Sort Numbers

3, 4, 2, 1
3, 2, 4, 1
2, 3, 4, 1
2, 3, 1, 4
2, 1, 3, 4
1, 2, 3, 4

Bubble Sort!

- **Can be explained to a child.**
- **More or less efficient.**

Is 193 a Number Prime?

First Idea:
For all numbers up to 193^(1/2), check whether it is a factor of 193.

-> Not too efficient for 2034324308234234 82133!

Not possible for 2^30402457-1

Prime Factorization?

Prime Factorization of 203432430823423482133??

No efficient solution is known!!

It took more than 50 years of computer time to factorize a 200 digit number!

It is assumed that there is no easy solution for prime factorization. In fact, most of the cryptography used today is based on the assumption that prime factorization is a hard problem and that it cannot be done efficiently.

Fundamental Questions

- What can a computer do?
- What can a computer do with limited resources?

Functions

- **Function:** A correspondence between a collection of possible input values and a collection of possible output values so that each possible input is assigned a single output

Functions (continued)

- **Computing a function:** Determining the output value associated with a given set of input values
- **Noncomputable function:** A function that cannot be computed by any algorithm

Figure 12.1 **An attempt to display the function that converts measurements in yards into meters**

Yards (input)	Meters (output)
1	0.9144
2	1.8288
3	2.7432
4	3.6576
5	4.5720
.	.
.	.
.	.

The Halting Problem

- Given the encoded version of any program, return 1 if the program is self-terminating, or 0 if the program is not.

Figure 12.6 **Testing a program for self-termination**

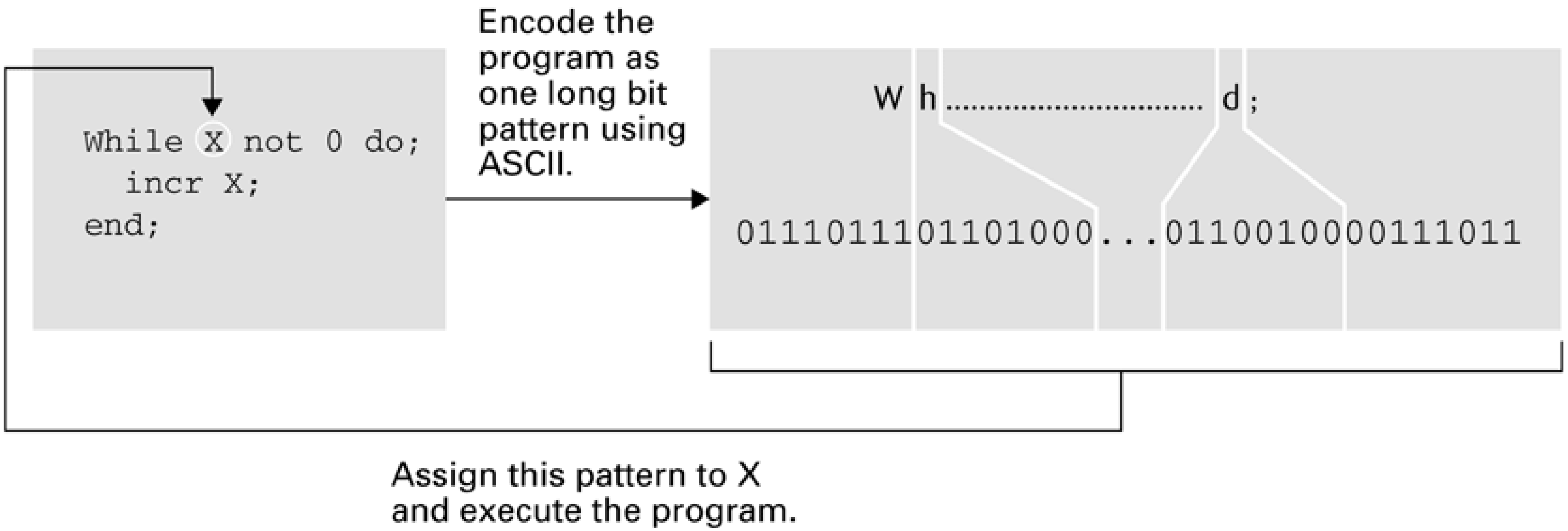
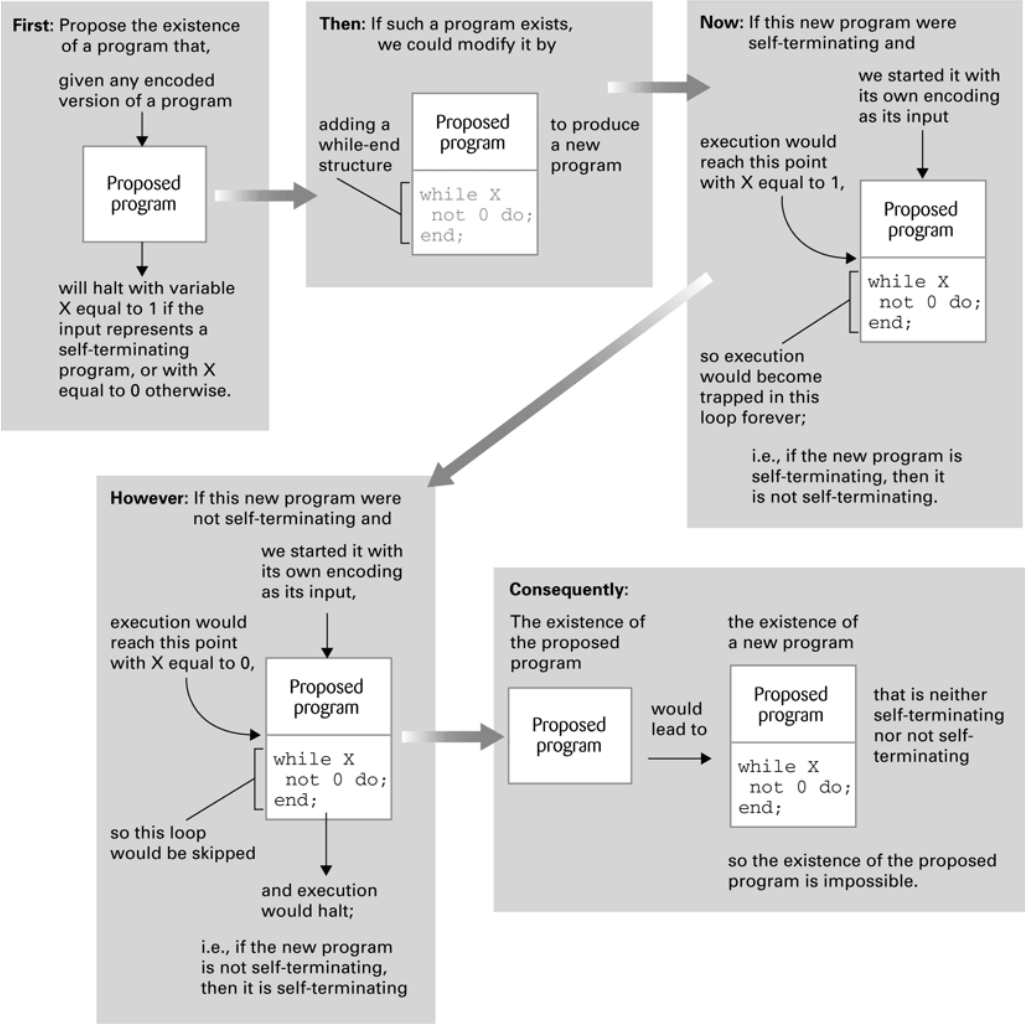


Figure 12.7 **Proving the unsolvability of the halting program**



Complexity of Problems

- **Time Complexity:** The number of instruction executions required
  - Unless otherwise noted, “complexity” means “time complexity.”
- A problem is in class O(f(n)) if it can be solved by an algorithm in Θ(f(n)).
- A problem is in class Θ(f(n)) if the best algorithm to solve it is in class Θ(f(n)).

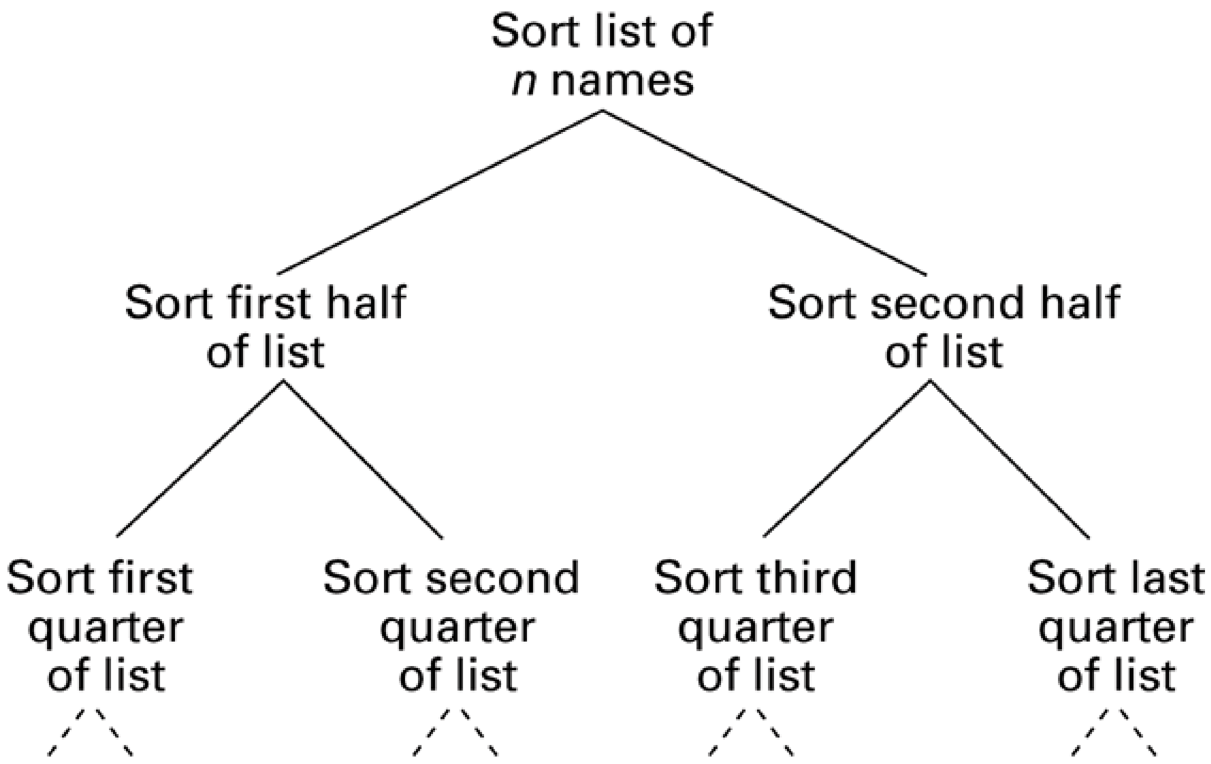
Figure 12.8 **A procedure MergeLists for merging two lists**

```
procedure MergeLists (InputListA, InputListB, OutputList)

if (both input lists are empty) then (Stop, with OutputList empty)
if (InputListA is empty)
  then (Declare it to be exhausted)
  else (Declare its first entry to be its current entry)
if (InputListB is empty)
  then (Declare it to be exhausted)
  else (Declare its first entry to be its current entry)
while (neither input list is exhausted) do
     (Put the "smaller" current entry in OutputList;
      if (that current entry is the last entry in its corresponding input list)
         then (Declare that input list to be exhausted)
         else (Declare the next entry in that input list to be the list's current entry )
     )

Starting with the current entry in the input list that is not exhausted,
     copy the remaining entries to OutputList.
```

Figure 12.9 **The merge sort algorithm implemented as a procedure MergeSort**

```
procedure MergeSort (List)

if (List has more than one entry)
  then (Apply the procedure MergeSort to sort the first half of List;
        Apply the procedure MergeSort to sort the second half of List;
        Apply the procedure MergeLists to merge the first and second
           halves of List to produce a sorted version of List
       )
```

Figure 12.10 **The hierarchy of problems generated by the merge sort algorithm**



Turing Machine Operation

Figure 12.2 **The components of a Turing machine**

- Inputs at each step
  - State
  - Value at current tape position
- Actions at each step
  - Write a value at current tape position
  - Move read/write head
  - Change state

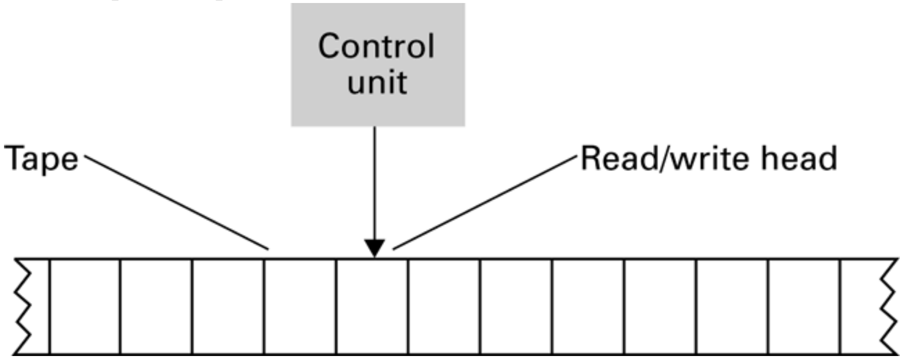


Figure 12.3 **A Turing machine for incrementing a value**

*1 0 1 *

↑

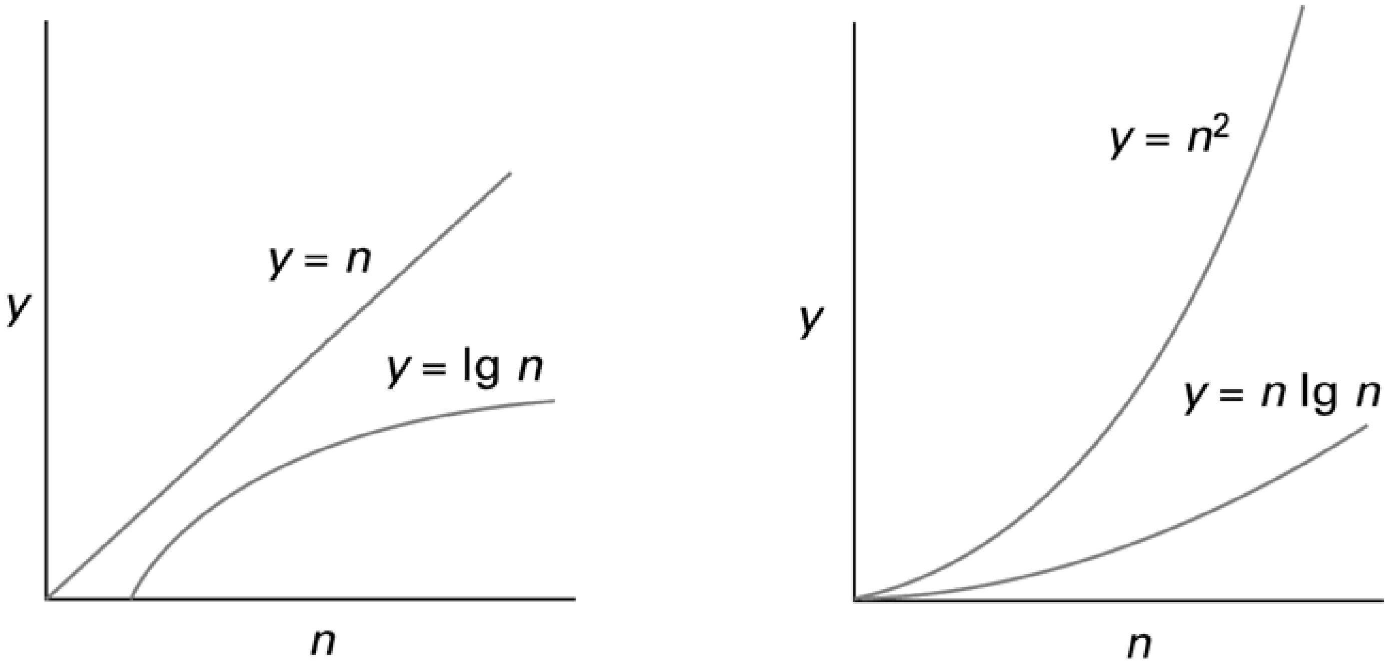
current state

Current state	Current cell content	Value to write	Direction to move	New state to enter
START	*	*	Left	ADD
ADD	0	1	Right	RETURN
ADD	1	0	Left	CARRY
ADD	*	*	Right	HALT
CARRY	0	1	Right	RETURN
CARRY	1	0	Left	CARRY
CARRY	*	1	Left	OVERFLOW
OVERFLOW	(Ignored)	*	Right	RETURN
RETURN	[illegible]	0	Right	RETURN
RETURN	*	1	Right	RETURN
RETURN	*	*	No move	HALT

Church-Turing Thesis

- The functions that are computable by a Turing machine are exactly the functions that can be computed by any algorithmic means.

Figure 12.11 **Graphs of the mathematical expressions *n*, lg *n*, *n* lg n, and n^2**



a. *n* versus lg *n*

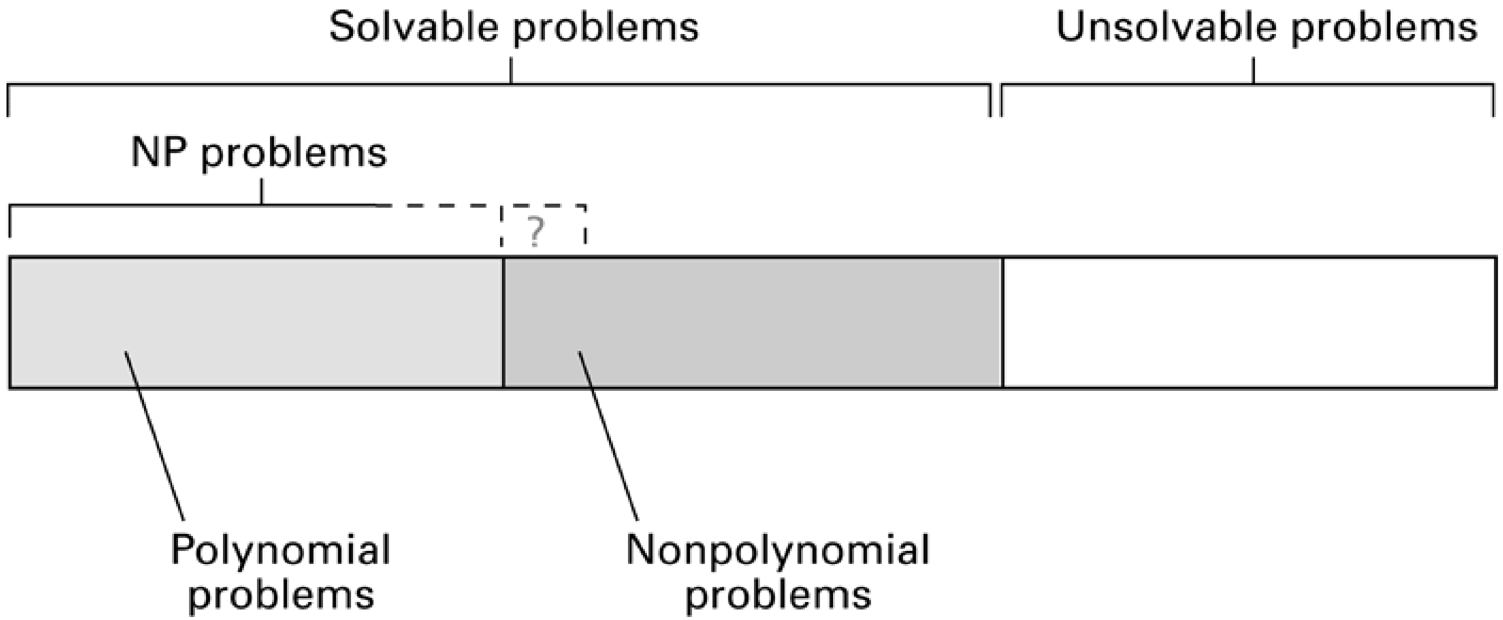
b. n^2 versus *n* lg *n*

P versus NP

- **Class P:** All problems in any class $\Theta(f(n))$, where f(n) is a polynomial
- **Class NP:** All problems that can be solved by a nondeterministic algorithm in polynomial time

  Nondeterministic algorithm = an “algorithm” whose steps may not be uniquely and completely determined by the process state
- Whether the class NP is bigger than class P is currently unknown.

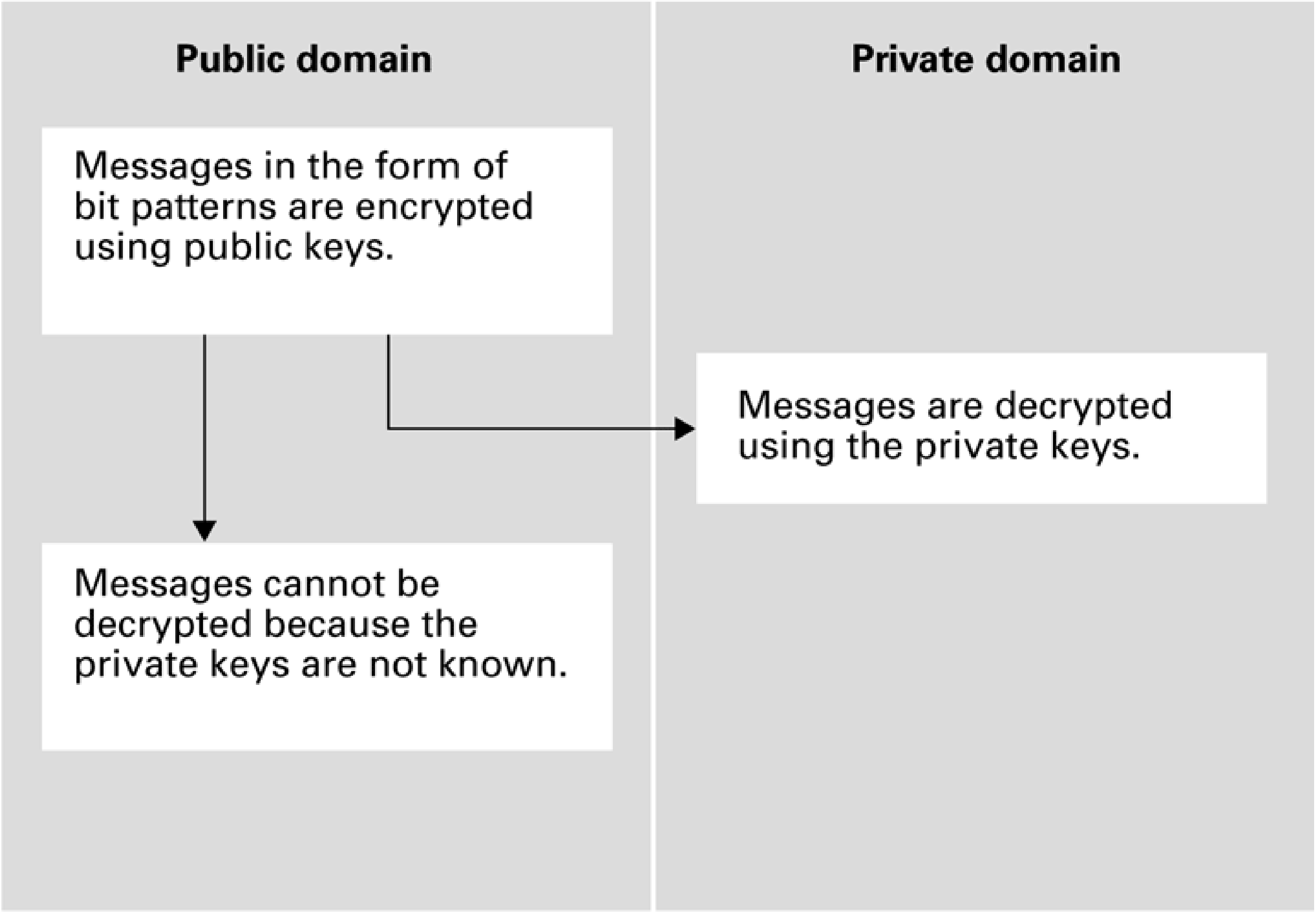
Figure 12.12 **A graphic summation of the problem classification**



Public-Key Cryptography

- **Key:** A value used to encrypt or decrypt a message
  - **Public key**: Used to encrypt messages
  - **Private key**: Used to decrypt messages
- **RSA:** A popular public key cryptographic algorithm
  - Relies on the (presumed) intractability of the problem of factoring large numbers

Figure 12.13 **Public key cryptography**



Cryptography By Factoring Large Numbers

- p,q为两个素数，m是0到pq的一个整数，对于任意的正整数k存在：$1=\mathbf{m}^{k(p-1)(q-1)}(\mathrm{mod}\ pq)$

- RSA公钥系统：选出两个不同的素数，p,q。n=pq。选出两个正整数：e,d。使得对某个正整数k，e*d=k(p-1)(q-1)+1。e,n为加密密钥，d,n为解密密钥，p,q用来构建加密系统。

- 加密：$c=m^e(\mathrm{mod}\ n)$
- 解密：$c^d(\mathrm{mod}\ n)$
  - 原理：$c^d=m^{e*d}(\mathrm{mod}\ n) = m^{k(p-1)(q-1)+1}\ (\mathrm{mod}\ n)=m^{k(p-1)(q-1)}\ (\mathrm{mod}\ n) * m^1\ (\mathrm{mod}\ n)=m\ (\mathrm{mod}\ n)= m.$

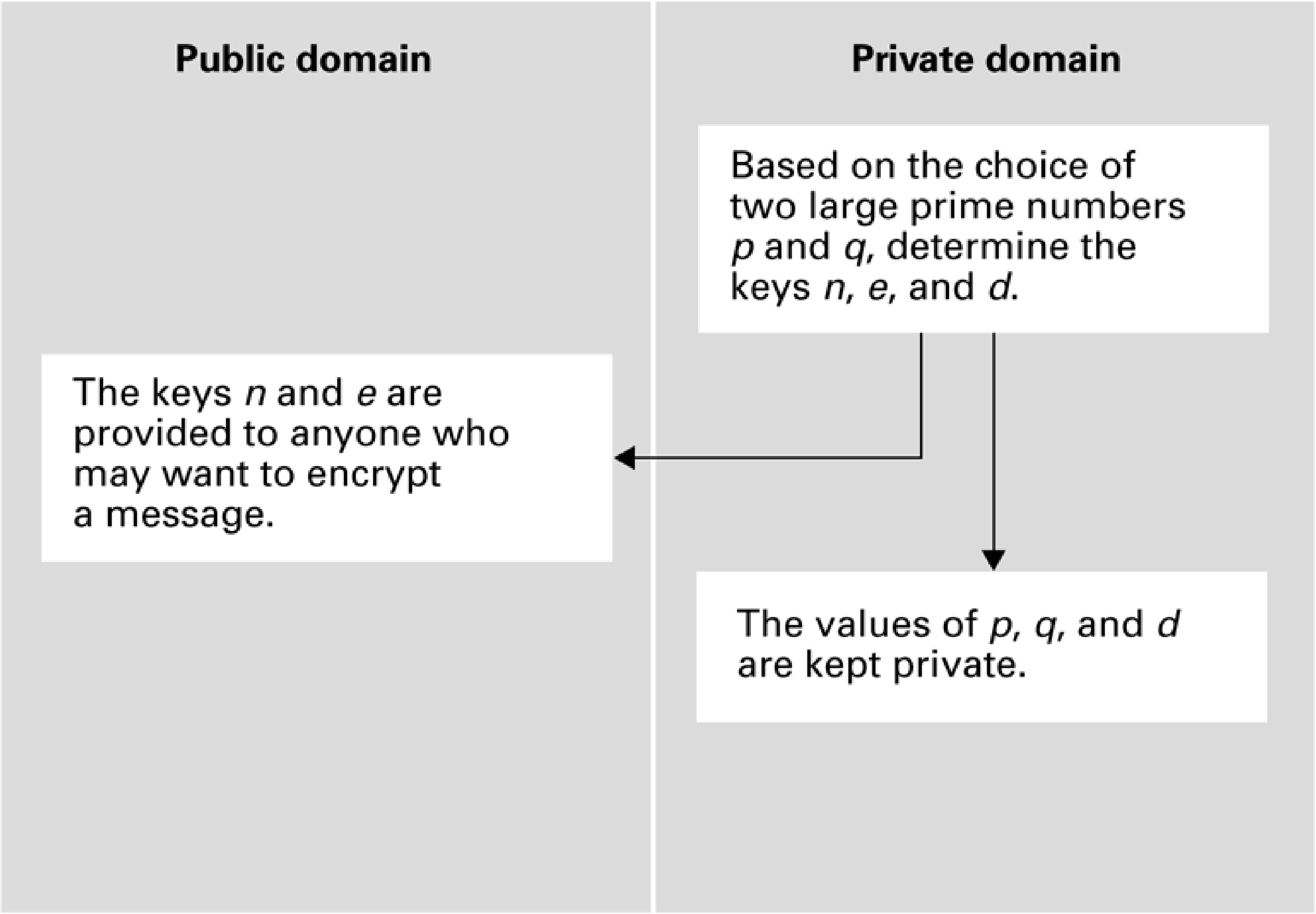
Encrypting the Message 10111

- Encrypting keys: $n = 91$ and $e = 5$
- $10111_{two} = 23_{ten}$
- $23^e = 23^5 = 6{,}436{,}343$
- $6{,}436{,}343 \div 91$ has a remainder of 4
- $4_{ten} = 100_{two}$
- Therefore, encrypted version of 10111 is 100.

Decrypting the Message 100

- Decrypting keys: d = 29, n = 91
- $100_{two} = 4_{ten}$
- $4^d = 4^{29} = 288,230,376,151,711,744$
- $288,230,376,151,711,744 \div 91$ has a remainder of 23
- $23_{ten} = 10111_{two}$
- Therefore, decrypted version of 100 is 10111.

Figure 12.14 **Establishing an RSA public key encryption system**



- Questions?