(Discrete Mathematics for Programming)

Phaphontee Yamchote

Contents

Part I

Basic Programming by Python

Chapter 1

Fundamental of Problem Solving

(problem solving)
(computational problem)

1.1 Problem Solving

. , , , , , ,
.

“ ”
( )
(

)

“

”

( ) (input) (

) (output)

(problem solving)

(problem statement)

- :
- :
- :

(computational problem)

(calculation)

1.2

4

1. (decomposition)
2. (pattern recognition)
3. (abstraction)
4. (algorithm design)

1.2.1 (decomposition)

,

$x + y + 12z = 30$ x, y z

(x, y, z)

30 0 30 $31 \times 31 \times 31 = 29791$

z

1 12 z $z = 0, 1, 2$

(30) 3

1. $z = 0$: $x + y = 30$
2. $z = 1$: $x + y = 18$
3. $z = 2$: $x + y = 6$

1 1 2

1.2.2 (pattern recognition)

1.2.3 (abstraction)

1.2.4 (algorithm design)

Chapter 2

Basic Python Syntax

Part II

Basic Mathematical Reasoning and Proving

Chapter 3

Mathematics as a Language

( )

4

(1)

(2) (3) (4)

“a

” S S a

a $\in S$

X

$S \subseteq X$

3.0.1:

A B A B $A \subseteq B$ x $x \in A$
$x \in B$

(1) $\mathrm{a} \in S$ (2) $S \subseteq X$ 3.0.1

- S A X B
- $\mathrm{a} \in S$ $x \in A$
- $S \subseteq X$ $A \subseteq B$

$x \in B$ ( ) $\mathrm{a} \in X$ a

“ ”

[1]

“x ” ($P(x)$) [2]
x (
)

[1]

[2] “ ”

Chapter 4

Basic Objects in Mathematics

Chapter 5

Logic, Reasoning and Proof

3
3 (1) Logic ( ) (2)
Reasoning ( ) (3) Proof ( ) 3 3
( )

(knowledge) (skill)
(methodology)

(.4 [1])

5.1

1

5.2

5.3

Chapter 6

Recursion and Mathematical Induction

Part III

Discrete Mathematics with Programming

Chapter 7

Set Theory: with more implementation

Chapter 8

Number Theory

THEORY PART

Fundamental Theorem of Arithmetic

(cryptography)

[1] (

)

[1] Principle of Mathematics Discrete Mathematics

8.1

0

10 5 ( ) 5 x

10 $5x = 10$ x 10

5 x

8.1.1: Divisibility

m n m n k $m = nk$

$n|m$

$5|10$ 2 10 5 $k = 2$

$10 = 5 \times 2$

$n|m$ k

$m = nk$ k

$m = nk$ ($\frac{m}{n}$)

k

(

) k n m

Example 8.1.2. $25|300$

Solution. 25 300

$300/25 = 12$ 25

. $300 = 25 \times 12$ $25|300$ $\square$

Example 8.1.3. $25 \nmid 310$

Solution. 25 310
$310/25 = 12.4$ ( )

. n $310 = 25n$ ( )
$310 = 25 \times 12 + 10$

$$\begin{aligned} 25n &= 25 \times 12 + 10 \\ 25n - 25 \times 12 &= 10 \\ 25(n-12) &= 10 \end{aligned}$$

x $0 \leq 25x < 25$ $x = 0$
$0 \leq 10 = 25(n-12) < 25$ $n - 12 = 0$
$10 = 25(n-12) = 25 \times 0 = 0$
n $310 = 25n$ $\square$
(
)

Exercise 8.1.4. (
Proof Part)

	$\forall x, xRx$			
	$\forall x \forall y \forall z, xRy \wedge yRz \to xRz$			
	$\forall x \forall y, xRy \to yRx$			
	$\forall x \forall y, xRy \to \neg yRx$			
	$\forall x \forall y, xRy \wedge yRx \to x = y$			

Solution. ...

8.1.5:

m, n, p

1. $1|m$ $m|m$
2. $m \neq 0$ $m|0$
3. $m|n$ $m|np$
4. $p \neq 0$ $m|n$ $pm|pn$
5. $m|n$ $m|p$ $m|(n+p)$
6. $m|n$ $m|p$ $m|(xn+yp)$ x, y
7. $m|n$ $|m| \leq |n|$

: [2]

1. 1 1
   $1 \cdot n = n$ n
2. 0 0 0
3. ($\frac{n}{m}$)
   (p) $\frac{np}{m}$
   p p
   (divisibility is preserved under numerator multiplication)
4. $\frac{n}{m} = \frac{pn}{pm}$
5. $\frac{n+p}{m} = \frac{n}{m} + \frac{p}{m}$
6. $xn + yp$ (linear combination) 3 5

---

[2]

( ) ( )
($\frac{n}{m} = \frac{pn}{pm}$)

8.2 : Division Algorithm

( )

(-21 5)

!

$$= \times +$$

( )
( ) (: well-defined)

8.2.1:

m n $n \neq 0$ q r $m = nq + r$
$0 \leq r < |n|$

8.2.2: Division Algorithm

m n $n \neq 0$ q r 8.2.1 (quotient)
(remainder)

PROOF PART

Exercise 8.1.4

. content... $\square$

8.1.5

. content... $\square$

8.2.1

. q r $m \geq 0$ $n > 0$ (?: 1)

$m = 0$ (m) n $0 = n \times 0 + 0$

m m

$m > 0$ m $n > 0$ q r $0 \leq r < n$ $m = nq + r$ $m + 1$ 2 [3] (1) $0 \leq r \leq n - 2$ (2) $r = n - 1$

1) $0 \leq r \leq n - 2$: $m + 1 = nq + r + 1 = nq + (r + 1)$ $0 < 0 + 1 \leq r + 1 \leq n - 2 + 1 = n - 1$ q $r + 1$

2) $r = n - 1$: $m+1 = nq+r+1 = nq+n-1+1 = nq+n = n(q+1)+0$ $q + 1$ 0

m n q r $m = nq + r$ $0 \leq r < n$

q' r' $m = nq' + r'$ $0 \leq r' < n$ $nq + r = nq' + r'$ $n(q - q') = r' - r$ $r, r' \in \{0, 1, \ldots, n - 1\}$ $0 \leq |r' - r| < n$ $0 \leq n|q' - q| < n$ $|q' - q| = 0$ $q = q'$ $r' - r = n(q - q') = n \times 0 = 0$ $r = r'$ $\square$

---

[3] 1 m $m + 1$ $n - 1$ n n

8.3 Theory Exercise

1. (8.2.1) $m \geq 0$ $n > 0$ $m = nq + r$ $0 \leq r < |n|$ q' r' $0 \leq r' < |n|$ $-m = nq' + r'$ ($m = (-n)q' + r'$ $-m = (-n)q' + r'$)

2. 8.2.1

PROGRAMMING PART

8.4 programming:

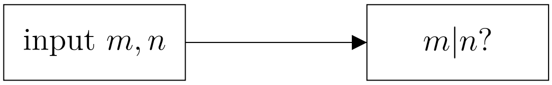

Figure 8.1: input m, n output

2 m

n "%" built-in Python

0 code

```
m%n == 0
```

code True False

`isDivisible` code

m n m n k $m = nk$

m n k

k k $m = nk$

Not complete divisibility checking

```
k = 1
while m != n*k:
    k += 1
```

```
# after exiting from while-loop, k should be an integer such that m = nk,
# i.e. n is a factor of m
```

$$m \nmid n \iff \quad k \in \mathbb{Z} \qquad m \neq nk$$

k k

-10 5 loop

$k = 1$ 1 k $k = -2$

k

m n $m|n$ $|n| \leq |m|$

m $|k| \leq |m|$

$k \in \{-m, -m+1, \ldots, -1, 0, 1, \ldots, m-1, m\}$

k

$$m|n \iff \quad k \in \{-m, -m+1, \ldots, m-1, m\} \qquad m = nk$$

8.4.1

Check divisibility

```python
def isDivisible_ver1(m,n):
    qoutList = range(-m,m+1)
    for k in qoutList:
        if m = n*k:
            return True
```

```
    return False
```

isDivisible True

False

8.4.2

$$m = nk$$

$$
\begin{aligned}
(-m) = nk &\iff m = n(-k) \\
m = (-n)k &\iff m = n(-k) \\
(-m) = (-n)k &\iff m = nk
\end{aligned}
$$

k

m n

$k \in \{1, 2, \ldots, m-1, m\}$

Check divisibility by positive

```
def isDivisible_ver2(m,n):
    if m < 0:
        m = -m
    if n < 0:
        n = -n
    qoutList = range(1,m+1)
```

```
    for k in qoutList:
        if m = n*k:
            return True
    return False
```

isDivisible_ver2

8.4.3

$k \cdot n := n + n + \cdots + n$ (k) k

n m m m

Check divisibility addition version

```
def isDivisible_ver3(m,n):
    product = 0
    while product < m:
        product += n
    if product == m:
        return True
    else:
        return False
```

n ( )

Check divisibility subtraction version

```
def isDivisible_ver4(m,n):
    while m >= n:
        m -= n
    if m == 0:
        return True
    else:
```

```
        return False
```

8.4.4

`isDivisible_ver4` m n 1
$m - n$ n

$$\texttt{isDivisible\_recur(m,n)} = \texttt{isDivisible\_recur(m - n,n)}$$

while-loop `isDivisible_ver4` m
n 0 0
0

$$\texttt{isDivisible\_recur(m,n)} = \begin{cases} \texttt{True} & \text{if } m = 0 \\ \texttt{False} & \text{if } 0 < m < n \end{cases}$$

Check divisibility recursion

```
def isDivisible_recur(m,n):
    if m < n:
        if m == 0:
            return True
        else:
            return False
    else:
        return isDivisible_recur(m-n,n)
```

8.5 programming:

8.5.1

n Figure ??

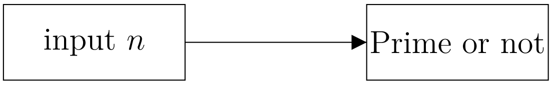

Figure 8.2: input n output

n 1 n n

1 1

n

n Figure ?? n

Figure 8.3: text

Check if it is prime

```
# assume we have a list `factorList` which is a list of all
↪  factors of n
factorList == [1,n]
```

True n 2 1 n n

n False

Figure 8.4: text

n (Python) 1 n

n `factorList`

Create factorList

```
factorList = []
for m in range(1,n+1):
    if isDivisible(n,m):
        factorList.append(m)
```

n

Check prime

```
def isPrime(n):

    factorList = []
    for m in range(1,n+1):
        if isDivisible(n,m):
            factorList.append(m)

    prime = (factorList == [1,n])
    return prime
```

1 n n

memory

1 built-in data

structure Python implement
array
implement Python

8.5.2 n

$$
\begin{aligned}
n > 1 \quad & \iff \quad 1 \quad n \quad n \\
& \iff \quad k \notin \{1, n\} \quad k \quad n \\
& \iff \quad k = 2, ..., n-1 \quad k \quad n
\end{aligned}
$$

$$n > 1 \quad \iff \quad k = 2, ..., n-1 \quad k \quad n$$

n 2 $n-1$
n ($\exists x, P(x)$)

Check prime version2

```python
def isPrime_ver2(n):

    prime = True                #set as default to be prime
    for k in range(2,n):
        if isDivisible(n,k):    #check if factor
            prime = False       #if k is a factor, set it to be
             ↪  not prime
            break               #stop for loop

    return prime
```

`isPrime_ver3` while-loop

8.5.3

`isPrime` $O(n)$ `isPrime_ver2`
$O(n)$ n 2 $n-1$

$\sqrt{n}$

n p n $p \leq \sqrt{n}$ n

$\sqrt{n}$ n
2 $\lfloor\sqrt{n}\rfloor$

n `isPrime_ver2`

Check prime version2.1

```
import math
def isPrime_ver2_1(n):
    prime = True                #set as default to be prime
    upper = int(math.sqrt(n))
    for k in range(2,upper):
        if isDivisible(n,k):    #check if factor
            prime = False       #if k is a factor, set it to be not prime
            break               #stop for loop
    return prime
```

8.6 programming:

Fundamental Theorem of Arithmetic

Fundamental Theorem of Arithmetic

n $p_1 < p_2 < \cdots < p_n$ $a_1, a_2, \ldots, a_n$

$$n = p_1^{a_1} p_2^{a_2} \cdots p_n^{a_n}$$

??

dictionary keys

values $1400 = 2^3 \times 5^2 \times 7$ `{2:3, 5:2, 7:1}`

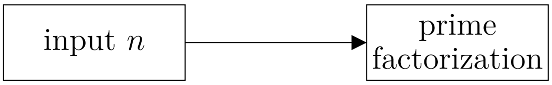

Figure 8.5: input n output
dictionary

8.6.1

n p

p

k $n = p^k \cdot A$ $p \nmid A$

8.6

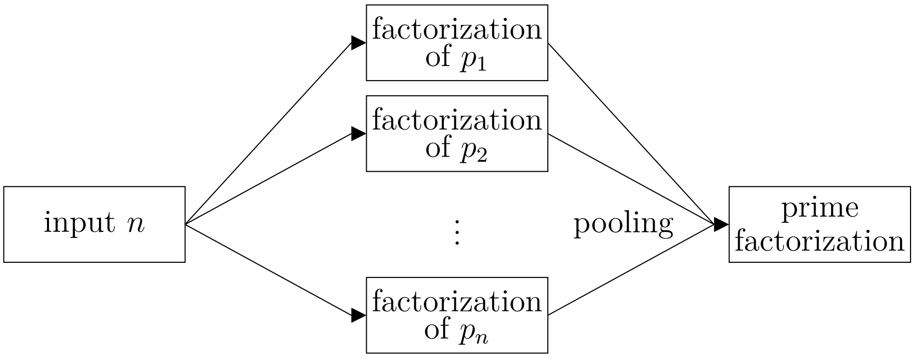

Figure 8.6: ...

n $p_1, \ldots, p_n$

n

8.7

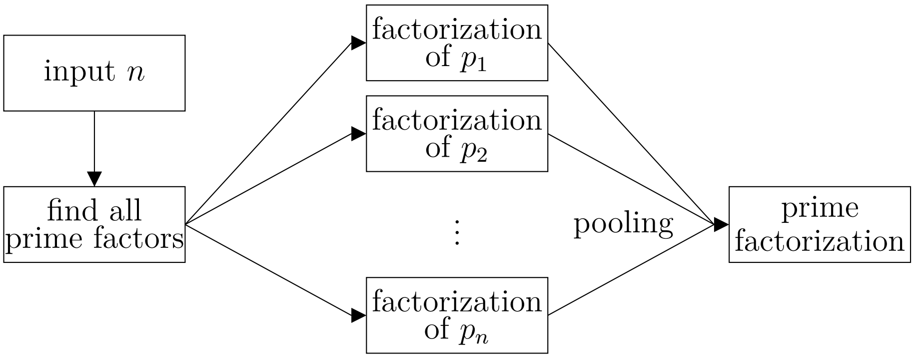

Figure 8.7: ...

for-loop

8.8

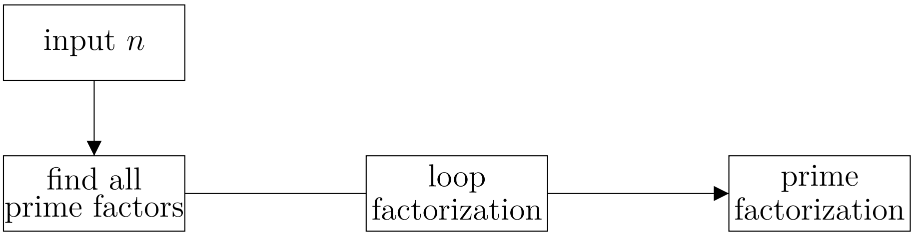

Figure 8.8: ...

n 6

n p

p

k $n = p^k \cdot A$ $p \nmid A$

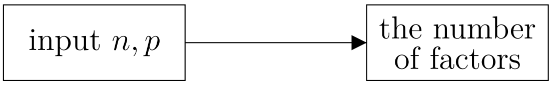

Figure 8.9: input n p output n p

“ (n%p == 0) ” (counter += 1) n = n//p

factorization of given prime p

```
def countFactor(n,p):
    count = 0
    while n%p == 0:
```

```
        count += 1
        n = n//p
    return count
```

countFactor p n ( ) findAllPrimeFactor
n n
2 8.8

Prime Factorization

```
def primeFactorize(n):
    primeList = findAllPrimeFactor(n)
    resultDict = {}
    for p in primeList:
        resultDict[p] = countFactor(n,p)
    return resultDict
```

8.6.2

countFactor
n n (n/p)
p
(p_1)
n/p_1 1

$$n = \underbrace{p_1^{a_1} p_2^{a_2} \cdots p_n^{a_n}}_{\texttt{algor(n)}} = p_1 \times \underbrace{(p_1^{a_1-1} p_2^{a_2} \cdots p_n^{a_n})}_{\texttt{algor(n/p\_1)}} = p_1 \times (n/p_1)$$

dictionary
dictionary n/p_1 p_1 1
`dict[key] = dict.get(key,0) + 1` (key 0 1 1 key
1 key)

6	`minPrimeFactor`

Recursive Prime Factorization

```python
def primeFactorize_recur(n):
    if n == 1:
        return {}
    else:
        min_p = minPrimeFactor(n)
        result_dic_recur = primeFactorize_recur(n//min_p)
        result_dic_recur[min_p] = result_dic_recur.get(min_p,0) + 1
        return result_dic_recur
```

8.7 programming:

8.8 Programming Exercise

1.
   operation ( ) 1

2. `isDivisible_recur` 8.4.4
   `isDivisible_ver2`

3.

4.

5. n 1 n $O(n^{\frac{3}{2}})$
   ($O(n^2)$ ver2 print
)

6. n n

7. n p (`countFactor`)

8. n n

9. n dictionary $n!$ (caution:
   `primeFactorize` 8.6 n
   $n!$)

10. 9 n 0 $n!$

Chapter 9

Combinations

THEORY PART

9.1

9.1.1

2 p q

$p + q$

Example 9.1.1. 33 40

Solution. ...

Example 9.1.2. $A = \{a, b, c, d\}$ $B = \{\alpha, \beta, \gamma\}$ A B

Solution. ...

9.1.2

A B $A \cap B = \emptyset$

$$|A \cup B| = |A| + |B|$$

2 2

m r_1 r_2
$\ldots$ m r_m $r_1+r_2+\cdots+r_m$

$A_1, \ldots, A_m$ $A_i \cap A_j = \emptyset$
$i \neq j$

$$|A_1 \cup \cdots \cup A_m| = |A_1| + \cdots + |A_m|$$

Example 9.1.3. $|\{(x,y) \in \mathbb{Z} \times \mathbb{Z} \colon x^2 + y^2 \leq 4\}|$

Solution. ...

9.1.2

p
q pq

p q

Example 9.1.4. 33 40

Solution. ...

Example 9.1.5. $A = \{a, b, c, d\}$ $B = \{\alpha, \beta, \gamma\}$ 2 A B

Solution. ...

A B $A \times B = \{(a, b) \colon a \in A, b \in B\}$

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

Example 9.1.6. 1000 10000

Solution. ...

m r_1 r_2

$\ldots$ m r_m

$r_1 \times r_2 \times \cdots \times r_m$

$A_1, \ldots, A_m$

$$|A_1 \times \cdots \times A_m| = |A_1| \times \cdots \times |A_m|$$

Example 9.1.7. 1000 10000

Solution. ...

Example 9.1.8. n 2^n

Solution. ...

Example 9.1.9. 15 (1)
(2)

Solution. ...

Example 9.1.10. 3 4 (1)
(2)

Solution. ...

Example 9.1.11. $441,000(= 2^3 \times 3^2 \times 5^3 \times 7^2)$

Solution. ...

Example 9.1.12. $441,000$ 2 (
$1 \times 441,000$ 441×1000)

Solution. ...

Example 9.1.13. $X = \{1, 2, 3, \ldots, 10\}$ $S = \{(a, b, c) \colon a, b, c \in X, a <$
b $a < c\}$ S

Solution. ...

9.2

9.2.1

$A = \{a_1, a_2, \ldots, a_n\}$ n $0 \le r \le n$ r
A (r-permutation) r A
$P(n, r)$

Example 9.2.1. $A = \{a, b, c, d\}$ 3 A

Solution. ...

n n
$P(n, r)$

r n

$P(n,r)$ $\{(x_1, x_2, \ldots, x_r) | x_i \in \{a_1, \ldots, a_n\}$ $x_i \neq x_j$ $i \neq j\}$

$$P(n,r) = \frac{n!}{(n-r)!}$$

Note

$$P(n,0) = 1 \qquad P(n,1) = n \qquad P(n,n) = n!$$

Example 9.2.2. 4 4 $\{a, b, c, d, e\}$

Solution. ...

Example 9.2.3. 6

Solution. ...

Example 9.2.4. 3 (1) (2)

Solution. ...

Example 9.2.5. 7 0 5 6

Solution. ...

Example 9.2.6.

$$P(n,n) = P(n,k) \times P(n-k, n-k)$$

Solution. ...

Note

9.2.6 **combinatorial proof** **double counting**

Example 9.2.7. 20000 70000

Solution. ...

Example 9.2.8. S $\{1,3,5,7\}$

1. $|S|$
2. $\sum_{n \in S} n$

Solution. ...

9.2.2

- ( )
-

Example 9.2.9. 4 $A = \{a, b, c, d\}$ $4! = 24$ ( )

Solution. ...

n

Example 9.2.10. 5 3

1.
2. B_1 G_1
3.

Solution. ...

Example 9.2.11. n

1.
2.

Solution. ...

Example 9.2.12. 9.3.1 3 $A = \{a, b, c, d\}$ $P(4,3) = 24$ ( )

Solution. ...

n r $Q(n,r)$

$$Q(n,r) = \frac{P(n,r)}{r}$$

9.2.3

n k n_1 n_2 ... k n_k $n_1 + n_2 + \cdots + n_k = n$ n

$$P(n; n_1, n_2, \ldots, n_k) =$$

Example 9.2.13. MISSISSIPPI

Solution. ...

9.3

$A = \{a_1, a_2, \ldots, a_n\}$ n $0 \leq r \leq n$ r A (r-combination) r A $C(n, r)$ $\binom{n}{r}$

Example 9.3.1. $A = \{a, b, c, d\}$ 3 A

Solution. ...

$C(n, r)$ r n

$$C(n, r) = \{\{x_1, x_2, \ldots, x_r\} | x_i \in \{a_1, \ldots, a_n\} \quad x_i \neq x_j \quad i \neq j\}$$

$$C(n, r) =$$

Example 9.3.2. 9 1

Solution. ...

Example 9.3.3. MISSISSIPPI (
)

Solution. ...

Example 9.3.4. 7 1
2

Solution. ...

Example 9.3.5. 6 10
10 2

Solution. ...

Example 9.3.6.

$$\binom{n}{r} = \binom{n}{n-r}$$

Solution. ...

Example 9.3.7.

$$\binom{n}{r} = \binom{n-1}{r-1} + \binom{n-1}{r}$$

Solution. ...

9.4

$\binom{n}{r}$ $C(n,r)$ r n

$r<0$ $r>n$

$$\binom{n}{r}=\begin{cases}\frac{n!}{r!(n-r)!} & 0\le r\le n\\ 0 & r>n \quad r<0\end{cases}$$

$\binom{n}{r}$ $(x+y)^n$

$\binom{n}{r}$

(binomial coefficient)

9.4.1

n

$$(x+y)^n=\binom{n}{0}x^n+\binom{n}{1}x^{n-1}y+\cdots+\binom{n}{n-1}xy^{n-1}+\binom{n}{n}y^n$$

!

Example 9.4.1. (easy exercise)

1. x^2y^6 $(2x+y^2)^5$
2. $\binom{n}{0}+\binom{n}{1}+\cdots+\binom{n}{n}$

Solution. ...

9.4.2

Example 9.4.2.

1. $\sum_{r=0}^{n}(-1)^r\binom{n}{r} = 0$
2. $\binom{n}{0} + \binom{n}{2} \cdots + \binom{n}{2k} + \cdots = \binom{n}{1} + \binom{n}{3} \cdots + \binom{n}{2k+1} + \cdots = 2^{n-1}$
3. $\sum_{r=1}^{n} r\binom{n}{r} = n \cdot 2^{n-1}$
4. $***\sum_{i=0}^{r}\binom{m}{i}\binom{n}{r-i} = \binom{m+n}{r}$

Solution. ...

9.4.3

Example 9.4.3. 1. $(0,0)$ $(11,5)$

2. 1 $(4,3)$
3. 1 $(2,3)$ $(3,3)$

Solution. ...

9.5 -

PROGRAMMING PART

9.6 Programming about Combinatorics

Chapter 10

Recurrence Relation

Chapter 11

Recursive Algorithm - an approach to functional programming

Chapter 12

Graph Theory

Index