Introduction

What is an algorithm?

- A sequence of unambiguous instructions for solving a problem

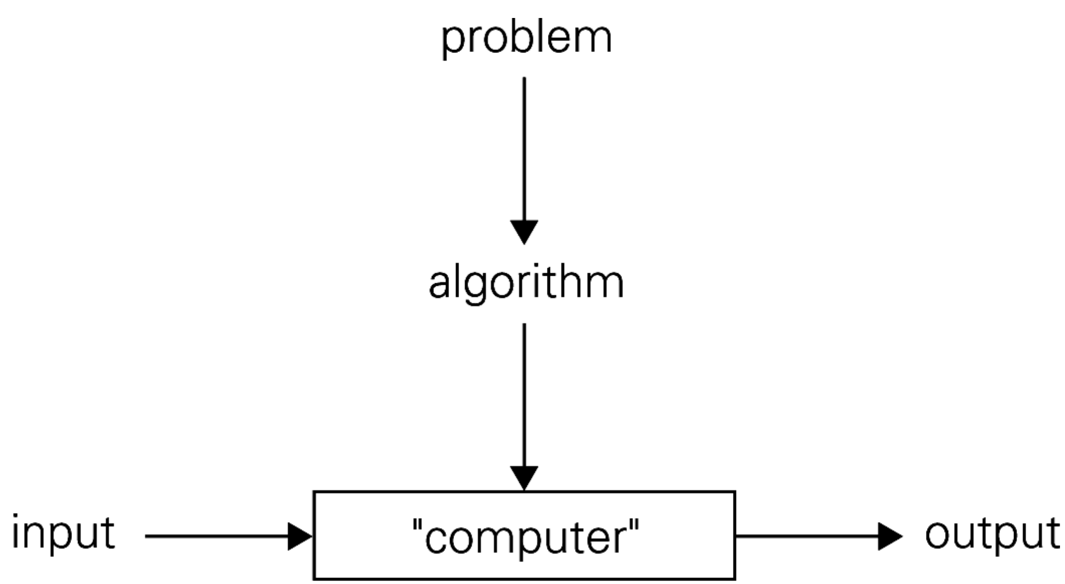

FIGURE 1.1 Notion of algorithm

Why study algorithms?

- As a computer professional:
  - Need to know a standard set of important algorithms
  - Be able to design and evaluate new algorithms
  - The study of algorithms is the core of computer science
- It is indispensable in almost all aspects of our lives
- It is useful for us to developing analytical skills

Important Points to Remember

- Nonambiguity requirement
- Carefully specify the range of inputs
- Same algorithm can have different representations
- Same problem can be solved with different algorithms that may have dramatically different speeds

An example

- **Problem**: computing the greatest common divisor of two integers, m and n.

- **Euclid's algorithm**:

  gcd(m, n) = gcd(n, m mod n)

  gcd(m, 0) = m

gcd(60,24) = ?, gcd(60,0) = ?, gcd(0,0) = ?

Two Descriptions

Euclid's algorithm:

$$\gcd(m, n) = \gcd(n, m \bmod n) \qquad \gcd(m, 0) = m$$

Step 1: If $n = 0$, return m and stop; otherwise go to Step 2
Step 2: Divide m by n and assign the value fo the remainder to r
Step 3: Assign the value of n to m and the value of r to n. Go to Step 1.

```
while n ≠ 0 do
    r ← m mod n
    m← n
    n ← r
return m
```

Will the algorithm eventually stop? Why?

An example

- **Problem**: computing the greatest common divisor of two integers, m and n.

- **Euclid's algorithm**:

$$\gcd(m, n) = \gcd(n, m \bmod n) \qquad \gcd(m, 0) = m$$

- **Consecutive integer checking algorithms**: based on the definition – the largest integer that divides *m* and *n* evenly

Consecutive integer checking algorithms

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

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

Step 3: Divide n by t. If the remainder is 0, return t and stop;
otherwise, go to Step 4

Step 4: Decrease t by 1 and go to Step 2

Will the algorithm eventually stop? Why?

An example

- **Problem**: computing the greatest common divisor of two integers, m and n.
- **Euclid's algorithm**:

$$\gcd(m, n) = \gcd(n, m \bmod n) \qquad \gcd(m, 0) = m$$

- **Consecutive integer checking algorithms**: based on the definition – the largest integer that divides *m* and *n* evenly

- **Middle-school procedure**

Middle-school procedure

Step 1: Find the prime factorization of *m*

Step 2: Find the prime factorization of *n*

Step 3: Find all the common prime factors

Step 4: Compute the product of all the common prime factors and return it as gcd(*m*,*n*)

Is this an algorithm?

Sieve of Eratosthenes

Input: Integer $n \geq 2$

Output: List of primes less than or equal to n

for $p \leftarrow 2$ to n do $A[p] \leftarrow p$

for $p \leftarrow 2$ to $\lfloor \sqrt{n} \rfloor$ do

  if $A[p] \neq 0$ //p hasn't been previously eliminated from the list

    $j \leftarrow p * p$

    while $j \leq n$ do

      $A[j] \leftarrow 0$ //mark element as eliminated

      $j \leftarrow j + p$

Discussion

Euclid's algorithm:

$\gcd(m, n) = \gcd(n, m \bmod n) \quad \gcd(m, 0) = m$

- What is the smallest number of divisions made by Euclid's algorithm among all inputs $1 \le m, n \le 10$?

- What is the largest number of divisions made by Euclid's algorithm among all inputs $1 \le m, n \le 10$?

Discussion (1-1)

Euclid's algorithm:

$\gcd(m, n) = \gcd(n, m \bmod n)$ $\quad \gcd(m, 0) = m$

- What is the smallest number of divisions made by Euclid's algorithm among all inputs $1 \le m, n \le 10$?
  - 1 for any input pair $m \ge n \ge 1$, in which m is a multiple of n
- What is the largest number of divisions made by Euclid's algorithm among all inputs $1 \le m, n \le 10$?
  - 5 , which is made by Euclid's algorithm in computing gcd(5, 8).

Discussion (1-1)

1	2	2	2	2	2	2	2	2	2
1	1	3	2	3	2	3	2	3	2
1	2	1	3	4	2	3	4	2	3
1	1	2	1	3	3	4	2	3	3
1	2	3	2	1	3	4	5	4	2
1	1	1	2	2	1	3	3	3	4
1	2	2	3	3	2	1	3	4	4
1	1	3	1	4	2	2	1	3	3
1	2	1	2	3	2	3	2	1	3
1	1	2	2	1	3	3	2	2	1

gcd(5, 8)

Algorithm Design and Analysis

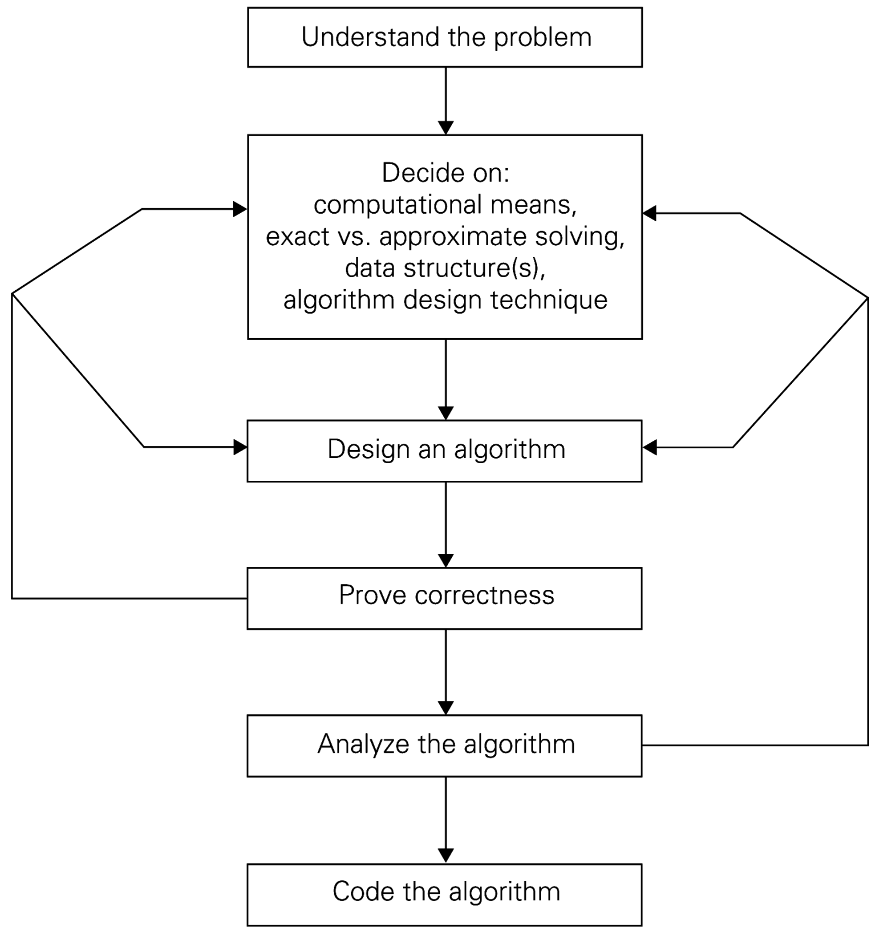

FIGURE 1.2 Algorithm design and analysis process

Algorithm Design Techniques

- Brute force
- Divide and conquer
- Decrease and conquer
- Transform and conquer
- Space and time tradeoffs
- Greedy Approach
- Dynamic Programming
- Iterative Improvement
- Backtracking
- Branch and bound

Analysis of Algorithm

- How good is an algorithm?
  - time efficiency
  - space efficiency

- Does there exist a better algorithm?
  - lower bounds
  - optimality

Fundamental data structures

- list
  - array
  - linked list
  - string
- stack
- queue
- priority queue

- graph
- tree
- set and dictionary

Important Problem Types

- sorting
- searching
- string processing
- graph problems
- combinatorial problems
- geometric problems
- numerical problems

Programming Exercise (1)

- Implement the three algorithms that compute the greatest common divisor of two integers, m and n:
  - Euclid's algorithm
  - Consecutive integer checking algorithms
  - Middle-school procedure

- Compare the performance of these algorithms. Please try on different pairs of integers and record your findings.
  - Compare number of basic operations such as addition, subtractions, divisions, …

  or

  - Measuring Computing Times: www.cs.rpi.edu/~musser/gp/timing.html
  - If you use python, please check https://realpython.com/python-timer/