2019

Theory of Computation

Kun-Ta Chuang
Department of Computer Science and Information Engineering
National Cheng Kung University

Outline

1. Course Preliminaries
2. Mathematical Preliminaries and Notation
3. Three Basic Concepts

Introduction

- 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

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

Introduction

- Automata Theory
  - An example

    How to design a vending machine?

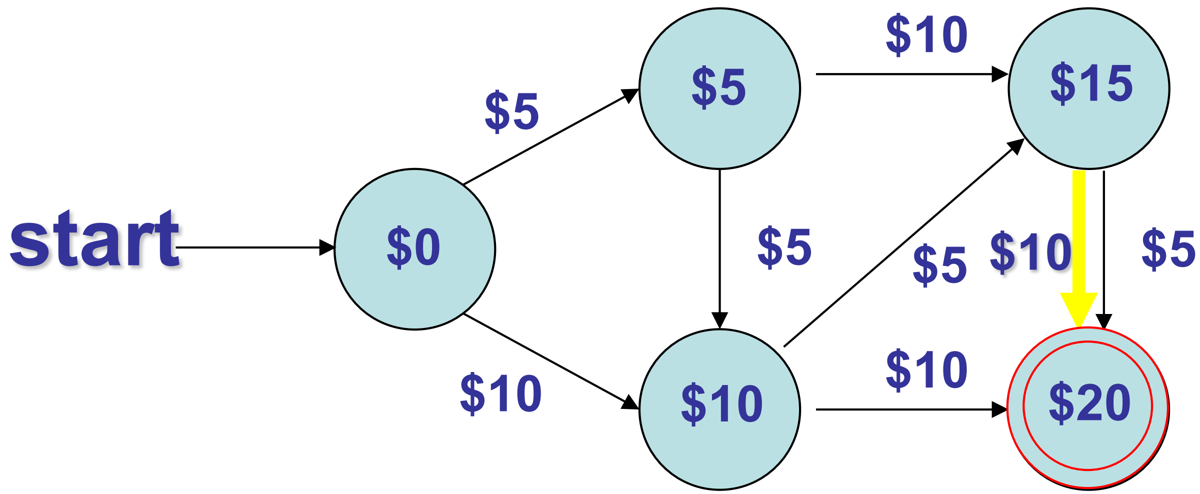
    → *Use a finite automaton!*

Assume (for simplicity):

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

Introduction

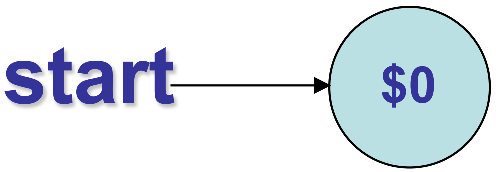
- Automata Theory
  - An example --- need "memory" called "states"



transition diagram

Notes:

$5 is returned as "output" (not shown)



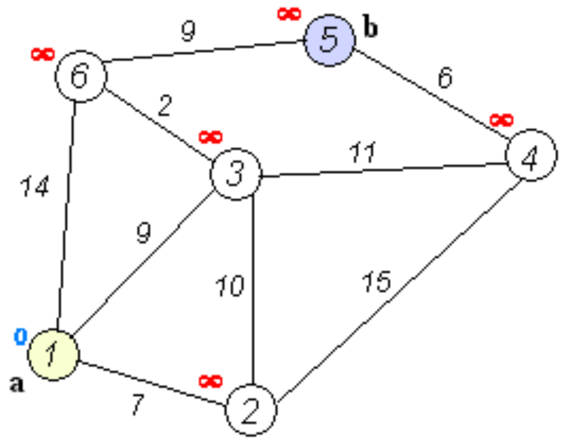
"Start state"

"Final state"

The Shortest Path Problem

P (Polynomial)

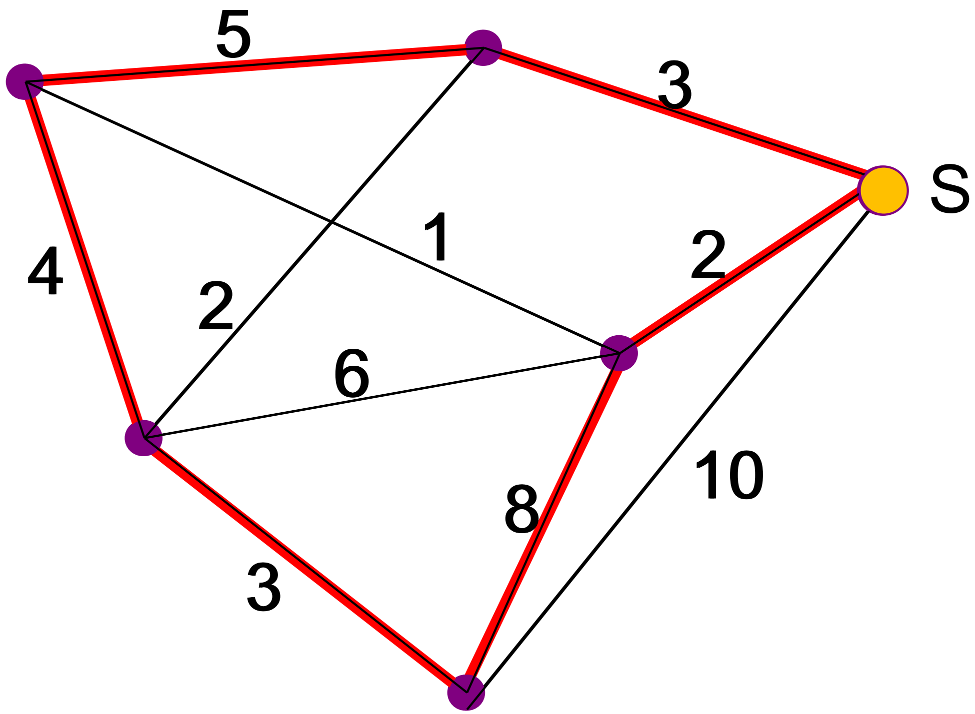
- Given:
  - Directed graph $G = (V, E)$
    - Length l_e = length of edge $e = (u, v) \in E$
      - Distance; time; cost
      - $l_e \geq 0$
  - Source s
- Goal:
  - Shortest path P_v from s to each other node $v \in V - \{s\}$
    - Length of path P: $l(P) = \Sigma_{e \in E}\, l_e$



$l(a \rightarrow b) = l(1 \rightarrow 3 \rightarrow 6 \rightarrow 5)$
$= 9+2+9 = 20$

Basic: $O(|V|^2)$
Fibonacci Heap: $O(|E| + |V| \log |V|)$

Example: the Traveling Salesman Problem



Cost: 25

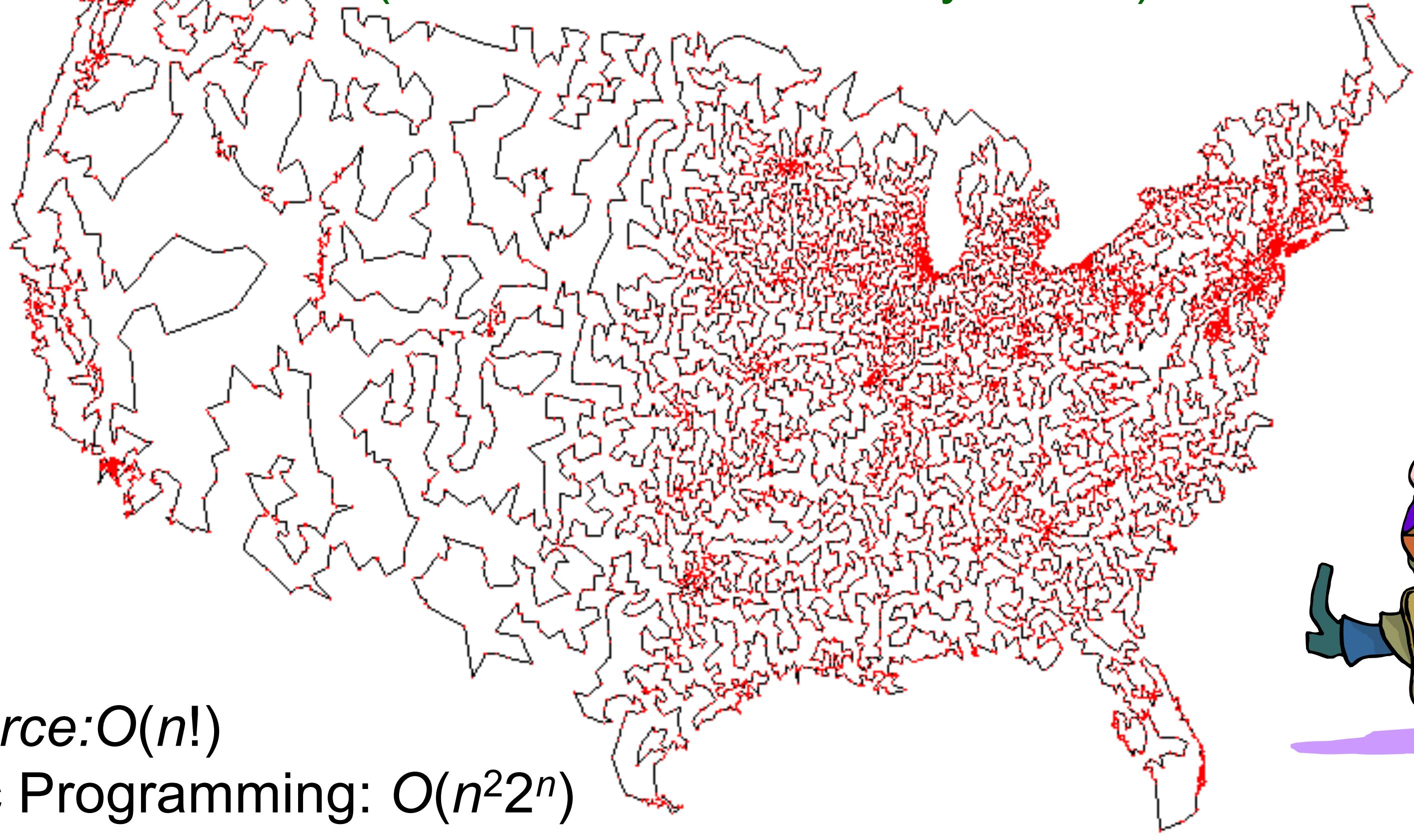
What is the least-cost round-trip route that visits each city exactly once and then returns to the starting city?

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

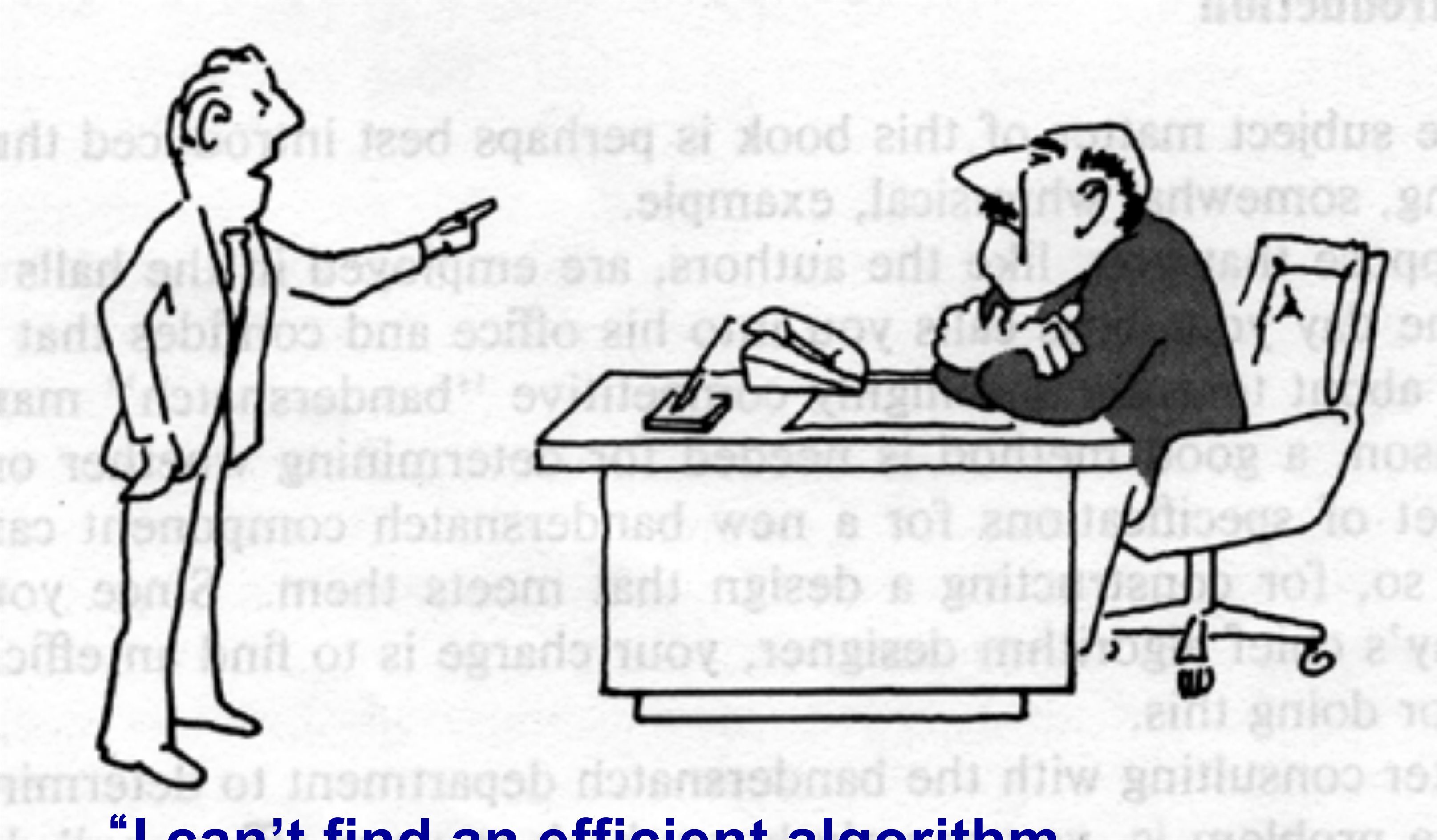
Reference: http://www.tsp.gatech.edu

Coping with a “Tough” Problem: Trilogy I

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

Courtesy of Prof. Y.-W. Chang

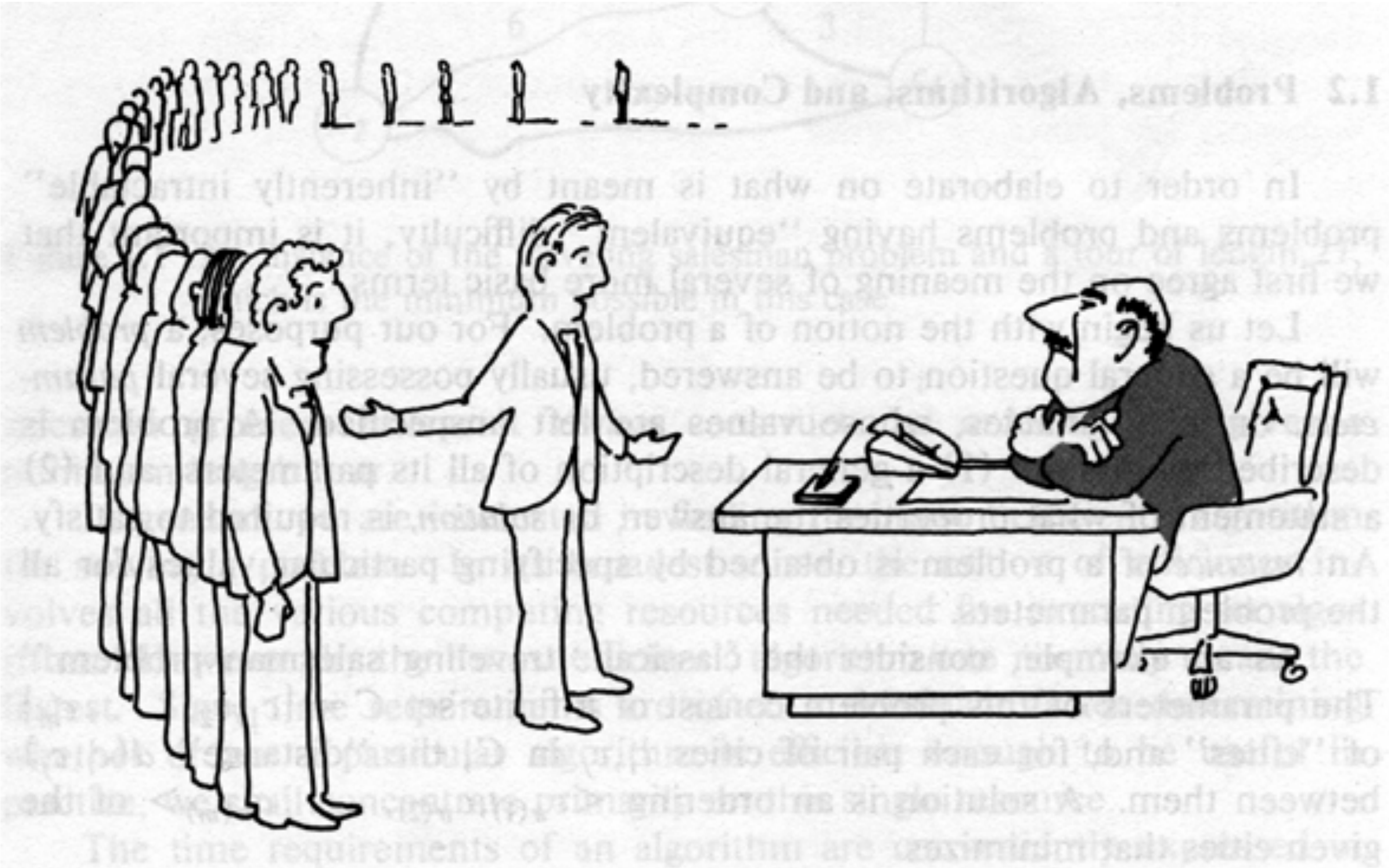
Coping with a "Tough" Problem: Trilogy II



"I can't find an efficient algorithm, because no such algorithm is possible!"

Courtesy of Prof. Y.-W. Chang

Coping with a “Tough” Problem: Trilogy III



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

Courtesy of Prof. Y.-W. Chang

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

1. Course Preliminaries
2. Mathematical Preliminaries and Notation
3. 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

$1 \in A$ ⇨ 1 is an element of the set A

$ship \notin B$ ⇨ ship is not an element of the set B

Set Representations

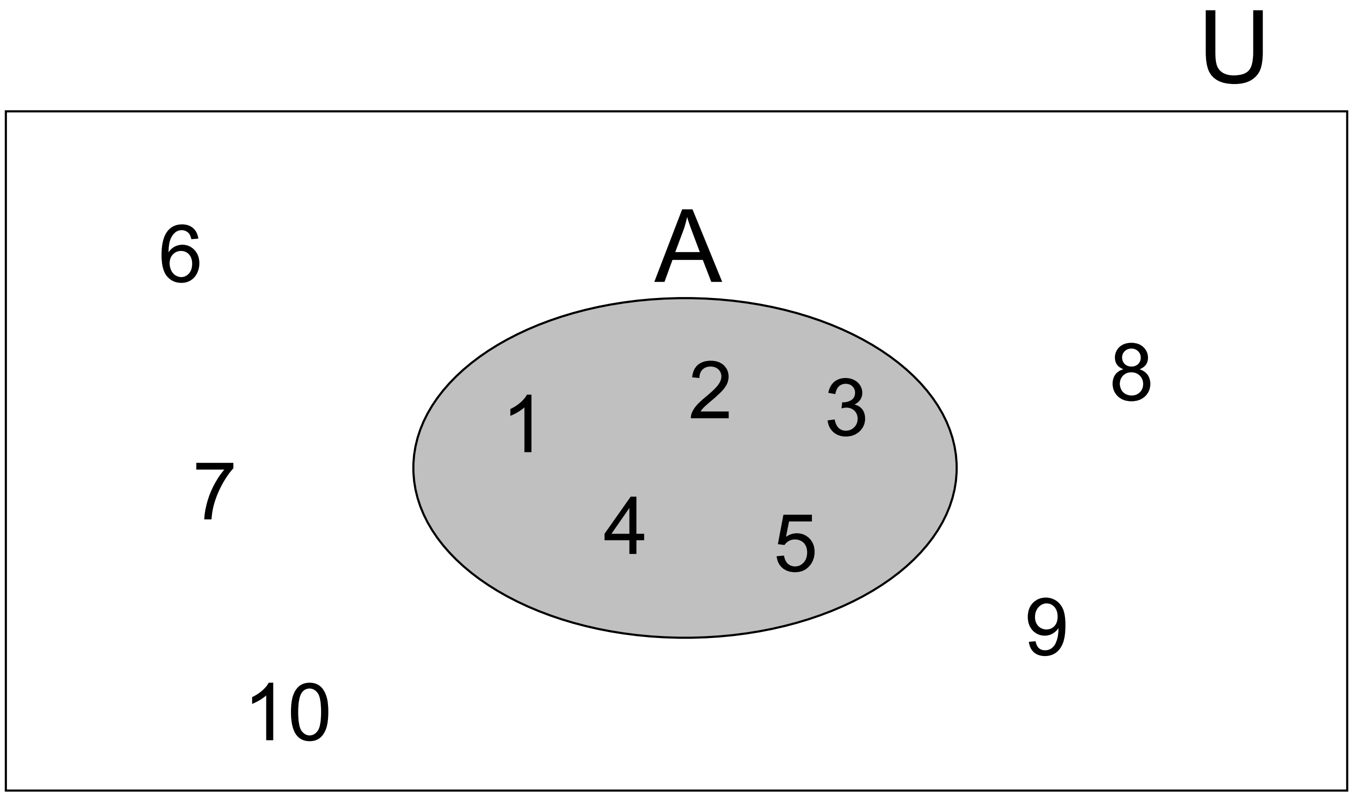
$C = \{ a, b, c, d, e, f, g, h, i, j, k \}$

$C = \{ a, b, \ldots, k \}$ → *finite set*

$S = \{ 2, 4, 6, \ldots \}$ → *infinite set*

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

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



Universal Set: all possible elements

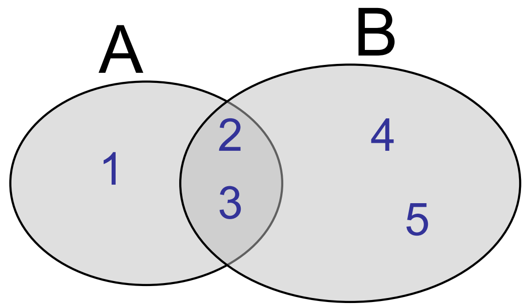
U = { 1 , … , 10 }

Set Operations

A = { 1, 2, 3 }   B = { 2, 3, 4, 5 }

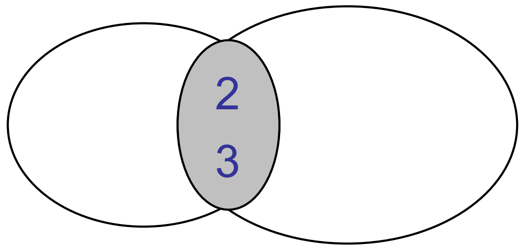
- Union (U)

  A U B = { 1, 2, 3, 4, 5 }



- Intersection (∩)

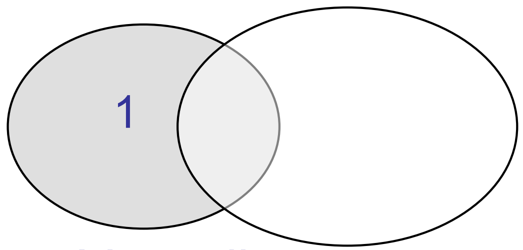
  A ∩ B = { 2, 3 }



- Difference (-)

  A - B = { 1 }

  B - A = { 4, 5 }

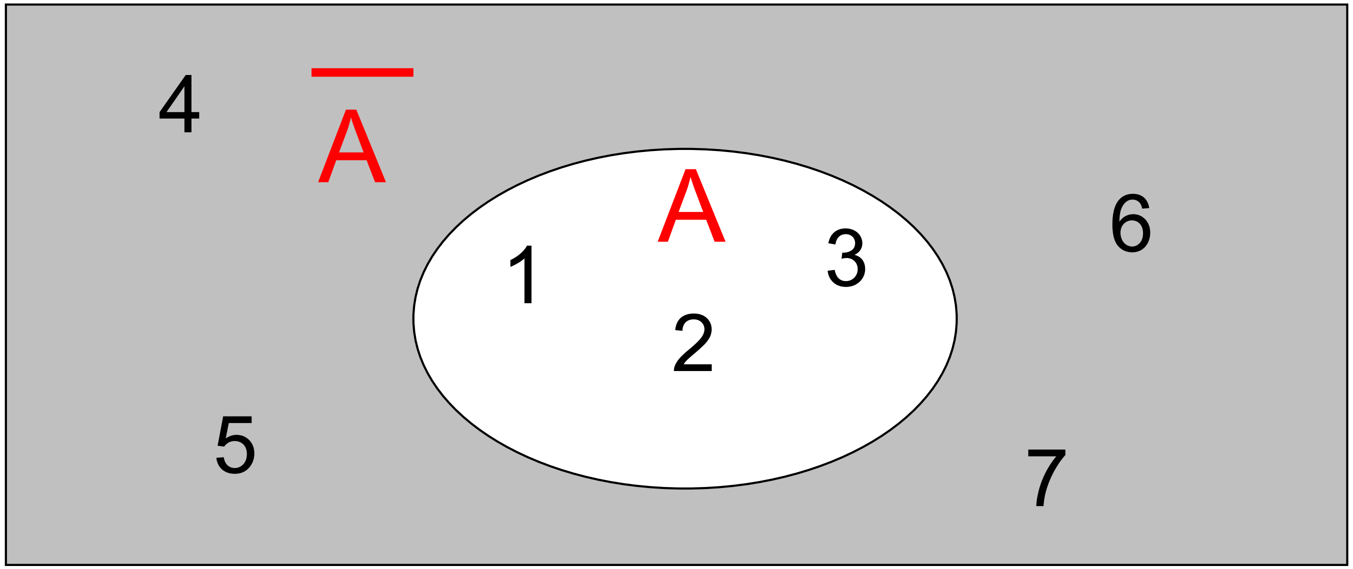


Venn diagrams

- Complement

Universal set = {1, …, 7}

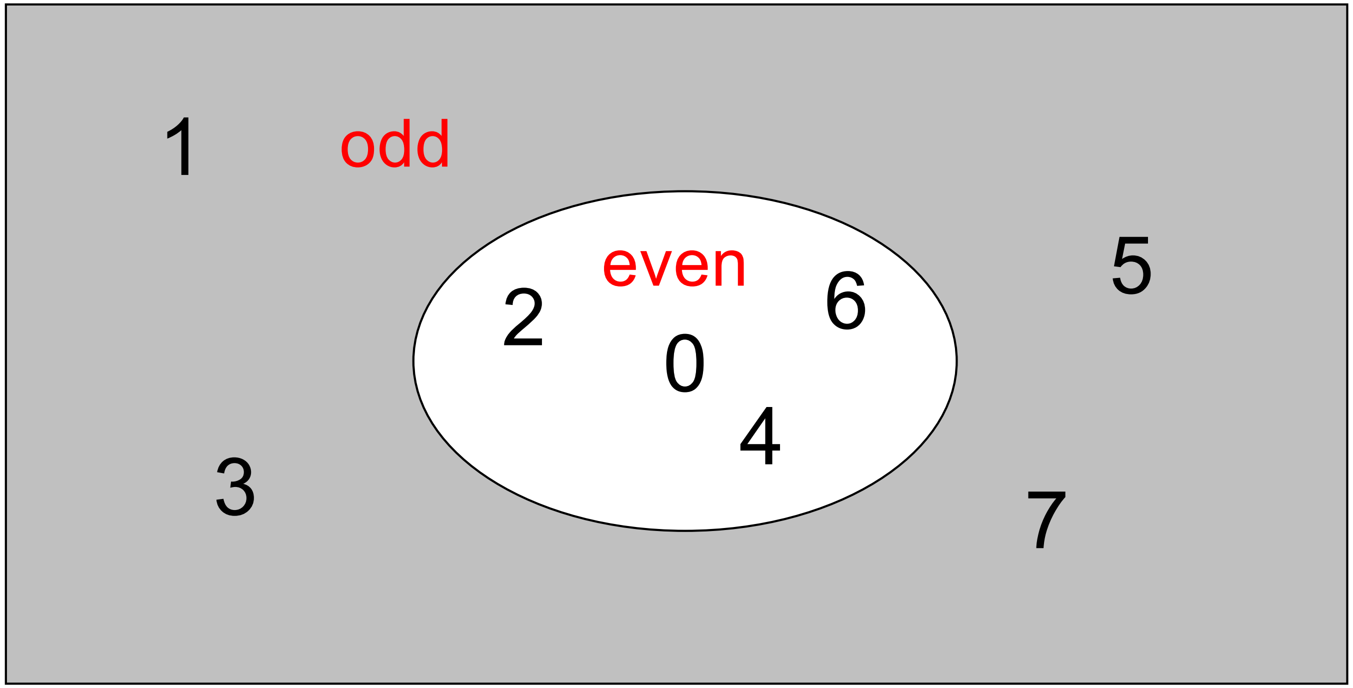
$A = \{1, 2, 3\} \Rightarrow \overline{A} = \{4, 5, 6, 7\}$



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

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

Integers



DeMorgan's Laws

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

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

Empty, Null Set: $\emptyset$

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

$S \cup \emptyset = S$

$S \cap \emptyset = \emptyset$

$S - \emptyset = S$

$\emptyset - S = \emptyset$

$\overline{\emptyset}$ = Universal Set

Subset

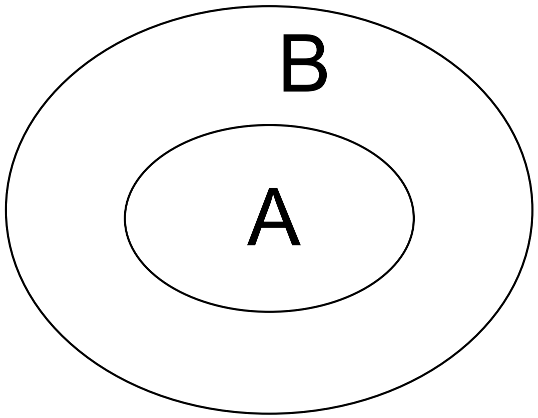
$A = \{ 1, 2, 3 \}$ $B = \{ 1, 2, 3, 4, 5 \}$

$A \subseteq B$

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

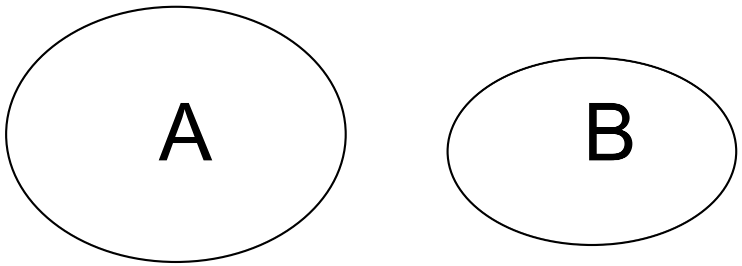
$A \subset B$



Disjoint Sets

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

$$A \cap B = \emptyset$$



Set Cardinality

- For finite sets

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

$|A| = 3$ (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

$A = \{ 2, 4 \}$ $\qquad$ $B = \{ 2, 3, 5, 6 \}$

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

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

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

The pair (4, 2) is in A X B, but (2, 4) is not

Partition

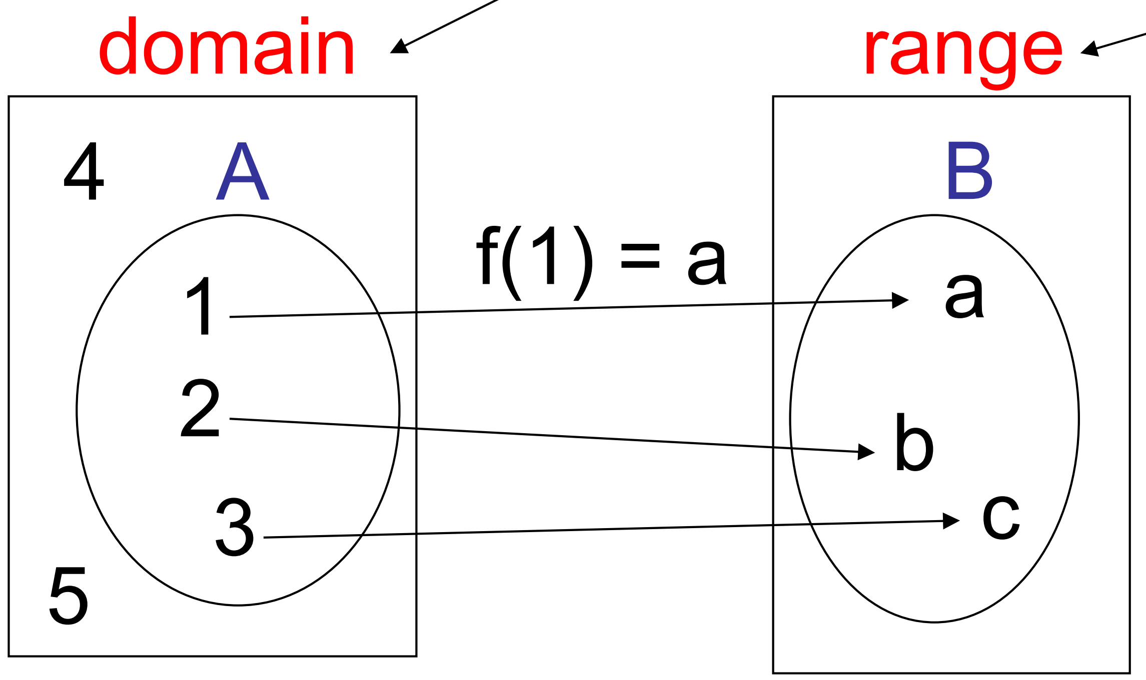
A set can be divided by separating it into a number of subsets. Suppose that S_1, S_2, …, S_n are subsets of a given set S and that the following holds:

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

Then S_1, S_2, …, S_n is called a ***partition*** of S.

FUNCTIONS

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



$f : A \rightarrow B$

If A = domain

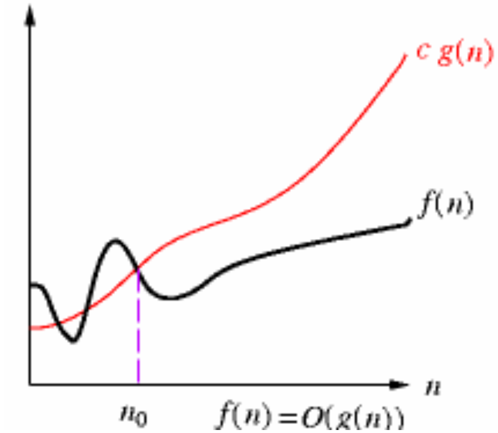
then f is a total function

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



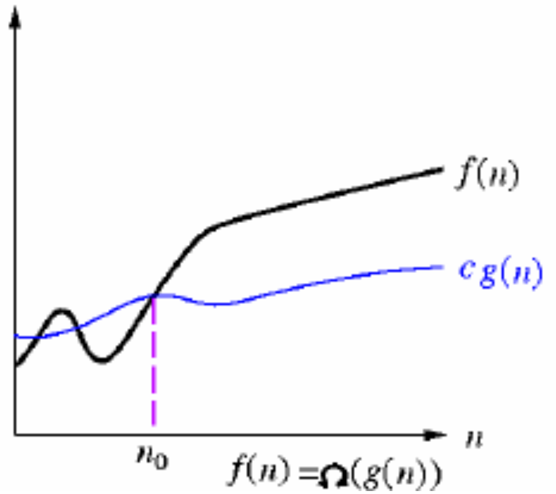
1. $3n^2 + n = O(n^2)$?
2. $3n^2 + n = O(n)$?
3. $3n^2 + n = O(n^3)$?

- $O(n) + O(n) = 2O(n)$?

Asymptotic Analysis

Ω : Lower Bounding Function

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

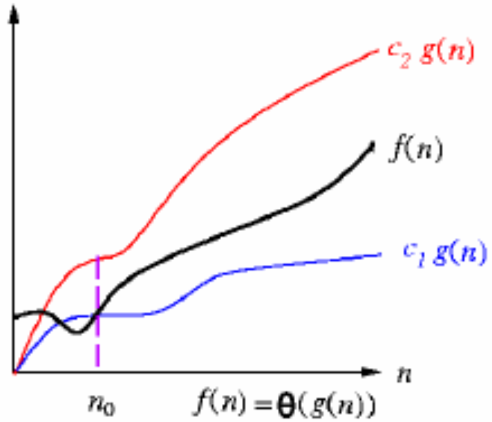


1. **$3n^2 + n = \Omega(n^2)$?**
2. **$3n^2 + n = \Omega(n)$?**
3. **$3n^2 + n = \Omega(n^3)$?**

Asymptotic Analysis

θ: Tightly Bounding Function

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

$f(n) = 2n^2 + 3n$,
$g(n) = n^3$,
$h(n) = 10n^2 + 100$.

$f(n) = \quad (g(n))$,
$g(n) = \quad (h(n))$
$f(n) = \quad (h(n))$

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.

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

$$x_i \; R \; y_i$$

e. g. if R = ‘>’: $2 > 1, \quad 3 > 2, \quad 3 > 1$

Equivalence Relations (≡)

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

Example: R ≡ '='

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

Example 1.4

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

$$x \equiv y$$

If and only if

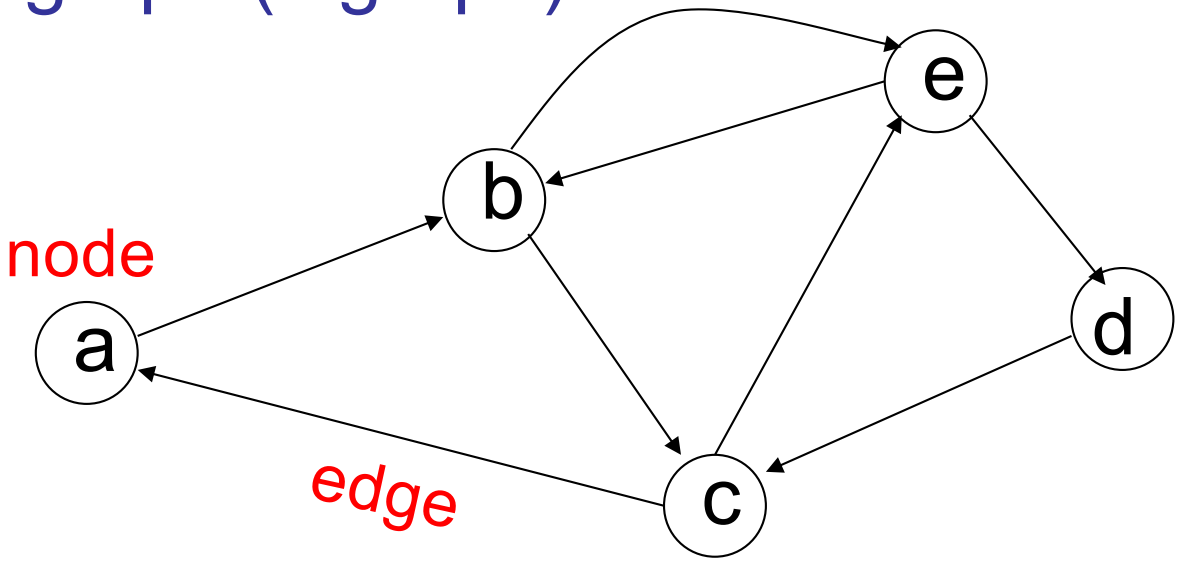
$$x \bmod 3 = y \bmod 3.$$

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)



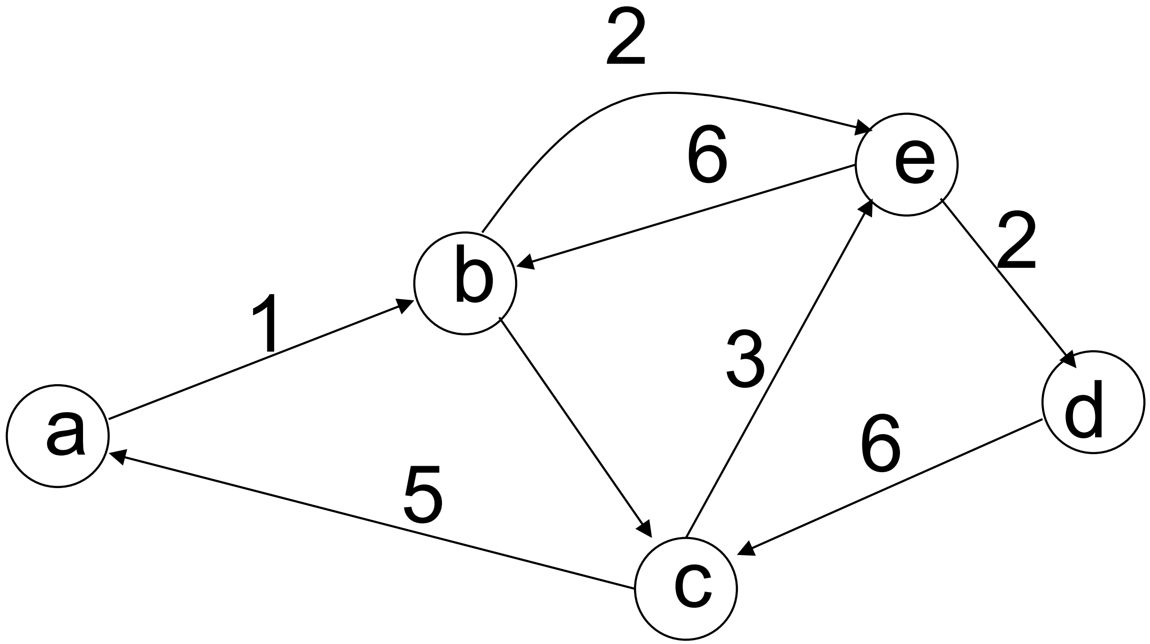
- Nodes (Vertices)

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

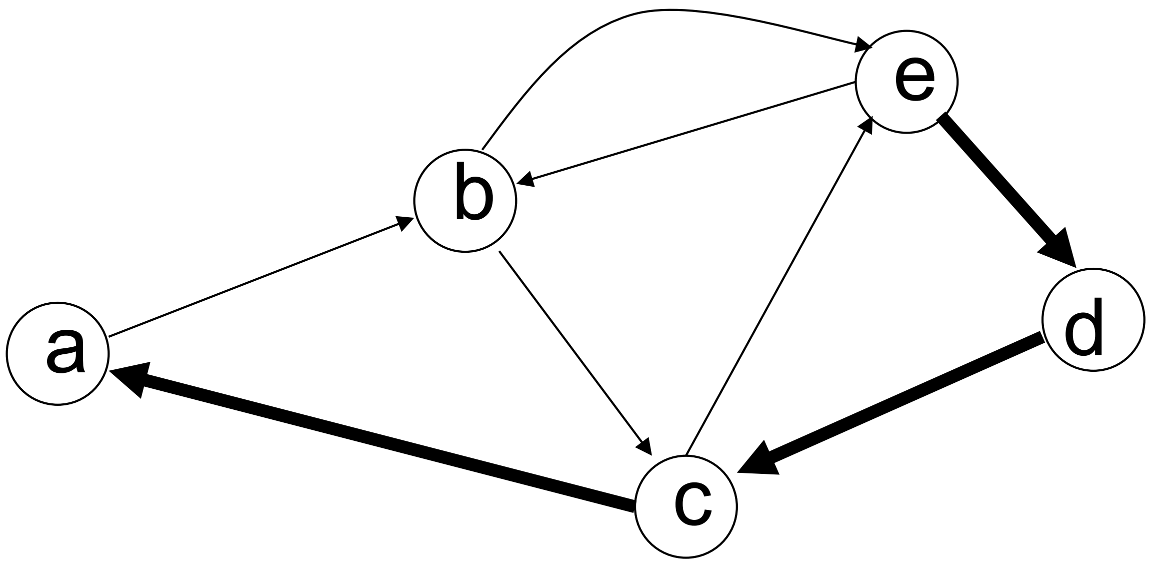
- Edges

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

Labeled Graph



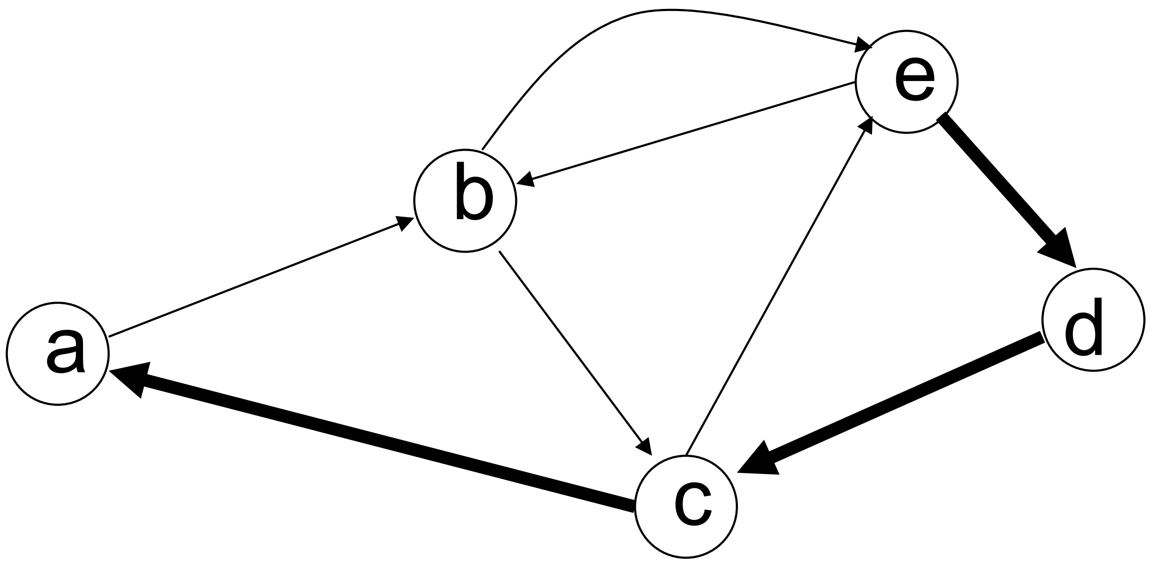
Walk



Walk is a sequence of adjacent edges

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

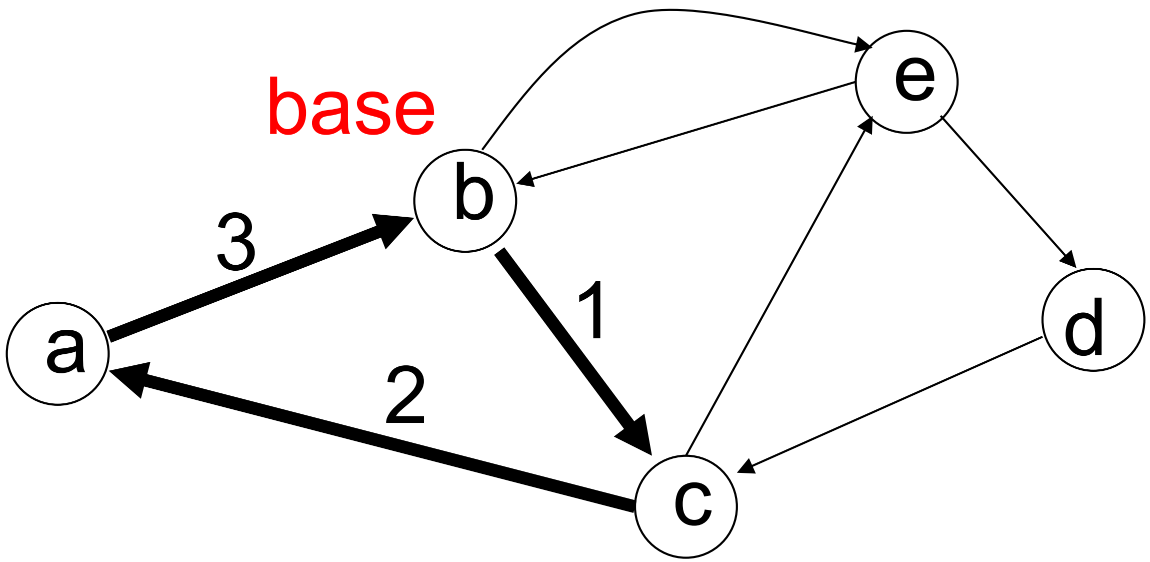
Path



Path: a walk where no edge is repeated

Simple path: no node is repeated

Cycle

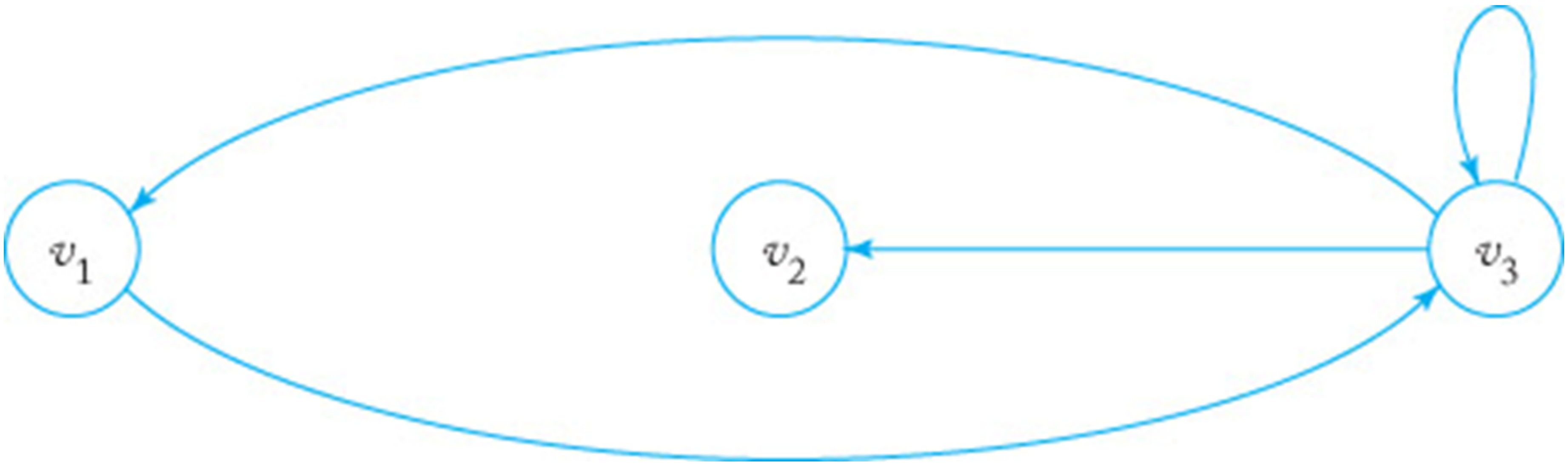


Cycle: a walk from a node (base) to itself

Simple cycle: only the base node is repeated

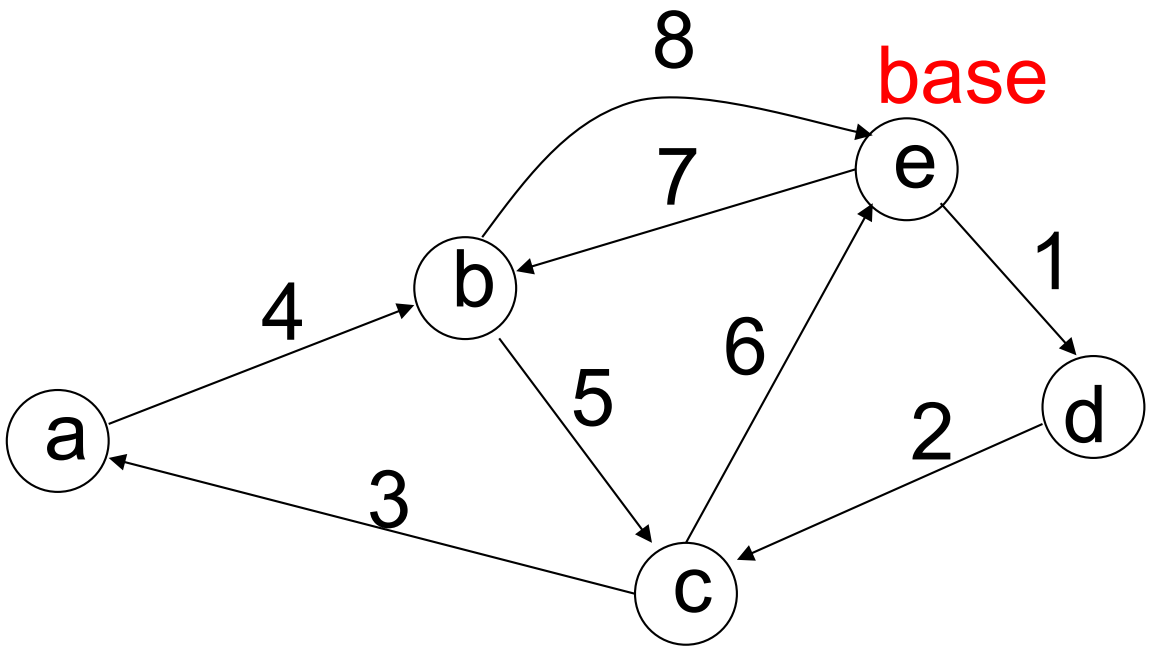
Loop

An edge from a vertex to itself



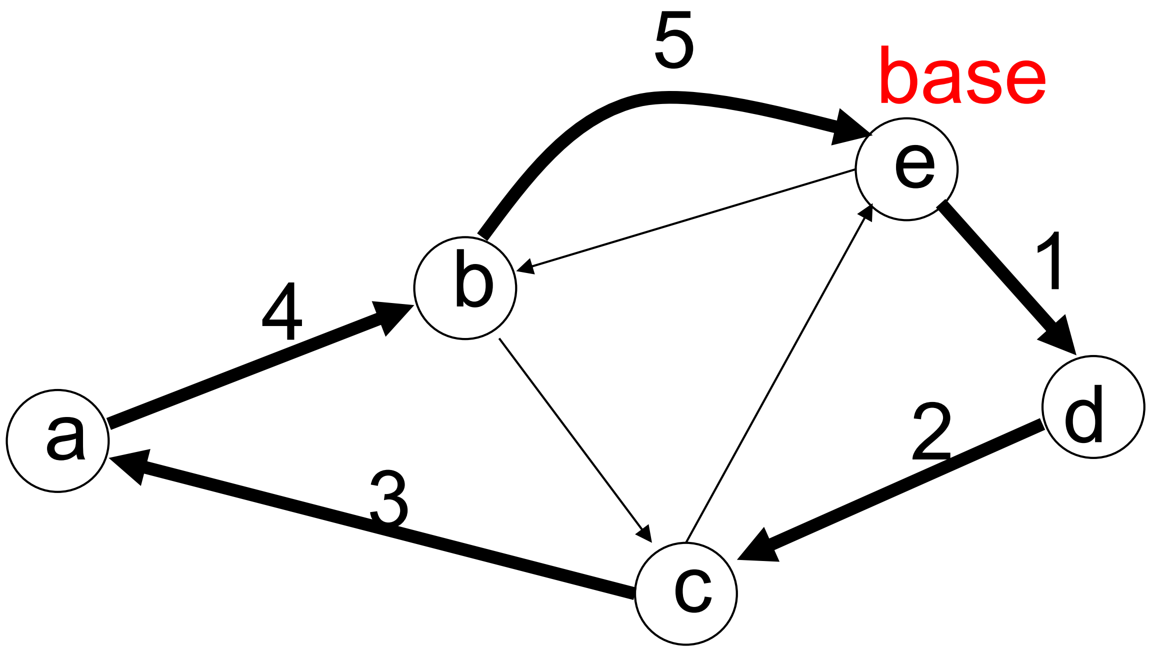
- $(v_1, v_3), (v_3, v_2)$ is a simple path from v_1 to v_2
- $(v_1, v_3), (v_3, v_3), (v_3, v_1)$ is a cycle (not simple one)
- There is a loop on vertex v_3

Euler Tour



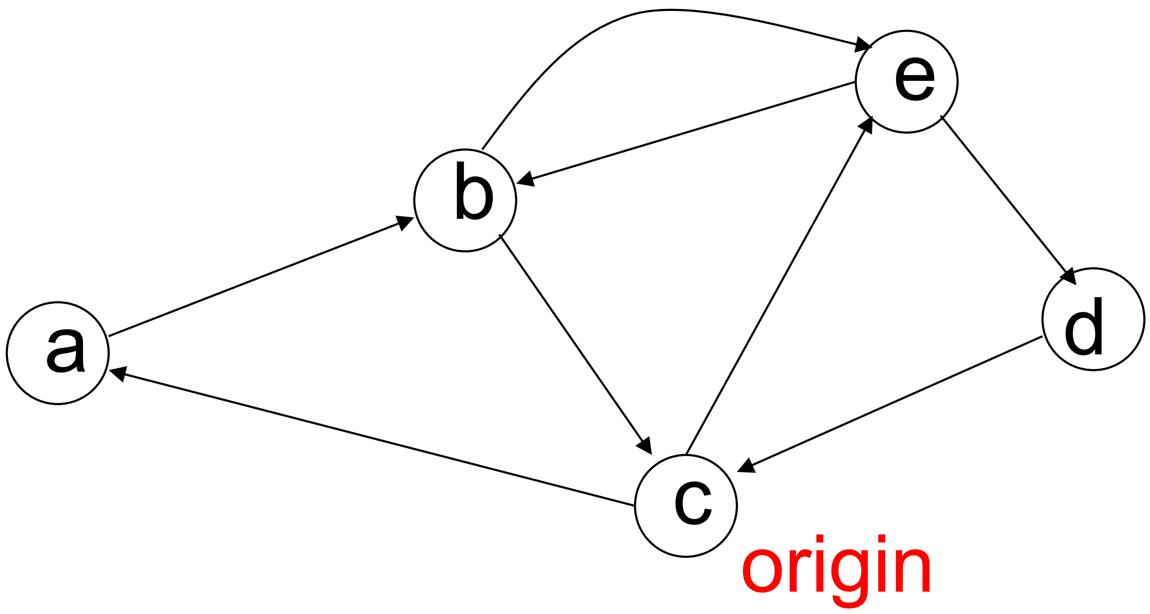
A cycle that contains each edge once

Hamiltonian Cycle

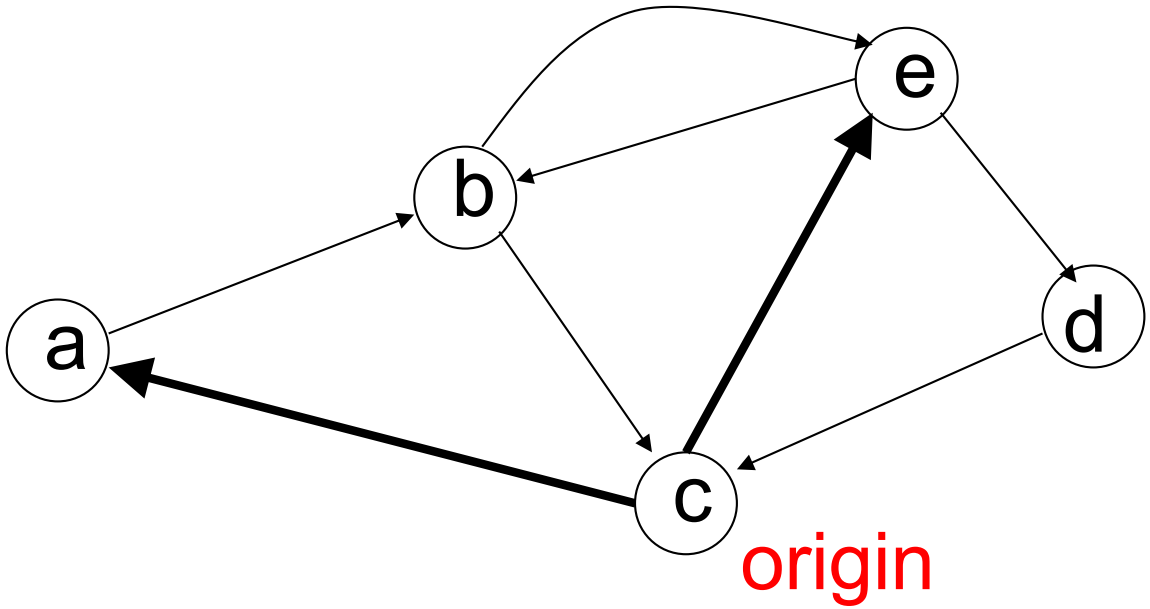


A simple cycle that contains all nodes

Finding All Simple Paths



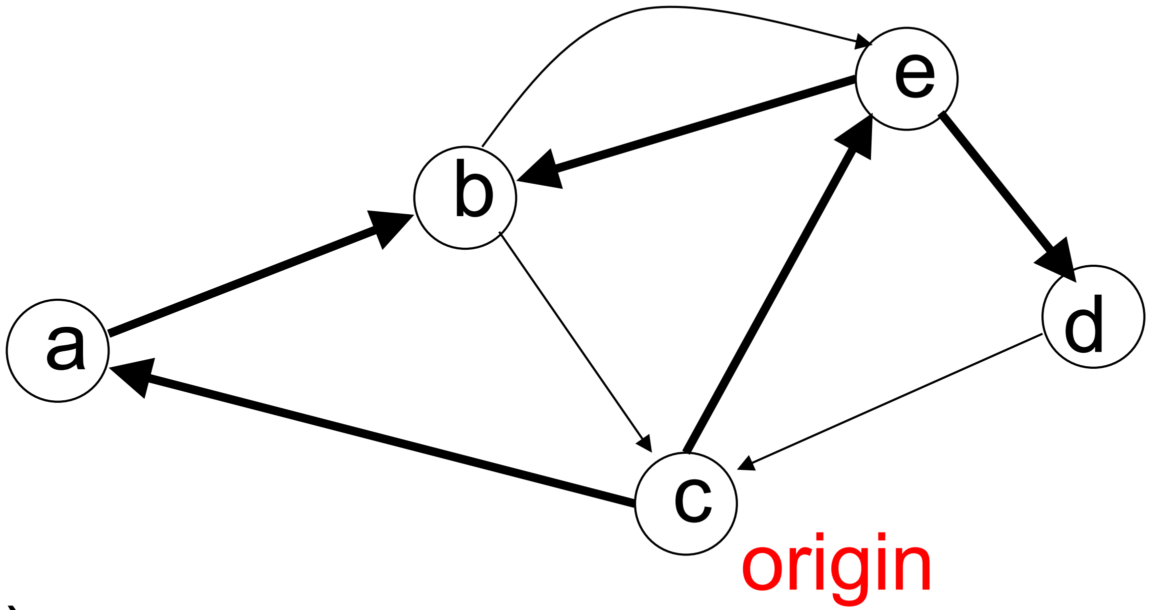
Step 1



(c, a)

(c, e)

Step 2



(c, a)

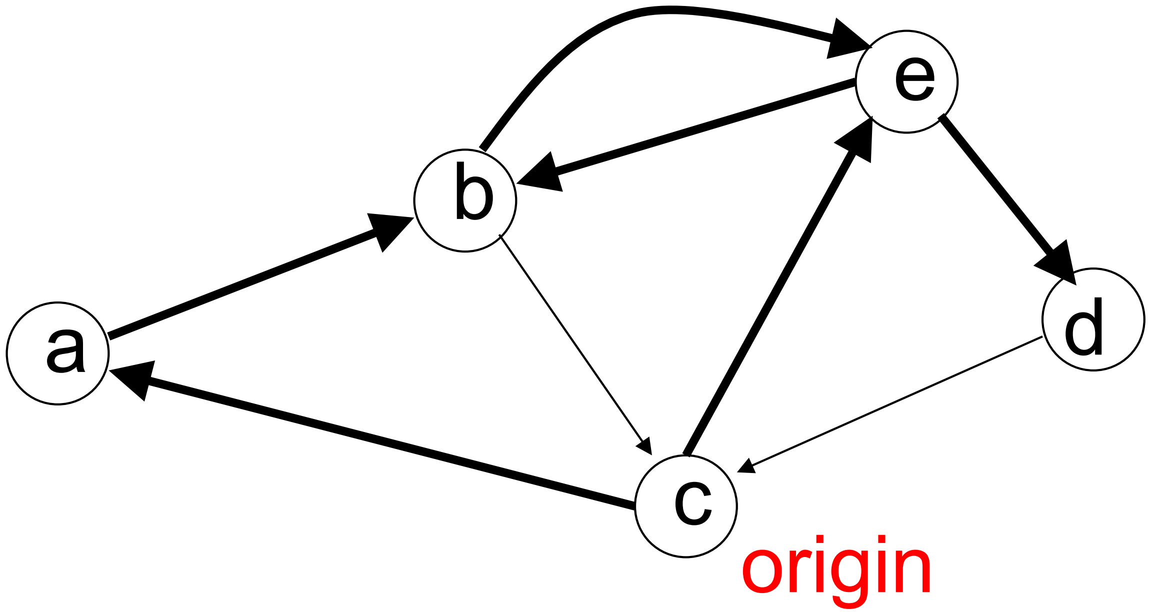
(c, a), (a, b)

(c, e)

(c, e), (e, b)

(c, e), (e, d)

Step 3



(c, a)

(c, a), (a, b)

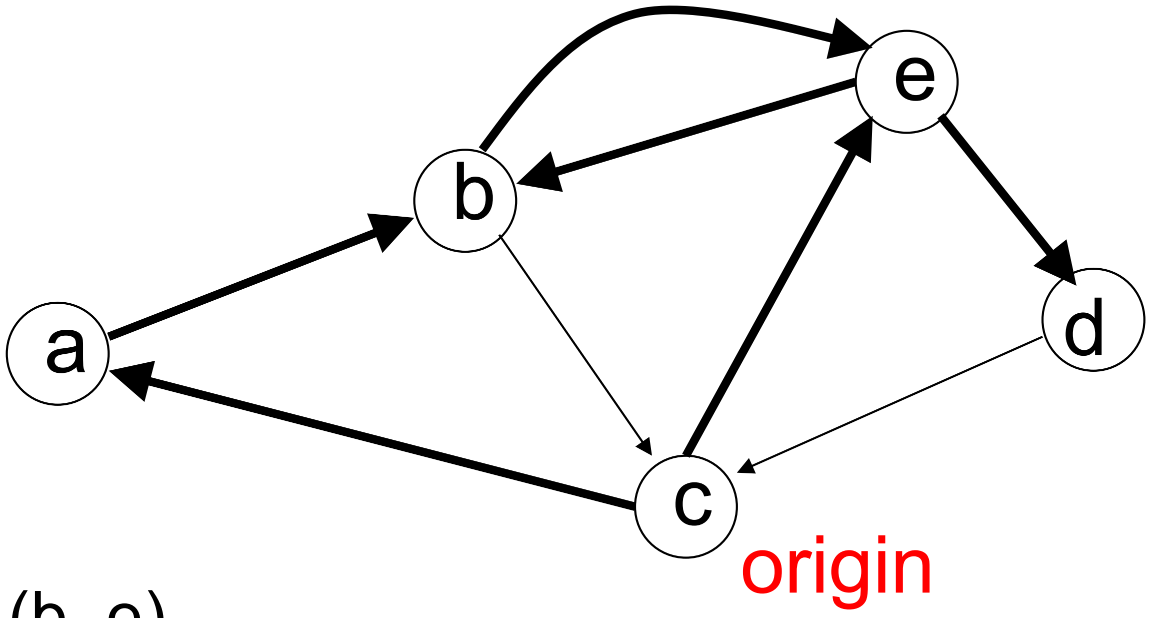
(c, a), (a, b), (b, e)

(c, e)

(c, e), (e, b)

(c, e), (e, d)

Step 4



(c, a)

(c, a), (a, b)

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

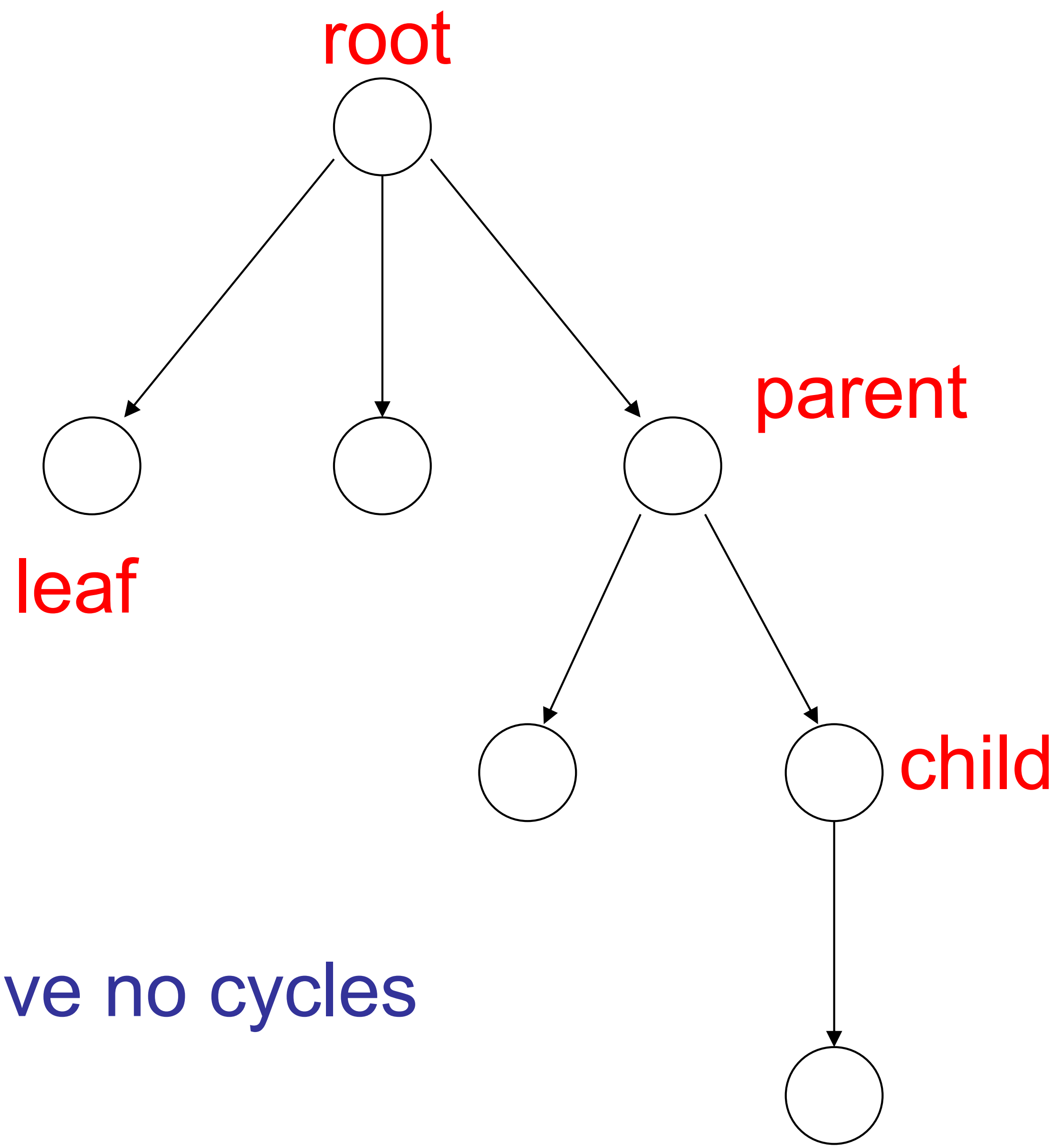
(c, a), (a, b), (b, e), (e,d)

(c, e)

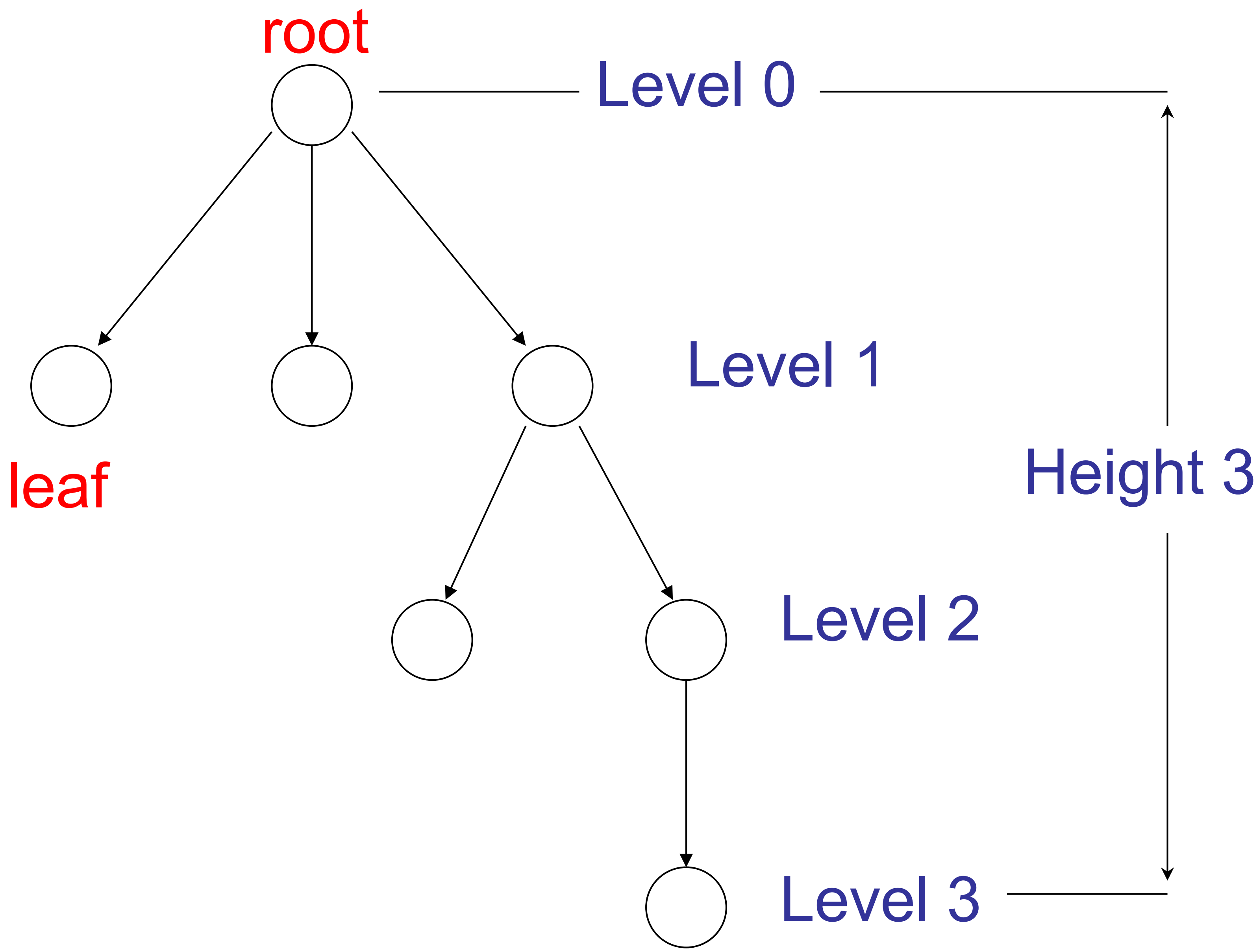
(c, e), (e, b)

(c, e), (e, d)

Trees



Trees have no cycles


root
Level 0
Level 1
leaf
Height 3
Level 2
Level 3

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

    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 <u>constructive</u> proof.

Induction

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

If we know

- for some b that $P_1, P_2, \ldots, P_b$ are true
- for any $k >= b$ that

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

Then

Every P_i is true

Proof by Induction

- Inductive basis

  Find P_1, P_2, …, P_b which are true

- Inductive hypothesis

  Let's assume P_1, P_2, …, P_k are true,

  for any $k >= b$

- Inductive step

  Show that P_{k+1} is true

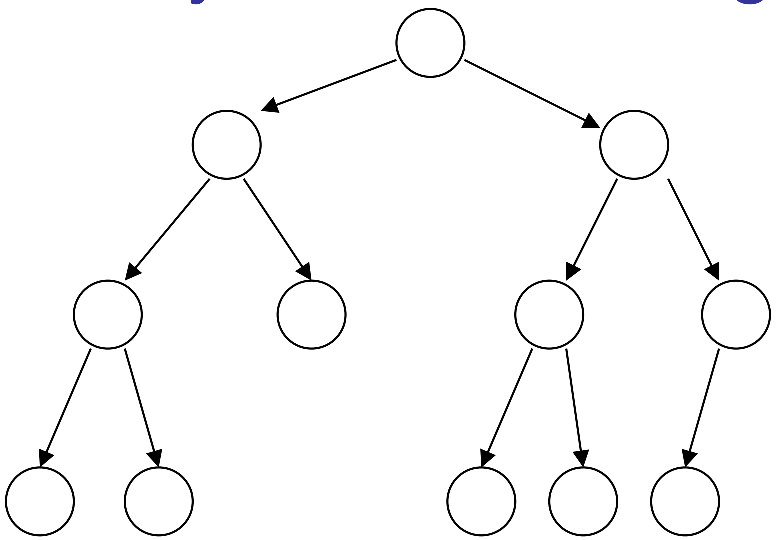
Example 1.5

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

Proof by induction:

let L(i) be the maximum number of

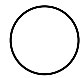
leaves of any subtree at height i



We want to show: $L(i) <= 2^i$

- Inductive basis

$L(0) = 1$ (the root node)



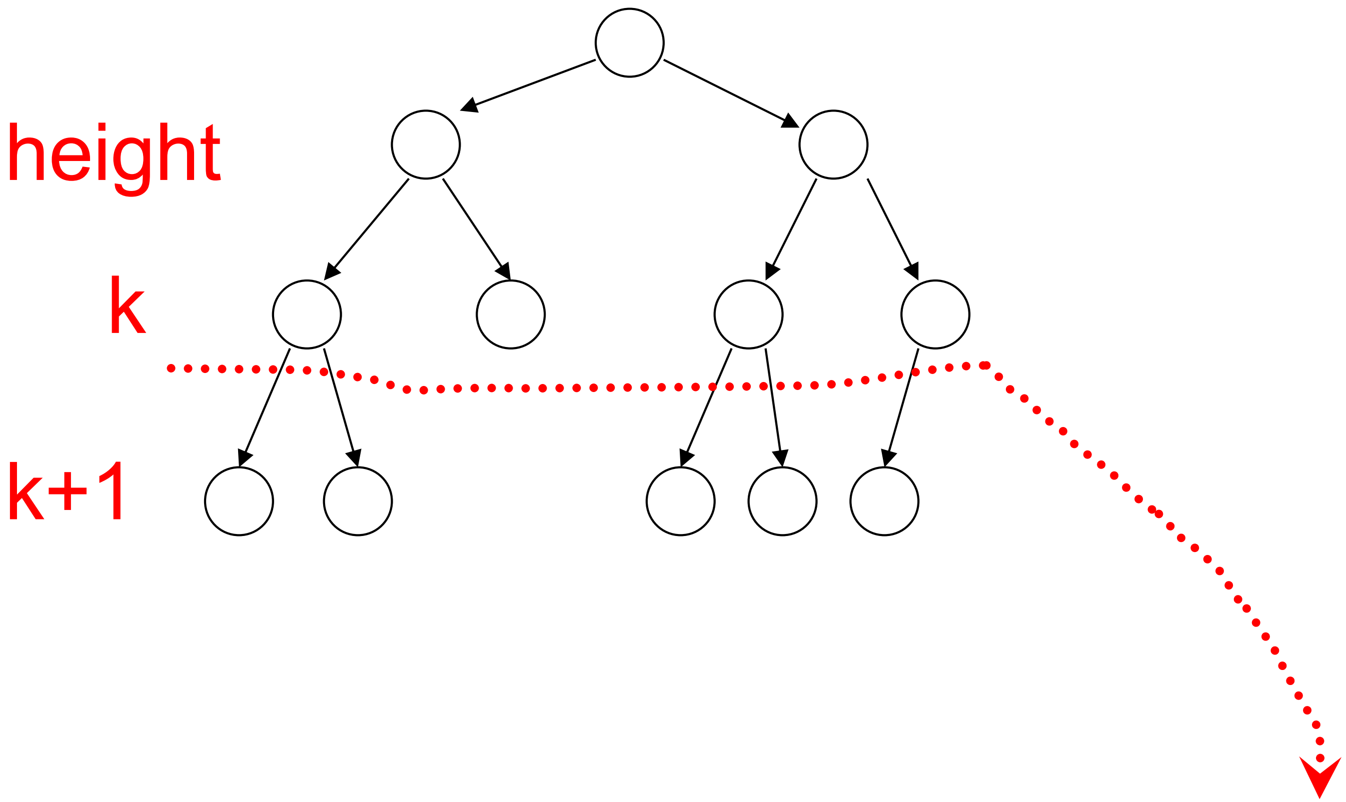
- Inductive hypothesis

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

- Induction step

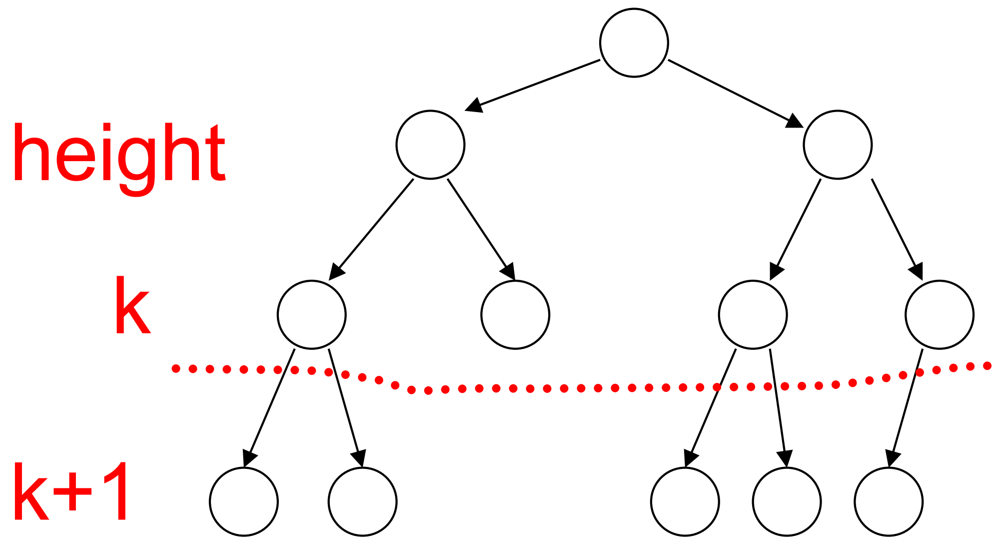
we need to show that $L(k + 1) <= 2^{k+1}$

Induction Step



From Inductive hypothesis: L(k) <= 2^k

Induction Step



$L(k) <= 2^k$

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

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

Remark

Recursion is another thing

Example of recursive function:

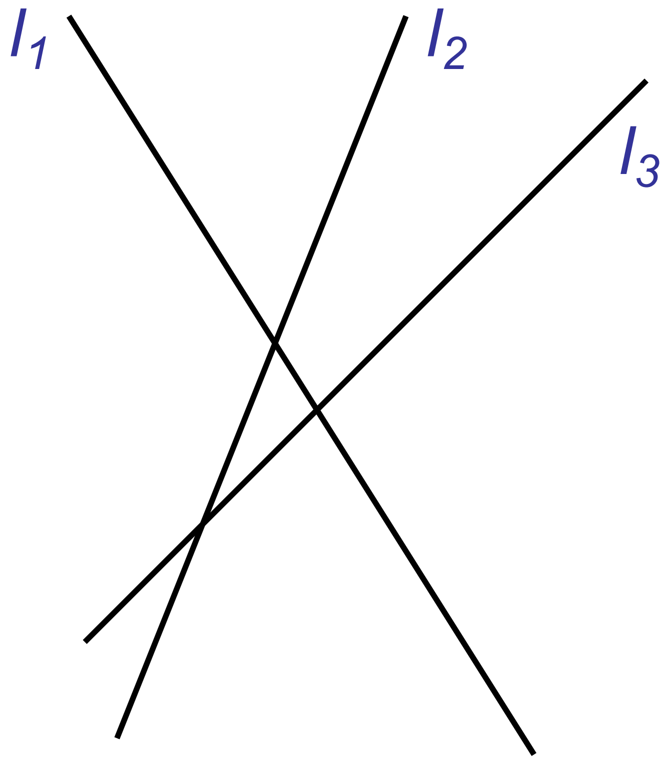
$$f(n) = f(n-1) + f(n-2)$$

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

Example 1.6

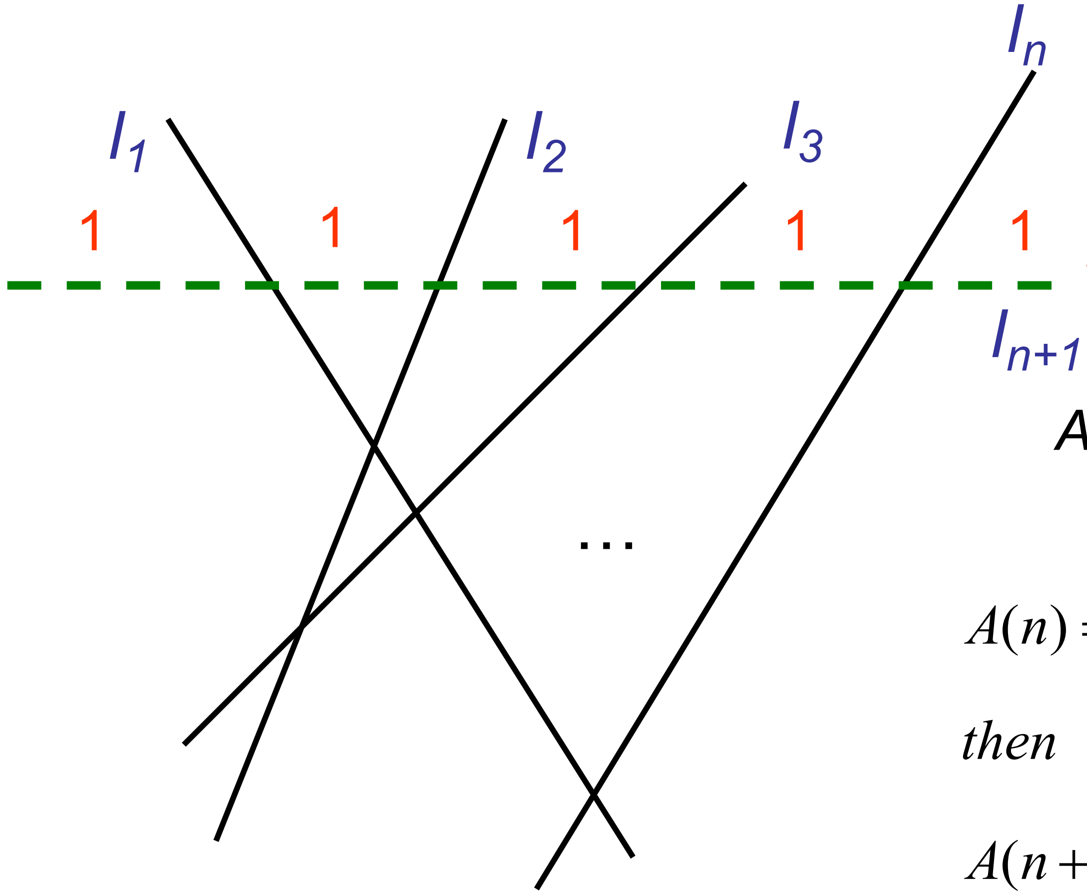
A set $l_1, l_2, \ldots, l_n$ of mutually intersecting straight lines divides the plane into a number of separated regions

1 line → 2 regions, 2 lines → 4 regions, 3 lines → 7 regions



Solve it recursively!!

Example 1.6



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

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

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

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

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

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

Therefore, n^2 is even $\Rightarrow$ n is even

$n = 2k$

$2m^2 = 4k^2 \Rightarrow m^2 = 2k^2 \Rightarrow$ m is even

$m = 2p$

Thus, m and n have common factor 2

Contradiction!

Outline

1. Course Preliminaries
2. Mathematical Preliminaries and Notation
3. 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:

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

Alphabets and Strings

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

a

ab $\qquad u = ab$

$abba$ $\qquad v = bbbaaa$

$baba$ $\qquad w = abba$

$aaabbbaabab$

String Operations

$w = a_1 a_2 \cdots a_n$ $\qquad$ $abba$

$v = b_1 b_2 \cdots b_m$ $\qquad$ $bbbaaa$

Concatenation

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

$$w = a_1 a_2 \cdots a_n$$

$$ababaaabbb$$

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: $u = aab, \quad |u| = 3$

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

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

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

Empty String

- A string with no letters: λ

- Observations: $|\lambda| = 0$

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

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

Substring

- Substring of string:
  - a subsequence of consecutive characters

String	Substring
$\underline{ab}bab$	ab
$\underline{abba}b$	$abba$
$ab\underline{b}ab$	b
$a\underline{bbab}$	$bbab$

Prefix and Suffix

$$abbab$$

Prefixes	Suffixes
λ	$abbab$
a	$bbab$
ab	bab
abb	ab
$abba$	b
$abbab$	λ

$$w = uv$$

prefix

suffix

Another Operation

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

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

- Definition: $w^0 = \lambda$

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

The * Operation

Σ^* : the set of all possible strings from alphabet Σ

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

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

The + Operation

Σ^+ : the set of all possible strings from alphabet Σ except λ

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

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

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

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

Languages

A language is any subset of Σ^*

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

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

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

$$\left.\begin{array}{l} \lambda \\ ab \\ aabb \\ aaaaabbbbb \end{array}\right\} \in L \qquad\qquad abb \notin L$$

Operations on Languages

- The usual set operations

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

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

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

- Complement: $\overline{L} = \Sigma * - L$

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

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

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

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

$$aabbaaabbb \in L^2$$

Note that n and m in the above are unrelated

Star-Closure (Kleene *)

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

- Example:

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

Positive Closure

- Definition: $L^{+} = L^{1} \bigcup L^{2} \bigcup \cdots$

$$= L^{*} - \{\lambda\}$$

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

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代表所有可以演化的變數

- 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

a,b為這Grammar中的terminal string

Ex:

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

開始的樣子為S

P: $S \rightarrow aSb$,

$S \rightarrow \lambda$

lambda 的目的是可以讓程式可以終止,
所以aSb可以轉換成aaSbb 或是ab

Grammars

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

- Let's formalize this a bit:

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

  They specify how the grammar transforms one string into another
- We say that γ can be derived from α 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 $\alpha \overset{*}{\Rightarrow} \gamma$ if γ can be derived from α (or say α derives γ) in zero or more steps.

Definition 1.2

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

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

is the language generated by G

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

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

is a derivation of the sentence w.

- $S, w_1, w_2, \ldots, w_n$ 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

$$S \rightarrow aSb,$$

$$S \rightarrow \lambda$$

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

Then

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

So we can write

$$S \overset{*}{\Rightarrow} aabb$$

- 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

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

$S \to SS,$

$S \to \lambda,$

$S \to aSb,$

$S \to bSa,$

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,

$$S \overset{*}{\Rightarrow} w_1$$

Then

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

Else

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

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:
  $S \to aSb$, $S \to \lambda$
- $G_2 = (\{S, A\}, \{a, b\}, S, P_2)$ with P_2:
  $S \to aAb \mid \lambda$, $A \to aAb \mid \lambda$

Questions?