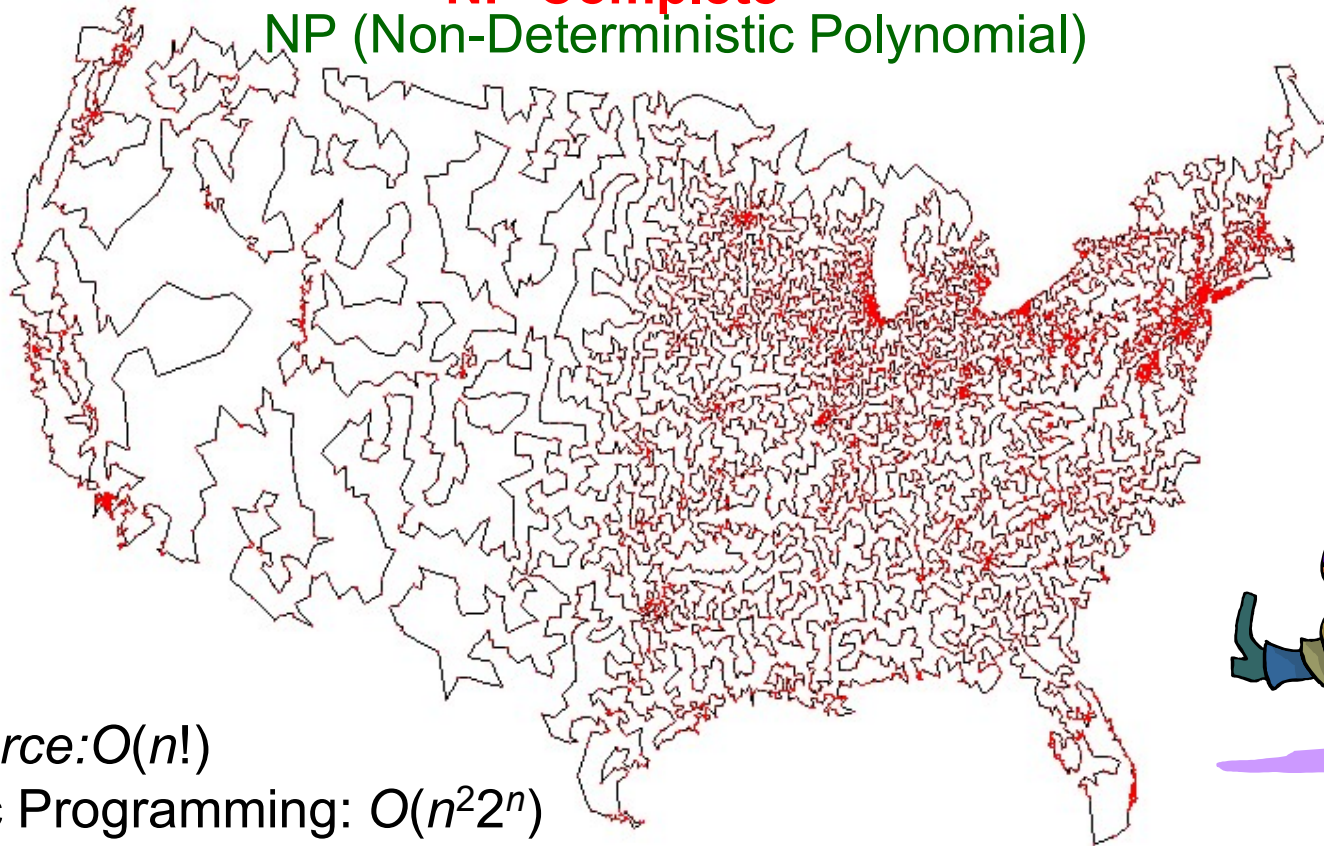


# Traveling Salesman Problem (TSP)

Given a set of cities and that distance between each pair of cities, find the distance of a “minimum route” starts and ends at a given city and visits every city exactly once.

**NP-Complete**

NP (Non-Deterministic Polynomial)



*Brute Force:*  $O(n!)$

*Dynamic Programming:*  $O(n^2 2^n)$

All 13,509 cities in US with a population of at least 500

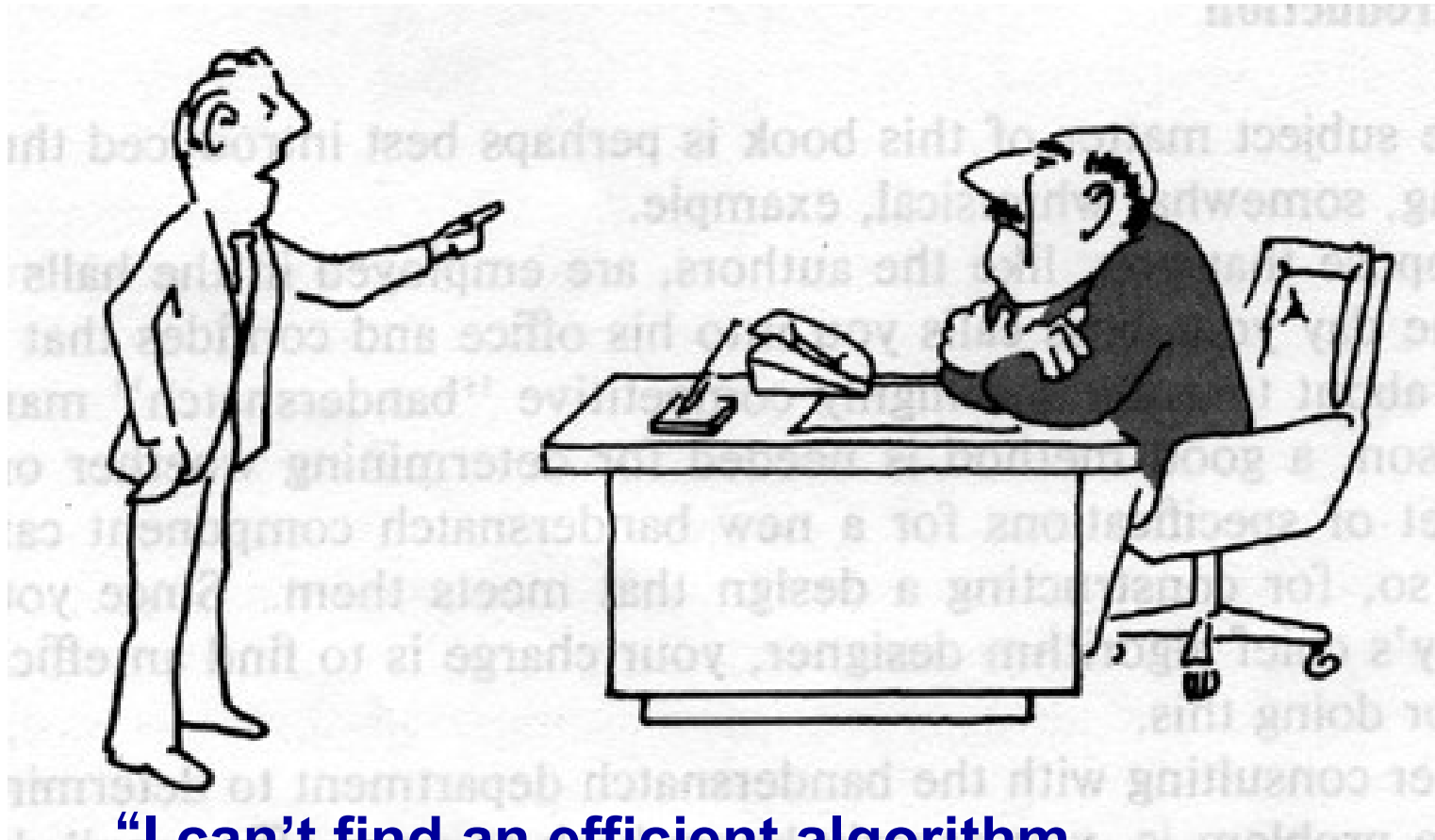
Reference: <http://www.tsp.gatech.edu>

# Coping with a “Tough” Problem: **Trilogy I**



**“I can’t find an efficient algorithm.  
I guess I’m just too dumb.”**

# Coping with a “Tough” Problem: **Trilogy II**



**“I can’t find an efficient algorithm,  
because no such algorithm is possible!”**

# Coping with a “Tough” Problem: **Trilogy III**



**“I can’t find an efficient algorithm,  
but neither can all these famous people.”**

# Fields Related to Theory of Computation

Fields	Related theory
Compiling theory	formal languages
Switching circuit theory	automata theory
Algorithm analysis	computational complexity
Natural language processing	formal languages
Syntactic pattern recognition	formal languages
Programming languages	formal languages
Artificial intelligence	formal languages and automata theory
Neural networks	automata theory

# Question?



# Outline



Course Preliminaries

---



Mathematical Preliminaries and Notation

---



Three Basic Concepts

---



# Mathematical Preliminaries

- Sets
- Functions
- Relations
- Graphs
- Proof Techniques



# SETS

A set is a collection of elements

$$A = \{1, 2, 3\}$$

$$B = \{train, bus, bicycle, airplane\}$$

We write

$$1 \in A \Rightarrow 1 \text{ is an element of the set } A$$

$$ship \notin B \Rightarrow \text{ship is not an element of the set } B$$

# Set Representations

$$C = \{ a, b, c, d, e, f, g, h, i, j, k \}$$

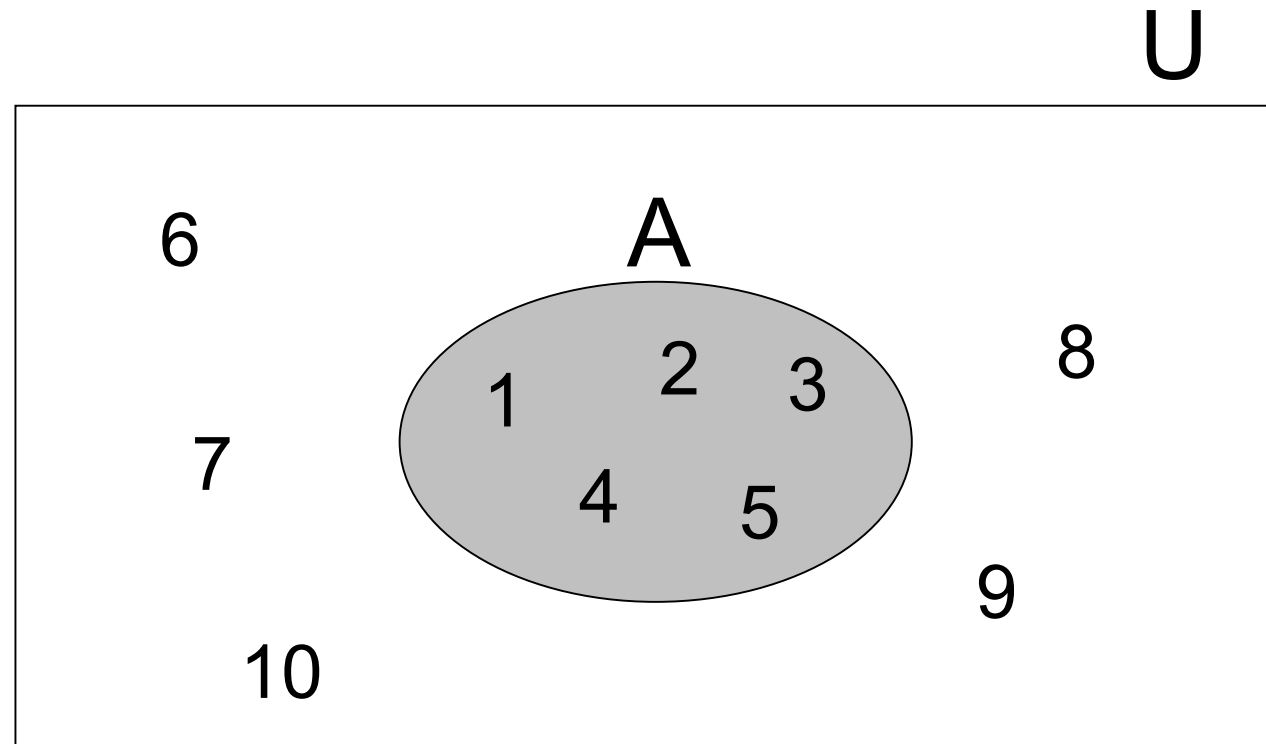
$$C = \{ a, b, \dots, k \} \longrightarrow \textit{finite set}$$

---

$$S = \{ 2, 4, 6, \dots \} \longrightarrow \textit{infinite set}$$

$$\begin{aligned} S &= \{ j : j > 0, \text{ and } j = 2k \text{ for some } k > 0 \} \\ S &= \{ j : j \text{ is nonnegative and even} \} \end{aligned} \quad \left. \vphantom{\begin{aligned} S &= \{ j : j > 0, \text{ and } j = 2k \text{ for some } k > 0 \} \\ S &= \{ j : j \text{ is nonnegative and even} \} \end{aligned}} \right\} \begin{array}{l} \text{Explicit} \\ \text{notation} \end{array}$$

$$A = \{ 1, 2, 3, 4, 5 \}$$



**Universal Set:** all possible elements

$$U = \{ 1, \dots, 10 \}$$

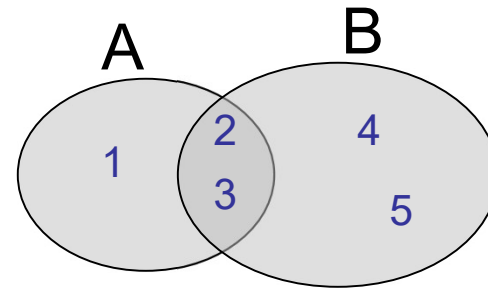
# Set Operations

$$A = \{ 1, 2, 3 \}$$

$$B = \{ 2, 3, 4, 5 \}$$

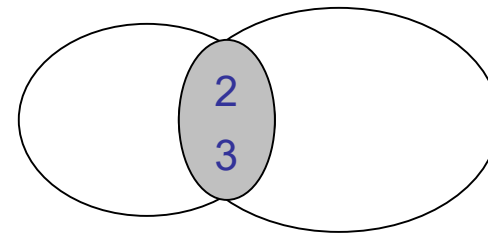
- Union (U)

$$A \cup B = \{ 1, 2, 3, 4, 5 \}$$



- Intersection ( $\cap$ )

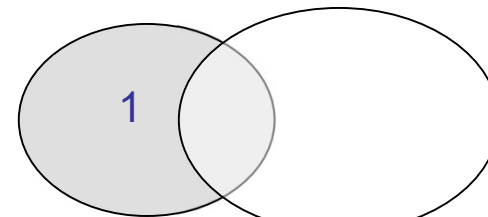
$$A \cap B = \{ 2, 3 \}$$



- Difference (-)

$$A - B = \{ 1 \}$$

$$B - A = \{ 4, 5 \}$$

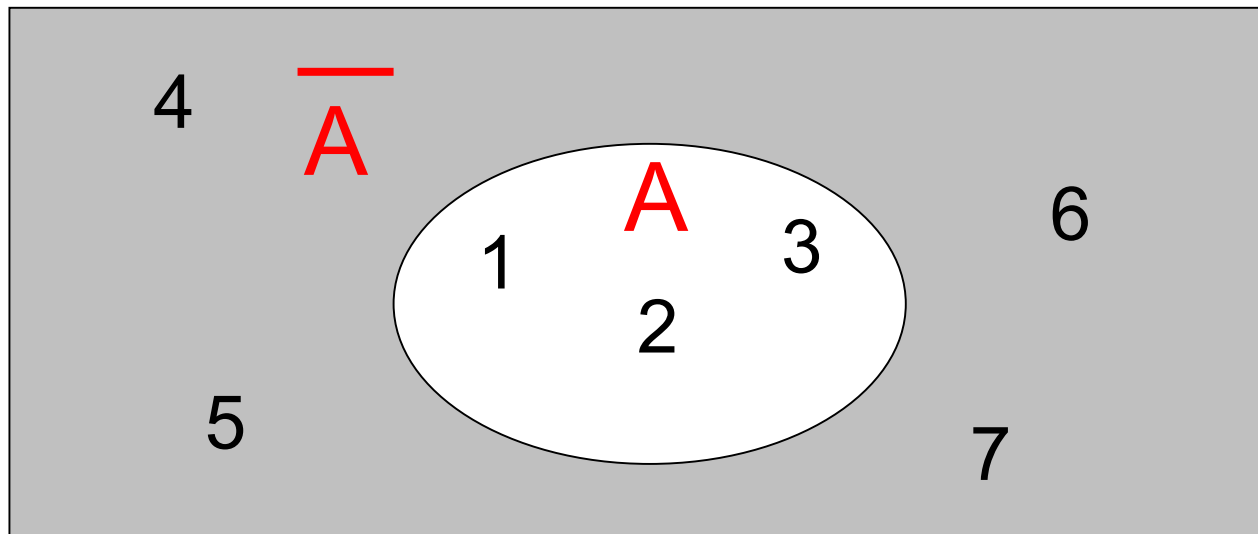


Venn diagrams

- Complement

Universal set =  $\{1, \dots, 7\}$

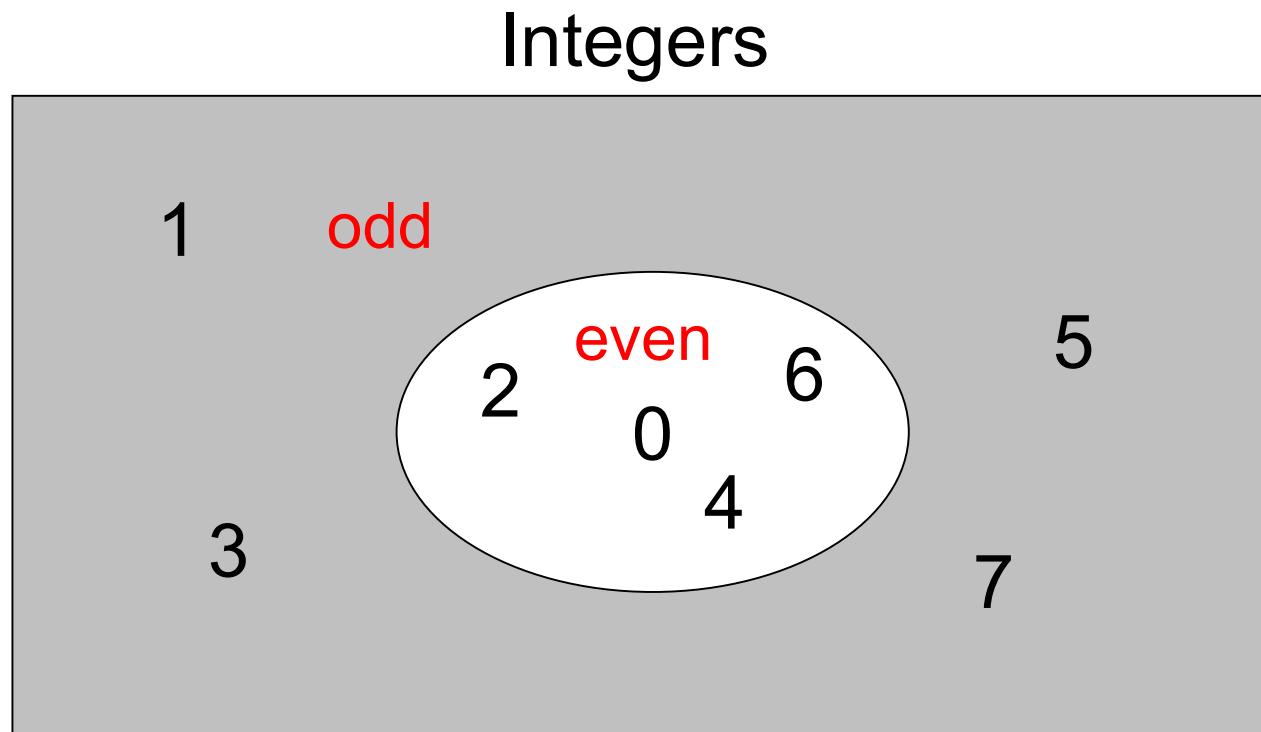
$$A = \{1, 2, 3\} \longrightarrow \overline{A} = \{4, 5, 6, 7\}$$



$$\overline{\overline{A}} = A$$

---

$$\{ \text{even integers} \} = \{ \text{odd integers} \}$$



# DeMorgan's Laws

$$\overline{A \cup B} = \bar{A} \cap \bar{B}$$

$$\overline{A \cap B} = \bar{A} \cup \bar{B}$$

# Empty, Null Set: $\emptyset$

$\emptyset = \{ \}$   The set with no elements

$$S \cup \emptyset = S$$

$$S \cap \emptyset = \emptyset$$

$$S - \emptyset = S$$

$$\emptyset - S = \emptyset$$

$$\overline{\emptyset} = \text{Universal Set}$$



# Subset

$$A = \{ 1, 2, 3 \}$$

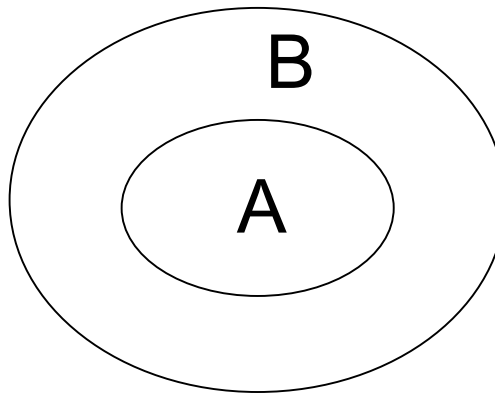
$$B = \{ 1, 2, 3, 4, 5 \}$$

$$A \subseteq B$$

**Subset:** If every element of A is also an element of B

**Proper Subset:** If  $A \subseteq B$ , but B contains an element not in A

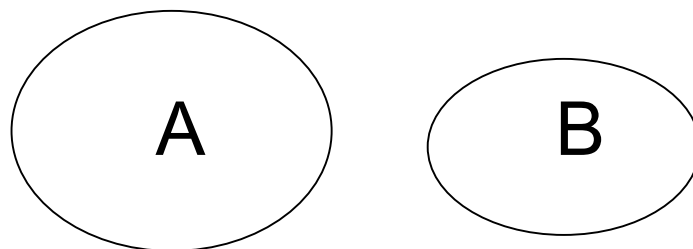
$$A \subset B$$



# Disjoint Sets

$$A = \{ 1, 2, 3 \} \quad B = \{ 5, 6 \}$$

$$A \cap B = \emptyset$$



# Set Cardinality

- For finite sets

$$A = \{ 2, 5, 7 \}$$

$$|A| = 3 \text{ (set size)}$$

# Powersets

A powerset is a set of sets

$$S = \{ a, b, c \}$$

Powerset of  $S$  = the set of all the subsets of  $S$

## Example 1.1

$$2^S = \{ \emptyset, \{a\}, \{b\}, \{c\}, \{a, b\}, \{a, c\}, \{b, c\}, \{a, b, c\} \}$$

Observation: if  $S$  is finite, then  $|2^S| = 2^{|S|}$  (  $8 = 2^3$  )

# Cartesian Product

## Example 1.2

$$A = \{ 2, 4 \}$$

$$B = \{ 2, 3, 5, 6 \}$$

$$A \times B = \{ (2, 2), (2, 3), (2, 5), (2, 6), \\ (4, 2), (4, 3), (4, 5), (4, 6) \}$$

$$|A \times B| = |A| |B|$$

Note that the **order** in which the elements of a pair are written matters

The pair (4, 2) is in  $A \times B$ , but (2, 4) is not

# Partition

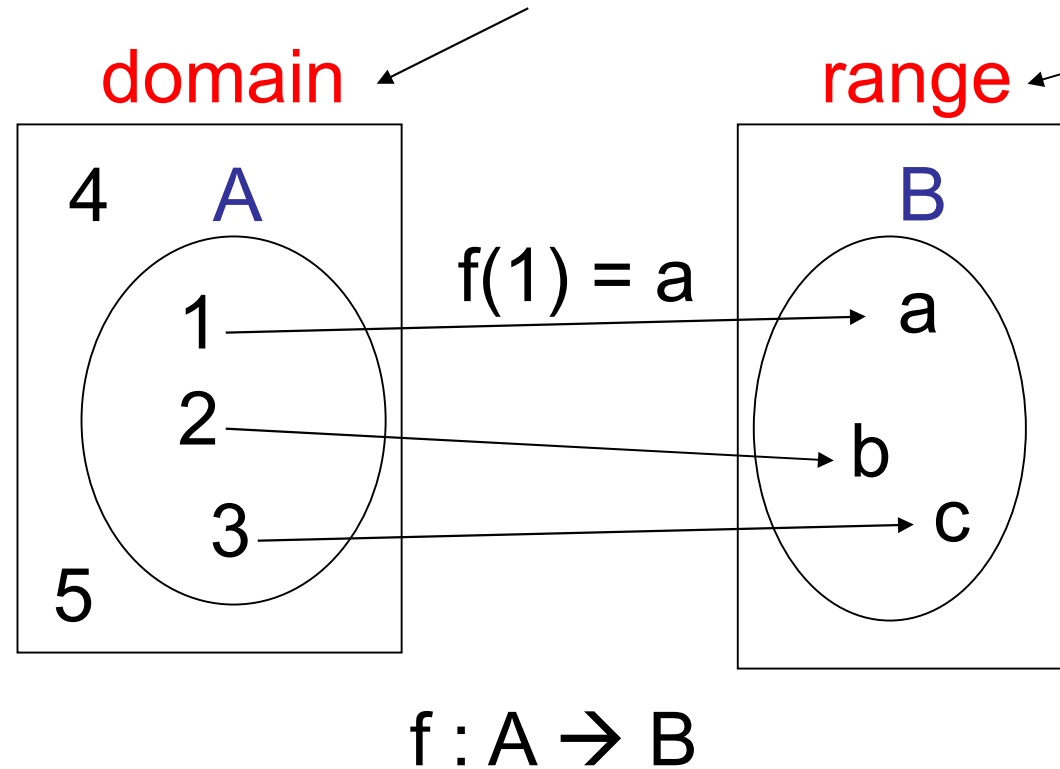
A set can be divided by separating it into a number of subsets. Suppose that  $S_1, S_2, \dots, S_n$  are subsets of a given set  $S$  and that the following holds:

1. The subset  $S_1, S_2, \dots, S_n$  are mutually disjoint;
2.  $S_1 \cup S_2 \cup \dots \cup S_n = S$ ;
3. None of the  $S_i$  is empty.

Then  $S_1, S_2, \dots, S_n$  is called a **partition** of  $S$ .

# FUNCTIONS

Rules that assign to elements of one set a unique element of another set



If  $A = \text{domain}$

then  $f$  is a **total function**

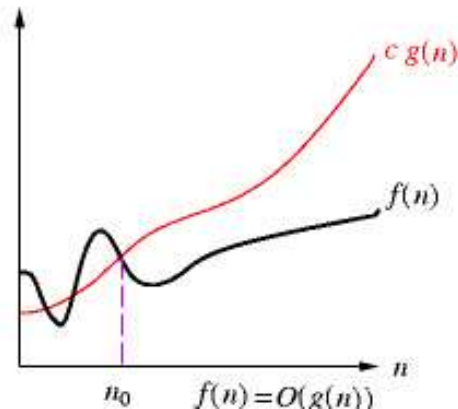
otherwise  $f$  is a **partial function**

# Asymptotic Analysis

## O: Upper Bounding Function

- **Def:**  $f(n) = O(g(n))$  if  $\exists c > 0$  and  $n_0 > 0$  such that  $0 \leq f(n) \leq cg(n)$  for all  $n \geq n_0$ .
- Intuition:  $f(n)$  " $\leq$ "  $g(n)$  when we ignore constant multiples and small values of  $n$ .
- How to show O (Big-Oh) relationships?
  - $f(n) = O(g(n))$  implies that  $\lim_{n \rightarrow \infty} \frac{f(n)}{g(n)} = c$  for some  $c \geq 0$ , if the limit exists.

1.  $3n^2 + n = O(n^2)$ ?
2.  $3n^2 + n = O(n)$ ?
3.  $3n^2 + n = O(n^3)$ ?



■  $O(n) + O(n) = 2O(n)$  ?



# Asymptotic Analysis

## $\Omega$ : Lower Bounding Function

- **Def:**  $f(n) = \Omega(g(n))$  if  $\exists c > 0$  and  $n_0 > 0$  such that  $0 \leq cg(n) \leq f(n)$  for all  $n \geq n_0$ .

- Intuition:  $f(n) \geq g(n)$  when we ignore constant multiples and small values of  $n$ .

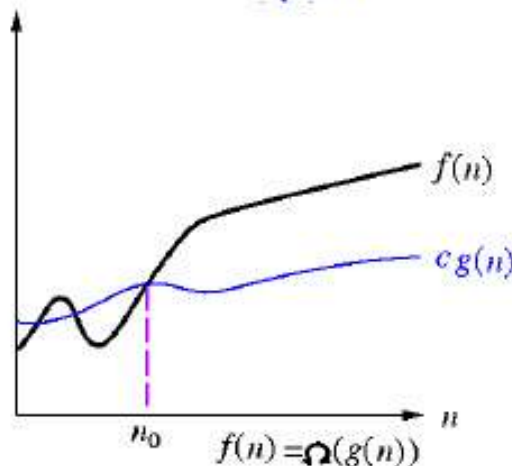
- How to show  $\Omega$  (Big-Omega) relationships?

–  $f(n) = \Omega(g(n))$  implies that  $\lim_{n \rightarrow \infty} \frac{g(n)}{f(n)} = c$  for some  $c \geq 0$  if the limit exists.

1.  $3n^2 + n = \Omega(n^2)$ ?

2.  $3n^2 + n = \Omega(n)$ ?

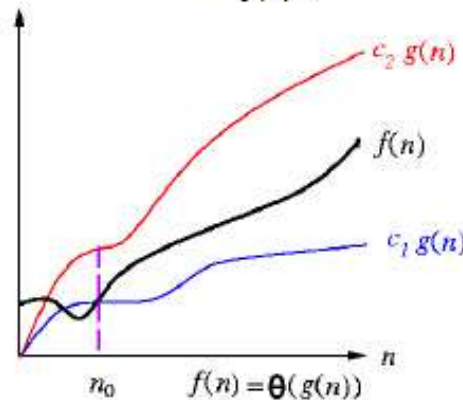
3.  $3n^2 + n = \Omega(n^3)$ ?



# Asymptotic Analysis

## $\theta$ : Tightly Bounding Function

- **Def:**  $f(n) = \theta(g(n))$  if  $\exists c_1, c_2 > 0$  and  $n_0 > 0$  such that  $0 \leq c_1 g(n) \leq f(n) \leq c_2 g(n)$  for all  $n \geq n_0$ .
- Intuition:  $f(n) \approx g(n)$  when we ignore constant multiples and small values of  $n$ .
- How to show  $\theta$  relationships?
  - Show both “big Oh” ( $O$ ) and “Big Omega” ( $\Omega$ ) relationships.
  - $f(n) = \theta(g(n))$  implies that  $\lim_{n \rightarrow \infty} \frac{f(n)}{g(n)} = c$  for some  $c > 0$ , if the limit exists.



1.  $3n^2 + n = \theta(n^2)$ ?

2.  $3n^2 + n = \theta(n)$ ?

3.  $3n^2 + n = \theta(n^3)$ ?

### Example 1.3

$$f(n) = 2n^2 + 3n,$$

$$g(n) = n^3,$$

$$h(n) = 10n^2 + 100.$$

$$f(n) = \theta(g(n)),$$

$$g(n) = \theta(h(n))$$

$$f(n) = \theta(h(n))$$

# RELATIONS

## Relations are more general than functions:

In a **function**, each element of the domains has **exactly one** associated element in the range;

In a **relation**, there may be **several** such elements in the range.

$$R = \{(x_1, y_1), (x_2, y_2), (x_3, y_3), \dots\}$$

$$x_i R y_i$$

e. g. if  $R = '>'$ :  $2 > 1$ ,  $3 > 2$ ,  $3 > 1$

# Equivalence Relations ( $\equiv$ )

- Reflexive:  $x R x$
- Symmetric:  $x R y \longrightarrow y R x$
- Transitive:  $x R y$  and  $y R z \longrightarrow x R z$

Example:  $R \equiv '='$

- $x = x$
- $x = y \longrightarrow y = x$
- $x = y$  and  $y = z \longrightarrow x = z$

## Example 1.4

On the set of nonnegative integers, we can define a relation

$$x \equiv y$$

If and only if

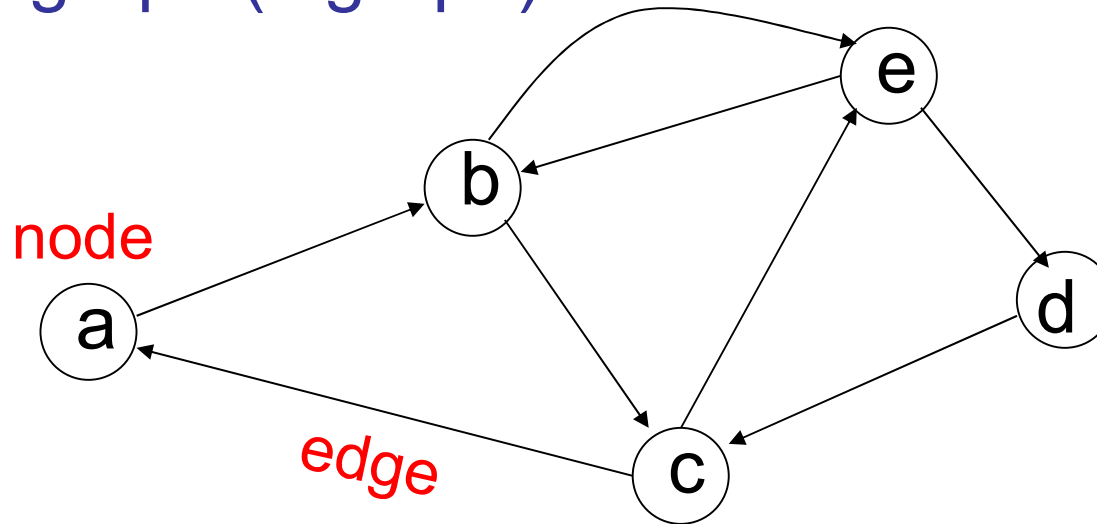
$$x \bmod 3 = y \bmod 3.$$

Then  $2 \equiv 5$ ,  $12 \equiv 0$ , and  $0 \equiv 36$ .

Clearly this is an **equivalence relation**, as it satisfies reflexivity, symmetry, and transitivity.

# GRAPHS

A directed graph (digraph)



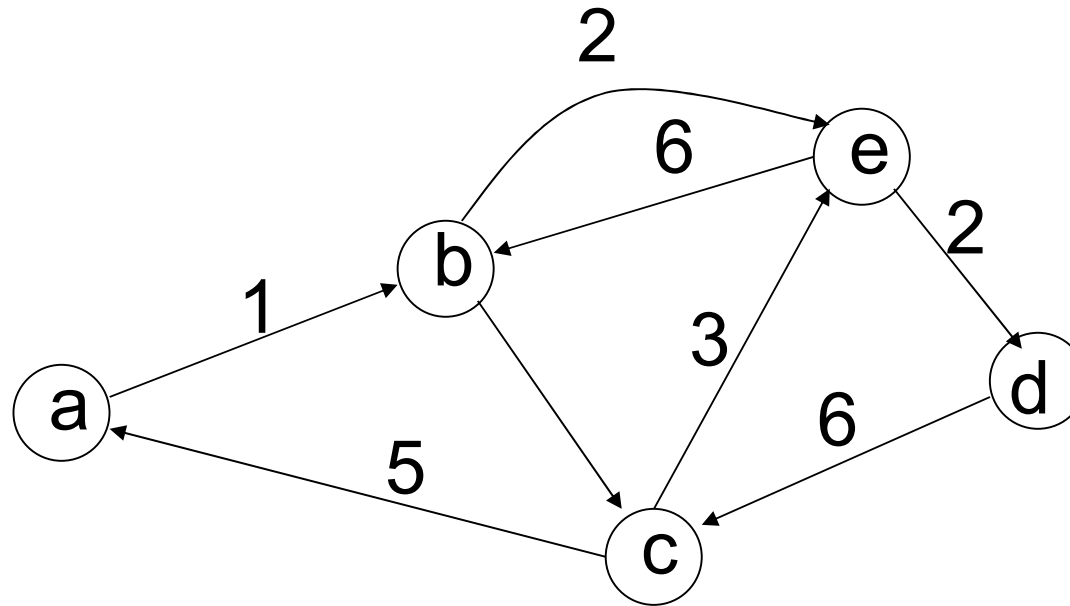
- Nodes (Vertices)

$$V = \{ a, b, c, d, e \}$$

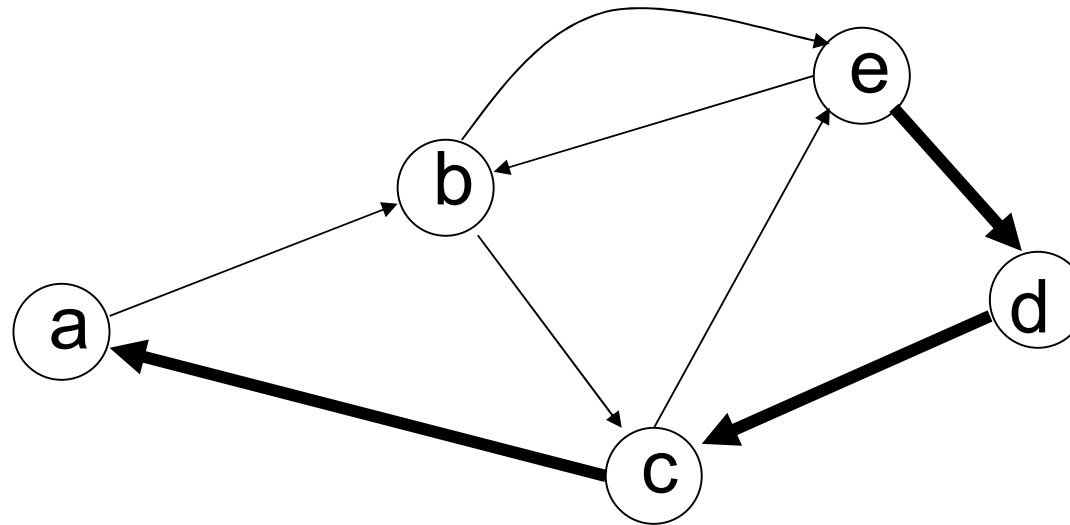
- Edges

$$E = \{ (a,b), (b,c), (b,e), (c,a), (c,e), (d,c), (e,b), (e,d) \}$$

# Labeled Graph



# Walk

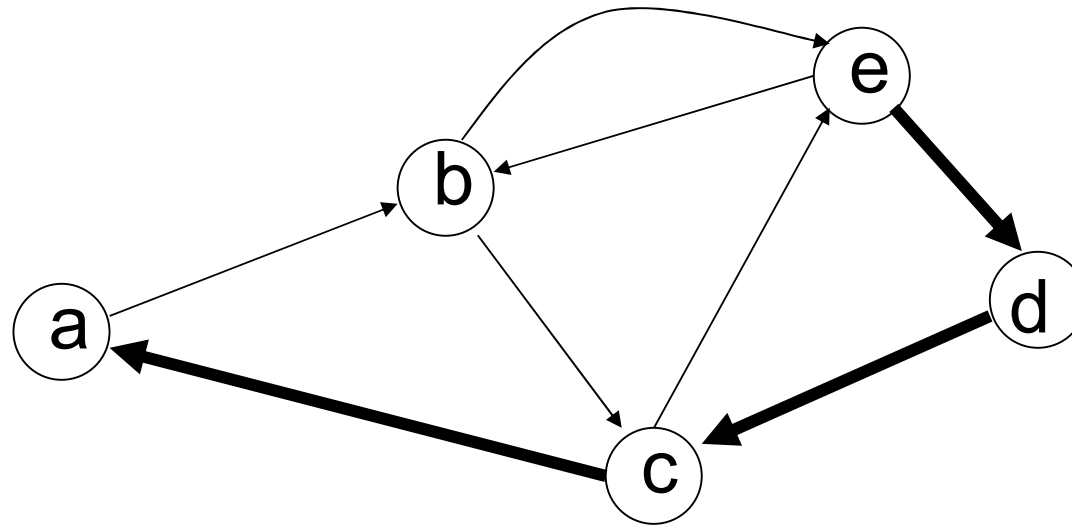


Walk is a sequence of adjacent edges

$(e, d), (d, c), (c, a)$



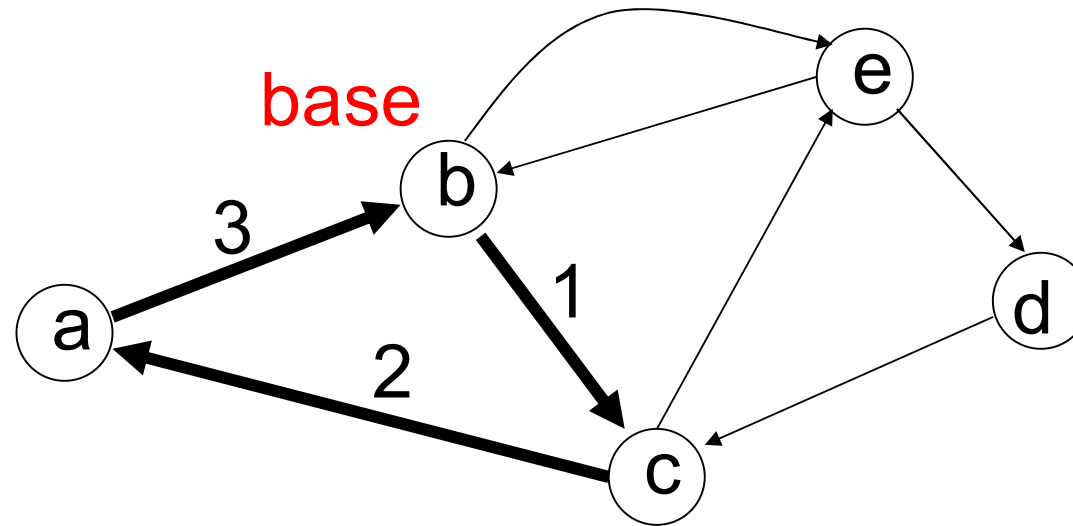
# Path



**Path:** a walk where no edge is repeated

**Simple path:** no node is repeated

# Cycle

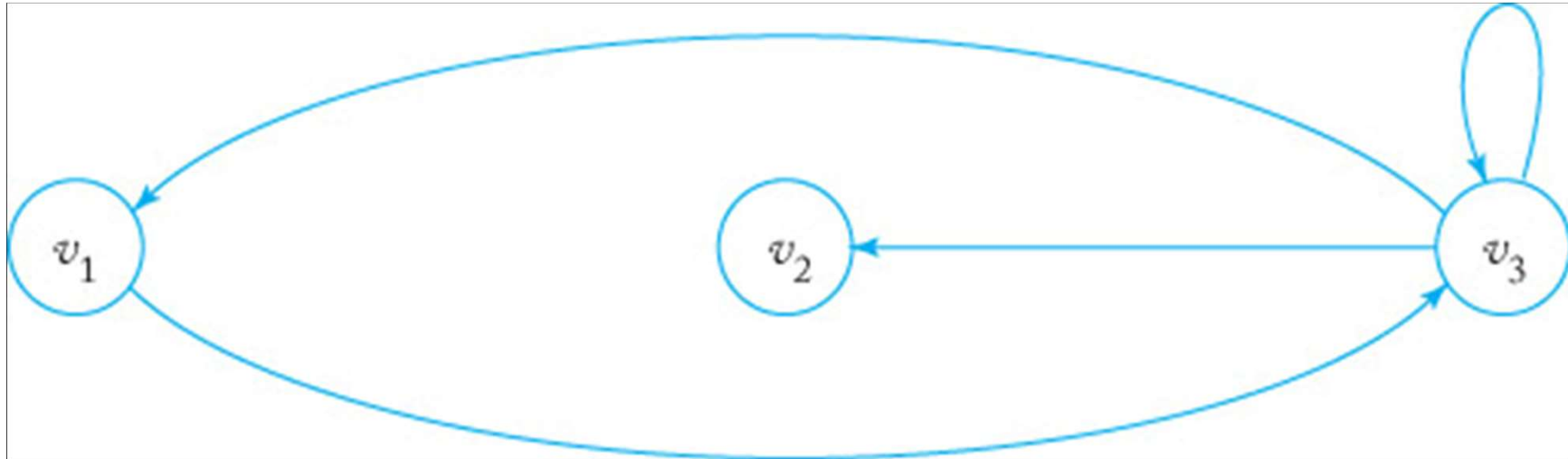


**Cycle:** a walk from a node (base) to itself

**Simple cycle:** only the base node is repeated

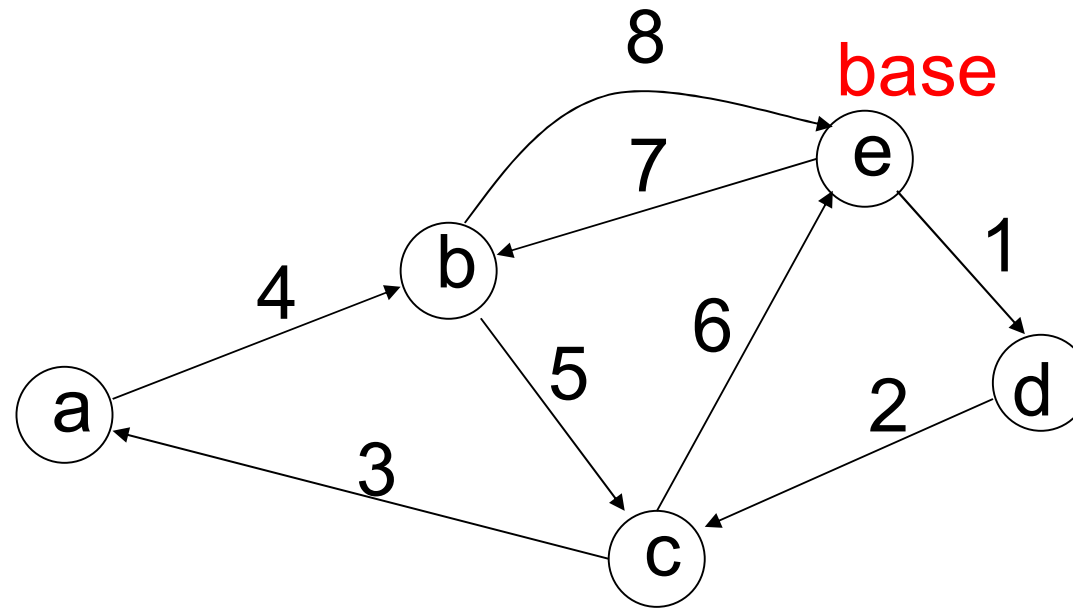
# Loop

An edge from a vertex to itself



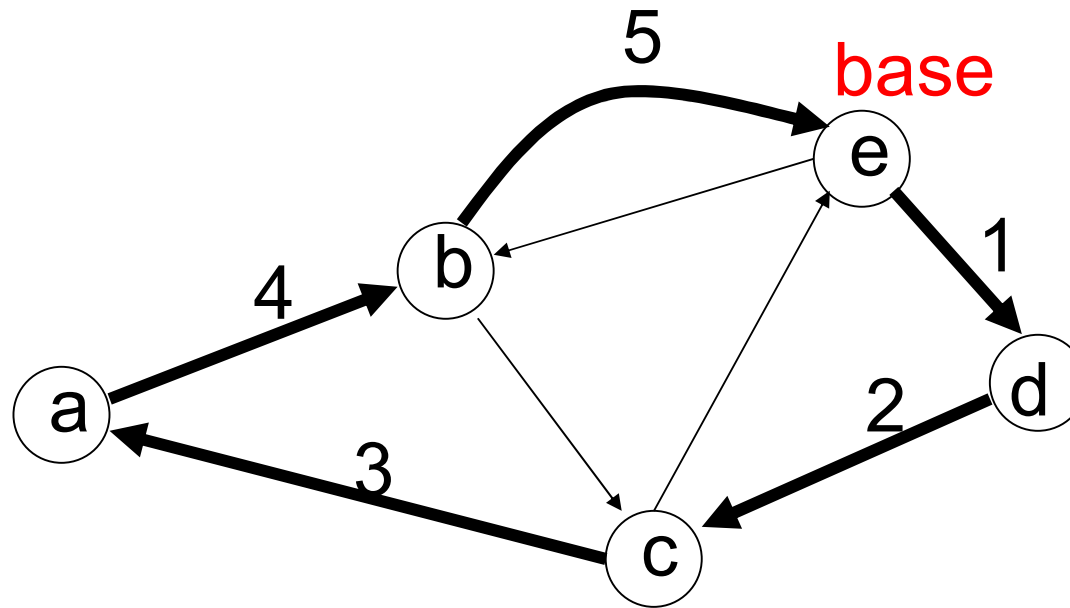
- $(v_1, v_3), (v_3, v_2)$  is a simple path from  $v_1$  to  $v_2$
- $(v_1, v_3), (v_3, v_3), (v_3, v_1)$  is a cycle (not simple one)
- There is a loop on vertex  $v_3$

# Euler Tour



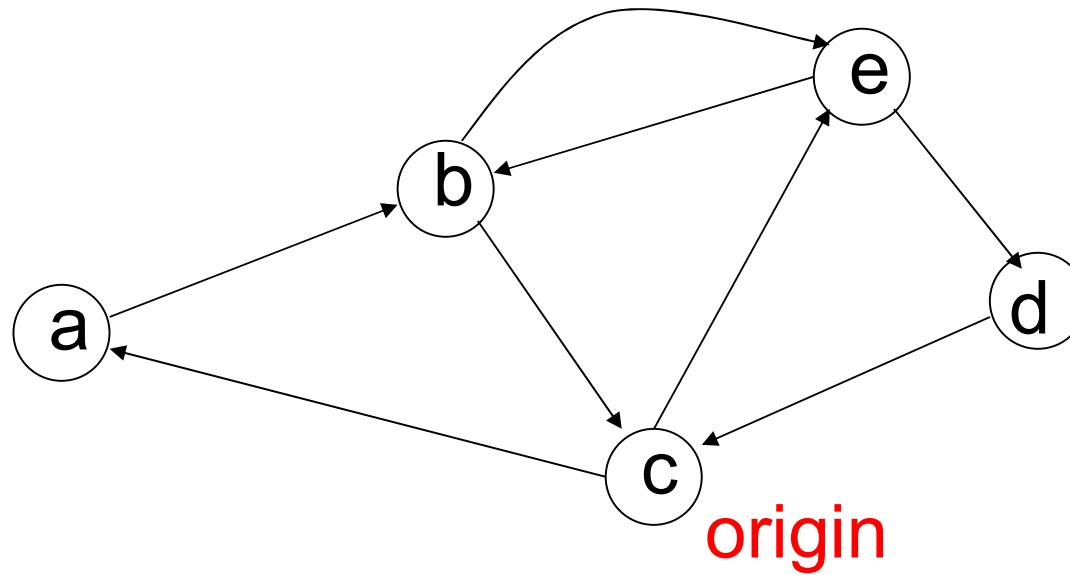
A cycle that contains each edge once

# Hamiltonian Cycle

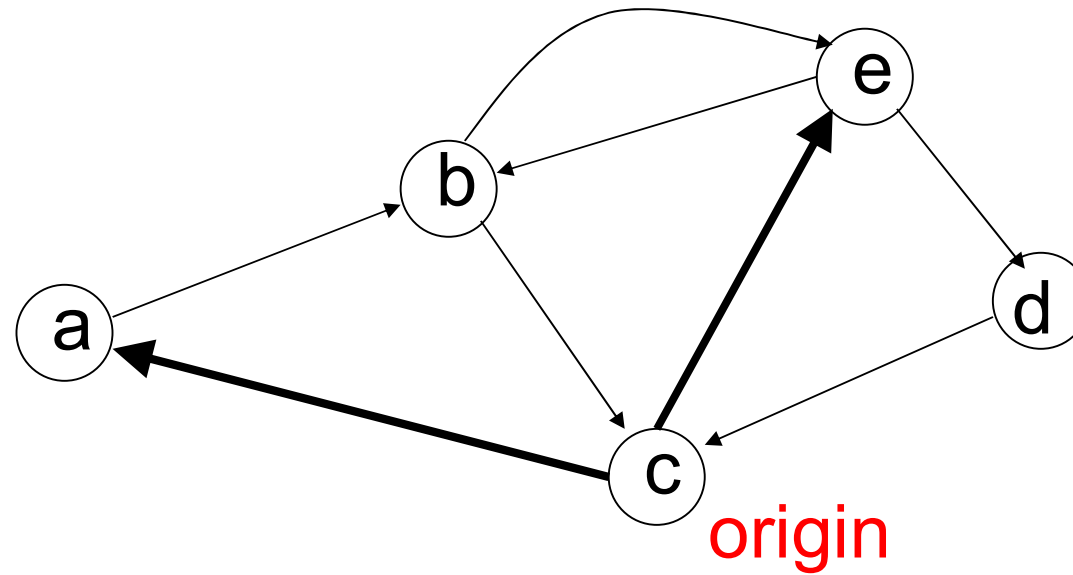


A simple cycle that contains all nodes

# Finding All Simple Paths



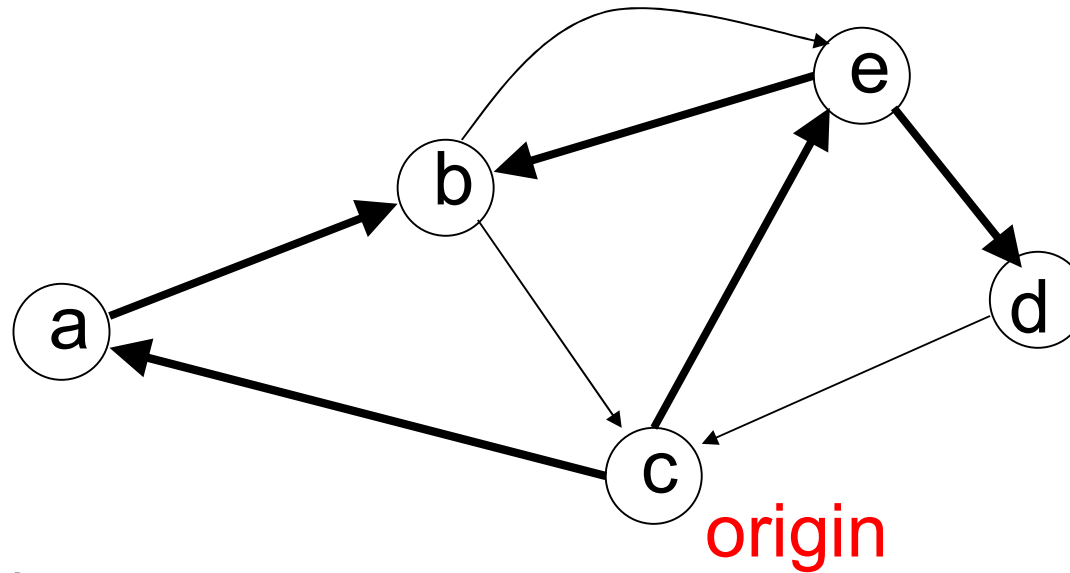
# Step 1



(c, a)

(c, e)

## Step 2



(c, a)

(c, a), (a, b)

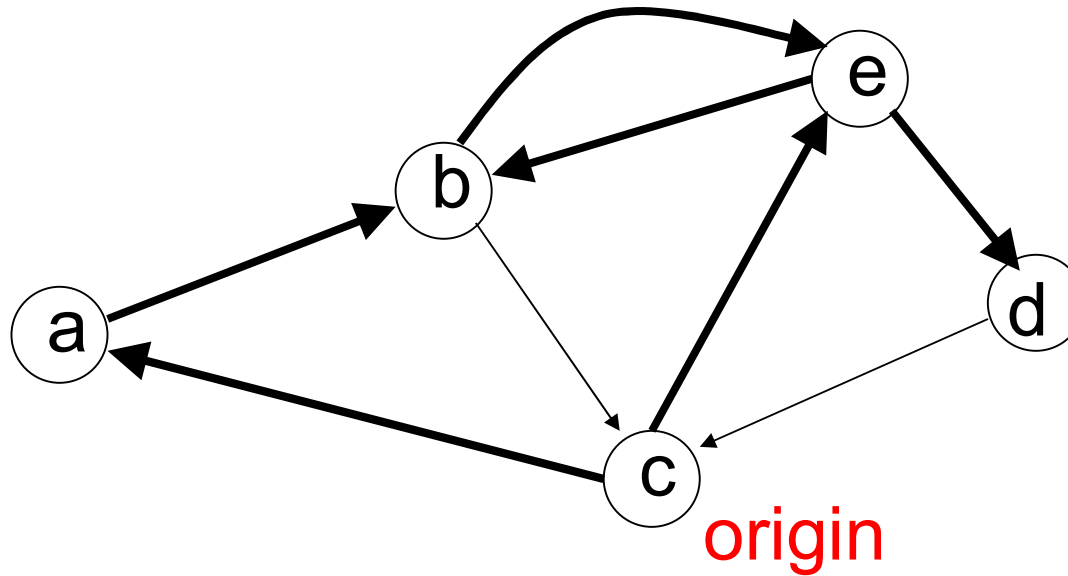
(c, e)

(c, e), (e, b)

(c, e), (e, d)



## Step 3



(c, a)

(c, a), (a, b)

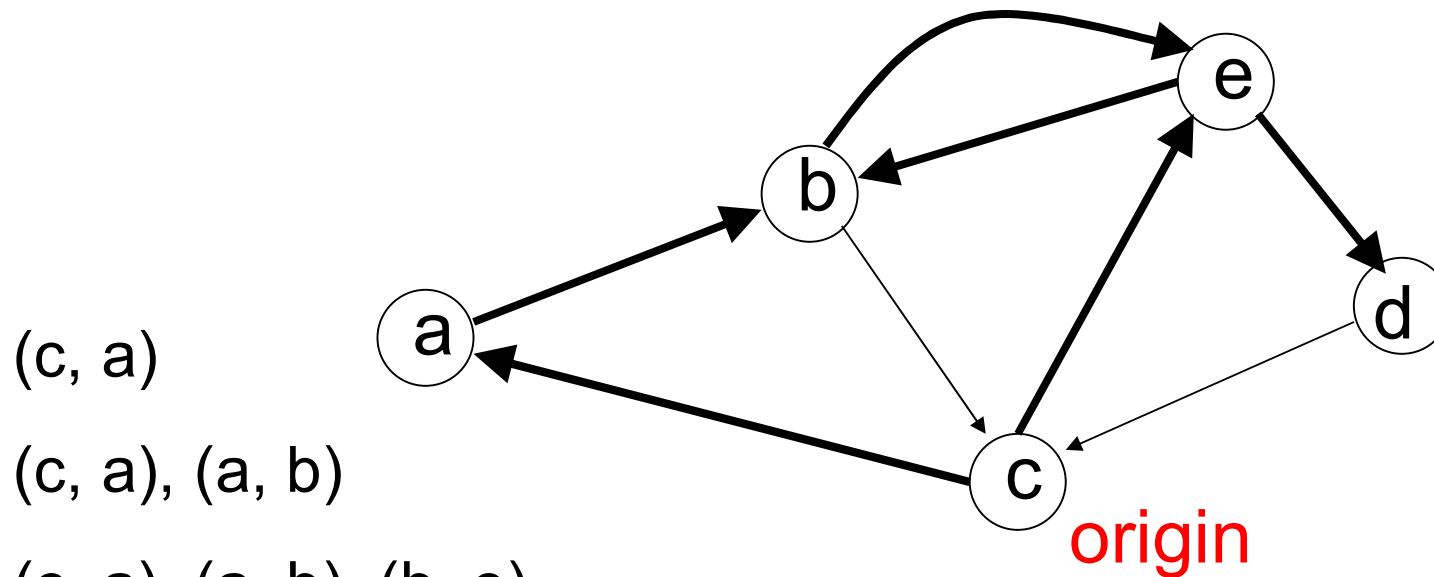
(c, a), (a, b), (b, e)

(c, e)

(c, e), (e, b)

49 (c, e), (e, d)

## Step 4



(c, a)

(c, a), (a, b)

(c, a), (a, b), (b, e)

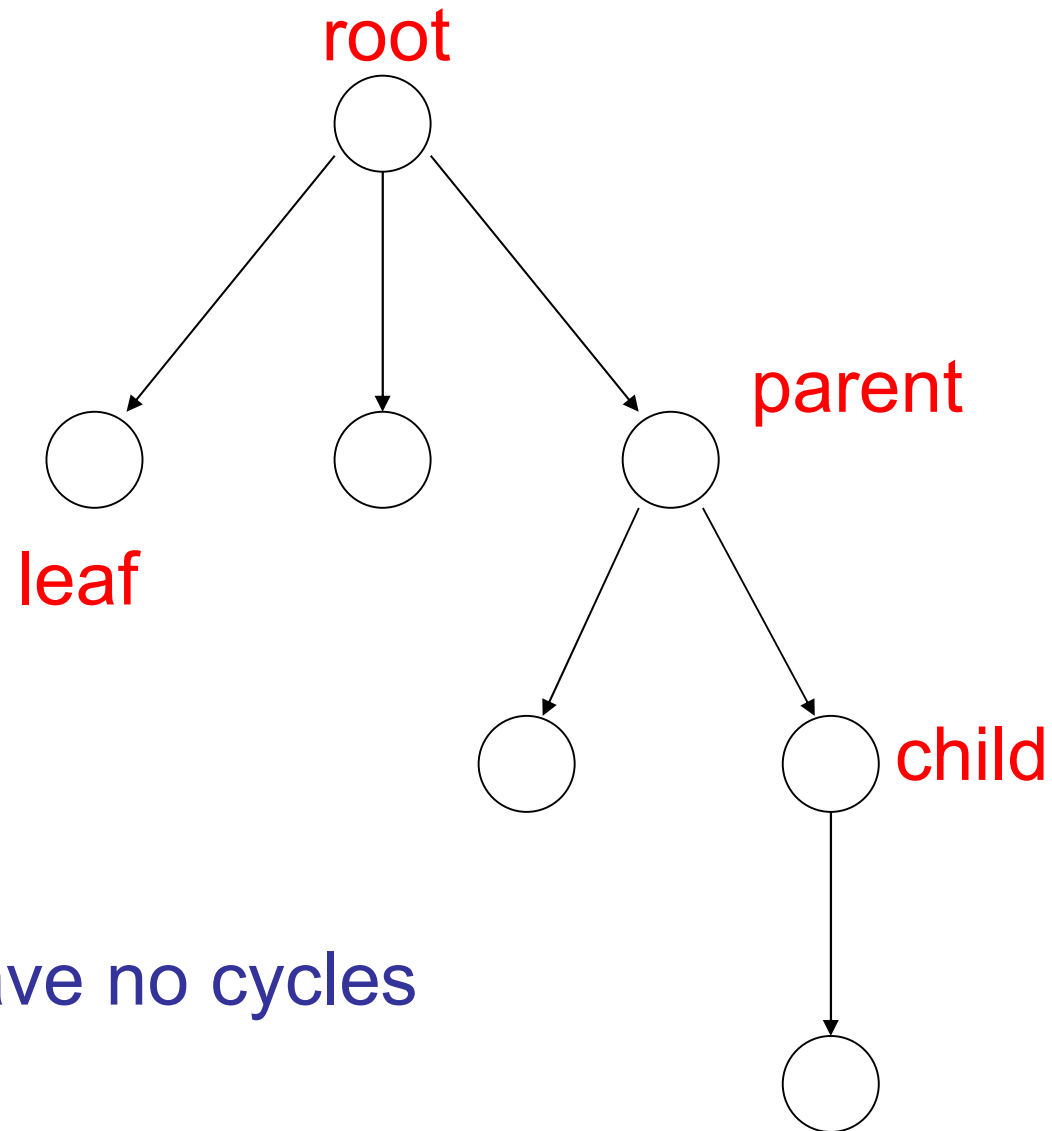
(c, a), (a, b), (b, e), (e, d)

(c, e)

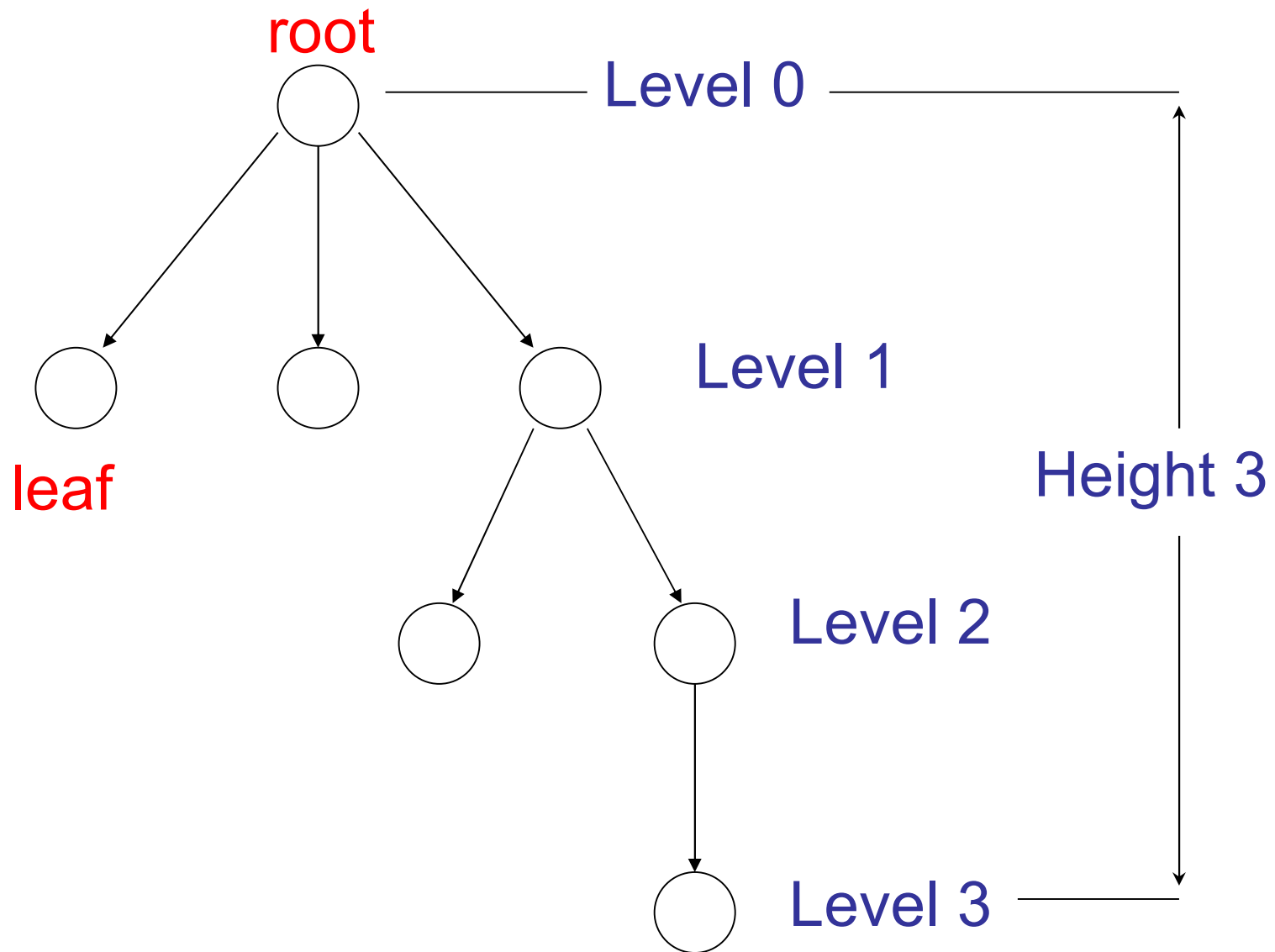
(c, e), (e, b)

(c, e), (e, d)

# Trees



Trees have no cycles



# Proof Techniques

- Direct/Constructive Proof
- Proof by Induction
- Proof by Contradiction

# Direct/Constructive Proof

- If  $X$ , then  $Y$
- Assume  $X$  is true, show directly that  $Y$  is true.  
(e.g.  $X$  = it rains,  $Y$  = sidewalk will wet)
  - Example:
    - For integers  $a, b$ : If  $a$  and  $b$  are odd, then  $ab$  is odd.
    - Given:  $a$  and  $b$  are odd integers
      - There exists integer  $x$  such that  $a = 2x + 1$
      - There exists integer  $y$  such that  $b = 2y + 1$
    - Must prove:  $a$  times  $b$  is also odd
      - There exists integer  $z$  such that  $ab = 2z + 1$

# Direct/Constructive Proof

- Perform the multiplication directly

- $ab = (2x + 1)(2y + 1)$   
 $= 4xy + 2x + 2y + 1$   
 $= 2(2xy + x + y) + 1$

$$\text{So } z = 2xy + x + y$$

Not only did you prove that a  $z$  exists, you constructed an “algorithm” for generating this  $z$ .

This is an example of a constructive proof.

# Induction

We have statements  $P_1, P_2, P_3, \dots$

If we know

- for some  $b$  that  $P_1, P_2, \dots, P_b$  are true
- for any  $k \geq b$  that

$$P_1, P_2, \dots, P_k \text{ imply } P_{k+1}$$

Then

Every  $P_i$  is true



# Proof by Induction

- Inductive basis

Find  $P_1, P_2, \dots, P_b$  which are true

- Inductive hypothesis

Let's assume  $P_1, P_2, \dots, P_k$  are true,  
for any  $k \geq b$

- Inductive step

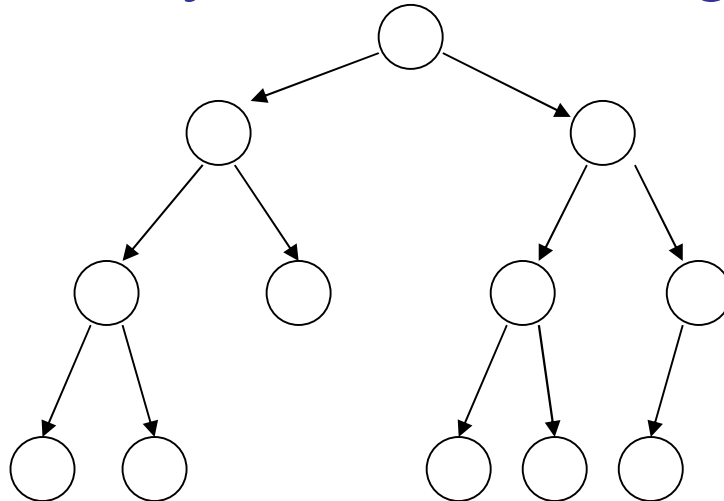
Show that  $P_{k+1}$  is true

## Example 1.5

**Theorem:** A binary tree of height  $n$  has at most  $2^n$  leaves.

**Proof by induction:**

let  $L(i)$  be the maximum number of leaves of any subtree at height  $i$



We want to show:  $L(i) \leq 2^i$

- Inductive basis

$$L(0) = 1 \quad (\text{the root node}) \quad \bigcirc$$

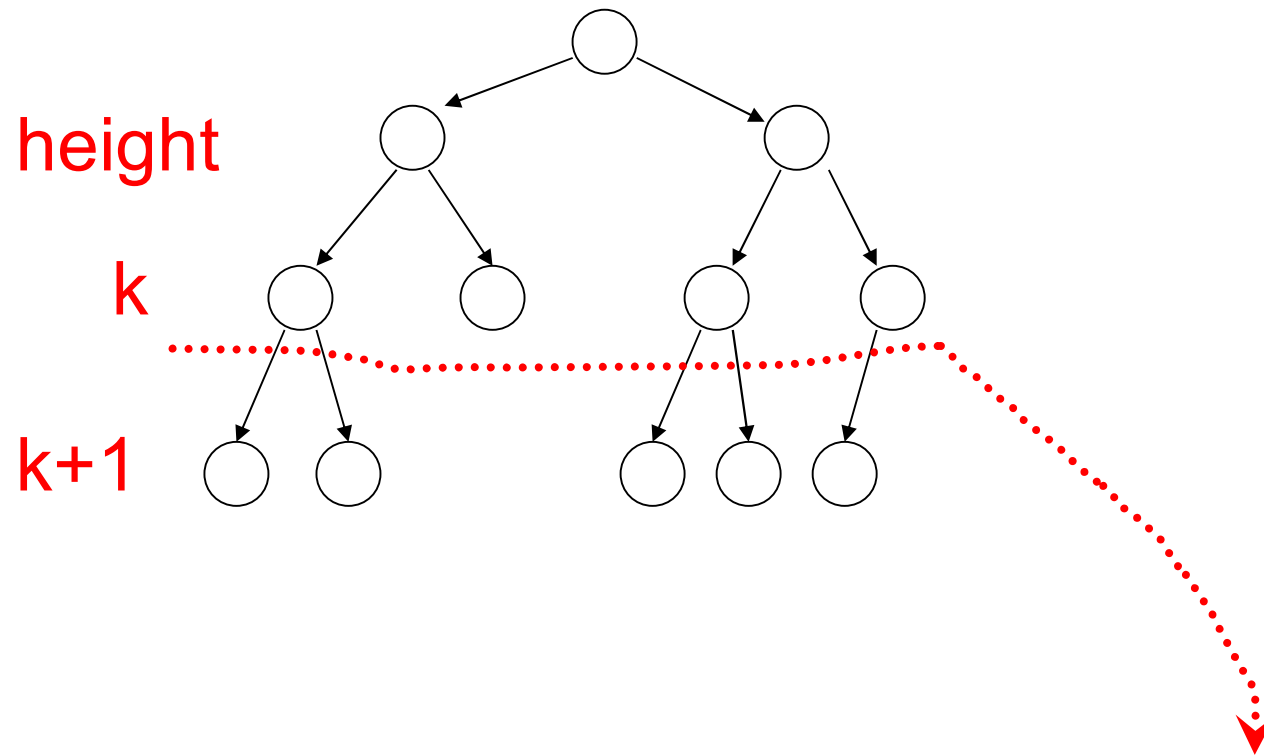
- Inductive hypothesis

Let's assume  $L(i) \leq 2^i$  for all  $i = 0, 1, \dots, k$

- Induction step

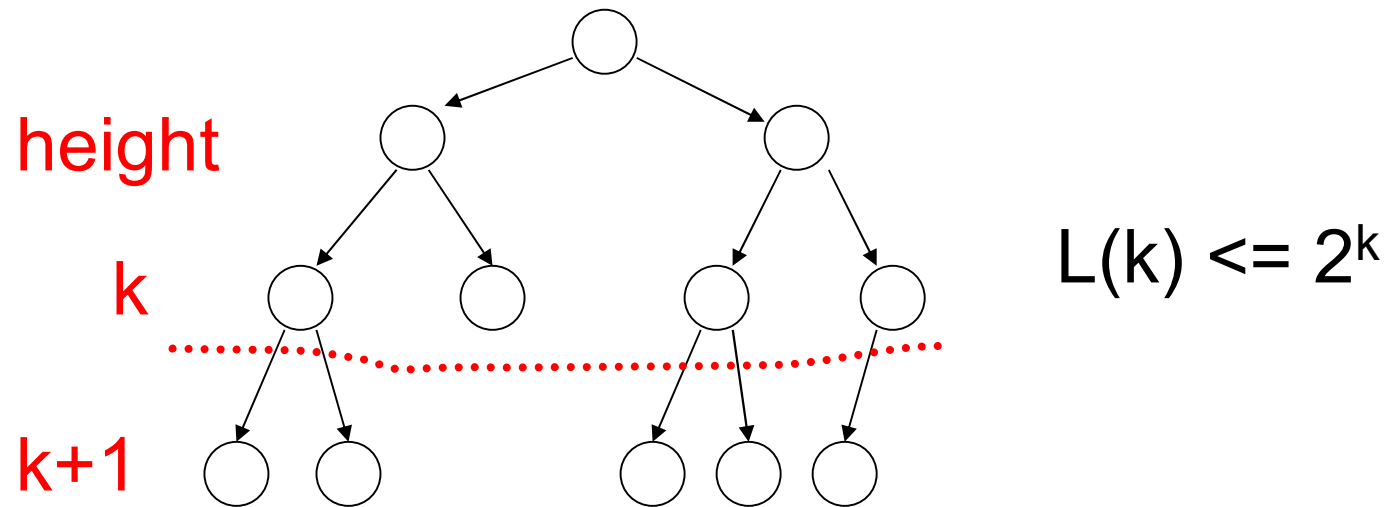
we need to show that  $L(k + 1) \leq 2^{k+1}$

# Induction Step



From Inductive hypothesis:  $L(k) \leq 2^k$

# Induction Step



$$L(k+1) \leq 2 * L(k) \leq 2 * 2^k = 2^{k+1}$$

(we add at most two nodes for every leaf of level k)

# Remark

Recursion is another thing

Example of recursive function:

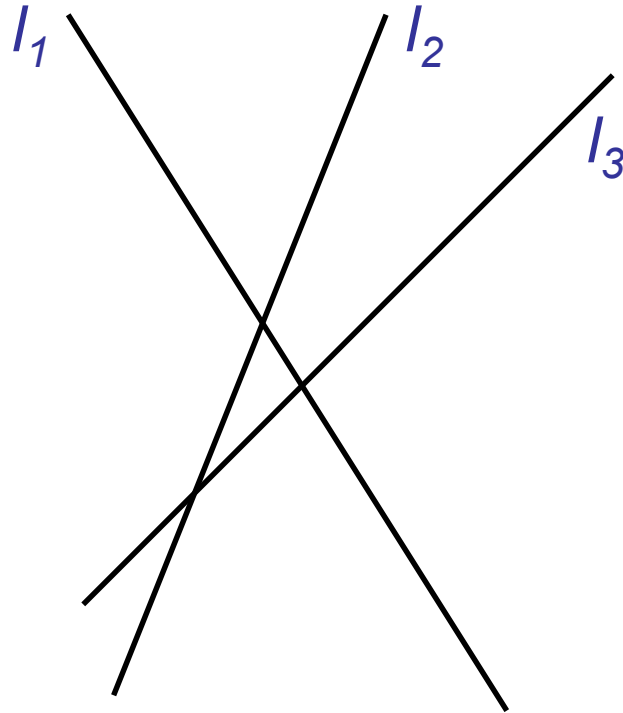
$$f(n) = f(n-1) + f(n-2)$$

$$f(0) = 1, \quad f(1) = 1$$

## Example 1.6

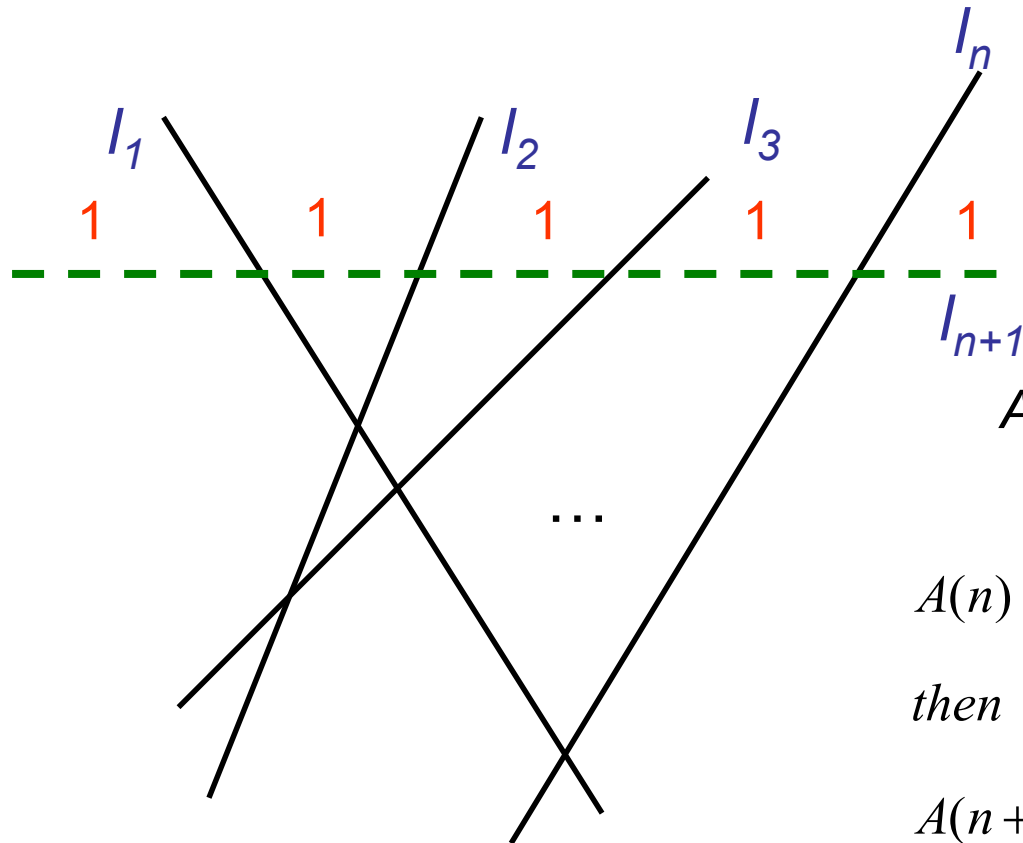
A set  $l_1, l_2, \dots, l_n$  of mutually intersecting straight lines divides the plane into a number of separated regions

1 line  $\rightarrow$  2 regions, 2 lines  $\rightarrow$  4 regions, 3 lines  $\rightarrow$  7 regions



Solve it  
recursively!!

# Example 1.6



Let  $A(n)$  denote the number of regions generated by  $n$  lines

$$A(n+1) = A(n) + n + 1, n = 1, 2, \dots,$$

$$A(1) = 2, A(2) = 4, A(3) = 7, A(4) = 11$$

$$A(n) = \frac{n(n+1)}{2} + 1$$

then

$$A(n+1) = \frac{n(n+1)}{2} + 1 + n + 1 = \frac{(n+1)(n+2)}{2} + 1$$



# Proof by Contradiction

We want to prove that a statement  $P$  is true

- we assume that  $P$  is false
- then we arrive at an incorrect conclusion
- therefore, statement  $P$  must be true

## Example 1.7

**Theorem:**  $\sqrt{2}$  is not rational

**Proof:**

Assume by contradiction that it is rational

$$\sqrt{2} = n/m$$

$n$  and  $m$  have no common factors

We will show that this is impossible

$$\sqrt{2} = n/m \quad \longrightarrow \quad 2 m^2 = n^2$$

Therefore,  $n^2$  is even  $\longrightarrow$   $n$  is even  
 $n = 2 k$

$$2 m^2 = 4k^2 \quad \longrightarrow \quad m^2 = 2k^2 \quad \longrightarrow \quad \begin{array}{l} m \text{ is even} \\ m = 2 p \end{array}$$

Thus,  $m$  and  $n$  have common factor 2

**Contradiction!**

# Outline



Course Preliminaries

---



Mathematical Preliminaries and Notation

---



Three Basic Concepts

---

# Three Basic Concepts

- Languages
- Grammars
- Automata (will discuss in Chap. 2)

A language is a set of **strings**

**String:** A sequence of symbols from the alphabet

- Examples: **“cat”, “dog”, “house”, ...**

- Defined over an **alphabet**:

$$\Sigma = \{a, b, c, \dots, z\}$$

# Alphabets and Strings

$$\Sigma = \{a, b\}$$

*a*

*ab*

*abba*

*baba*

*aaabbbbaabab*

$$u = ab$$

$$v = bbbbaaa$$

$$w = abba$$

# String Operations

$$w = a_1 a_2 \cdots a_n$$

*abba*

$$v = b_1 b_2 \cdots b_m$$

*bbbaaa*

## Concatenation

$$wv = a_1 a_2 \cdots a_n b_1 b_2 \cdots b_m$$

*abbabbbaaa*



$$w = a_1 a_2 \cdots a_n$$

*ababaaabbb*

Reverse

$$w^R = a_n \cdots a_2 a_1$$

*bbbaaababa*

# String Length

$$w = a_1 a_2 \cdots a_n$$

- Length:  $|w| = n$
- Examples:  $|abba| = 4$   
 $|aa| = 2$   
 $|a| = 1$

# Length of Concatenation

$$|uv| = |u| + |v|$$

- Example:  $u = aab$ ,  $|u| = 3$

$$v = abaab, \quad |v| = 5$$

$$|uv| = |aababaab| = 8$$

$$|uv| = |u| + |v| = 3 + 5 = 8$$

# Empty String

- A string with no letters:  $\lambda$

- Observations:  $|\lambda| = 0$

$$\lambda w = w \lambda = w$$

$$\lambda abba = abba \lambda = abba$$

# Substring

- Substring of string:
  - a subsequence of **consecutive** characters

String

abbab

abbab

abbab

abbab

Substring

ab

abba

b

bbab

# Prefix and Suffix

*abbab*  
**Prefixes**      **Suffixes**

$\lambda$

*abbab*

*a*

*bbab*

*ab*

*bab*

*abb*

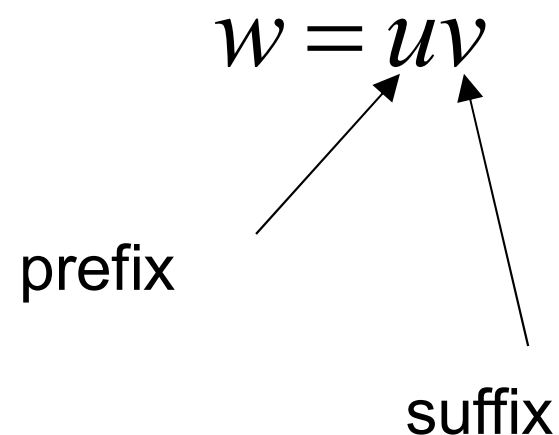
*ab*

*abba*

*b*

*abbab*

$\lambda$



# Another Operation

$$w^n = \underbrace{ww \cdots w}_n$$

- Example:  $(abba)^2 = abbaabba$

- Definition:  $w^0 = \lambda$

$$(abba)^0 = \lambda$$

# The $*$ Operation

$\Sigma^*$  : the set of all possible strings from  
alphabet  $\Sigma$

$$\Sigma = \{a, b\}$$

$$\Sigma^* = \{\lambda, a, b, aa, ab, ba, bb, aaa, aab, \dots\}$$



# The + Operation

$\Sigma^+$  : the set of all possible strings from alphabet  $\Sigma$  except  $\lambda$

$$\Sigma = \{a, b\}$$

$$\Sigma^* = \{\lambda, a, b, aa, ab, ba, bb, aaa, aab, \dots\}$$

$$\Sigma^+ = \Sigma^* - \lambda$$

$$\Sigma^+ = \{a, b, aa, ab, ba, bb, aaa, aab, \dots\}$$

# Languages

A language is any subset of  $\Sigma^*$

Example:  $\Sigma = \{a, b\}$

$$\Sigma^* = \{\lambda, a, b, aa, ab, ba, bb, aaa, \dots\}$$

Languages:  $\{\lambda\}$   
(Finite)

$$\{a, aa, aab\}$$

$$\{\lambda, abba, baba, aa, ab, aaaaaa\}$$

Note that:

Sets

$$\emptyset = \{\} \neq \{\lambda\}$$

Set size

$$|\{\}| = |\emptyset| = 0$$

Set size

$$|\{\lambda\}| = 1$$

String length

$$|\lambda| = 0$$

# Another Example

- An **infinite** language  $L = \{a^n b^n : n \geq 0\}$

$\lambda$   
 $ab$   
 $aabb$   
 $aaaaabbbbbb$

}  $\in L$        $abb \notin L$

# Operations on Languages

- The usual set operations

$$\{a, ab, aaaa\} \cup \{bb, ab\} = \{a, ab, bb, aaaa\}$$

$$\{a, ab, aaaa\} \cap \{bb, ab\} = \{ab\}$$

$$\{a, ab, aaaa\} - \{bb, ab\} = \{a, aaaa\}$$

- Complement:  $\bar{L} = \Sigma^* - L$

$$\overline{\{a, ba\}} = \{\lambda, b, aa, ab, bb, aaa, \dots\}$$

# Reverse

Definition:  $L^R = \{w^R : w \in L\}$

Examples:  $\{ab, aab, baba\}^R = \{ba, baa, abab\}$

$$L = \{a^n b^n : n \geq 0\}$$

$$L^R = \{b^n a^n : n \geq 0\}$$

# Concatenation

Definition:  $L_1 L_2 = \{xy : x \in L_1, y \in L_2\}$

Example:  $\{a, ab, ba\}\{b, aa\}$

$$= \{ab, aaa, abb, abaa, bab, baaa\}$$

# Another Operation

- Definition:  $L^n = \underbrace{LL \cdots L}_n$

$$\{a,b\}^3 = \{a,b\}\{a,b\}\{a,b\} = \\ \{aaa, aab, aba, abb, baa, bab, bba, bbb\}$$

- Special case:  $L^0 = \{\lambda\}$

$$\{a, bba, aaa\}^0 = \{\lambda\}$$



# More Examples

$$L = \{a^n b^n : n \geq 0\}$$

$$L^2 = \{a^n b^n a^m b^m : n, m \geq 0\}$$

$$aabbbaaabb \in L^2$$

Note that  $n$  and  $m$  in the above are unrelated

# Star-Closure (Kleene \*)

- Definition:  $L^* = L^0 \cup L^1 \cup L^2 \dots$

- Example:

$$\{a, bb\}^* = \left\{ \begin{array}{l} \lambda, \\ a, bb, \\ aa, abb, bba, bbbb, \\ aaa, aabb, abba, abbbb, \dots \end{array} \right\}$$

# Positive Closure

- Definition:  $L^+ = L^1 \cup L^2 \cup \dots$   
 $= L^* - \{\lambda\}$

$$\{a, bb\}^+ = \left\{ \begin{array}{l} a, bb, \\ aa, abb, bba, bbbb, \\ aaa, aabb, abba, abbbb, \dots \end{array} \right\}$$

# Grammars

Wikipedia says:

- Languages can be described as a system of symbols and the **grammars (rules)** by which the symbols are manipulated
- Grammar is the study of **rules** governing the use of language.

# Grammars

- Think back to your days of learning English
- Rules for constructing a simple sentence

**Sentence = noun phrase + verb phrase**

- Noun phrase =
  - Name (Joe)
  - Article + noun (the car)
- Verb Phrase =
  - Verb (runs)
  - Verb + prepositional phrase
- Prepositional Phrase =
  - Preposition + noun phrase (from the car)

# Grammars

- Look at the sentence. Is this grammatically correct?

Joe runs from the car.

Sentence = noun phrase + verb phrase  
= noun + verb phrase  
= Name + verb phrase  
= Joe + verb phrase  
= Joe + verb + prepositional phrase  
= Joe + verb + preposition + noun phrase  
= Joe + verb + from + noun phrase  
= Joe + verb + from + article + noun  
= Joe + verb + from + article + noun  
= Joe + verb + from + the + car  
= Joe + runs + from + the + car

Valid sentence!

# Definition 1.1

- A grammar  $G$  is defined as a 4-tuple:

$$G = (V, T, S, P)$$

where

- $V$  is a finite set of variables
- $T$  is a finite set of terminals
- $S \in V$ , called start variable
- $P$  is a finite set of production rules

Ex:

$$G = (\{S\}, \{a, b\}, S, P)$$

$$P: S \rightarrow aSb, \\ S \rightarrow \lambda$$

# Grammars

$$G = (V, T, S, P)$$

- Let's formalize this a bit:

**Production rules**  $(x \rightarrow y)$  where  $\begin{matrix} x \in (V, T)^+ \\ y \in (V, T)^* \end{matrix}$

They specify how the grammar transforms one string into another

- We say that  $\gamma$  can be derived from  $\alpha$  in one step:

$A \rightarrow \beta$  is a production rule

$$\alpha = \alpha_1 A \alpha_2$$

$$\gamma = \alpha_1 \beta \alpha_2$$

$$\alpha \Rightarrow \gamma$$

- We write  $\alpha \xRightarrow{*} \gamma$  if  $\gamma$  can be derived from  $\alpha$  (or say  $\alpha$  derives  $\gamma$ ) in zero or more steps.



# Definition 1.2

- Let  $G = (V, T, S, P)$  be a grammar. Then the set

$$L(G) = \{w \in T^*: S \xRightarrow{*} w\}$$

is the **language** generated by  $G$

- If  $w \in L(G)$ , then the sequence

$$S \Rightarrow w_1 \Rightarrow w_2 \Rightarrow \dots \Rightarrow w_n \Rightarrow w$$

is a derivation of the sentence  $w$ .

- $S, w_1, w_2, \dots, w_n$  are called **sentential forms**

# Example 1.11

$$G = (V, T, S, P)$$

Consider the grammar:

$$G = (\{S\}, \{a, b\}, S, P)$$

$L(G)$ ?

With  $P$  given by

$$L(G) = \{a^n b^n : n \geq 0\}$$

$$S \rightarrow aSb,$$

$$S \rightarrow \lambda$$

Then

$$S \Rightarrow aSb \Rightarrow aaSbb \Rightarrow aabb,$$

So we can write

$$S \xRightarrow{*} aabb$$

- String  $aabb$  is a sentence in the language generated by  $G$
- $aaSbb$  is a **sentential form**

# Example 1.12

$$G = (V, T, S, P)$$

Find a grammar that generates

$$L = \{a^n b^{n+1} : n \geq 0\}$$

Previous example

$G = (\{S\}, \{a, b\}, S, P)$  with  $P: S \rightarrow aSb, S \rightarrow \lambda$

All we need to do is generate an extra b

$G = (\{S, A\}, \{a, b\}, S, P)$ , with productions

$$S \rightarrow Ab,$$

$$A \rightarrow aAb,$$

$$A \rightarrow \lambda$$

# Example 1.13

$$G = (V, T, S, P)$$

Consider the grammar:

$$G = (\{S\}, \{a, b\}, S, P)$$

With P given by

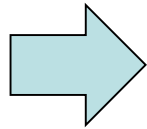
$$S \rightarrow SS,$$

$$S \rightarrow \lambda,$$

$$S \rightarrow aSb,$$

$$S \rightarrow bSa,$$

$L(G)?$



Take  $\Sigma = \{a, b\}$ , and let  $n_a(w)$  and  $n_b(w)$  denote the number of a's and b's in the string  $w$

$$L = \{w: n_a(w) = n_b(w)\}$$

Does this grammar indeed generate the language?

Proof by induction!!

Assume that all  $w \in L$  with  $|w| \leq 2n$  can be derived with  $G$

For  $n = 1$ , trivial

# Example 1.13

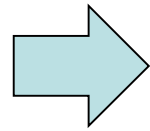
$$G = (V, T, S, P)$$

$$S \rightarrow SS,$$

$$S \rightarrow \lambda,$$

$$S \rightarrow aSb,$$

$$S \rightarrow bSa,$$



Take  $\Sigma = \{a, b\}$ , and let  $n_a(w)$  and  $n_b(w)$  denote the number of a's and b's in the string  $w$

$$L = \{w: n_a(w) = n_b(w)\}$$

Assume that all  $w \in L$  with  $|w| \leq 2n$  can be derived with  $G$

Take any  $w \in L$  of length  $2n+2$ .

If  $w = aw_1b$ , then  $w_1$  is in  $L$ , and  $|w_1| = 2n$ . By assumption,

$$S \xRightarrow{*} w_1$$

Then

$$S \Rightarrow aSb \xRightarrow{*} aw_1b = w \quad (\text{so is } bSa)$$

Else

$$S \Rightarrow SS \xRightarrow{*} w_1S \xRightarrow{*} w_1w_2 = w$$

# Equivalent of Grammars

- Two grammars  $G_1$  and  $G_2$  are **equivalent** if they generate the same language ( $L(G_1) = L(G_2)$ )
- Example 1.14
- $G_1 = (\{S\}, \{a, b\}, S, P_1)$  with  $P_1$ :  
 $S \rightarrow aSb, S \rightarrow \lambda$
- $G_2 = (\{S, A\}, \{a, b\}, S, P_2)$  with  $P_2$ :  
 $S \rightarrow aAb \mid \lambda, A \rightarrow aAb \mid \lambda$

# Questions?