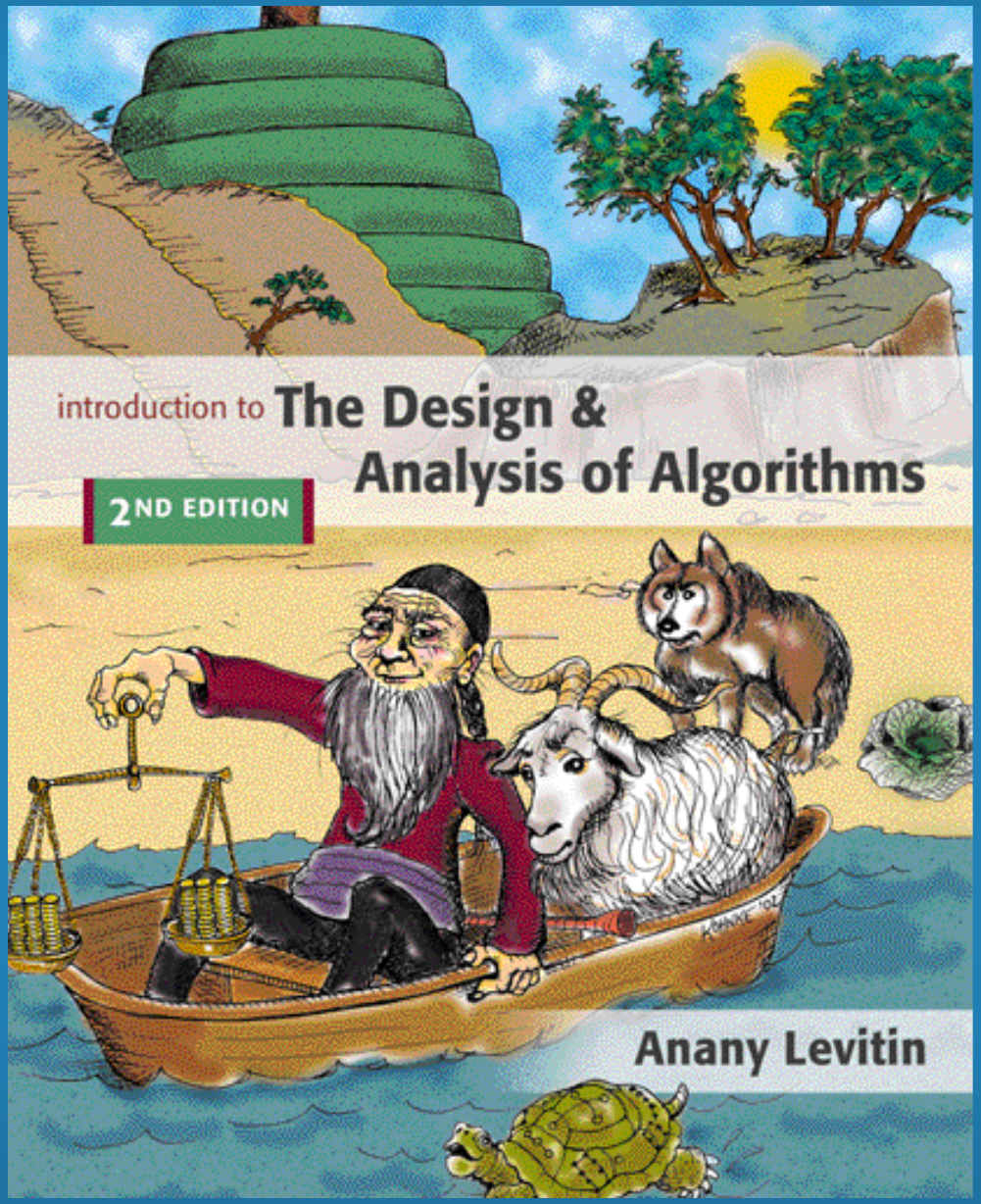
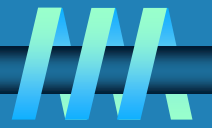
Chapter 1

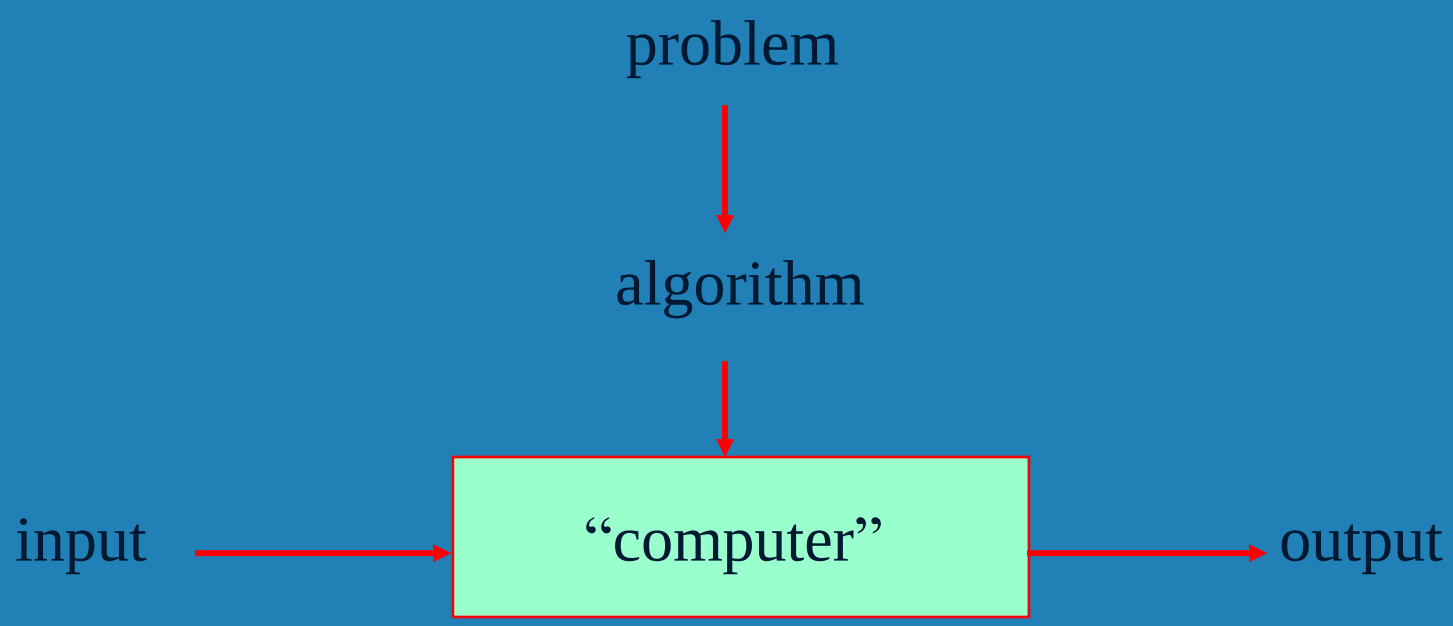
Introduction



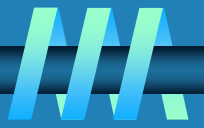
What is an algorithm?



An *algorithm* is a sequence of unambiguous instructions for solving a problem, i.e., for obtaining a required output for any legitimate input in a finite amount of time.

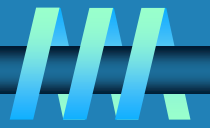


Historical Perspective



- **Muhammad ibn Musa al-Khwarizmi – 9th century mathematician**
  - **Al-Khwarizmi (9th century C.E.) was a great Arabic scholar, most famous for his algebra textbook.**
  - **First examples of algorithms can be found in Mesopotamian tablets and in Egyptians scrolls.**
  - **An important role in the development of numerical algorithms was played in the ninth century by the Persian mathematician al-Khwarizmi, who introduced the Indian numeration systems to the Arab world and from whom we derived the name 'algorithm' to denote computing procedures.**
  - **In the Middle Ages algorithms for commercial transactions were widely used, but it was not until the nineteenth century that the problem of characterizing the power of algorithms was addressed. The precise definition of 'algorithm' and of the notion of computability were established by A.M. Turing in the 1930s. His work is also considered the beginning of the history of Computer Science.**

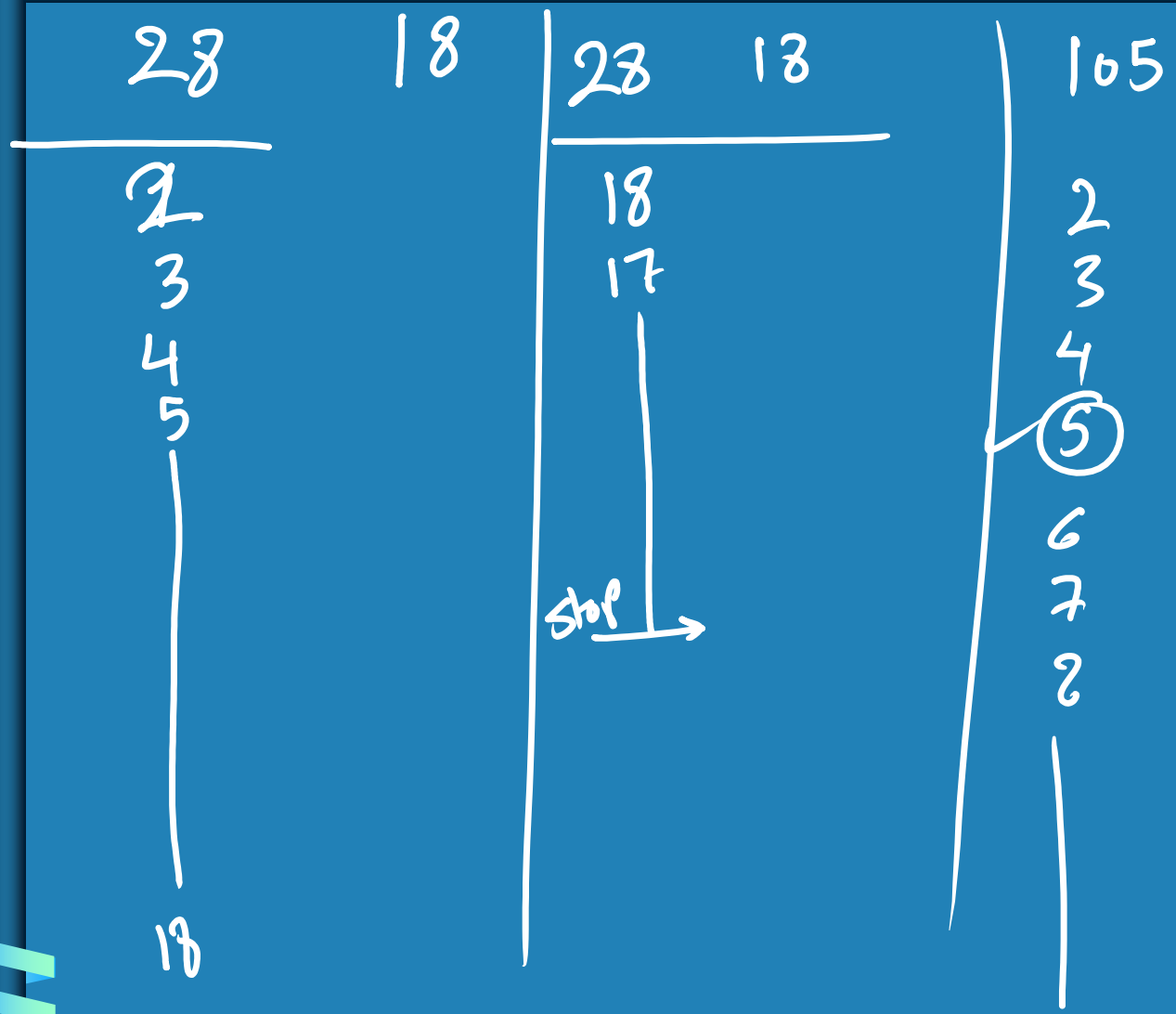
What is an algorithm?



- **An *algorithm* is a sequence of unambiguous instructions for solving a problem, i.e., for obtaining a required output for any legitimate input in a finite amount of time.**

  - Unambiguity/clearness
  - Can be represented various forms
  - Specified range of inputs
  - Several algorithms may solve the same problem with different ideas and different speeds

GCD(m, n)



What is an algorithm?

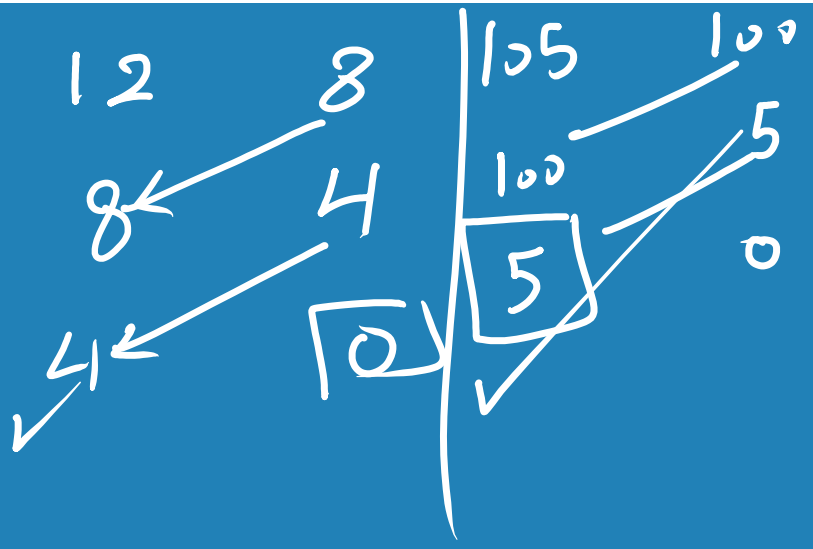
- **Euclid's algorithm for finding the greatest common divisor**
- **gcd(m, n) = gcd(n, m mod n)**

Euclid's algorithm for computing gcd(m, n)

Step 1 If $n = 0$, return the value of m as the answer and stop; otherwise, proceed to Step 2.

Step 2 Divide m by n and assign the value of the remainder to r.

Step 3 Assign the value of n to m and the value of r to n. Go to Step 1.



$$GCD(m,n) = \begin{cases} GCD(n, m\%n) & \\ m & n=0 \end{cases}$$

What is an algorithm?

- **Alternatively, we can express the same algorithm in pseudocode:**

ALGORITHM *Euclid*(m, n)

//Computes gcd(m, n) by Euclid's algorithm
//Input: Two nonnegative, not-both-zero integers m and n
//Output: Greatest common divisor of m and n
while $n \neq 0$ **do**
    $r \leftarrow m \bmod n$
    $m \leftarrow n$
    $n \leftarrow r$
return m

What is an algorithm?

- **The second algorithm for computing the greatest common divisor (Consecutive integer checking algorithm).**

Consecutive integer checking algorithm for computing gcd(m, n)

Step 1 Assign the value of min{m, n} to t.

Step 2 Divide m by t. If the remainder of this division is 0, go to Step 3; otherwise, go to Step 4.

Step 3 Divide n by t. If the remainder of this division is 0, return the value of t as the answer and stop; otherwise, proceed to Step 4.

Step 4 Decrease the value of t by 1. Go to Step 2.

```
t = min(m,n)
while t ≠ 1
  if m%t==0 & n%t==0
      return t
  else t--
```

if t==1 no gcd

What is an algorithm?

- **The third algorithm for computing the greatest common divisor (Middle-school procedure).**

Middle-school procedure for computing gcd(m, n)

Step 1 Find the prime factors of m.

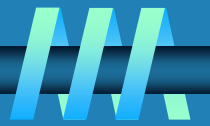
Step 2 Find the prime factors of n.

Step 3 Identify all the common factors in the two prime expansions found in Step 1 and Step 2. (If p is a common factor occurring p_m and p_n times in m and n, respectively, it should be repeated $\min\{p_m, p_n\}$ times.)

Step 4 Compute the product of all the common factors and return it as the greatest common divisor of the numbers given.

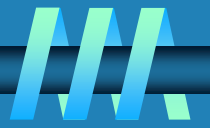
- Example
  - 60 = 2 . 2 . 3 . 5
  - 24 = 2 . 2 . 2 . 3
  - gcd(60, 24) = 2 . 2 . 3 = 12.

Important Problem Types



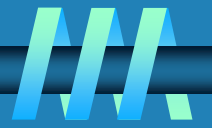
- Sorting
- Searching
- String processing
- Graph problems
- Combinatorial problems
- Geometric problems
- Numerical problems

Sorting



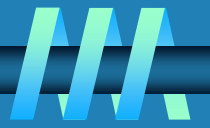
- The sorting problem is to rearrange the items of a given list in nondecreasing order.
- We sort lists of numbers, characters, and records (students, employees) using their key.
- Why would we want a sorted list?
  - A sorted list can be a required output (ranking Internet search results or ranking students by their GPAscores)
  - Sorting makes many questions about the list easier to answer (searching)
  - Sorting is used as an auxiliary step (The greedy approach)
- There are two important properties of sorting algorithms, stable and in-place.

Searching



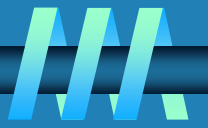
- Searching problem deals with finding a search key, in a given set (or a multiset)
- They range from sequential search to binary search.
- There is no single algorithm that fits all situations best (fast but require more memory; fast but require sorting).

String Processing



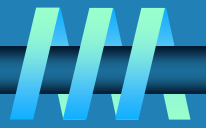
- A string is a sequence of characters from an alphabet.
- Strings may be:
  - Text strings
  - Bit strings
  - Gene sequences ({A, C, G, T})
- String matching is an example.

Graph Problems



- A graph is a collection of points called vertices, some of which are connected by line segments called edges.
- Graphs can be used for modeling applications like transportation, communication, and games.
- Some graph problems are computationally very hard
  - The traveling salesman problem
  - The graph-coloring problem

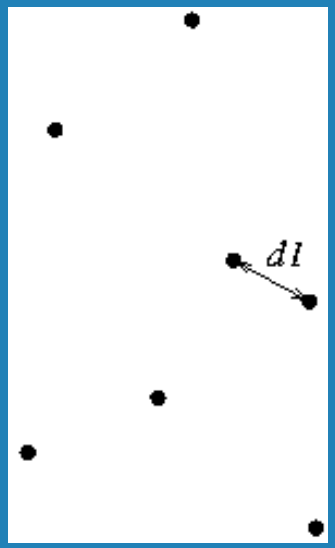
Combinatorial Problems

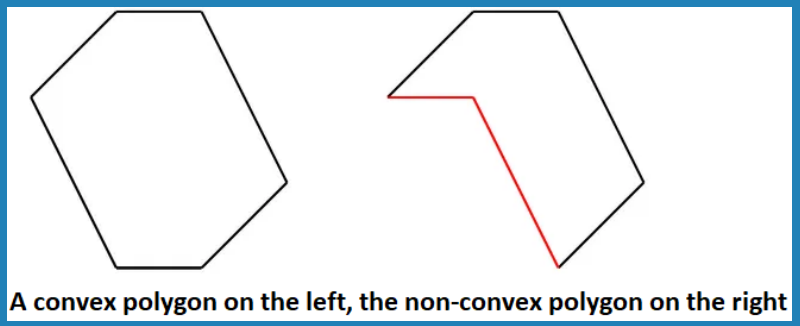


- These are problems that ask, to find a combinatorial object (additional property such as a maximum value or a minimum cost may be required)
- Combinatorial problems are the most difficult problems
  - The number of combinatorial objects typically grows extremely fast with a problem's size
  - There are no known algorithms for solving most such problems exactly in an acceptable amount of time.
  - There is a conjecture that such algorithms do not exist.
- The shortest-path problem is an exception.

Geometric Problems

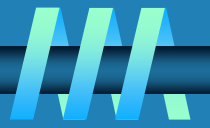
- Deal with geometric objects.
- First tackled by ancient Greek
- Now it is applied in computer graphics, robotics, and tomography.
- Examples of geometric problems are: the closest-pair problem and the convex-hull problem.





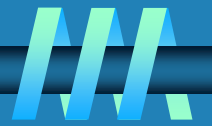
A convex polygon on the left, the non-convex polygon on the right

Numerical Problems



- Involve mathematical objects (solving equations, computing definite integrals, evaluating functions.
- Difficulties.
  - Solved approximately.
  - Require manipulating real numbers, which can be represented in a computer only approximately
  - Can lead to an accumulation of the round-off error
- First, applied in many scientific and engineering applications.
- Recently, applied in business applications.

Fundamental data structures

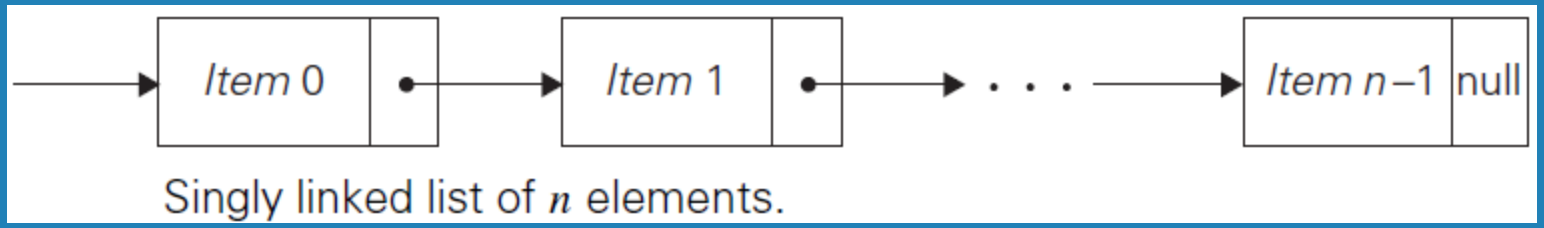


- **Data types**
  - **Elementary**
  - **Data structures**

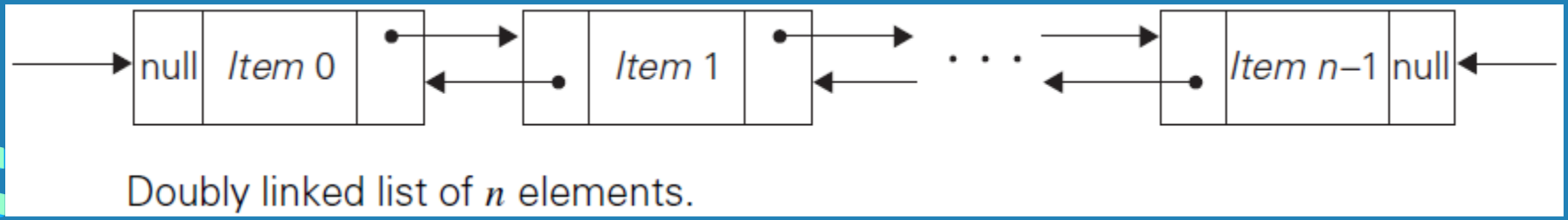
Linear Data Structures

- **Array**

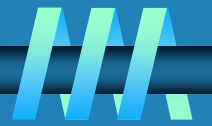
- **Linked list**



- **Doubly linked list**



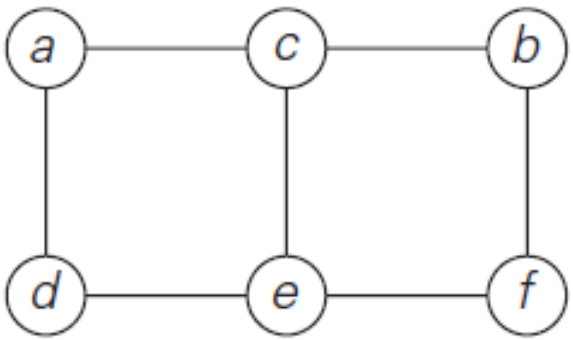
Linear Data Structures



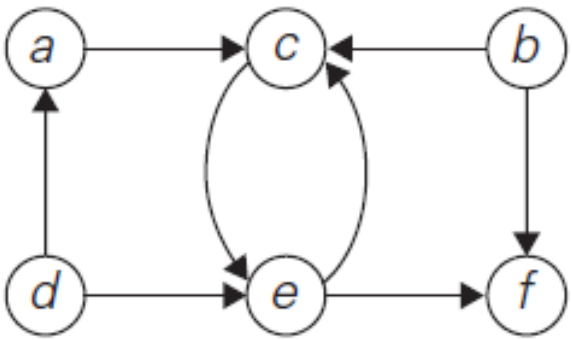
- List
  - Stack
  - Queue
    - Priority queue

Graphs

- **Graph**
  - **Undirected**
  - **Directed (Digraph)**



(a)

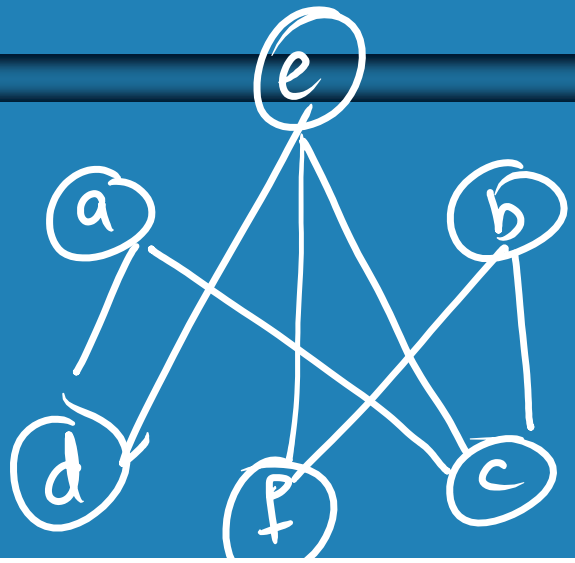


(b)

FIGURE 1.6 (a) Undirected graph. (b) Digraph.

Graphs

- **Graph Representations**
  - **Adjacency matrix**
  - **Adjacency lists**



	a	b	c	d	e	f
a	0	0	1	1	0	0
b	0	0	1	0	0	1
c	1	1	0	0	1	0
d	1	0	0	0	1	0
e	0	0	1	1	0	1
f	0	1	0	0	1	0

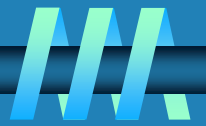
(a)

a	→	c	→	d			
b	→	c	→	f			
c	→	a	→	b	→	e	
d	→	a	→	e			
e	→	c	→	d	→	f	
f	→	b	→	e			

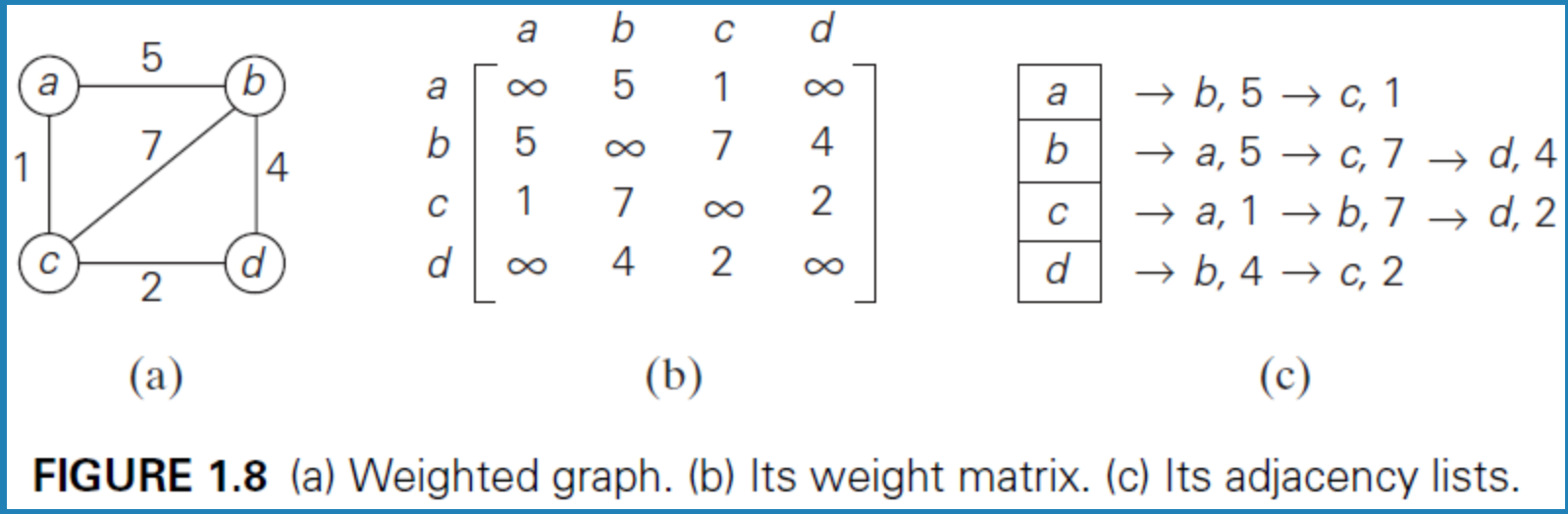
(b)

(a) Adjacency matrix and (b) adjacency lists of the graph in Figure 1.6a.

Graphs



- **Weighted Graphs (or weighted digraph)**



	a	b	c	d
a	∞	5	1	∞
b	5	∞	7	4
c	1	7	∞	2
d	∞	4	2	∞

a	→ b, 5 → c, 1
b	→ a, 5 → c, 7 → d, 4
c	→ a, 1 → b, 7 → d, 2
d	→ b, 4 → c, 2

Graphs

- Paths and Cycles
- Connected graph
- Acyclic graph

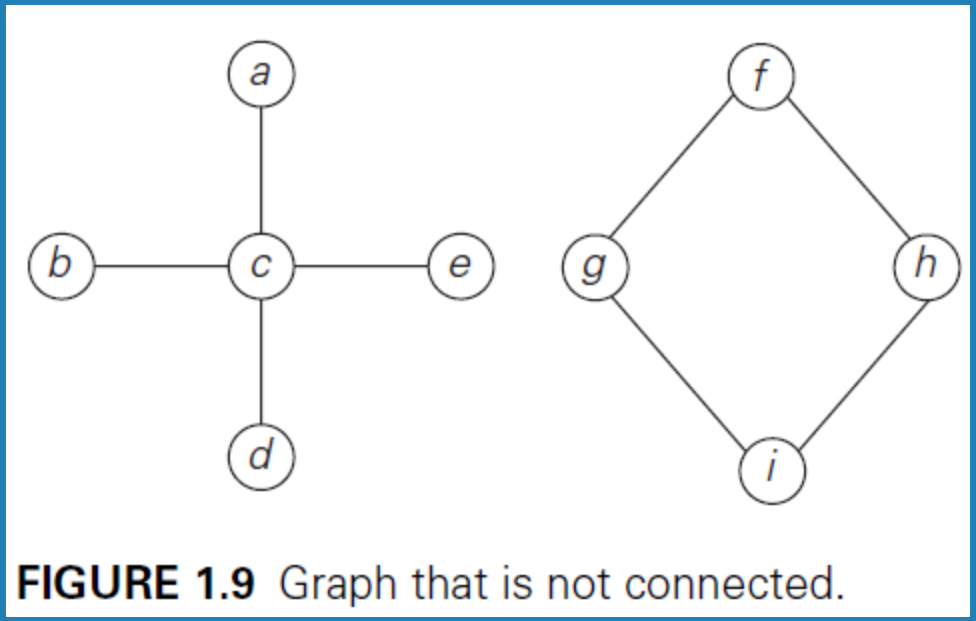


FIGURE 1.9 Graph that is not connected.

Trees

- Tree
- Forest

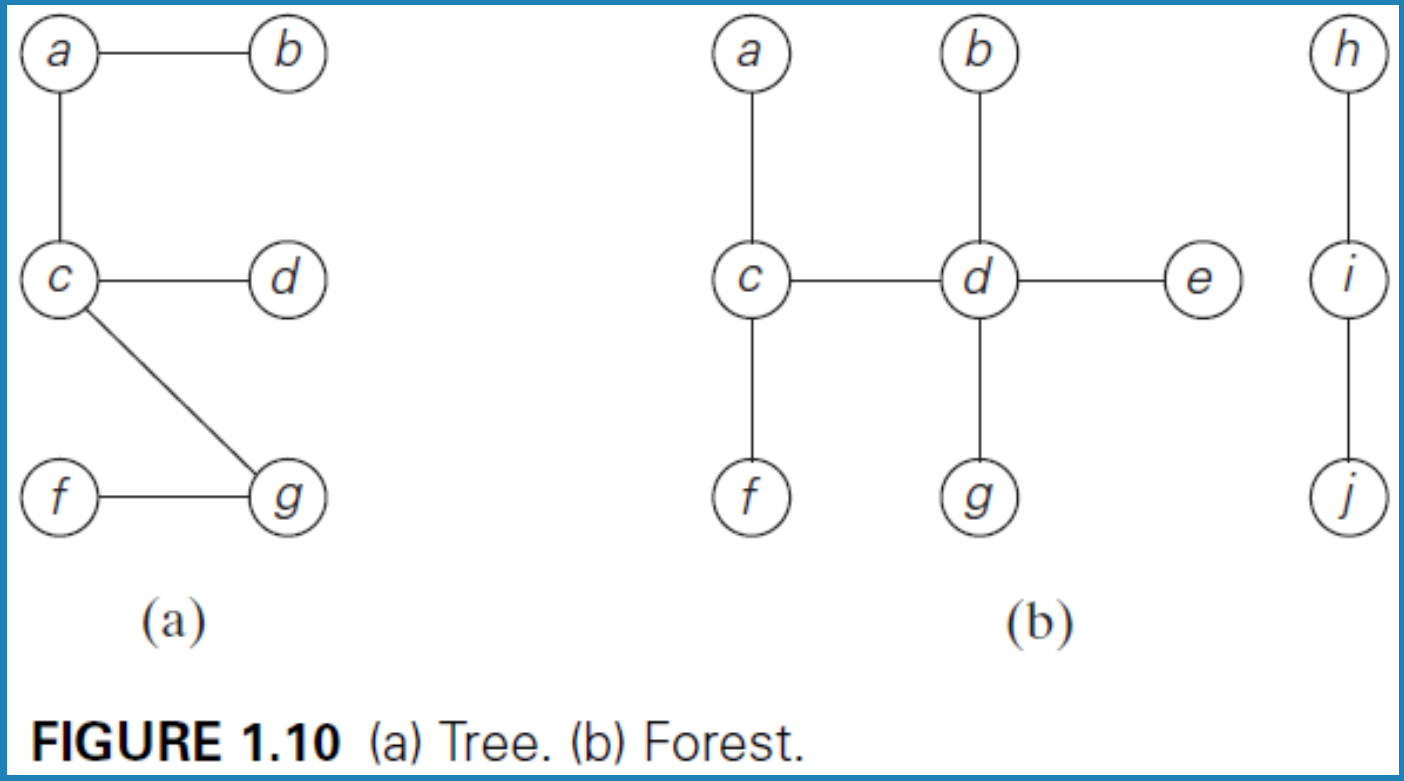


FIGURE 1.10 (a) Tree. (b) Forest.

Trees

- **Rooted Trees**

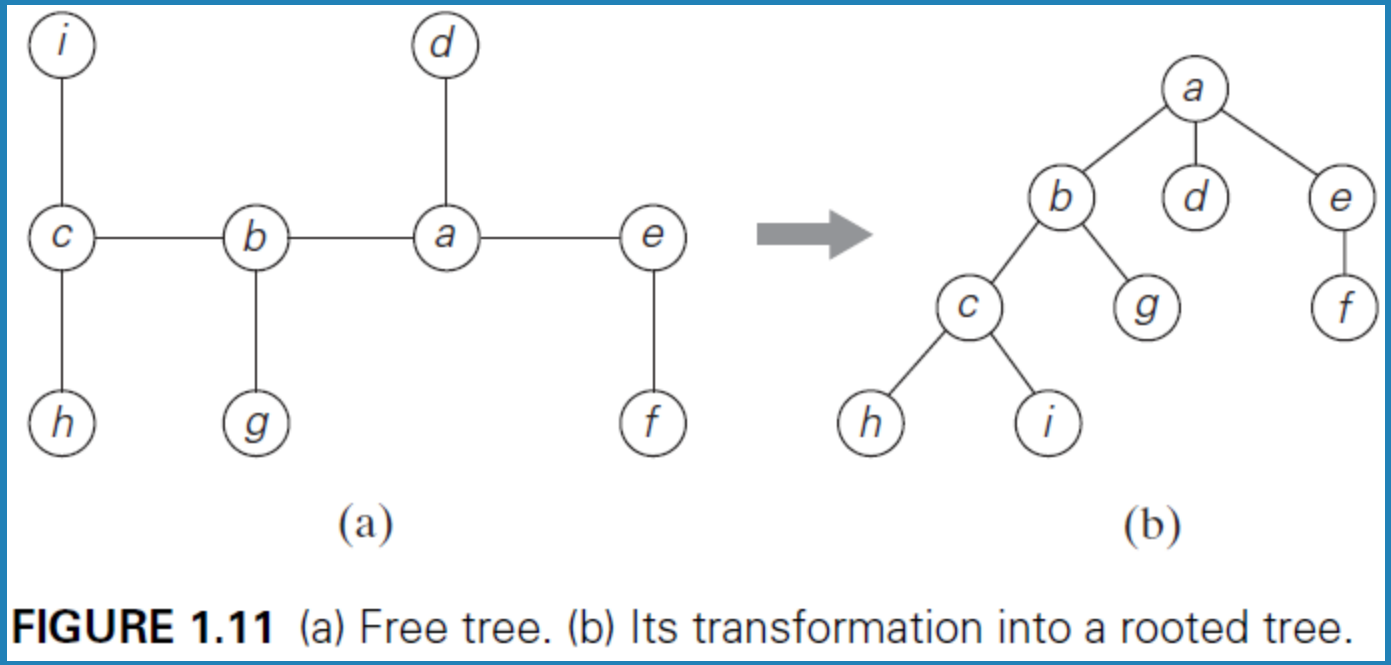


FIGURE 1.11 (a) Free tree. (b) Its transformation into a rooted tree.

Trees

- Ordered Trees
- Binary tree

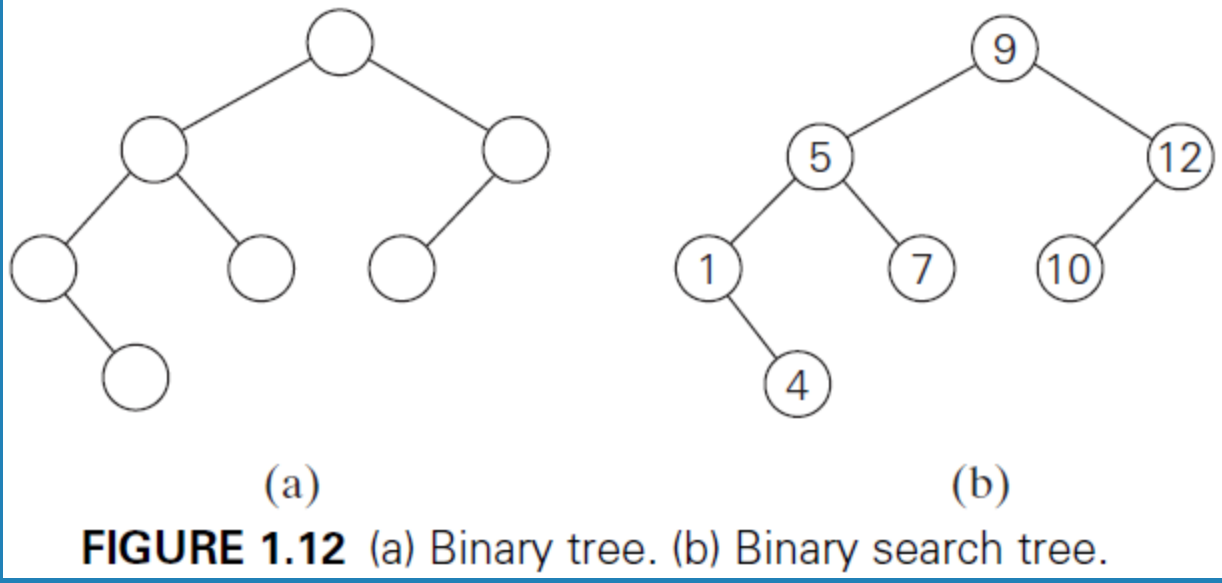


FIGURE 1.12 (a) Binary tree. (b) Binary search tree.

- $\lfloor log_2 n \rfloor \leq h \leq n - 1$

Trees

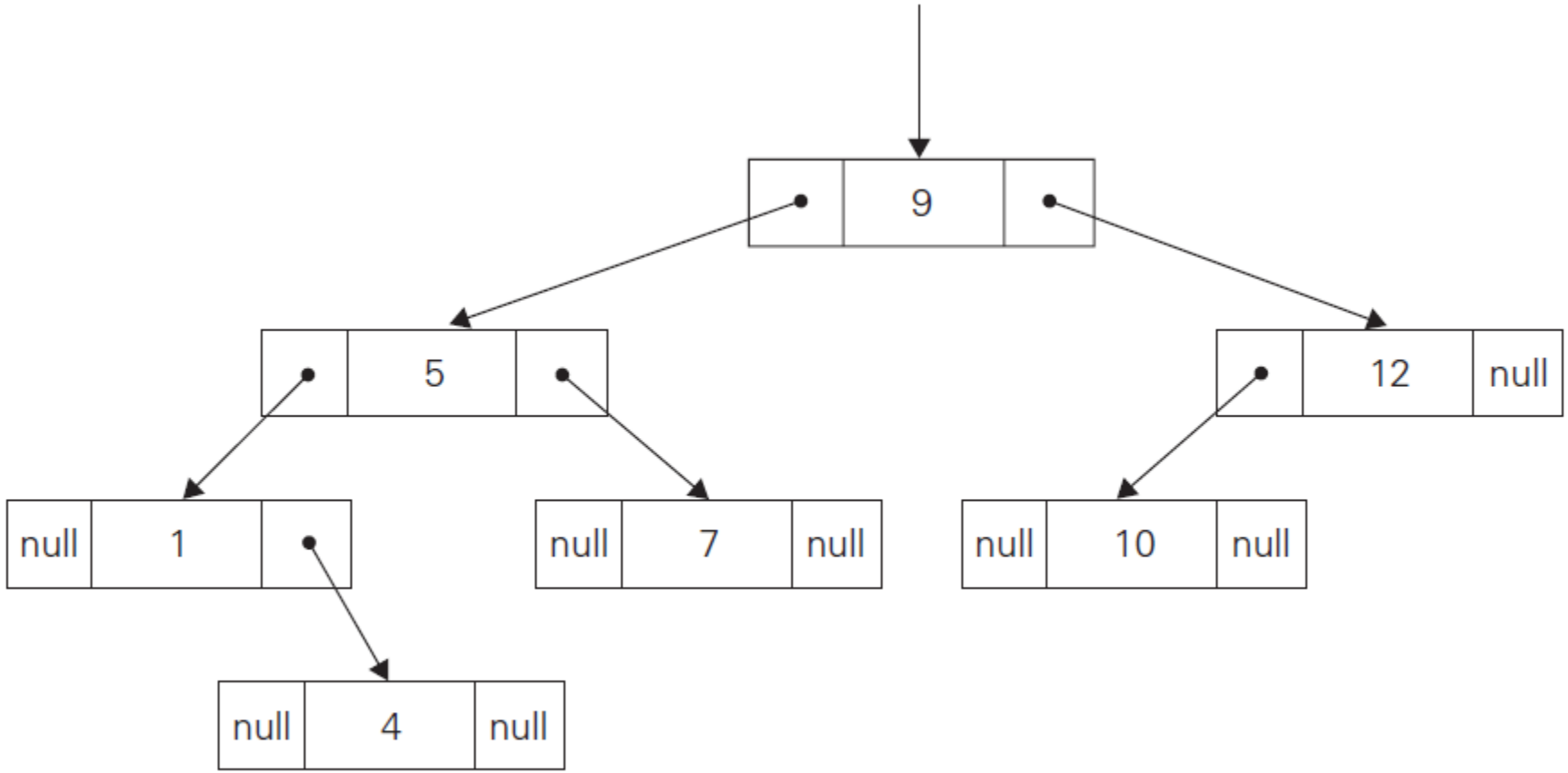
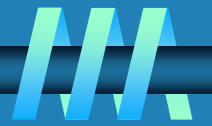


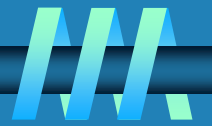
FIGURE 1.13 Standard implementation of the binary search tree in Figure 1.12b.

Sets and Dictionaries



- Set.
- Set operations
  - Check membership
  - Union
  - Intersection
- Set implementation
  - universal set & bit string representation
  - List.

Sets and Dictionaries



- **Dictionary**
- **Abstract Data Type (ADT)**